Assignment 2: Coding Basics

Li Jia Go

OVERVIEW

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

Directions

1. Rename this file `<FirstLast>_A02_CodingBasics.Rmd` (replacing `<FirstLast>` with your first and last name).
2. Change "Student Name" on line 3 (above) with your name.
3. Work through the steps, **creating code and output** that fulfill each instruction.
4. Be sure to **answer the questions** in this assignment document.
5. When you have completed the assignment, **Knit** the text and code into a single PDF file.
6. After Knitting, submit the completed exercise (PDF file) to Sakai.

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. using sequence function to generate sequence of numbers from 1 to 100
# increasing by fours. i.e. seq (from, to, by)
seq(1,100,4)
```

```
##  [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
```

```
# generating sequence from 1 to 100, increasing by 4. naming sequence
# 'hundredseq'
hundredseq <- seq(1,100,4)

# return result
hundredseq
```

```
##  [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.
# computing the mean of the sequence using the mean function and assigning a
# name to the result
hundredseq_mean <- mean(hundredseq)

# return result
hundredseq_mean
```

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

```
# computing the median of the sequence using the median function and assigning a
# name to the result
hundredseq_median <- median(hundredseq)
hundredseq_median # return result
```

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

```
#3.
hundredseq_mean > hundredseq_median # asking R if mean is greater than median
```

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

```
hundredseq_mean < hundredseq_median # asking R if mean is less than median
```

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

```
hundredseq_mean == hundredseq_median # asking R if mean and median are equal
```

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

```
hundredseq_mean != hundredseq_median # asking R if mean and median are not equal
```

```
## [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.
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.
#creating a vector of student names
student_names <- c ("John", "Peter", "Sarah", "Jill")
student_names #return result
```

```
## [1] "John"  "Peter" "Sarah" "Jill"
```

```
#creating a vector of student's scores
student_scores <- c(80, 90, 40, 48)
student_scores #return result
```

```
## [1] 80 90 40 48
```

```
#creating a vector showing whether students have passed the test or not
student_pass <- c(TRUE, TRUE, FALSE, FALSE)
student_pass #return result
```

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

```
#6.
class(student_names) #character
```

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

```
class(student_scores) #numeric
```

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

```r
class(student_pass) #logical
```

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

```r
#7.
# transforming each vector into a data frame
df_classnames <- as.data.frame(student_names)
df_classscores <- as.data.frame(student_scores)
df_classpass <- as.data.frame(student_pass)

# combining all the columns of the data frame
df_classtestscores <- cbind(df_classnames, df_classscores, df_classpass)
df_classtestscores #print result
```

```
##   student_names student_scores student_pass
## 1          John             80         TRUE
## 2         Peter             90         TRUE
## 3         Sarah             40        FALSE
## 4          Jill             48        FALSE
```

```r
#8.
# renaming names of columns of data frame
colnames(df_classtestscores) <- c("names", "scores", "passed")
df_classtestscores #return result
```

```
##   names scores passed
## 1  John     80   TRUE
## 2 Peter     90   TRUE
## 3 Sarah     40  FALSE
## 4  Jill     48  FALSE
```

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

   Answer: Matrices can only store a single data type, while this data frame consists of different classes of data (i.e. characters, numbers, logic etc)

10. Create a function with an if/else statement. Your function should take a **vector** of test scores and print (not return) whether a given 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.

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

```r
#creating a function to see if given test score is of a passing grade or not
passfifty<- function(x) {
ifelse(x >50, "TRUE", "FALSE")
}

answer<-passfifty(student_scores)
print(answer) #print result
```

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

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

    Answer: Only 'ifelse' worked. The if() statement can only work with elements that are length 1, i.e. a single student. However, by using the 'ifelse' expression, we are able to check if the entire class of students passed the test at once.