Q. 1)

Write down a general regular expression to match the following:

(a) Words with any punctuation in them, e.g., h@te or v|c0din

(I suppose the single punctuation only appears once and it does not appear at the beginning or the end of a word. Then the regular expression can be the following one.)

[[:alnum:]]+[[:punct:]][[:alnum:]]+

(b) An IP address (Four sets of 1 to 3 digits separated by periods, e.g., 100.12.162.0)

[[:digit:]]{1,3}\\.[[:digit:]]{1,3}\\.[[:digit:]]{1,3}\\.[[:digit:]]{1,3}$

(c) An email address that ends with .com, .edu, .net, .org, or .gov

[[:alnum:]]+@[[:alnum:]]+\\.(com|edu|net|org|gov)$

Q. 2)

Carry out the following exercises on the State of the Union speeches database (available in moodle).

(a) Use readLines() to read in the speeches (available as a text file in moodle) where the return value is: character vector with one element/character string per line in the file

```
> con = file("C:/Documents/GSLIS/490 Introduction to Data Science/Data and
 Code/stateoftheunion1790-2012.txt", open = "r")
> allText = readLines(con = con)
> class(allText)
[1] "character"
> length(allText)
[1] 169641
```

(b) Use regular expressions to find ***

# Return the indices of lines.

# I leave out the last line in the text: *** END OF COMPLETE ADDRESSES ***

# Actually this line of text is not we are concerned about.

# If you want to keep the last line, then the code can be modified into grep("***",allText)

```
> asterisk.index = grep("^[*]{3}$",allText)
```

(c) Use *** to identify the date of the speech.

# Each date appears four lines after the "***".

```
> date.index = asterisk.index+4
> date = allText[date.index]
> head(date)
[1] "January 8, 1790"   "December 8, 1790"  "October 25, 1791"  "November
6, 1792"
[5] "December 3, 1793"  "November 19, 1794"
```

(d) Use regular expressions to extract the year.

# I did not convert the type of the vector from character into numeric.

# If numeric vector is needed, then you can add as.numeric() to the year vector.

```
> year.index = regexpr("[[:digit:]]{4}", date)
> year = substring(date, year.index, nchar(date))
> head(year)
[1] "1790" "1790" "1791" "1792" "1793" "1794"
```

(e) Use regular expressions to extract the month.

```
> month.index = regexpr(" [[:digit:]]{1,2}, ", date)
> month = substring(date, 1, month.index-1)
> head(month)
[1] "January"  "December" "October"  "November" "December" "November"
```

(f) Use *** to extract the name of the president State of the union speeches.

# Each name of president appears 3 lines after the "***".

```
> president.index = asterisk.index+3
> president = allText[president.index]
> head(president)
[1] "George Washington" "George Washington" "George Washington" "George Wa
shington"
[5] "George Washington" "George Washington"
```

(g) Use regular expressions and R to return the number of speeches in the dataset, and the number of presidents that gave speeches.

```
> numberOfSpeeches = length(grep("^[*]{3}$",allText))
> numberOfSpeeches
[1] 222
```

```
> numberOfPresidents = length(unique(president))
> numberOfPresidents
[1] 41
```

(h) Chop the speeches up into a list there is one element for each speech. Each element is a character vector. Check: does your number of list elements match your answer above?

