STAT 412/612 Week 9 Homework

stringr and Regular Expressions

Yunting

3/22/2020

Question 1: Scrabble Words

1. Load into R the list of acceptable (2015) Scrabble words fromhttps://dcgerard.github.io/stat__412__612/data/words.txt.

Hint: "NA" is an actual word. It means "no" or "not".

```r
library(tidyverse)
```

```
## -- Attaching packages ---- tidyverse 1.3.0 --
```

```
## <U+2713> ggplot2 3.3.0     <U+2713> purrr   0.3.3
## <U+2713> tibble  2.1.3     <U+2713> dplyr   0.8.3
## <U+2713> tidyr   1.0.0     <U+2713> stringr 1.4.0
## <U+2713> readr   1.3.1     <U+2713> forcats 0.4.0
```

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

```r
text_mania <- read_tsv(file = "https://dcgerard.github.io/stat_412_612/data/words.txt",
                       na = character())
```

```
## Parsed with column specification:
## cols(
##   word = col_character()
## )
```

```r
sample_n(text_mania,10)
```

```
## # A tibble: 10 x 1
##    word
##    <chr>
##  1 AROMATHERAPIST
##  2 UNTAPPED
##   3 ENTOMIC
##   4 SUBMISSIVELY
##   5 FRIGORIFIC
##   6 OILNUTS
##   7 BUSHLAND
##   8 MISSORTS
##   9 PAIOCKE
##  10 STRONGARMS
```

```
#check
text_mania %>%
  filter(is.na(word))
```

```
## # A tibble: 0 x 1
## # ... with 1 variable: word <chr>
```

2. How many words either begin or end in "X"?

Ans: 885 units

```
text_mania %>%
  filter(str_count(word, "^X") | str_count(word, "X$")) %>%
  nrow()
```

```
## [1] 885
```

3. How many words contain all of the vowels (A, E, I, O, and U)?

Ans: 3476 units

```
text_mania %>%
filter(str_detect(word, "A") & str_detect(word, "E")
       & str_detect(word, "I") & str_detect(word, "O")
       & str_detect(word, "U")) -> text_vowels

nrow(text_vowels)
```

```
## [1] 3476
```

4. What are the shortest words that contain all of the vowels? (there should be five of them)

Ans: DOULEIA, EULOGIA, MIAOUED, MOINEAU and SEQUOIA

```
text_vowels %>%
  mutate(count = str_length(word)) %>% # the length of shortest words is seven.
  arrange(count) %>%
  filter(count == 7)
```

```
## # A tibble: 5 x 2
##   word    count
##   <chr>   <int>
## 1 DOULEIA     7
## 2 EULOGIA     7
## 3 MIAOUED     7
## 4 MOINEAU     7
## 5 SEQUOIA     7
```

5. Switch the first and last letters of all of the words. How many of them are still words?

Ans: 21287 units

```
text_mania %>%
   mutate(convert_word = str_replace_all
          (word,"^([A-Z])(.*)([A-Z])$", "\\3\\2\\1")) %>%
   mutate( is_word = convert_word %in% word) %>%
  filter(is_word == TRUE)  -> still_words

head(still_words)
```

```
## # A tibble: 6 x 3
##   word   convert_word is_word
##   <chr>  <chr>        <lgl>
## 1 AA     AA           TRUE
## 2 AB     BA           TRUE
## 3 ABA    ABA          TRUE
## 4 ABACA  ABACA        TRUE
## 5 ABAKA  ABAKA        TRUE
## 6 ABASIA ABASIA       TRUE
```

```
count(still_words)
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1 21287
```

6. How many of the words that are still words after switching the first and last letters have *different* first and last letters?

Ans: 1696 units

```
still_words %>%
  mutate(same_first_last = str_sub(word,str_length(word), )
         == str_sub(word,1,1)) %>%
  # we can use "word" or "convert_word" columns.
  filter(same_first_last == FALSE) -> still_words_firstlast_different

count(still_words_firstlast_different)
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1  1696
```

7. What are the longest words that are still words after switching the first and last letters and where the first and last letters are different? You should end up with six words (three pairs of words).

```
still_words_firstlast_different %>%
  mutate(length = str_length(word)) %>%
  arrange(desc(length)) %>%
  head(6)
```

```
## # A tibble: 6 x 5
##   word           convert_word   is_word same_first_last length
##   <chr>          <chr>          <lgl>   <lgl>            <int>
## 1 DECOMMISSIONER RECOMMISSIONED TRUE    FALSE               14
## 2 DEMYTHOLOGISER REMYTHOLOGISED TRUE    FALSE               14
## 3 DEMYTHOLOGIZER REMYTHOLOGIZED TRUE    FALSE               14
## 4 RECOMMISSIONED DECOMMISSIONER TRUE    FALSE               14
## 5 REMYTHOLOGISED DEMYTHOLOGISER TRUE    FALSE               14
## 6 REMYTHOLOGIZED DEMYTHOLOGIZER TRUE    FALSE               14
```

Question 2 Bank Data

The US Federal Reserve publishes data on the largest commercial banks chartered in the United States.

