```
    |
|    0%

| In this lesson, we'll cover matrices and data frames. Both represent 'recta
ngular' data types,
| meaning that they are used to store tabular data, with rows and columns.

...

    |===
|    3%
| The main difference, as you'll see, is that matrices can only contain a sin
gle class of data, while
| data frames can consist of many different classes of data.

...

    |=====
|    6%
| Let's create a vector containing the numbers 1 through 20 using the `:` ope
rator. Store the result
| in a variable called my_vector.

> my_vector<-1:20

| Keep up the great work!

    |========
|    8%
| View the contents of the vector you just created.

> my_vector
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20

| You are doing so well!

    |==========
|   11%
| The dim() function tells us the 'dimensions' of an object. What happens if
we do dim(my_vector)?
| Give it a try.

> dim(my_vector)
NULL

| You are doing so well!

    |=============
|   14%
| Clearly, that's not very helpful! Since my_vector is a vector, it doesn't h
ave a `dim` attribute (so
| it's just NULL), but we can find its length using the length() function. Tr
y that now.

> length(my_vector)
[1] 20

| All that hard work is paying off!

    |================
|   17%
| Ah! That's what we wanted. But, what happens if we give my_vector a `dim` a
ttribute? Let's give it a
```

```
| try. Type dim(my_vector) <- c(4, 5).

> dim(my_vector)<-c(4,5)

| Nice work!

  |==================
|  19%
| It's okay if that last command seemed a little strange to you. It should! T
he dim() function allows
| you to get OR set the `dim` attribute for an R object. In this case, we ass
igned the value c(4, 5)
| to the `dim` attribute of my_vector.

...

  |=====================
|  22%
| Use dim(my_vector) to confirm that we've set the `dim` attribute correctly.

> dim(my_vector)
[1] 4 5

| You're the best!

  |=======================
|  25%
| Another way to see this is by calling the attributes() function on my_vecto
r. Try it now.

> attributes(my_vector)
$dim
[1] 4 5


| You are quite good my friend!

  |==========================
|  28%
| Just like in math class, when dealing with a 2-dimensional object (think re
ctangular table), the
| first number is the number of rows and the second is the number of columns.
Therefore, we just gave
| my_vector 4 rows and 5 columns.

...

  |============================
|  31%
| But, wait! That doesn't sound like a vector any more. Well, it's not. Now i
t's a matrix. View the
| contents of my_vector now to see what it looks like.

> my_vector
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20

| You are amazing!

  |===============================
|  33%
| Now, let's confirm it's actually a matrix by using the class() function. Ty
pe class(my_vector) to
| see what I mean.

> class(my_vector)
[1] "matrix"

| That's the answer I was looking for.

  |====================================
|  36%
| Sure enough, my_vector is now a matrix. We should store it in a new variabl
e that helps us remember
| what it is. Store the value of my_vector in a new variable called my_matrix
.

> my_matrix<-my_vector

| You are amazing!

  |======================================
|  39%
| The example that we've used so far was meant to illustrate the point that a
matrix is simply an
| atomic vector with a dimension attribute. A more direct method of creating
the same matrix uses the
| matrix() function.

...

  |=========================================
|  42%
| Bring up the help file for the matrix() function now using the `?` function
.

> ?matrix

| You're the best!

  |===========================================
|  44%
| Now, look at the documentation for the matrix function and see if you can f
igure out how to create a
| matrix containing the same numbers (1-20) and dimensions (4 rows, 5 columns
) by calling the matrix()
| function. Store the result in a variable called my_matrix2.

>
> my_matrix2<-matrix(data = 1:20,nrow = 4,ncol = 5,byrow = FALSE)

| Nice work!

  |==============================================
|  47%
| Finally, let's confirm that my_matrix and my_matrix2 are actually identical
. The identical()
| function will tell us if its first two arguments are the same. Try it out.

> identical(my_matrix,mymatr)
Error in identical(my_matrix, mymatr) : object 'mymatr' not found
> identical(my_matrix,my_matrix2)
[1] TRUE

| Excellent job!
