HW: Week 3

36-350 – Statistical Computing

Week 3 – Fall 2020

Name: Kimberly Zhang

Andrew ID: kyz

You must submit **your own** lab as a PDF file on Gradescope.

```
trump.lines = readLines("http://www.stat.cmu.edu/~pfreeman/trump.txt")
```

Question 1

(10 points)

Display the lines of text in `trump.lines` that contain both of the strings "America" *and* "great" (in any order, separated by any amount of text). Do so only using regexp literals.

```
grep("(America.*great)|(great.*America)", trump.lines, value = TRUE)
```

```
## [1] "I have seen firsthand how the system is rigged against our citizens,
just like it was rigged against Bernie Sanders - he never had a chance. But his
supporters will join our movement, because we will fix his biggest issue:
trade. Millions of Democrats will join our movement, because we are going to
fix the system so it works fairly and justly for each and every American. In
this cause, I am proud to have at my side the next Vice President of the United
States: Governor Mike Pence of Indiana. And a great guy."
## [2] "We are going to build a great border wall to stop illegal immigration,
to stop the gangs and the violence, and to stop the drugs from pouring into our
communities. I have been honored to receive the endorsement of America's Border
Patrol Agents, and will work directly with them to protect the integrity of our
lawful, lawful immigration System. By ending catch-and-release on the border,
we will stop the cycle of human smuggling and violence."
## [3] "We are going to be considerate and compassionate to everyone. But my
greatest compassion will be for our own struggling citizens. My plan is the
exact opposite of the radical and dangerous immigration policy of Hillary
Clinton. Americans want relief from uncontrolled immigration. Communities want
relief from uncontrolled immigration which is what we have now. Communities
want relief."
## [4] "Next comes the reform of our tax laws, regulations and energy rules.
While Hillary Clinton plans a massive tax increase, I have proposed the largest
tax reduction of any candidate who has declared for the presidential race this
year - Democrat or Republican. Middle-income Americans will experience profound
relief, and taxes will be greatly simplified for everyone I mean everyone."
## [5] "America is one of the highest-taxed nations in the world. Reducing
```

```
taxes will cause new companies and new jobs to come roaring back into our
country. Believe it will happen and it will happen fast. Then we are going to
deal with the issue of regulation, one of the greatest job-killers of them
all."
## [6] "We can accomplish these great things and so much more. All we need to
do is start believing in ourselves and in our country again. Start believing.
It is time to show the whole world that America is become, bigger and better
and stronger our country again. Start believing. It is time to show the whole
world that America is become, bigger and better and stronger than ever before."
## [7] "I have had a truly great life in business, but now my sole and
exclusive mission is to go to work for our country, to go to work for you. It's
time to deliver a victory for the American people. We don't win anymore, but we
are going to start winning again. But to do that, we must break free from the
petty politics of the past."
## [8] "To all Americans tonight in all of our cities and in all of our towns,
I make this promise -- we will make America strong again. We will make America
proud again. We will make America safe again. And we will make America great
again."
```

Question 2

(10 points)

Retrieve (but don't display) the lines of text in `trump.lines` that contain "Trump". Then break the retrieved lines into individual words (`strsplit(input," ")` to split the character vector `input` into words separated by spaces), and merge those words into a single character vector (`unlist()`!). How many unique words are there? Display the top five most commonly occurring words and how often they occur (combine `sort()` and `table()`!)

```
input = grep("Trump", trump.lines, value = TRUE)
words = strsplit(input, " ")
head(sort(table(unlist(words)), decreasing = TRUE), 5)
```

```
##
##   of  the  and    a love
##    7    7    6    4    4
```

Question 3

(10 points)

In Q25 of Lab 3, you coded a regex to match all patterns of the following form: any letter (1 or more), then a punctuation mark, then "ve" or "ll" or "t", then a space or a punctuation mark. You called it `my.pattern`. Use `my.pattern`, along with `regexpr()` and `regmatches()`, to extract and display all the occurrences of the pattern in `trump.lines`. Then repeat the exercise using `gregexpr()` instead of `regexpr`; note that here, you'll want to `unlist()` the output from `regmatches()`. Do you get the same vector of character strings? Why or why not?

```
my.pattern = "[[:alpha:]]+[[:punct:]](ve|ll|t)( |[[:punct:]])"
reg_exp = regexpr(my.pattern, trump.lines, useBytes = TRUE)
regmatches(trump.lines, reg_exp)
```

```
##  [1] "would've " "I've "     "won't "    "don't "    "won't "    "we'll "
##  [7] "I'll "     "don't "    "can't "    "don't "
```

```
greg_exp = gregexpr(my.pattern, trump.lines)
unlist(regmatches(trump.lines, greg_exp))
```

```
##  [1] "would've " "would've " "would've " "I've "     "wasn't "   "won't "
##  [7] "can't "    "don't "    "won't "    "we'll "    "don't "    "I'll "
## [13] "they've "  "don't "    "can't "    "wouldn't " "doesn't "  "don't "
## [19] "Don't "    "don't "
```

```
You don't get the same vector of character strings because regexpr only gets
the first matching substring while gregexpr gets all matching substrings for
a given string.
```

