# WEEK – 1

**ALY6000 – Introduction to Analytics**

# PROJECT - 1 NORTHEASTERN UNIVERSITY

A red and white logo

Description automatically generated

**College of Professional Studies, Northeastern University, Boston, MA 02215**

# Submitted By

**Name:** Rohit Kosamkar

**NUID:** 002478862

**Dated:** January 14, 2024

# PROJECT 1

1. Write lines of code to compute all the following. Include the answers in your  
   written report.

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

Answer:

#1:

print(123 \* 453)

print(5^2 \* 40)

print(TRUE & FALSE)

print(TRUE | FALSE)

print(75 %% 10)

print(75/10)

[1] 55719

[1] 1000

[1] FALSE

[1] TRUE

[1] 5

[1] 7.5

1. Create a vector using the c function with the values 17, 12, -33, 5 and assign it to a  
   variable called first\_vector

Answer:

#2.

first\_vector <- c(17, 12, -33, 5)

print(first\_vector)

[1] 17 12 -33 5

1. 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

Answer:

#3.

counting\_by\_fives <- c(5,10,15,20,25,30,35)

print(counting\_by\_fives)

[1] 5 10 15 20 25 30 35

1. Create a vector using the range operator (the colon), that contains the numbers  
   from 20 down to 1 . Store the result in a variable called second\_vector.

Answer:

#4.

second\_vector <- 20:1

print(second\_vector)

[1] 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

1. Create a vector using the range operator that contains the number from 5 to 15.  
   Store the result in a variable called counting\_vector

Answer:

#5.

counting\_vector <- 5:15

print(counting\_vector)

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

1. Create a vector with the values (96, 100, 85, 92, 81, 72). Store the result in a variable  
   called grades

Answer:

#6.

grades <- c(96,100,85,92,81,72)

print(grades)

[1] 96 100 85 92 81 72

1. Add the number 3 to the vector grades. Store the result in a variable called  
   bonus\_points\_added.

Answer:

#7.

bonus\_points\_added <- grades + 3

print(bonus\_points\_added)

[1] 99 103 88 95 84 75

1. Create a vector with the values 1 – 100 and store it in a variable called  
   one\_to\_one\_hundred. Do not type out all 100 numbers.

Answer:

#8.

one\_to\_one\_hundred <- 1:100

print(one\_to\_one\_hundred)

[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

1. Write each of the following lines of code. Add a one-sentence comment above each  
   line explaining what is computed. Include your comments in the written report.  
   second\_vector + 20  
   second\_vector \* 20  
   second\_vector >= 20  
   second\_vector != 20 # != means "not equal"

Answer:

#9.1 adding 20 to each element in second\_vector

print (second\_vector + 20)

[1] 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21

#9.2 multiplying by 20 to each element in second vector

print(second\_vector \* 20)

[1] 400 380 360 340 320 300 280 260 240 220 200 180 160 140 120 100 80 60

[19] 40 20

#9.3 This code gives a Boolean value output where elements in second\_vector which are greater than equal to 20 are ‘TRUE’ otherwise ‘FALSE’

print(second\_vector >= 20)

[1] TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

#9.4 This code gives a Boolean value output where elements in second\_vector which are not equal to 20 are ‘TRUE’ otherwise ‘FALSE’

print(second\_vector != 20)

[1] FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

[13] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

1. Using the built in sum function, compute the sum of one\_to\_one\_hundred. Store  
   the result in a variable called total.

Answer:

#10.

total <- sum(one\_to\_one\_hundred)

print(total)

[1] 5050

1. Using the built in mean function, compute the average of one\_to\_one\_hundred.  
   Store the result in a variable called average\_value

Answer:

#11.

average\_value <- mean(one\_to\_one\_hundred)

print(average\_value)

[1] 50.5

1. Using the built in median function, compute the average of one\_to\_one\_hundred.  
   Store the result in a variable called median\_value

Answer:

#12.

median\_value <- median(one\_to\_one\_hundred)

print(median\_value)

[1] 50.5

1. Using the built in max function, compute the max of one\_to\_one\_hundred. Store  
   the result in a variable called max\_value

Answer:

#13.

max\_value <- max(one\_to\_one\_hundred)

print(max\_value)

[1] 100

1. Using the built in min function, compute the min of one\_to\_one\_hundred. Store the  
   result in a variable called min\_value

Answer:

#14.

min\_value <- min(one\_to\_one\_hundred)

print(min\_value)

[1] 1

1. Using brackets, extract the first value from second\_vector and store it in a variable  
   called first\_value

Answer:

#15.

first\_value <- second\_vector[1]

print(first\_value)

[1] 20

1. Using brackets, extract the first, second and third values from second\_vector. Store  
   the result in a variable called first\_three\_values.

Answer:

#16.

first\_three\_values <- second\_vector[1:3]

print(first\_three\_values)

[1] 20 19 18

1. Using brackets, extract the 1st, 5th, 10th, and 11th elements of second\_vector. Store  
   the resulting vector in a variable called vector\_from\_brackets

Answer:

#17.

vector\_from\_brackets <- second\_vector[c(1,5,10,11)]

print(vector\_from\_brackets)

[1] 20 16 11 10

1. Use the brackets to extract elements from 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. Include the  
   answer in your written report.

