Assignment 2: Coding Basics

Sam Vanasse

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.

```
hundred.seq <- seq(1, 100, 4)
hundred.seq
```

```
##  [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(hundred.seq)
```

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

```
median(hundred.seq)
```

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

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

```
mean(hundred.seq) < median(hundred.seq)
```

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

```
mean(hundred.seq) > median(hundred.seq)
```

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

```
mean(hundred.seq) != median(hundred.seq)
```

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

```
mean(hundred.seq) == median(hundred.seq)
```

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

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

```
#1. Generating a sequence from 1 to 100, increasing the values by 4 and naming the sequence hundred.seq

#2. Generating the mean and median of hundred.seq

#3. Demonstration that the mean and the medians are equal by showing the median is not greater than tha
```

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.

```
vector.names <- c("Carl", "William", "Matt", "Henry") #Character
vector.test.scores <- c(90,60,40,80) #Numeric
vector.pass.fail <- c(TRUE,TRUE,FALSE,TRUE) #Logical

student.dataframe <- cbind(vector.names,vector.test.scores,vector.pass.fail)
student.dataframe <- data.frame("Name"=vector.names,"Score"=vector.test.scores,"Passed"=vector.pass.fai
student.dataframe
```

```
##      Name Score Passed
## 1    Carl    90   TRUE
## 2 William    60   TRUE
## 3    Matt    40  FALSE
## 4   Henry    80   TRUE
```

```
class(vector.names)
```

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

```
class(vector.test.scores)
```

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

```
class(vector.pass.fail)
```

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

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

   Answer: A matrix requires equal objects in the rows and columns whereas a dataframe like above can have more rows than columns or vice-versa.

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.

```
#Option 1
ifelse.test.score <- function(x){
  if(x<=49){"FAIL"}
  else {"PASS"}
}
```

```r
score1 <- ifelse.test.score(90); score1
```

```
## [1] "PASS"
```

```r
score2 <- ifelse.test.score(60); score2
```

```
## [1] "PASS"
```

```r
score3 <- ifelse.test.score(40); score3
```

```
## [1] "FAIL"
```

```r
score4 <- ifelse.test.score(80); score4
```

```
## [1] "PASS"
```

```r
score.all <- ifelse.test.score(vector.test.scores); score.all #Intentionally running an error to prove
```

```
## Warning in if (x <= 49) {: the condition has length > 1 and only the first
## element will be used
```

```
## [1] "PASS"
```

```r
#Option 2
ifelse.test.score2 <- function(x){
  ifelse(x>50, "PASS", "FAIL")
}
ifelse.score1 <- ifelse.test.score2(90); ifelse.score1
```

```
## [1] "PASS"
```

```r
ifelse.score2 <- ifelse.test.score2(60); ifelse.score2
```

```
## [1] "PASS"
```

```r
ifelse.score3 <- ifelse.test.score2(40); ifelse.score3
```

```
## [1] "FAIL"
```

```r
ifelse.score4 <- ifelse.test.score2(80); ifelse.score4
```

```
## [1] "PASS"
```

```r
ifelse.score.all <- ifelse.test.score2(vector.test.scores); ifelse.score.all #Only this option works fo
```

```
## [1] "PASS" "PASS" "FAIL" "PASS"
```

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

    Answer: In order to use the if-else statement on all test scores in the vector, only the 'ifelse' statement will work.