Week 3 Exercises

11/09/2024

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.4.2
```

```
## ── 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 become 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)
{
  newname <-  str_split_fixed(gsub(" ","",last_first),",",2)
  name_out <-paste(newname[2],newname[1],sep=" ")
  return(cat(name_out))
}

reorder_name(name_in)
```

```
## 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)
{
  powers.df <- data.frame(x,xsqrd=x^2,xsqrt=sqrt(x))
}
print(powers_df(x))
```

```
##    x xsqrd    xsqrt
## 1  1     1 1.000000
## 2  3     9 1.732051
## 3  5    25 2.236068
## 4  7    49 2.645751
## 5  9    81 3.000000
## 6 11   121 3.316625
## 7 13   169 3.605551
```

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.
```

```
x=-17
zero_proximity<-function(x)
{
  if(x<0)
  {
    y= 0.3*x
  }else
  {
    y=0.5*x #if x>=0
  }
}
print(zero_proximity(x))
```

```
## [1] -5.1
```

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

```
x <- 10

power2greaterx<-function(x)
{
  i <- 0
  while (TRUE)
  {
    y=2^i
    if(y>x)
    {
     break
    }
     i <- i + 1
  }
 y <- (paste("2 ^",i,"= ",y))
}
cat(power2greaterx(x))
```

```
## 2 ^ 4 =  16
```

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

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


two_sum <- function(nums_vector,target)
{

if(!(length(nums_vector<=104) && length(nums_vector)>=2))
  {
    cat("nums_vector length violation - must be 2 <= nums.length <= 104 \n")
    return(NULL)
  }
if(!(nums_vector[which.max(nums_vector)]<=109 && nums_vector[which.min(nums_vector)]>= -
109))
  {
    cat("vector value violation - must be -109 <= [value] <= 109 \n")
    return(NULL)
  }
if(!((target<=109) && (target>= -109)))
  {
    cat("target value violation - must be -109 <= value <= 109 \n")
    return(NULL)
  }

  for(i in seq_along(nums_vector))
  {
    nums.length <- length(nums_vector)
    y <- i
    while(y<nums.length)
    {
      nums <- nums_vector[i]+nums_vector[y+1]
      if(nums==target)
      {
        sums <- c(which(nums_vector==nums_vector[i]),which(nums_vector==nums_vector[y+
1]))
        print(sums)
      }
      y=y+1
    }
  }
}
two_sum(nums_vector,target)
```

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

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:]]

-How do you specify a specific number of repeated characters

```
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 We
d"

pattern <- "\\d{3}\\b-\\d+-\\b\\d{4}"
# ## Match data from regexpr()
ma <- regexpr(pattern, a)
mb <- regexpr(pattern, b)
mc <- regexpr(pattern, c)

print("Printing using RegEx commands...")
```

```
## [1] "Printing using RegEx commands..."
```

```
regmatches(a, ma)
```

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

```
regmatches(b, mb)
```

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

```
regmatches(c, mc)
```

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

```
print("Printing using Stringr commands...")
```

```
## [1] "Printing using Stringr commands..."
```

```
str_extract(a,"\\d{3}\\b-\\d+-\\b\\d{4}")
```

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

```
str_extract(b,"\\d{3}\\b-\\d+-\\b\\d{4}")
```

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

```
str_extract(c,"\\d{3}\\b-\\d+-\\b\\d{4}")
```

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

```
cat("-\\\\d matches any digit","")
```

```
## -\\d matches any digit
```

```
cat("- specify a repeated number of specific characters by using:","\n"," \\D{x} where x
is a number of how many specific repeated characters - e.g. ","\n")
```

```
## - specify a repeated number of specific characters by using:
##   \D{x} where x is a number of how many specific repeated characters - e.g.
```

```
str_extract(a,"\\D{3}")
```

```
## [1] "Ple"
```

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?"

gsub("@","",str_extract(d,"@\\D{1,}\\.\\D{1,3}"))
```

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

```
gsub("@","",str_extract(e,"@\\D{1,}\\.\\D{1,3}"))
```

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

```
gsub("@","",str_extract(f,"@\\D{1,}\\.\\D{1,3}"))
```

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