Worksheet-1 in R

Worksheet for R Programming

Instructions:

• Use RStudio or the RStudio Cloud accomplish this worksheet.

• Create an .RMD file and name the file as *RWorksheet_lastname#1.Rmd*. Knit the rmd file into a pdf, save it as *RWorksheet_lastname#1.pdf*
• Create your own *GitHub repository* and push the R script as well as this pdf worksheet to your own repo (see Unit 2).

Accomplish this worksheet by answering the questions being asked and writing the code manually.

Using functions:

seq(), assign(), min(), max(), c(), sort(), sum(), filter()

1. Set up a vector named age, consisting of 34, 28, 22, 36, 27, 18, 52, 39, 42, 29, 35, 31, 27, 22, 37, 34, 19, 20, 57, 49, 50, 37, 46, 25, 17, 37, 42, 53, 41, 51, 35, 24, 33, 41.
   a. How many data points?
      There are 34 data points.
   b. Write the R code and its output.
   Code:
   age <- c(34, 28, 22, 36, 27, 18, 52, 39, 42, 29, 35, 31, 27, 22, 37, 34, 19, 20, 57, 49, 50, 37, 46, 25, 17, 37, 42, 53, 41, 51, 35, 24, 33, 41)

   AgeDataP <- length(age)
   AgeDataP
   Output:

```
> AgeDataP <- length(age)
> AgeDataP
[1] 34
```

2. Find the reciprocal of the values for age.

Write the R code and its output.

Code:

ReciproAge <- 1 / age

ReciproAge

Output:

```
> ReciproAge ← 1 / age
> ReciproAge
 [1] 0.02941176 0.03571429 0.04545455 0.02777778 0.03703704 0.05555556 0.01923077 0.02564103 0.02380952
[10] 0.03448276 0.02857143 0.03225806 0.03703704 0.04545455 0.02702703 0.02941176 0.05263158 0.05000000
[19] 0.01754386 0.02040816 0.02000000 0.02702703 0.02173913 0.04000000 0.05882353 0.02702703 0.02380952
[28] 0.01886792 0.02439024 0.01960784 0.02857143 0.04166667 0.03030303 0.02439024
```

3. Assign also new_age <- c(age, 0, age).

What happen to the new_age?

THE new_age obj has been assigned the age, 0 and age values inside the vector.

4. Sort the values for age.

Write the R code and its output.

Code:

SortDecrease <- sort(age, decreasing = TRUE)

SortDecrease

SortIncrease <- sort(age, decreasing = FALSE)

SortIncrease

Output:

```
> SortDecrease ← sort(age, decreasing = TRUE)
> SortDecrease
 [1] 57 53 52 51 50 49 46 42 42 41 41 39 37 37 37 36 35 35 34 34 33 31 29 28 27 27 25 24 22 22 20 19 18 17
>
> SortIncrease ← sort(age, decreasing = FALSE)
> SortIncrease
 [1] 17 18 19 20 22 22 24 25 27 27 28 29 31 33 34 34 35 35 36 37 37 37 39 41 41 42 42 46 49 50 51 52 53 57
```

5. Find the minimum and maximum value for age.

Write the R code and its output.

Code:

Age_min <- min(age)

Age_min

Age_max <- max(age)

Age_max

Output:

```
> Age_min ← min(age)
> Age_min
[1] 17
>
> Age_max ← max(age)
> Age_max
[1] 57
```

6. Set up a vector named data, consisting of 2.4, 2.8, 2.1, 2.5, 2.4, 2.2, 2.5, 2.3, 2.5, 2.3, 2.4, and 2.7.

a. How many data points?

There are 12 data points.

b. Write the R code and its output.

Code:

data <- c(2.4, 2.8, 2.1, 2.5, 2.4, 2.2, 2.5, 2.3, 2.5, 2.3, 2.4, 2.7)

dataDataP <- length(data)

dataDataP

Output:

```
> data ← c(2.4, 2.8, 2.1, 2.5, 2.4, 2.2, 2.5, 2.3, 2.5, 2.3, 2.4, 2.7)
>
> dataDataP ← length(data)
> dataDataP
[1] 12
```

7. Generates a new vector for data where you double every value of the data. | What happen to the data?

All the values of the vector "data" has been doubled or multiplied by 2.

8. Generate a sequence for the following scenario:

8.1 Integers from 1 to 100.

OneTOHundred <- seq(1:100)

OneTOHundred

8.2 Numbers from 20 to 60

TwentyTOSixty <- seq(from = 20, to = 60, by = 1)

TwentyTOSixty

*8.3 Mean of numbers from 20 to 60

Mean20TO60 <- mean(TwentyTOSixty)

Mean20TO60

*8.4 Sum of numbers from 51 to 91

Sum51To91 <- sum(seq(from = 51, to = 91, by =1))

Sum51To91

*8.5 Integers from 1 to 1,000

INTEGER1TO1000 <- seq(1:1000)

INTEGER1TO1000

a. How many data points from 8.1 to 8.4?_____

There are 143 data points.

b. Write the R code and its output from 8.1 to 8.4.

Code:

length(c(OneTOHundred, TwentyTOSixty, Mean20TO60, Sum51To91))

Output:

```
> length(c(OneTOHundred, TwentyTOSixty, Mean20TO60, Sum51To91))
[1] 143
```

c. For 8.5 find only maximum data points until 10.

