

> my_div

[1] 3.478505 3.181981 2.146460

| You got it right!

| ====== | 100%

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

1: Yes

2: No

Selection: 2

| Your dedication is inspiring!

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

| After you finish this lesson delete the 'testdir' directory that you just left (and everything in 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 . . .

...

| ===== | 97%

| 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?

3: 1010.0
4: 119.0
5: 5.00

Selection: 2

| That's the answer I was looking for.

| ====== | 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?

| 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.

...

| 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()`.

...

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

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

| You are really on a roll!

| ====== | 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: Yes

2: No

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

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

Selection: 4

| 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?

```
> colnames(my_data) <- cnames
```

| Excellent work!

| ===== | 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
```

| All that practice is paying off!

| ===== | 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.

...

| All that hard work is paying off!

| 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 got it!

| 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?

| 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

| That's the answer I was looking for.

| 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

| All that practice is paying off!

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

| 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.

...

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

...

| 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".

...

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

```
> 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
```

| That's correct!

| ===== | 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 1 1 1 1 1 1 1 1 1 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?

...

|=====| 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?

| 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?

| ====== | 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 0.009526523 days
```

| You are quite good my friend!

| ====== | 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?

...

| ====== | 96%

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

> `hist(mtcars$mpg)`

| You are doing so well!

| ====== | 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?