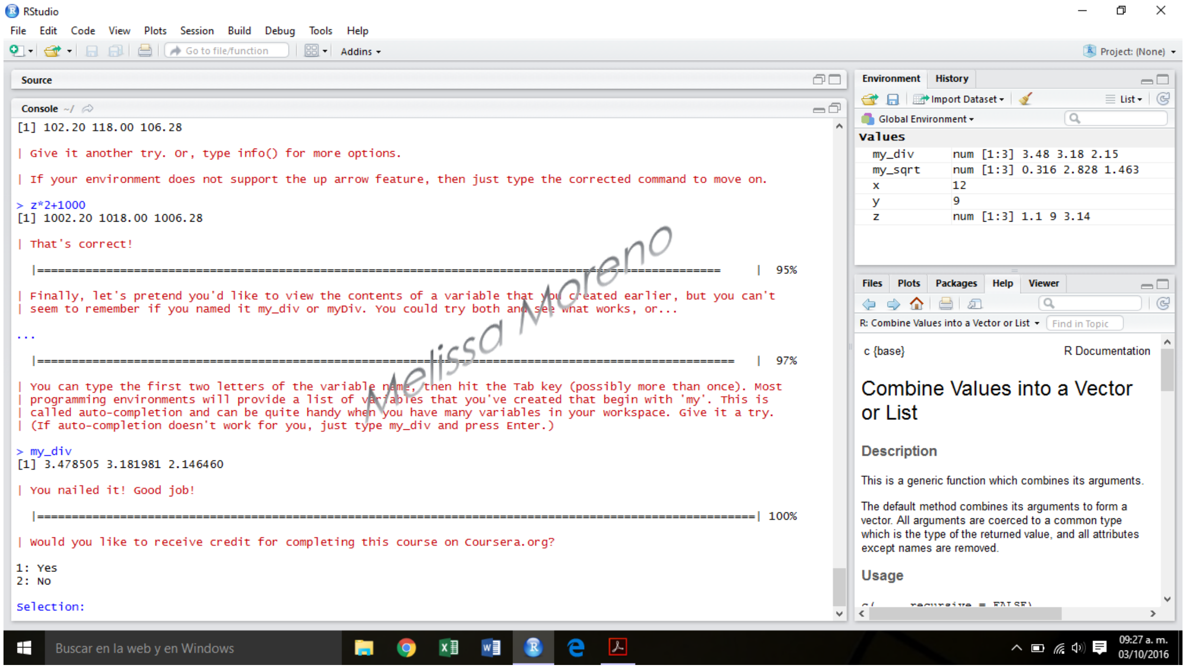
1: Basic Building Blocks


RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Addins
Project: (None)
Source
Console ~/
Environment History
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
R: Combine Values into a Vector or List
Find in Topic
Buscar en la web y en Windows
09:27 a. m.
03/10/2016

```
[1] 102.20 118.00 106.28

| Give it another try. Or, type info() for more options.

| If your environment does not support the up arrow feature, then just type the corrected command to move on.

> z*2+1000
[1] 1002.20 1018.00 1006.28

| That's correct!

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

| Finally, let's pretend you'd like to view the contents of a variable that you created earlier, but you can't
| seem to remember if you named it my_div or myDiv. You could try both and see what works, or...

...

  |==============================================================================================  |  97%

| You can type the first two letters of the variable name, then hit the Tab key (possibly more than once). Most
| programming environments will provide a list of variables that you've created that begin with 'my'. This is
| called auto-completion and can be quite handy when you have many variables in your workspace. Give it a try.
| (If auto-completion doesn't work for you, just type my_div and press Enter.)

> my_div
[1] 3.478505 3.181981 2.146460

| You nailed it! Good job!

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

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

1: Yes
2: No

Selection:
```

```
Values
  my_div      num [1:3] 3.48 3.18 2.15
  my_sqrt     num [1:3] 0.316 2.828 1.463
  x           12
  y           9
  z           num [1:3] 1.1 9 3.14
```

c {base} R Documentation

Combine Values into a Vector or List

Description

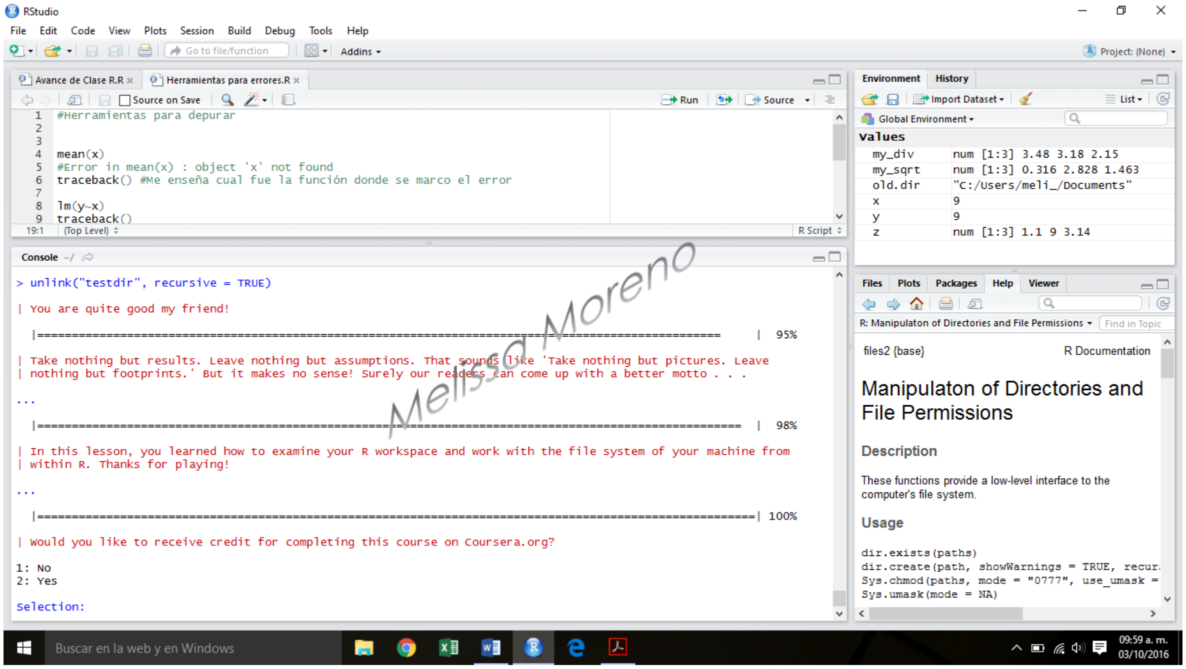
This is a generic function which combines its arguments.

