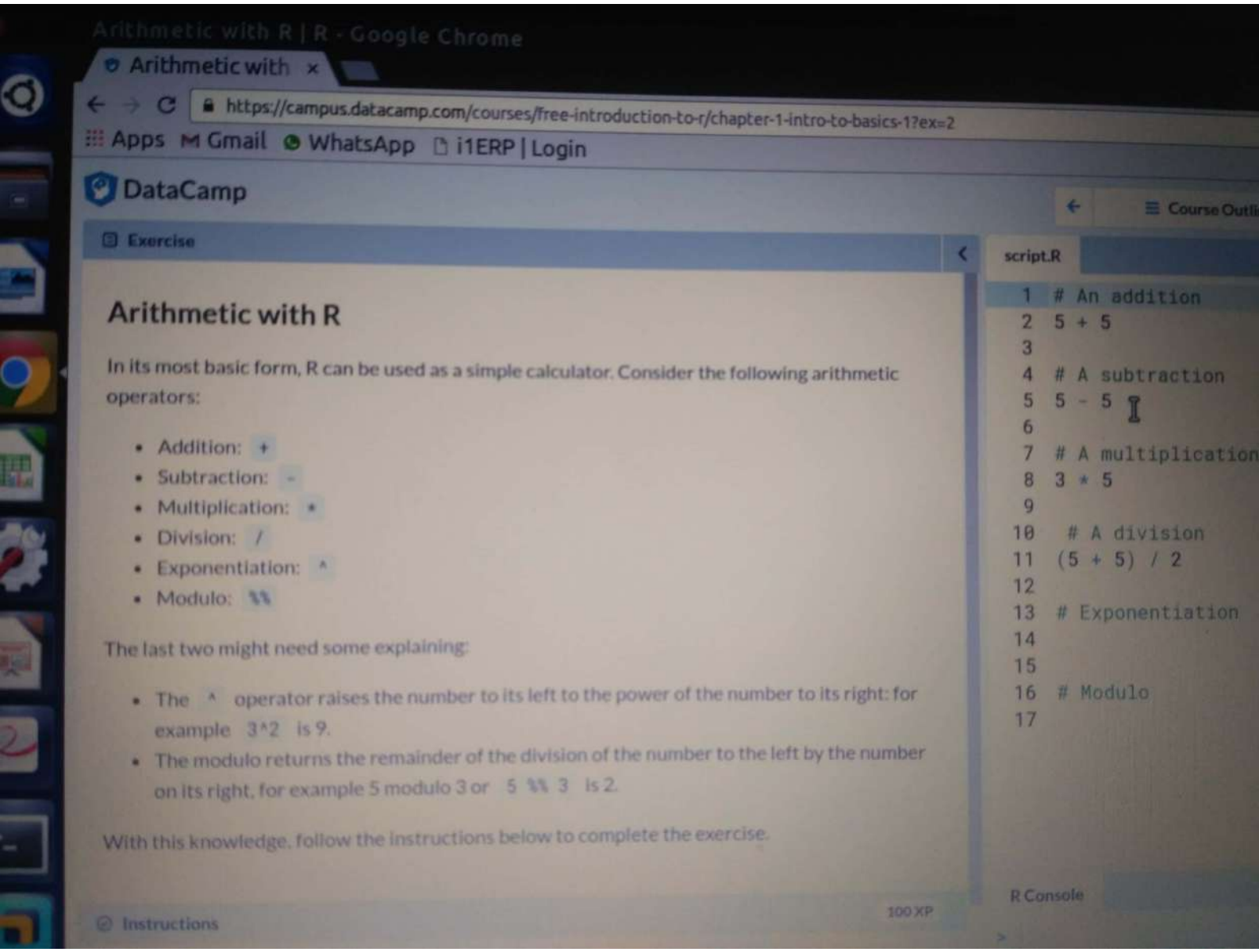
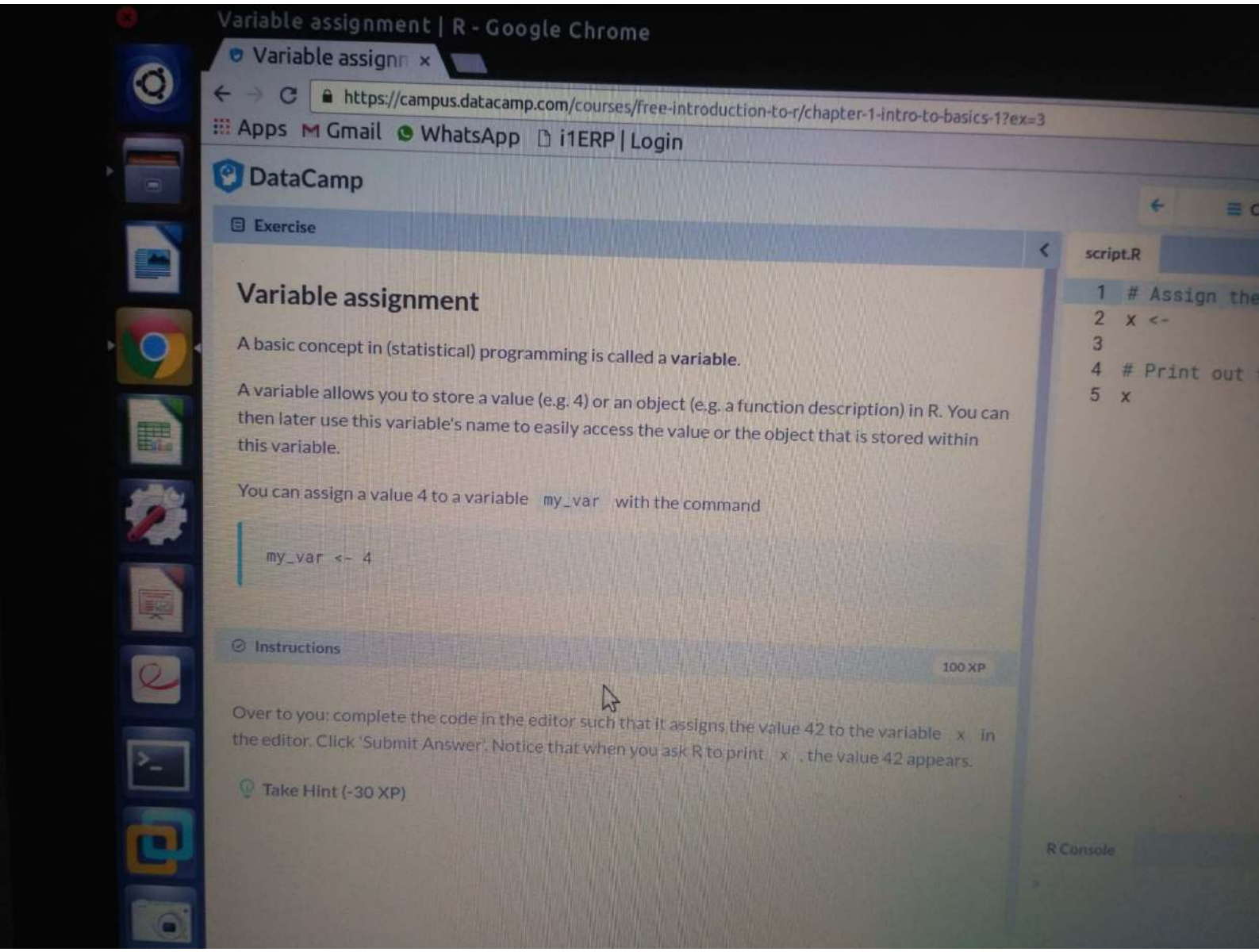
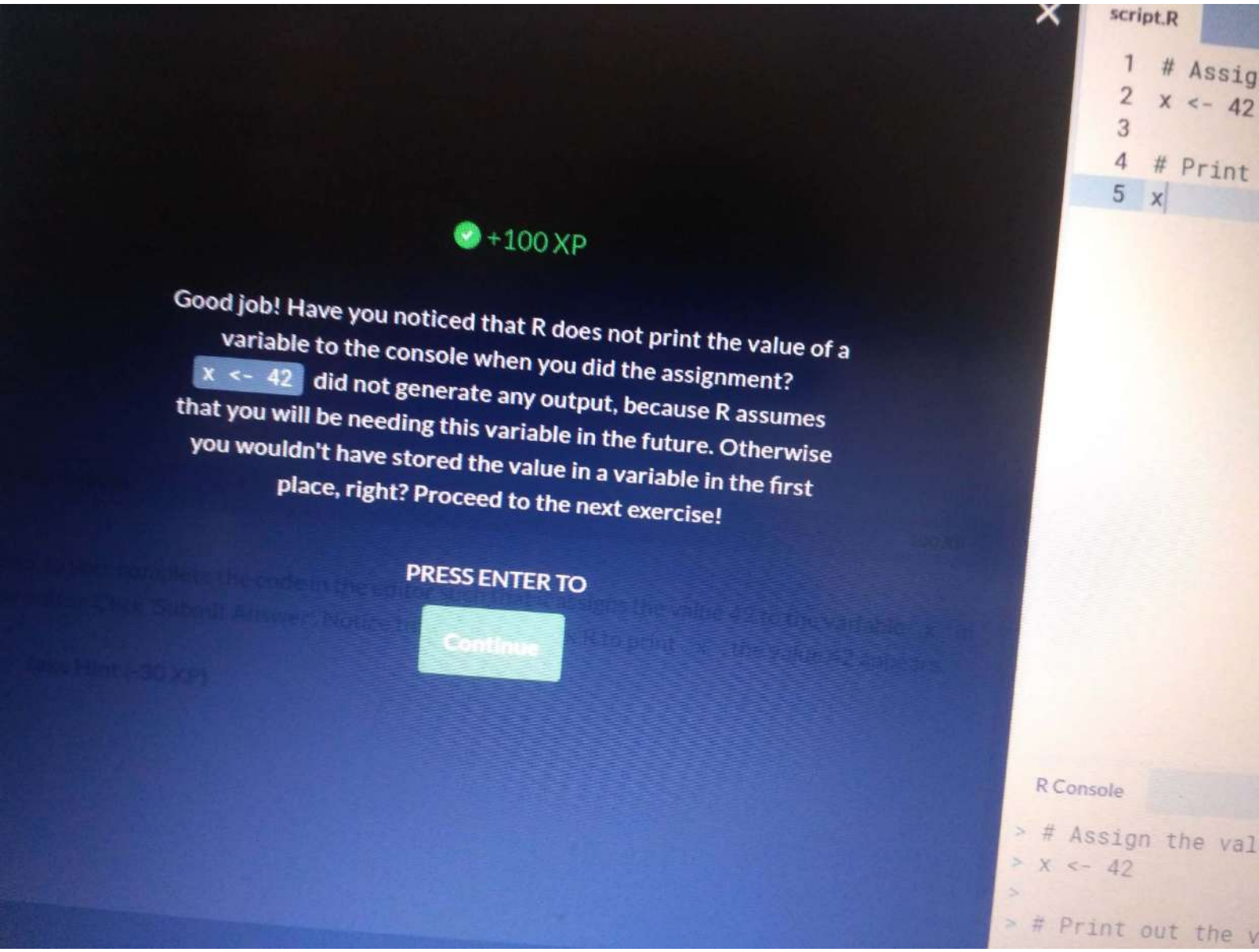
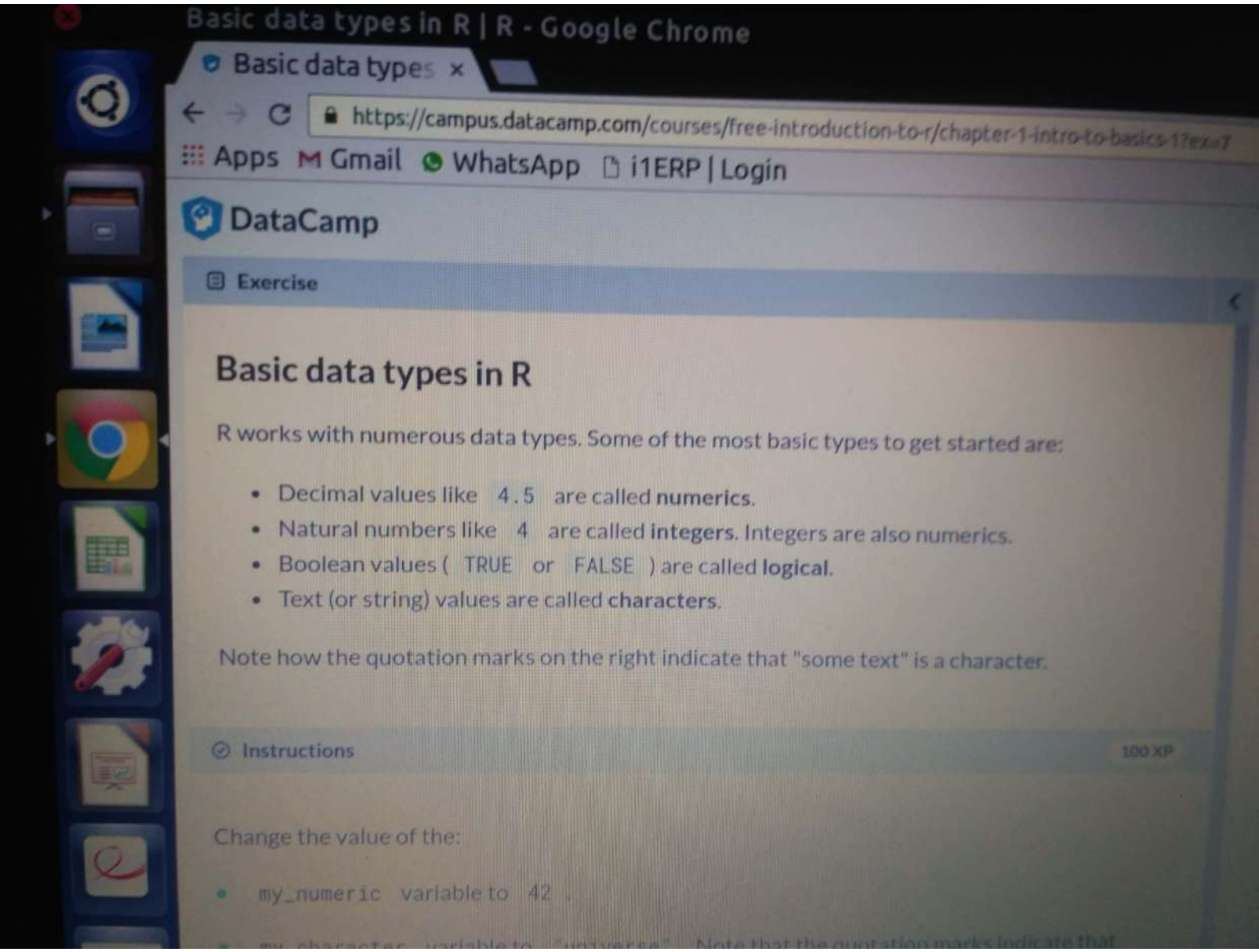
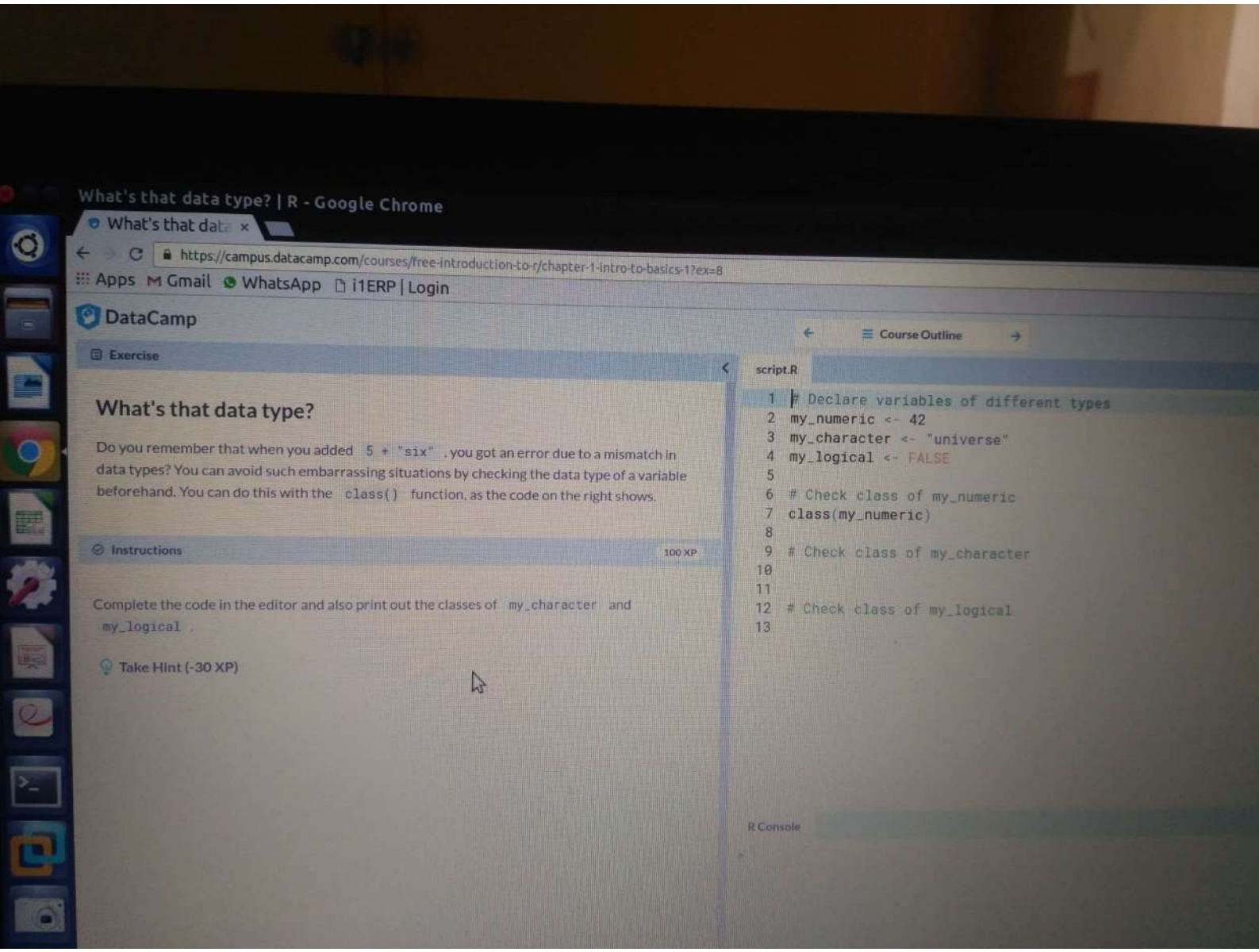
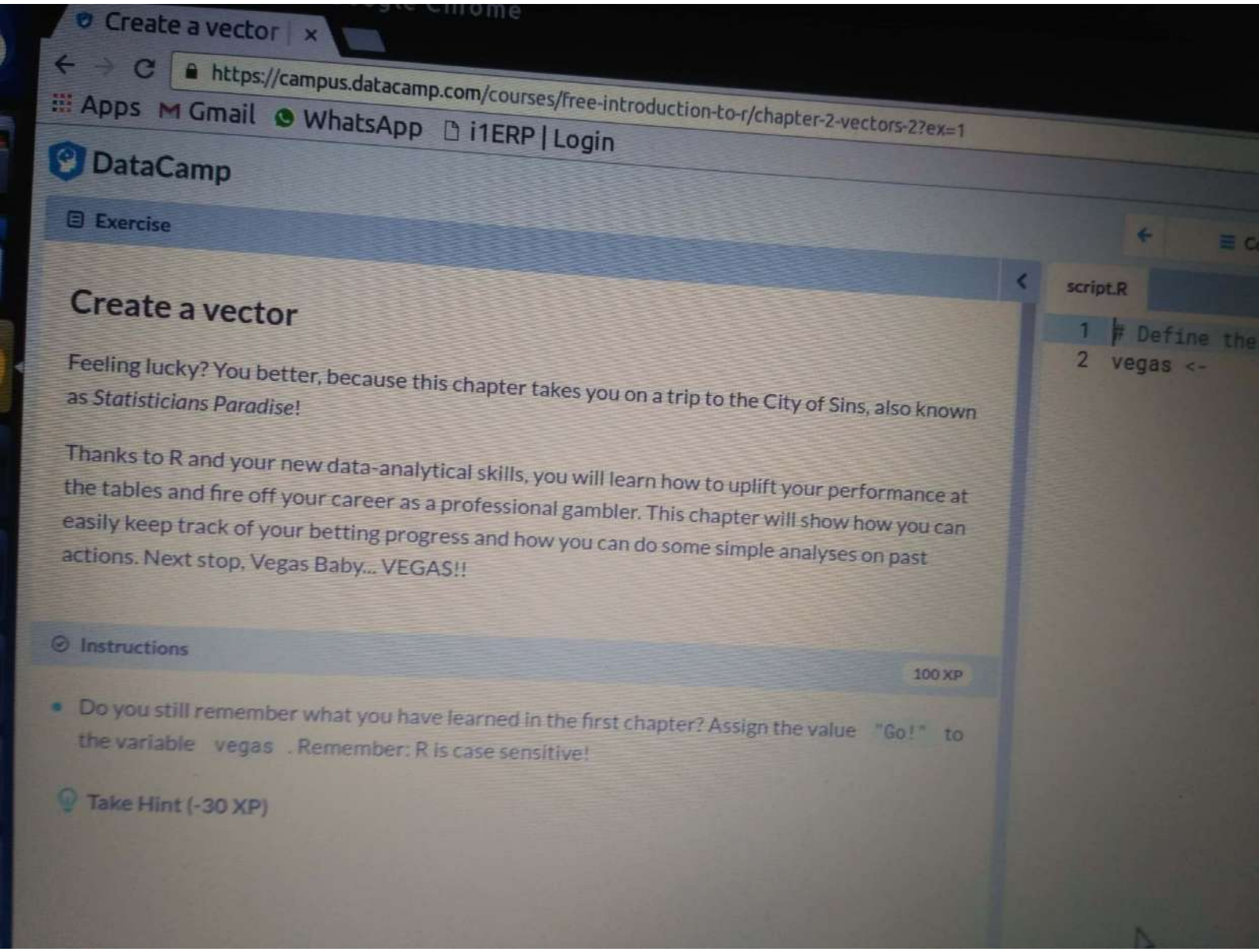












Exercise

Create a vector (2)

Let us focus first!

On your way from rags to riches, you will make extensive use of vectors. Vectors are one-dimension arrays that can hold numeric data, character data, or logical data. In other words, a vector is a simple tool to store data. For example, you can store your daily gains and losses in the casinos.

In R, you create a vector with the combine function `c()`. You place the vector elements separated by a comma between the parentheses. For example:

```
numeric_vector <- c(1, 2, 3)
character_vector <- c("a", "b", "c")
```

Once you have created these vectors in R, you can use them to do calculations.

Instructions

100 XP

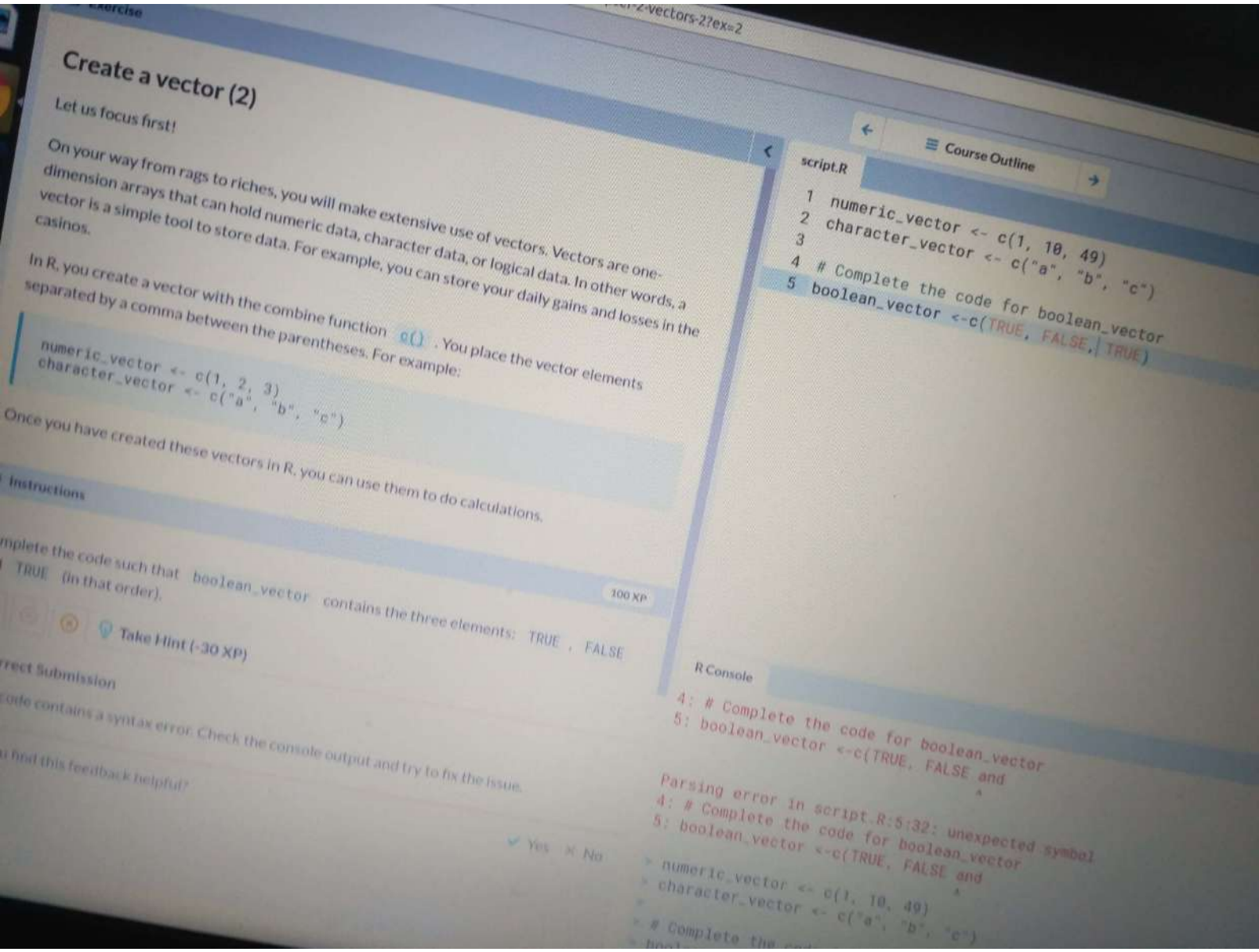
Complete the code such that `boolean_vector` contains the three elements: `TRUE`, `FALSE` and `TRUE` (in that order).

Take Hint (-30 XP)

script

1
2
3
4
5

R Console



Exercise

Naming a vector

As a data analyst, it is important to have a clear view on the data that you are using. Understanding what each element refers to is therefore essential.

In the previous exercise, we created a vector with your winnings over the week. Each vector element refers to a day of the week but it is hard to tell which element belongs to which day. It would be nice if you could show that in the vector itself.

You can give a name to the elements of a vector with the `names()` function. Have a look at this example:

```
some_vector <- c("John Doe", "poker player")
names(some_vector) <- c("Name", "Profession")
```

This code first creates a vector `some_vector` and then gives the two elements a name. The first element is assigned the name `Name`, while the second element is labeled `Profession`. Printing the contents to the console yields following output:

```
        Name       Profession
"John Doe"  "poker player"
```

R Consol

Instructions

100 XP

- The code on the right names the elements in `poker_vector` with the days of the week. Add code to do the same thing for `roulette_vector`.

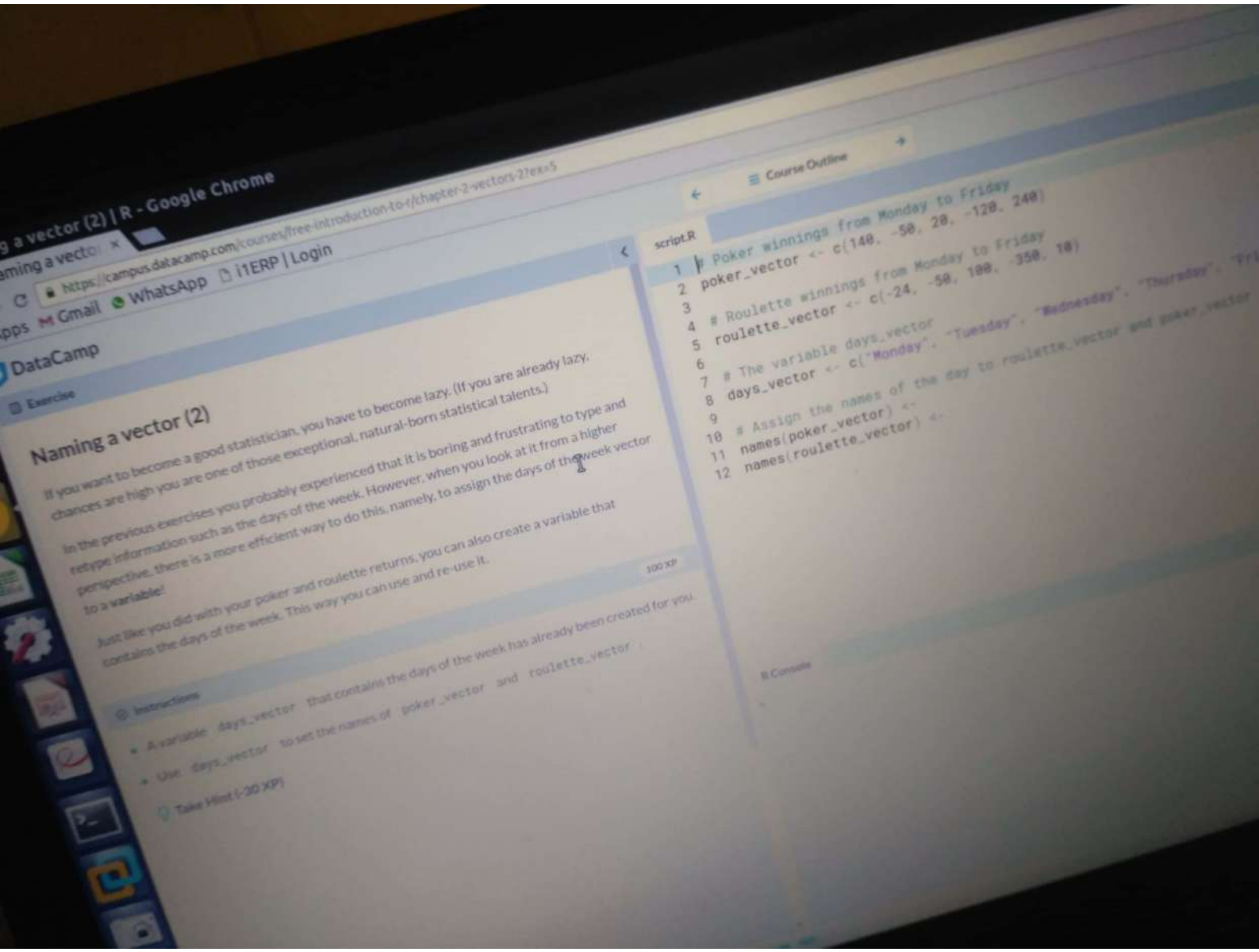
Take Hint (-30 XP)

Course Outline

script.R

```r
# Poker winnings from Monday to Friday
poker_vector <- c(140, -50, 20, -120, 240)

# Roulette winnings from Monday to Friday
roulette_vector <- c(-24, -50, 100, -350, 10)

# Assign days as names of poker_vector
names(poker_vector) <- c("Monday", "Tuesday", "Wednesday", "Thursday", "

# Assign days as names of roulette_vector

```

R Console



Naming a vector (2) | R - Google Chrome

Naming a vector ×

https://campus.datacamp.com/courses/free-introduction-to-r/chapter-2-vectors-2?ex=5

Apps Gmail WhatsApp i1ERP | Login

DataCamp

Exercise

Naming a vector (2)

If you want to become a good statistician, you have to become lazy. (If you are already lazy, chances are high you are one of those exceptional, natural-born statistical talents.)