Answer:

#18. In the vector first\_vector, represented as [17, 12, -33, 5], applying a boolean vector for extraction results in retrieving only the values corresponding to the 'true' entries in the boolean vector

vector\_from\_boolean\_brackets <- first\_vector[c(FALSE,TRUE,FALSE,TRUE)]

print(vector\_from\_boolean\_brackets)

[1] 12 5

1. Examine the following piece of code and write a one sentence comment explaining  
   what is happening. Include the answer in your written report.  
   second\_vector >= 10

Answer:

#19. This code evaluates each element in second\_vector to determine if it is greater than 10, generating a logical vector with 'true' for elements greater than or equal to 10, and 'false' otherwise

print(second\_vector >= 10)

[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE

[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

1. Examine the following piece of code and write a one sentence comment explaining  
   what is happening and assuming one\_to\_one\_hundred was computed in the  
   previous problem. Include the answers in your written report.  
   one\_to\_one\_hundred[one\_to\_one\_hundred >= 20]

Answer:

#20.

print(one\_to\_one\_hundred[one\_to\_one\_hundred >= 20])

print(one\_to\_one\_hundred)

#this code creates a new vector where all elements from one\_to\_one\_hundred vector which are greater than or equal to 20 are included

[1] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37

[19] 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55

[37] 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73

[55] 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91

[73] 92 93 94 95 96 97 98 99 100

1. Using the same approach as in the previous question, create a new vector from the  
   grades vector with only values larger than 85. Store the result in a variable called  
   lowest\_grades\_removed.

Answer:

#21.

lowest\_grades\_removed <- grades[grades >85]

print(lowest\_grades\_removed)

[1] 96 100 92

1. Use the grades vector to create a new vector with the 3rd and 4th elements of  
   grades removed. Store the result in a variable called middle\_grades\_removed. Try  
   utilizing a vector of negative indexes to complete this task.

Answer:

#22.

middle\_grades\_removed <- grades[-c(3,4)]

print(middle\_grades\_removed)

[1] 96 100 81 72

1. Use bracket notation to remove the 5th and 10th elements of second\_vector. Store  
   the result in a variable called fifth\_vector.

Answer:

#23.

fifth\_vector <- second\_vector[-c(5,10)]

print(fifth\_vector)

[1] 20 19 18 17 15 14 13 12 10 9 8 7 6 5 4 3 2 1

1. Write the following code. This creates a variable called random\_vector that will be  
   utilized in problems 25 - 30.

Answer:

#24.

set.seed(5)

random\_vector <- runif(n=10, min=0, max=1000)

print(random\_vector)

[1] 200.2145 685.2186 916.8758 284.3995 104.6501 701.0575 527.9600 807.9352

[9] 956.5001 110.4530

1. Use the sum function to compute the total of random\_vector. Store the result in a  
   variable called sum\_vector

Answer:

#25.

sum\_vector <- sum(random\_vector)

print(sum\_vector)

[1] 5295.264

1. Use the cumsum function to compute the cumulative sum of random\_vector. Store  
   the result in a variable called called cumsum\_vector

Answer:

#26.

cumsum\_vector <- cumsum(random\_vector)

print(cumsum\_vector)

[1] 200.2145 885.4330 1802.3088 2086.7083 2191.3584 2892.4159 3420.3759

[8] 4228.3111 5184.8112 5295.2642

1. Use the mean function to compute the mean of random\_vector. Store the result in a  
   variable called mean\_vector

Answer:

#27.

mean\_vector <- mean(random\_vector)

print(mean\_vector)

[1] 529.5264

1. Use the sd function to compute the standard deviation of random\_vector. Store the  
   result in a variable called sd\_vector

Answer:

#28.

sd\_vector <- sd(random\_vector)

print(sd\_vector)

[1] 331.3606

1. Use the round function to round the values of random\_vector Store the result in a  
   variable called round\_vector

Answer:

#29.

round\_vector <- round(random\_vector)

print(round\_vector)

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

1. Use the sort function to sort the values of random\_vector. Store the result in a  
   variable called sort\_vector

Answer:

#30.

sort\_vector <- sort(random\_vector)

print(sort\_vector)

[1] 104.6501 110.4530 200.2145 284.3995 527.9600 685.2186 701.0575 807.9352

[9] 916.8758 956.5001

1. Download the datafile ds\_salaries.csv from Canvas. Save it on your computer in the  
   same folder (directory) where your .R file for this project is located.

Answer:

#31. file downloaded and saved at (../Kosamkar-Project1/ds\_salaries.csv)

1. Use the function read.csv to read the ds\_salaries.csv file. Store the result of the  
   read into a variable called first\_dataframe.

Answer:

first\_dataframe <- read.csv("ds\_salaries.csv")

print(head(first\_dataframe))

1. Use the summary function with first\_dataframe to produce summary statistics  
   based on each column of the dataframe

Answer:

print(summary(first\_dataframe)

A screenshot of a computer

Description automatically generated

Thank You.