STAT 4410/8416 Homework 3

Gerjol, Nicholas

Due on Oct 30, 2021

1. Text Data analysis: Download "lincoln-last-speech.txt" from Canvas which contains 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.

```
lAddress <- 'lincoln-last-speech.txt'
lAddress <- readChar(lAddress, file.info(lAddress)$size)
substr(lAddress, start=1, stop=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.

```
library(stringr)
vWord <- tolower(lAddress)
vWord <- str_extract_all(vWord, "[a-z]+")
lapply(vWord,head)
```

```
## [[1]]
## [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.

```
library(tm)
sWord <- stopwords()
sWord[1:7]
```

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

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

```
check <- as.data.frame(unlist(vWord) %in% sWord)
cleanWord <- as.data.frame(unlist(vWord))
cleanWord$check <- check
colnames(cleanWord) <- c("a","b")
partf <- cleanWord
cleanWord <- subset(cleanWord, b==FALSE)
cleanWord <- cleanWord$a
head(cleanWord)
```

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

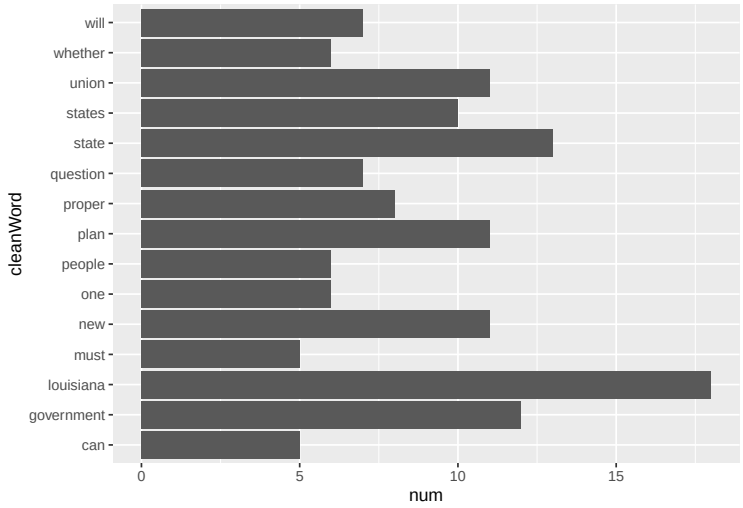
e) `cleanWord` contains all the cleaned words used in Lincoln's address. We would like to see which words are more frequently used. Find 15 most frequently used clean words and store the result in `fWord`. Display first 5 words from `fWord` along with their frequencies.

```
cleanWord <- as.data.table(cleanWord)
cleanWord[, `num` := .N, by = cleanWord]
data <- arrange(cleanWord,desc(num))
cleanWord <- data %>% filter(duplicated(cleanWord) == FALSE)
cleanWord[1:5]
```

```
##     cleanWord num
## 1:  louisiana  18
## 2:      state  13
## 3: government  12
## 4:       plan  11
## 5:        new  11
```

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.

```
library(ggplot2)
cleanplot <- cleanWord[1:15]
p<-ggplot(data=cleanplot, aes(x=cleanWord, y=num)) +
  geom_bar(stat="identity") +coord_flip()
p
```



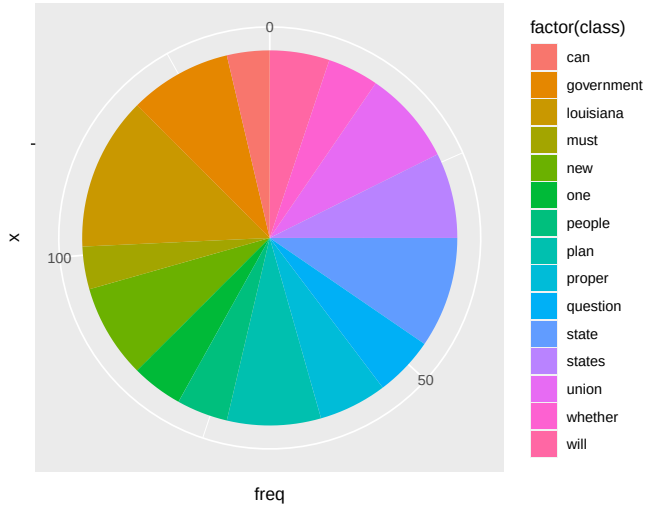
g) What is the reason for adding a layer `+coord_flip()` with the plot in question (2f). Explain what would happen if we would not have done that.

```
# we need the x axis on the left side of the graph so that we can read all the words as the labels.
#If the words are all on the bottom the text overlaps as there isn't space for them all.
```