In the previous exercises you probably experienced that it is boring and frustrating to type and retype information such as the days of the week. However, when you look at it from a higher perspective, there is a more efficient way to do this, namely, to assign the days of the week vector to a **variable**!

Just like you did with your poker and roulette returns, you can also create a variable that contains the days of the week. This way you can use and re-use it.

Instructions

100 XP

- A variable `days_vector` that contains the days of the week has already been created for you.
- Use `days_vector` to set the names of `poker_vector` and `roulette_vector`.

Take Hint (-30 XP)

Course Outline

cript.R

```
1  # Poker winnings from Monday to Friday
2  poker_vector <- c(140, -50, 20, -120, 240)
3
4  # Roulette winnings from Monday to Friday
5  roulette_vector <- c(-24, -50, 100, -350, 10)
6
7  # The variable days_vector
8  days_vector <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
9
10 # Assign the names of the day to roulette_vector and poker_vector
11 names(poker_vector) <- (days_vector)
12 names(roulette_vector) <-(days_vector)
```

Run Code

R Console

```
> # Poker winnings from Monday to Friday
> poker_vector <- c(140, -50, 20, -120, 240)
>
> # Roulette winnings from Monday to Friday
> roulette_vector <- c(-24, -50, 100, -350, 10)
>
> # The variable days_vector
> days_vector <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
>
> # Assign the names of the day to roulette_vector and poker_vector
> names(poker_vector) <- (days_vector)
> names(roulette_vector) <-(days_vector)
>
```

- Have you lost money over the week in total?
- Are you winning/losing money on poker or on roulette?

To get the answers, you have to do arithmetic calculations on vectors.

It is important to know that if you sum two vectors in R, it takes the element-wise sum. For example, the following three statements are completely equivalent:

```
c(1, 2, 3) + c(4, 5, 6)
c(1 + 4, 2 + 5, 3 + 6)
c(5, 7, 9)
```

You can also do the calculations with variables that represent vectors:

```
a <- c(1, 2, 3)
b <- c(4, 5, 6)
c <- a + b
```

R Console

>

Instructions

100 XP

- Take the sum of the variables `A_vector` and `B_vector` and assign it to `total_vector`
- Inspect the result by printing out `total_vector`

Take Hint (-30 XP)

DataCamp

Exercise

Calculating total winnings (3)

Based on the previous analysis, it looks like you had a mix of good and bad days. This is not what your ego expected, and you wonder if there may be a very tiny chance you have lost money over the week in total?

A function that helps you to answer this question is `sum()` . It calculates the sum of all elements of a vector. For example, to calculate the total amount of money you have lost/won with poker you do:

```
total_poker <- sum(poker_vector)
```

Instructions

100 XP

- Calculate the total amount of money that you have won/lost with roulette and assign to the variable `total_roulette` .
- Now that you have the totals for roulette and poker, you can easily calculate `total_week` (which is the sum of all gains and losses of the week).
- Print out `total_week` .

Take Hint (-30 XP)

DataCamp

Exercise

Comparing total winnings

Oops, it seems like you are losing money. Time to rethink and adapt your strategy! This will require some deeper analysis...

After a short brainstorm in your hotel's jacuzzi, you realize that a possible explanation might be that your skills in roulette are not as well developed as your skills in poker. So maybe your total gains in poker are higher (or `>`) than in roulette.

Instructions

100 XP

- Calculate `total_poker` and `total_roulette` as in the previous exercise. Use the `sum()` function twice.
- Check if your total gains in poker are higher than for roulette by using a comparison. Simply print out the result of this comparison. What do you conclude, should you focus on roulette or on poker?

Take Hint (-30 XP)

DataCamp

Exercise

Vector selection: the good times

Your hunch seemed to be right. It appears that the poker game is more your cup of tea than roulette.

Another possible route for investigation is your performance at the beginning of the working week compared to the end of it. You did have a couple of Margarita cocktails at the end of the week...

To answer that question, you only want to focus on a selection of the `total_vector`. In other words, our goal is to select specific elements of the vector. To select elements of a vector (and later matrices, data frames, ...), you can use square brackets. Between the square brackets, you indicate what elements to select. For example, to select the first element of the vector, you type `poker_vector[1]`. To select the second element of the vector, you type `poker_vector[2]`, etc. Notice that the first element in a vector has index 1, not 0 as in many other programming languages.

Instructions

100 XP

Assign the poker results of Wednesday to the variable `poker_wednesday`.

Take Hint (-30 XP)

Vector selection: the good times (2) | R - Google Chrome

Vector selection ×

https://campus.datacamp.com/courses/free-introduction-to-r/chapter-2-vectors-2?ex=11

Apps Gmail WhatsApp i1ERP | Login

DataCamp

Exercise

Vector selection: the good times (2)

How about analyzing your midweek results?

To select multiple elements from a vector, you can add square brackets at the end of it. You can indicate between the brackets what elements should be selected. For example: suppose you want to select the first and the fifth day of the week: use the vector `c(1, 5)` between the square brackets. For example, the code below selects the first and fifth element of `poker_vector` :

```
poker_vector[c(1, 5)]
```

Instructions

100 XP

Assign the poker results of Tuesday, Wednesday and Thursday to the variable `poker_midweek` .

Take Hint (-30 XP)

Course Outline

script.R

```
1  # Poker and roulette winnings from Monday to Friday:
2  poker_vector <- c(140, -50, 20, -120, 240)
3  roulette_vector <- c(-24, -50, 100, -350, 10)
4  days_vector <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday"
5  names(poker_vector) <- days_vector
6  names(roulette_vector) <- days_vector
7
8  # Define a new variable based on a selection
9  poker_midweek <- poker_vector[c(2,3,4)]
```

R Console

```
>
> # Define a new variable based on a selection
> poker_midweek <- poker_vector[2,3,4]
Error: incorrect number of dimensions
> # Poker and roulette winnings from Monday to Friday:
> poker_vector <- c(140, -50, 20, -120, 240)
> roulette_vector <- c(-24, -50, 100, -350, 10)
> days_vector <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
> names(poker_vector) <- days_vector
> names(roulette_vector) <- days_vector
>
> # Define a new variable based on a selection
> poker_midweek <- poker_vector[c(2,3,4)]
>
```

Vector selection: the good times (3) | R - Google Chrome

Vector selection ×

https://campus.datacamp.com/courses/free-introduction-to-r/chapter-2-vectors-2?ex=12

Apps Gmail WhatsApp i1ERP | Login

DataCamp

Exercise

Vector selection: the good times (3)

Selecting multiple elements of `poker_vector` with `c(2, 3, 4)` is not very convenient. Many statisticians are lazy people by nature, so they created an easier way to do this: `c(2, 3, 4)` can be abbreviated to `2:4`, which generates a vector with all natural numbers from 2 up to 4.

So, another way to find the mid-week results is `poker_vector[2:4]`. Notice how the vector `2:4` is placed between the square brackets to select element 2 up to 4.

Instructions

100 XP

Assign to `roulette_selection_vector` the roulette results from Tuesday up to Friday; make use of `:` if it makes things easier for you.

Take Hint (-30 XP)

R Cons

DataCamp

Exercise

Vector selection: the good times (4)

Another way to tackle the previous exercise is by using the names of the vector elements (Monday, Tuesday, ...) instead of their numeric positions. For example,

```
poker_vector["Monday"]
```

will select the first element of `poker_vector` since `"Monday"` is the name of that first element.

Just like you did in the previous exercise with numerics, you can also use the element names to select multiple elements, for example:

```
poker_vector[c("Monday","Tuesday")]
```

Instructions

100 XP

- Select the first three elements in `poker_vector` by using their names: `"Monday"`, `"Tuesday"` and `"Wednesday"`. Assign the result of the selection to `poker_start`.
- Calculate the average of the values in `poker_start` with the `mean()` function. Simply print out the result so you can inspect it.

Take Hint (-30 XP)

Selection by comparison - Step 1

By making use of comparison operators, we can approach the previous question in a more proactive way.

The (logical) comparison operators known to R are:

- `<` for less than
- `>` for greater than
- `<=` for less than or equal to
- `>=` for greater than or equal to
- `==` for equal to each other
- `!=` not equal to each other

As seen in the previous chapter, stating `6 > 5` returns `TRUE`. The nice thing about R is that you can use these comparison operators also on vectors. For example:

```
> c(4, 5, 6) > 5
[1] FALSE FALSE  TRUE
```

This command tests for every element of the vector if the condition stated by the comparison operator is `TRUE` or `FALSE`.

Instructions

100 XP

- Check which elements in `poker_vector` are positive (i.e. > 0) and assign this to `selection_vector`.
- Print out `selection_vector` so you can inspect it. The printout tells you whether you won (`TRUE`) or lost (`FALSE`) any money for each day.

Take Hint (-30 XP)

scrip

1
2
3
4
5
6
7
8 #
9 se
10
11 #
12

R Console

>

Selection by comparison - Step 2 | R - Google Chrome

Selection by co... | Login - HackerR...

https://campus.datacamp.com/courses/free-introduction-to-r/chapter-2-vectors-2?ex=15

Apps | Gmail | WhatsApp | i1ERP | Login

DataCamp

Exercise

Selection by comparison - Step 2

Working with comparisons will make your data analytical life easier. Instead of selecting a subset of days to investigate yourself (like before), you can simply ask R to return only those days where you realized a positive return for poker.

In the previous exercises you used `selection_vector <- poker_vector > 0` to find the days on which you had a positive poker return. Now, you would like to know not only the days on which you won, but also how much you won on those days.

You can select the desired elements, by putting `selection_vector` between the square brackets that follow `poker_vector` :

```
poker_vector[selection_vector]
```

R knows what to do when you pass a logical vector in square brackets: it will only select the elements that correspond to `TRUE` in `selection_vector` .

Instructions

100 XP

Use `selection_vector` in square brackets to assign the amounts that you won on the profitable days to the variable `poker_winning_days` .

Take Hint (-30 XP)

DataCamp

Exercise

What's a matrix?

In R, a matrix is a collection of elements of the same data type (numeric, character, or logical) arranged into a fixed number of rows and columns. Since you are only working with rows and columns, a matrix is called two-dimensional.

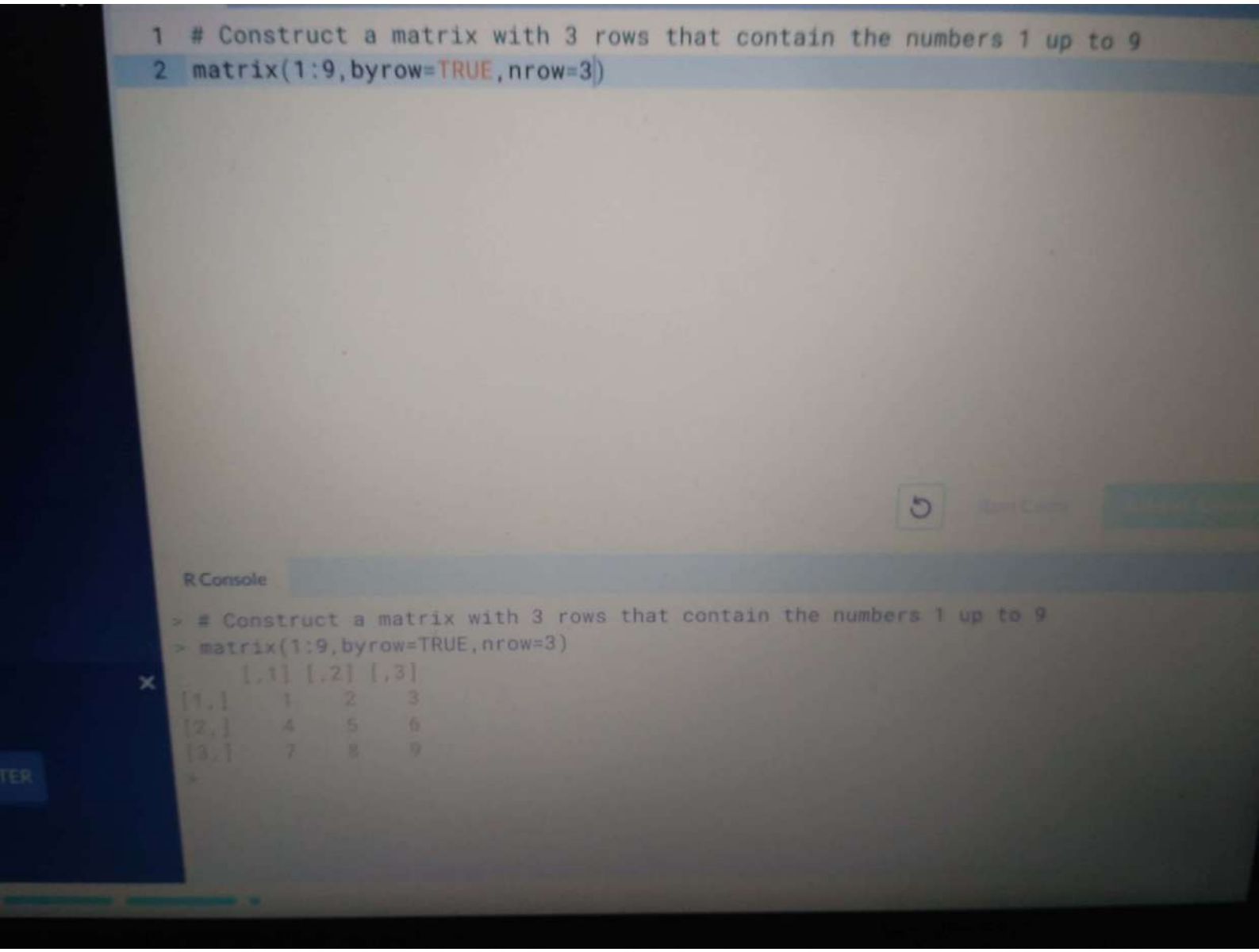
You can construct a matrix in R with the `matrix()` function. Consider the following example:

```
matrix(1:9, byrow = TRUE, nrow = 3)
```

In the `matrix()` function:

- The first argument is the collection of elements that R will arrange into the rows and columns of the matrix. Here, we use `1:9` which is a shortcut for `c(1, 2, 3, 4, 5, 6, 7, 8, 9)`.
- The argument `byrow` indicates that the matrix is filled by the rows. If we want the matrix to be filled by the columns, we just place `byrow = FALSE`.
- The third argument `nrow` indicates that the matrix should have three rows.

Instructions 100 XP



It is now time to get your hands dirty. In the following exercises you will analyze the box office numbers of the Star Wars franchise. May the force be with you!

In the editor, three vectors are defined. Each one represents the box office numbers from the first three Star Wars movies. The first element of each vector indicates the US box office revenue, the second element refers to the Non-US box office (source: Wikipedia).

In this exercise, you'll combine all these figures into a single vector. Next, you'll build a matrix from this vector.

100 XP

Instructions

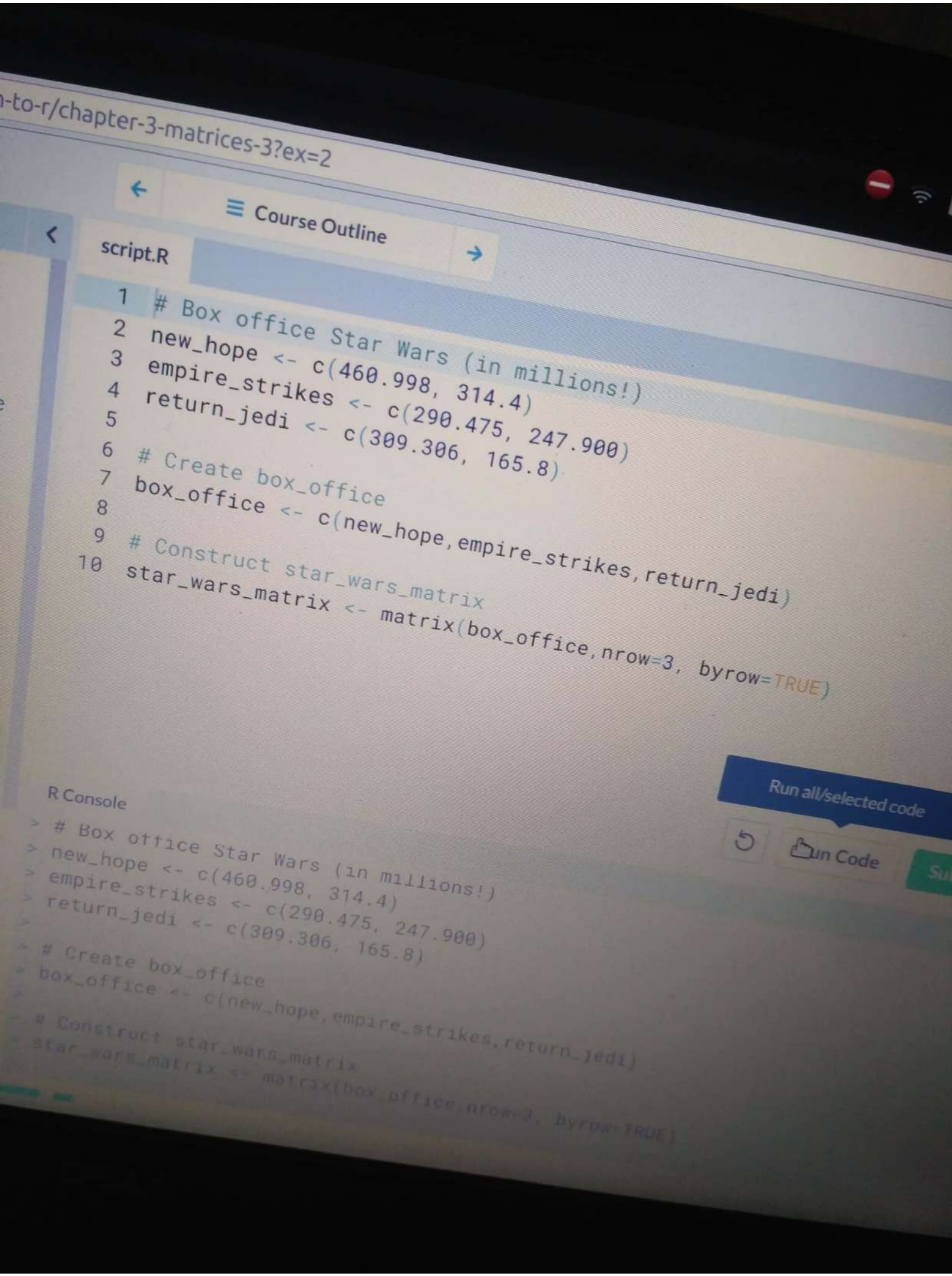
- Use `c(new_hope, empire_strikes, return_jedi)` to combine the three vectors into one vector. Call this vector `box_office`.
- Construct a matrix with 3 rows, where each row represents a movie. Use the `matrix()` function to do this. The first argument is the vector `box_office`, containing all box office figures. Next, you'll have to specify `nrow = 3` and `byrow = TRUE`. Name the resulting matrix `star_wars_matrix`.

Take Hint (-30 XP)

```
3 empir
4 return
5
6 # Crea
7 box_of
8
9 # Const
10 star_wa
```

R Console

>



To help you remember what is stored in `star_wars_matrix`, you would like to add the names of the movies for the rows. Not only does this help you to read the data, but it is also useful to select certain elements from the matrix.

Similar to vectors, you can add names for the rows and the columns of a matrix

```
rownames(my_matrix) <- row_names_vector
colnames(my_matrix) <- col_names_vector
```

We went ahead and prepared two vectors for you: `region`, and `titles`. You will need these vectors to name the columns and rows of `star_wars_matrix`, respectively.

Instructions 100 XP

- Use `colnames()` to name the columns of `star_wars_matrix` with the `region` vector.
- Use `rownames()` to name the rows of `star_wars_matrix` with the `titles` vector.
- Print out `star_wars_matrix` to see the result of your work.

```
2  new
3  emp
4  ret
5
6  # Co
7  star
   byro
8
9  # Vec
10 regio
11 title
12
13 # Name
```

R Console

>

```
    byrow = TRUE)
8
9  # Vectors region and titles, used for naming
10 region <- c("US", "non-US")
11 titles <- c("A New Hope", "The Empire Strikes Back",
12
13 # Name the columns with region
14 colnames(star_wars_matrix)<-region
15
16 # Name the rows with titles
17 rownames(star_wars_matrix)<-titles
18
19 # Print out star_wars_matrix
20 star_wars_matrix
```

R Console

```
> colnames(star_wars_matrix)<-region
>
> # Name the rows with titles
> rownames(star_wars_matrix)<-titles
>
> # Print out star_wars_matrix
> star_wars_matrix
                        US non-US
A New Hope             460.998  314.4
The Empire Strikes Back 290.475  247.9
Return of the Jedi      309.306  165.8
```

Calculating the worldwide box office | R - Chromium

Calculating the worl ×

https://campus.datacamp.com/courses/free-introduction-to-r/cha

DataCamp

Exercise

Calculating the worldwide box office

The single most important thing for a movie in order to become an instant legend in Tinseltown is its worldwide box office figures.

To calculate the total box office revenue for the three Star Wars movies, you have to take the sum of the US revenue column and the non-US revenue column.

In R, the function `rowSums()` conveniently calculates the totals for each row of a matrix. This function creates a new vector:

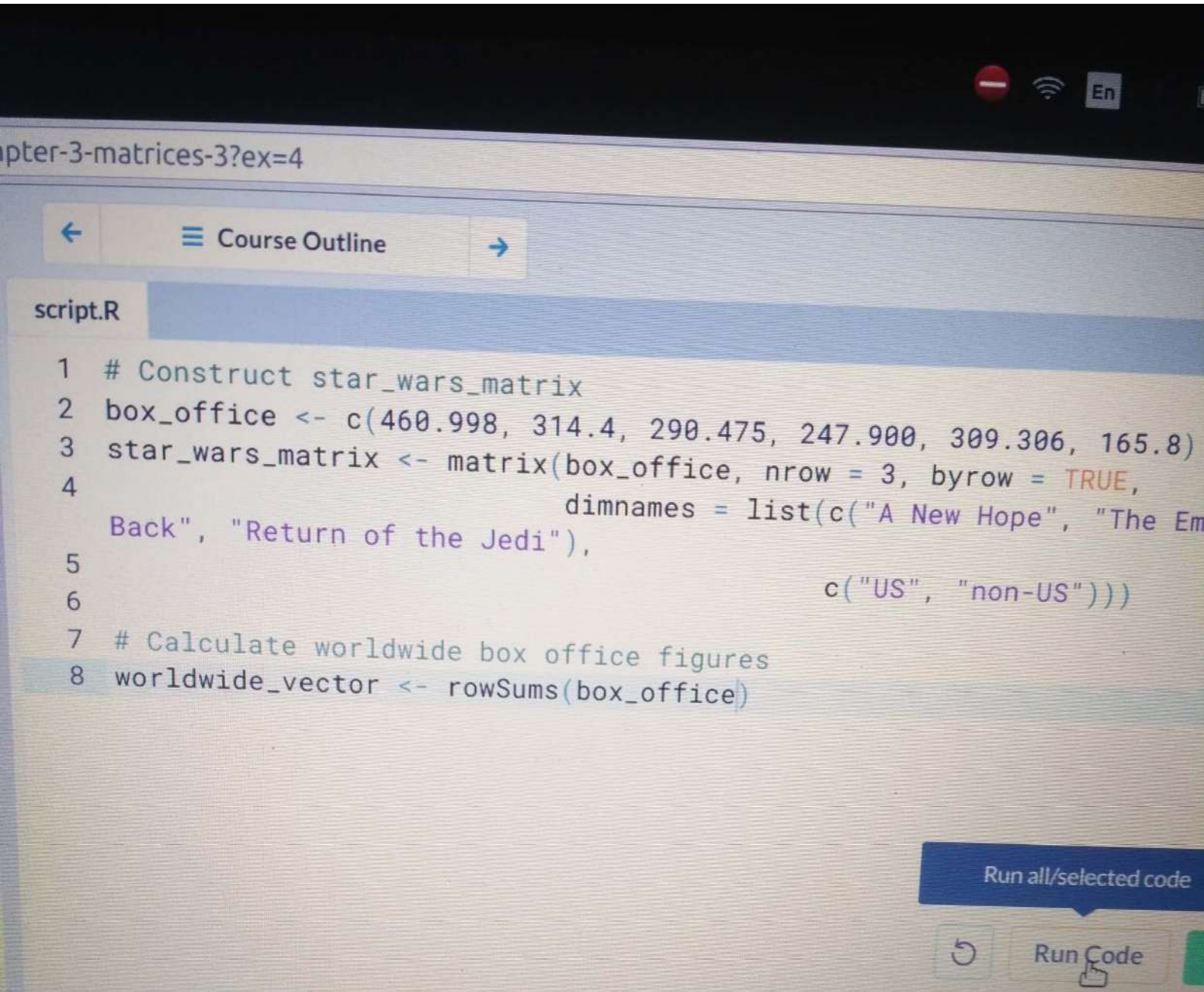
```
rowSums(my_matrix)
```

Instructions

100 XP

Calculate the worldwide box office figures for the three movies and put these in the vector named `worldwide_vector`.

Take Hint (-30 XP)



ecome an instant

ar Wars movies,
the non-US

s the totals for each

100 XP

ee movies and put these

```
3 star_wars_matrix <- matrix(box_office, nrow = 3, byrow = TRUE,
4                            dimnames = list(c("A New Hope", "The Empire St
  Back", "Return of the Jedi"),
5                                            c("US", "non-US")))
6
7 # Calculate worldwide box office figures
8 worldwide_vector <- rowSums(star_wars_matrix)
```

Run all/selected code

Run Code

Submit An

R Console

```
> # Construct star_wars_matrix
> box_office <- c(460.998, 314.4, 290.475, 247.900, 309.306, 165.8)
> star_wars_matrix <- matrix(box_office, nrow = 3, byrow = TRUE,
                             dimnames = list(c("A New Hope", "The Empire Strikes Back",
                                               "Return of the Jedi"),
                                             c("US", "non-US")))
>
> # Calculate worldwide box office figures
> worldwide_vector <- rowSums(star_wars_matrix)
>
```

Adding a column for the Worldwide box office | R - Chromium

Adding a column for ×

https://campus.datacamp.com/courses/free-introduction-to-r/chapter

DataCamp

Exercise

Adding a column for the Worldwide box office

In the previous exercise you calculated the vector that contained the worldwide box office receipt for each of the three Star Wars movies. However, this vector is not yet part of `star_wars_matrix` .

You can add a column or multiple columns to a matrix with the `cbind()` function, which merges matrices and/or vectors together by column. For example:

```
big_matrix <- cbind(matrix1, matrix2, vector1 ...)
```

Instructions

100 XP

Add `worldwide_vector` as a new column to the `star_wars_matrix` and assign the result to `all_wars_matrix` . Use the `cbind()` function.

