607_homework_3

Alec

9/9/2021

Load Libraries

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(stringr)
library(htmlwidgets)
```

Question 1

Using the 173 majors listed in fivethirtyeight.com's College Majors dataset [https://fivethirtyeight.com/features/the-economic-guide-to-picking-a-college-major/], provide code that identifies the majors that contain either "DATA" or "STATISTICS"

Load the data from github using readr package

The readr package is a great way to load data from github

```
library(readr)

major_data <- read_csv("https://raw.githubusercontent.com/fivethirtyeight/data/master/college-majors/ma
```

```
## Rows: 174 Columns: 3
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): FOD1P, Major, Major_Category

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

write code that receives the major_dataset, a character vector containing desired major search terms, and returns the matching major names

```
retrieve_majors_with_x <- function(major_data, words) {

  pipe_words <- str_c(words, collapse= "|")

  return(major_data$Major %>% str_subset(pipe_words))
}
```

test the formula with "DATA" and "STATISTICS"

```
input_major_string <- c("DATA","STATISTICS")

final_value <- retrieve_majors_with_x(major_data, input_major_string)

final_value
```

```
## [1] "MANAGEMENT INFORMATION SYSTEMS AND STATISTICS"
## [2] "COMPUTER PROGRAMMING AND DATA PROCESSING"
## [3] "STATISTICS AND DECISION SCIENCE"
```

test the formula with anothr set of search terms

```
input_major_string <- c("BUSINESS","DATA","STATISTICS","PSYCHOLOGY")

final_value <- retrieve_majors_with_x(major_data, input_major_string)

final_value
```

```
##  [1] "COGNITIVE SCIENCE AND BIOPSYCHOLOGY"
##  [2] "GENERAL BUSINESS"
##  [3] "BUSINESS MANAGEMENT AND ADMINISTRATION"
##  [4] "BUSINESS ECONOMICS"
##  [5] "INTERNATIONAL BUSINESS"
##  [6] "MANAGEMENT INFORMATION SYSTEMS AND STATISTICS"
##  [7] "MISCELLANEOUS BUSINESS & MEDICAL ADMINISTRATION"
##  [8] "COMPUTER PROGRAMMING AND DATA PROCESSING"
##  [9] "STATISTICS AND DECISION SCIENCE"
```

```
## [10] "PSYCHOLOGY"
## [11] "EDUCATIONAL PSYCHOLOGY"
## [12] "CLINICAL PSYCHOLOGY"
## [13] "COUNSELING PSYCHOLOGY"
## [14] "INDUSTRIAL AND ORGANIZATIONAL PSYCHOLOGY"
## [15] "SOCIAL PSYCHOLOGY"
## [16] "MISCELLANEOUS PSYCHOLOGY"
```

Question 2

Write code that transforms 'input_data' (in the form of a .CSV file) into 'expected_output'

load the input and expected output data

```
input_data <- read.csv("homework_2_q_2.csv", header = FALSE)
```

```
## Warning in read.table(file = file, header = header, sep = sep, quote = quote, :
## incomplete final line found by readTableHeader on 'homework_2_q_2.csv'
```

```
expected_output <- c("bell pepper", "bilberry", "blackberry", "blood orange", "blueberry", "cantaloupe"

input_data
```

```
##                                                          V1
## 1 [1] "bell pepper" "bilberry" "blackberry" "blood orange"
## 2 [5] "blueberry" "cantaloupe" "chili pepper" "cloudberry"
## 3              [9] "elderberry" "lime" "lychee" "mulberry"
## 4                              [13] "olive" "salal berry"
```

write code that converts input to output

My approach is to first concatenate all of the rows of the csv file into one string. Following this I will use regex to remove the [##] values. At this point, I can extract any text that is surrounded by quotation marks. Following this extraction, I remove the quotation marks, and the result will match the expected data.

```
format_conversion <- function(input_data) {
  # first we concatenate all the rows of the .CSV file

  concat <- ""

  for (x in 1:nrow(input_data)) {
    concat <- str_c(concat, input_data[x,])
  }

  # next we remove all '[#]'s from the concatenated string and extract everything where there is text/s

  extracted_values <- concat %>%
```

```
    str_replace_all("\\[\\d+\\]\\s", "") %>%
    str_extract_all("\"[a-z\\s]+\"")

  #finally, convert extracted_values into a character vector, and remove all quotation marks

  final_format <- str_replace_all(unlist(extracted_values[1]),"\"","")

  return(final_format)
}
```