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

Usage

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

2: Workspace and Files



```
#Herramientas para depurar


mean(x)
#Error in mean(x) : object 'x' not found
traceback() #Me enseña cual fue la función donde se marco el error

lm(y~x)
traceback()
```

```
> unlink("testdir", recursive = TRUE)

| You are quite good my friend!

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

| 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 . . .

...

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

| 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%

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

1: No
2: Yes

Selection:
```

```
Values
  my_div      num [1:3] 3.48 3.18 2.15
  my_sqrt     num [1:3] 0.316 2.828 1.463
  old.dir     "C:/Users/meli_/Documents"
  x           9
  y           9
  z           num [1:3] 1.1 9 3.14
```

files2 {base} R Documentation

Manipulaton of Directories and File Permissions

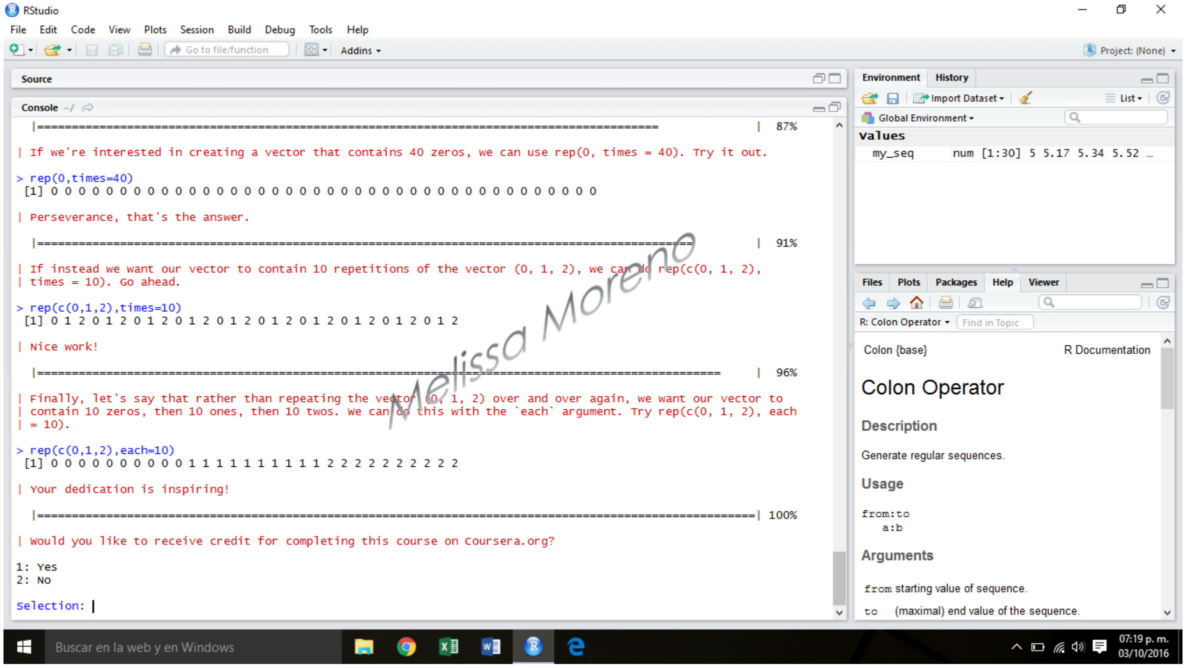
Description

These functions provide a low-level interface to the computer's file system.

Usage

```
dir.exists(paths)
dir.create(path, showWarnings = TRUE, recur
Sys.chmod(paths, mode = "0777", use_umask =
Sys.umask(mode = NA)
```

3: Sequences of Numbers



```
  |==========================================================================================              |  87%

| If we're interested in creating a vector that contains 40 zeros, we can use rep(0, times = 40). Try it out.

> rep(0,times=40)
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

| Perseverance, that's the answer.

  |===============================================================================================         |  91%

| If instead we want our vector to contain 10 repetitions of the vector (0, 1, 2), we can do rep(c(0, 1, 2),
| times = 10). Go ahead.

> rep(c(0,1,2),times=10)
 [1] 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2

| Nice work!

  |====================================================================================================    |  96%

| Finally, let's say that rather than repeating the vector (0, 1, 2) over and over again, we want our vector to
| contain 10 zeros, then 10 ones, then 10 twos. We can do this with the `each` argument. Try rep(c(0, 1, 2), each
| = 10).

> rep(c(0,1,2),each=10)
 [1] 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2

| Your dedication is inspiring!

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

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

1: Yes
2: No

Selection: |
```

```
Values
  my_seq        num [1:30] 5 5.17 5.34 5.52 …
```

Colon {base}   R Documentation

Colon Operator

Description

Generate regular sequences.

Usage

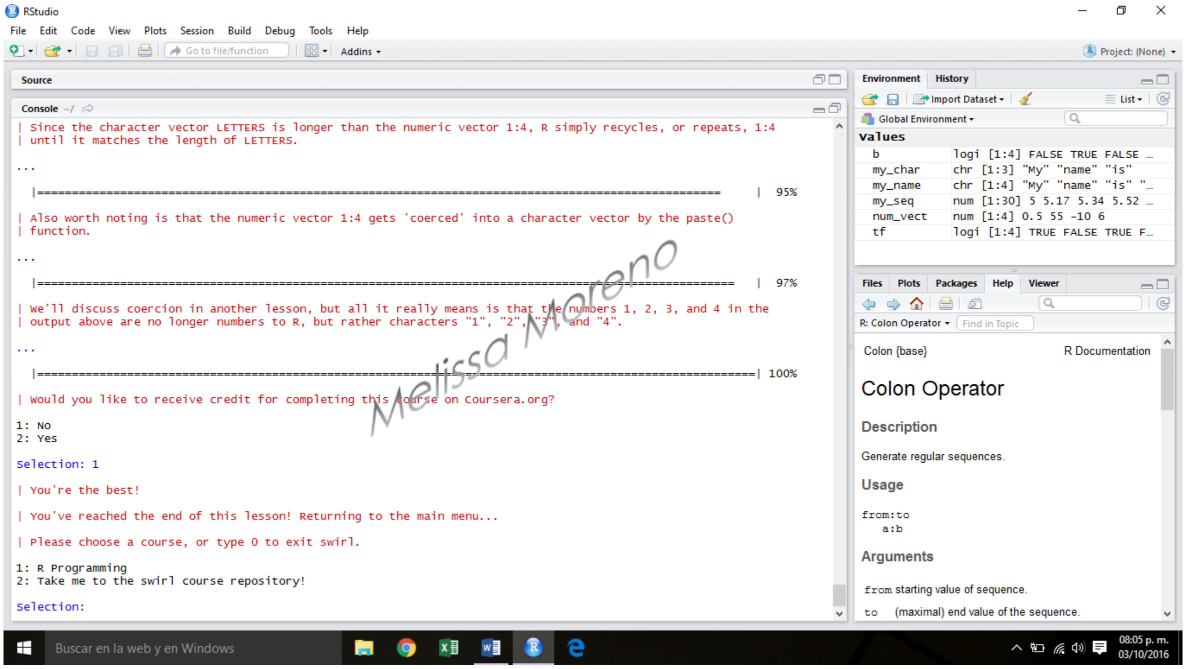
```
from:to
   a:b
```

Arguments

`from` starting value of sequence.

`to` (maximal) end value of the sequence.

4: Vectors



```
| Since the character vector LETTERS is longer than the numeric vector 1:4, R simply recycles, or repeats, 1:4
| until it matches the length of LETTERS.

...

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

| Also worth noting is that the numeric vector 1:4 gets 'coerced' into a character vector by the paste()
| function.

...

  |=====================================================================================================   |  97%

| We'll discuss coercion in another lesson, but all it really means is that the numbers 1, 2, 3, and 4 in the
| output above are no longer numbers to R, but rather characters "1", "2", "3", and "4".

...

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

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

1: No
2: Yes

Selection: 1

| You're the best!

| You've reached the end of this lesson! Returning to the main menu...

| Please choose a course, or type 0 to exit swirl.

1: R Programming
2: Take me to the swirl course repository!

Selection:
```

```
Values
  b             logi [1:4] FALSE TRUE FALSE …
  my_char       chr [1:3] "My" "name" "is"
  my_name       chr [1:4] "My" "name" "is" "…
  my_seq        num [1:30] 5 5.17 5.34 5.52 …
  num_vect      num [1:4] 0.5 55 -10 6
  tf            logi [1:4] TRUE FALSE TRUE F…
```

Colon {base}   R Documentation

Colon Operator

Description

Generate regular sequences.