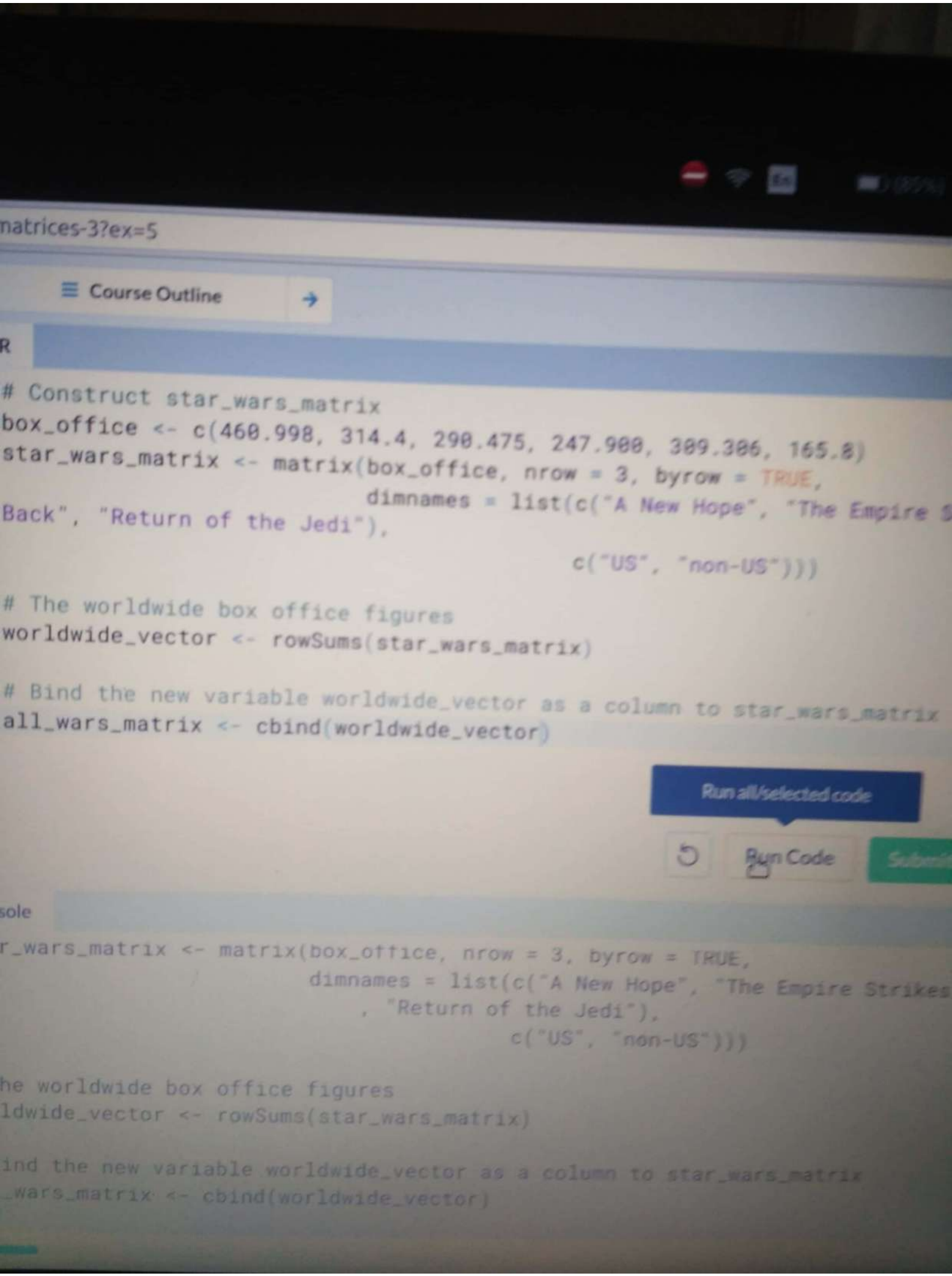
Take Hint (-30 XP)

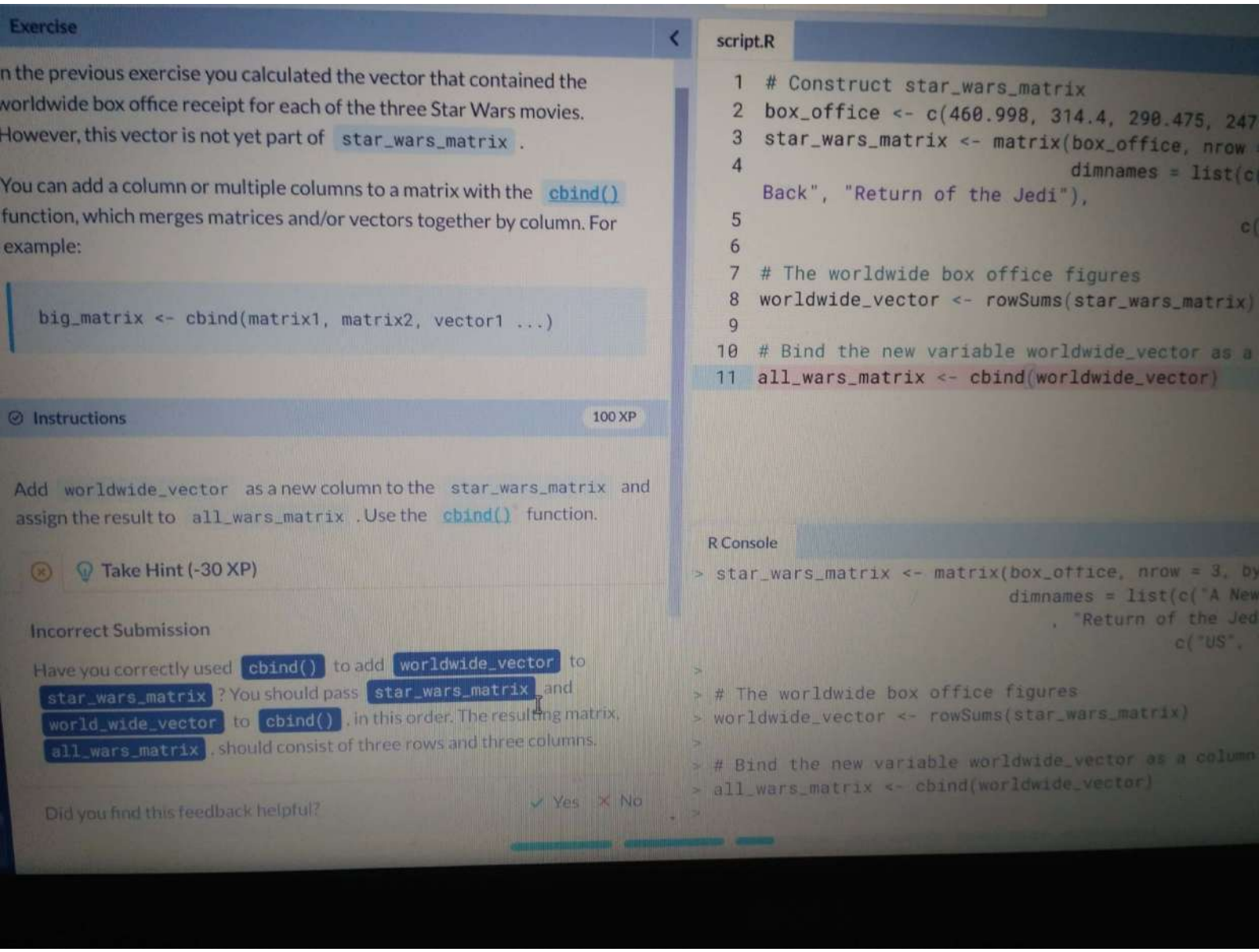
scri

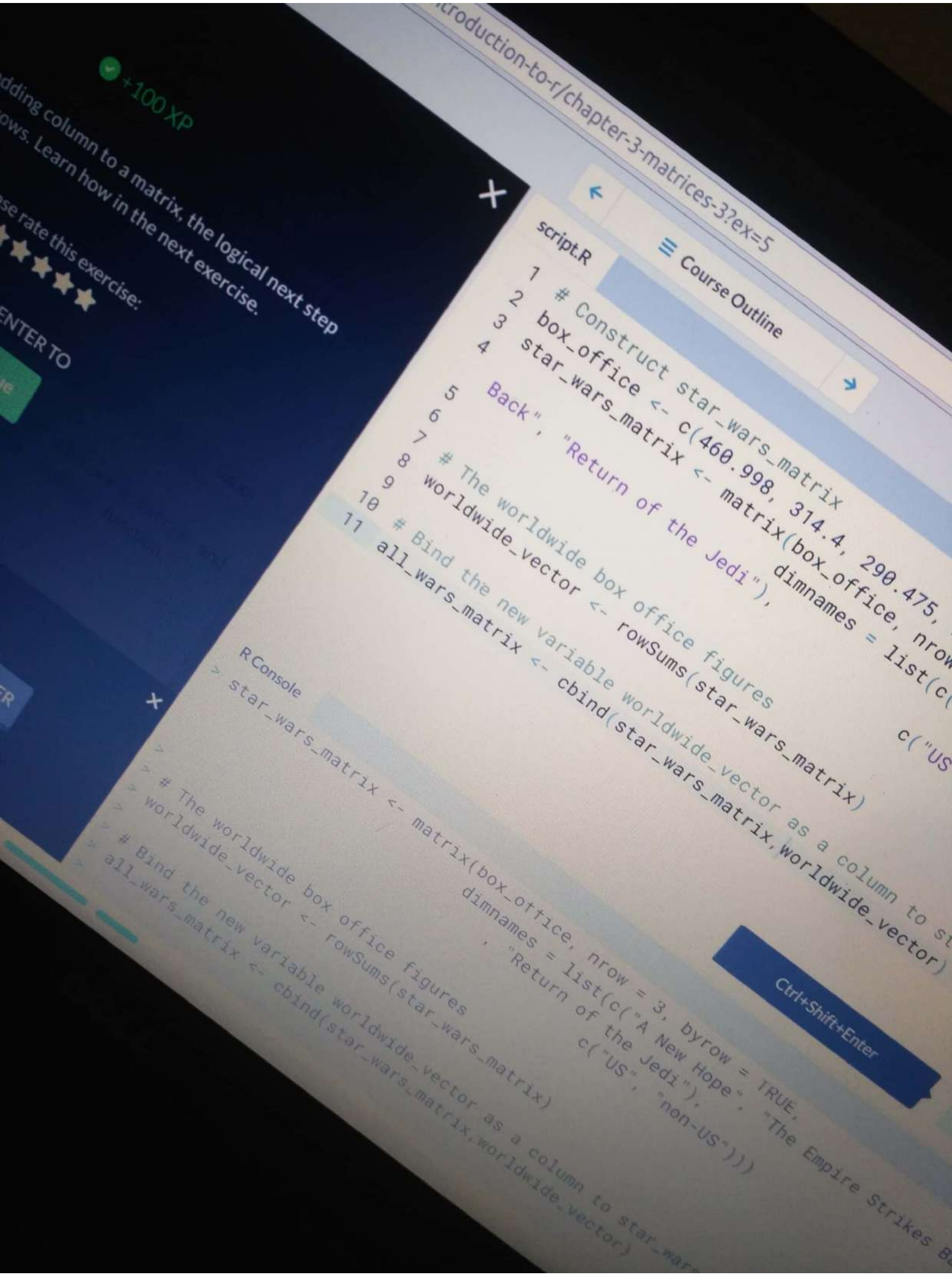
1
2
3
4
5
6
7
8
9
10
11

R Consol

>







The Workspace

The workspace is your current R working environment and includes any user-defined objects (vectors, matrices, data frames, lists, functions). At the end of an R session, the user can save an image of the current workspace that is automatically reloaded the next time R is started. Commands are entered interactively at the R user prompt. **Up** and **down arrow keys** scroll through your command history.

You will probably want to keep different projects in different physical directories. Here are some standard commands for managing your workspace.

```
getwd() # print the current working directory - cwd
ls()    # list the objects in the current workspace
```

```
setwd(mydirectory)      # change to mydirectory
setwd("c:/docs/mydir")  # note / instead of \ in windows
setwd("/usr/rob/mydir") # on linux
```

```
# view and set options for the session
help(options) # learn about available options
options() # view current option settings
options(digits=3) # number of digits to print on output
```

Adding a row | R - Chromium

Adding a row | R ×

https://campus.datacamp.com/courses/free-introduction-to-

DataCamp

Exercise

Adding a row

Just like every action has a reaction, every `cbind()` has an `rbind()`. (We admit, we are pretty bad with metaphors.)

Your R workspace, where all variables you defined 'live' (check out what a workspace is), has already been initialized and contains two matrices:

- `star_wars_matrix` that we have used all along, with data on the original trilogy,
- `star_wars_matrix2`, with similar data for the prequels trilogy.

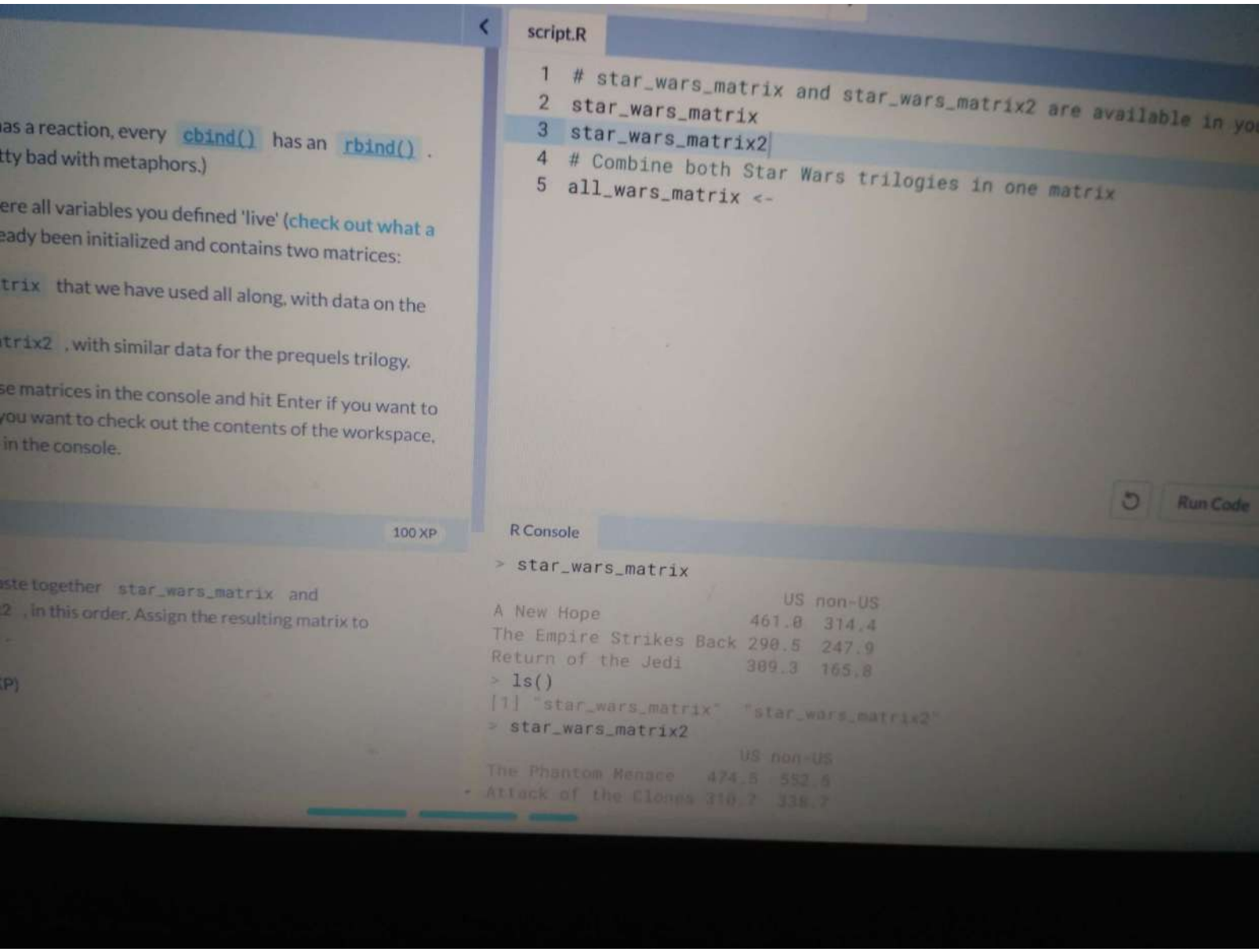
Type the name of these matrices in the console and hit Enter if you want to have a closer look. If you want to check out the contents of the workspace, you can type `ls()` in the console.

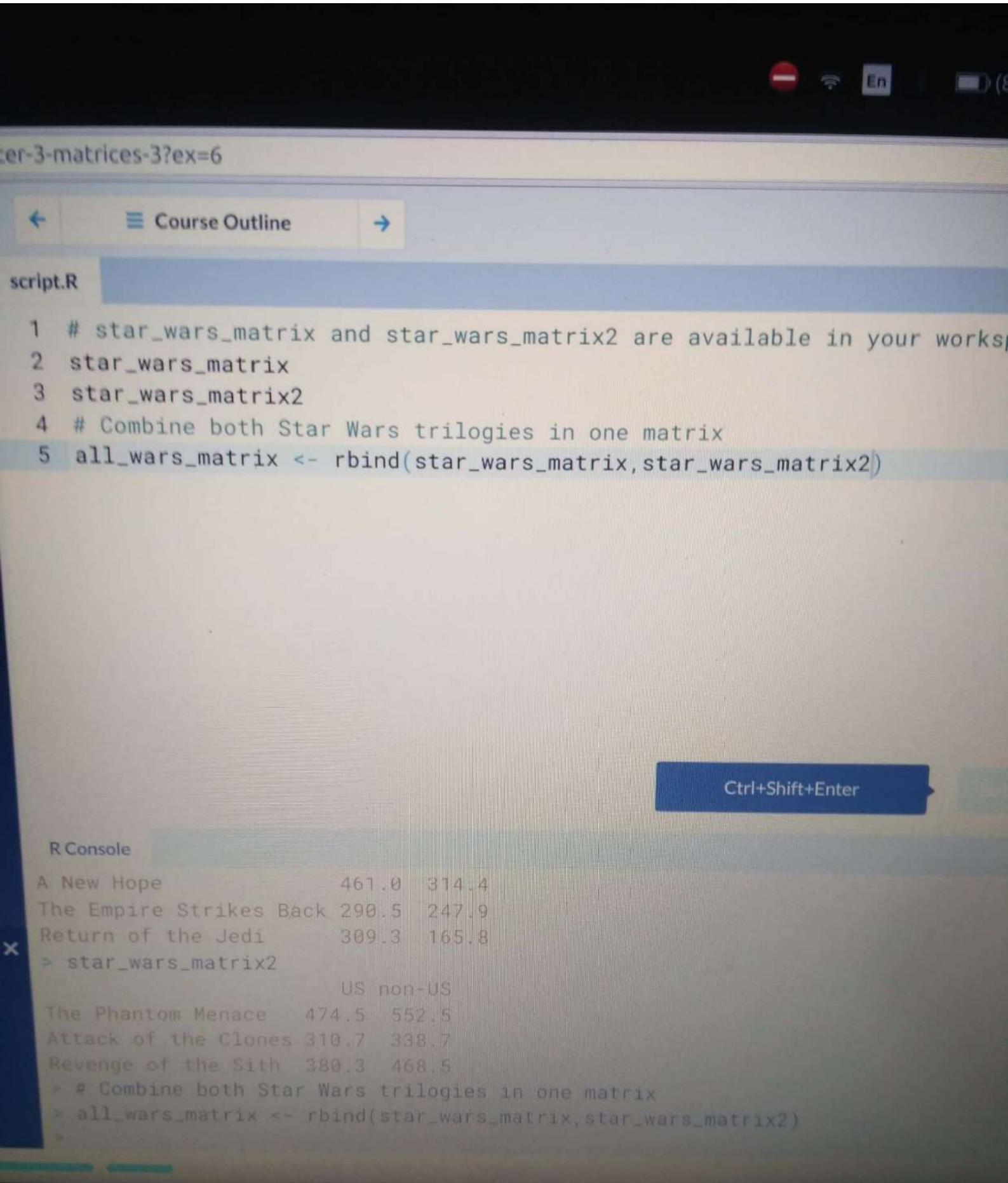
Instructions

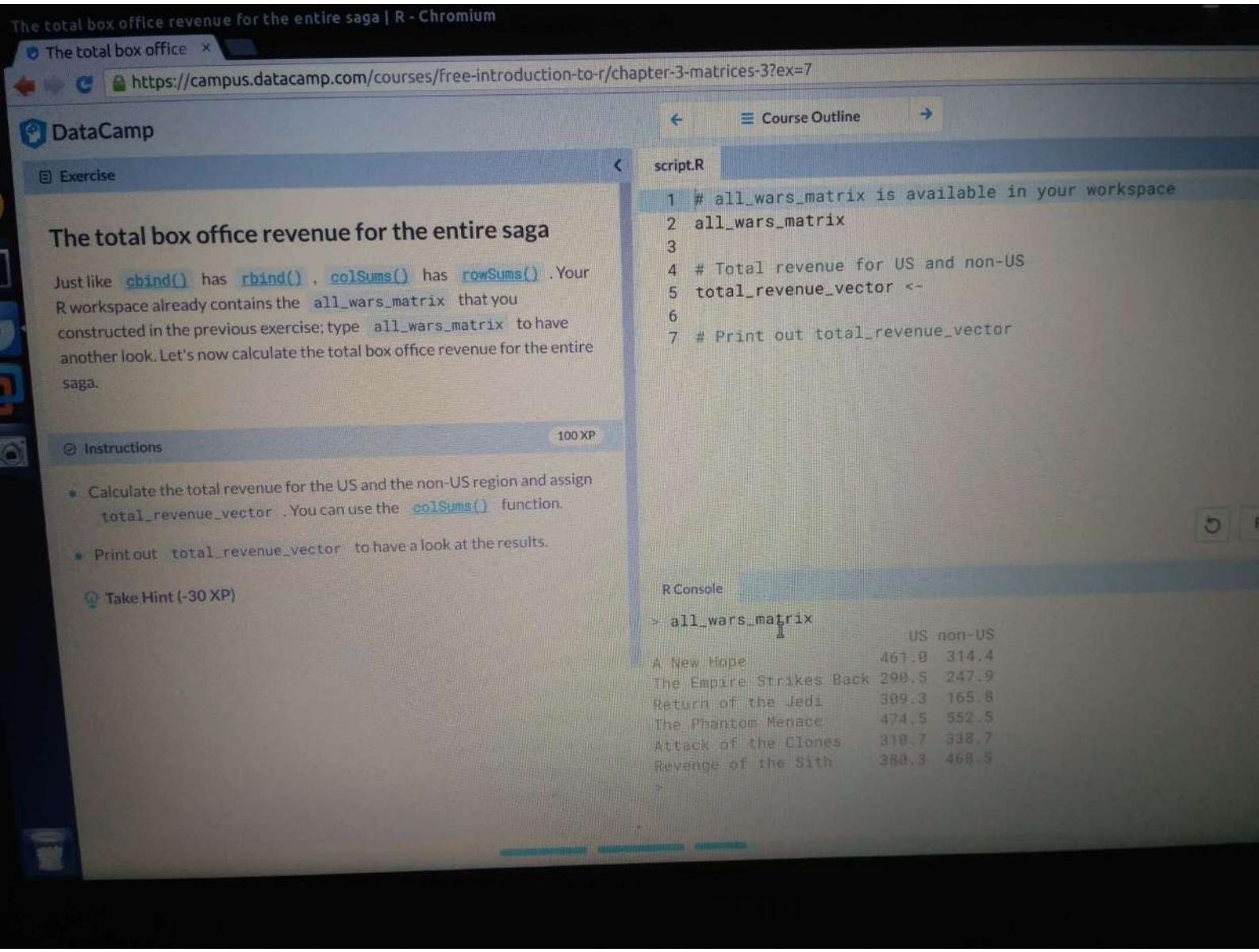
100 XP

Use `rbind()` to paste together `star_wars_matrix` and `star_wars_matrix2`, in this order. Assign the resulting matrix to `all_wars_matrix`.

Take Hint (-30 XP)







Course Outline

script.R

```
1 # all_wars_matrix is available in your workspace
2 all_wars_matrix
3
4 # Total revenue for US and non-US
5 total_revenue_vector <- colSums(all_wars_matrix)
6
7 # Print out total_revenue_vector
8 total_revenue_vector
```

Run all

R Console

```
Attack of the Clones    310.7  338.7
Revenge of the Sith     380.3  468.5
>
> # Total revenue for US and non-US
> total_revenue_vector <- colSums(all_wars_matrix)
>
> # Print out total_revenue_vector
> total_revenue_vector
      US non-US
  2226.3 2087.8
>
```

Selection of matrix elements

Similar to vectors, you can use the square brackets `[ ]` to select one or multiple elements from a matrix. Whereas vectors have one dimension, matrices have two dimensions. You should therefore use a comma to separate the rows you want to select from the columns. For example:

- `my_matrix[1,2]` selects the element at the first row and second column.
- `my_matrix[1:3,2:4]` results in a matrix with the data on the rows 1, 2, 3 and columns 2, 3, 4.

If you want to select all elements of a row or a column, no number is needed before or after the comma, respectively:

- `my_matrix[,1]` selects all elements of the first column.
- `my_matrix[1,]` selects all elements of the first row.

Back to Star Wars with this newly acquired knowledge! As in the previous exercise, `all_wars_matrix` is already available in your workspace.

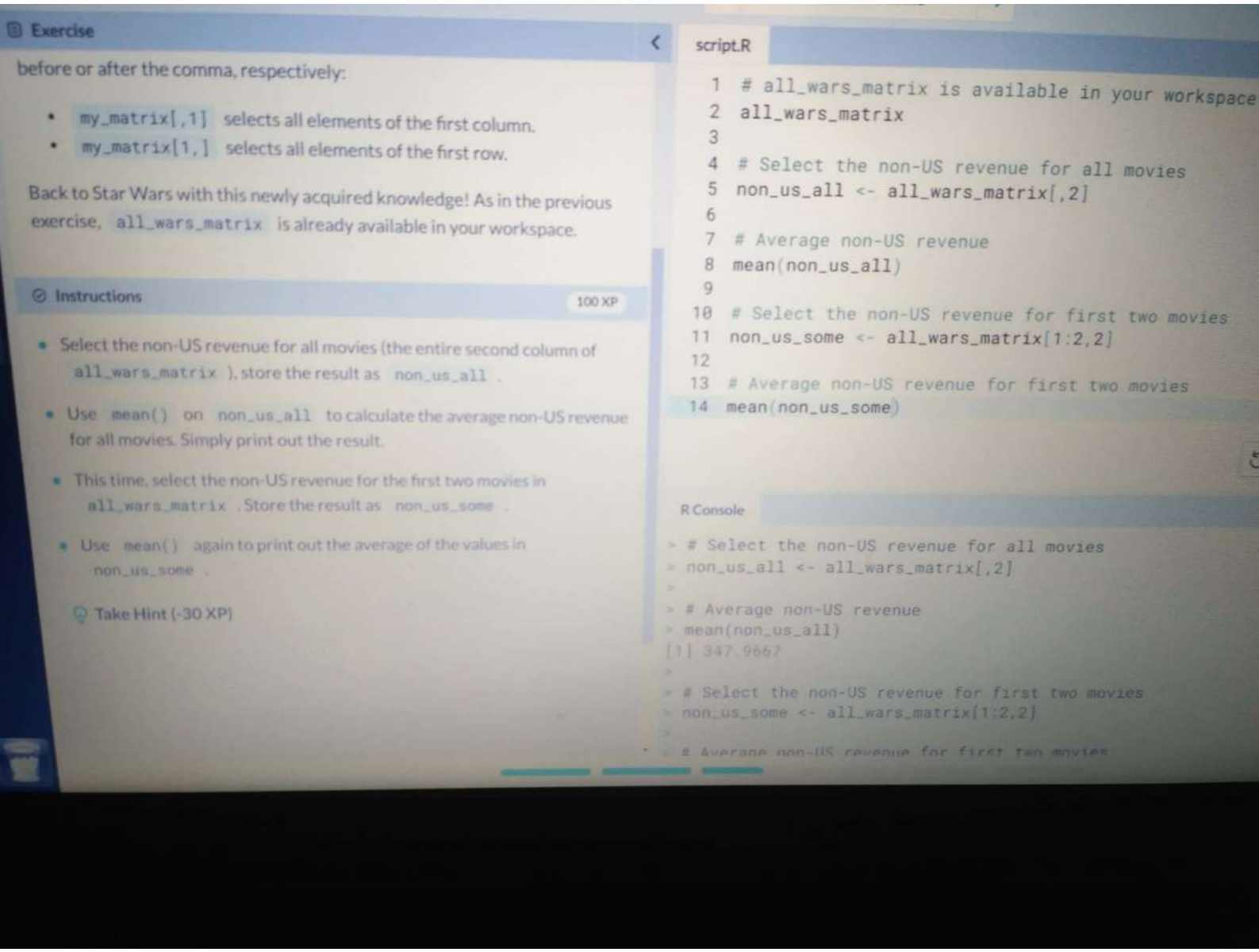
Instructions 100 XP

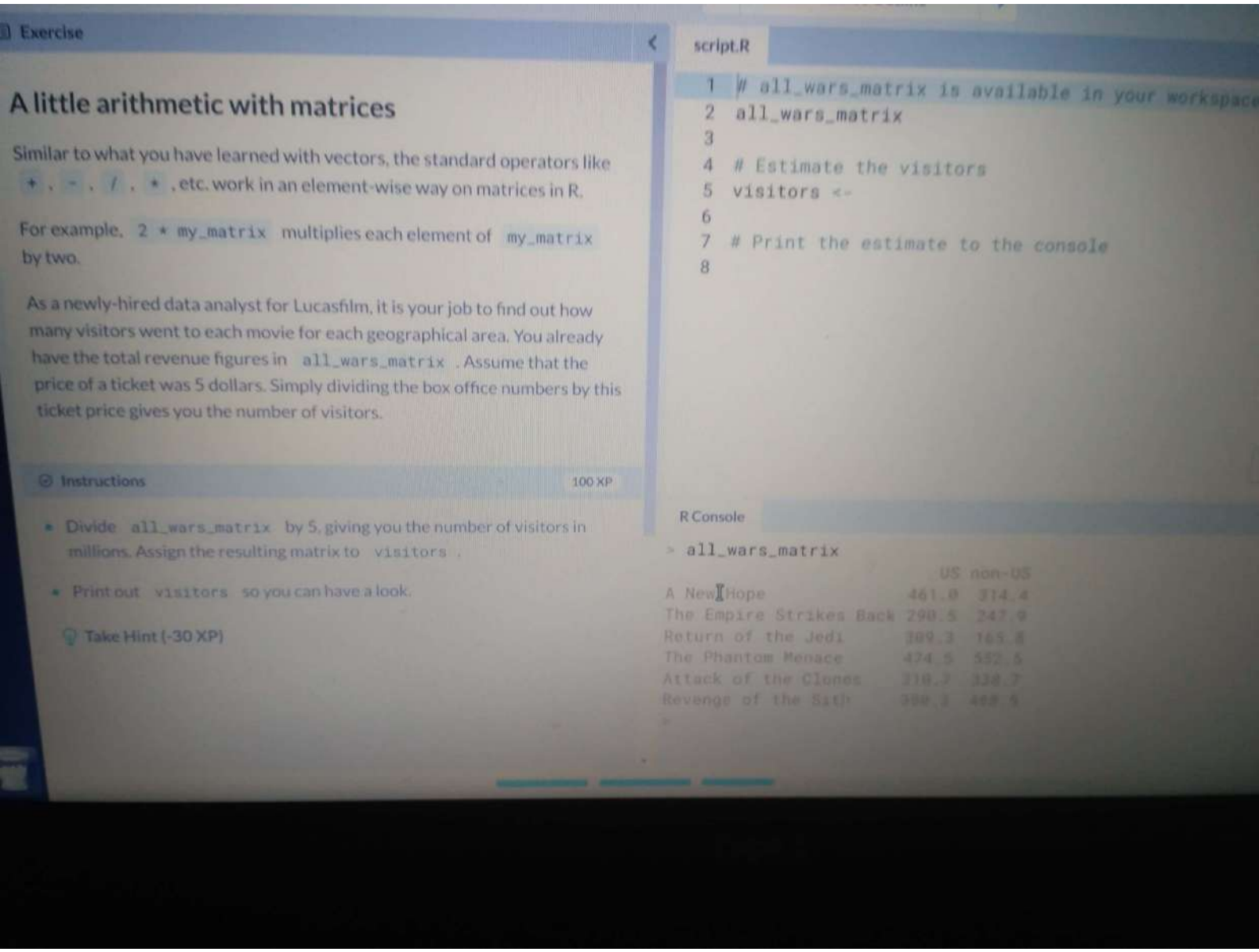
- Select the non-US revenue for all movies (the entire second column of `all_wars_matrix`), store the result as `non_us_all`.
- Use `mean()` on `non_us_all` to calculate the average non-US revenue for all movies. Simply print out the result.

```
1  # all_wars_matrix i
2  all_wars_matrix
3
4  # Select the non-US
5  non_us_all <-
6
7  # Average non-US reve
8
9
10 # Select the non-US r
11 non_us_some <-
12
13 # Average non-US reven
14
```