Another, simpler function I made after some more thought:

```
convert_input <- function(input_data) {

  ret <- str_c(input_data$V1, collapse="") %>%
    str_extract_all('[A-Za-z]+.?[A-Za-z]+')

  return(unlist(ret[1]))
}
```

run code, and verify that output == expectd_output for both functions

```
output <- format_conversion(input_data)

output2 <- convert_input(input_data)
```

```
output == expected_output
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
output2 == expected_output
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

Quesiton 3 - Describe what these expressions will match

(.)\1\1

Without quotations, this should be interpreted as a "numeric expression". As such, this expression would match against any substring that occurs three times in a row within a string.

But if you try to enter this into R with quotations, you will match against something entirely different, and unexpcted. You would match against a substring that contains any character, followed by two escaped 1's:

```
example <- c("brrr\1\1r","hmmm\1","assass\2\2in")
str_extract(example,"(.)\1\1")
```

```
## [1] "r\001\001" NA          NA
```

If you want to get three characters in a row, then you need to convert the numeric expression to a string expression, by escaping the backslashes: "(.)\1\1"

The regex "(.)\1\1" would match substrings with three repeating characters.

```
example <- c("brrrr","hmmmattt","assassin")
str_extract_all(example,"(.)\\1\\1")
```

```
## [[1]]
## [1] "rrr"
##
## [[2]]
## [1] "mmm" "ttt"
##
## [[3]]
## character(0)
```

"(.)(.)\2\1"

This will match 4 character palindrome substrings

```
example <- c("racecar","woowzers","rocks")
str_extract(example,"(.)(.)\\2\\1")
```

```
## [1] NA     "woow" NA
```

(..)\1

This is similar to the first one. It will match any two characters followed by an escaped 1 (\1) unless you convert to a string expression.

```
example <- c("hello\1","h\1","bingo")
str_extract(example, "(..)\1")
```

```
## [1] "lo\001" NA       NA
```

When using double slashes, like "(..)\1", then it would match a four character string, starting with the first two characters repeating

```
example <- c("banana","titilating","money")
str_extract(example, "(..)\\1")
```

```
## [1] "anan" "titi" NA
```

"(.).\1.\1"

This will match a character X, followed by a single character (unspecificed), followed by X, followed by a character (unspecified), followed by X.

```
example <- c("T-T6T Freight Engine", "Monopoly","battleship")
str_extract(example, "(.).\\1.\\1")
```

```
## [1] "T-T6T" "onopo" NA
```

"(.)(.)(.).*\3\2\1"

This will match a substring whose last three characters are the reverse of the first three characters. However there must be any number of characters between the first three, and the last three.

```
example <- c("123thisisnotmypassword321","hello_mister_sim","dogs")
str_extract(example, "(.)(.)(.).*\\3\\2\\1")
```

```
## [1] "123thisisnotmypassword321" "mister_sim"
## [3] NA
```

Question 4 - construct regex to match the following

Start and end with the same character.

```
pattern <- "(.).*\\1"
example <- c("nylon","velcro","Mr. Greenberg")
str_extract(example, pattern)
```

```
## [1] "nylon"       NA            "r. Greenber"
```

Contain a repeated pair of letters (e.g. "church" contains "ch" repeated twice.)

```
pattern <- ".*(..).*\\1.*"
example <- c("church","bigbird","other")
str_extract(example, pattern)
```

```
## [1] "church"  "bigbird" NA
```

Contain one letter repeated in at least three places (e.g. "eleven" contains three "e"s.)

```
pattern <- ".*(.).*\\1.*\\1.*"
example <- c("eleven","bananana","mount sinai")
str_extract(example, pattern)
```

```
## [1] "eleven"   "bananana" NA
```