Analysis of Environmental Data
Lab 02: R Fundamentals 2
Jessica Martinez

Logical Subsetting I

1. **Show the R code you used to create `vec_2`.**

   I used the following code to create vec_2:

   ```
   n = 12345
   vec_1 = sample (12, n, replace = TRUE)
   head (vec_1)
   vec_2 = vec_1==3
   vec_2
   ```

2. **Give two reasons why determining which elements in `vec_1` have value `3` by visual inspection is a bad idea.**

   First, determining which elements in vec_1 that have a value of 3 by visual inspection is a bad idea because the vector is quite large in size, making visually eyeballing returns of TRUE values (those that have a value of 3) in a sea of FALSES (every other number without a value of 3) very tedious and inefficient. Secondly, the accuracy of visual inspection for a large vector is questionable. Using logical subsetting operators (such as [[ ]]) within the code to rather tell R to select elements with the value of 3 instead of by visual inspection is much more accurate, efficient and reproducible.

Logical Subsetting II

3. **Why didn't you always get the same count of `3` entries each time?**

   R generated a different count of 3 entries each time because the sample() function was used. The sample function randomly reordered integers 1-12, but since the size was set as n, which in this case equals 10, two numbers will always be left out each time the code is run.

4. **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 safe way to select entries with a value of 3 because it will assure that entries with a value of 3 will be selected as TRUE, especially in situations where random sampling occurs (/sample () is used) and the size of samples is less than the range of integers.

5. **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.**

   Logical 'by hand' subsetting is a bad practice because it is inefficient, time consuming, lacks ensured accuracy and not very friendly in terms of reusability of code with other collaborators. With small datasets, where looking at all the elements at once is manageable/non overwhelming, visual inspection or by hand subsetting is possible. However, with larger datasets using logical subsetting operators in the code is a much more reliable solution to selecting the targeted elements you want. By writing code using these subsetting operators, it allows for reusability and shareability with others. For circumstances where the subsetting code needs to be shared, it should be complete and self-contained such that the name of the of data-frames are comprehensive and reproducible in a different/new R workspace environment. Performing logical 'by hand' subsetting does not sustain this same type of reproducibility and accuracy.

Basic Loops

6. **Provide the code for your modified loop. It must run as a self-contained example on a fresh R session on my computer.**

   I used the following code for my modified loop:

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

Intermediate Loops

7. **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.**

   I used the following code for the modified loop that executes *n* times:

```
n=5

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

Intermediate Loops 2

8. **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.**

   I used the following code to create the n, vec_1 and the loop:

```
n=17
vec_1=sample(10,n, replace=TRUE)
for(i in 1:n)
 {print(
    paste0("The element of vec_1 at index: "  ,i,  " is ",
         vec_1[i]))
}
```

Functions

9. **Provide the code you used to build your function.**

   I used the following code to build the function named create_and_print_vec()

```
create_and_print_vec = function( n, min =1 , max =10)
{
 my_random_vec=sample(min:max, n, replace=TRUE)

 for (i in 1:n)
 {print(
  paste0("The element at index: "  ,i, "  is  ",
       my_random_vec[i]))
 }
}
create_and_print_vec(n=5, min=1, max=10)
```