R Console

```
> all_wars_matrix
                      US
A New Hope           461.0
The Empire Strikes Back 290.5
Return of the Jedi    309.3
The Phantom Menace    474.5
Attack of the Clones  310.7
Revenge of the Sith   380.3
>
```





d operators like
natrices in R.

of my_matrix

to find out how
area. You already
ssume that the
fice numbers by this

100 XP

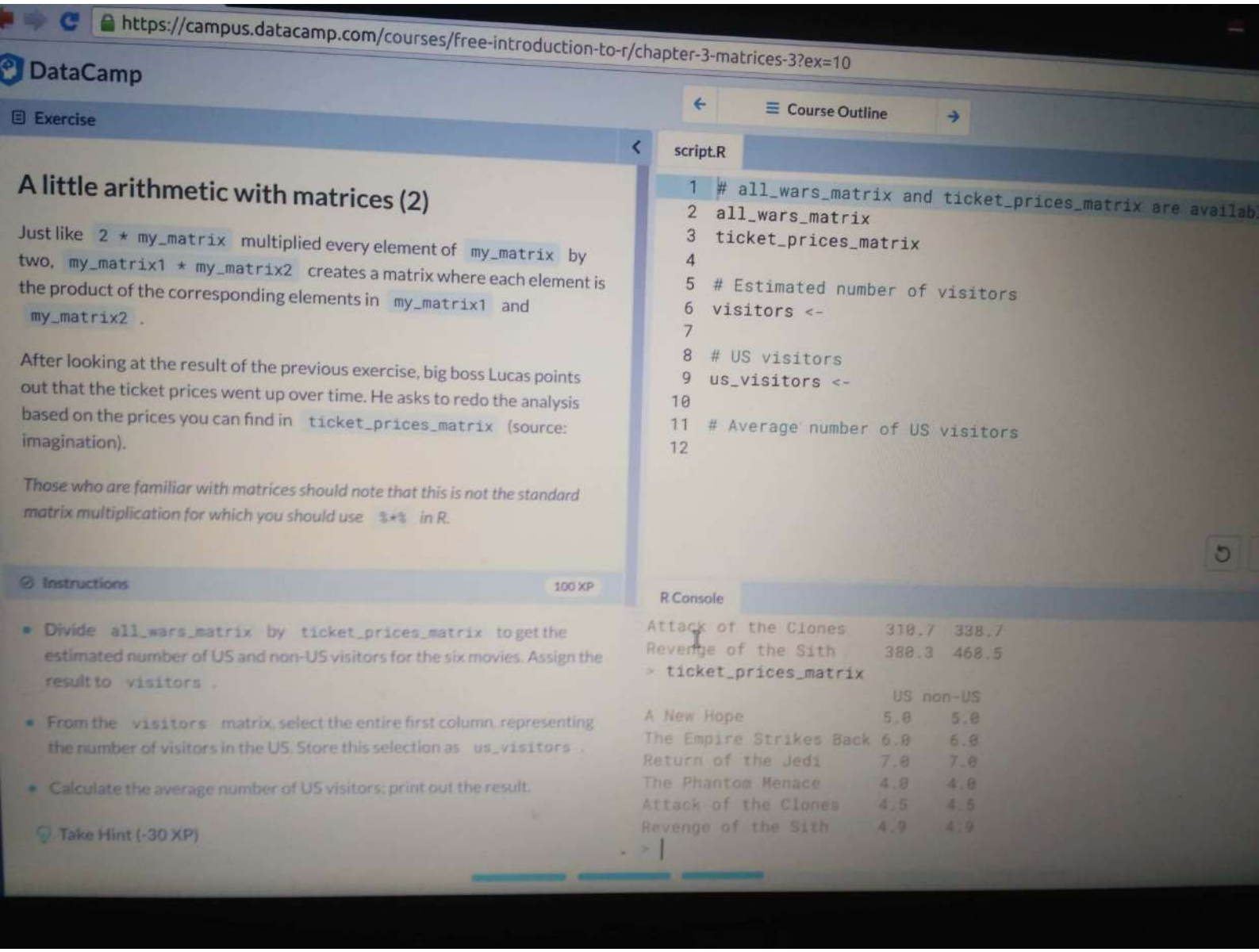
nber of visitors in

```
1 # all_wars_matrix is available in your workspace
2 all_wars_matrix
3
4 # Estimate the visitors
5 visitors <- (all_wars_matrix)/5
6
7 # Print the estimate to the console
8 visitors
```

Run Code

R Console

```
> visitors <- (all_wars_matrix)/5
>
> # Print the estimate to the console
> visitors
                          US non-US
A New Hope              92.20  62.88
The Empire Strikes Back 58.10  49.58
Return of the Jedi      61.86  33.16
The Phantom Menace      94.90 110.50
Attack of the Clones    62.14  67.74
```



matrices-3?ex=10

Course Outline

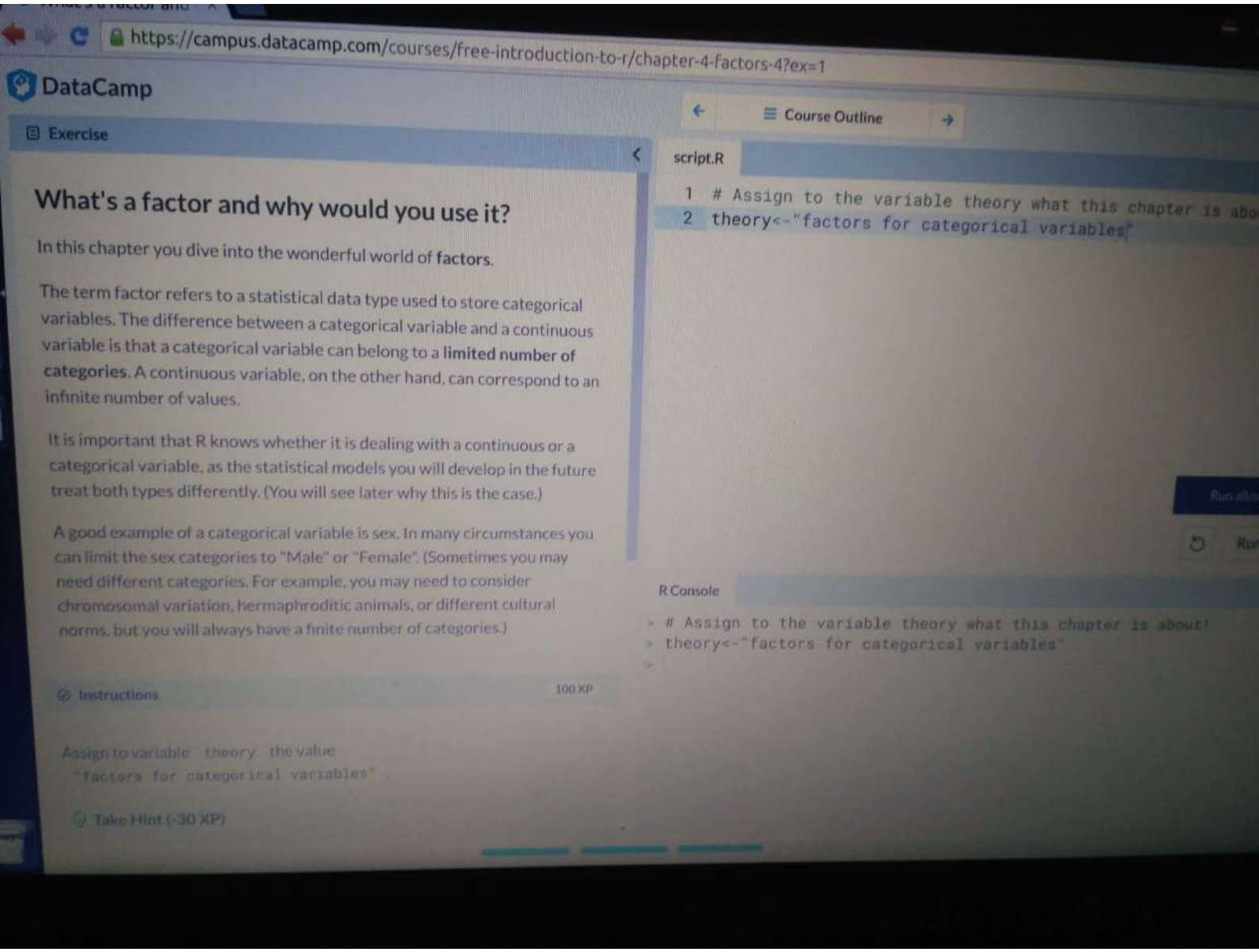
script.R

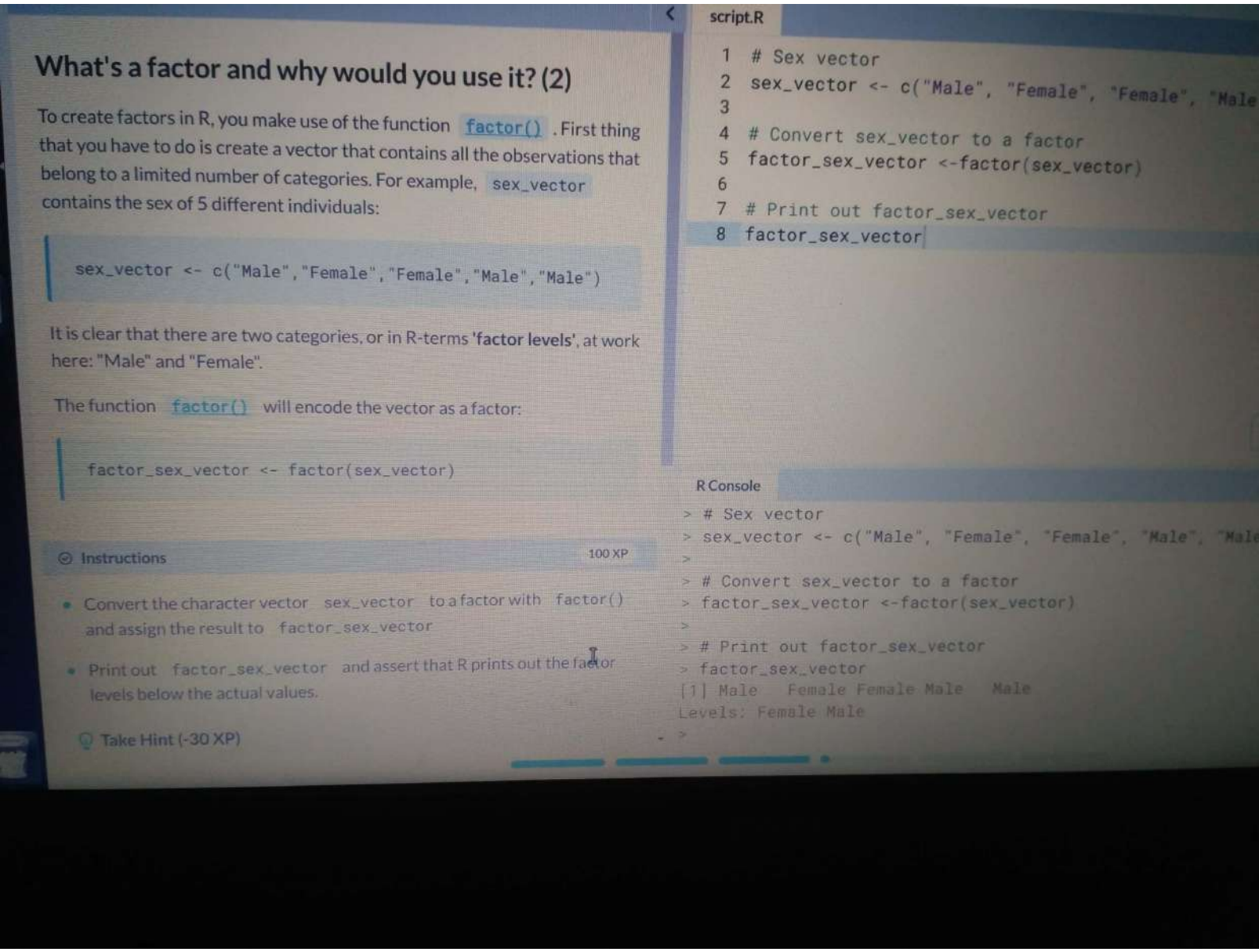
```
1  # all_wars_matrix and ticket_prices_matrix are available in you
2  all_wars_matrix
3  ticket_prices_matrix
4
5  # Estimated number of visitors
6  visitors <- all_wars_matrix/ticket_prices_matrix
7
8  # US visitors
9  us_visitors <- visitors[,1]
10
11 # Average number of US visitors
12 mean(us_visitors)
13 # neeraj:printing result
14 us_visitors
```

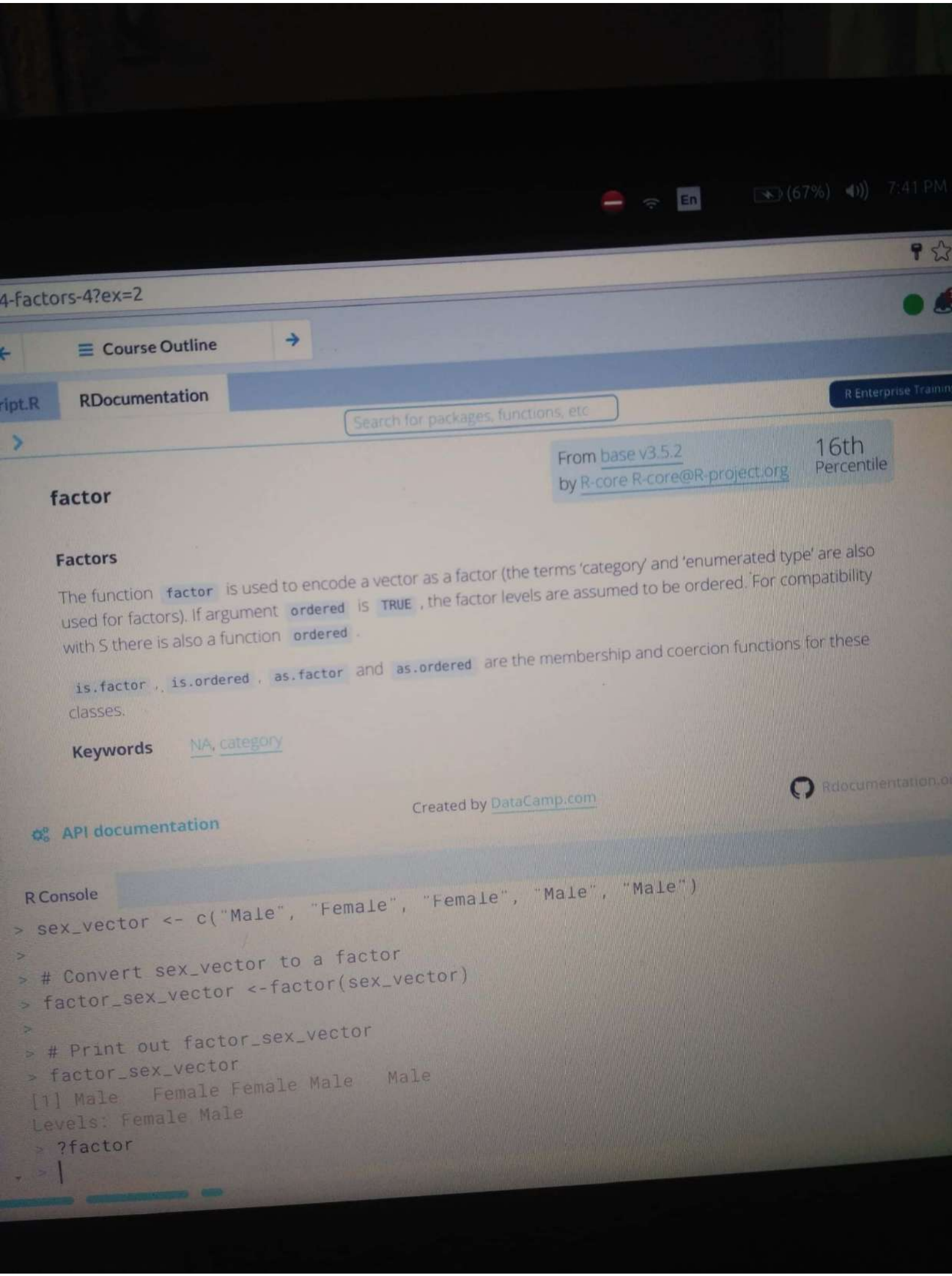
Run Code

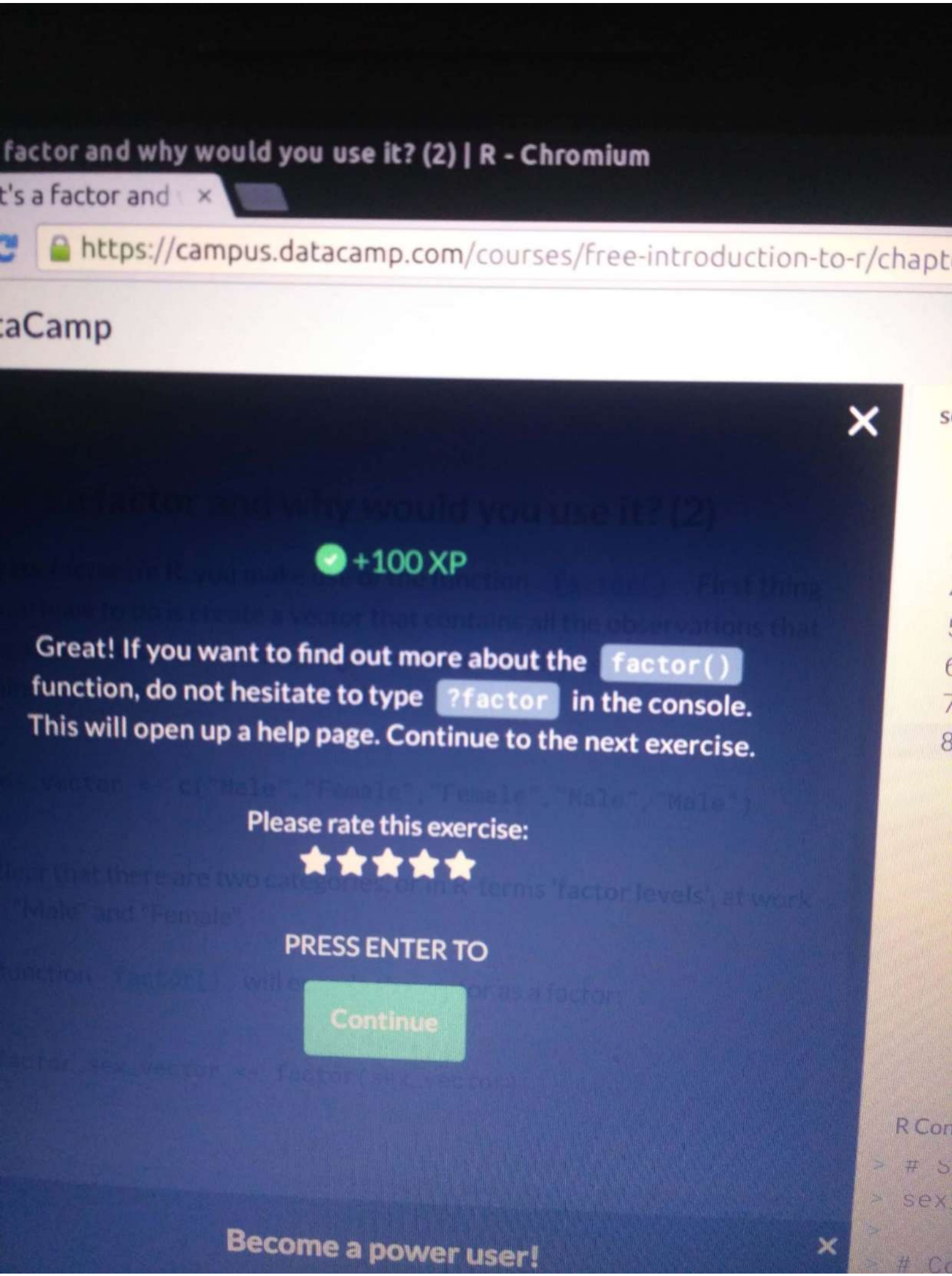
R Console

```
> us_visitors <- visitors[,1]
>
> # Average number of US visitors
> mean(us_visitors)
[1] 75.01401
> # neeraj:printing result
> us_visitors
          A New Hope The Empire Strikes Back      Return of the Jedi
            92.20000                48.41667                44.18571
The Phantom Menace     Attack of the Clones     Revenge of the Sith
           118.62500                69.04444                77.61224
```









DataCamp

Exercise

What's a factor and why would you use it? (3)

There are two types of categorical variables: a **nominal categorical variable** and an **ordinal categorical variable**.

A nominal variable is a categorical variable without an implied order. This means that it is impossible to say that 'one is worth more than the other'. For example, think of the categorical variable `animals_vector` with the categories `"Elephant"`, `"Giraffe"`, `"Donkey"` and `"Horse"`. Here, it is impossible to say that one stands above or below the other. (Note that some of you might disagree ;-)).

In contrast, ordinal variables do have a natural ordering. Consider for example the categorical variable `temperature_vector` with the categories: `"Low"`, `"Medium"` and `"High"`. Here it is obvious that `"Medium"` stands above `"Low"`, and `"High"` stands above `"Medium"`.

Instructions 100 XP

Click 'Submit Answer' to check how R constructs and prints nominal and ordinal variables. Do not worry if you do not understand all the code just yet, we will get to that.

Take Hint (-30 XP)

ied order. This
an the other'.
ector with the
"Horse" .
the other. (Note

onsider for
with the
s obvious that
above

100 XP

s nominal and
ll the code just yet.

```
2 animals_vector <- c("Elephant", "Giraffe", "Donkey", "Horse")
3 factor_animals_vector <- factor(animals_vector)
4 factor_animals_vector
5
6 # Temperature
7 temperature_vector <- c("High", "Low", "High","Low", "Medium")
8 factor_temperature_vector <- factor(temperature_vector, order = TRUE, levels
  ("Low", "Medium", "High"))
9 factor_temperature_vector
```

Run Code | Submit Ans

R Console

```
> factor_animals_vector
[1] Elephant Giraffe  Donkey   Horse
Levels: Donkey Elephant Giraffe Horse
>
> # Temperature
> temperature_vector <- c("High", "Low", "High","Low", "Medium")
> factor_temperature_vector <- factor(temperature_vector, order = TRUE, levels = c
    ("Low", "Medium", "High"))
> factor_temperature_vector
[1] High   Low    High   Low    Medium
Levels: Low < Medium < High
```

DataCamp

Exercise

Factor levels

When you first get a data set, you will often notice that it contains factors with specific factor levels. However, sometimes you will want to change the names of these levels for clarity or other reasons. R allows you to do this with the function `levels()` :

```
levels(factor_vector) <- c("name1", "name2",...)
```

A good illustration is the raw data that is provided to you by a survey. A common question for every questionnaire is the sex of the respondent. Here, for simplicity, just two categories were recorded, `"M"` and `"F"` . (You usually need more categories for survey data; either way, you use a factor to store the categorical data.)

```
survey_vector <- c("M", "F", "F", "M", "M")
```

Recording the sex with the abbreviations `"M"` and `"F"` can be convenient if you are collecting data with pen and paper, but it can introduce confusion when analyzing the data. At that point, you will often want to change the factor levels to `"Male"` and `"Female"` instead of `"M"` and `"F"` for clarity.

Watch out: the order with which you assign the levels is important. If you type `levels(factor_survey_vector)` , you'll see that it outputs

Instructions

100 XP

DataCamp

Exercise

Recording the sex with the abbreviations `"M"` and `"F"` can be convenient if you are collecting data with pen and paper, but it can introduce confusion when analyzing the data. At that point, you will often want to change the factor levels to `"Male"` and `"Female"` instead of `"M"` and `"F"` for clarity.

Watch out: the order with which you assign the levels is important. If you type `levels(factor_survey_vector)`, you'll see that it outputs `[1] "F" "M"`. If you don't specify the levels of the factor when creating the vector, `R` will automatically assign them alphabetically. To correctly map `"F"` to `"Female"` and `"M"` to `"Male"`, the levels should be set to `c("Female", "Male")`, in this order.

Instructions

100 XP

- Check out the code that builds a factor vector from `survey_vector`. You should use `factor_survey_vector` in the next instruction.
- Change the factor levels of `factor_survey_vector` to `c("Female", "Male")`. Mind the order of the vector elements here.

Take Hint (-30 XP)

chapter-4-factors-4?ex=4

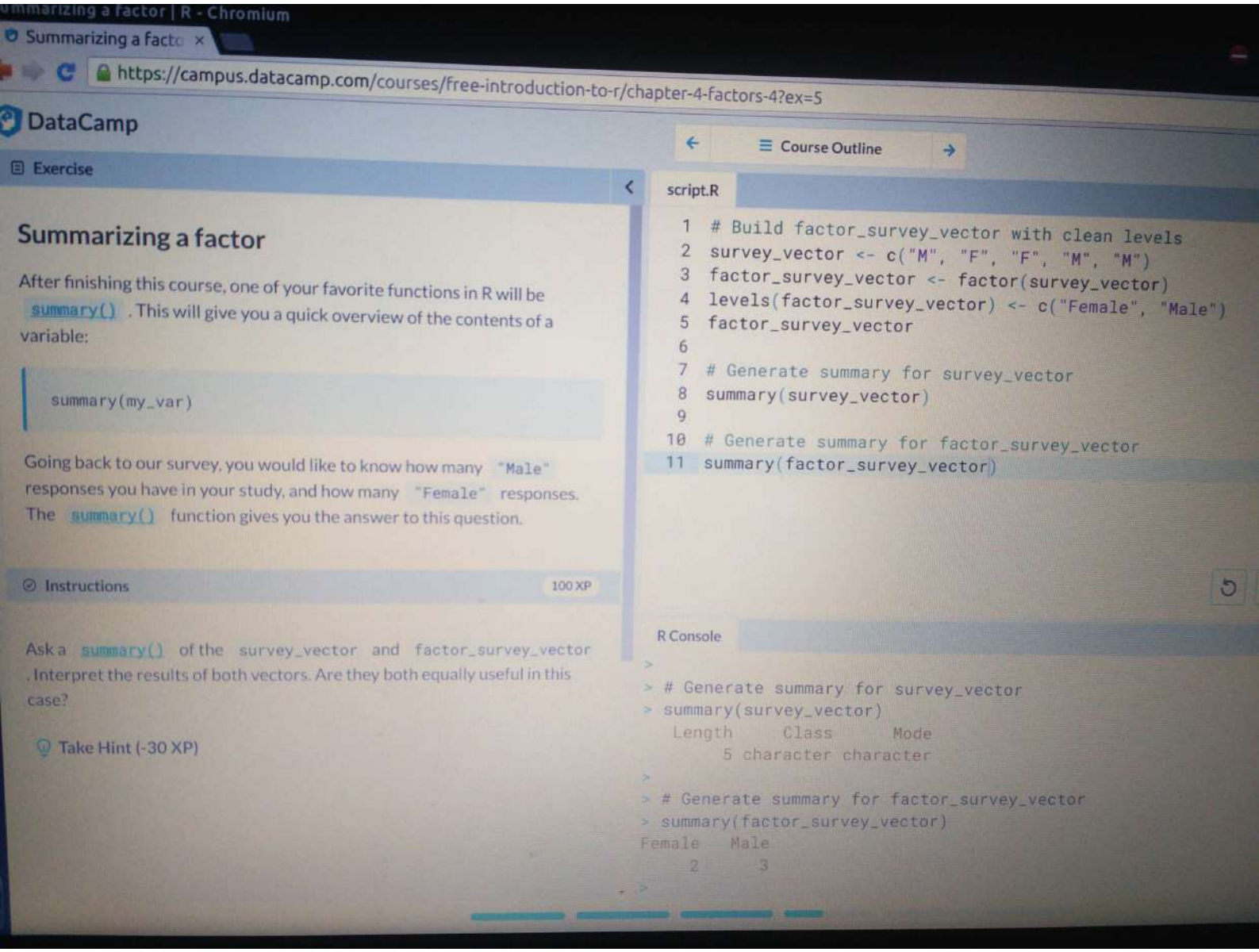
Course Outline

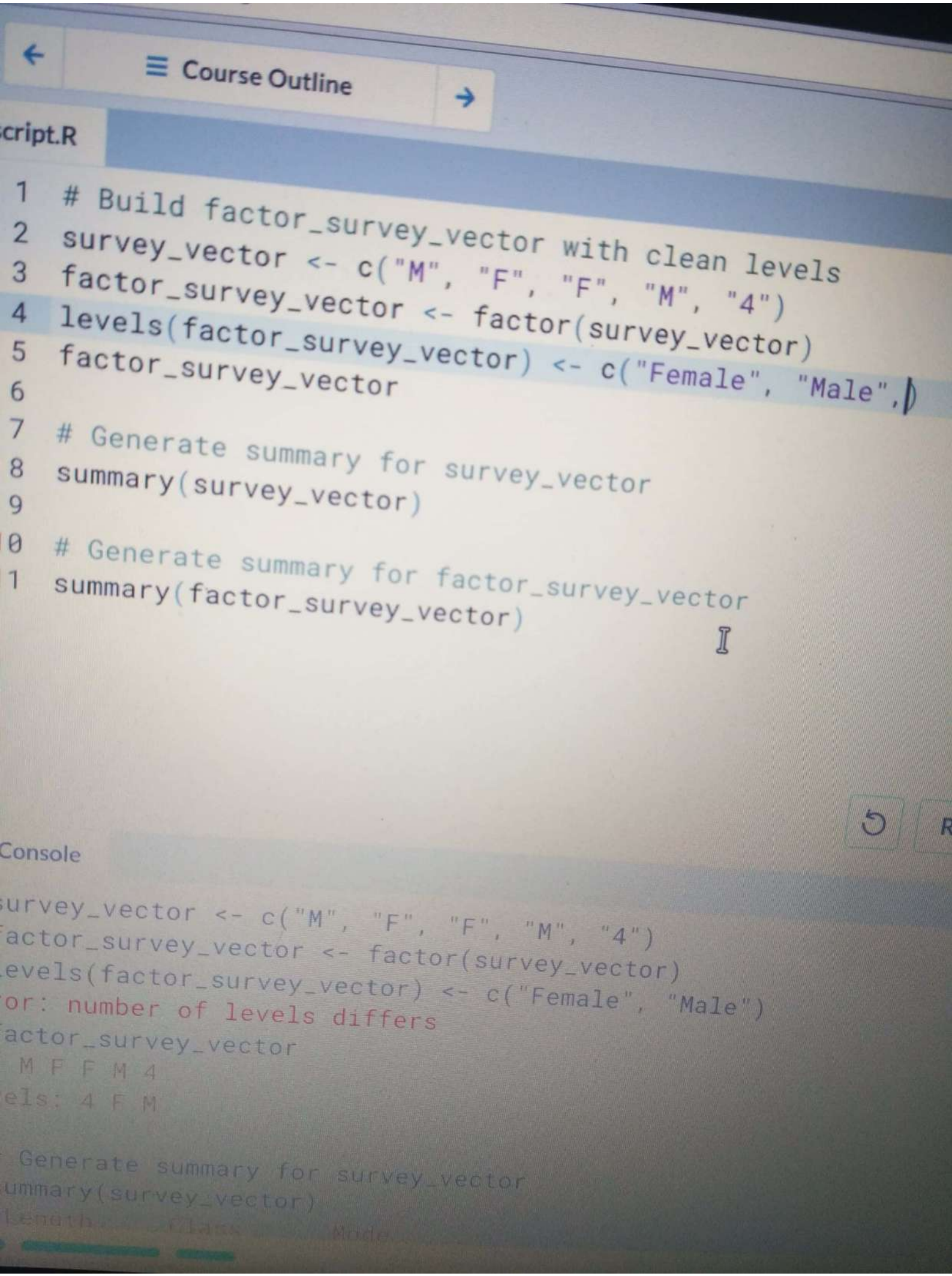
script.R

```r
# Code to build factor_survey_vector
survey_vector <- c("M", "F", "F", "M", "M")
factor_survey_vector <- factor(survey_vector)

# Specify the levels of factor_survey_vector
levels(factor_survey_vector) <-c("Female","Male")

factor_survey_vector
```

