week7_lab

December 2, 2022

1 Week 7: Functions

1.1 In-Class Work

[1]:
```r
library(tidyverse)
library(docstring)
```

```
Warning message in system("timedatectl", intern = TRUE):
"running command 'timedatectl' had status 1"
  Attaching packages                                  tidyverse
1.3.2
  ggplot2 3.3.6       purrr   0.3.4
  tibble  3.1.8       dplyr   1.0.9
  tidyr   1.2.0       stringr 1.4.1
  readr   2.1.2       forcats 0.5.2
  Conflicts
tidyverse_conflicts()
  dplyr::filter() masks stats::filter()
  dplyr::lag()    masks stats::lag()

Attaching package: 'docstring'


The following object is masked from 'package:grDevices':

    ?
```

1.1.1 Problem 1.

Standardizing a variable means subtracting the mean and then dividing through by the standard deviation. Create a function called `standardize_me()` that takes a numeric vector as an argument, and returns the standardized version of the vector. Test this on a vector to see that the mean and standard deviation are correct.

[2]:
```r
# Solution:
standardize_me <- function(x) {
    #'Standardizes a numeric vector x, note ignores NA values in mean and sd
    #'@param x numeric vector
    #'@return standardized version of the vector x

    # Fill in
}
```

[3]:
```
# Test case:
# x <- c(1,2,3)
# y <- standardize_me(x)
# testthat::expect_equal(mean(y), 0)
# testthat::expect_equal(sd(y), 1)
```

1.2 Problem 2

Sometimes in our data we might prefer to not record instances with value zero since it inflates the size of our data. Write a function `sd_padded` that takes in a numeric vector `x` and an integer `n_zeros` and calculates the standard deviation of `x` as if `n_zeros` zeros were appended to `x`. You should not actually append the zeros in your solution (since if we had many, many zeros that would be inefficient). Instead, think about using the equation below for the standard deviation. I've written a few test cases for you. **Hint:** the sum of x is the same with or without the zeros.

$$\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \text{mean}(x))^2$$

[4]:
```
# Solution:
```

[5]:
```
# Test cases:
# testthat::expect_equal(sd_padded(vector("numeric"), 5), 0)
# testthat::expect_equal(sd_padded(c(1), 3), 0.5)
# testthat::expect_equal(sd_padded(c(1,1), 100), 0.139, 0.001)
```

1.2.1 Problem 3.

Create your own `unique` function, which given a vector will return a new vector with the elements of the first vector with duplicated elements removed. Again, I have provided a few test cases.

[6]:
```
# Solution:
```

[7]:
```
# Test cases:
# testthat::expect_equal(unique(vector()), NULL)
# testthat::expect_equal(unique(c(1,2,1)), c(1,2))
# testtthat::expect_equal(unique(c(3,3,1,2,1)), c(3,1,2))
```

1.2.2 Problem 4.

The function below is supposed to take in a positive integer and calculate how many positive integer divisors it has (other than 1 and itself). However, the function is not getting the right results. Debug the function. Then, think about ways you could improve this function by changing the structure, documentation, and adding argument checks.

[8]:
```r
divisors <- function(x){
    if (x <= 0 | floor(x) != x ) stop("x must be a positive integer")
    if (x <= 2) return(0)

    total <- 0 # keep track of total number of divisors

    for(i in 2:(x-1)){
        if (x %% i == 0){
            total <- total + 1
        }
    }
    return(total)
}
```

[9]:
```r
divisors(2)
divisors(6)
```

0

2

2 Week 7 Assignment

2.0.1 Problem 1.

Write a function that takes in a data frame and highlights pairs of numeric columns that are highly correlated. The structure of the function and output is up to you. For example, you may choose to write a function that takes in a threshold and returns all pairs of columns whose correlation exceeds that threshold. Or you may choose to return a plot, table, or data frame with some information about correlations. Write a few sentences talking about your function.

2560 Only: Then, test your code on the American College Scorecard data set (https://collegescorecard.ed.gov/). Comment on your results (variable information can be found in the data dictionary also on Canvas).

I have provided some code below to subset the data to only numeric columns.

Solution:

[10]:
```r
# Get only numeric columns for a data frame
df <- data.frame(x = c(1,2,3,4,5), y = c("a", "b", "c","d","e"), z =␣
 ↪c(0,0,1,2,3), w = c(0.1,0.1,0.2, 0.1, 0.1))
data_num <- select_if(df, is.numeric)
```

[11]:
```r
# Solution:
college_df <- read.csv("/home/jovyan/data/college_scorecard.csv")
```