Homework 3
STAT 4410/8416 Sections 001/002
Brian Detweiler Fall 2016
Due: October 23, 2016 by midnight

1. **Regular expression:** Write a regular expression to match patterns in the following strings. Demonstrate that your regular expression indeed matched that pattern by including codes and results. Carefully review how the first problem is solved for you.

   (a) We have a vector `vText` as follows. Write a regular expression that matches `g, og, go or ogo` in `vText` and replace the matches with dots.

```
library(stringr)
library(tm)
library(data.table)
library(dplyr)
library(reshape2)
library(ggplot2)
library(SnowballC)
library(RColorBrewer)
library(wordcloud)
```

```
vText <- c('google','logo','dig', 'blog', 'boogie' )
```

   Answer:

```
pattern <- 'o?go?'
gsub(pattern, '.', vText)
```

```
## [1] "..le"  "l."    "di."   "bl."   "bo.ie"
```

   (b) Replace only the 2 or 3 digit numbers with the word `found` in the following vector. Please make sure that 4, 5 or 1 digit numbers do not get changed.

```
vPhone <- c('35','6783','345', '20', '46349', '8' )
```

   Answer:

```
pattern <- '^[0-9]{2,3}$'
gsub(pattern, 'found', vPhone)
```

```
## [1] "found" "6783"  "found" "found" "46349" "8"
```

   (c) Replace all the characters that are not among the 26 English characters or a space. Please replace with an empty spring.

```
myText <- "#y%o$u @g!o*t t9h(e) so#lu!tio$n c%or_r+e%ct"
```

   Answer:

```
pattern <- '[^A-Za-z ]+'
gsub(pattern, '', myText)
```

```
## [1] "you got the solution correct"
```

   (d) In the following text, replace all the words that are exactly 3 characters long with triple dots '...'

```
myText <- "All the three character words should be gone now"
```

Answer:

```
pattern <- '(^\\w{3}?|(( )\\w{3}\\b)|\\w{3}$)'
gsub(pattern, '\\3...', myText)

## [1] "... ... three character words should be gone ..."
```

(e) Extract all the three numbers embedded in the following text.

```
bigText <- 'There were three 20@14 numbers hidden in 500 texts'
```

Answer:

```
pattern <- '([^0-9])'
trimws(gsub(pattern, ' ', bigText))

## [1] "20 14                   500"

nums <- unique(as.numeric(unlist(strsplit(gsub(pattern, ' ', bigText), split = ' '))))
nums[which(!is.na(nums))]

## [1]  20  14 500
```

(f) Extract all the words between parenthesis from the following string text and count number of words.

```
myText <- 'The salries are reported (in millions) for every companies.'
```

Answer:

```
pattern <- '.*\\((.*)\\).*'
parens <- gsub(pattern, '\\1', myText)
parens

## [1] "in millions"

vParens <- unlist(strsplit(parens, split = " "))
length(vParens)

## [1] 2
```

(g) Extract only the word in between pair of symbols $. Count number of words you have found between pair of dollar sign $.

```
myText <- 'Math sumbols are $written$ in $between$ dollar $signs$'
```

Answer:

```
vMyText <- unlist(strsplit(myText, split = ' '))
detect.pattern <- '^\\$.*\\$$'
replace.pattern <- '\\$'
words.found <- gsub(replace.pattern, '', vMyText[str_detect(vMyText, detect.pattern)])
words.found

## [1] "written" "between" "signs"

length(words.found)

## [1] 3
```

(h) Extract all the letters of the following sentence and prove that it contains all the 26 letters.

```
myText <- 'the quick brown fox jumps over the lazay dog'
```

Answer:

```
pattern <- ' ?([a-z]) ?'
alpha <- sort(unique(unlist(strsplit(gsub(pattern, '\\1 ', myText), split = c(' ')))))
alpha
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q"
## [18] "r" "s" "t" "u" "v" "w" "x" "y" "z"
```

```
length(alpha)
```

```
## [1] 26
```

2. Download the sample of a big data from blackboard. Note that the data is in csv format and compressed for easy handling. Now answer the following questions.

```
big.data <- fread('bigDataSample.csv')
```

(a) Read the data and select only the columns that contains the word 'human'. Store the data in an object `dat`. Report first few rows of your data.