Usage

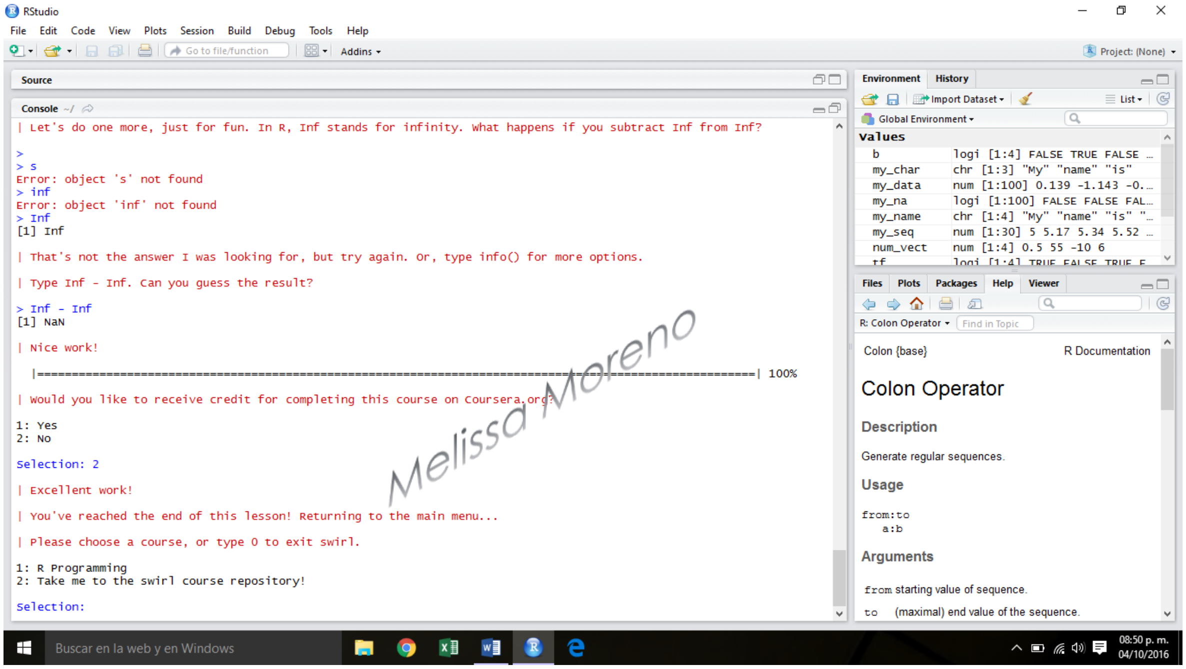
```
from:to
   a:b
```

Arguments

`from` starting value of sequence.

`to` (maximal) end value of the sequence.

5: Missing Values



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

>
> s
Error: object 's' not found
> inf
Error: object 'inf' not found
> Inf
[1] Inf

| That's not the answer I was looking for, but try again. Or, type info() for more options.

| Type Inf - Inf. Can you guess the result?

> Inf - Inf
[1] NaN

| Nice work!

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

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

1: Yes
2: No

Selection: 2

| Excellent work!

| You've reached the end of this lesson! Returning to the main menu...

| Please choose a course, or type 0 to exit swirl.

1: R Programming
2: Take me to the swirl course repository!

Selection:
```

```
Values
  b          logi [1:4] FALSE TRUE FALSE ...
  my_char    chr [1:3] "My" "name" "is"
  my_data    num [1:100] 0.139 -1.143 -0....
  my_na      logi [1:100] FALSE FALSE FAL...
  my_name    chr [1:4] "My" "name" "is" "...
  my_seq     num [1:30] 5 5.17 5.34 5.52 ...
  num_vect   num [1:4] 0.5 55 -10 6
  tf         logi [1:4] TRUE FALSE TRUE F...
```

Colon {base} R Documentation

Colon Operator

Description

Generate regular sequences.

Usage

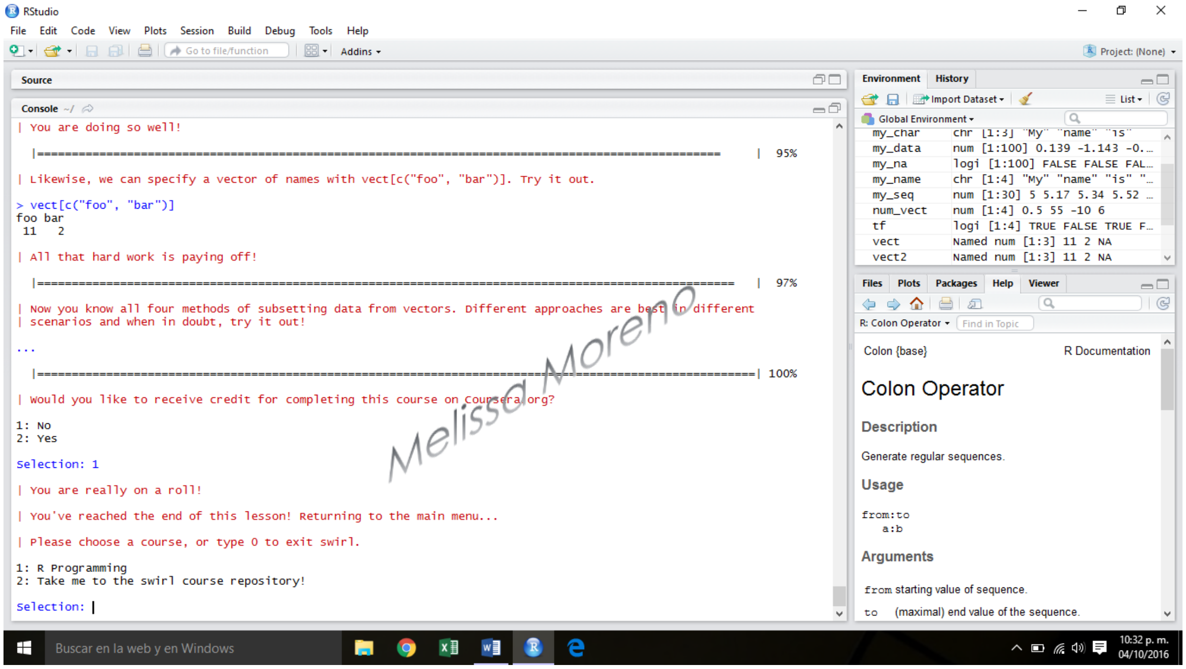
```
from:to
   a:b
```

Arguments

`from` starting value of sequence.

`to` (maximal) end value of the sequence.

6: Subsetting Vectors



```
| You are doing so well!

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

| Likewise, we can specify a vector of names with vect[c("foo", "bar")]. Try it out.

> vect[c("foo", "bar")]
foo bar
 11   2

| All that hard work is paying off!

  |=========================================================================================================   |  97%

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

...

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

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

1: No
2: Yes

Selection: 1

| You are really on a roll!

| You've reached the end of this lesson! Returning to the main menu...

| Please choose a course, or type 0 to exit swirl.

1: R Programming
2: Take me to the swirl course repository!

