Assignment 2: Coding Basics

Keanu Valibia

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, Part 1

1. Generate a sequence of numbers from one to 30, increasing by threes. 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.

# Generate a sequence of numbers, 1 - 30, skipping every 3 numbers
genseq <- seq(1,30,3)
genseq
```

```
##  [1]  1  4  7 10 13 16 19 22 25 28
```

```
#2.

# Use the mean() function to generate mean of my genseq object.
#Store this value as a separate object.
mean_genseq <- mean(genseq)

# Show mean value
mean_genseq
```

```
## [1] 14.5
```

```
# Use the median() function to generate median of my genseq object.
#Store this value as a separate object.
med_genseq <- median(genseq)

# Show median value
med_genseq
```

```
## [1] 14.5
```

```
#3.

# Compare mean_genseq and med_genseq. Prints TRUE or FALSE value.
mean_genseq > med_genseq
```

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

Basics, Part 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.

```
# Create vector that holds student names
studentnames <- c("Keanu","Jay","Camila","Chris")
# Print vector
studentnames
```

```
## [1] "Keanu"  "Jay"    "Camila" "Chris"
```

```
# Create vector that holds test scores
test_scores <- c(100,25,50,49)
# Print vector
test_scores
```

```
## [1] 100  25  50  49
```

```
# Create vector that holds logical values
test_results <- c(TRUE,FALSE,TRUE,FALSE)
# Print vector
test_results
```

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

```
# Create data frame, combining above three vectors
df_student_test_results <- data.frame("StudentName" = studentnames,
  "TestScore" = test_scores,
  "TestResult" = test_results)

# Call data frame I just created
df_student_test_results
```

```
##   StudentName TestScore TestResult
## 1       Keanu       100       TRUE
## 2         Jay        25      FALSE
## 3      Camila        50       TRUE
## 4       Chris        49      FALSE
```

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

   Answer: A data frame allows for multiple classes to be held in the same object, whereas a matrix only allows usage of a single type of class.

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.

```
# Create function using 'ifelse' logic
resultcheck <- function(result) {
  # If 'result < 50' is TRUE, print "FALSE...."
  print(ifelse(result < 50, FALSE, TRUE))  # ...otherwise, print "TRUE"
}

for (i in length(test_scores)) { # For every value in 'test_scores' vector...
  resultcheck(test_scores) #...applies 'resultcheck' function and print results
}
```

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

```
resultcheck2 <- function(result) {
  if (result < 50) { # If logic check: if result is less than 50...
    print(FALSE) # ..print FALSE
  }
  else { # else: if result is 50 or larger...
    print(TRUE) # ...print TRUE
  }
}

#  for (i in length(test_scores)) { For every value in 'test_scores' vector...
#   resultcheck2(test_scores) Applies 'resultcheck' function and print results
# }
#
# But this fails because it only tests for the first component in the vector
```

```r
# Below logic resolves the error by explicitly asking to test individual
# components in 'test_scores' until it reaches end of length
for (i in 1:length(test_scores)) { # For every component in 'test_scores'...
  resultcheck2(test_scores[i]) # Applies 'resultcheck' and print results
}
```

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

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

    Answer: The 'ifelse' statement worked. The 'if' logic seems to only check for the first value in the vector, and returns an error message beyond the first component. 'ifelse' is a vectorized function, and so is able to automatically test each component within a vector.