Answer:

```
dat <- select(big.data, contains('human'))
head(dat)
```

```
##    var_human_1_g var_human_1_p var_human_1_b var_human_1_e var_human_1_n
## 1:      18.99545            21             1    21.6321136      26.03268
## 2:      15.02303            34             3     0.3838458      26.92529
## 3:      37.44410            28             2    33.4801022      39.30039
## 4:      36.33714            26             2     2.8761174      33.75177
## 5:      21.06330            25             1     3.1657313      26.19248
## 6:      16.52637            35             2     5.3108922      25.07192
```

(b) The data frame `dat` should have 5 columns. Rename the column names keeping only the last character of the column names. So each column name will have only one character. Report first few rows of your data now.

Answer:

```
cols <- colnames(dat)
pattern <- '.*([a-z])$'
cols <- gsub(pattern, '\\1', cols)
colnames(dat) <- cols
head(dat)
```

```
##           g  p b          e        n
## 1: 18.99545 21 1 21.6321136 26.03268
## 2: 15.02303 34 3  0.3838458 26.92529
## 3: 37.44410 28 2 33.4801022 39.30039
## 4: 36.33714 26 2  2.8761174 33.75177
## 5: 21.06330 25 1  3.1657313 26.19248
## 6: 16.52637 35 2  5.3108922 25.07192
```

(c) Compute and report the means of each columns group by column b in a nice table.

Answer:

```
dat %>%
  group_by(b) %>%
  summarise_each(funs(mean))
```

```
## # A tibble: 4  5
##       b        g        p         e        n
##   <int>    <dbl>    <dbl>     <dbl>    <dbl>
## 1     0 28.74877 23.75862 12.214718 29.44332
## 2     1 22.47859 25.28302 10.418129 29.34315
## 3     2 23.85395 24.94624  9.615341 30.62800
## 4     3 23.81182 25.40909 10.481746 30.25341
```

(d) Change the data into long form using id='b' and store the data in `mdat`. Report first few rows of data.

Answer:

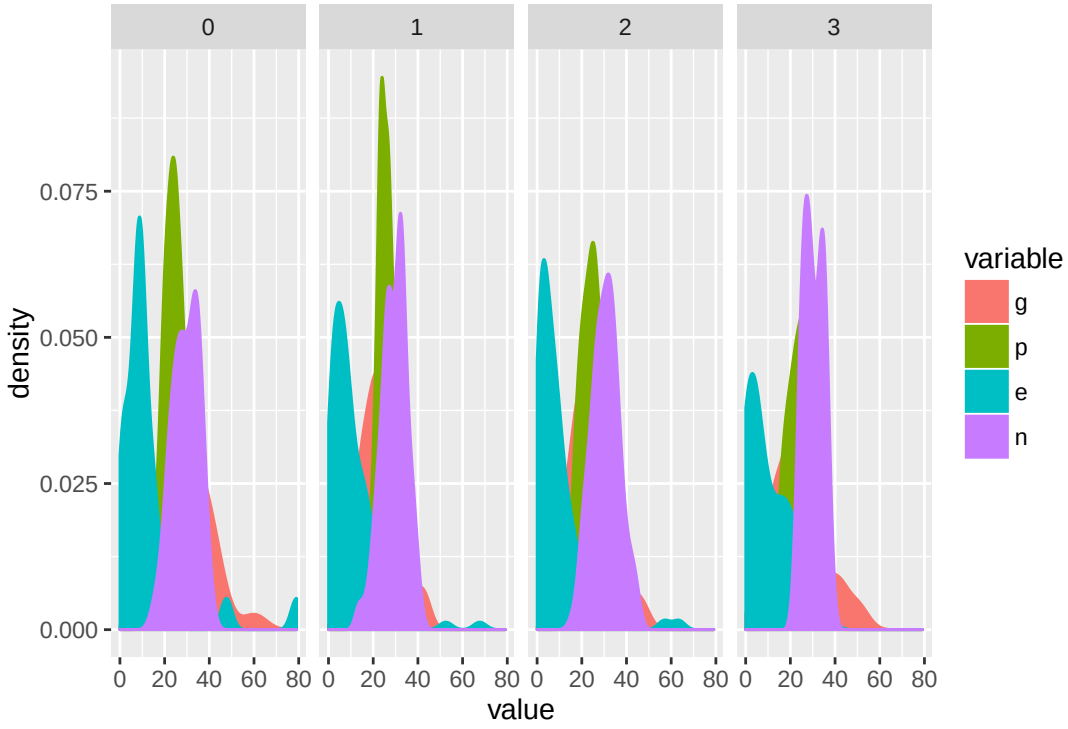
```
mdat <- melt(dat, id.vars = 'b')
head(mdat)
```

```
##    b variable    value
## 1: 1        g 18.99545
## 2: 3        g 15.02303
## 3: 2        g 37.44410
## 4: 2        g 36.33714
## 5: 1        g 21.06330
## 6: 2        g 16.52637
```

(e) The data frame `mdat` is now ready for plotting. Generate density plots of value, color and fill by variable and facet by b.