Selection: |
```

```
  my_char    chr [1:3] "My" "name" "is"
  my_data    num [1:100] 0.139 -1.143 -0....
  my_na      logi [1:100] FALSE FALSE FAL...
  my_name    chr [1:4] "My" "name" "is" "...
  my_seq     num [1:30] 5 5.17 5.34 5.52 ...
  num_vect   num [1:4] 0.5 55 -10 6
  tf         logi [1:4] TRUE FALSE TRUE F...
  vect       Named num [1:3] 11 2 NA
  vect2      Named num [1:3] 11 2 NA
```

Colon {base} R Documentation

Colon Operator

Description

Generate regular sequences.

Usage

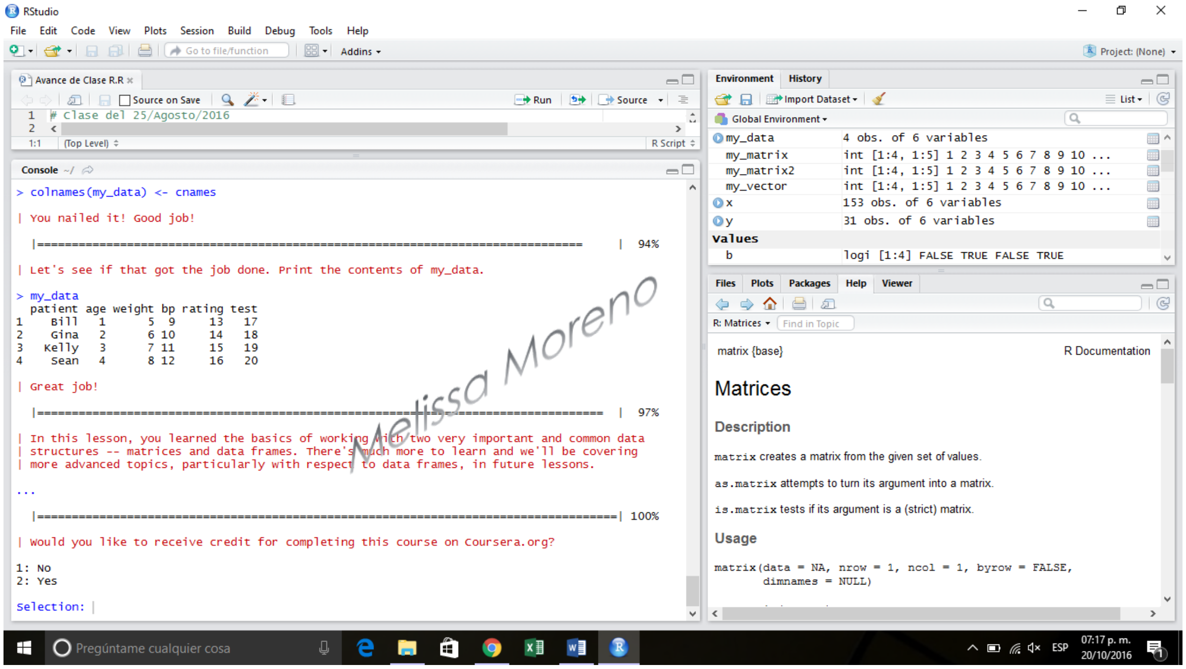
```
from:to
   a:b
```

Arguments

`from` starting value of sequence.

`to` (maximal) end value of the sequence.

7: Matrices and Data Frames



```
1  # Clase del 25/Agosto/2016
2
```

```
> colnames(my_data) <- cnames

| You nailed it! Good job!

  |=============================================================================     |  94%

| Let's see if that got the job done. Print the contents of my_data.

> 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

| Great job!

  |================================================================================  |  97%

| 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 be covering
| more advanced topics, particularly with respect to data frames, in future lessons.

...

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

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

1: No
2: Yes

Selection: |
```

```
my_data        4 obs. of 6 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 ...
x              153 obs. of 6 variables
y              31 obs. of 6 variables
Values
b              logi [1:4] FALSE TRUE FALSE TRUE
```

matrix {base} R Documentation

Matrices

Description

`matrix` creates a matrix from the given set of values.

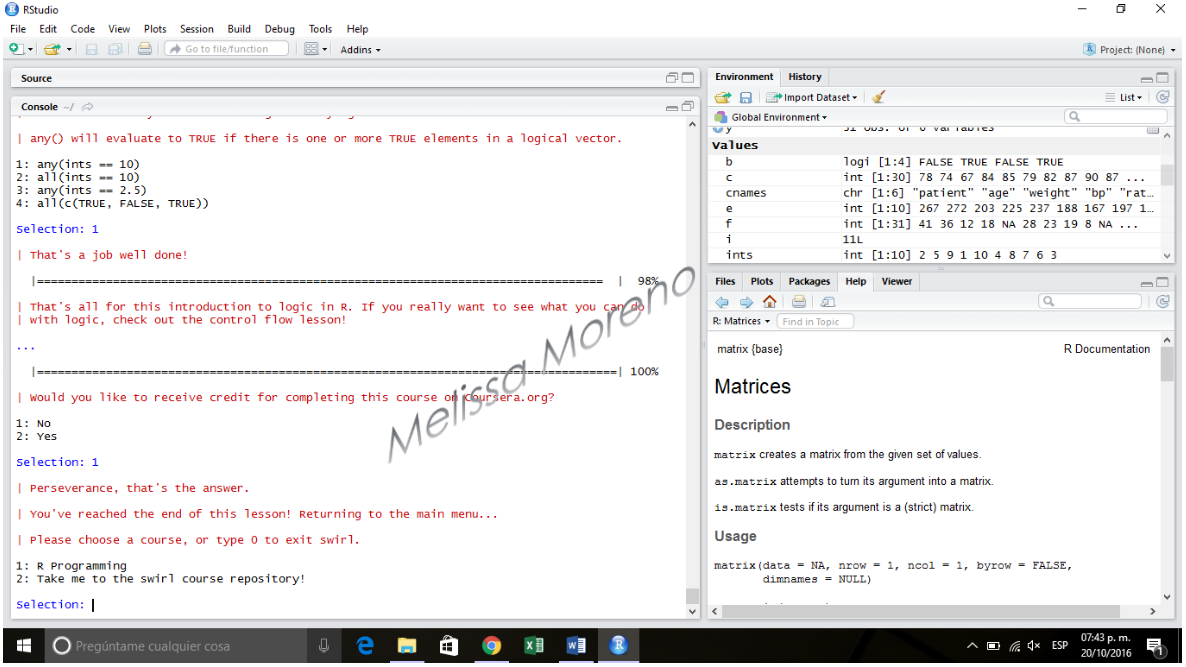
`as.matrix` attempts to turn its argument into a matrix.

`is.matrix` tests if its argument is a (strict) matrix.

Usage

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,
       dimnames = NULL)
```

8: Logic



```
| any() will evaluate to TRUE if there is one or more TRUE elements in a logical vector.

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

Selection: 1

| That's a job well done!

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

| 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!

...

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

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

1: No
2: Yes

Selection: 1

| Perseverance, that's the answer.

| You've reached the end of this lesson! Returning to the main menu...

| Please choose a course, or type 0 to exit swirl.

1: R Programming
2: Take me to the swirl course repository!

Selection: |
```

```
Values
b              logi [1:4] FALSE TRUE FALSE TRUE
c              int [1:30] 78 74 67 84 85 79 82 87 90 87 ...
cnames         chr [1:6] "patient" "age" "weight" "bp" "rat…
e              int [1:10] 267 272 203 225 237 188 167 197 1…
f              int [1:31] 41 36 12 18 NA 28 23 19 8 NA ...
i              11L
ints           int [1:10] 2 5 9 1 10 4 8 7 6 3
```

matrix {base} R Documentation

Matrices

Description

`matrix` creates a matrix from the given set of values.