1. Read in the provided fed_large_c_bank_ratings.csv file to answer the following questions

```
US_commercial_banks <- read.csv(file = "./data/fed_large_c_bank_ratings.csv")

sample_n(US_commercial_banks,10)
```

```
##                                            name rank charter
## 1            SIMMONS BK/SIMMONS FIRST NAT CORP    84     SMB
## 2                                   ALDEN ST BK/ 1637     SMB
## 3           TENNESSEE ST BK/TENNESSEE ST BSHRS   925     SMB
## 4                     OAKSTAR BK/OAKSTAR BSHRS   651     SMB
## 5         NANO BANC/ALLEGIANT UNITED HOLDS LLC   793     SMB
## 6                        ADAMS B&TC/ADAGE LLC   774     SMB
## 7                            COMMENCEMENT BK/ 1490     SMB
## 8             BANK OF FAYETTE CTY/MOSCOW BSHRS   952     SMB
## 9           WEST MI CMNTY BK/NORTHSTAR FNCL GRP 1021     SMB
## 10 FARMERS BK OF NORTHERN MO/NORTHERN MO BSHRS 1507     SMB
##    consolidated_assets
## 1                17659
## 2                  345
## 3                  678
## 4                  996
## 5                  815
## 6                  839
## 7                  388
## 8                  656
## 9                  596
## 10                 383
```

2. Many of the banks have more than one name. Separate out the multiple names into different columns called name and alternate name, ignore any additional names, and update the data frame.

```
US_commercial_banks %>%
  separate("name", into = c("name", "alternate_name"), sep = "/") ->
  US_commercial_banks_V02
```

```
## Warning: Expected 2 pieces. Additional pieces discarded in 1 rows [15].
```

```
head(US_commercial_banks_V02)
```

```
##                    name          alternate_name rank charter
## 1    BANK OF NY MELLON  BANK OF NY MELLON CORP   10     SMB
## 2    STATE STREET B&TC       STATE STREET CORP   11     SMB
## 3 GOLDMAN SACHS BK USA GOLDMAN SACHS GROUP THE   12     SMB
## 4              ALLY BK               ALLY FNCL   15     SMB
## 5          NORTHERN TC        NORTHERN TR CORP   20     SMB
## 6           REGIONS BK              REGIONS FC   22     SMB
##   consolidated_assets
## 1              311387
## 2              242148
## 3              228836
## 4              167492
## 5              135885
## 6              125641
```

```
# recheck if they have additional names
# US_commercial_banks_V02 %>%
# mutate(add_name = str_detect(alternate_name, "/")) %>%
# filter(add_name == TRUE)
```

3. How many bank names begin with a digit?

Ans: 2

```
US_commercial_banks_V02 %>%
  filter(str_detect(name, "^\\d")) %>%
  nrow()
```

```
## [1] 2
```

4. How many bank names have the word "BANK" in them?

Ans: 41

```
US_commercial_banks_V02 %>%
   filter(str_detect(name, "BANK")) %>%
  nrow()
```

```
## [1] 41
```