Answer:

```
ggplot(mdat, aes(x = value, colour = variable, fill = variable)) +
  geom_density() +
  facet_grid(. ~ b)
```



(f) The data set `bigDataSample.csv` is a sample of much bigger data set. Here we read the data set and then selected the desired column. Do you think it would be wise do the same thing with the actual larger data set? Explain how you will solve this problem of selecting few columns (as we did in question 2a) without reading the whole data set first. Demonstrate that showing your codes.

Answer: We can first get the column names using nrows,

```
columns <- colnames(read.csv(file = 'bigDataSample.csv', nrows = 1, header = T))
length(columns)
```

```
## [1] 2501
```

Then, we can determine what columns we want, using `stringr`,

```
columns <- columns[str_detect(columns, '.*human.*')]
columns
```

```
## [1] "var_human_1_g" "var_human_1_p" "var_human_1_b" "var_human_1_e"
## [5] "var_human_1_n"
```

Finally, we can read just those columns very quickly using `data.table::fread`,

```
dat <- fread('bigDataSample.csv',
             select = columns)
head(dat)
```

```
##    var_human_1_g var_human_1_p var_human_1_b var_human_1_e var_human_1_n
## 1:      18.99545            21             1    21.6321136      26.03268
## 2:      15.02303            34             3     0.3838458      26.92529
## 3:      37.44410            28             2    33.4801022      39.30039
## 4:      36.33714            26             2     2.8761174      33.75177
## 5:      21.06330            25             1     3.1657313      26.19248
## 6:      16.52637            35             2     5.3108922      25.07192
```

3. **Extracting data from web:** Our plan is to extract data from web sources. This includes email addresses, phone numbers or other useful data. The function `readLines()` is very useful for this purpose.

   (a) Please read all the text in http://www.unomaha.edu/mahbubulmajumder/index.html and store your texts in `myText`. Show first few rows of `myText` and examine the structure of the data.
   Answer:

```
unomaha <- 'http://www.unomaha.edu/mahbubulmajumder/index.html'
myText <- readLines(unomaha)
head(myText)
```

```
## [1] "<!DOCTYPE html>"
## [2] "<head>"
## [3] "<link rel=\"stylesheet\" href=\"script/style.css\">"
## [4] "<link href=\"http://maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-awesome.min.cs
## [5] ""
## [6] "<title>Mahbubul Majumder</title>"
```

   (b) Write a regular expression that would extract all the http web links addresses from `myText`. Include your codes and display the results that show only the http web link addresses and nothing else.
   Answer:

```
pattern <- '.*<a +href="(.*)" ?(target.*)?.*'
links <- myText[str_detect(myText, pattern)]
gsub(pattern, '\\1', links)
```

```
## [1] "index.html"                    "html/publication.html"
## [3] "html/teaching.html"            "html/experiments.html"
## [5] "html/datavisualization.html"   "html/webresources.html"
## [7] "documents/resume.pdf"          "mailto:mmajumder@unomaha.edu"
## [9] "http://www.unomaha.edu/math/"
```

   (c) Now write a regular expression that would extract all the emails from `myText`. Include your codes and display the results that show only the email addresses and nothing else.
   Answer:

```
pattern <- '.*(\\b[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}\\b).*'
emails <- myText[str_detect(myText, pattern)]
gsub(pattern, '\\1', emails)
```

```
## [1] "mmajumder@unomaha.edu"
```

   (d) Now we want to extract all the phone/fax numbers in `myText`. Write a regular expression that would do this. Demonstrate your codes showing the results.
   Answer:

```
pattern <- '[0-9]'
phone <- myText[str_detect(myText, pattern)]
pattern <- '[^0-9]'
phone <- gsub(pattern, '', phone)
phone <- phone[str_length(phone) == 10]
pattern <- '([0-9]{3})([0-9]{3})([0-9]{4})'
gsub(pattern, '(\\1) \\2-\\3', phone)
```

```
## [1] "(402) 554-2734" "(402) 554-2975"
```

(e) The link of ggplot2 documentation is http://docs.ggplot2.org/current/ and we would like to get the list of ggplot2 geoms from there. Write a regular expression that would extract all the geoms names (geom_bar is one of them) from this link and display the unique geoms. How many unique geoms does it have?