10 is the output.

9. *Print a vector with the integers between 1 and 100 that are not divisible by 3, 5 and 7 using filter option.

Filter(function(i) { all(i %% c(3,5,7) != 0) }, seq(100)) Write the R code and its output.

Code:

divisibleVector <- Filter(function(i) { all(i %% c(3,5,7) != 0) }, seq(1:100))

divisibleVector

Output:

```
> divisibleVector ← Filter(function(i) { all(i %% c(3,5,7) ≠ 0) }, seq(1:100))
> divisibleVector
 [1]  1  2  4  8 11 13 16 17 19 22 23 26 29 31 32 34 37 38 41 43 44 46 47 52 53 58 59 61 62 64 67 68 71 73 74 76
[37] 79 82 83 86 88 89 92 94 97
```

10. Generate a sequence backwards of the integers from 1 to 100.

Write the R code and its output.

Code:

reverse1to100 <- sort(seq(1:100), decreasing = TRUE)

reverse1to100

Output:

```
> reverse1to100 ← sort(seq(1:100), decreasing = TRUE)
> reverse1to100
 [1] 100  99  98  97  96  95  94  93  92  91  90  89  88  87  86  85  84  83  82  81  80  79  78  77  76  75
[27]  74  73  72  71  70  69  68  67  66  65  64  63  62  61  60  59  58  57  56  55  54  53  52  51  50  49
[53]  48  47  46  45  44  43  42  41  40  39  38  37  36  35  34  33  32  31  30  29  28  27  26  25  24  23
[79]  22  21  20  19  18  17  16  15  14  13  12  11  10   9   8   7   6   5   4   3   2   1
```

11. List all the natural numbers below 25 that are multiples of 3 or 5.

Find the sum of these multiples.

a. How many data points from 10 to 11?

There are 112 data points in 10 to 11.

b. Write the R code and its output from 10 and 11.

Code:

reverse1to100 <- sort(seq(1:100), decreasing = TRUE)

reverse1to100

multiples <- Filter(function(i) {any(i %% c(3,5) == 0)}, seq(24))

multiples

SumOfMultiples <- sum(multiples)

SumOfMultiples

Output:

```
> reverse1to100 <- sort(seq(1:100), decreasing = TRUE)
> reverse1to100
  [1] 100  99  98  97  96  95  94  93  92  91  90  89  88  87  86  85  84  83  82  81  80  79  78  77  76  75
 [27]  74  73  72  71  70  69  68  67  66  65  64  63  62  61  60  59  58  57  56  55  54  53  52  51  50  49
 [53]  48  47  46  45  44  43  42  41  40  39  38  37  36  35  34  33  32  31  30  29  28  27  26  25  24  23
 [79]  22  21  20  19  18  17  16  15  14  13  12  11  10   9   8   7   6   5   4   3   2   1
> multiples <- Filter(function(i) {any(i %% c(3,5) == 0)}, seq(24))
> multiples
 [1]  3  5  6  9 10 12 15 18 20 21 24
> SumOfMultiples <- sum(multiples)
> SumOfMultiples
[1] 143
```

12. Statements can be grouped together using braces '{' and '}'. A group of statements is sometimes called a **block**. Single statements are evaluated when a new line is typed at the end of the syntactically complete statement. Blocks are not evaluated until a new line is entered after the closing brace.

    Enter this statement:

    x <- {0 + x + 5 + }

    Describe the output.

    It displays an error because it expects a continuation after the plus sign.

13. *Set up a vector named score, consisting of 72, 86, 92, 63, 88, 89, 91, 92, 75, 75 and 77. To access individual elements of an atomic vector, one generally uses the x[i] construction.

    Find x[2] and x[3]. Write the R code and its output.

    Code:

    score <- c(72, 86, 92, 63, 88, 89, 91, 92, 75, 75, 77)

    score2 <- score[2]

    score2

    score3 <- score[3]

    score3

    Output:

```
> score <- c(72, 86, 92, 63, 88, 89, 91, 92, 75, 75, 77)
> score2 <- score[2]
> score2
[1] 86
> score3 <- score[3]
> score3
[1] 92
```

14. *Create a vector a = c(1,2,NA,4,NA,6,7).

    a. Change the NA to 999 using the codes print(a,na.print="-999").

b. Write the R code and its output. Describe the output.

Code:

a = c(1,2,NA,4,NA,6,7)

print(a,na.print="-999")

Output:

```
> #14.*Create a vector a = c(1,2,NA,4,NA,6,7)
> a = c(1,2,NA,4,NA,6,7)
>
> #a.Change the NA to 999 using the codes print(a,na.print="-999").
> print(a,na.print="-999")
[1]    1    2 -999    4 -999    6    7
```

The NA value is replaced by 999.

15. A special type of function calls can appear on the left hand side of the assignment operator as in > class(x) <- "foo".

Follow the codes below:

name = readline(prompt="Input your name: ") age = readline(prompt="Input your age: ") print(paste("My name is",name, "and I am",age ,"years old."))

print(R.version.string)

What is the output of the above code?

The output is messy and the age variable uses the values of the earlier age variable that's been declared from the problem one.