```
| All that practice is paying off!

  |=======================================================   |  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...

...my_div

  |========================================================  |  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

| Perseverance, that's the answer.

  |========================================================| 100%
| Would you like to receive credit for completing this course on
| Coursera.org?

1: Yes
2: No

Selection: 1
what is your email address? sowenreign@gmail.com
what is your assignment token?
Grade submission failed.
Press ESC if you want to exit this lesson and you
want to try to submit your grade at a later time.

| Try again. Getting it right on the first try is boring anyway!

|

1: Yes
2: No

Selection: 0

| 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: Owen Santiago
```

```
> setwd()
Error in setwd() : argument "dir" is missing, with no default
> setwd("testdir")
Error in setwd("testdir") : cannot change working directory
> skip()

| Entering the following correct answer for you...

> setwd(old.dir)

| You are amazing!

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

...

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

1: No
2: Yes

Selection: 1

| Nice 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: Owen Santiago
```

```
| If R has a built-in function for a particular task, it's likely that function is highly optimized for that purpose and
| is your best option. As you become a more advanced R programmer, you'll design your own functions to perform tasks
| when there are no better options. We'll explore writing your own functions in future lessons.

...

  |=========================================================================================                |  83%
| One more function related to creating sequences of numbers is rep(), which stands for 'replicate'. Let's look at a few
| uses.

...

  |=============================================================================================            |  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 0 0 0 0

| You got it right!

  |=============================================================================================            |  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

| Perseverance, that's the answer.

  |=================================================================================================        |  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

| That's correct!

  |=================================================================================================================| 100%
| Would you like to receive credit for completing this course on Coursera.org?

1: Yes
2: No

Selection: 2

| 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: Owen Santiago
```

```
> paste(1:3, c("X", "Y", "Z"), sep = "")
[1] "1X" "2Y" "3Z"

| You got it right!

  |=========================================================================================================               |  87%
| What do you think will happen if our vectors are of different length? (Hint: we talked about this in a previous
| lesson.)

...

  |=============================================================================================================           |  89%
| Vector recycling! Try paste(LETTERS, 1:4, sep = "-"), where LETTERS is a predefined variable in R containing a
| character vector of all 26 letters in the English alphabet.

> paste(LETTERS, 1:4, sep = "-")
 [1] "A-1" "B-2" "C-3" "D-4" "E-1" "F-2" "G-3" "H-4" "I-1" "J-2" "K-3" "L-4" "M-1" "N-2" "O-3" "P-4" "Q-1" "R-2" "S-3"
[20] "T-4" "U-1" "V-2" "W-3" "X-4" "Y-1" "Z-2"

| Excellent job!

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

| Great job!

| 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: Owen Santiago
```

```
 [33]           NA -0.392081141           NA           NA           NA -1.025759332 -0.602723582           NA
 [41]           NA  1.719903415 -0.755560775  0.309530271  1.849773663           NA  0.638320226 -2.225879090
 [49] -1.118641588           NA           NA -0.483211853           NA -1.437815563           NA           NA
 [57]           NA  0.028421042           NA -0.521892743  0.026666970           NA           NA  0.262084295
 [65] -2.214255965           NA  0.287716804           NA           NA           NA           NA           NA
 [73]           NA           NA -0.092139596  0.006693318  0.071054363  1.384827186           NA           NA
 [81]           NA           NA           NA           NA           NA  0.160420428  0.896150925  1.070398288
 [89]  2.192292876           NA           NA           NA  0.591881386           NA -0.654265561 -1.448153969
 [97]           NA           NA           NA           NA

| You are quite good my friend!

  |=========================================================================================================            |  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

| You are amazing!

  |=================================================================================================================    |  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

| You almost had it, but not quite. Try again. Or, type info() for more options.

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

> Inf - Inf
[1] NaN

| You got it right!

  |=================================================================================================================| 100%
| Would you like to receive credit for completing this course on Coursera.org?

1: Yes
2: No

Selection: 2

| Great job!

| 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: Owen Santiago
```

```
| You are quite good my friend!

  |=========================================================================================================          |  92%
| Now, try it out.

>
> ====
Error: unexpected '==' in "=="
> ==
Error: unexpected '==' in "=="
> skip()

| Entering the following correct answer for you...

> vect["bar"]
bar
  2

| Excellent 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 doing so well!

  |=========================================================================================================          |  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: 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: Owen Santiago
```

```
> cnames <- c("patient", "age", "weight", "bp", "rating", "test")

| Excellent job!

  |=========================================================================================================          |  92%
| Now, use the colnames() function to set the `colnames` attribute for our data frame. This is similar to the way we
| used the dim() function earlier in this lesson.

> colnames(cnames )
NULL

| Try again. Getting it right on the first try is boring anyway! Or, type info() for more options.

| Try colnames(my_data) <- cnames.

> 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

| Your dedication is inspiring!

  |=================================================================================================================  |  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: 1

| That's correct!

| 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: Owen Santiago
```

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

Selection: 3

| You almost had it, but not quite. Try again.

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

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

Selection: 4

| That's correct!

  |=========================================================================================================  |  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: 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: Owen Santiago
```