# Since all the texts are separated by line, they are not a whole. So I did not use strsplit() function.

# There are 169641 lines in the text file. So 169641 elements in the allText vector.

# There are 222 elements of the asterisk.index vector. So there should be 222 speeches in all.

```
listOfSpeeches = list()

for (i in 1:222) {

  listOfSpeeches[[i]] = character()

}

asterisk.index[223] = 169641
```

```
for (j in 1:222) {
  for (i in (asterisk.index[j]+2):(asterisk.index[j+1]-2)) {
    listOfSpeeches[[j]] = append(listOfSpeeches[[j]], allText[i])
  }
}
> length(listOfSpeeches)
[1] 222
> class(listOfSpeeches[[1]])
[1] "character"
> head(listOfSpeeches[[2]])
[1] "State of the Union Address"
[2] "George Washington"
[3] "December 8, 1790"
[4] ""
[5] "Fellow-Citizens of the Senate and House of Representatives:"
[6]

# Each element in the list is a character vector of lines of text in that speech.
# I delete the *** from each speech text information.


# I am going to create a function to achieve from (i) to (m).
(i) Eliminate apostrophes, numbers, and the phrase: (Applause.)
(j) Make all the characters lower case.
(k) Split the sentences up where there are blanks and punctuation to create “words”.
(l) Drop any empty words that resulted from this split.
(m) Create a word vector for each speech.


install.packages("Rstem")
library("Rstem", lib.loc="C:/Program Files for operation/R-3.3.1/library")


speechToWords = function(lines){
  # lines is a character vector of lines for each speech
  # Use gsub to eliminate apostrophes and numbers
  s1=gsub("'","",lines)
  s2=gsub("[[:digit:]]+","",s1)

  # Drop the phrase (Applause.)
  # I think the exact phrase with parentheses should be eliminated. But the word applause inside
```

```
  # the sentences should not be removed.
  s3=gsub("\\(Applause\\.\\)","",s2)

  # Turn characters to lower case.
  s4=tolower(s3)

  # Use strsplit to split the text up by blanks and punctuation
  s5=strsplit(s4,"[[:blank:]]|[[:punct:]]")
  # Unlist the return value
  s6=unlist(s5)

  # Drop any empty words
  s7=s6[s6!=""]

  # Use wordStem() to stem the words
  # s8=wordStem(s7)
  # But the results are not correct such as words like januari and georg. So I give up using stem.

  # return a character vector of all words in the speech
  return(s7)
}

# Create a list of words. Each character vector of each speech consists of only one element.
# The element is a string of all the text of that speech.
listOfWords = list()
for (i in 1:222) {
  listOfWords[[i]] = character()
}
for (i in 1:222) {
  listOfWords[[i]] = paste(listOfSpeeches[[i]], sep=" ", collapse = " ")
}
```

# Create a word vector for each speech.

speechWords = lapply(listOfWords, speechToWords)

```
> length(speechWords)
[1] 222
> head(speechWords[[1]])
[1] "state"   "of"      "the"     "union"   "address" "george"
```

# For each speech, the vector contains all the words with duplicates in the speech text.

# I did not remove the duplicates here because the following tasks will deal with this problem.

(n) Normalize the word vectors to get term frequencies.

# Stemming was not correct when using wordStem(). So I give up it.

# Unlist the return value and use unique() to get the bag of words.

# Alphabetize the bag of words, and name it uniqueWords.

```
> uniqueWords= sort(unique(unlist(speechWords)))
> length(uniqueWords)
[1] 23102

> tempVector = rep(0, length(uniqueWords))
> names(tempVector) = uniqueWords

> wordVectors = lapply(speechWords, function(x){
+   tempVector[names(table(x))]=table(x)
+   return(tempVector)
+ })


> length(wordVectors[[1]])
[1] 23102
```

# Convert the list from the lapply function into a matrix

```
> wordMatrix = matrix(unlist(wordVectors), ncol = length(wordVectors))
> dim(wordMatrix)
[1] 23102   222
> rownames(wordMatrix) = uniqueWords
```

# So each column represents each speech. There are 23102 unique words in total.

# The value of each cell is the term frequency for that word in that speech.

(o) (5 points) Carry out some exploratory analysis of the data and term frequencies. For example, find the number of sentences, extract the long words, and the political party. Plot and interpret the term frequencies. What are your observations?