R Console

```
> # Code to build factor_survey_vector
> survey_vector <- c("M", "F", "F", "M", "M")
> factor_survey_vector <- factor(survey_vector)
>
> # Specify the levels of factor_survey_vector
> levels(factor_survey_vector) <-c("Female","Male")
>
> factor_survey_vector
[1] Male   Female Female Male   Male
Levels: Female Male
```





oter-4-factors-4?ex=5

Course Outline

script.R

```r
# Build factor_survey_vector with clean levels
survey_vector <- c("M", "F", "F", "M", "4")
factor_survey_vector <- factor(survey_vector)
levels(factor_survey_vector) <- c("Female", "Male","44")
factor_survey_vector

# Generate summary for survey_vector
summary(survey_vector)

# Generate summary for factor_survey_vector
summary(factor_survey_vector)
```

Run Cod

R Console

```
> # Build factor_survey_vector with clean levels
> survey_vector <- c("M", "F", "F", "M", "4")
> factor_survey_vector <- factor(survey_vector)
> levels(factor_survey_vector) <- c("Female", "Male","44")
> factor_survey_vector
[1] 44     Male   Male   44     Female
Levels: Female Male 44
>
> # Generate summary for survey_vector
> summary(survey_vector)
   Length     Class      Mode
```

-r/chapter-4-factors-4?ex=5

Course Outline

script.R

```
1  # Build factor_survey_vector with clean levels
2  survey_vector <- c(43,4)
3  factor_survey_vector <- factor(survey_vector)
4  levels(factor_survey_vector) <- c(2,4)
5  factor_survey_vector
6
7  # Generate summary for survey_vector
8  summary(survey_vector)
9
10 # Generate summary for factor_survey_vector
11 summary(factor_survey_vector)
```

R Console

```
>
> # Generate summary for survey_vector
> summary(survey_vector)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   4.00   13.75   23.50   23.50   33.25   43.00
>
> # Generate summary for factor_survey_vector
> summary(factor_survey_vector)
2 4
1 1
>
```

r/chapter-4-factors-4?ex=6

Course Outline

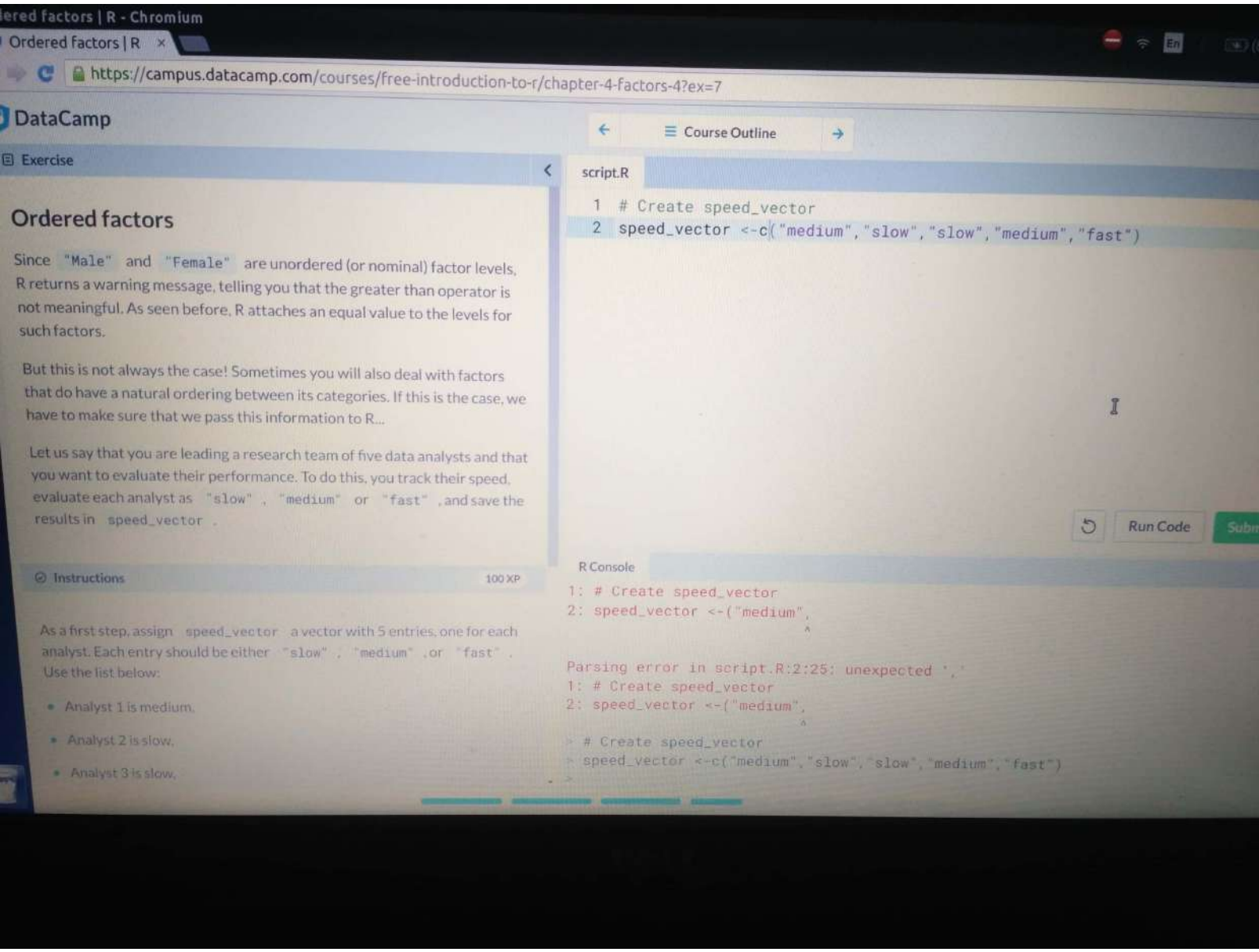
script.R

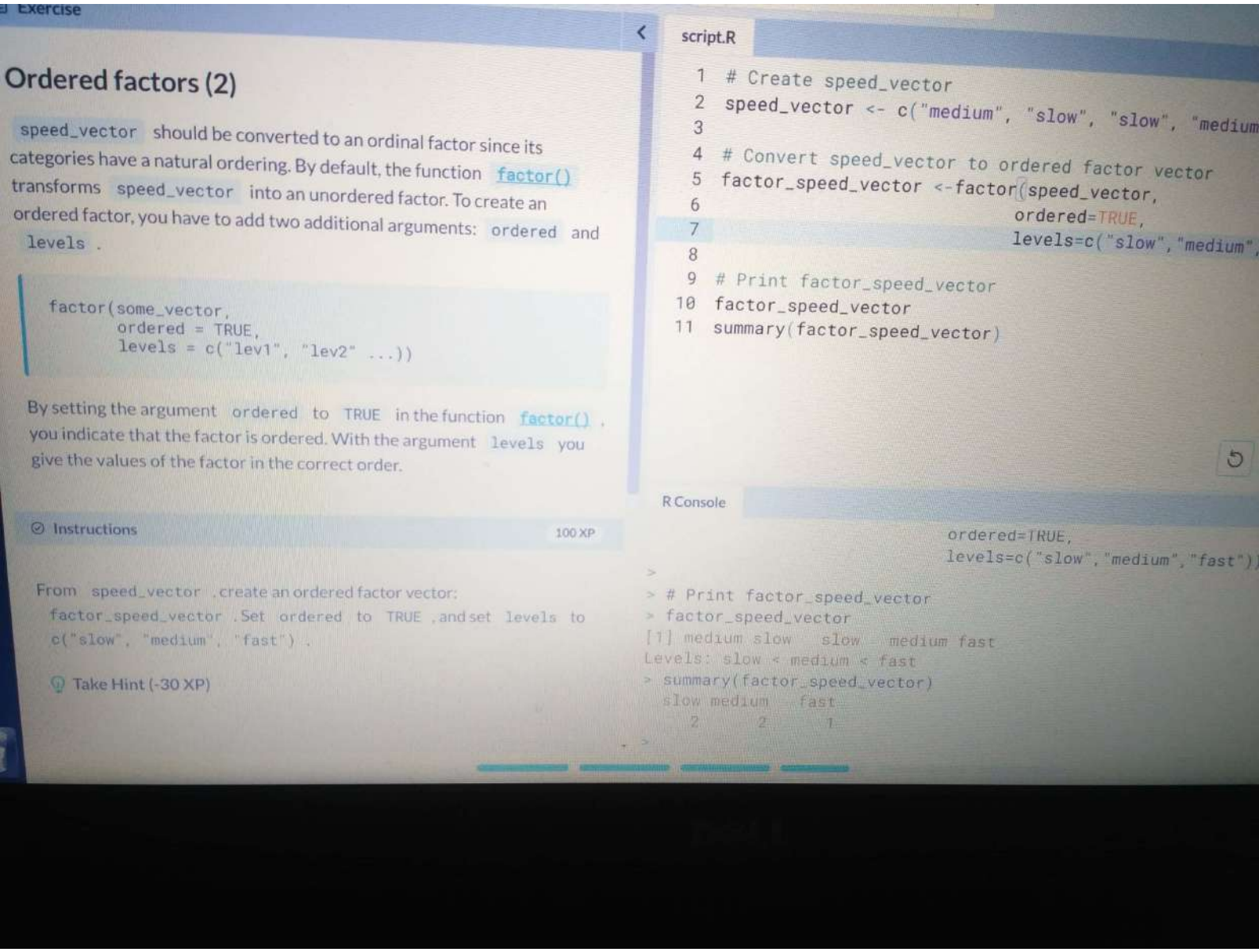
```r
# Build factor_survey_vector with clean levels
survey_vector <- c("M", "F", "F", "M", "M")
factor_survey_vector <- factor(survey_vector)
levels(factor_survey_vector) <- c("Female", "Male")

# Male
male <- factor_survey_vector[1]

#neeraj:male
male

# Female
female <- factor_survey_vector[2]
```

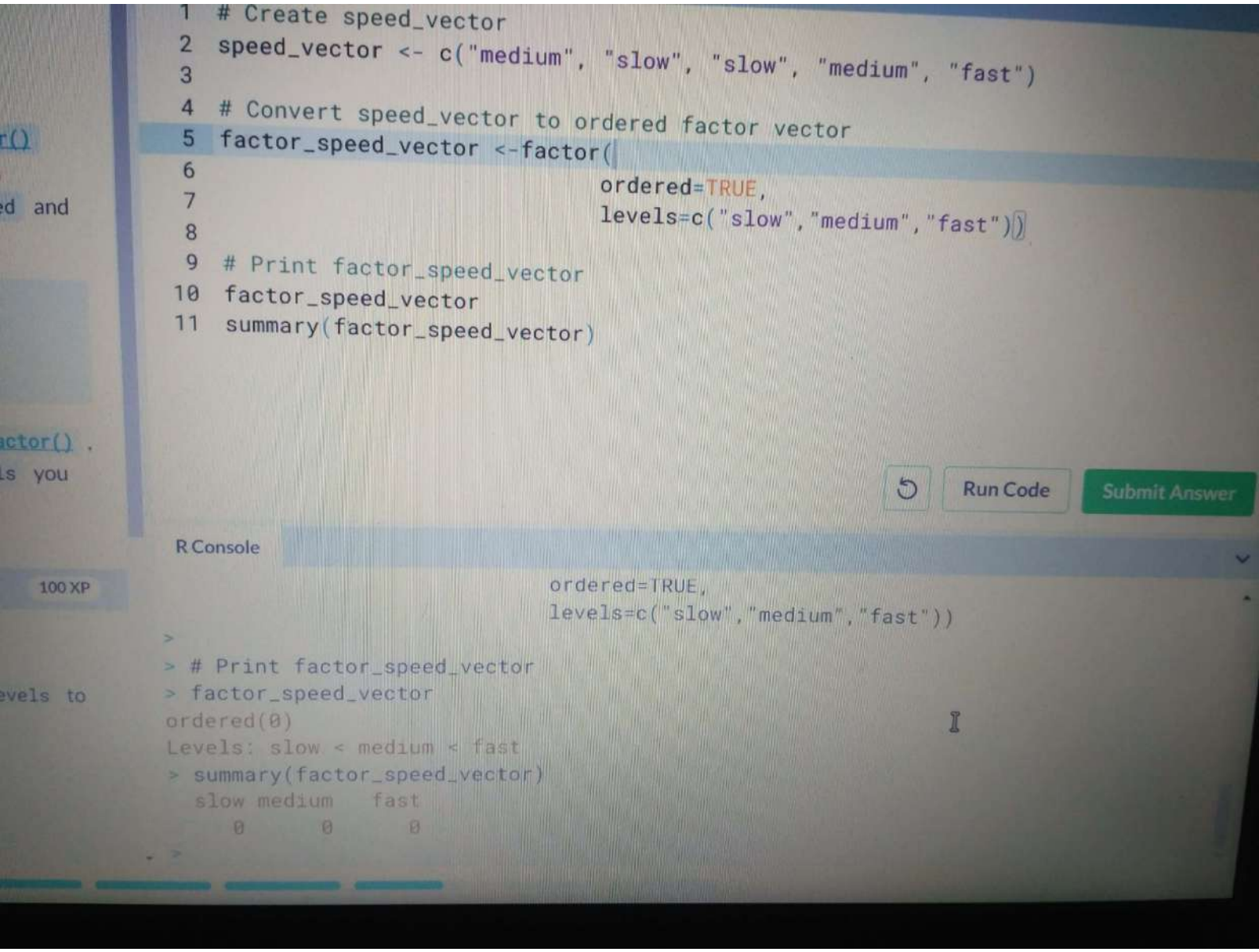
Run Code

R Console

```
> male
[1] Male
Levels: Female Male
>
> # Female
> female <- factor_survey_vector[2]
>
> # Battle of the sexes: Male 'larger' than female?
> male > female
Warning message: '>' not meaningful for factors
[1] NA
```





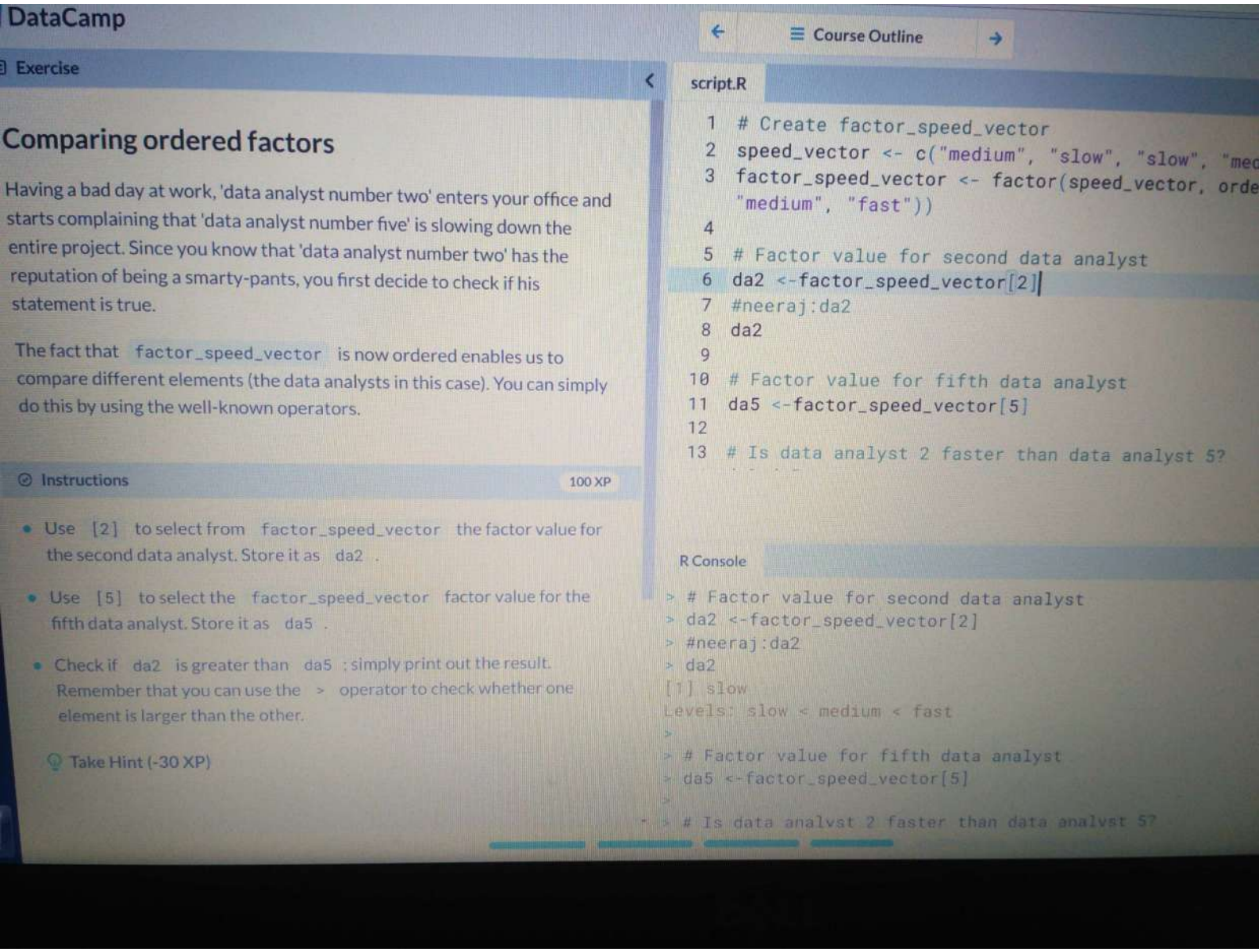


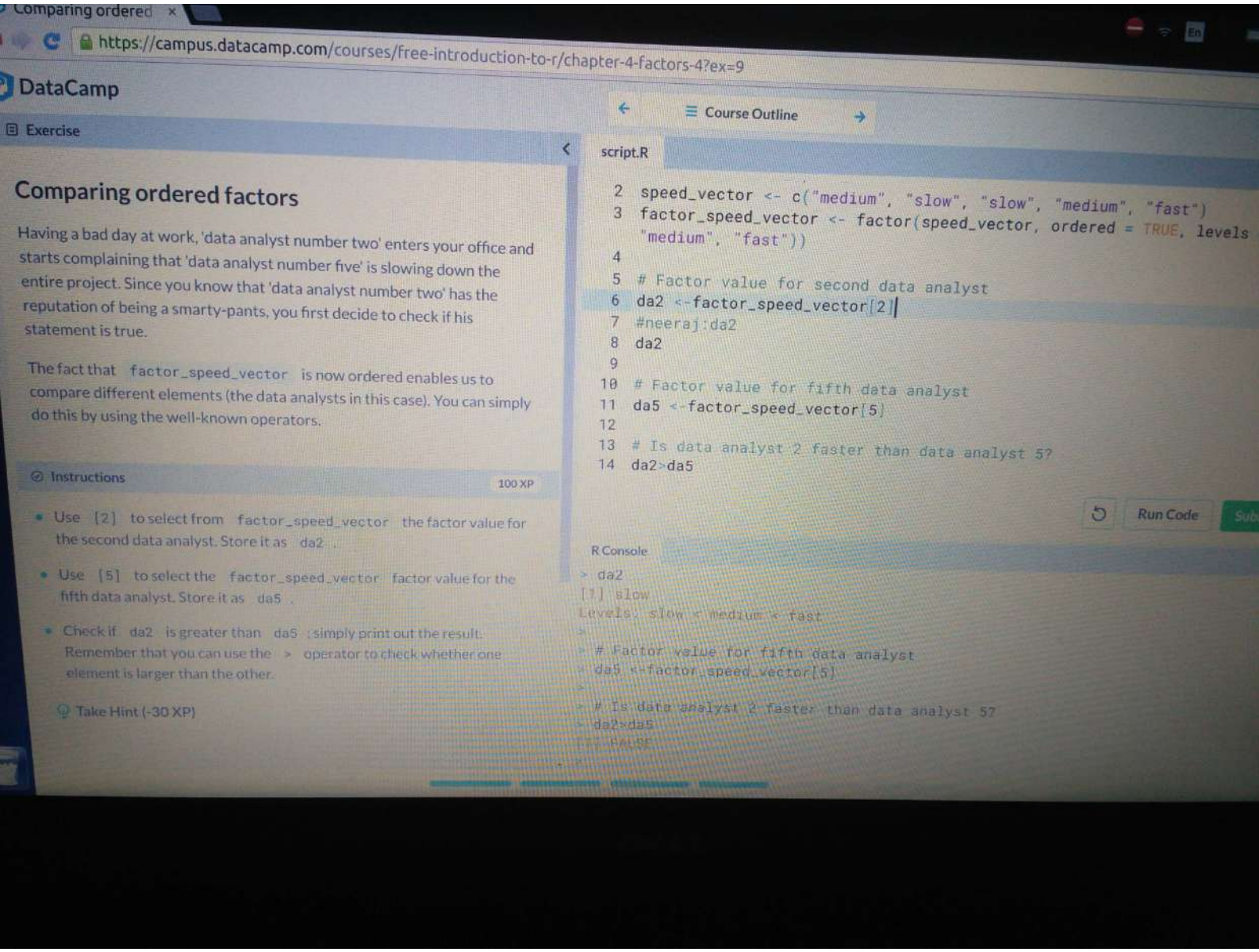
```
4  # Convert speed_vector to ordered factor vector
5  factor_speed_vector <-factor(speed_vector,
6
7                               ordered=FALSE,
8                               levels=c("slow","medium","fast"))
9  # Print factor_speed_vector
10 factor_speed_vector
11 summary(factor_speed_vector)
```

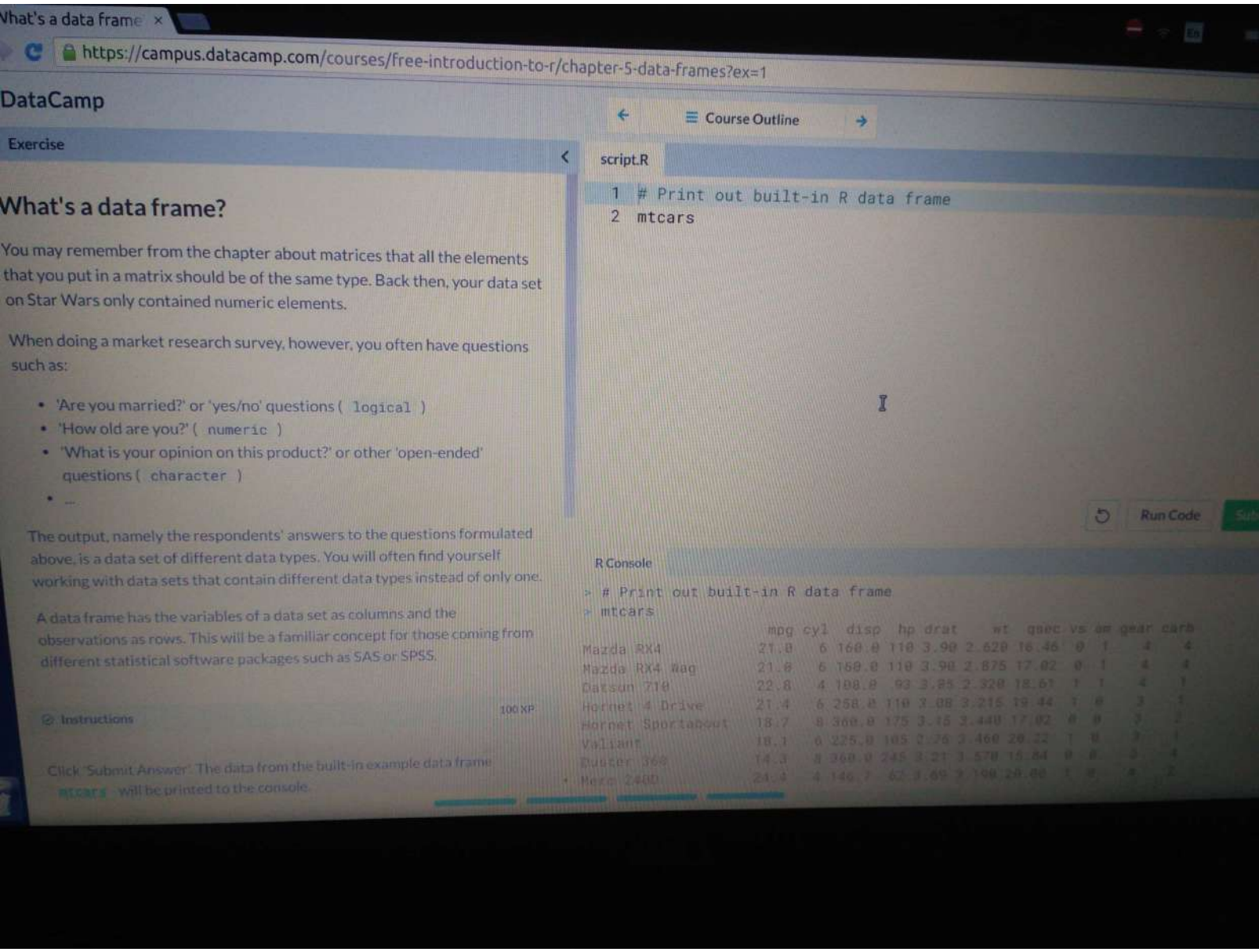
Run Code | Submit Answer

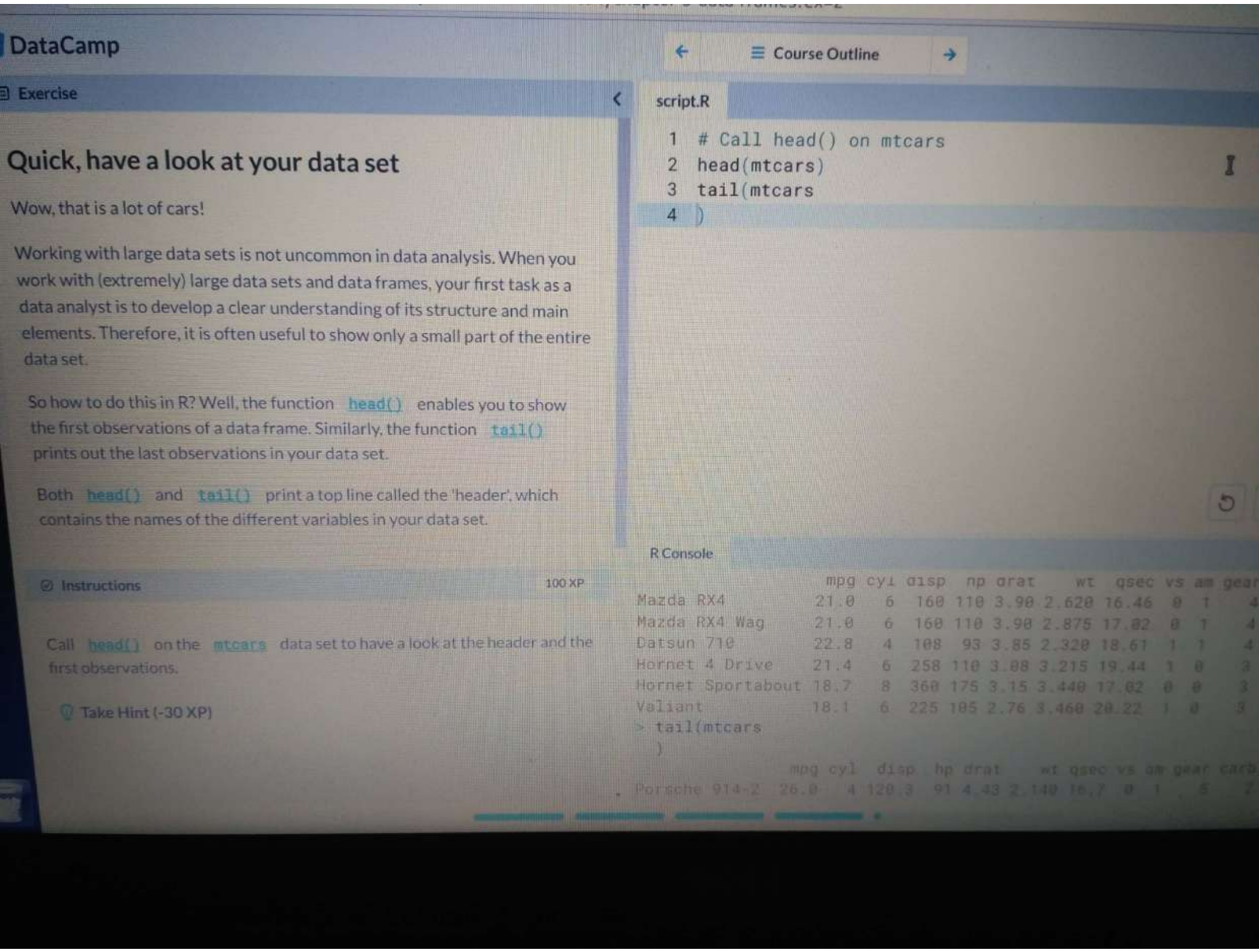
R Console

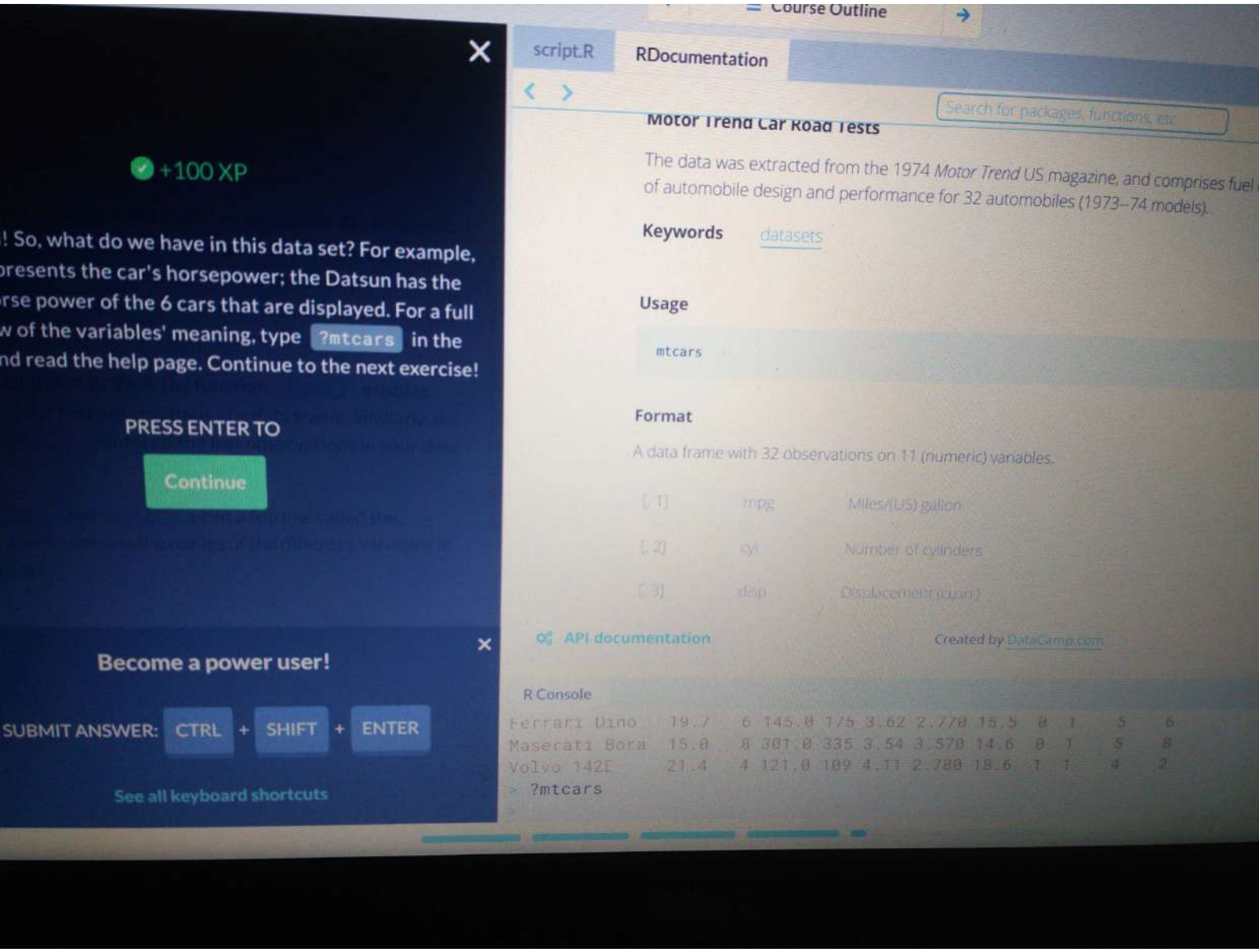
```
                        ordered=FALSE,
                        levels=c("slow","medium","fast"))
>
> # Print factor_speed_vector
> factor_speed_vector
[1] medium slow   slow   medium fast
Levels: slow medium fast
> summary(factor_speed_vector)
  slow medium   fast
     2      2      1
>
```

