Assignment 2: Coding Basics

Eva May

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.

```
#1.
cent <- seq(1, 100, 4) #cent is name of sequence
#(from, to, by)
cent
```

```
##  [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(cent); median(cent)
```

```
## [1] 49

## [1] 49
```

```
#mean of created sequence and median of created sequence
#; separates 2 commands like separate lines of code would

#3.
mean(cent) > median(cent)
```

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

```
#using > returns T/F statement about the computed numbers from the 2 commands
```

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.
6. Label each vector with a comment on what type of vector it is.
7. Combine each of the vectors into a data frame. Assign the data frame an informative name.
8. Label the columns of your data frame with informative titles.

```
#5 and 6

name <- c("Anna", "Jeff", "Zoe", "Bob")
name
```

```
## [1] "Anna" "Jeff" "Zoe"  "Bob"
```

```
typeof(name)
```

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

```
#character vector - contains a list of names

score <- c(65, 40, 80, 70)
score
```

```
## [1] 65 40 80 70
```

```
typeof(score)
```

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

```
#double numeric vector - automatic kind of vector created for integers is
#a double-precision numerical one

passed <- function(x){
  ifelse(x>49, TRUE, FALSE)
}
#created a function with ifelse statement to describe if the test scores
#in score passed - making this vector correspond with vector score
pass <- c(passed(score))
pass
```

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

```
typeof(pass)
```

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

```
#logical vector - contains logical values of T and F



#7

student.scores <- data.frame(name, score, pass)
#creating data frame - each vector becomes a column

#8
```

```r
colnames(student.scores)
```

```
## [1] "name"  "score" "pass"
```

```r
#getting current column names
names(student.scores)[names(student.scores) == "name"
                      ] <- "Student Name"
names(student.scores)[names(student.scores) == "score"
                      ] <- "Test Score"
names(student.scores)[names(student.scores) == "pass"
                      ] <- "Student Passed?"
#changing each column name individually

student.scores
```

```
##   Student Name Test Score Student Passed?
## 1         Anna         65            TRUE
## 2         Jeff         40           FALSE
## 3          Zoe         80            TRUE
## 4          Bob         70            TRUE
```

```r
#returning data frame to make sure names changed
```

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

   Answer: Each column in this data frame contains a different type of data - character, numeric, and logical. A matrix would not allow this, and would require that all data be the same type (e.g. 3 character columns).

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.

    Answer [oops sorry - kind of did this above so will just adapt that one]

```r
passing.score <- function(x){
  ifelse(x>49, TRUE, FALSE)
}
#returns true if x>49 and false if x<=49
#did not include 'print' here as function automatically returns either TRUE or FALSE.
#including 'print' is redundant and causes function to print TRUE and FALSE
#in addition to the logic responses related to each x.
```

11. Apply your function to the vector with test scores that you created in number 5. > [again sorry I did this above..hope that's okay!]

```r
passing.score(score)
```

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

```r
#applying fxn to vector of test scores

passpass <- function(x){
  if(x>49) TRUE
  else FALSE
}
#creating if and else fxn to see if it will also work (for answer to #12)
passpass(score)
```

```
## Warning in if (x > 49) TRUE else FALSE: the condition has length > 1 and only
## the first element will be used
```

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

```
#error because score is a vector with length >1
passpass(2)
```

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

```
#no error bc only 1 numerical value is input
```

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

    Answer: I used ifelse rather than the longer if and else, but I initially thought both would work because they do the same thing - I just chose to use the quicker one. However, in the block of code above, I tried creating an if and else function as well (passpass), which R accepts as a function but won't run on the vector, as it seems to only work with single numerical values, e.g. scalars (and ifelse are designed more for vectors with length > 1). So, I would say that the ifelse function works for this scenario because we are running it on a vector. You'll also see in the code above that if passpass is run on a single numerical value, R is fine with it, so if score were a vector with only 1 value, either function would work.