5. Convert the abbreviation "BK" to the word "BANK". What are the relative proportions of names that have "BANK" as the first word, the last word, somewhere other than first or last, or not at all?

```
US_commercial_banks_V02 %>%
  mutate(name = str_replace_all(name, "BK", "BANK")
         , position_banks = if_else(str_detect(name, "^BANK "), "first",
                            if_else(str_detect(name, " BANK$"), "last",
                            if_else(str_detect(name, " BANK "), "middle","none")))) -> US_commercial_ban


US_commercial_banks_V03 %>%
  group_by(position_banks) %>%
  summarize(proportions = n()/nrow(US_commercial_banks_V02))
```

```
## # A tibble: 4 x 2
##   position_banks proportions
##   <chr>                <dbl>
## 1 first                0.056
## 2 last                 0.664
## 3 middle               0.096
## 4 none                 0.184
```

```
# traditional method
US_commercial_banks_V02 %>%
  mutate(name = str_replace_all(name, "BK", "BANK")) %>%
 filter(str_detect(name, "^BANK ")) -> first_banks

nrow(first_banks) #21 have "BANK" as the first word
```

```
## [1] 21
```

```r
prop_first <- (nrow(first_banks)/ nrow(US_commercial_banks_V02))*100
prop_first
```

```
## [1] 5.6
```

```r
# traditional method

US_commercial_banks_V02 %>%
  mutate(name = str_replace_all(name, "BK", "BANK")) %>%
 filter(str_detect(name, " BANK$")) -> last_banks

nrow(last_banks) #249 have "BANK" as the last word
```

```
## [1] 249
```

```r
prop_last<- (nrow(last_banks)/ nrow(US_commercial_banks_V02))*100
prop_last
```

```
## [1] 66.4
```

```r
# traditional method

US_commercial_banks_V02 %>%
  mutate(name = str_replace_all(name, "BK", "BANK")) %>%
 filter(str_detect(name, " BANK ")) -> middle_banks

nrow(middle_banks)  #36 have "BANK" neither in the first nor in the last word
```

```
## [1] 36
```

```r
prop_middle <- (nrow(middle_banks)/ nrow(US_commercial_banks_V02))*100
prop_middle
```

```
## [1] 9.6
```

```r
# traditional method

US_commercial_banks_V02 %>%
  mutate(name = str_replace_all(name, "BK", "BANK")) %>% #
  anti_join(first_banks) %>%
  anti_join(last_banks) %>%
  anti_join(middle_banks) -> no_banks
```

```
## Joining, by = c("name", "alternate_name", "rank", "charter",
## "consolidated_assets")Joining, by = c("name", "alternate_name", "rank",
## "charter", "consolidated_assets")Joining, by = c("name", "alternate_name",
## "rank", "charter", "consolidated_assets")
```

```
nrow(no_banks) # 69 have no bank words
```

```
## [1] 69
```

```
prop_no <- (nrow(no_banks)/ nrow(US_commercial_banks_V02))*100
prop_no
```