h) The plot in question (2f) 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.

```
colnames(cleanplot) <- c("class", "freq")
pie <- ggplot(cleanplot, aes(x = "", y=freq, fill = factor(class))) +
  geom_bar(width = 1, stat = "identity") +
  theme(axis.line = element_blank(),
        plot.title = element_text(hjust=0.5))
```

```
pie + coord_polar(theta = "y", start=0)
```



i) In the question (2c), you removed words that are called `stop-words`. Now please answer the following:
   a) Count the total stop words from `lAddress` and store it in `stopWordsCount`
   b) Count the total words (including stop-words) from `lAddress` and store it in `lAddressCount`
   c) Divide `stopWordsCount` by `lAddressCount` and report the percentage
   d) Explain in your own words what does the percentage indicate in this context?

```
#a:
partf <- subset(partf, b==TRUE)
partf <- partf$a
stopWordsCount <- length(partf)
stopWordsCount
```

```
## [1] 982
```

```
#b:
lAddressCount <-lapply(vWord,length)
lAddressCount
```

```
## [[1]]
## [1] 1824
```

```
#c:
percentage <- as.numeric(stopWordsCount)/as.numeric(lAddressCount)
percentage
```

```
## [1] 0.5383772
```

```
#d: This shows that approximately 53.8% of the total words in the address are stop words.
# This percentage is also the amount of words we removed from the address.
```

2. Regular Expressions: 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 ‘:.

```
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 5 or 6 digit numbers with the word "found" in the following vector. Please make sure that 3, 4, or 7 digit numbers do not get changed.

```
vPhone <- c('874','6783','345345', '32120', '468349', '8149674' )
gsub('^\\d{5,6}$', 'found', vPhone)
```

```
## [1] "874"     "6783"    "found"   "found"   "found"   "8149674"
```

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"
myText <- gsub("[^[:alpha:] ]", "", myText)
myText
```

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

d) In the following text, replace all the words that are exactly 3 or 4 characters long with triple dots '...'

```
myText <- "Each of the three and four character words will be gone now"
myText <- strsplit(myText, " ")[[1]]
myText <- paste0(myText," ")
myText <- gsub("^\\w{3,4}\\s","... ",myText)
myText <- paste(myText, collapse="")
myText
```

```
## [1] "... of ... three ... ... character words ... be ... ... "
```

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

```
bigText <- 'There are four 20@14 numbers hid989den in the 500 texts'
bigText <- as.numeric(str_extract_all(bigText, "[0-9]{3}")[[1]])
bigText
```

```
## [1] 989 500
```

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

```
myText <- 'The salries are reported (in test millions) for every company.'
myText <- as.character(str_extract(myText, "[:punct:]+[^[:punct:]]+[:punct:]"))
max <- nchar(myText)
myText <- substr(myText,2,max-1)
myText
```

```
## [1] "in test millions"
```

```
wordsplit <- strsplit(myText," ")
numword <- lapply(wordsplit,length)
numword
```

```
## [[1]]
## [1] 3
```

g) Extract the texts in between _ and dot(.) in the following vector. Your output should be 'bill', 'pay', 'fine-book'.

```
myText <- c("H_bill.xls", "Big_H_pay.xls", "Use_case_fine-book.pdf")
myText <- as.character(str_extract(myText, "\\_+[^\\_]+\\."))
max <- nchar(myText)
myText <- substr(myText,2,max-1)
myText
```

```
## [1] "bill"      "pay"       "fine-book"
```