Question 4

(10 points)

Come up with a strategy that splits punctuation marks or spaces, except that it keeps intact words like "I've" or "wasn't", that have a punctuation mark in the middle, in between two letters. (Or when the punctuation mark is at the beginning, as in "'em", or when there is a dollar sign at the beginning.) Apply your strategy to `trump.words` as defined below such that you display only those words with punctuation marks and/or dollar signs. (Note that I end up with 102 [not necessarily unique, but total] words when I implement this strategy. Some include '", which we can easily remove in a subsequent post-processing step if we so choose. Note also that a dollar sign is *not* a punctuation mark; this will affect how you define your regex. Hint: `[[:alnum:]]` is a good thing to use here.)

```
match_pattern = "(^[[:punct:]][[:alnum:]])|([[:alnum:]][[:punct:]][[:alnum:]])"
data = unlist(strsplit(trump.lines, split = "[[:space:]]|[^([[:punct:]][[:alnum:]])]"))
length(grep(match_pattern, data, value = TRUE))
```

```
## [1] 98
```

```
grep(match_pattern, data, value = TRUE)
```

```
##  [1] "would've"          "would've"          "would've"
##  [4] "carefully-crafted" "Administration's"  "America's"
##  [7] "That's"            "nation's"          "President's"
## [10] "2,000"             "4,000"             "180,000"
## [13] "border-crosser"    "years-old,"        "4.0"
## [16] "I've"              "Sarah's"           "wasn't"
## [19] "African-American"  "African-American"  "$4,000"
## [22] "that's"            "$800"              "$800"
## [25] "We're"             "$19"               "forty-three"
## [28] "$150"              "America's"         "Let's"
## [31] "Let's"             "pre-Hillary,"      "Clinton's"
## [34] "America's"         "nation's"          "it's"
## [37] "Clinton's"         "laid-off"          "they're"
## [40] "won't"             "33,000"            "can't"
## [43] "\"extremely"       "\"negligent,\""    "America's"
## [46] "America's"         "It's"              "It's"
## [49] "It's"              "It's"              "we're"
## [52] "don't"             "there's"           "African-American"
## [55] "it's"              "America's"         "catch-and-release"
## [58] "won't"             "It's"              "I'm"
## [61] "nearly-one"        "China's"           "husband's"
## [64] "it's"              "China's"           "we'll"
## [67] "don't"             "Middle-income"     "highest-taxed"
## [70] "job-killers"       "$2"                "that's"
## [73] "she's"             "that's"            "she's"
## [76] "We're"             "ten-point"         "I'm"
## [79] "I'll"              "they've"           "I'm"
```

```
## [82] "I'm"              "he'd"             "It's"
## [85] "there's"          "'em"              "It's"
## [88] "don't"            "can't"            "wouldn't"
## [91] "doesn't"          "don't"            "Don't"
## [94] "don't"            "It's"             "three-word"
## [97] "\"I'm"            "\"I'm"
```

Below, we read in lines of data from the Advanced National Seismic System (ANSS), on earthquakes of magnitude 6+, between 2002 and 2017. (You don't have to do anything yet.)

```
anss.lines = readLines("http://www.stat.cmu.edu/~pfreeman/anss.htm")
date.pattern = "[0-9]{4}/[0-9]{2}/[0-9]{2}"
date.lines = grep(date.pattern,anss.lines,value=TRUE)
```

Question 5

(10 points)

Check that all the lines in `date.lines` actually start with a date, of the form YYYY/MM/DD. rather than contain a date of this form somewhere in the middle of the text. (Hint: it might help to note that you can look for non-matches, as opposed to matches, by changing one of `grep()`'s logical arguments.)

```
grep("^[0-9]{4}/[0-9]{2}/[0-9]{2}", date.lines, invert = TRUE, value = TRUE)
```

```
## character(0)
```

Question 6

(10 points)

Which five days witnessed the most earthquakes, and how many were there, these days? Also, what happened on the day with the most earthquakes: can you find any references to this day in the news?

```
split_data = unlist(strsplit(date.lines, split = " +"))
years = grep("[0-9]{4}/[0-9]{2}/[0-9]{2}", split_data, value = TRUE)
head(sort(table(years), decreasing = TRUE), 5)
```

```
## years
## 2011/03/11 2010/02/27 2004/12/26 2006/11/15 2013/02/06
##         42         12         11          7          7
```

The five days that witnessed the most earthquakes were 2011/03/11, 2010/02/27, 2004/12/26, 2006/11/15, and 2013/02/06; there were 42, 12, 11, 7, and 7 earthquakes respectively. 2011/03/11 was when the 9.0 magnitude earthquake and tsunami hit Japan; it was called the Great Tohoku Earthquake. It was the most powerful earthquake to have ever hit Japan, the fourth most powerful in the world since 1900s.

