RWorksheet_Calzado#1.pdf

Michael Angelo S. Calzado

2024-09-04

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?
34

b. Write the R code and its output.

```
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)
length(age)
```

```
## [1] 34
```

2. Find the reciprocal of the values for age.

Write the R code and its output.

```
library(MASS)
fractions(reciprocal_age <- 1 / age)
```

```
##  [1] 1/34 1/28 1/22 1/36 1/27 1/18 1/52 1/39 1/42 1/29 1/35 1/31 1/27 1/22 1/37
## [16] 1/34 1/19 1/20 1/57 1/49 1/50 1/37 1/46 1/25 1/17 1/37 1/42 1/53 1/41 1/51
## [31] 1/35 1/24 1/33 1/41
```

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

What happen to the new_age?

```
new_age <- c(age, 0, age)

print(new_age)
```

```
##  [1] 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
## [26] 37 42 53 41 51 35 24 33 41  0 34 28 22 36 27 18 52 39 42 29 35 31 27 22 37
## [51] 34 19 20 57 49 50 37 46 25 17 37 42 53 41 51 35 24 33 41
```

4. Sort the values for age.

Write the R code and its output.

```
sort(age)
```

```
##  [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
## [26] 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.

```
min_age <- min(age)
max_age <- max(age)
print(paste("Minimum value:", min_age))
```

```
## [1] "Minimum value: 17"
```

```
print(paste("Maximum value:", max_age))
```

```
## [1] "Maximum value: 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?
b. Write the R code and its 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)

number_of_data_points <- length(data)

print(paste("Number of data points:", number_of_data_points))
```

```
## [1] "Number of data points: 12"
```

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

```
doubled_data <- data * 2

print("Original data:")
```

```
## [1] "Original data:"
```

```
print(data)
```

```
##  [1] 2.4 2.8 2.1 2.5 2.4 2.2 2.5 2.3 2.5 2.3 2.4 2.7
```

```
print("Doubled data:")
```

```
## [1] "Doubled data:"
```

```
print(doubled_data)
```

```
##  [1] 4.8 5.6 4.2 5.0 4.8 4.4 5.0 4.6 5.0 4.6 4.8 5.4
```

8. Generate a sequence for the following scenario:
9. Integers from 1 to 100.
10. Numbers from 20 to 60
11. Mean of numbers from 20 to 60
12. Sum of numbers from 51 to 91
13. Integers from 1 to 1,000

a. How many data points from 8.1 to 8.4?_____
b. Write the R code and its output from 8.1 to 8.4.

```
integers_1_to_100 <- 1:100
print(length(integers_1_to_100))
```

```
## [1] 100
```

```r
numbers_20_to_60 <- 20:60
print(length(numbers_20_to_60))
```

```
## [1] 41
```

```r
mean_20_to_60 <- mean(numbers_20_to_60)
print(mean_20_to_60)
```

```
## [1] 40
```

```r
numbers_51_to_91 <- 51:91
sum_51_to_91 <- sum(numbers_51_to_91)
print(sum_51_to_91)
```

```
## [1] 2911
```

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

```r
max_data_points_until_10 <- 1:10
print(max_data_points_until_10)
```

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

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.

```r
sequence <- seq(100)

filtered_numbers <- Filter(function(i) { all(i %% c(3, 5, 7) != 0) }, sequence)

print(filtered_numbers)
```

```
##  [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
## [26] 58 59 61 62 64 67 68 71 73 74 76 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.

```r
sequence_forward <- 1:100

sequence_backward <- rev(sequence_forward)

print(sequence_backward)
```

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

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

```r
numbers_below_25 <- 1:24
```

```r
multiples_of_3_or_5 <- numbers_below_25[numbers_below_25 %% 3 == 0 | numbers_below_25 %% 5 == 0]

sum_multiples <- sum(multiples_of_3_or_5)

print(multiples_of_3_or_5)
```

```
##  [1]  3  5  6  9 10 12 15 18 20 21 24
```

```r
print(sum_multiples)
```

```
## [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.

x <- {0 + x + 5 + }

It doesn't work becuse you need to specify the complete equation if you want to assign a value of x.

13. Setup 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.

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

score_2 <- score[2]
score_3 <- score[3]

print(score_2)
```

```
## [1] 86
```

```r
print(score_3)
```

```
## [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.

```r
a <- c(1, 2, NA, 4, NA, 6, 7)

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

```
## [1]    1    2 -999    4 -999    6    7
```

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?

```
name = readline(prompt="Input your name: ")
```

```
## Input your name:
```

```
age = readline(prompt="Input your age: ")
```

```
## Input your age:
```

```
print(paste("My name is", name, "and I am", age, "years old."))
```

```
## [1] "My name is  and I am  years old."
```

```
print(R.version.string)
```

```
## [1] "R version 4.4.1 (2024-06-14)"
```