h) Extract the numbers (return only integers) that are followed by the units 'ml' or 'lb' in the following text.

```
myText <- 'Received 10 apples with 200ml water at 8pm with 15 lb meat and 2lb salt'
myText <- gsub(" ", "", myText)
myText <- gsub("lb", "lb ", myText)
myText <- gsub("ml", "ml ", myText)
myText <- str_extract_all(myText, "[0-9]+(lb|ml)")
myText
```

```
## [[1]]
## [1] "200ml" "15lb"  "2lb"
```

```
numwords <- length(myText)
numwords
```

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

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

```
myText <- 'Math symbols are $written$ in $between$ dollar $signs$'
myText <- str_extract_all(myText, "\\$+[^\\$]+\\$")
myText
```

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

```
lengths(myText)
```

```
## [1] 3
```

j) Extract all the valid equations in the following text.

```
myText <- 'equation1: 2+3=5, equation2 is: 2*3=6, do not extract 2w3=6'
myText <- gsub(",", "", myText)
myText <- str_extract_all(myText, "[0-9]{1}\\W{1}[0-9]{1}\\W{1}[0-9]{1}")
myText
```

```
## [[1]]
## [1] "2+3=5" "2*3=6"
```

k. Extract all the letters of the following sentence and check if it contains all 26 letters in the alphabet. If not, produce code that will return the total number of unique letters that are included and show the letters that are missing.

```
myText <- 'there are five wizard boxing matches to be judged'
myText <- gsub(" ", "", myText)
myText <- str_extract_all(myText,".")
myText <- lapply(myText,unique)
```

```
myText <- lapply(myText,sort)
myText
```

```
## [[1]]
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "m" "n" "o" "r" "s" "t" "u" "v" "w"
## [20] "x" "z"
```

```
lapply(myText, length)
```

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

```
#this gives the number of letters
test <- str_detect(myText,letters)
```

```
## Warning in stri_detect_regex(string, pattern, negate = negate, opts_regex =
## opts(pattern)): argument is not an atomic vector; coercing
```

```
df <- data.frame(letters)
df$match <- test
df <- data.table(df)
df[match == FALSE,]
```

```
##    letters match
## 1:       k FALSE
## 2:       l FALSE
## 3:       p FALSE
## 4:       q FALSE
## 5:       y FALSE
```

```
#these letters are missing
```

3. Extracting data from the 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) Read all the text in http://mamajumder.github.io/index.html and store your texts in `myText`. Show first few rows of `myText` and examine the structure of the data.

```
myText <- readLines('http://mamajumder.github.io/index.html')
```

```
## Warning in readLines("http://mamajumder.github.io/index.html"): incomplete final
## line found on 'http://mamajumder.github.io/index.html'
```

```
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.css\" rel=\
## [5] ""
## [6] "<title>Mahbubul Majumder</title>"
```

```
# The data looks to be an html script
```

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.

```
myText <- str_c(myText,collapse=', ')
myTextweb <- str_extract_all(myText, "h{1}t{1}t{1}p{1}[^\"]+")
myTextweb
```

```
## [[1]]
## [1] "http://maxcdn.bootstrapcdn.com/font-awesome/4.2.0/css/font-awesome.min.css"
## [2] "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.

```
myTextemail <- str_extract_all(myText, "m{1}a{1}i{1}l{1}t{1}o{1}[^\"]+")
myTextemail <- gsub("^.......","",myTextemail)
myTextemail
```

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

d. Write a regular expression to extract words with 11 or more letters in the text. Include your codes and display the result that shows the words without duplication.

```
myTextwords <- str_extract_all(myText, "\\w{11,}")
myTextwords <- lapply(myTextwords,unique)
myTextwords
```

```
## [[1]]
##  [1] "bootstrapcdn"      "Mathematics"       "publication"
##  [4] "Publications"      "experiments"       "Experiments"
##  [7] "datavisualization" "visualization"     "webresources"
## [10] "specialization"    "statistical"       "opportunity"
## [13] "Exploratory"       "quantification"    "Statistical"
```

e) 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.