# I would like to pick Barack Obama and George Washington's speeches and compare

# how their frequent terms differ.

```
which(president=="George Washington")
# [1] 1 2 3 4 5 6 7 8

which(president=="Barack Obama")
# [1] 219 220 221 222

washingtonWords = wordMatrix[,1:8]
washingtonWords = apply(washingtonWords, 1, sum)
washingtonTopwords = matrix(nrow = 50, ncol = 2)
washingtonTopwords[,2] = head(sort(washingtonWords, decreasing = TRUE), 50)
washingtonTopwords[,1] = names(head(sort(washingtonWords, decreasing = TRUE), 50))
> washingtonTopwords
      [,1]         [,2]
 [1,] "the"        "1553"
 [2,] "of"         "1106"
 [3,] "to"         "669"
 [4,] "and"        "517"
 [5,] "in"         "292"
 [6,] "a"          "288"
 [7,] "be"         "243"
 [8,] "which"      "222"
 [9,] "that"       "197"
[10,] "with"       "176"
[11,] "for"        "169"
[12,] "have"       "167"
[13,] "it"         "157"
[14,] "by"         "152"
[15,] "our"        "152"
[16,] "will"       "146"
[17,] "been"       "135"
[18,] "as"         "128"
[19,] "is"         "124"
[20,] "not"        "110"
[21,] "i"          "107"
[22,] "their"      "99"
[23,] "states"     "94"
[24,] "on"         "93"
[25,] "united"     "86"
[26,] "an"         "82"
[27,] "from"       "80"
[28,] "has"        "80"
[29,] "may"        "69"
[30,] "you"        "69"
[31,] "this"       "68"
[32,] "are"        "60"
[33,] "them"       "58"
[34,] "public"     "54"
[35,] "at"         "51"
[36,] "can"        "51"
[37,] "government" "47"
[38,] "your"       "47"
[39,] "but"        "46"
[40,] "more"       "45"
[41,] "or"         "44"
```

```
[42,] "were"        "44"
[43,] "citizens"    "41"
[44,] "made"        "41"
[45,] "all"         "40"
[46,] "such"        "39"
[47,] "they"        "39"
[48,] "some"        "38"
[49,] "upon"        "38"
[50,] "my"          "37"
```

obamaWords = wordMatrix[,219:222]

obamaWords = apply(obamaWords, 1, sum)

obamaTopwords = matrix(nrow = 50, ncol = 2)

obamaTopwords[,2] = head(sort(obamaWords, decreasing = TRUE), 50)

obamaTopwords[,1] = names(head(sort(obamaWords, decreasing = TRUE), 50))

```
> obamaTopwords
      [,1]        [,2]
 [1,] "the"       "1258"
 [2,] "and"       "908"
 [3,] "to"        "893"
 [4,] "of"        "697"
 [5,] "a"         "572"
 [6,] "that"      "556"
 [7,] "we"        "488"
 [8,] "our"       "440"
 [9,] "in"        "437"
[10,] "for"       "289"
[11,] "will"      "260"
[12,] "i"         "250"
[13,] "is"        "243"
[14,] "this"      "241"
[15,] "it"        "207"
[16,] "on"        "185"
[17,] "are"       "170"
[18,] "have"      "170"
[19,] "but"       "161"
[20,] "with"      "156"
[21,] "more"      "150"
[22,] "not"       "133"
[23,] "as"        "130"
[24,] "be"        "130"
[25,] "their"     "128"
[26,] "who"       "123"
[27,] "from"      "122"
[28,] "or"        "121"
[29,] "they"      "119"
[30,] "you"       "119"
[31,] "can"       "116"
[32,] "now"       "113"
[33,] "people"    "105"
[34,] "by"        "104"
[35,] "new"       "104"
[36,] "all"       "98"
[37,] "american"  "97"
[38,] "do"        "95"
[39,] "its"       "95"
[40,] "jobs"      "95"
[41,] "so"        "95"
[42,] "has"       "90"
[43,] "thats"     "89"
[44,] "because"   "85"
[45,] "america"   "84"
```

```
[46,] "us"        "83"
[47,] "no"        "80"
[48,] "years"     "77"
[49,] "what"      "75"
[50,] "americans" "74"
```