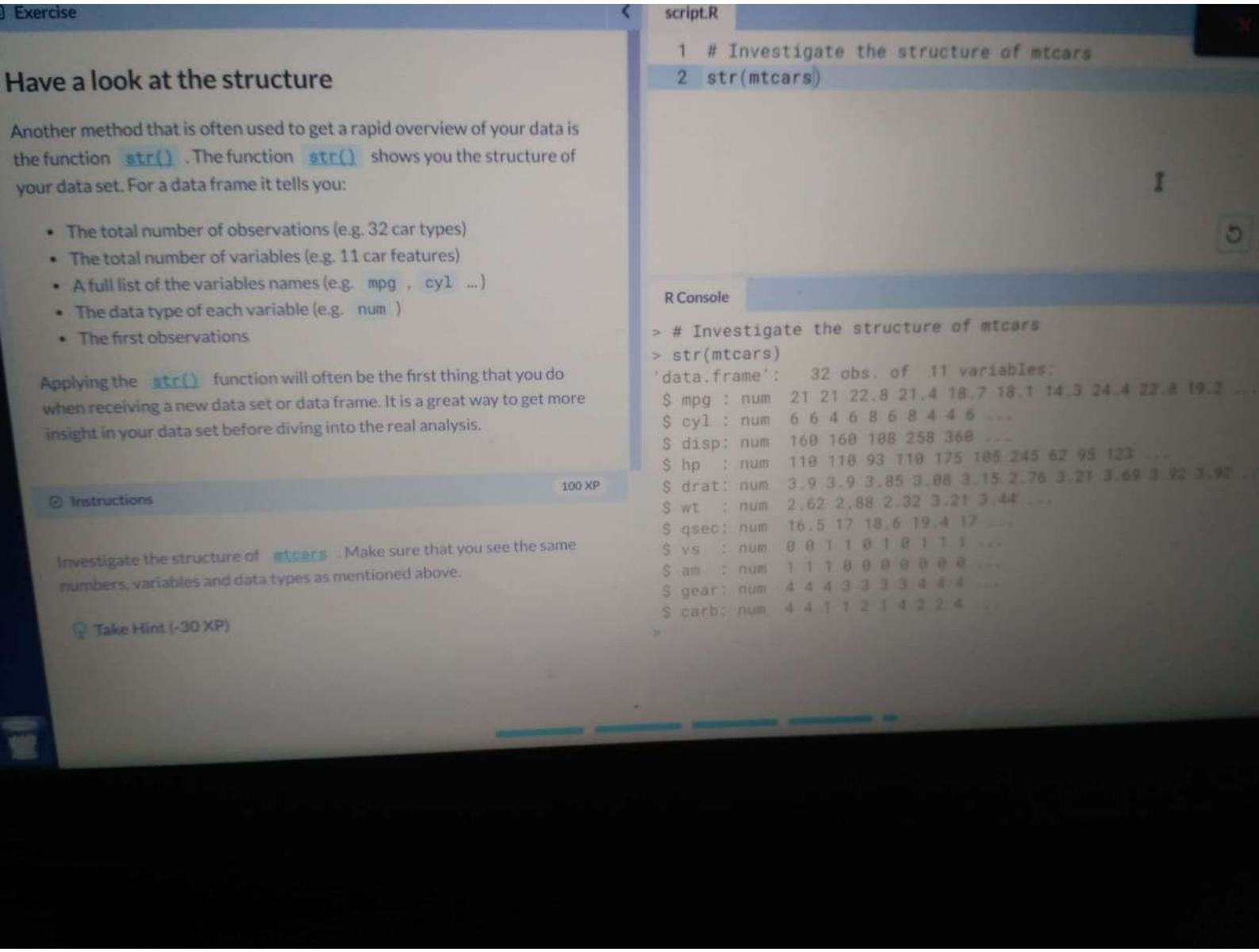


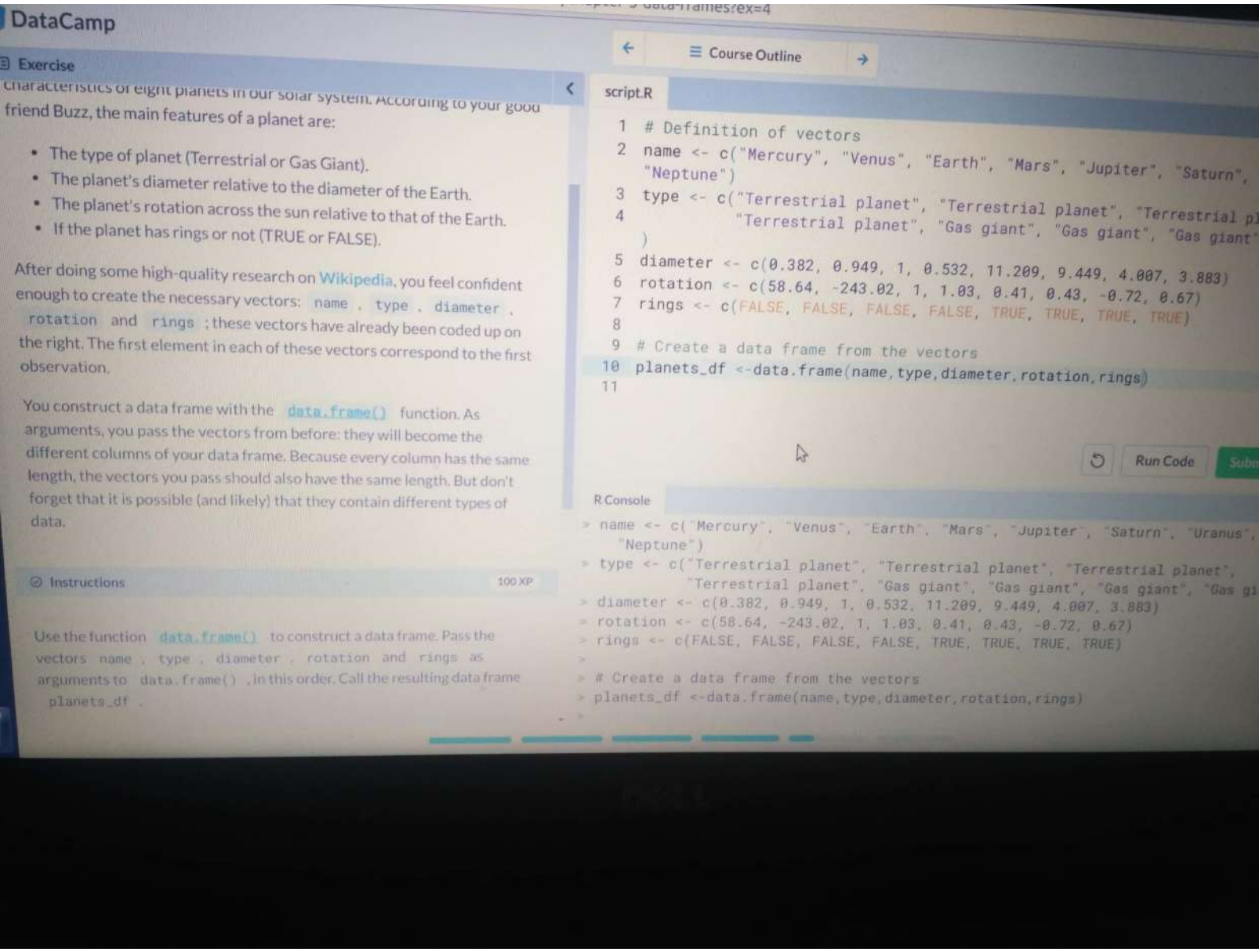


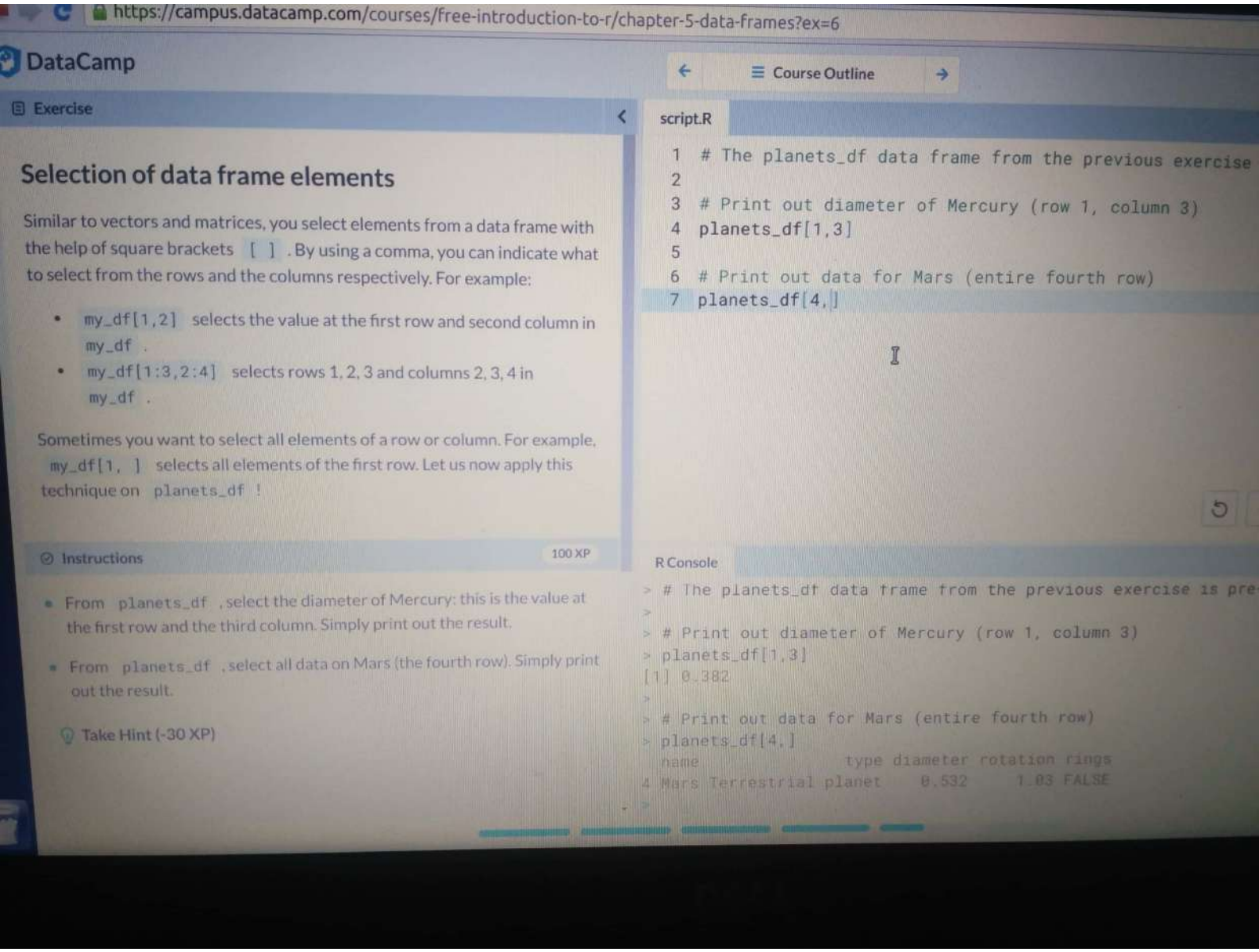


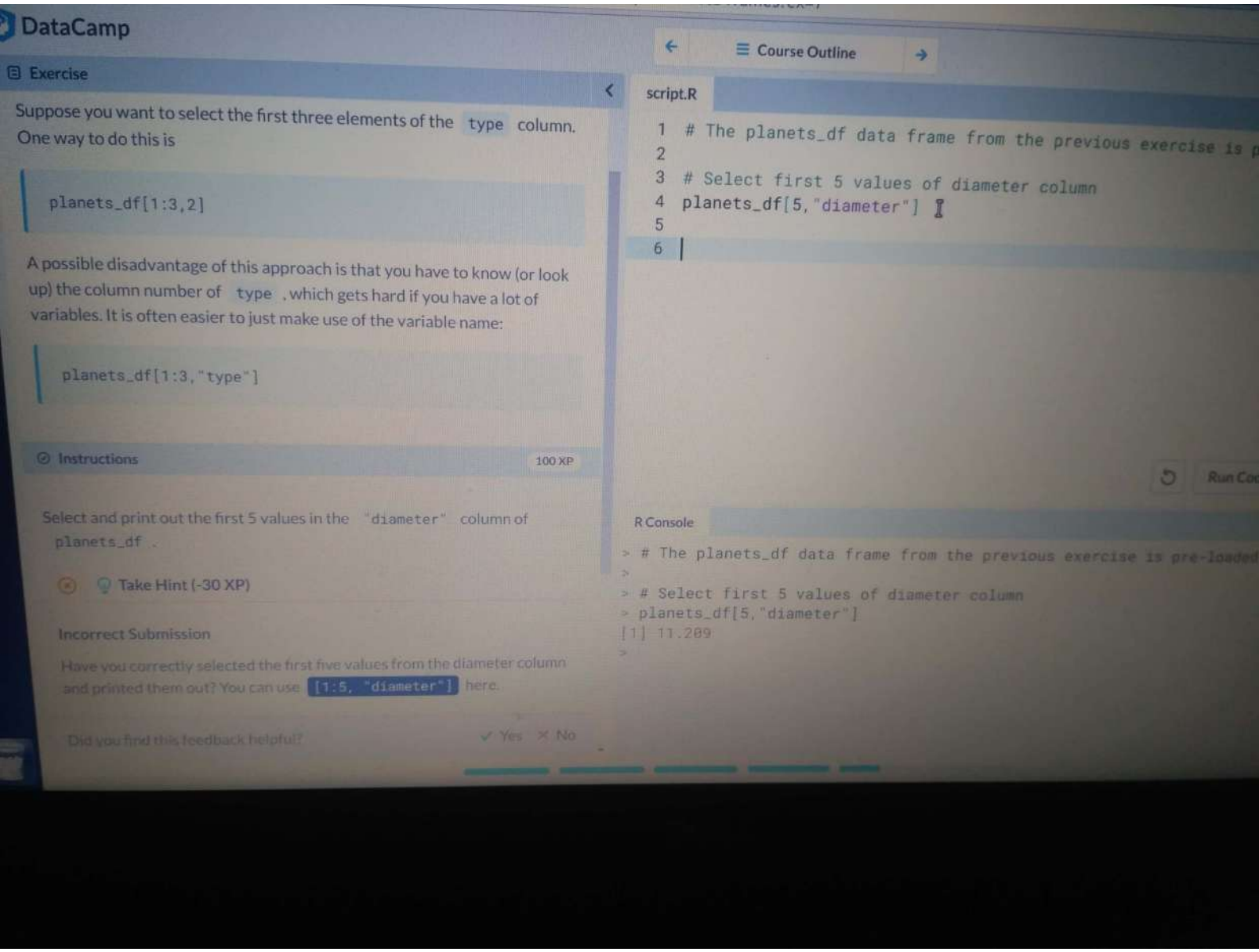












Course Outline

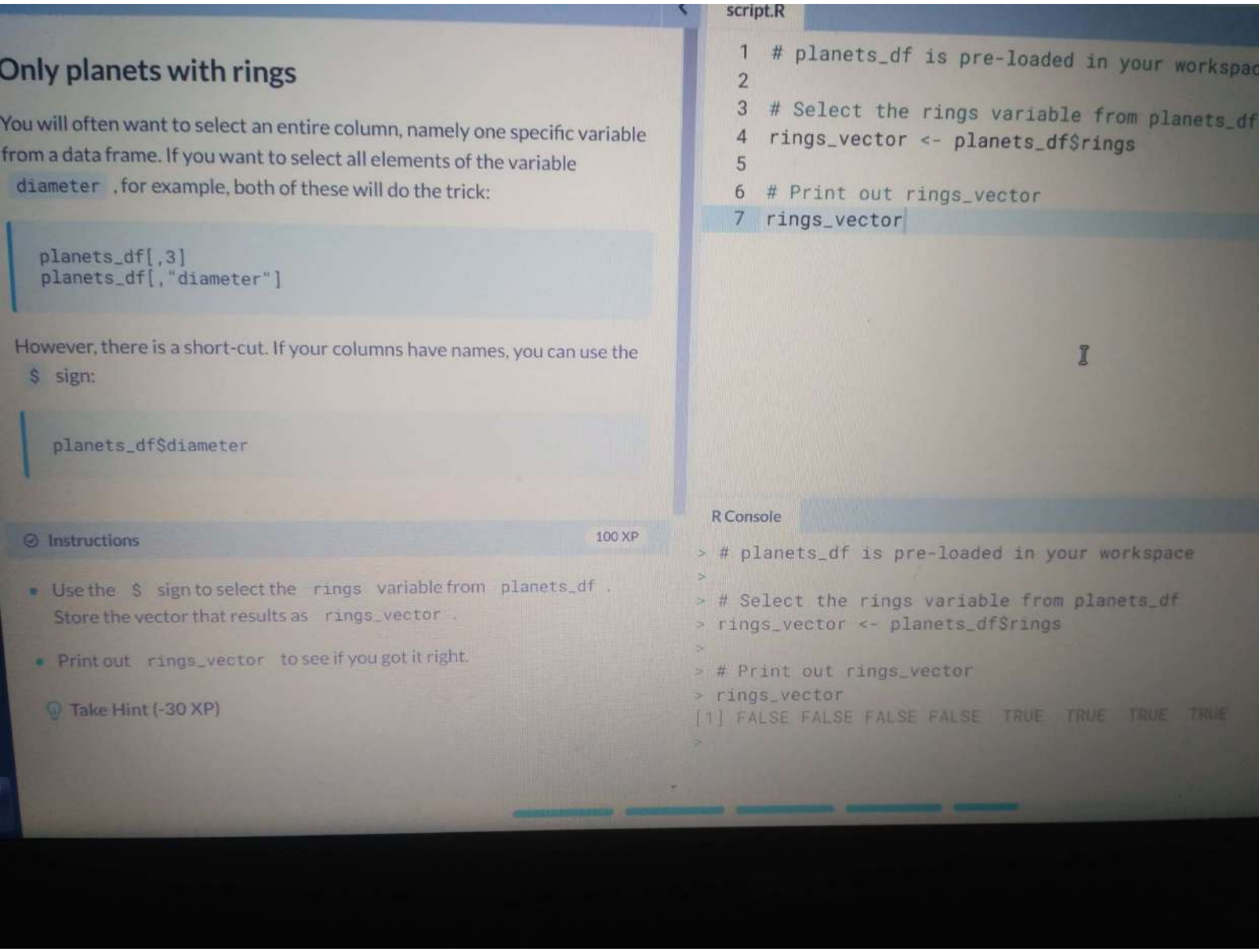
script.R

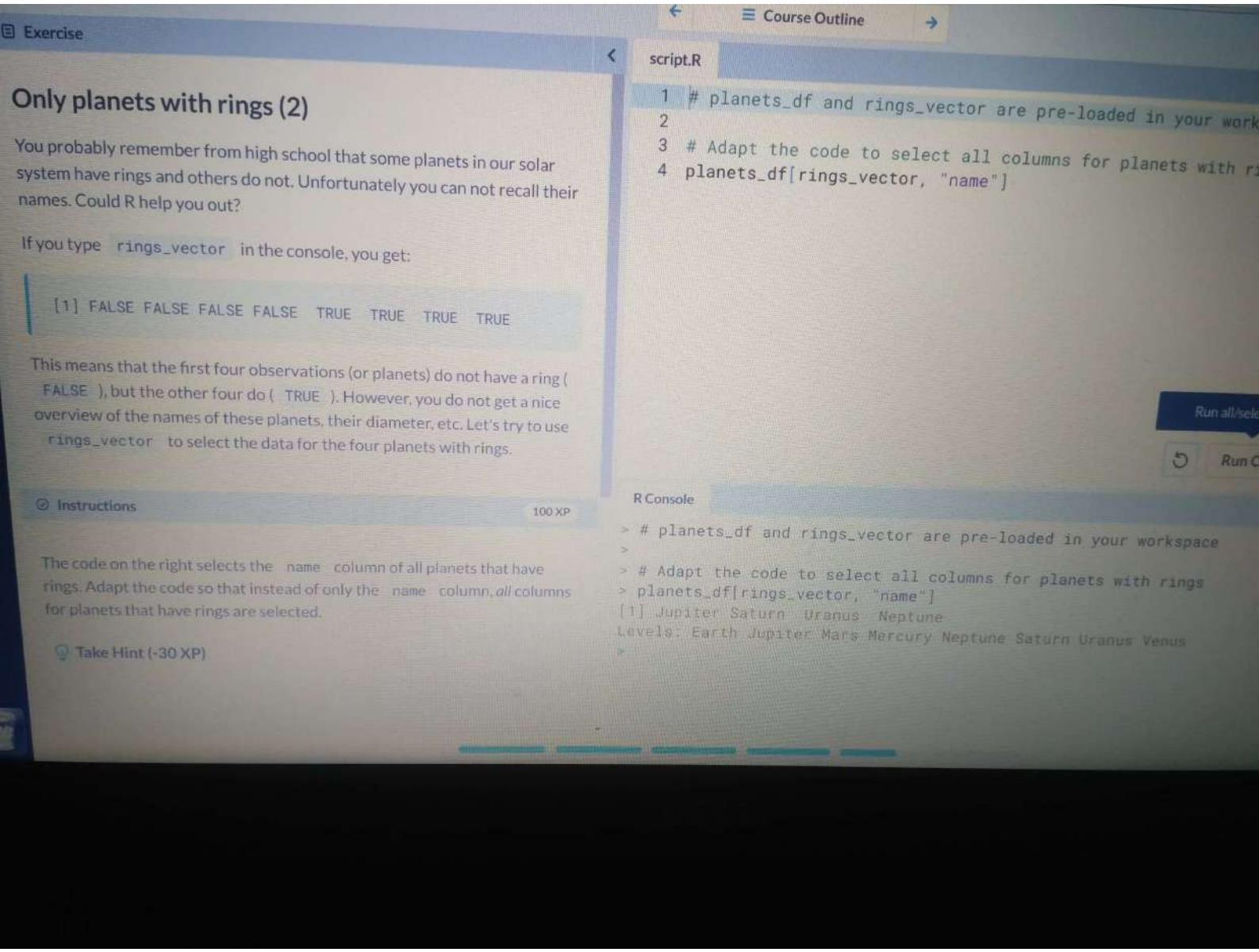
```
1 # The planets_df data frame from the previous exercise is pre-loaded
2
3 # Select first 5 values of diameter column
4 planets_df[1:5,"diameter"]
```