```
myTestphones <- str_extract_all(myText,"\\({1}\\d{3}\\){1}\\s{1}\\d{3}\\D{1}\\d{4}")
myTestphones
```

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

f) The link of ggplot2 documentation is https://ggplot2-book.org/individual-geoms.html and we would like to get the list of individual 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 page list?

```
myggtext <- readLines('https://ggplot2-book.org/individual-geoms.html')
myggtext <- str_c(myggtext,collapse=', ')
myggtext <- str_extract_all(myggtext,"g{1}e{1}o{1}m{1}\\_{1}\\w+")
myggtext <- lapply(myggtext,unique)
myggtext
```

```
## [[1]]
##  [1] "geom_ribbon"  "geom_area"    "geom_bar"     "geom_path"    "geom_line"
##  [6] "geom_point"   "geom_polygon" "geom_tile"    "geom_rect"    "geom_raster"
## [11] "geom_text"    "geom_smooth"  "geom_boxplot" "geom_violin"
```

```
lapply(myggtext,length)
```

```
## [[1]]
## [1] 14
```

4. Big data problem: Download the sample of big data from canvas. Note that the data is in csv format and compressed for easy handling. You may need to uncompress it before using. Now answer the following questions.

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.

```
fulldat <- fread("bigDataSample.csv")
dat <- select(fulldat, 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.

```
dat <- setnames(dat, gsub("^[^_]*_[^_]*_[^_]*_","", names(dat) ))
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.

```
datmean <- group_by(dat,b) %>% summarise_all("mean")
datmean
```

```
## # A tibble: 4 x 5
##       b     g     p     e     n
##   <int> <dbl> <dbl> <dbl> <dbl>
## 1     0  28.7  23.8 12.2   29.4
## 2     1  22.5  25.3 10.4   29.3
## 3     2  23.9  24.9  9.62  30.6
## 4     3  23.8  25.4 10.5   30.3
```

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

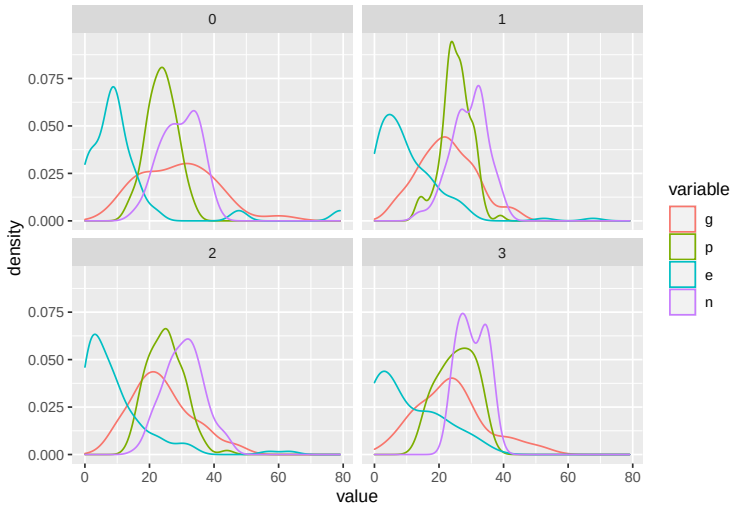
```
library(reshape)

mdat <- melt(dat, id="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.

```
p <- ggplot(mdat, aes(x=value, color=variable)) +
  geom_density() + facet_wrap(~b)
p
```



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 6a) without reading the whole data set first. Demonstrate that showing your codes.

```
# we don't want to have to load the whole data set and then delete the parts we don't need. It wastes t
cols <- colnames(fread("bigDataSample.csv",nrows=1))
fulldat <- fread("bigDataSample.csv", select=grep("h{1}u{1}m{1}a{1}n{1}",cols,value = TRUE))
head(fulldat)
```

```
##    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
```