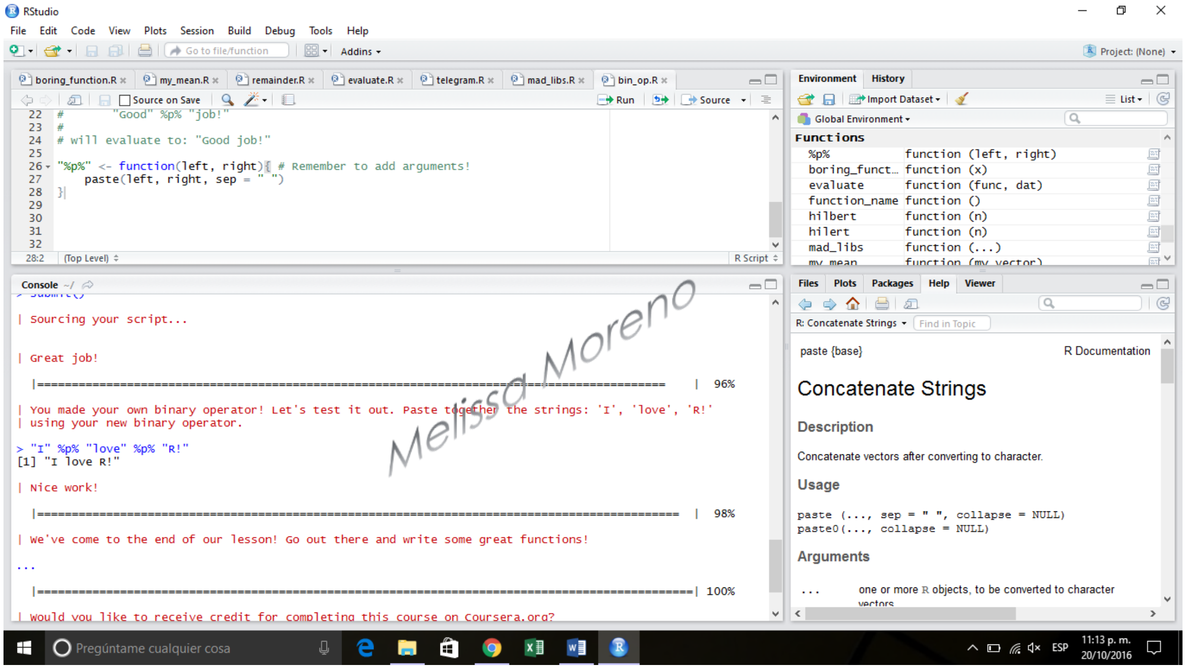
`as.matrix` attempts to turn its argument into a matrix.

`is.matrix` tests if its argument is a (strict) matrix.

Usage

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,
       dimnames = NULL)
```

9: Functions



```r
#        "Good" %p% "job!"
#
# will evaluate to: "Good job!"

"%p%" <- function(left, right){ # Remember to add arguments!
    paste(left, right, sep = " ")
}
```

```
> submit()

| Sourcing your script...


| Great job!

  |======================================================================   |  96%

| You made your own binary operator! Let's test it out. Paste together the strings: 'I', 'love', 'R!'
| using your new binary operator.

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

| Nice work!

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

| We've come to the end of our lesson! Go out there and write some great functions!

...

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

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

```
Functions
  %p%            function (left, right)
  boring_funct…  function (x)
  evaluate       function (func, dat)
  function_name  function ()
  hilbert        function (n)
  hilert         function (n)
  mad_libs       function (...)
  my_mean        function (my_vector)
```

paste {base} R Documentation

Concatenate Strings

Description

Concatenate vectors after converting to character.

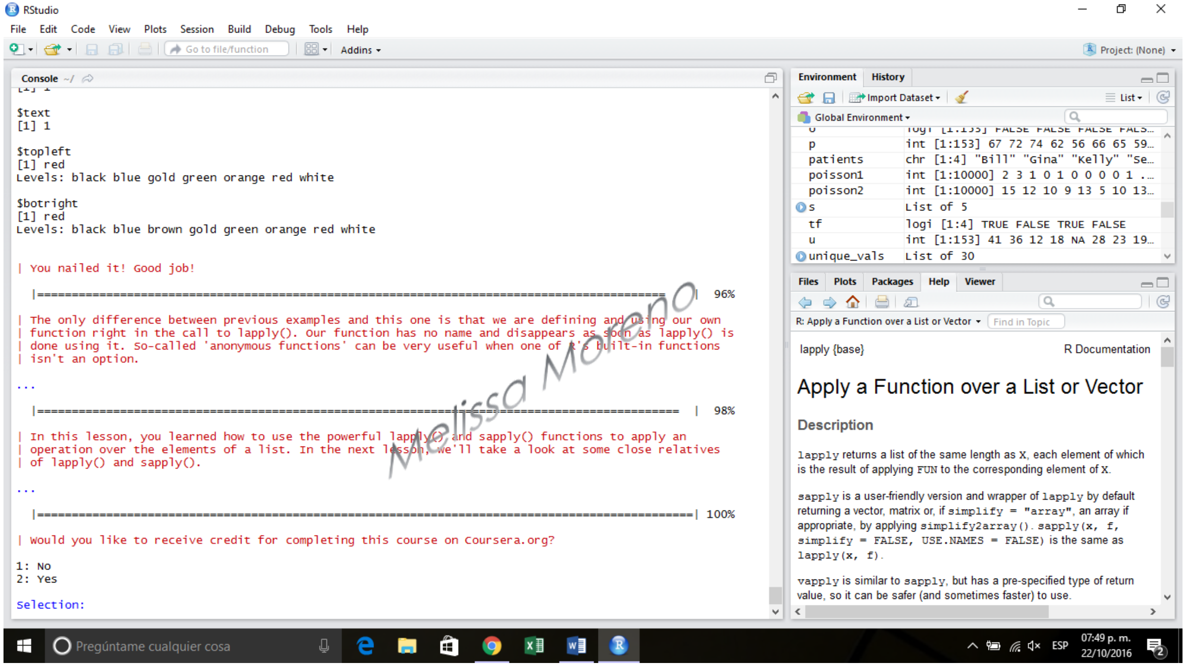
Usage

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

Arguments

... one or more R objects, to be converted to character vectors

10: lapply and sapply



```
[1] 1

$text
[1] 1

$topleft
[1] red
Levels: black blue gold green orange red white

$botright
[1] red
Levels: black blue brown gold green orange red white


| You nailed it! Good job!

  |======================================================================   |  96%

| The only difference between previous examples and this one is that we are defining and using our own
| function right in the call to lapply(). Our function has no name and disappears as soon as lapply() is
| done using it. So-called 'anonymous functions' can be very useful when one of R's built-in functions
| isn't an option.

...

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

| In this lesson, you learned how to use the powerful lapply() and sapply() functions to apply an
| operation over the elements of a list. In the next lesson, we'll take a look at some close relatives
| of lapply() and sapply().

...

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

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

1: No
2: Yes

Selection:
```

```
  o            logi [1:153] FALSE FALSE FALSE FALS…
  p            int [1:153] 67 72 74 62 56 66 65 59…
  patients     chr [1:4] "Bill" "Gina" "Kelly" "Se…
  poisson1     int [1:10000] 2 3 1 0 1 0 0 0 0 1 .…
  poisson2     int [1:10000] 15 12 10 9 13 5 10 13…
  s            List of 5
  tf           logi [1:4] TRUE FALSE TRUE FALSE
  u            int [1:153] 41 36 12 18 NA 28 23 19…
  unique_vals  List of 30
```