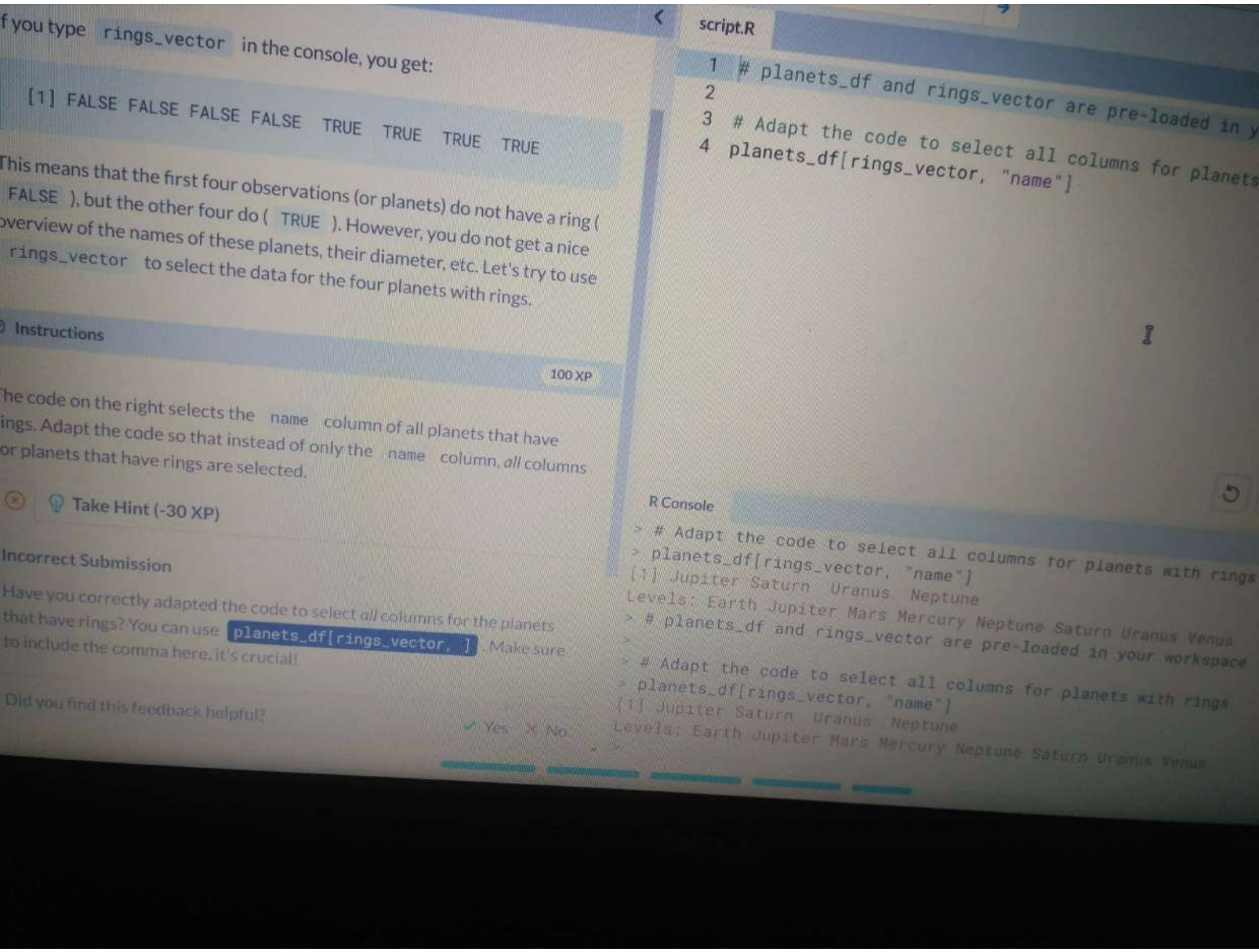
Run Code

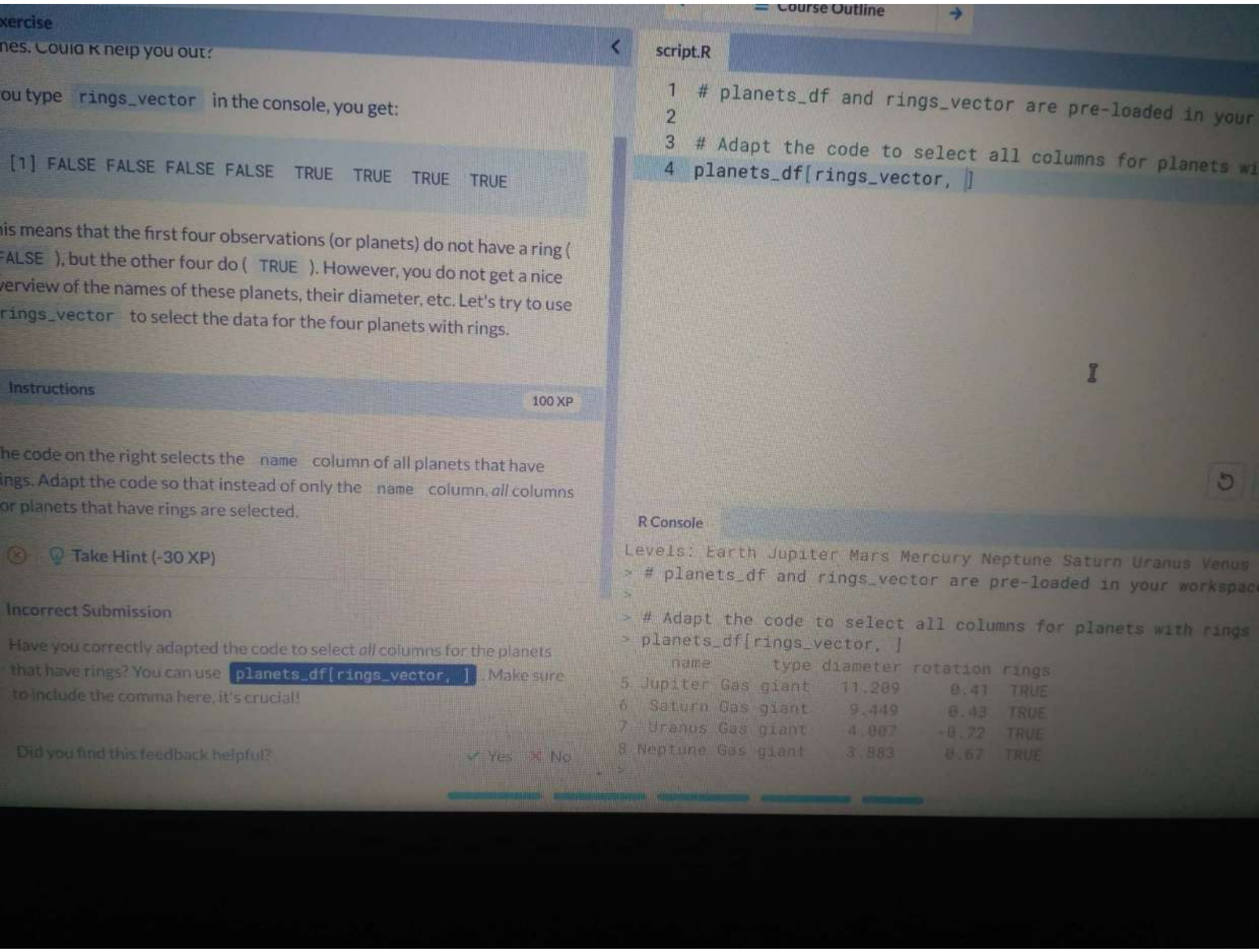
R Console

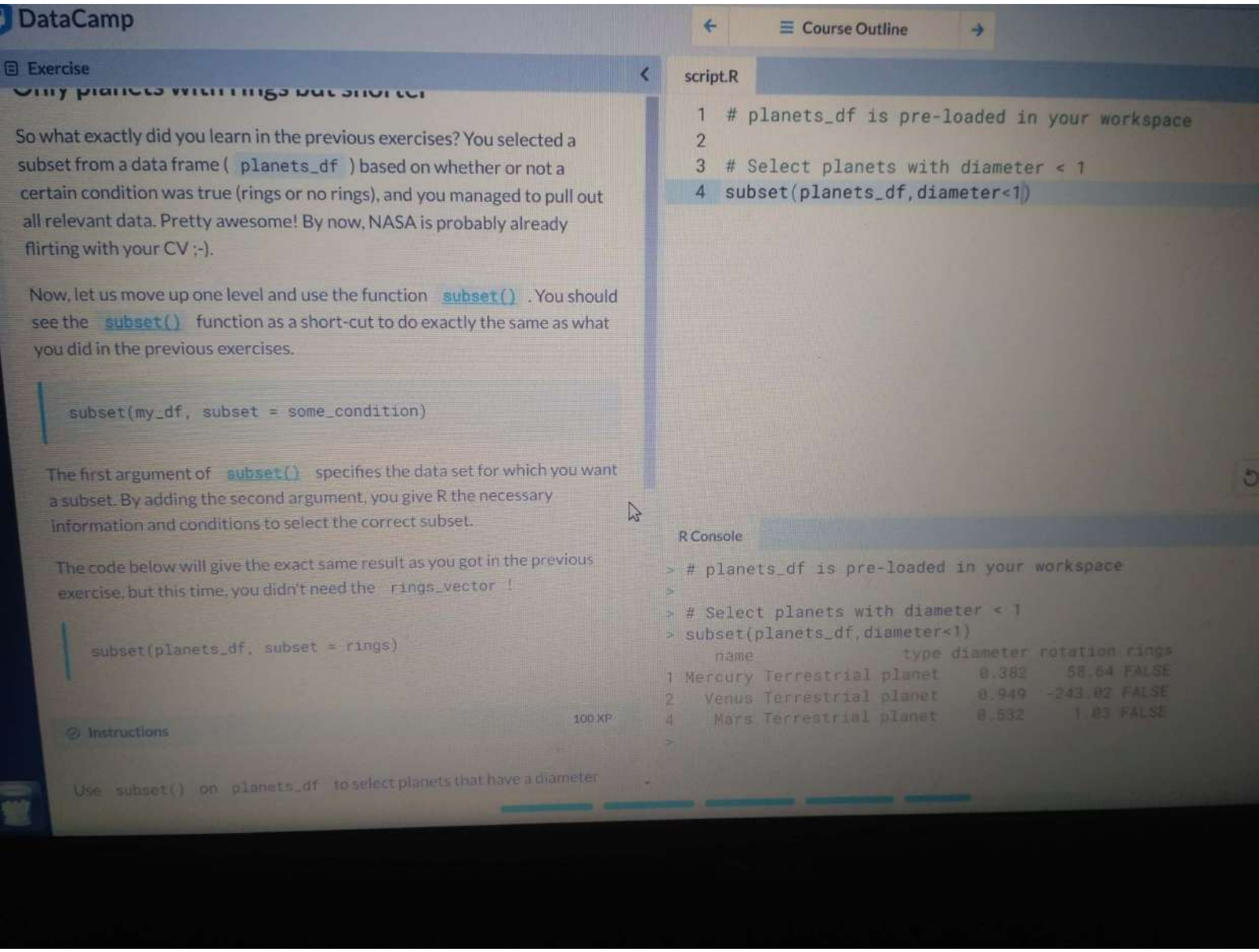
```
> # Select first 5 values of diameter column
> planets_df[5,"diameter"]
[1] 11.209
>
> # The planets_df data frame from the previous exercise is pre-loaded
>
> # Select first 5 values of diameter column
> planets_df[1:5,"diameter"]
[1]  0.382  0.949  1.000  0.532 11.209
>
```

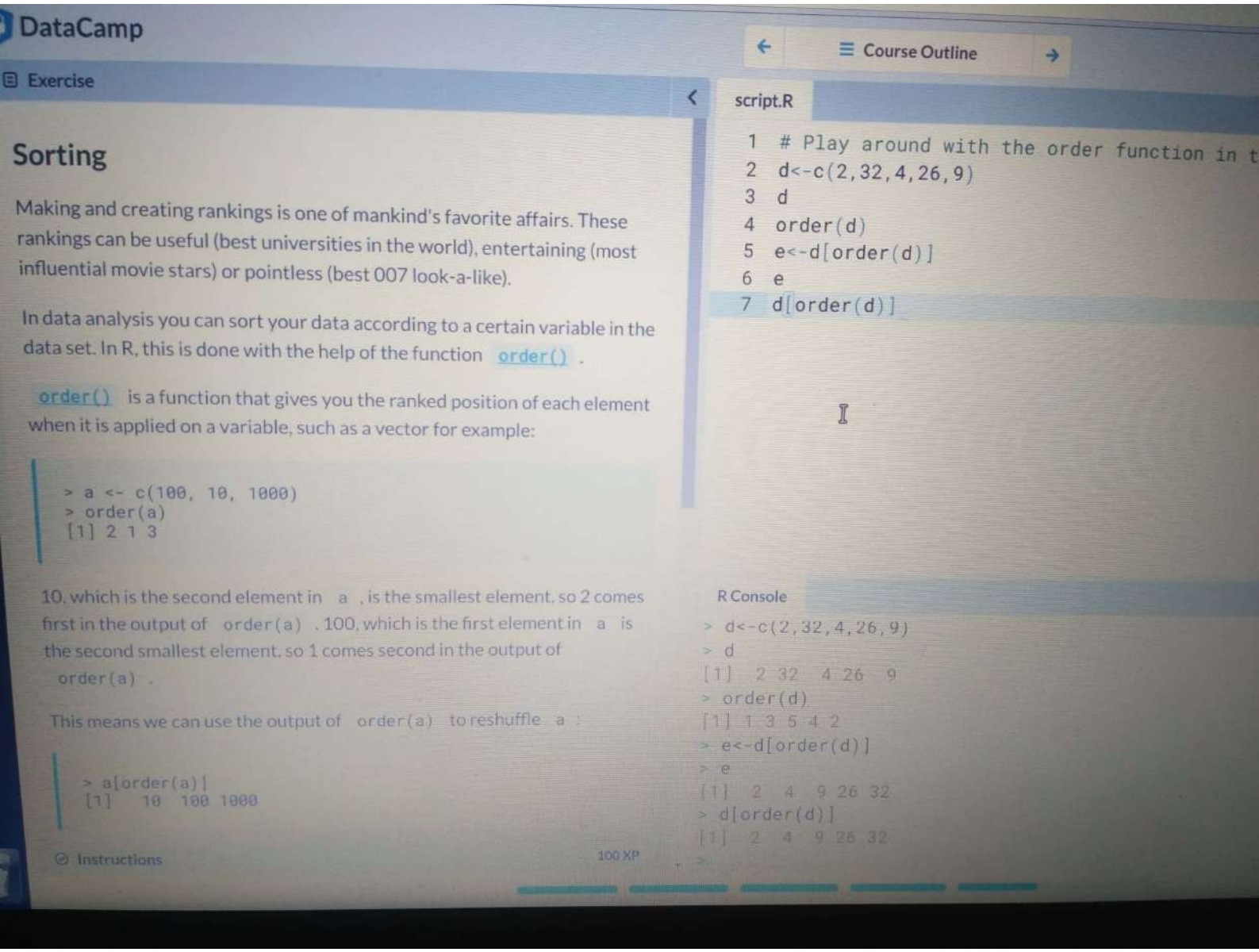


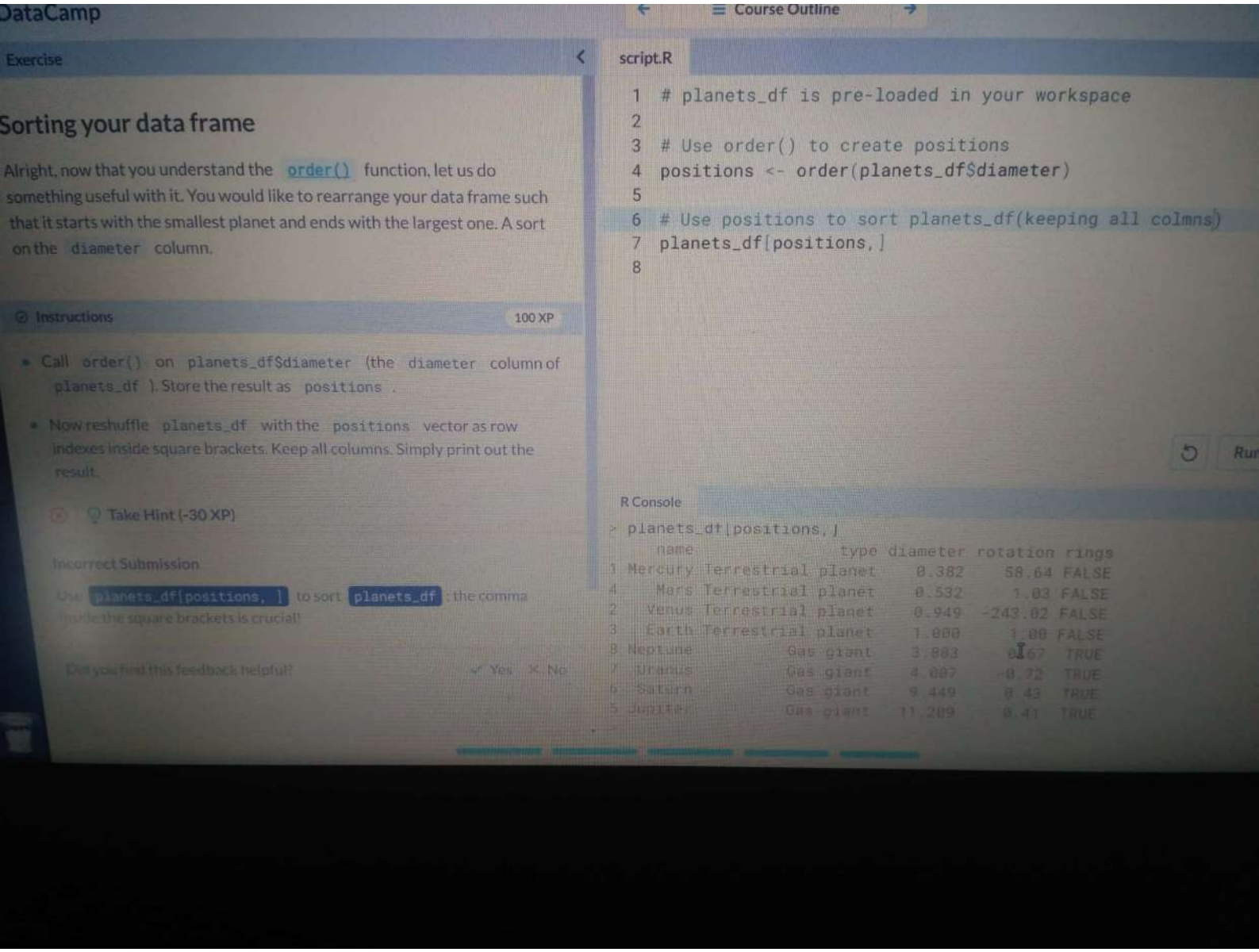












Exercise

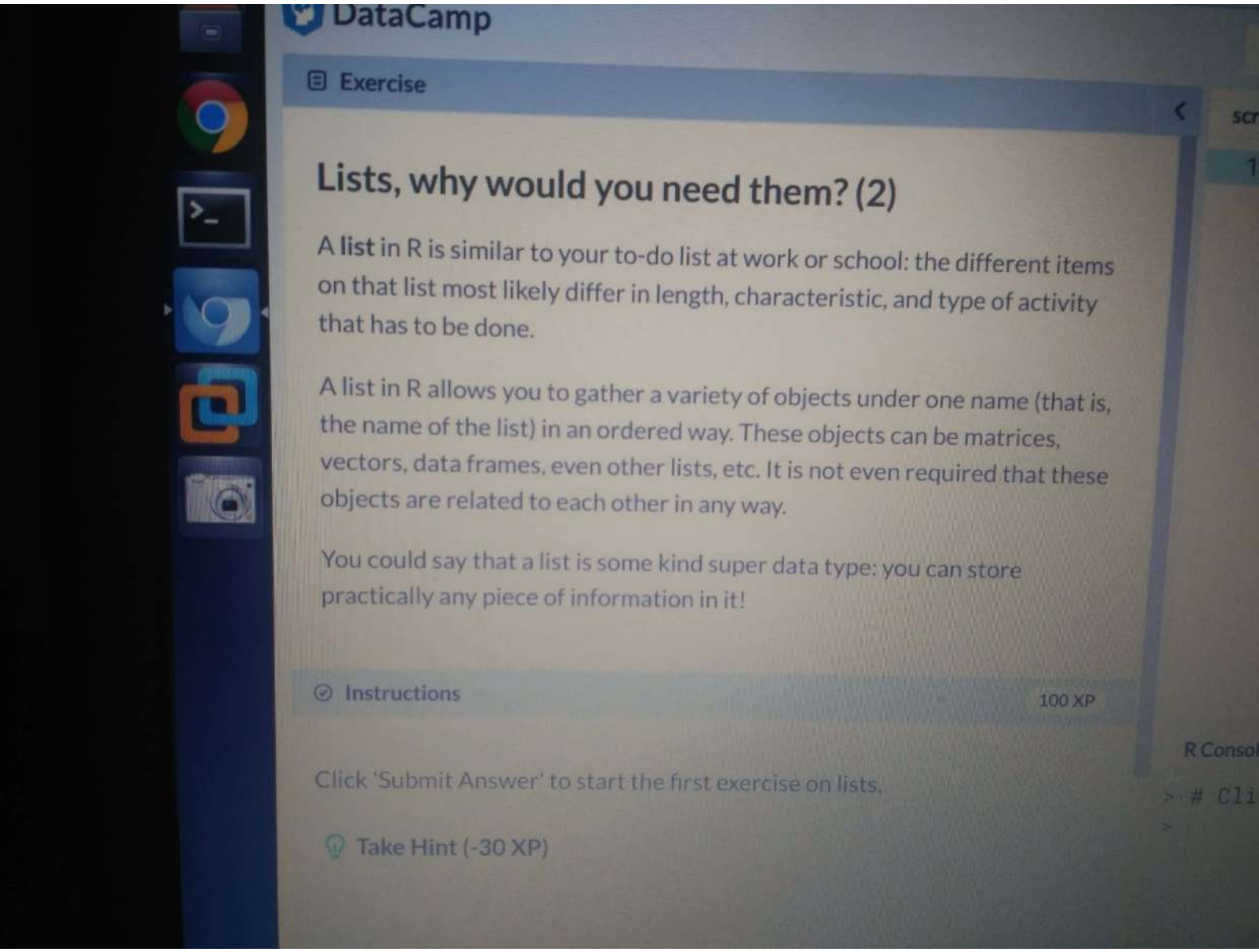
Lists, why would you need them?

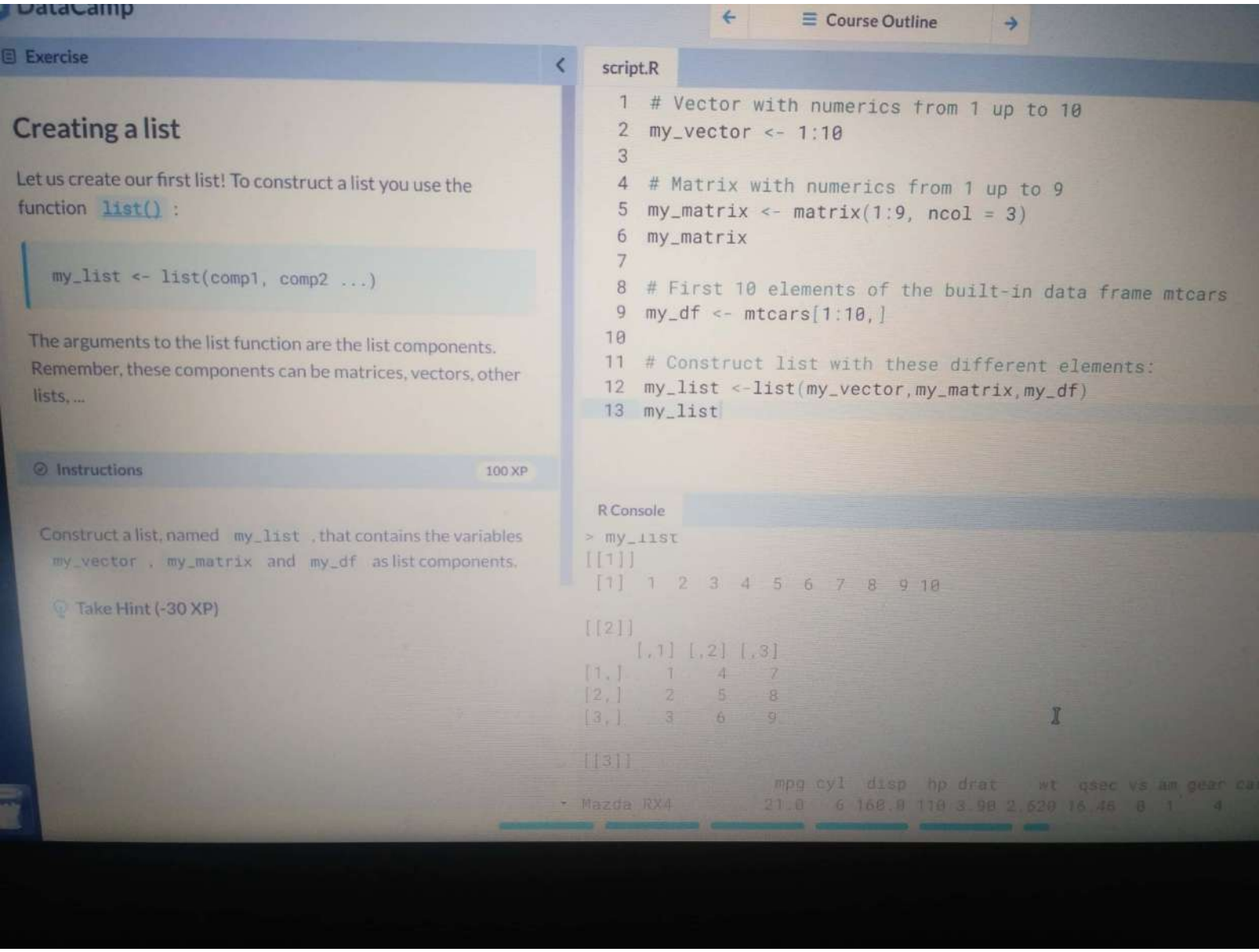
Congratulations! At this point in the course you are already familiar with:

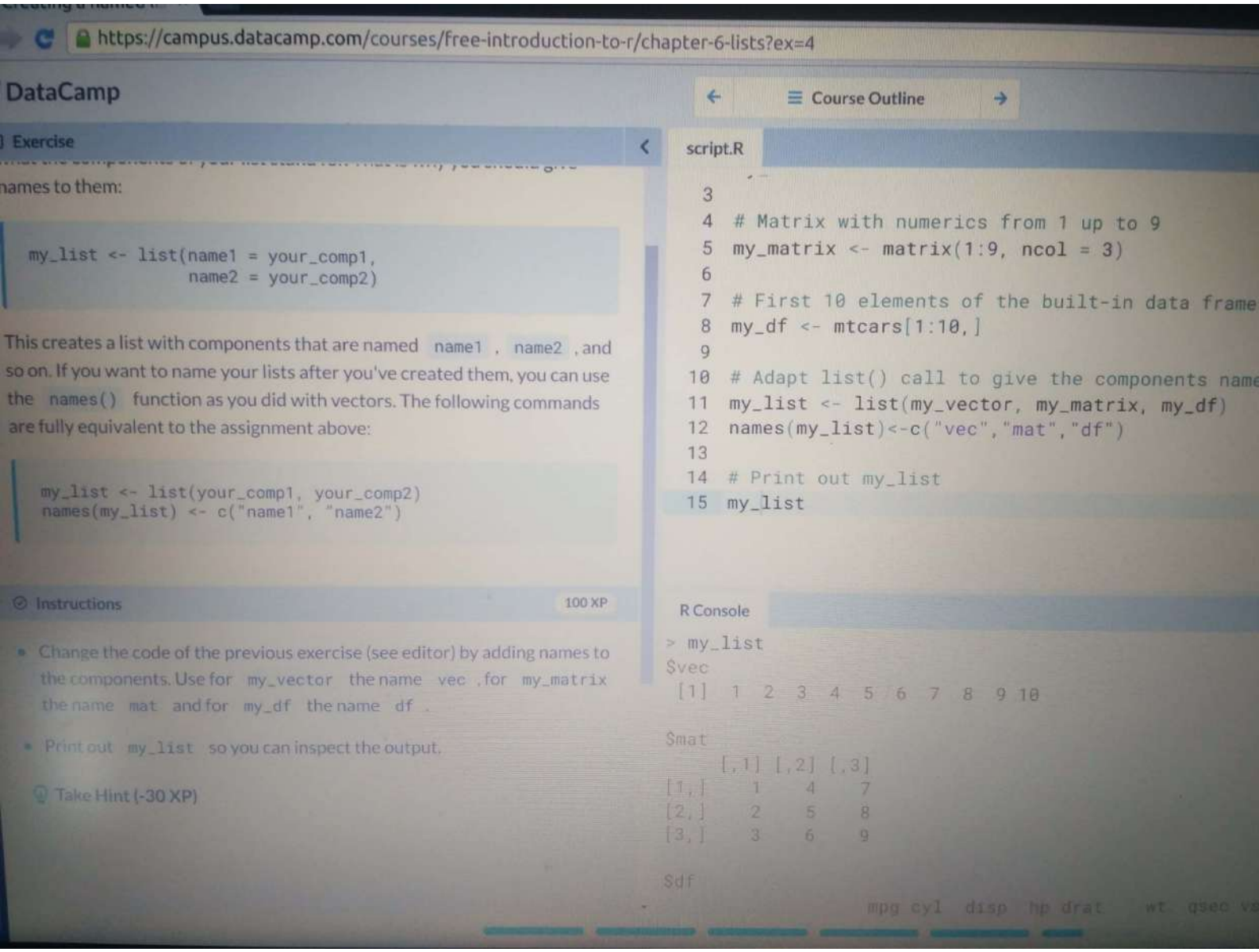
- **Vectors** (one dimensional array): can hold numeric, character or logical values. The elements in a vector all have the same data type.
- **Matrices** (two dimensional array): can hold numeric, character or logical values. The elements in a matrix all have the same data type.
- **Data frames** (two-dimensional objects): can hold numeric, character or logical values. Within a column all elements have the same data type, but different columns can be of different data type.

