Assignment 2: Coding Basics

Meilin Chan

OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics on coding basics.

Directions

1. Change "Student Name" on line 3 (above) with your name.
2. Work through the steps, **creating code and output** that fulfill each instruction.
3. Be sure to **answer the questions** in this assignment document.
4. When you have completed the assignment, **Knit** the text and code into a single PDF file.
5. After Knitting, submit the completed exercise (PDF file) to the dropbox in Sakai. Add your first and last name into the file name (e.g., "FirstLast_A02_CodingBasics.Rmd") prior to submission.

Basics Day 1

1. Generate a sequence of numbers from one to 100, increasing by fours. Assign this sequence a name.

```
one_hundred <- seq(1,100,4)
one_hundred
```

```
##  [1]  1  5  9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93 97
```

2. Compute the mean and median of this sequence.

```
mean_one <- mean(one_hundred)
mean_one
```

```
## [1] 49
```

```
median_one <- median(one_hundred)
median_one
```

```
## [1] 49
```

3. Ask R to determine whether the mean is greater than the median.

```
question <- function(x) {
  if(x>median_one) {
    return(TRUE)
  }
  else {
    return(FALSE)
  }
}

question(mean_one)
```

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

```
question(70)
```

```
## [1] TRUE
```

4. Insert comments in your code to describe what you are doing.

*1. Using the seq(x,y,z) I am establishing a sequence of numbers from x=1 to y=100 increasing by z=4. I then assigned this sequence to a name "one_hundred"

*2. Using functions mean(one_hundred) and median(one_hundred) I can see the mean and median of the sequence I created in #1

*3. I created a function assigned to the name "question" - in my function I set an "if else" statement where if the input into the function is greater than median_one, then TRUE is returned, else FALSE is returned. To test this out, I input mean_one into my function and since mean_one is NOT greater than median_one, FALSE was returned. However, if I input the number 70 into the question function, then TRUE is returned.

Basics Day 2

5. Create a series of vectors, each with four components, consisting of (a) names of students, (b) test scores out of a total 100 points, and (c) whether or not they have passed the test (TRUE or FALSE) with a passing grade of 50.

```
names <- c("Nicholas","Toddi","Luana","John")
scores <- c(20,65,100,98)
pass <- c(FALSE,TRUE,TRUE,TRUE)
```

6. Label each vector with a comment on what type of vector it is.

```
class(names) #character
```

```
## [1] "character"
```

```
class(scores) #numeric
```

```
## [1] "numeric"
```

```
class(pass) #logical
```

```
## [1] "logical"
```

7. Combine each of the vectors into a data frame. Assign the data frame an informative name.

```
student_grades <- data.frame("student"=names,"score"=scores,"status"=pass)
student_grades
```

```
##    student score status
## 1 Nicholas    20  FALSE
## 2    Toddi    65   TRUE
## 3    Luana   100   TRUE
## 4     John    98   TRUE
```

8. Label the columns of your data frame with informative titles.

```
#see above
```

9. QUESTION: How is this data frame different from a matrix?

   Answer: This is a data frame because it contains columns of different types of information. There are character, numeric, and logical expressions. In a matrix, it would only contain one type of information (ie: only numeric)

10. Create a function with an if/else statement. Your function should determine whether a test score is a passing grade of 50 or above (TRUE or FALSE). You will need to choose either the `if` and `else` statements or the `ifelse` statement. Hint: Use `print`, not `return`. The name of your function should be informative.

11. Apply your function to the vector with test scores that you created in number 5.

```
pass_fail <- function(x) {
  if(x>50) {
    return(TRUE)
  }
  else {
    return(FALSE)
  }
}

pass_fail(student_grades$score)
```

```
## Warning in if (x > 50) {: the condition has length > 1 and only the first
## element will be used
```

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

```
pass_fail2 <- function(x) {
  ifelse(x>=50,TRUE,FALSE)
}
pass_fail2(student_grades$score)
```

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

12. QUESTION: Which option of `if` and `else` vs. `ifelse` worked? Why?

    Answer: 'ifelse' worked, but 'if and else' did not. I believe this is because the number of factors I was trying to input into my pass_fail function was greater than 1 so this 'if and else' set up could not process all these inputs. However, with my pass_fail2 statement, this 'ifelse' statement could process more than 1 input so it was able to process each score for each student and determine if the student passes or not.