```

```
  |==============================================
|  50%
| Now, imagine that the numbers in our table represent some measurements from
a clinical experiment,
| where each row represents one patient and each column represents one variab
le for which measurements
| were taken.

...

  |=================================================
|  53%
| We may want to label the rows, so that we know which numbers belong to each
patient in the
| experiment. One way to do this is to add a column to the matrix, which cont
ains the names of all
| four people.

...

  |====================================================
|  56%
| Let's start by creating a character vector containing the names of our pati
ents -- Bill, Gina,
| Kelly, and Sean. Remember that double quotes tell R that something is a cha
racter string. Store the
| result in a variable called patients.

> patients<-c("Bill","Gina","Kelly","Sean")

| All that practice is paying off!

  |======================================================
|  58%
| Now we'll use the cbind() function to 'combine columns'. Don't worry about
storing the result in a
| new variable. Just call cbind() with two arguments -- the patients vector a
nd my_matrix.

> cbind(patients,my_matrix)
     patients
[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"

| Nice work!

  |=========================================================
|  61%
| Something is fishy about our result! It appears that combining the characte
r vector with our matrix
| of numbers caused everything to be enclosed in double quotes. This means we
're left with a matrix of
| character strings, which is no good.

...

  |===========================================================
|  64%
| If you remember back to the beginning of this lesson, I told you that matri
ces can only contain ONE
```

```
| class of data. Therefore, when we tried to combine a character vector with
a numeric matrix, R was
| forced to 'coerce' the numbers to characters, hence the double quotes.

...

  |==============================================================
|  67%
| This is called 'implicit coercion', because we didn't ask for it. It just h
appened. But why didn't R
| just convert the names of our patients to numbers? I'll let you ponder that
question on your own.

...

  |=================================================================
|  69%
| So, we're still left with the question of how to include the names of our p
atients in the table
| without destroying the integrity of our numeric data. Try the following --
my_data <-
| data.frame(patients, my_matrix)

> my_data<-data.frame(patients,my_matrix)

| Nice work!

  |===================================================================
|  72%
| Now view the contents of my_data to see what we've come up with.

> my_data
  patients X1 X2 X3 X4 X5
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

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

  |======================================================================
|  75%
| It looks like the data.frame() function allowed us to store our character v
ector of names right
| alongside our matrix of numbers. That's exactly what we were hoping for!

...

  |========================================================================
|  78%
| Behind the scenes, the data.frame() function takes any number of arguments
and returns a single
| object of class `data.frame` that is composed of the original objects.

...

  |==========================================================================
=                  |  81%
| Let's confirm this by calling the class() function on our newly created dat
a frame.

> class(my_data)
[1] "data.frame"
```

```
| You are really on a roll!

  |===========================================================================
====                |  83%
| It's also possible to assign names to the individual rows and columns of a
data frame, which
| presents another possible way of determining which row of values in our tab
le belongs to each
| patient.

...

  |===========================================================================
======              |  86%
| However, since we've already solved that problem, let's solve a different p
roblem by assigning names
| to the columns of our data frame so that we know what type of measurement e
ach column represents.

...

  |===========================================================================
=========           |  89%
| Since we have six columns (including patient names), we'll need to first cr
eate a vector containing
| one element for each column. Create a character vector called cnames that c
ontains the following
| values (in order) -- "patient", "age", "weight", "bp", "rating", "test".

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

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

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

> colnames(cnames,my_data)
Error in if (do.NULL) NULL else if (nc > 0L) paste0(prefix, seq_len(nc)) else
character() :
  argument is not interpretable as logical
In addition: Warning message:
In if (do.NULL) NULL else if (nc > 0L) paste0(prefix, seq_len(nc)) else chara
cter() :
  the condition has length > 1 and only the first element will be used
> colnames(my_data)<-cnames

| Nice 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 hard work is paying off!
```

```
  |===============================================================================
=================   |  97%
| In this lesson, you learned the basics of working with two very important a
nd common data structures
| -- matrices and data frames. There's much more to learn and we'll be coveri
ng 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: 1
What is your email address? robteotia@gmail.com
What is your assignment token? WzVmvwiPDedKW2R2
Grade submission succeeded!

| You got it right!

| 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!
```