lapply {base} R Documentation

Apply a Function over a List or Vector

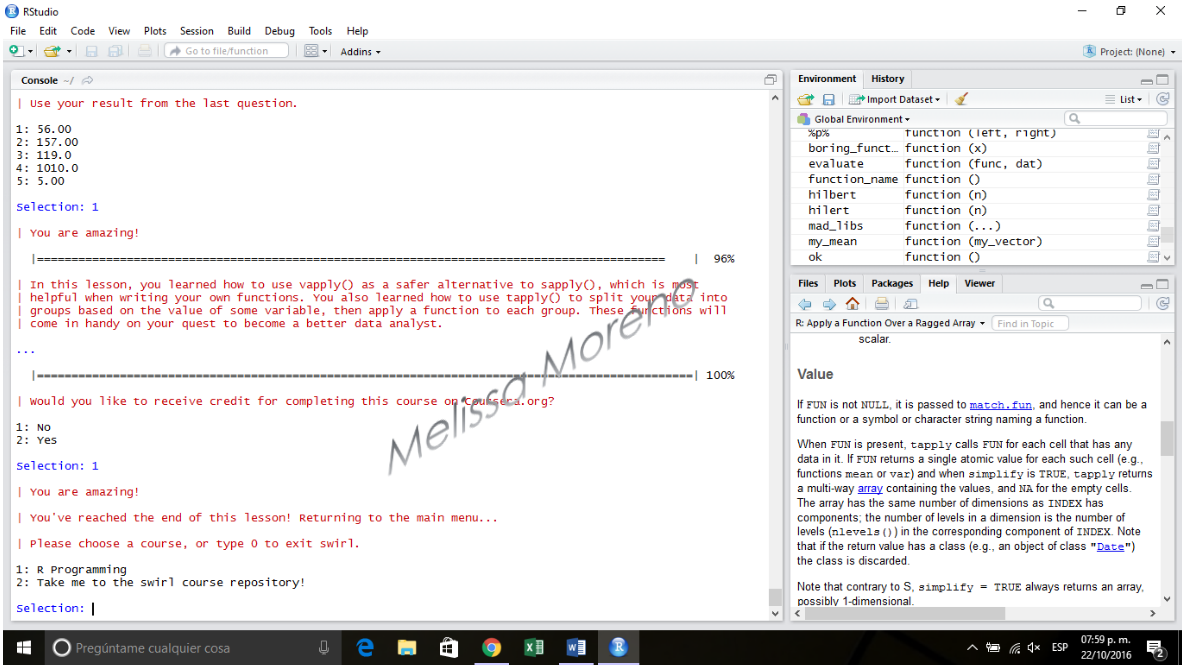
Description

`lapply` returns a list of the same length as `X`, each element of which is the result of applying `FUN` to the corresponding element of `X`.

`sapply` is a user-friendly version and wrapper of `lapply` by default returning a vector, matrix or, if `simplify = "array"`, an array if appropriate, by applying `simplify2array()`. `sapply(x, f, simplify = FALSE, USE.NAMES = FALSE)` is the same as `lapply(x, f)`.

`vapply` is similar to `sapply`, but has a pre-specified type of return value, so it can be safer (and sometimes faster) to use.

11: vapply and tapply


RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Addins
Project: (None)
Console ~/
Environment History
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
R: Apply a Function Over a Ragged Array
Find in Topic
Pregúntame cualquier cosa
ESP
07:59 p. m.
22/10/2016

```
| Use your result from the last question.

1: 56.00
2: 157.00
3: 119.0
4: 1010.0
5: 5.00

Selection: 1

| You are amazing!

  |==========================================================================================    |  96%

| In this lesson, you learned how to use vapply() as a safer alternative to sapply(), which is most
| helpful when writing your own functions. You also learned how to use tapply() to split your data into
| groups based on the value of some variable, then apply a function to each group. These functions will
| come in handy on your quest to become a better data analyst.

...

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

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

1: No
2: Yes

Selection: 1

| You are amazing!

| You've reached the end of this lesson! Returning to the main menu...

| Please choose a course, or type 0 to exit swirl.

1: R Programming
2: Take me to the swirl course repository!

Selection: |
```

```
%p%            function (left, right)
boring_funct…  function (x)
evaluate       function (func, dat)
function_name  function ()
hilbert        function (n)
hilert         function (n)
mad_libs       function (...)
my_mean        function (my_vector)
ok             function ()
```

scalar.

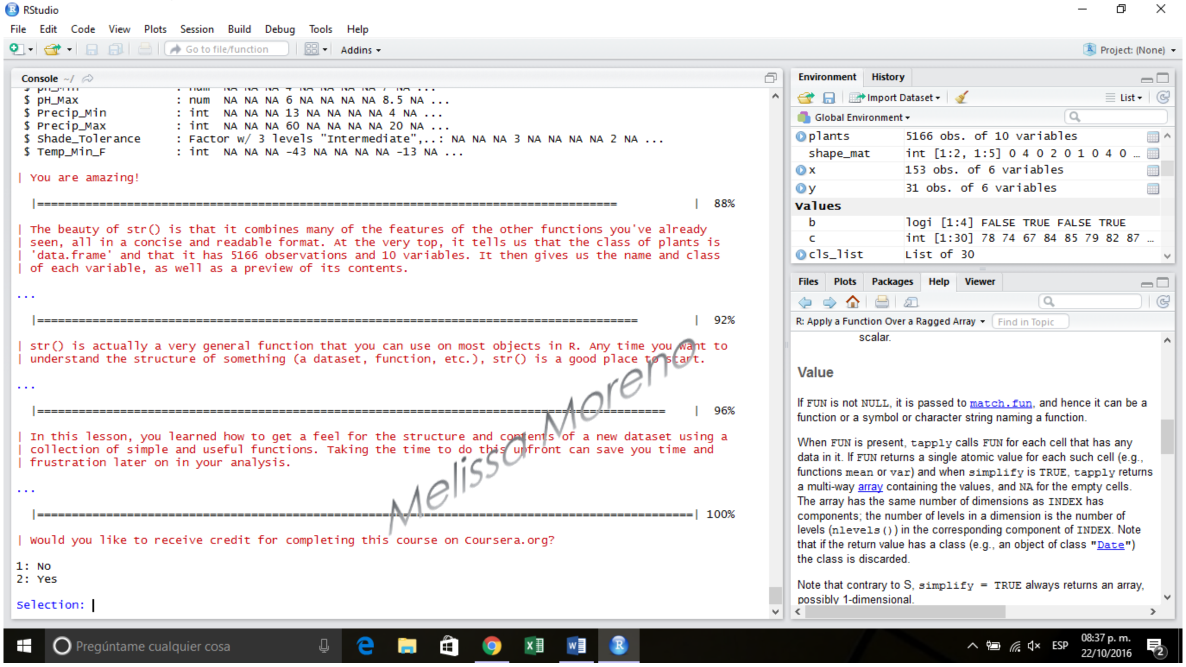
Value

If `FUN` is not `NULL`, it is passed to `match.fun`, and hence it can be a function or a symbol or character string naming a function.