Question 7

(10 points)

Go back to the data in `date.lines`. Following steps similar to the ones you used in the lab to extract the latitude and longitude of earthquakes, extract the depth and magnitude of earthquakes. In the end, you should have one numeric vector of depths, and one numeric vector of magnitudes. Show the first three depths and the first three magnitudes. (Hint: if you use `regexpr()` and `regmatches()`, then the output

from the latter will be a vector of strings. Look at this vector. The last four characters always represent the magnitudes. Use a combination of `substr()` and `as.numeric()` to create the numeric vector of magnitudes. Then use the fact that everything but the last four characters represents the depths. There are a myriad of ways to do this exercise, but this suggested way is the most concise.)

```
pattern = "[0-9]{2,3}.[0-9]{2} +?[0-9].[0-9]{2}"
reg_exp = regexpr(pattern, date.lines, useBytes = TRUE)
matches = regmatches(date.lines, reg_exp)
depth = as.numeric(substr(matches, 1, nchar(matches)-4))
head(depth, 3)
```

```
## [1]  10.00  22.73 665.80
```

```
magnitude= as.numeric(substr(matches, nchar(matches)-3, nchar(matches)))
head(magnitude, 3)
```

```
## [1] 6.0 6.3 6.2
```

Here we read in text containing the fastest men's 100-meter sprint times. We retain only the lines that correspond to the sprint data, for times 9.99 seconds or better.

```
sprint.lines = readLines("http://www.stat.cmu.edu/~pfreeman/men_100m.html")
data.lines = grep(" +(9|10)\\.",sprint.lines)
sprint.lines = sprint.lines[min(data.lines):max(data.lines)]
```

Question 8

(10 points)

Extract the years in which the sprint times were recorded. Display them in table format. Do the same for the months. Be sure to extract the month of the sprint, not the birth month of the sprinter! (Hint: the month of the sprint is followed by a four-digit year; other instances of two digits in any given line are not. So you may have to extract more than you need, then apply `strsplit()`.)

```
yr_pattern = "[0-9]{2}.[0-9]{4}"
reg_exp = regexpr(yr_pattern, sprint.lines, useBytes = TRUE)
dates = regmatches(sprint.lines, reg_exp)
split_dates = unlist(strsplit(dates, split = "\\."))
months = split_dates[c(TRUE, FALSE)]
years = split_dates[c(FALSE, TRUE)]
table(years)
```

```
## years
## 1968 1977 1983 1984 1985 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996
1997
## 1 1 3 4 1 1 9 2 2 13 6 8 16 5 24 38
## 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
2013
## 33 26 14 16 16 9 33 16 24 18 53 45 36 57 61 59
## 2014 2015 2016 2017 2018 2019
## 32 91 60 48 57 50
```

```
table(months)
```

```
## months
```

```
##  02  03  04  05  06  07  08  09  10
##   3   7  28  99 244 223 277 104   3
```

Question 9

(10 points)

Extract the countries of origin (for the sprinters). Note that countries of origin are given as a capitalized three-letter abbreviation. Display the table of country origins. Display the proportion of the list that is accounted for by sprinters from the US and Jamaica.

```
country_pattern = "[A-Z]{3}"
reg_exp = regexpr(country_pattern, sprint.lines, useBytes = TRUE)
matches = regmatches(sprint.lines, reg_exp)
table(matches)
```

```
## matches
## AHO ANT AUS BAH BAR CAN CAY CHN CIV CUB FRA GBR GHA ITA JAM JPN NAM NED NGR
NOR
## 5 7 1 3 4 37 1 7 11 2 30 35 4 1 267 4 27 1 25 1
## OMA POR QAT RSA SKN TTO TUR USA ZIM
## 1 4 8 29 10 51 3 406 3
```

```
as.numeric((table(matches)['USA'] + table(matches)['JAM'])/sum(as.numeric(table(matches))))
```

```
## [1] 0.6811741
```

Question 10

(10 points)

We conclude with a web scraping exercise. I want you to go to this web site. On it, you see there is a set of 12 bold-faced four-digit numbers: this is the number of submitted astrophysics articles for each month of 2019. I want you to extract these numbers and place them into a single vector, with each vector element having a name: Jan for the first vector element, Feb for the second, etc. You would use `readLines()` to read in the page (pass the URL directly to the function!); this creates a vector of strings. You would then use `regexpr()` and `regmatches()` to extract the numbers (plus potentially some other stuff that you may have to pare off using `substr()`). If necessary, use "view source" to look at the html code for the web page itself to determine how best to extract the 12 numbers and nothing else. You don't want to create a table; you simply want to output the vector of four-digit numbers and add the appropriate names. (Hint: see the documentation for `Constants`. `month.abb` might be helpful here.)

```
article.lines = readLines("https://arxiv.org/year/astro-ph/19")
article.lines = grep("\\([[:alpha:]]{3} 2019\\)",article.lines, value = TRUE)
reg_exp = regexpr("<b>[0-9]+</b>", article.lines, useBytes = TRUE)
matches = regmatches(article.lines, reg_exp)
clean_matches = as.numeric(substr(matches, 4, nchar(matches)-4))
names(clean_matches) = month.abb
clean_matches
```

```
##  Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 1125 1038 1475 1258 1159 1023 1277 1225 1344 1295 1100 1100
```