Project 1 – R Practice
ALY 6000

Project Instructions

In this project, you will use an R Markdown template file. Put all your code and visualizations in it and submit the file. Be sure to follow all formatting guidelines.

Setting Up Your Project

1. Download and open the R Markdown file “LastName-FirstName-Project1.Rmd.” Replace LastName and FirstName with your own last name and first name.

2. Run the following code at the very top of the file. This will clear out the environment each time you run your entire code and prevent past actions from interfering with current work.

```
cat("\014")  # clears console
rm(list = ls())  # clears global environment
try(dev.off(dev.list()["RStudioGD"]), silent = TRUE) # clears plots
try(p_unload(p_loaded(), character.only = TRUE), silent = TRUE) #
clears packages
options(scipen = 100) # disables scientific notation for entire R
session
```

3. Install the *pacman* package. This is a simple, user-friendly package that makes installing and loading other packages a one-line process.

```
# You should do this line only once in the entire course.
install.packages("pacman")
# Once you have done the install line, the following line is what you
will always need to do to utilize the pacman package in R
library(pacman)
```

4. For each question, write the code to answer it within its own cell in the file. All outputs and visualizations should appear underneath the cell.

Testing Your Solution

The file project1_tests.R contains a set of test cases which will affirm that some of the problems you have solved are correct. To use this file, save it into the same folder as your R Markdown file.

In your R Markdown file (named LastName-FirstName-Project1.Rmd), the following code will check your answers using the file project1_tests.R.

```
library(pacman)
p_load(testthat)
test_file("project1_tests.R")
```

Tests can be run at any time. This code will report the total number of tested items that currently succeed as well as any errors preventing you from passing others. A single problem may have multiple tests while others have none. You should be able to complete this project with 100% accuracy on all tests. Use the exact names of variables stated in each problem, pay extra attention to matching any upper and lower case letters.

Project 1 Instructions

Complete the following problems in the R script you created above. Be sure to name specified variables **exactly** as they are given, paying special attention to spellings and the use of upper and lower case letters.

1. Write lines of code to compute all of the following. Run the code. The results should appear under the cell for question 1 in the R Markdown file.

```
123 * 453
5^2 * 40
TRUE & FALSE
TRUE | FALSE
75 %% 10
75 / 10
```

2. Create a vector using the **c** function with the values 17, 12, -33, 5 and assign it to a variable called **first_vector**.

```
[1]  17  12 -33   5
```

3. Create a vector using the **c** function with the values 5, 10, 15, 20, 25, 30, 35 and assign it to a variable called **counting_by_fives**.

4. Create a vector using the **seq** function containing every even number between 10 and 30 inclusive and assign it to a variable called **second_vector**.

```
 [1] 10 12 14 16 18 20 22 24 26 28 30
```

5. Create a vector using the seq function containing the values 5, 10, 15, 20, 25, 30 , 35 and assign it to a variable called **counting_by_fives_with_seq**.

```
[1]  5 10 15 20 25 30 35
```

6. Create a vector using the function **rep** and provide it with **first_vector** as its first argument and 10 as its second argument. Assign the result to a variable called **third_vector**.

```
 [1]  17  12 -33   5  17  12 -33   5  17  12 -33   5  17  12 -33   5  
17  12 -33
[20]   5  17  12 -33   5  17  12 -33   5  17  12 -33   5  17  12 -33   
5  17  12
[39] -33   5
```

7. Using the **rep** function, create a vector containing the number zero, 20 times. Store the result in a variable called **rep_vector**.

```
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

8. Create a vector using the range operator (the colon), that contains the numbers from 10 to 1 . Store the result in a variable called **fourth_vector**.

```
 [1] 10  9  8  7  6  5  4  3  2  1
```

9. Create a vector using the range operator that contains the numbers from 5 to 15. Store the result in a variable called **counting_vector**.

```
 [1]  5  6  7  8  9 10 11 12 13 14 15
```

10. Create a vector with the values (96, 100, 85, 92, 81, 72) and store it in a variable called **grades.**

```
[1]  96 100  85  92  81  72
```

11. Add the number 3 to the vector **grades**. Store the result in a variable called **bonus_points_added.**

```
[1]  99 103  88  95  84  75
```

12. Create a vector with the values 1 – 100. Store it in a variable called **one_to_one_hundred**. Do not type out all 100 numbers.