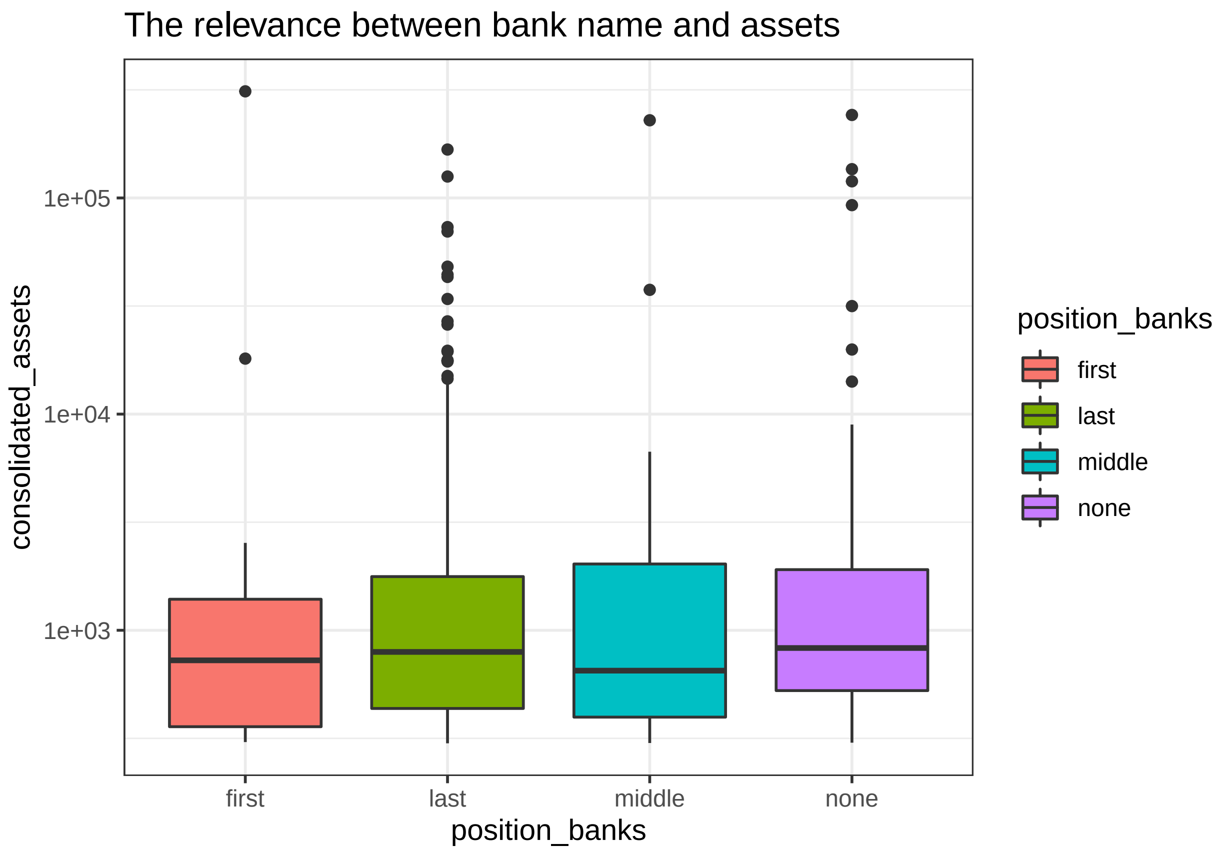
```
## [1] 18.4
```

6. Extra Credit: Use a boxplot to compare the distributions of the log of the combined total assets of the banks based on where the word "BANK" appears in their name. Does position seem to make a difference?

Ans: the position of the bank's name does not have inseparable effects on total assets.

```
US_commercial_banks_V03 %>%
  ggplot(aes(x = position_banks, y = consolidated_assets, fill = position_banks))+
  geom_boxplot()+
  scale_y_log10()+
  theme_bw()+
  ggtitle("The relevance between bank name and assets")
```

```
## Warning: Removed 1 rows containing non-finite values (stat_boxplot).
```



```
# Note:
# 1e+03 = 1 * 10^3 = 1000
# 1e+04 = 1 * 10^4 = 10000
# 1e+05 = 1 * 10^5 = 100000
# 1e-03 = 1 * 10^(-3) = 0.001
```

Extra Credit Question: Create your own date parser.

1. Create a function called my_d_parser() that takes as input a string of eight digits and a format (specified by with a #Y, #m, and #d;for YYYY, MM, and DD, respectively) and returns a vector of length 3 where the first element is the year, the second is the month, and the third is the day.

Hints:

– I used the following functions: *str_replace()*, *str_locate()*, *rank()*, *c()*, *str_match()*, *as.numeric()*.
– Thinking in reverse order, you need to create a regex pattern based on the order of the terms in the input format you can use to parse the input string into vector of year, month, and day.
– Start by using regex to convert the input format (#Y, #m, #d) into a new regex string
* As an example, use regex to replace "#Y" with "([0-9]{4})"
– Create a vector with the order of the elements in the input format.
– Create a vector of the matches for the input string with the new parsing regex
– Using the vector you created for the order of the input format to return the elements matching year, month, and day
– Make sure each element of the output vector is a number.

- You are not allowed to use any pre-built date parsers.
- You have to use regular expressions.
- Test out your parser on the following three inputs.

```
my_d_parser <- function(string, pattern) {
 pattern_2 <- str_replace(pattern, "#Y", "YYYY")
 pattern_2 <- str_replace(pattern_2, "#m", "mm")
 pattern_2 <- str_replace(pattern_2, "#d", "dd")

 y_index <- str_locate(pattern_2, "YYYY")
 year <- as.numeric(str_sub(string, y_index[1], y_index[2]))

 m_index <- str_locate(pattern_2, "mm")
 month <- as.numeric(str_sub(string, m_index[1], m_index[2]))

 d_index <- str_locate(pattern_2, "dd")
 day <- as.numeric(str_sub(string, d_index[1], d_index[2]))

 return(c(year, month, day))
}
```

```
pattern <- "#Y, #d, #m"
string <- "2021, 12, 02"
my_d_parser(string, pattern)
```

```
## [1] 2021    2   12
```

```
pattern <- "#d-#Y,#m"
string <- "01-2020,05"
my_d_parser(string, pattern)
```

```
## [1] 2020    5    1
```

```
pattern <- "#m/#d/#Y"
string <- "05/29/2017"
my_d_parser(string, pattern)
```

```
## [1] 2017    5   29
```

```
#another solution

my_d_parser_anotherway <- function(string, pattern){
  str_replace(pattern, pattern = "#Y", "([0-9]{4})") %>% #{} exactly{n}
    str_replace(pattern = "#m", "([0-9]{2})") %>%
    str_replace(pattern = "#d", "([0-9]{2})") -> pattern_V02

  dpattern <- str_locate(pattern, "#d")[1]
  mpattern <- str_locate(pattern, "#m")[1]
  ypattern <- str_locate(pattern, "#Y")[1]
  combined <- rank(c(ypattern, mpattern, dpattern))
  str_match(string = string, pattern = pattern_V02)[, -1] %>%
    as.numeric() -> finalparser
  return(finalparser[combined])

}
```