Lab 02: R fundamentals 2

David Murillo

09/14/2022

You've used logical subsetting to select elements of a matrices and vectors. With small data sets it's possible to look at all of the elements at once and visually detect the indices of the elements you want. This is not possible with larger data sets.

Run the following code to create a large vector containing randomly generated integers between 1 and 12:

```
n = 12345
vec_1 = sample(12, n, replace = TRUE)
head(vec_1)
```

```
## [1]  7 12  2  2  6  9
```

Use a logical test operator to create a Boolean vector (called vec_2) whose entries are TRUE if the corresponding entry in vec_1 is 3 and FALSE otherwise.

```
vec_2 <- vec_1 == 3
head(vec_2)
```

```
## [1] FALSE FALSE FALSE FALSE FALSE FALSE
```

Self test: you can use vec_2 to retrieve all of the 3 elements of vec_1 using the following:

```
head(vec_1[vec_2])
```

```
## [1] 3 3 3 3 3 3
```

Q1 (2 pts.): Show the R code you used to create vec_2.

Answer: vec_2 <- vec_1 == 3

Q2 (2 pts.): Give two reasons why determining which elements in vec_1 have value 3 by visual inspection is a bad idea.

Answer: 1) There are many data to see what are number 3, this will take much time.

2) We have to annotate by hand what element are the number 3, and also this take much time. We need to be efficient and use our time of the better way possible.

Run the following code to create a large vector containing randomly generated integers between 1 and 12:

```r
n = 12345
vec_1 = sample(12, n, replace = TRUE)
head(vec_1)
```

```
## [1]  6  1  8  4 10  1
```

Use the function length() to determine how many elements are in vec_1.

```r
length(vec_1)
```

```
## [1] 12345
```

Now, run the following line to check how many entries have the value 3:

```r
sum(vec_1 == 3)
```

```
## [1] 1011
```

Finally, run the following code several times taking note of how many 3 entries appear each time you run it.

```r
n = 10
vec_1 = sample(12, n, replace = TRUE)
paste0("Sum of elements with value 3: ", sum(vec_1 == 3))
```

```
## [1] "Sum of elements with value 3: 0"
```

Q3 (1 pt.): Why didn't you always get the same count of 3 entries each time?

Answer: Because we are taking difference random samples, and every time we run the code we are using different data, because that we have diferent result.

Q4 (3 pts.): Considering the different vectors generated each time, explain why using a logical test is a safe way to select entries with a value of 3.

Answer: Because the logical test is automatic tool, so although the number of 3 and their posisition change, the logical test will detect without problem.

Q5 (5 pts.): Explain why performing logical 'by hand' subsetting is very very bad practice. You may want consider re-usability of code, working with different sized data sets, and sharing code with collaborators.

Answer: For improve the time, and use different data set, we could use other tool instain logical, for example we could make a function, or we could use a function already done in R.

Many time, our college will want to run our code with their data, o with diffent paramaters, if our code is made with our expecification our college will have to change the code and this could produce delay ans loss time.

You may want to review the for-loop example in the lab walkthrough.

```r
for (i in 1:10)
{
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

Modify the code in the body of the loop to print out a message like "This is loop iteration: 1" for each run through the loop.

```r
for (i in 1:10)
{
  print(paste0("This is loop iteration:", i))
}
```

```
## [1] "This is loop iteration:1"
## [1] "This is loop iteration:2"
## [1] "This is loop iteration:3"
## [1] "This is loop iteration:4"
## [1] "This is loop iteration:5"
## [1] "This is loop iteration:6"
## [1] "This is loop iteration:7"
## [1] "This is loop iteration:8"
## [1] "This is loop iteration:9"
## [1] "This is loop iteration:10"
```

Q6 (3 pts.): Provide the code for your modified loop. It must run as a self-contained example on a fresh R session on my computer.

Answer:

```r
for (i in 1:10)
{
  print(paste0("This is loop iteration:", i))
}
```

```
## [1] "This is loop iteration:1"
## [1] "This is loop iteration:2"
## [1] "This is loop iteration:3"
## [1] "This is loop iteration:4"
## [1] "This is loop iteration:5"
## [1] "This is loop iteration:6"
## [1] "This is loop iteration:7"
## [1] "This is loop iteration:8"
## [1] "This is loop iteration:9"
## [1] "This is loop iteration:10"
```

You may want to review the for-loop example in the lab walkthrough.

Run the following code on your computer:

```
for (i in 1:10)
{
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

Note that the loop runs through exactly 10 iterations…

What if you wanted the loop to execute an arbitrary number of times?