Answer:

```
geoms <- readLines('http://docs.ggplot2.org/current')
pattern <- '.*(geom_[a-z]+)\\.html.*'
geoms <- geoms[str_detect(geoms, pattern)]
geoms <- gsub(pattern, '\\1', geoms)
geoms
```

```
##  [1] "geom_abline"    "geom_bar"       "geom_blank"     "geom_boxplot"
##  [5] "geom_contour"   "geom_count"     "geom_linerange" "geom_density"
##  [9] "geom_dotplot"   "geom_errorbarh" "geom_histogram" "geom_hex"
## [13] "geom_jitter"    "geom_text"      "geom_map"       "geom_path"
## [17] "geom_point"     "geom_polygon"   "geom_quantile"  "geom_tile"
## [21] "geom_ribbon"    "geom_rug"       "geom_segment"   "geom_smooth"
## [25] "geom_violin"    "geom_spoke"
```

4. Download `lincoln-last-speech.txt` from the blackboard which contains the Lincoln's last public address. Now answer the following questions and include your codes.

(a) Read the text and store the text in `lAddress`. Show the first 70 characters from the first element of the text.

Answer:

```
lAddress <- readLines('lincoln-last-speech.txt')
substr(lAddress[1], 1, 70)
```

```
## [1] "We meet this evening, not in sorrow, but in gladness of heart. The eva"
```

(b) Now we are interested in the words used in his speech. Extract all the words from `lAddress`, convert all of them to lower case and store the result in `vWord`. Display first few words.

Answer:

```
vWord <- unlist(strsplit(lAddress, split = ' '))
head(vWord)
```

```
## [1] "We"       "meet"     "this"     "evening," "not"      "in"
```

(c) The words like 'am', 'is', 'my' or 'through' are not much of our interest and these types of words are called stop-words. The package 'tm' has a function called `stopwords()`. Get all the English stop words and store them in `sWord`. Display few stop words in your report.

Answer:

```
sWord <- stopwords(kind = 'en')
head(sWord)
```

```
## [1] "i"      "me"     "my"     "myself" "we"     "our"
```

(d) Remove all the `sWord` from `vWord` and store the result in `cleanWord`. Display first few clean words.

Answer:

```
cleanWord <- vWord
# Remove hyphens
cleanWord <- gsub("-", "", cleanWord)
# Remove non-english characters
cleanWord <- gsub("\\W*(\\w+)\\W*", "\\1", cleanWord)
# Lowercase everything
cleanWord <- tolower(cleanWord)
# Remove stopwords
cleanWord <-  cleanWord[which(!(cleanWord %in% sWord))]

head(cleanWord)
```

```
## [1] "meet"       "evening"    "sorrow"     "gladness"   "heart"
## [6] "evacuation"
```

(e) `cleanWord` contains all the cleaned words used in Lincoln's address. We would like to see which words are more frequently used. Find 15 most frequently used clean words and store the result in `fWord`. Display first 5 words from `fWord` along with their frequencies.

Answer:

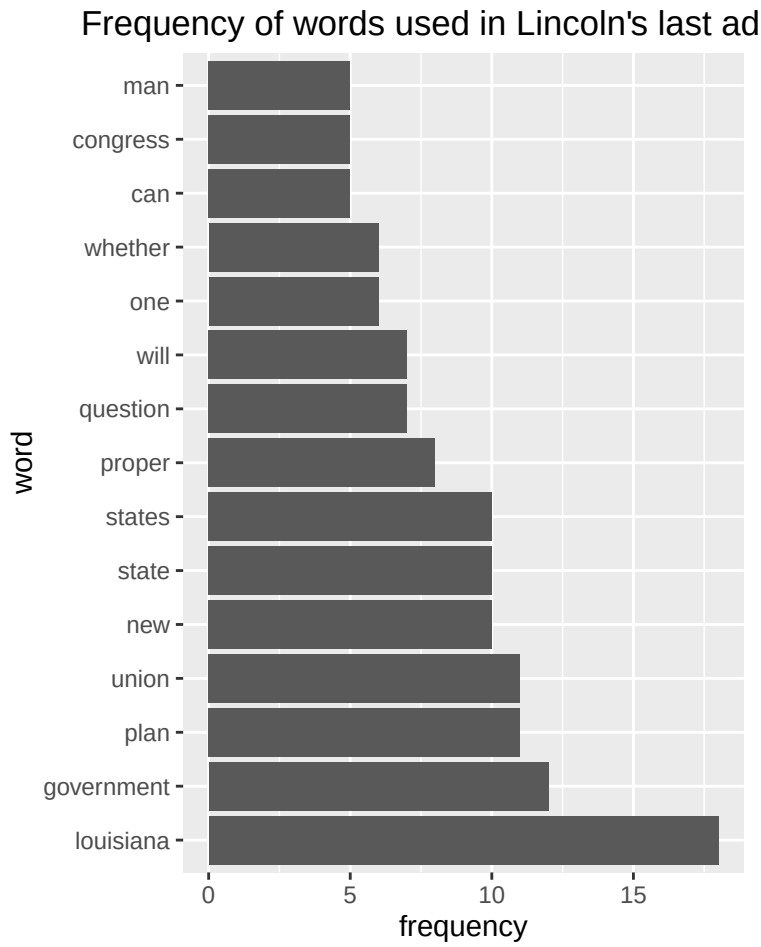
```
# Remove stopwords

fWord <- sort(table(cleanWord), decreasing = TRUE)
fWord[1:5]
```

```
## cleanWord
##  louisiana government       plan      union        new
##         18         12         11         11         10
```