```
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  
17  18
 [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34
```

```
35  36
 [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52
53  54
 [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70
71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88
89  90
 [91]  91  92  93  94  95  96  97  98  99 100
```

13. Create a vector with values from 100 to -100 by 3s. Store the result in a variable called **reverse_numbers.** To clarify, the first 3 numbers in this vector will be (100, 97, 94…)

```
 [1] 100  97  94  91  88  85  82  79  76  73  70  67  64  61  58  55
52  49  46
[20]  43  40  37  34  31  28  25  22  19  16  13  10   7   4   1  -2  -
5  -8 -11
[39] -14 -17 -20 -23 -26 -29 -32 -35 -38 -41 -44 -47 -50 -53 -56 -59 -
62 -65 -68
[58] -71 -74 -77 -80 -83 -86 -89 -92 -95 -98
```

14. Write each of the following lines of code. Add a one-sentence comment above each line explaining what is happening. Include your comments in the R Markdown file.

```
second_vector  +  20
second_vector  *  20
second_vector >= 20
second_vector != 20  # != means "not equal"
```

15. Using the built in **sum** function, compute the sum of **one_to_one_hundred** and store it in a variable called **total.**

```
[1] 5050
```

16. Using the built in **mean** function, compute the average of **one_to_one_hundred** and store the result in a variable called **average_value**.

```
[1] 50.5
```

17. Using the built in **median** function, compute the average of **one_to_one_hundred** and store the result in a variable called **median_value.**

```
[1] 50.5
```

18. Using the built in **max** function, compute the average of **one_to_one_hundred** and store the result in a variable called **max_value.**

```
[1] 100
```

19. Using the built in **min** function, compute the average of **one_to_one_hundred** and store the result in a variable called **min_value.**

```
[1] 1
```

20. Using brackets, extract the first value from **second_vector** and store it in a variable called **first_value.**

```
[1] 10
```

21. Using brackets, extract the first, second and third values from **second_vector** and store it in a variable called **first_three_values.**

```
[1] 10 12 14
```

22. Using brackets, extract the 1st, 5th, 10th, and 11th elements of **second_vector**. Store the resulting vector in a variable called **vector_from_brackets**.

```
[1] 10 18 28 30
```

23. Use the brackets to extract elements from the first_vector using the following vector **c(FALSE, TRUE, FALSE, TRUE)**. Store the result in a variable called **vector_from_boolean_brackets**. Explain in a comment what happens.

```
[1] 12  5
```

24. Examine the following piece of code and write a one-sentence comment explaining what is happening.

```
second_vector >= 20
```

25. Examine the following piece of code and write a one-sentence comment explaining what is happening.

```
ages_vector <- seq(from = 10, to = 30, by = 2)
```

26. Examine the following piece of code and write a one-sentence comment explaining what is happening, assuming **ages_vector** was computed in the previous problem.

```
ages_vector [ages_vector >= 20]
```

27. Using the same approach as the previous question, create a new vector by removing from the **grades** vector all values lower than or equal to 85. Store the new vector in a variable called **lowest_grades_removed**.

```
[1]  96 100  85  92
```

28. Use the **grades** vector to create a new vector with the 3rd and 4th elements of **grades** removed. Store this in a variable called **middle_grades_removed**. Try utilizing a vector of negative indexes to complete this task.

```
[1]  96 100  81  72
```

29. Use bracket notation to remove the 5th and 10th elements of **second_vector**. Store the result in a variable called **fifth_vector**.

```
[1] 10 12 14 16 20 22 24 26 30
```

30. Write the following code. Explain in a comment what you think the code is doing.

```r
set.seed(5)
random_vector <- runif(n=10, min = 0, max = 1000)
```

31. Use the **sum** function to compute the total of **random_vector**. Store the result in a variable called **sum_vector**.

```
[1] 5295.264
```

32. Use the **cumsum** function to compute the cumulative sum of **random_vector**. Store the result in a variable called **cumsum_vector**.

```
 [1]  200.2145  885.4330 1802.3088 2086.7083 2191.3584 2892.4159
3420.3759
 [8] 4228.3111 5184.8112 5295.2642
```

33. Use the **mean** function to compute the mean of **random_vector**. Store the result in a variable called **mean_vector**.

```
[1] 529.5264
```

34. Use the **sd** function to compute the standard deviation of **random_vector**. Store the result in a variable called **sd_vector**.

```
[1] 331.3606
```

35. Use the **round** function to round the values of **random_vector**. Store the result in a variable called **round_vector**.

