ECO 634 - Lab 2 – R Fundamentals
Johanna Ravenhurst
Worked with: Sarah Guitart
September 21, 2022

Logical Subsetting 1

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

- Your code should be a complete and self-contained example. I should be able to paste your code into a fresh R session on my computer and re-create your vec_2

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

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

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

- In a large dataset like this one (n=12345), it would be time-consuming to search through vec_1 for values of 3 by visual inspection.
- Using only visual inspection instead of creating a new vector means that there is no way to use the elements with values of 3 in future functions/coding in R.

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

- This code is creating a vector with 10 randomly generated integers between 1 and 12. We don't get the same count of 3 entries each time because vec_1 was created from a random sample and a new set of random integers is selected each time you run the code.

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.

- Using a logical test is a quick and accurate way to select entries with a value of 3 each time a new vector is generated. You can simply re-run the logical test each time you create the vector which contains a randomly generated sample of integers. You could also create a new vector for each logical test you run, therefore saving information about the number of elements with a value of 3 each time you run the logical test (if, for example, you wanted to run the logical test after creating several different vectors).

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.

Your answer should cite at least *two* reasons why 'by hand' subsetting is bad.

- If you perform logical subsetting "by hand", there is no way to give your code to another person so they can re-create what you did to complete your analysis. This limits the reproducibility of your work.
- It might be doable to perform logical "by hand" subsetting for a small dataset, but once you are working with larger datasets it becomes unreasonable to complete this type of work by hand. Using R code will speed up the process and reduce (or eliminate) the chance of errors.

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.

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

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.

```
n=150
for (i in 1:n)
{
print(i)
}
```

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.

```
n <-17
vec_1 <- sample(10, n, replace = TRUE)


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

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.

```
create_and_print_vec = function(n, min = 1, max = 10, print=TRUE)
{
vec_3 = sample(x = min:max, size = n, replace = TRUE)
 for (i in 1:n)
 {
print(paste("The element at index", i, "is", vec_3[i] ))
}
}

#Code to check the function:

create_and_print_vec(20, min = 1, max = 10)

create_and_print_vec(10, min = 100, max = 2000)
```