Week 3 Exercises

M. Cristina Fernandez

11/9/2024

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```
library(stringr)
```

Please complete all exercises below. You may use any library that we have covered in class UP TO THIS POINT.

For each function, show that it works, by using the provided data as a test and by feeding in some test data that you create to test your function

Add comments to your function to explain what each line is doing

1.) Write a function that takes in a string with a person's name in the form

"Sheets, Dave"

and returns a string of the form

"Dave Sheets"

Note:

-assume no middle initial ever -remove the comma -be sure there is white space between the first and last name

You will probably want to use stringr

```
name_in = "Sheets, Dave"

reorder_name <- function (last_first) {
  name_order <- str_split(last_first, ", ")[[1]]  # split into first and last name
  paste(name_order[2], name_order[1])             # reversing the order of the name so the surna
me is first
}

standard_name <- reorder_name(name_in)           # wasn't really sure what to name the variable,
but it seems to work
print(standard_name)
```

```
## [1] "Dave Sheets"
```

2.) Write a function that takes in a string of values x, and returns a data frame with three columns, x, x^2 and the square root of x

```
x = c(1,3,5,7,9,11,13)

powers_df <- function(x)
{
  x <- as.numeric(x)    # convert x to numeric vector
  powers_df_col <- data.frame(x, x^2, sqrt(x)) #creating the data frame with the 3 columns

  return(powers_df_col)
}

powers_df(x)
```

x <dbl>	x.2 <dbl>	sqrt.x. <dbl>
1	1	1.000000
3	9	1.732051
5	25	2.236068
7	49	2.645751
9	81	3.000000
11	121	3.316625
13	169	3.605551

7 rows

3.) Write in a function that takes in a value x and returns

```
y= 0.3x if x<0
y=0.5x if x>=0

This is a variant on a relu function as used in some neural networks.
```

```
reluvariant <- function(x) {
  if (x < 0) {
    return(0.3 * x)
  } else {
    return(0.5 * x)
  }
}

reluvariant(-2) # testing out the code for x < 0.
```

```
## [1] -0.6
```

```
reluvariant(0)  # since x = 0, I expect the output to be 0, though I don't know if I could reall
y tell which one it ran!
```

```
## [1] 0
```

```
reluvariant(9)  # testing for x > 0.
```

```
## [1] 4.5
```

4.) Write a function that takes in a value x and returns the first power of two greater than x (use a While loop)

```
next_power_of_two <- function(x) {
  power_of_two <- 1
  while (power_of_two <= x) {
    power_of_two <- power_of_two * 2
  }
return(power_of_two)
  }

next_power_of_two(4)
```

```
## [1] 8
```

```
next_power_of_two(18)
```

```
## [1] 32
```

5. Two Sum - Write a function named two_sum()

Given a vector of integers nums and an integer target, return indices of the two numbers such that they add up to target.

You may assume that each input would have exactly one solution, and you may not use the same element twice.

You can return the answer in any order.

Example 1:

Input: nums = [2,7,11,15], target = 9 Output: [0,1] Explanation: Because nums[0] + nums[1] == 9, we return [0, 1]. Example 2:

Input: nums = [3,2,4], target = 6 Output: [1,2] Example 3:

Input: nums = [3,3], target = 6 Output: [0,1]

Constraints:

2 <= nums.length <= 104 –109 <= nums[i] <= 109 –109 <= target <= 109 Only one valid answer exists.

Note: For the first problem I want you to use a brute force approach (loop inside a loop)

The brute force approach is simple. Loop through each element x and find if there is another value that equals to target – x

Use the function seq_along to iterate

```
two_sum <- function(nums_vector, target){
  for (i in seq_along(nums_vector)) {
    for (j in seq_along(nums_vector)) {
      if (i !=j && nums_vector[i] + nums_vector[j] == target) {
        return(c(i, j)) #I suspect this part of the code is incomplete
      }
    }
  }
 return(NULL)
}

# Test code
nums_vector <- c(5,7,12,34,6,10,8,9)
target <- 13

z = two_sum(nums_vector,target)
print(z)
```

```
## [1] 1 7
```

```
#expected answers
#[1] 1 7
#[1] 2 5
#[1] 5 2
```

I’m sure I’m missing something super basic, but I’m not sure why I’m only getting one of the expected answers.

6.) Write one piece of code that will use a regex command to extract a phone number written in the form

```
456-123-2329
```

The sentences to use are located below

use the str_extract function from stringr

use the same regex search pattern from each

-What does \d match to? or alternatively [[:digit:]] #It looks for digits within the text

-How do you specify a specific number of repeated characters #Enter said number within the curly brackets, so to look up a string of 3 digits, we’d write \d{3}

```
a="Please call me at 456-123-2329, asap"
b="Hey, we have a code 234 on machine a-234-12, call me at 678-321-98766"
c="On 12-23-2022, Joe over at 122 Turnpike, dialled 912-835-4756, tell me by 9:02 pm Wed"

#use regex to match the phone number format: 3 digits, hyphen, 3 digits, hyphen, 4 digits
#this will stop it from returning the other numbers that aren't written in that specific phone n
umber format
phone_number_form <- "\\d{3}-\\d{3}-\\d{4}"
sentence_a <- str_extract(a, phone_number_form) #extract phone numbers from each sentence
sentence_b <- str_extract(b, phone_number_form)
sentence_c <- str_extract(c, phone_number_form)

print(sentence_a)
```

```
## [1] "456-123-2329"
```

```
print(sentence_b)
```

```
## [1] "678-321-9876"
```

```
print(sentence_c)
```

```
## [1] "912-835-4756"
```

7.) For lines below, extract the domains (ie the part of the address after @)

```
d="jimmy.halibut@gmail.com"
e="His address is:  c.brown@hopeles.org, do write him"
f="h.potter@hogwarts.edu is bouncing back on me, I wonder why?"

extract_domain <- function(email) {
  domain_format <- "(?<=@)[a-zA-Z0-9.-]+" #I had to ask ChatGPT for help with this line; I tried '@(.+)$' and it was giving me too much info
  email_domain <- str_extract(email, domain_format)
  return(email_domain)
}

email_d <- extract_domain(d)
email_e <- extract_domain(e)
email_f <- extract_domain(f)

print(email_d)
```

```
## [1] "gmail.com"
```

```
print(email_e)
```

```
## [1] "hopeles.org"
```

```
print(email_f)
```

```
## [1] "hogwarts.edu"
```