```
 [1] 200 685 917 284 105 701 528 808 957 110
```

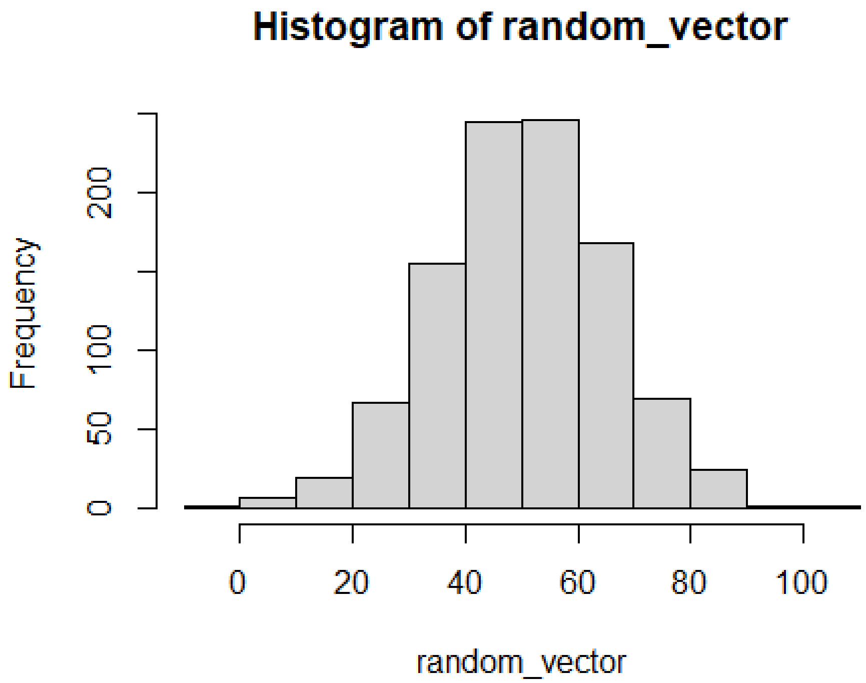
36. Use the **sort** function to sort the values of **random_vector.** Store the result in a variable called **sort_vector**.

```
 [1] 104.6501 110.4530 200.2145 284.3995 527.9600 685.2186 701.0575
807.9352
 [9] 916.8758 956.5001
```

37. Consider the following code. Explain in a comment what you think the code is doing.

```r
set.seed(5)
random_vector <- rnorm(n=1000, mean = 50, sd = 15)
```

38. Use the **hist** function and provide it with **random_vector.** Explain the result in a comment.



39. Download the datafile **ds_salaries.csv** from Canvas. Save it on your computer in the same folder (directory) where your R Markdown file for this project is located.

40. Using the **p_load** function, load the tidyverse set of libraries. Pacman will both install and load the tidyverse libraries.

```
p_load(tidyverse)
```

41. The tidyverse has a function called **read_csv** to read files when you specify the filename. Store the result of the read into a variable called **first_dataframe**.

42. Try each of the following blocks of code. Add a one-sentence comment describing what you believe is happening.

```
head(first_dataframe)

head(first_dataframe, n = 7)

names(first_dataframe)

smaller_dataframe <- select(first_dataframe, job_title, salary_in_usd)

smaller_dataframe
```

```
better_smaller_dataframe <- arrange(smaller_dataframe,
desc(salary_in_usd))

better_smaller_dataframe

better_smaller_dataframe <- filter(smaller_dataframe, salary_in_usd >
80000)

better_smaller_dataframe

better_smaller_dataframe <-
  mutate(smaller_dataframe, salary_in_euros = salary_in_usd * .94)

better_smaller_dataframe

better_smaller_dataframe <- slice(smaller_dataframe, 1, 1, 2, 3, 4, 10,
1)

better_smaller_dataframe

ggplot(better_smaller_dataframe) +
  geom_col(mapping = aes(x = job_title, y = salary_in_usd), fill =
"blue") +
  xlab("Job Title") + ylab("Salary in
  US Dollars") + labs(title =
  "Comparison of Jobs ") +
  scale_y_continuous(labels = scales::dollar) +
  theme(axis.text.x = element_text(angle = 50, hjust = 1))
```

Submitting to Canvas

When you are satisfied with your solution, take the following steps:

1. **Remove** any lines in your code with “**install.packages**.”
2. **Remove** any lines in your code that use the **view** function.
3. Submit your R Markdown file under the appropriate assignment in Canvas:

Congratulations on completing your first R project!