

```
Console ~/
| You are amazing!

===== | 95%

| Finally, let's pretend you'd like to view the contents of a variable that you created e
arlier, 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 y
our 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 .146460

| Perseverance, that's the answer.

===== | 100%
```

```
Console ~/
| Use unlink("testdir", recursive = TRUE).

> unlink("testdir", recursive = TRUE)

| You got it!

===== | 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: Yes
2: No

Selection: |
```

```
Console ~/ |
| 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 0 1 2
| You are doing so well!
|=====| 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 2 2
| You are amazing!
|=====| 100%
| Would you like to receive credit for completing this course on Coursera.org?
1: Yes
2: No
Selection: |
```

```
Console ~/ |
| 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: Yes
2: No
Selection: |
```

```
Console ~/1 ↵

| You nailed it! Good job!

|=====| 90%

| Now that we've got NAs down pat, let's look at a second type of missing value -- NaN, which stands for 'not a
| number'. To generate NaN, try dividing (using a forward slash) 0 by 0 now..

> 0/0
[1] NaN

| Your dedication is inspiring!

|=====| 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

| Keep up the great work!

|=====| 100%

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

1: No
2: Yes

Selection:
```

```
Console ~/1 ↵

bar
2

| Great job!

|=====| 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

| You are amazing!

|=====| 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: Yes
2: No

Selection: |
```

```
Console ~/ |
=====
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

| You are really on a roll!

=====
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: Yes
2: No

Selection: 2
```

```
Console ~/ |
=====
96%

| Which of the following evaluates to TRUE?

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

Selection: 4

| Your dedication is inspiring!

=====
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: Yes
2: No

Selection: |
```

```
Console ~/  
| Sourcing your script...

| All that hard work is paying off!

|=====
| 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!"

| Excellent 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?

1: No
2: Yes

Selection: |
```

```
Console ~/  
| Excellent 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: |
```

```
Console ~/  
1: 119.0
2: 1010.0
3: 5.00
4: 157.00
5: 56.00

Selection: 5

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

=====
| 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 lapply() to split your data into groups based on the value of some variable, and 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: Yes
2: No

Selection: |
```

```
Console ~/  
review 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: Yes
2: No

Selection: |
```

```
Console ~/ 
m 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: |
```

```
Console ~/
...
===== | 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 0.005582975 days

| Great job!

===== | 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: Yes
2: No

Selection: |
```

```
Console ~/GitHub/Programacion_Actuarial_3/
| 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)

| That's a job well done!

===== | 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: Yes
2: No

Selection: |
```