Assignment 2: Coding Basics

David Amanfu

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.
2. Compute the mean and median of this sequence.
3. Ask R to determine whether the mean is greater than the median.
4. Insert comments in your code to describe what you are doing.

For completeness/thoroughness/clarity I doublechecked help for seq, and took note of the syntax seq(from,to,by,...), and viola. Then I assigned it to the variable name "onemodfour" as in all of these numbers in the sequence are equivalent to one mod four.

```
#1.
onemodfour <- seq(1,100,4) # Create Sequence
onemodfour #Call sequence
```

```
##  [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.
mean_omf <- mean(onemodfour) # save the mean
med_omf <- median(onemodfour) #save the median
mean_omf # call the mean
```

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

```
med_omf  #call the median
```

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

```
#3.
mean_omf > med_omf # is the mean greater than the median?
```

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

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.

So I get that there should be four apiece but I'm going with six.

```
studname <- c("Daisy","Daniel","Dottie","Yakko","Wakko","Dot")
tstscr <- ceiling(rnorm(6,62,13)) #
passed <- tstscr > 50
```

Here I definitely could have used the following for "tstscr":

```
tstscr2 <- sample(0:100,6,replace=FALSE)  #using tstscr2 for Q6 followup
#or
floor(runif(6,0,100))
```

```
## [1] 70 13  3 92 91 13
```

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

So here I'm unclear if we're supposed to assign a label to the vector or just add a comment to our code?

```
typeof(studname) #Returns Character
```

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

```
typeof(tstscr) #Returns Double
```

```
## [1] "double"
```

```
typeof(tstscr2) #Returns Integer
```

```
## [1] "integer"
```

```
typeof(passed) #Returns Logical
```

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

7. Combine each of the vectors into a data frame. Assign the data frame an informative name. We can do this in one step:

```
testdf <- data.frame("Name"=studname,"Score"=tstscr,"Passing"=passed,row.names=studname)
testdf
```

```
##          Name Score Passing
## Daisy   Daisy    71    TRUE
## Daniel Daniel    74    TRUE
## Dottie Dottie    71    TRUE
## Yakko   Yakko    45   FALSE
## Wakko   Wakko    63    TRUE
## Dot       Dot    67    TRUE
```

8. Label the columns of your data frame with informative titles. Or in two steps with:

```
testdf2 <- data.frame(studname,tstscr,passed)
names(testdf2) <- c("Name", "Score", "Passed")
testdf2
```

```
##     Name Score Passed
## 1  Daisy    71   TRUE
## 2 Daniel    74   TRUE
## 3 Dottie    71   TRUE
## 4  Yakko    45  FALSE
## 5  Wakko    63   TRUE
## 6    Dot    67   TRUE
```

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

   Answer: The difference is that each column can have a different data type rather thna being forced to be the same across all columns. Apparently in a matrix you can have a bunch of integers and no characters, or a bunch of characters and no integers, but not both. In comes the data frame.

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.

```
didtheyfail <- function(x){
  passornah <- passed
  for (i in 1:6) {
    if(x[i]>50){
      passornah[i] <- "Pass"
    }
```

```
    else{
      passornah[i] <- "Fail"
    }
  }
  print(passornah)
}
```

or more simply:

```
didtheyfail2 <- function(x){
  ifelse(x>50,"Pass","Fail")
}
```

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

```
passornot <- didtheyfail2(tstscr)
passornot
```

```
## [1] "Pass" "Pass" "Pass" "Fail" "Pass" "Pass"
```

```
passornah <- didtheyfail(tstscr)
```

```
## [1] "Pass" "Pass" "Pass" "Fail" "Pass" "Pass"
```

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

    Answer: Both worked in this case. I had to employ a for loop for the first function to work, but they both got there eventually. No clue, but my intuition was that if and else has a harder time interpreting lists/vectors as opposed to items with length > 1 or more than one element.