Create a variable, n, that contains an integer value. Modify the code for the loop so that it runs n times.

```
n <- sample(c(1:20), 1)

for (i in 1:n)
{
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
```

Q7 (2 pts.): Provide the code for the modified loop that executes n times. It needs to be a self contained example. I should be able to set the value of n and then run your loop on my computer.

Answer:

```r
n <- sample(c(1:20), 1)

for (i in 1:n)
{
  print(i)
}
```

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

Create an integer variable, n, that holds the value 17.

```r
n <- 17
```

Write code to create a vector called vec_1 of length n. vec_1 should contain [pseudo]randomly generated integers between 1 and 10.

```r
vec_1 <- sample(c(1:10), n, replace = TRUE)
```

Now, create a loop that:

Iterates n times (once for each element of vec_1). Prints a message that includes the iteration number as well as the corresponding element of vec_1

Q8 (4 pts.): Provide the code you used to create the n, vec_1, and the loop. As always, it should run as a stand-alone example in a fresh R session on my computer.

```r
n <- 17

vec_1 <- sample(c(1:10), n, replace = TRUE)

for (i in 1:n)
{
  print(paste0("The element of vec_1 at index ", i, " is: ", vec_1[i] ))
}
```

```
## [1] "The element of vec_1 at index 1 is: 10"
## [1] "The element of vec_1 at index 2 is: 10"
## [1] "The element of vec_1 at index 3 is: 7"
## [1] "The element of vec_1 at index 4 is: 7"
## [1] "The element of vec_1 at index 5 is: 8"
## [1] "The element of vec_1 at index 6 is: 10"
## [1] "The element of vec_1 at index 7 is: 1"
## [1] "The element of vec_1 at index 8 is: 4"
## [1] "The element of vec_1 at index 9 is: 4"
## [1] "The element of vec_1 at index 10 is: 3"
## [1] "The element of vec_1 at index 11 is: 1"
## [1] "The element of vec_1 at index 12 is: 7"
## [1] "The element of vec_1 at index 13 is: 1"
## [1] "The element of vec_1 at index 14 is: 3"
## [1] "The element of vec_1 at index 15 is: 5"
## [1] "The element of vec_1 at index 16 is: 2"
## [1] "The element of vec_1 at index 17 is: 4"
```

Write a function create_and_print_vec().

You've created a loop that can print a message with the value of each element of a vector of arbitrary length.

Now you'll wrap it all into a custom function. Review the lab materials about writing custom functions in the lab walkthrough.

```r
create_and_print_vec <- function(n, min = 1, max = 10){
  n = sample(min:max, 10)

  for (i in 1:n)
{
  print(paste0("The element at index ", i, " is: ", n[i] ))
  }
}

create_and_print_vec(n)
```

```
## Warning in 1:n: numerical expression has 10 elements: only the first used

## [1] "The element at index 1 is: 10"
## [1] "The element at index 2 is: 8"
## [1] "The element at index 3 is: 4"
## [1] "The element at index 4 is: 6"
## [1] "The element at index 5 is: 1"
## [1] "The element at index 6 is: 3"
## [1] "The element at index 7 is: 2"
## [1] "The element at index 8 is: 5"
## [1] "The element at index 9 is: 7"
## [1] "The element at index 10 is: 9"
```

Q9 (10 pts.): Provide the code you used to build your function.

To receive full credit your code must run without error on a new R session and produce output similar to the examples given in the instructions.

Answer:

```r
create_and_print_vec <- function(n, min = 1, max = 10){
  n = sample(min:max, 10)

  for (i in 1:n)
{
  print(paste0("The element at index ", i, " is: ", n[i] ))
  }
}

create_and_print_vec(n)
```

```
## Warning in 1:n: numerical expression has 10 elements: only the first used
```

```
## [1] "The element at index 1 is: 10"
## [1] "The element at index 2 is: 3"
## [1] "The element at index 3 is: 1"
## [1] "The element at index 4 is: 6"
## [1] "The element at index 5 is: 9"
## [1] "The element at index 6 is: 5"
## [1] "The element at index 7 is: 4"
## [1] "The element at index 8 is: 7"
## [1] "The element at index 9 is: 2"
## [1] "The element at index 10 is: 8"
```