# From the two matrices of Washington and Obama top words, we can see that

# apart from the stop words there are some differences between them.

# Washington spoke more words like "united", "sates", "public", "government" and "citizens".

# But Obama spoke more words like "american", "america", "americans", "people", "jobs" and "years".

# We can find that the statement for the country related words has changed from "United States"

# to "American". Maybe "U.S." could not be reflected for this term frequency comparison.

# Also we can find during George Washington period, the public, government and citizens are

# usually been mentioned. But during Barack Obama period, things about people and jobs

# have more concerns.

# I am interested in the term frequency of "jobs" and "job" over the time.

which(rownames(wordMatrix)=="job")

# [1] 11416

which(rownames(wordMatrix)=="jobs")

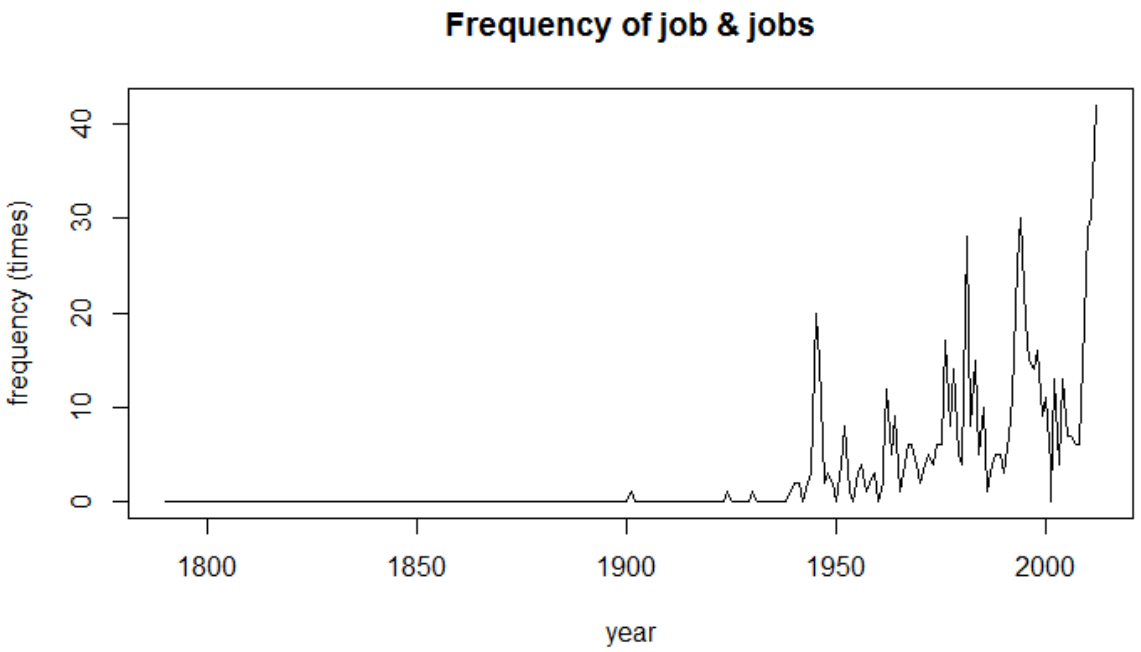
# [1] 11420

wordMatrix[11416,]

wordMatrix[11420,]

frequency.job = wordMatrix[11416,] + wordMatrix[11420,]

plot(year, frequency.job, type = "l", main = "Frequency of job & jobs", xlab = "year", ylab = "frequency (times)")




# From the graph we can see that the word "job" has never been mentioned in speeches before 1900.

# Right after 1900, "job" appeared in speeches of presidents. Then there was a peak around 1948.

# After that, there were fluctuations but generally the frequency was rising. Most recently,

# the word "job" was quite frequently used in the speech. The graph of the frequency generally

# reflects that job demands are increasing and the employment is getting more concerns.