RWorksheet_Perez#1

2024-09-17

#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? length(age) 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)
```

```
age
```

```
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
```

#2. Find the reciprocal of the values for age. Write the R code and its output.

```
reciprocal <- function(x){y = 1/x return (y)}
```

```
reciprocal(age)
```

```
0.02941176 0.03571429 0.04545455 0.02777778 0.03703704 0.05555556 0.01923077 0.02564103 0.02380952
0.03448276 0.02857143 0.03225806 0.03703704 0.04545455 0.02702703 0.02941176 0.05263158 0.05000000
0.01754386 0.02040816 0.02000000 0.02702703 0.02173913 0.04000000 0.05882353 0.02702703 0.02380952
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?

As seen from the output, new_age shows a 0 between the two age vectors.

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

```
new_age
```

```
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 0 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
```

#4. Sort the values for age. Write the R code and its output.

```
sort(age, decreasing = FALSE)
```

```
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.

```
min(age)
```

```
17
```

```
max(age)
```

```
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?

```
length(data)
```

```
12
```

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)
```

```
data
```

```
2.4 2.8 2.1 2.5 2.4 2.2 2.5 2.3 2.5 2.3 2.4 2.7
```

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

```
new_data <- rep(data, each=2)
```

```
new_data
```

```
2.4 2.4 2.8 2.8 2.1 2.1 2.5 2.5 2.4 2.4 2.2 2.2 2.5 2.5 2.3 2.3 2.5 2.5 2.3 2.3 2.4 2.4 2.7 2.7
```

#8. Generate a sequence for the following scenario:

8.1 Integers from 1 to 100.

8.2 Numbers from 20 to 60

*8.3 Mean of numbers from 20 to 60

*8.4 Sum of numbers from 51 to 91

*8.5 Integers from 1 to 1,000

a. How many data points from 8.1 to 8.4?

```
length(c(int, twenty_sixty, mean2060, sum5191))
```

```
143
```

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

8.1.

```
int <- seq(100)
```

```
int
```

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

8.2.

```
twenty_sixty <- seq(from = 20, to = 60)
```

```
twenty_sixty
```

```
20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
56 57 58 59 60
```

8.3.

```
mean2060 <- (mean(twenty_sixty))
```

```
mean2060
```

```
40
```

8.4.

```
five_nine <- (seq(from = 51, to = 91))
```

```
five_nine
```

```
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86
87 88 89 90 91
```

```
sum5191 <- sum(five_nine)
```

```
sum5191
```

```
2911
```

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

```
int1000 <- seq(1000)
```

```
int1000
```

```
maxdp <- int1000[1:10]
```

```
maxdp
```

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

```
length(maxdp)
```

```
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.

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

```
not_div
```

```
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 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.

```
int100 <- seq(from = 100, to = 1)
```

```
int100
```

```
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 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 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 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.

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

```
multiples
```

```
3 5 6 9 10 12 15 18 20 21 24
```

```
sum_multiples <- sum(multiples)
```

```
sum_multiples
```

```
143
```

a. How many data points from 10 to 11?

```
length(c(int100, multiples, sum_multiples))
```

```
112
```

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

10.

```
int100 <- seq(from = 100, to = 1)
```

```
int100
```

```
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 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 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 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
```

11.

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

```
multiples
```

```
3 5 6 9 10 12 15 18 20 21 24
```

```
sum_multiples <- sum(multiples)
```

```
sum_multiples
```

```
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.

The output shows an error (Error: unexpected '}' in "x <- {0 + x + 5 + }"). This might be because of the incomplete statement.

#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.

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

```
score [2]
```

```
86
```

```
score [3]
```

```
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").

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

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

1 2 999 4 999 6 7 b. Write the R code and its output. Describe the output.

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

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

```
1 2 -999 4 -999 6 7
```

The output now has a value substituted for NA using the print function.

#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.”))

“My name is and I am years old.”

print(R.version.string)

“R version 4.4.1 (2024-06-14 ucrt)”
```