(f) Construct a bar chart showing the count of each words for the 15 most frequently used words. Add a layer '`+coord_flip()`' with your plot.

Answer:

```
wordsDF <- as.data.frame(fWord[1:15])
ggplot(wordsDF, aes(x = cleanWord, y = Freq)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "Frequency of words used in Lincoln's last address",
       x = "word",
       y = "frequency")
```



(g) What is the reason for adding a layer '`+coord_flip()`' with the plot in question (4f). Explain what would happen if we would not have done that.

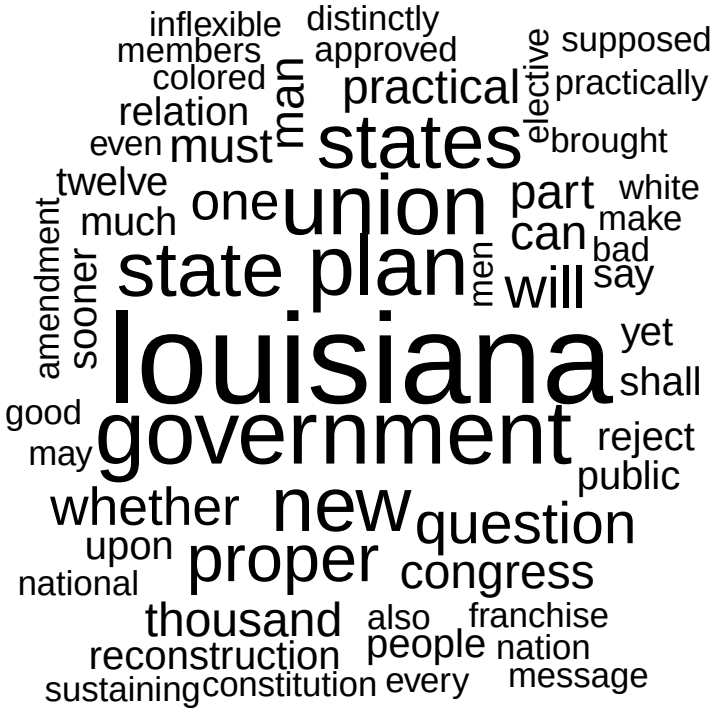
Answer: `coord_flip()` swaps the x-y axes of the plot. If we had not done this, the word labels can easily overwrite each other and the plot becomes very difficult, if not impossible to read. Another approach is to angle the labels 45 degrees, but this is slightly more difficult to read since the labels don't line up directly with the bars.

(h) The plot in question (4f) uses bar plot to display the data. Can you think of another plot that delivers the same information but looks much simpler? Demonstrate your answer by generating such a plot.

Answer:

We can display this data in an aesthetically pleasing wordcloud. Wordclouds are not good for quantitative analysis, but can be useful for exploratory analysis or as a supplement to quantitative reporting.

```
corp <- Corpus(VectorSource(cleanWord))

wordcloud(corp, max.words = 100, random.order = FALSE)
```


louisiana
government
union
states
plan
state
new
proper
question
congress
practical
thousand
reconstruction
people
inflexible
distinctly
members
approved
supposed
colored
practically
relation
brought
even
must
twelve
one
part
white
much
make
can
bad
say
will
yet
shall
good
may
reject
public
whether
upon
national
also
franchise
nation
every
message
sustaining
constitution
amendment
sooner
men
man
elective