When `FUN` is present, `tapply` calls `FUN` for each cell that has any data in it. If `FUN` returns a single atomic value for each such cell (e.g., functions `mean` or `var`) and when `simplify` is `TRUE`, `tapply` returns a multi-way array containing the values, and `NA` for the empty cells. The array has the same number of dimensions as `INDEX` has components; the number of levels in a dimension is the number of levels (`nlevels()`) in the corresponding component of `INDEX`. Note that if the return value has a class (e.g., an object of class "`Date`") the class is discarded.

Note that contrary to S, `simplify = TRUE` always returns an array, possibly 1-dimensional.

12: Looking at Data


RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Addins
Project: (None)
Console ~/
Environment History
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
R: Apply a Function Over a Ragged Array
Find in Topic
Pregúntame cualquier cosa
ESP
08:37 p. m.
22/10/2016

```
 $ pH_Max               : num  NA NA NA 6 NA NA NA NA 8.5 NA ...
 $ Precip_Min           : int  NA NA NA 13 NA NA NA NA 4 NA ...
 $ Precip_Max           : int  NA NA NA 60 NA NA NA NA 20 NA ...
 $ Shade_Tolerance      : Factor w/ 3 levels "Intermediate",..: NA NA NA 3 NA NA NA NA 2 NA ...
 $ Temp_Min_F           : int  NA NA NA -43 NA NA NA NA -13 NA ...

| You are amazing!

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

| The beauty of str() is that it combines many of the features of the other functions you've already
| seen, all in a concise and readable format. At the very top, it tells us that the class of plants is
| 'data.frame' and that it has 5166 observations and 10 variables. It then gives us the name and class
| of each variable, as well as a preview of its contents.

...

  |=====================================================================================        |  92%

| str() is actually a very general function that you can use on most objects in R. Any time you want to
| understand the structure of something (a dataset, function, etc.), str() is a good place to start.

...

  |=========================================================================================    |  96%

| In this lesson, you learned how to get a feel for the structure and contents of a new dataset using a
| collection of simple and useful functions. Taking the time to do this upfront can save you time and
| frustration later on in your analysis.

...

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

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

1: No
2: Yes

Selection: |
```

```
plants      5166 obs. of 10 variables
shape_mat   int [1:2, 1:5] 0 4 0 2 0 1 0 4 0 …
x           153 obs. of 6 variables
y           31 obs. of 6 variables
Values
b           logi [1:4] FALSE TRUE FALSE TRUE
c           int [1:30] 78 74 67 84 85 79 82 87 …
cls_list    List of 30
```

scalar.

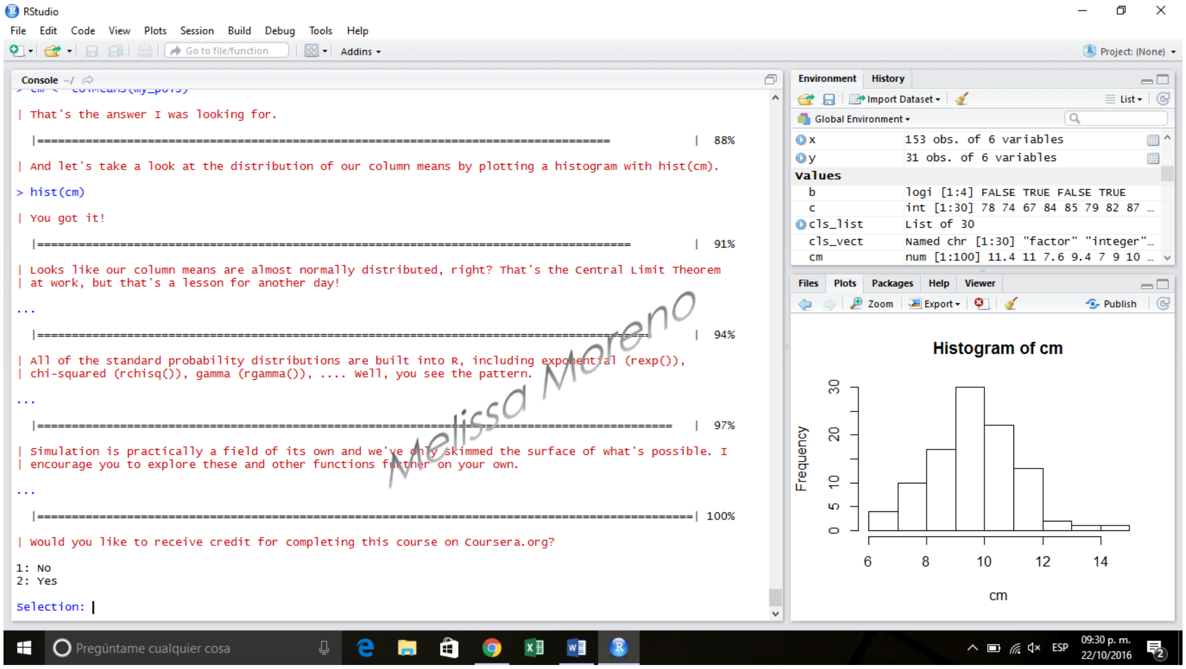
Value

If `FUN` is not `NULL`, it is passed to `match.fun`, and hence it can be a function or a symbol or character string naming a function.

When `FUN` is present, `tapply` calls `FUN` for each cell that has any data in it. If `FUN` returns a single atomic value for each such cell (e.g., functions `mean` or `var`) and when `simplify` is `TRUE`, `tapply` returns a multi-way array containing the values, and `NA` for the empty cells. The array has the same number of dimensions as `INDEX` has components; the number of levels in a dimension is the number of levels (`nlevels()`) in the corresponding component of `INDEX`. Note that if the return value has a class (e.g., an object of class "`Date`") the class is discarded.

Note that contrary to S, `simplify = TRUE` always returns an array, possibly 1-dimensional.

