Lab3

SAF

2025-03-09

Part 1: Iterations

Question 1 - *Creating For and While loops (using large_vector data set)*

1A. Use a while loop to assign a new variable, mean_while, as the mean of large_vector. Use print to display this value.

```
## [1] "1A. mean of the large_vector using while loop "
```

```
## [1] 7.900514
```

1B. Use a for loop to assign a new variable, mean_for, as the mean of large_vector. Use print to display this value.

```
## [1] "1B. mean of the large_vector using for loop"
```

```
## [1] 15.80103
```

1C. How are these methods similar and how might vectorized functions be superior?

These two loops, for and while, using similar methods but with different count validation statements at the top of each loop. Where the for loop uses (i in count:countLim) for iteration checks and the while loop uses (count <= countLim) to validate the loop count.

One example of how vector functions are superior are how they can be mathematically manipulate large amounts of elements within the vector. For example, the vector "large_vector" holds 2,461,837 elements or numbers. A simple for/while loop is able to quickly run a summation of each vector element and find isn't mean.

Question 2 - *Loops vs vectorized functions*

2A. *Use vectorized functions to assign a new variable, norm_vectorized, using the normalization described in the question stem. For this question, it is permissible to use vectorized functions to verify that the normalization was done correctly.*

```
## [1] "2A. Using vectorized functions to Normalize large_vector."
```

```
## Time lapse
```

```
## 0.04 sec elapsed
```

2B. *Use a for loop to assign a new variable, norm_loop, using the normalization described in the question stem. For this question, it IS NOT permissible to use vectorized function to calculate total sum (use a for loop).*

```
## Time lapse
```

```
## 0.12 sec elapsed
```

2C. *Please explain the differences between these methods in terms of implementation, readability, and computation time. Using your own experiences in the workplace (or speculation of experiences for a job you*

would like to have), how might these differences be significant and what impact would they have? Method 2B. using the for loop takes roughly twice as long. Larger data sets could easily take 30-60mins to process,

Question 3 - *Apply vs loops*

3A. *Use apply to assign a new variable, mean_list_apply, the mean of each vector in large_list.*

```
## Mean of Sample A matrix: 7.90051412827088
```

```
## Mean of Sample B matrix: 12.8120342654692
```

```
## Mean of Sample C matrix: 9.0484057230434
```

```
## Time lapse
```

```
## 5.72 sec elapsed
```

3B. *Use for loop to assign a new variable, mean_list_for, the mean of each vector in large_list.*

```
## Mean for list.vector 1: 7.90051412827088
```

```
## Mean for list.vector 2: 12.8120342654692
```

```
## Mean for list.vector 3: 9.0484057230434
```

```
## Time lapse
```

```
## 0.02 sec elapsed
```

3C. *Please explain the differences between these methods in terms of implementation, readability, and computation time. Were these differences surprising? How do you predict these differences will change if we used lists with much more elements (i.e. length(list) > 1000)? The difference in processing time between 3A. and 3B. is 10x the amount. Using the apply() method took dramatically longer than simply using a for loop and R vector functions. If we were to work with length(list) greater than 1000 the workstation could be processing the list of data for more than an hour*

Part 2: Control

Question 4 - *Doing it the Un-R way*

4A. *Use the Un-R way, with a for loop and if/else statements, to assign a new variable, nbr_zeros_loop, as the number of zeros that are in large_vector. Print the value of this variable.* **4B.** *Use the Un-R way, with a for loop and if/else statements, to assign a new variable, smaller_vector_loop, as all the values of large_vector that are not equal to zero. Vectorized functions (i.e. na.omit) are permitted but a for loop must be used to iterate through vector. Show that the sum of nbr_zeros_loop and the length of smaller_vector_loop are equal to the length of larger_vector.*

```
## 4A. Zero's counted: 3537222
```

```
## 4B. Non-zero's counted: 3848289
```

```
## Zero + non-zero values counted: 7385511
```

```
## CHECK! total function length of entire list of vectors: 7385511
```

```
## Time lapse
```

```
## 0.63 sec elapsed
```

Question 5 - *Doing it the R-way* use the large_vector data set.

5A. *Use the R way, proper subsetting and vectorized functions, to assign a new variable, nbr_zeros, as the number of zeros that are in large_vector. Print this variable. Show that nbr_zeros and nbr_zeros_loop are equal.*

5B. *Use the R way, proper subsetting and vectorized functions, to assign a new variable, smaller_vector, as all the values of large_vector that are not equal to zero. Show that the sum of nbr_zeros and the length of smaller_vector are equal to the length of larger_vector.*

```
## Quantity of zeros in large_list: 3537222
```

```
## Quantity of non-zeros in large_list: 3848289
```

```
## CHECK! Total: 7385511
```

```
## Time lapse
```

```
## 0.17 sec elapsed
```