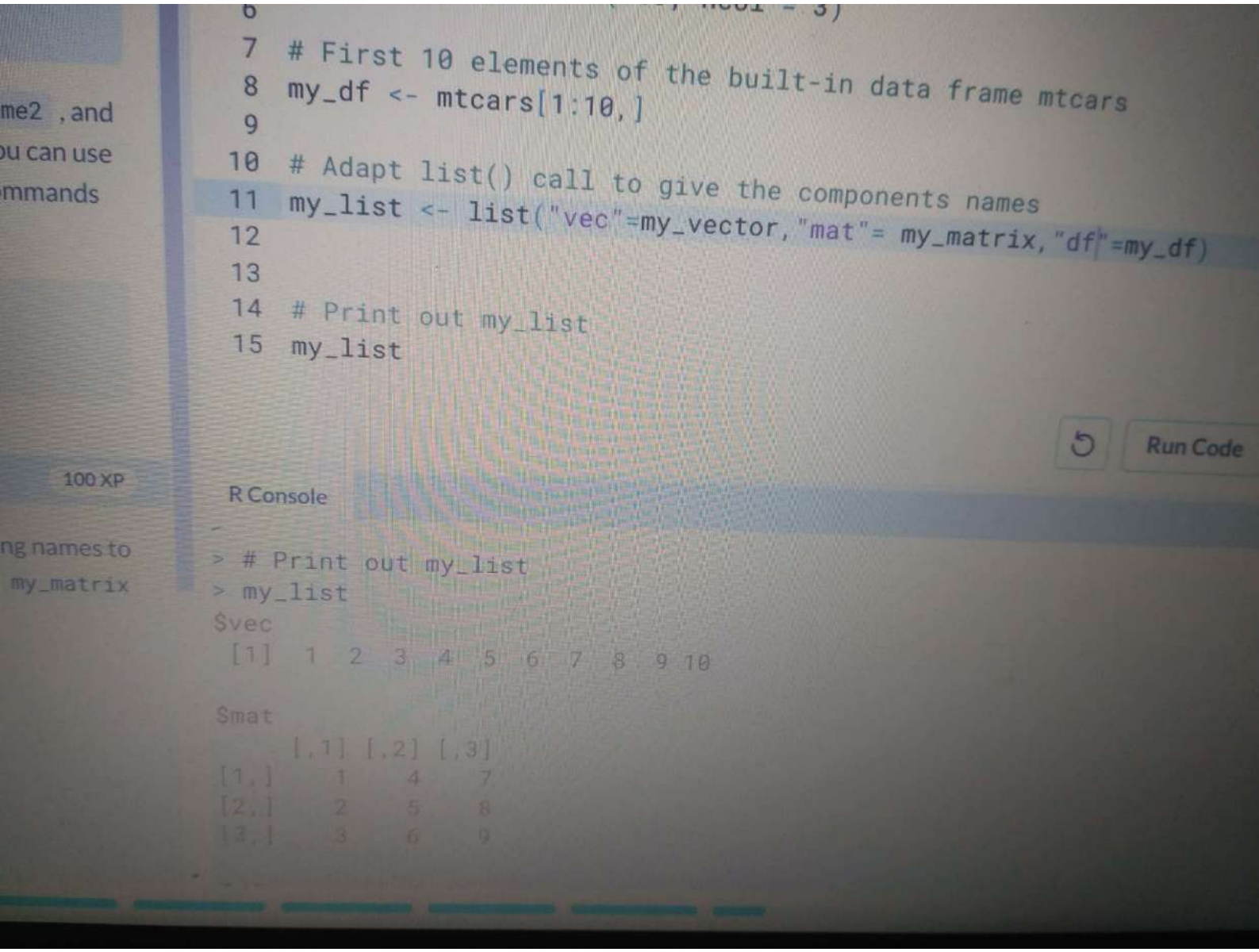
Pretty sweet for an R newbie, right? ;-)

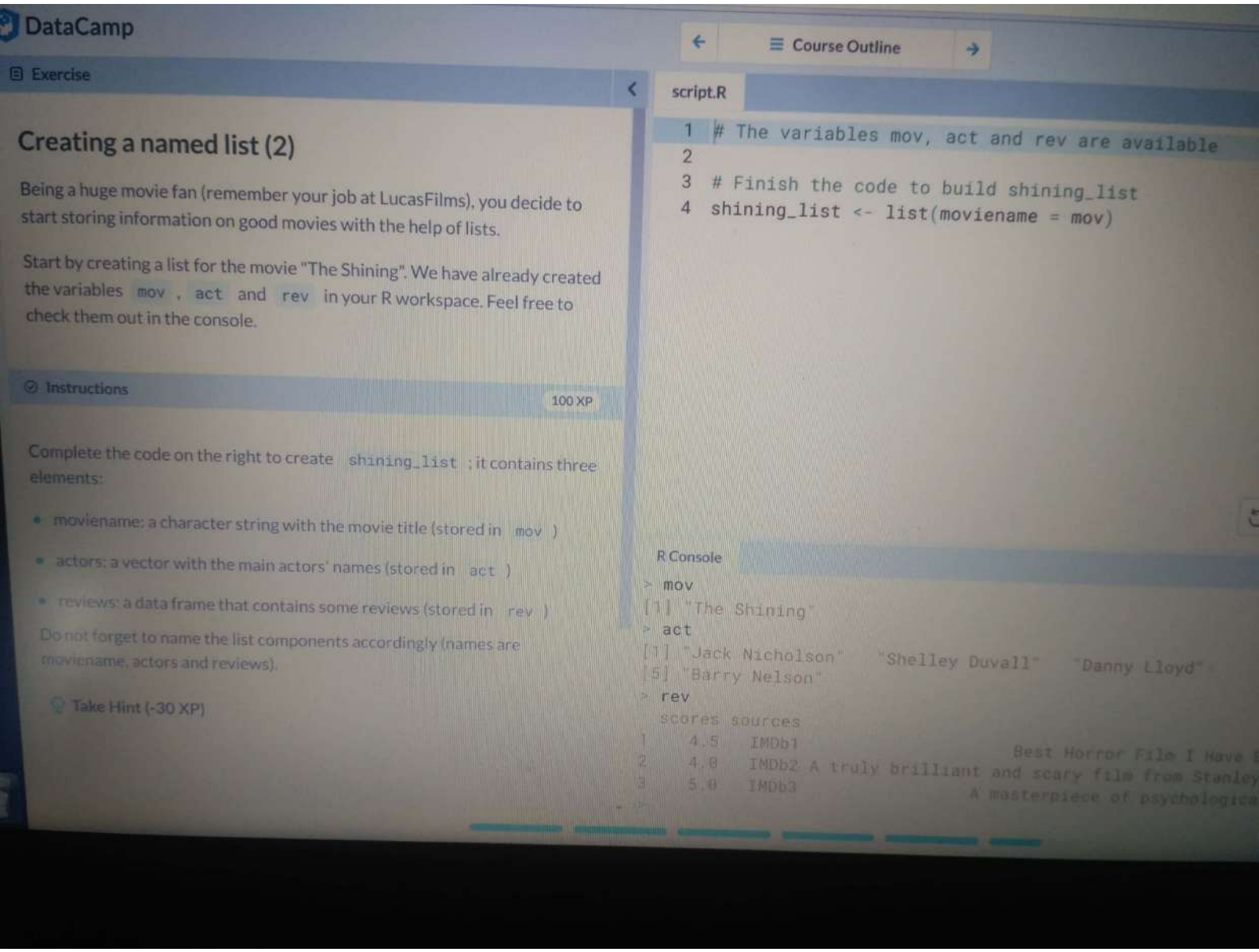
Instructions 100 XP

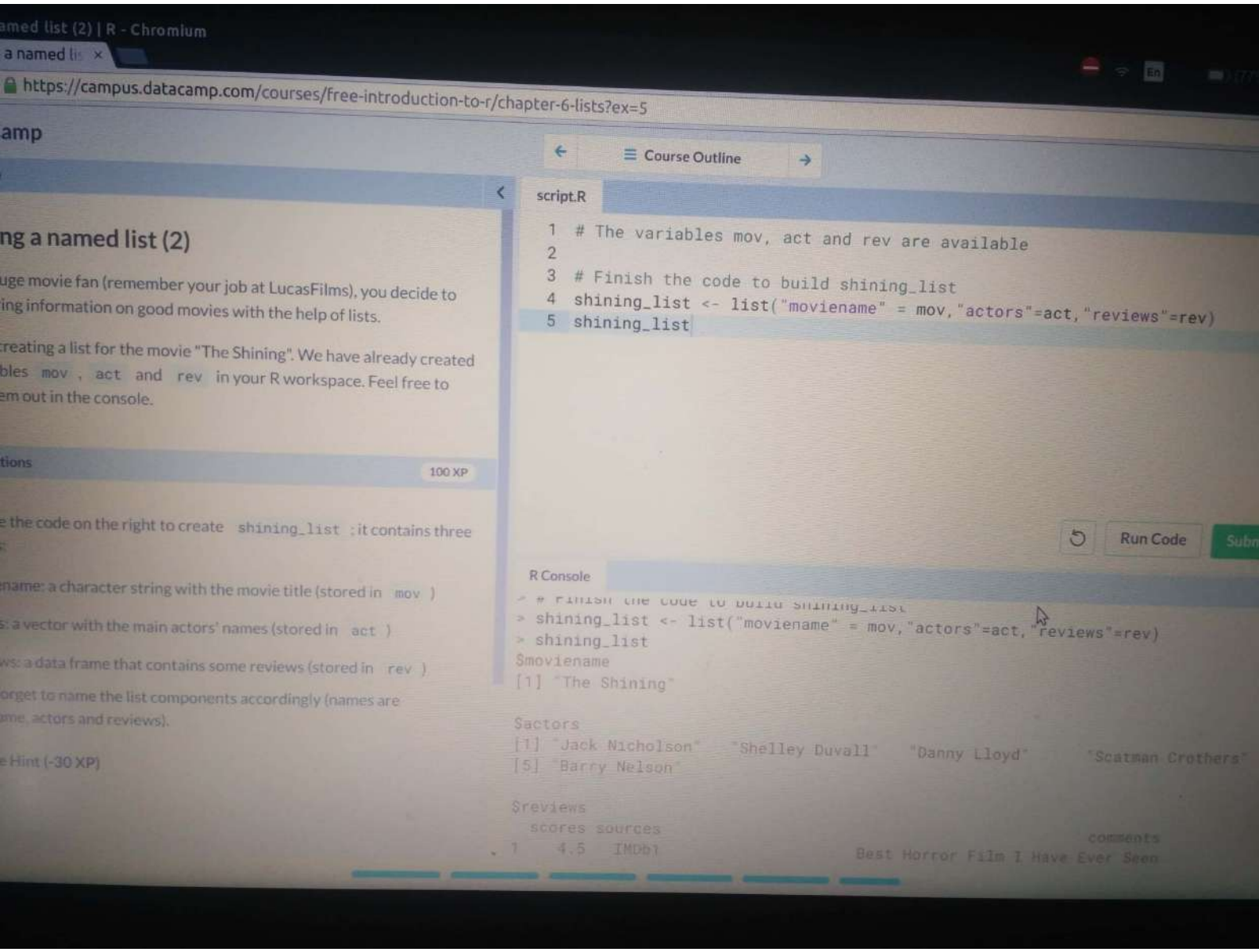


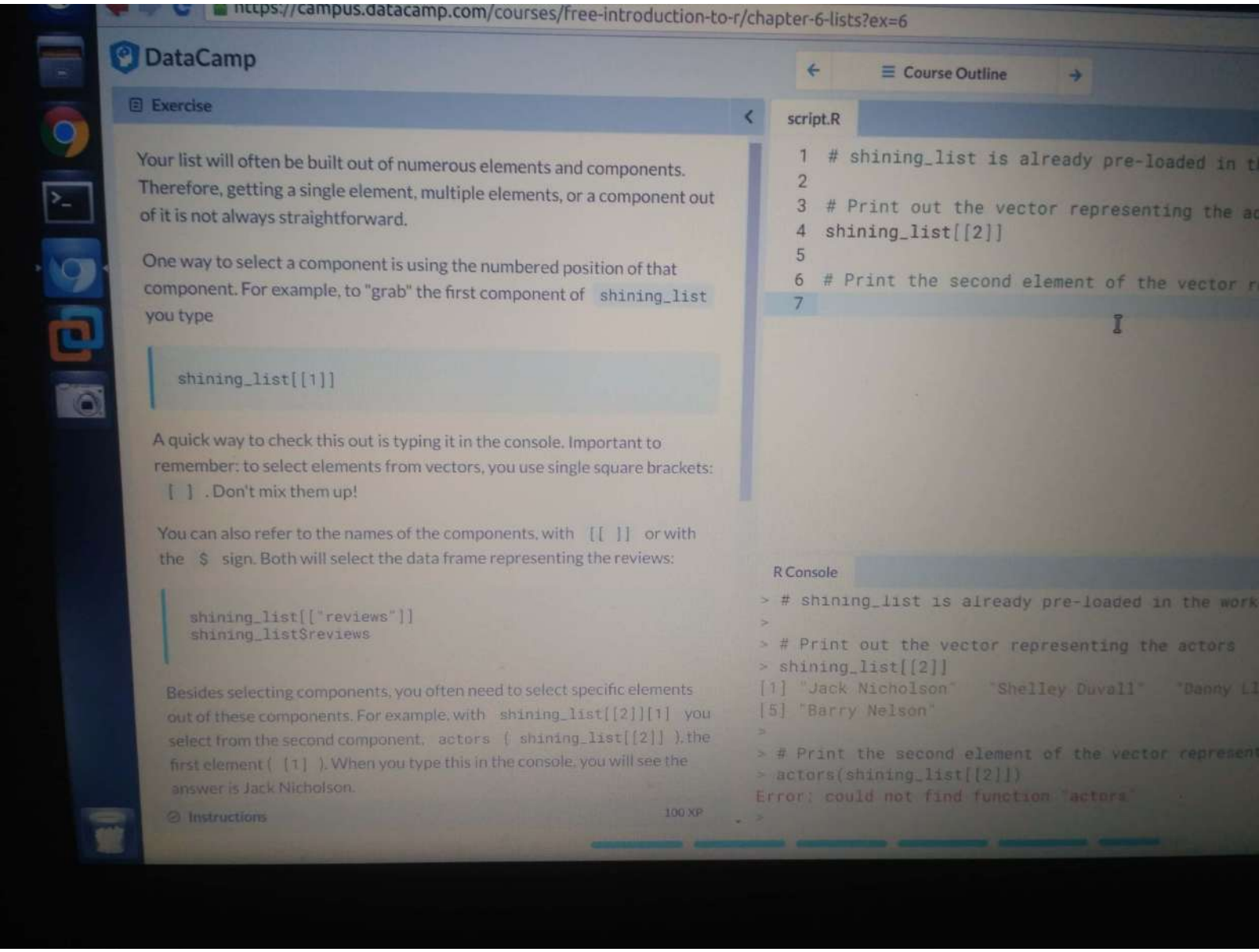


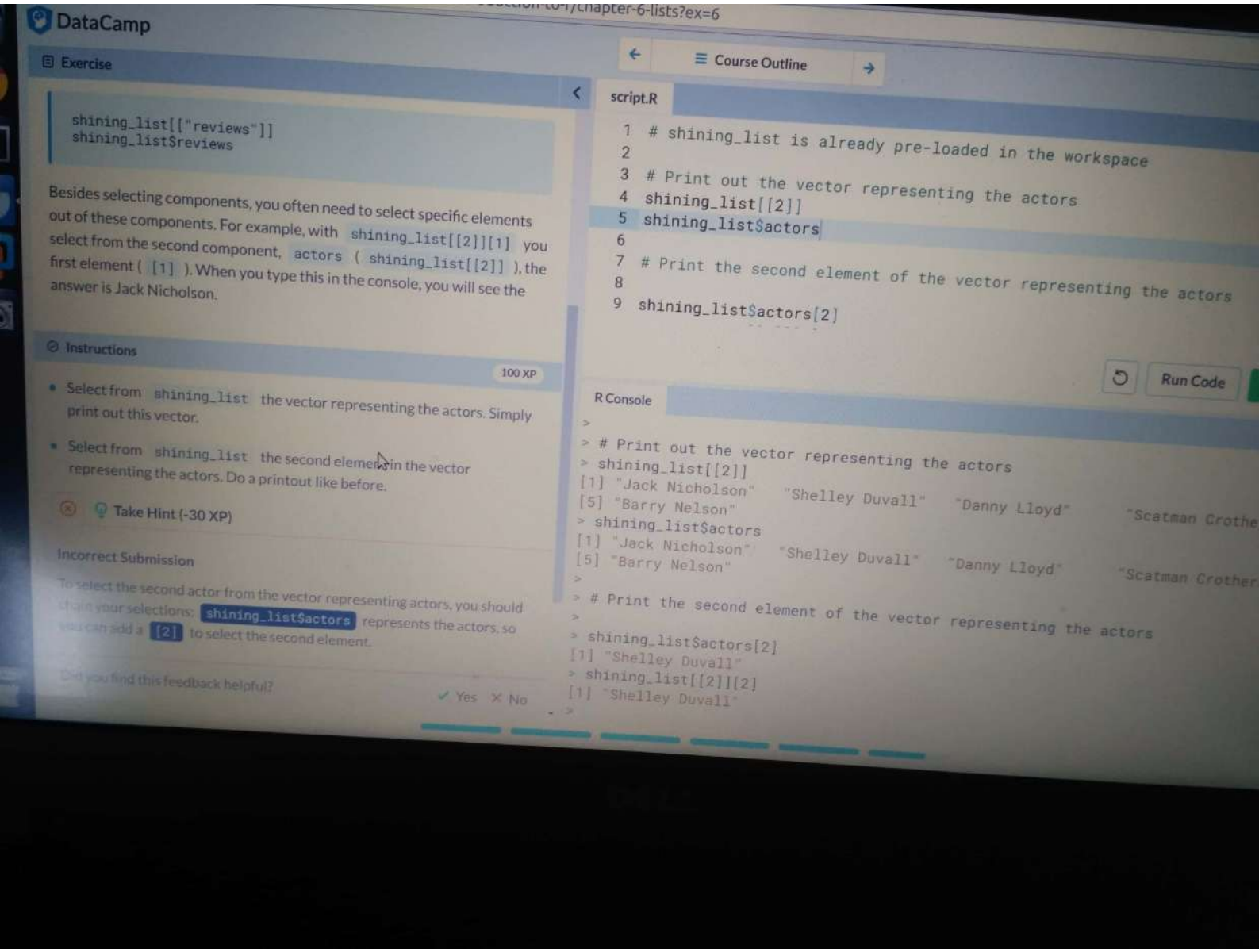


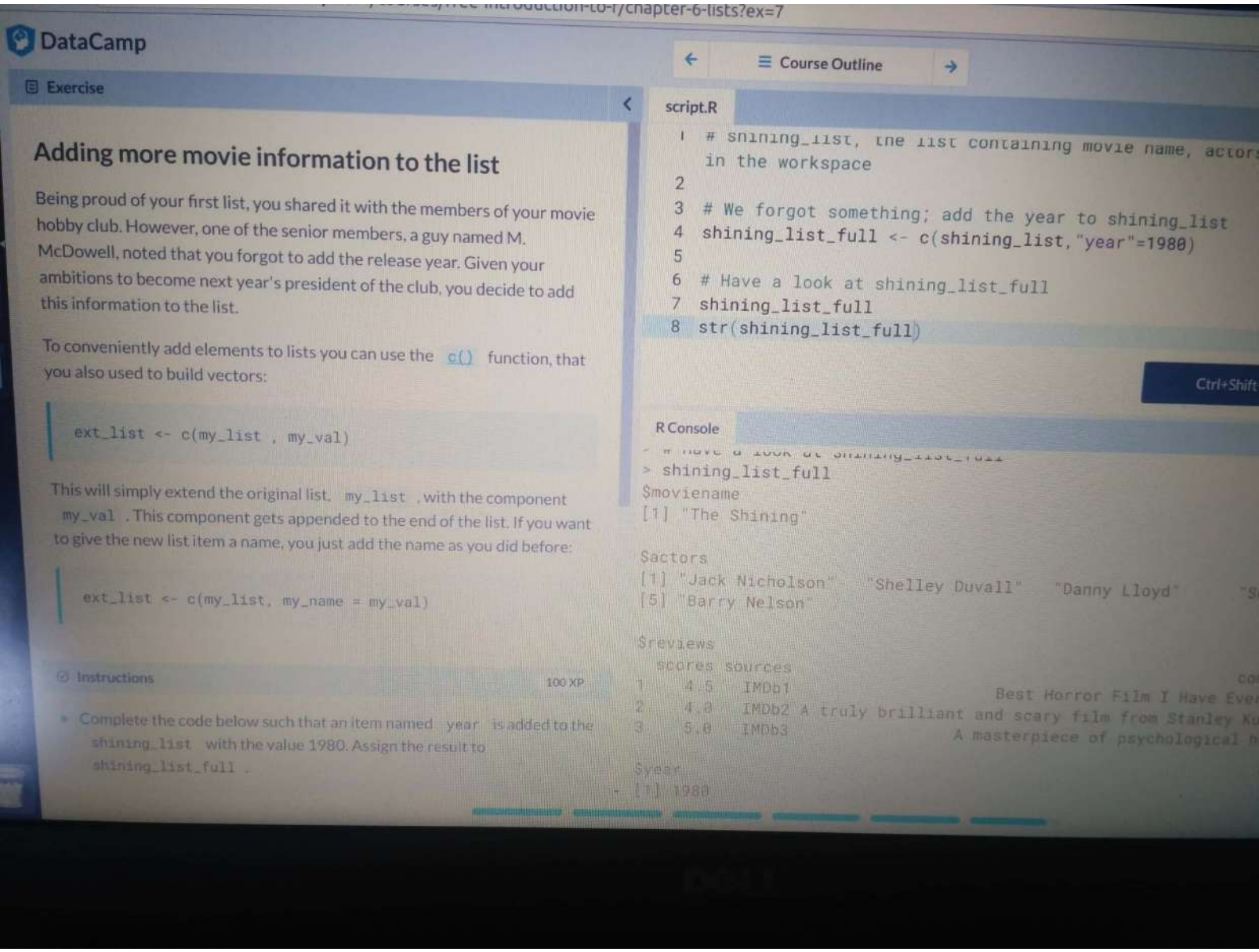


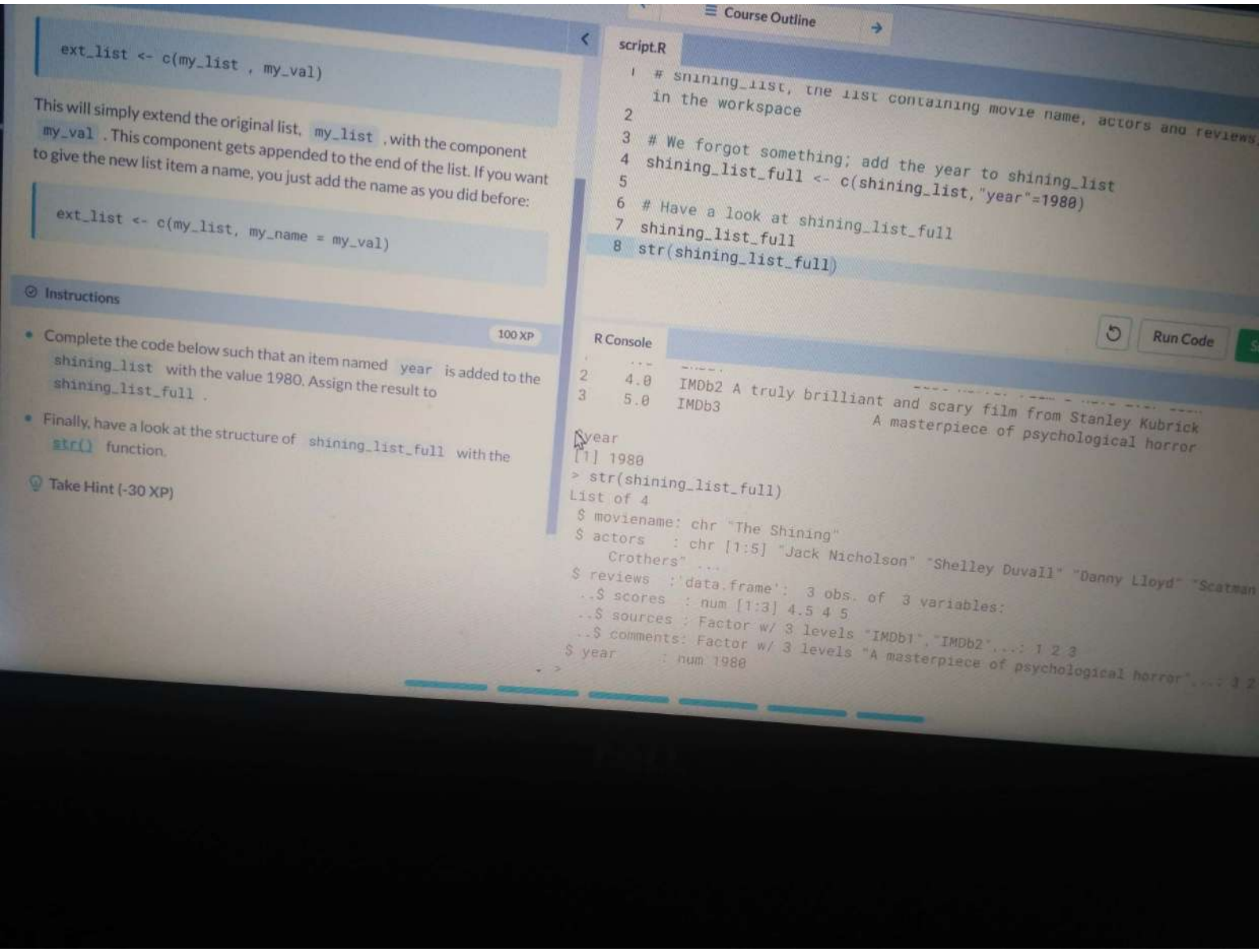


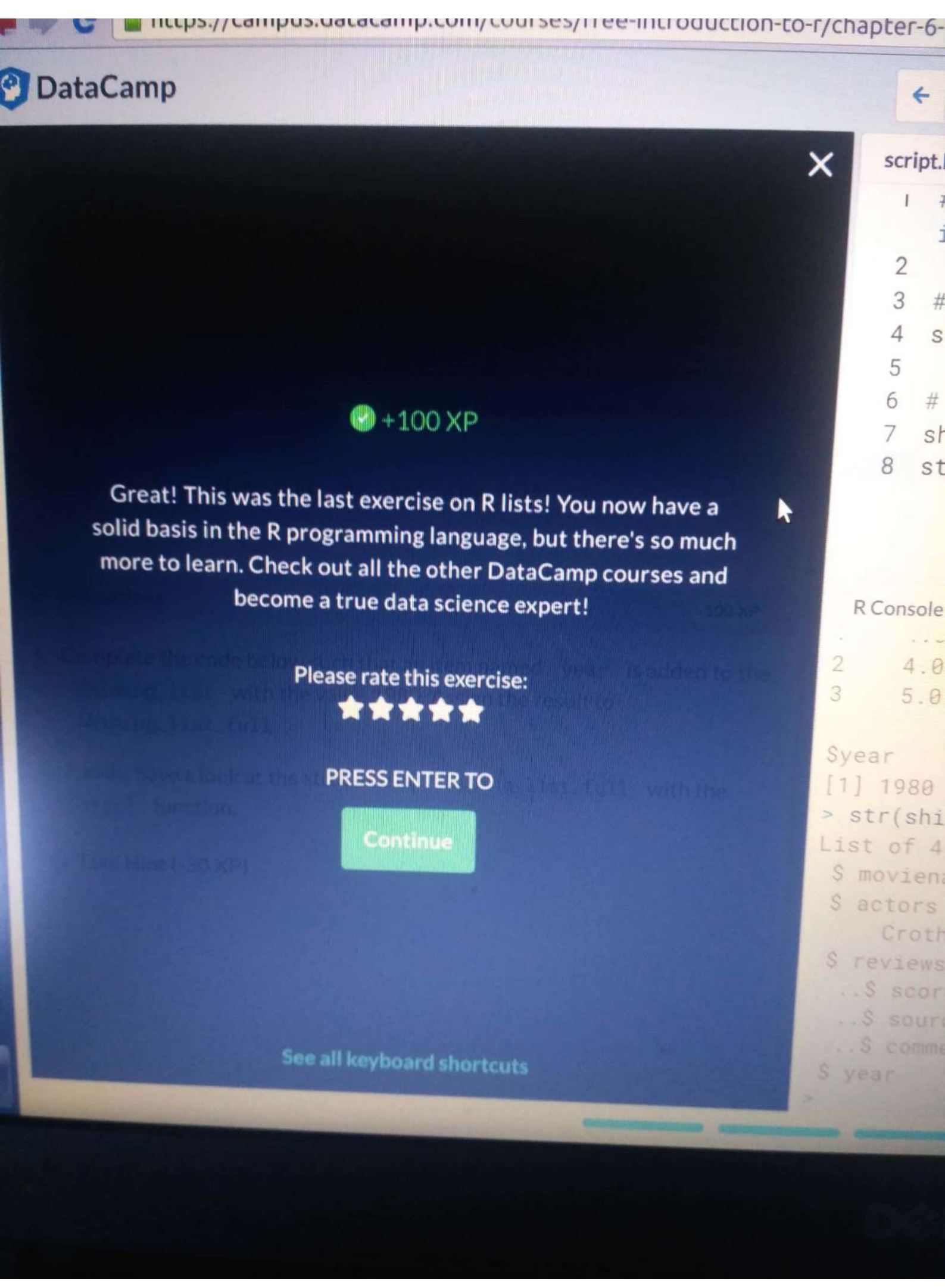


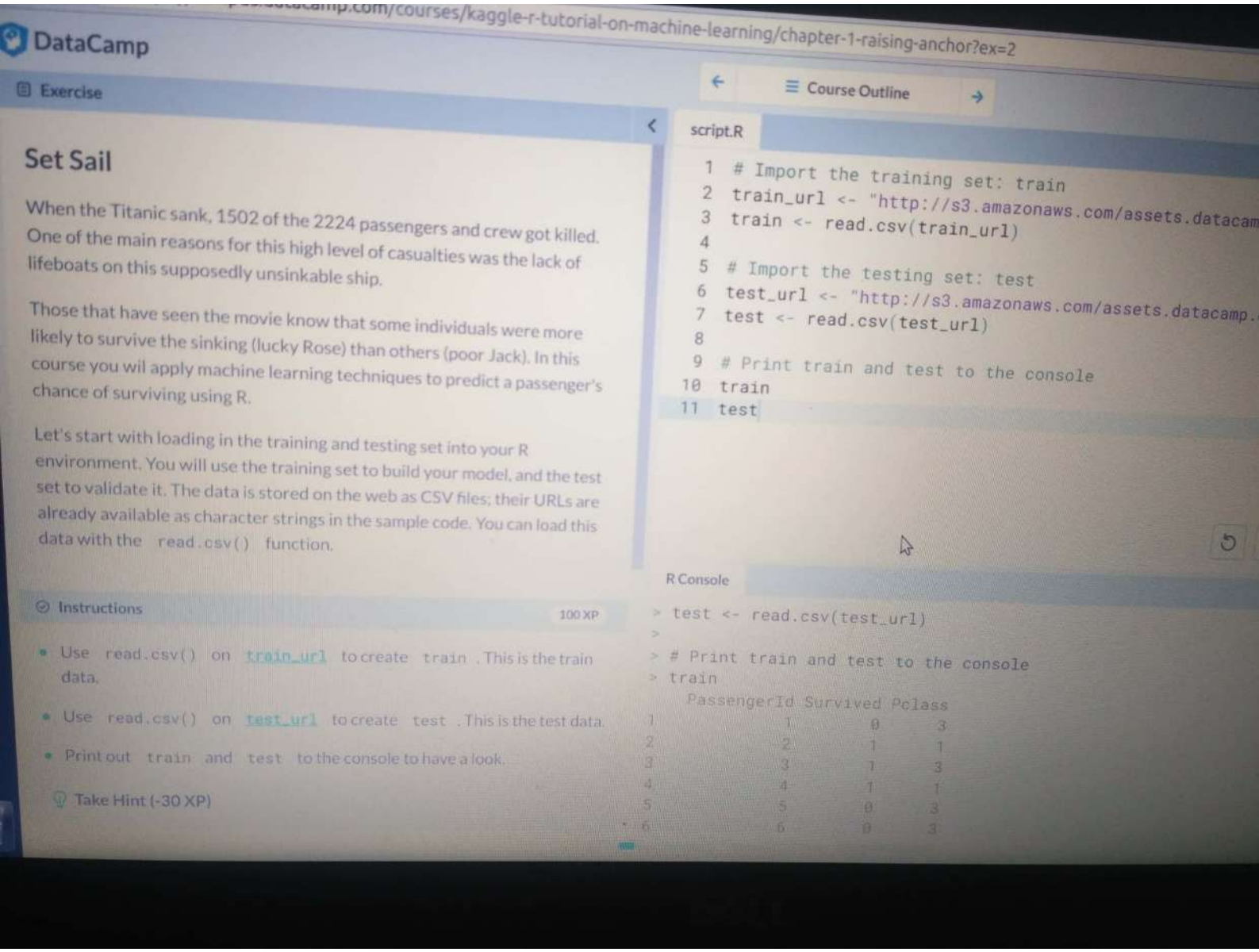












Exercise

Rose vs Jack, or Female vs Male

How many people in your training set survived the disaster with the Titanic? To see this, you can use the `table()` command in combination with the `$` -operator to select a single column of a data frame:

```
# absolute numbers
table(train$Survived)

# proportions
prop.table(table(train$Survived))
```

If you run these commands in the console, you'll see that 549 individuals died (62%) and 342 survived (38%). A simple prediction heuristic could thus be "majority wins": you predict every unseen observation to not survive.

In general, the `table()` command can help you to explore which variables have predictive value. For example, maybe gender could play a role as well? For a two-way comparison, also including gender, you can use

```
table(train$Sex, train$Survived)
```

To get proportions, you can again wrap `prop.table()` around `table()`, but you'll have to specify whether you want row-wise or column-wise proportions: This is done by setting the second argument of

Instructions

100 XP

script.R

```
1  # Your train and test set are stil
2  str(train)
3  str(test)
4
5  # Survival rates in absolute number
6
7
8  # Survival rates in proportions
9
10
11 # Two-way comparison: Sex and Surviv
12
13
14 # Two-way comparison: row-wise propo
```

R Console

```
> table(train$Survived)

  0   1
549 342
> prop.table(train$Survived)
  [1] 0.000000000 0.002923977 0.002923977 0
  [7] 0.000000000 0.000000000 0.002923977 0
 [13] 0.000000000 0.000000000 0.000000000 0
 [19] 0.000000000 0.002923977 0.000000000 0
 [25] 0.000000000 0.002923977 0.000000000 0
 [31] 0.000000000 0.002923977 0.002923977 0
```