CSCI 385 - Introduction to Data Science
Introduction to R

August 17, 2021

Write your solutions to the problems below in an R markdown file called `assignment_1.Rmd` and commit it along with a knitted PDF to your GitHub repository in a folder called "Assignment 1" before **September 12th at 11:59 pm**. Each solution should include the problem statement and 3-5 simple test cases showing that the solution behaves as expected. Make sure to think about important edge cases. There are a total of 105 points on this assignment, 5 of which are extra credit.

These problems are intended to help familiarize you with the basics of R and of working in RStudio before we jump into using R for data science. If there is an existing function in R that solves a problem directly (or nearly so), avoid using it in your solution. For example, `rev(v)` reverses the given vector `v`. Don't use `rev` to write your own `reverse` function! If you are unsure whether or not a particular function makes a problem too easy, ask in the forum or on Discord and I will let you know.

There will be time in class to discuss these problems in small groups but it is important that you write up your own solutions and that you understand how they work. If you want more practice programming in R, let me know. I am happy to point you in the direction of good problem sets. Have fun!

Problems

1. (5 pts) Write a function `sum_mults(nums, n)` that returns the sum of all multiples of values in the vector `nums` less than `n`. For example, `sum_mults(c(3,5), 30)` should return 195. Assume that the elements of `nums` are positive integers.

2. (5 pts) Given any positive integer `n`, define

$$f(n) := \begin{cases} n/2 & , n \text{ even} \\ 3n+1 & , n \text{ odd.} \end{cases}$$

   The Collatz conjecture states that the sequence

$$a_i := \begin{cases} n & , i = 1 \\ f(a_{i-1}) & , i > 1 \end{cases}$$

   eventually reaches 1. Write a function `collatz_len(n)` that determines the first i for which $a_i = 1$ for a given n. For example, when $n = 17$, the sequence a_i begins

$$17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, \ldots$$

and $a_{13} = 1$. Thus, `collatz_len(17)` should return 13.

3. (5 pts) Write a function `reverse(v)` that reverses the vector `v`. So `reverse(c(1,2,3))` should return `c(3,2,1)`. `reverse` should return `NULL`, not `NA`, when `v` is `c()`.

4. (5 pts) Write a function `drop(v, n)` that drops every n^{th} element from the vector `v`. `drop(c(1,2,3,4,5), 2)` should return `c(1,3,5)`. `drop` should return `NULL` when $n \leq 1$.

5. (10 pts) Write a function `intersect_3(v, w, x)` that returns a vector of the elements that appear in each of the vectors `v`, `w`, and `x`. `intersect_3(c(1,2,3,1), c(1,1,3,2), c(3,1,9,1))` should return `c(1,3)`.

6. (10 pts) Write a function `filter_vec(v, p)` that returns a vector containing all the elements of `v` for which the predicate function `p` returns `TRUE`. For example,

```
p <- function(x){ return(x>3) }
l <- 1:6
m <- filter_vec(l, p)
```

   results in `m` being equal to the vector `c(4,5,6)`. Make sure that `filter_vec` returns `NULL` when `p` is `FALSE` for all elements of `v`.

7. (10 pts) Write a function `n_fibs(n)` that creates a vector of the first `n` Fibonacci numbers where the first and second Fibonacci numbers are 1 (`n_fibs(2)` returns `c(1,1)`).

8. (15 pts) Write a function `shift(v, n)` that shifts the elements of a vector `n` places to the right. If `n` is negative, the function should shift the vector to the left. For example, `shift(c(1,2,3,4), 2)` should return `c(3,4,1,2)` while `shift(c(1,2,3,4), -3)` should return `c(4,1,2,3)`.

9. (20 pts) Write a function `rem_consec_dups(v)` that removes consecutive duplicates from the vector `v`. `rem_consec_dups(c(1,1,1,2,3,3,1,2,2))` should return `c(1,2,3,1,2)`. Do not use the built-in function `rle` in your solution.

10. (20 pts) Write a function `n_even_fibs(n)` that creates a list of the first `n` even Fibonacci numbers. The name of each value should be its position in the Fibonacci sequence as a string. For example, `n_even_fibs(5)` should create a list with the structure

```
List of 5
  $ 3 : num 2
  $ 6 : num 8
  $ 9 : num 34
  $ 12: num 144
  $ 15: num 610
```

    The `toString` function might be useful.