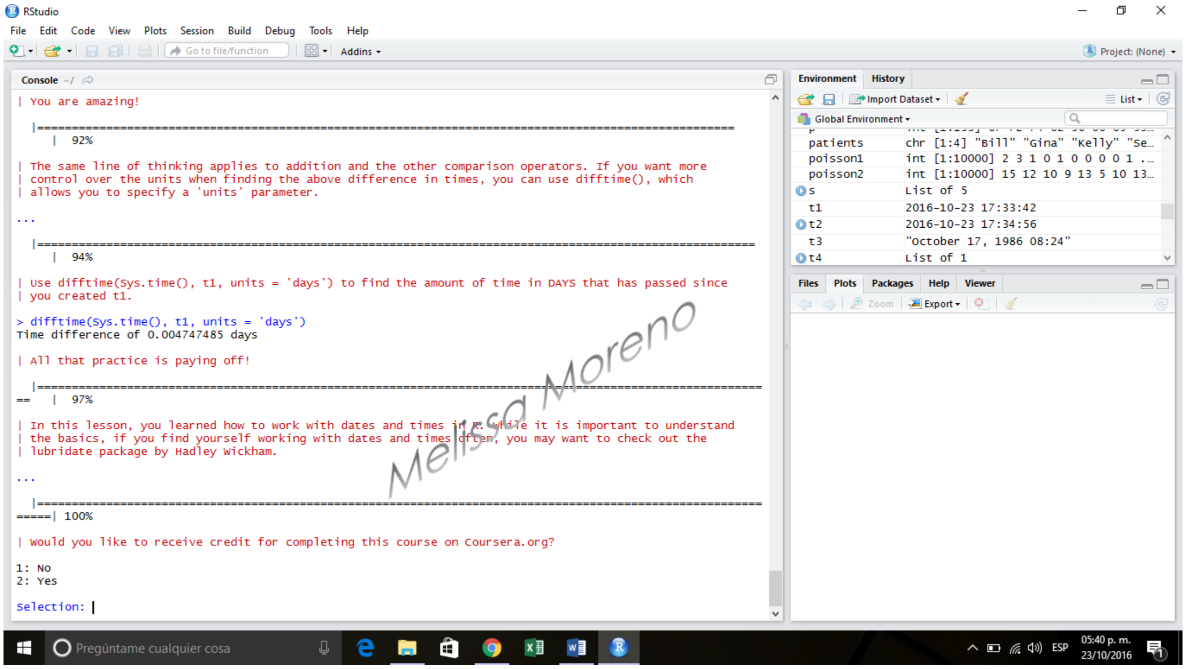
13: Simulation



```
> cm <- colMeans(my_pois)

| That's the answer I was looking for.

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

| And let's take a look at the distribution of our column means by plotting a histogram with hist(cm).

> hist(cm)

| You got it!

  |========================================================================         |  91%

| Looks like our column means are almost normally distributed, right? That's the Central Limit Theorem
| at work, but that's a lesson for another day!

...

  |===========================================================================      |  94%

| All of the standard probability distributions are built into R, including exponential (rexp()),
| chi-squared (rchisq()), gamma (rgamma()), .... Well, you see the pattern.

...

  |=============================================================================    |  97%

| Simulation is practically a field of its own and we've only skimmed the surface of what's possible. I
| encourage you to explore these and other functions further on your own.

...

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

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

1: No
2: Yes

Selection: |
```

```
x            153 obs. of 6 variables
y            31 obs. of 6 variables
Values
  b          logi [1:4] FALSE TRUE FALSE TRUE
  c          int [1:30] 78 74 67 84 85 79 82 87 …
cls_list     List of 30
  cls_vect   Named chr [1:30] "factor" "integer"…
  cm         num [1:100] 11.4 11 7.6 9.4 7 9 10 …
```

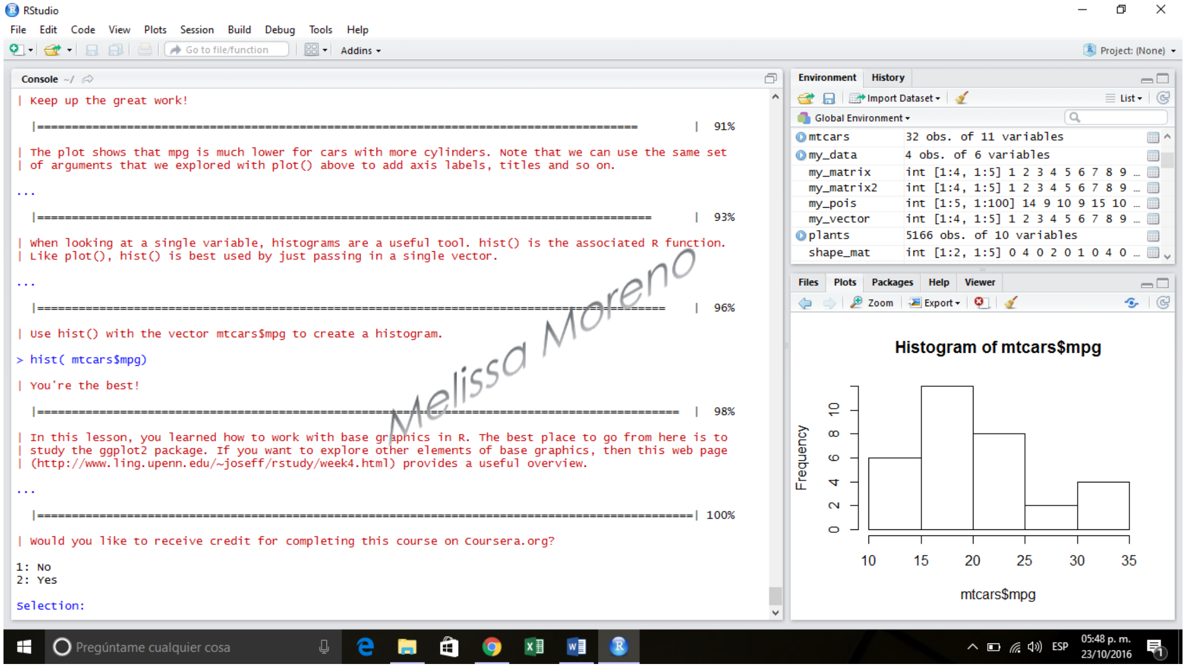
14: Dates and Times



```
| You are amazing!

  |===================================================================================================
    |  92%

| The same line of thinking applies to addition and the other comparison operators. If you want more
| control over the units when finding the above difference in times, you can use difftime(), which
| allows you to specify a 'units' parameter.

...

  |======================================================================================================
    |  94%

| Use difftime(Sys.time(), t1, units = 'days') to find the amount of time in DAYS that has passed since
| you created t1.

> difftime(Sys.time(), t1, units = 'days')
Time difference of 0.004747485 days

| All that practice is paying off!

  |=======================================================================================================
==    |  97%

| In this lesson, you learned how to work with dates and times in R. While it is important to understand
| the basics, if you find yourself working with dates and times often, you may want to check out the
| lubridate package by Hadley Wickham.

...

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

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

1: No
2: Yes

Selection: |
```

```
  patients   chr [1:4] "Bill" "Gina" "Kelly" "Se…
  poisson1   int [1:10000] 2 3 1 0 1 0 0 0 0 1 .…
  poisson2   int [1:10000] 15 12 10 9 13 5 10 13…
s            List of 5
  t1         2016-10-23 17:33:42
t2           2016-10-23 17:34:56
  t3         "October 17, 1986 08:24"
t4           List of 1
```

15: Base Graphics



```
| Keep up the great work!

  |==========================================================================        |  91%

| The plot shows that mpg is much lower for cars with more cylinders. Note that we can use the same set
| of arguments that we explored with plot() above to add axis labels, titles and so on.

...

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

| When looking at a single variable, histograms are a useful tool. hist() is the associated R function.
| Like plot(), hist() is best used by just passing in a single vector.

...

  |==============================================================================    |  96%

| Use hist() with the vector mtcars$mpg to create a histogram.

> hist( mtcars$mpg)

| You're the best!

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

| In this lesson, you learned how to work with base graphics in R. The best place to go from here is to
| study the ggplot2 package. If you want to explore other elements of base graphics, then this web page
| (http://www.ling.upenn.edu/~joseff/rstudy/week4.html) provides a useful overview.

...

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

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

1: No
2: Yes

Selection:
```

```
mtcars      32 obs. of 11 variables
my_data     4 obs. of 6 variables
my_matrix   int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 ...
my_matrix2  int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 ...
my_pois     int [1:5, 1:100] 14 9 10 9 15 10 ...
my_vector   int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 ...
plants      5166 obs. of 10 variables
shape_mat   int [1:2, 1:5] 0 4 0 2 0 1 0 4 0 ...
```