M8L2 Homework Assignment

Joshua Conte

November 12, 2017

1 M8L2 Homework Assignment

R studio was configured with the following parameters before beginning the project:

```
# clears the console in RStudio
cat("\014")
```

```
# clears environment
rm(list = ls())

# Load required packages
library(RTextTools)
library(tm)
library(wordcloud)
library(SnowballC)
library(stringr)
```

1.1 Create regular expressions for the following:

1.1.1 Match any of the following punctuation characters in the ASCII table: ¡‘#$%&’()+

I would use [[:punct:]]. This will find all of the punctuation in a vector and will allow you to modify it, for example:

```
# String with punctuation
test<-"Match any of the following punctuation characters in the ASCII table: !\"#$%&'()+\'"

# This will print out the string with punctuation
grep(pattern="[[:punct:]]", test, perl=TRUE, value=TRUE)
```

```
## [1] "Match any of the following punctuation characters in the ASCII table: !\"#$%&'()+'"
```

```
# This will replace the punctuation with what you define
gsub(pattern="[[:punct:]]", test, replacement="")
```

```
## [1] "Match any of the following punctuation characters in the ASCII table "
```

1.1.2 Create one regular expression to match all common misspellings of calendar

(see https://en.wikipedia.org/wiki/Wikipedia:Lists_of_common_misspellings/C) One way to do this is to use brakets ([]) with an or statement (|) as shown below:

```
# Define misspellings
misspellings<-c("calandar", "calander", "calender")

# Find misspellings
grep("cal[a|e]nd[a|e]r", misspellings, perl=TRUE, value=TRUE)
```

```
## [1] "calandar" "calander" "calender"
```

1.1.3 Create one regular expression to match any character except line breaks.

I used \r and \n for page breaks in R.

```
# Vector with line breaks:
character<-c("calandar", "\n", "calander", "calender", "\n", "\t")

# This is a regular expression that will match anything but line breaks
grep("(?:[^\\r\\n]|\\r(?!\\n))",character, perl=TRUE, value=TRUE)
```

```
## [1] "calandar" "calander" "calender" "\t"
```

1.1.4 Validate a ZIP code

You need to validate a ZIP code (U.S. postal code), allowing both the five-digit and nine-digit (called ZIP+4) formats. The regex should match 02115 and 02115-5515, but not 2115, 2115-5515, 21155515,021155515, etc.

I use \d with {}, the {} let you define how many characters you need:

```
# Vecor with good and bad zip codes:
zip <- c("02115", "02115-5515", "2115", "2115-5515", "21155515", "021155515")

# Regular expression to find only valid zip codes
grep("^\\d{5}$|^\\d{5}-\\d{4}$",zip, perl=TRUE, value=TRUE)
```

```
## [1] "02115"      "02115-5515"
```

1.1.5 Password Validation

You need to validate a legit any password for your website. Passwords have the following complexity requirements: Length between 8 and 32 characters, ASCII visible and space characters only, One or more uppercase letters, One or more lowercase letters, One or more special characters (ASCII punctuation).

I started by defining what is needed first, lower case, then upper case, then punctuation. Then I have a string of what is acceptatble, any ascii charachter, with the minimum and maximum charachters allowed in {}. I used grepl to give me a TRUE or FALSE answer:

```
password <- "jC11111111111111111vvvvvv@"
grepl("^(?=.*[a-z])(?=.*[A-Z])(?=.*[$@$!%*?&])[a-zA-Z$@$!%*?&0-9]{8,32}$",password, perl=TRUE)
```

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

1.1.6 Load the file ML.Tweets.csv

(it is online at 'http://nikbearbrown.com/YouTube/MachineLearning/Twitter/')

```
# With a fast internet connection
# data_url <- 'http://nikbearbrown.com/YouTube/MachineLearning/Twitter/ML.Tweets.csv'
# twitter <- read.csv(url(data_url))

# Load file from directory
if (!exists("twitter")) {
twitter <-
  read.csv2("ML.Tweets.csv",
    sep = ",",
    stringsAsFactors = FALSE,
    header = FALSE,
    blank.lines.skip = TRUE,
    na.strings=c("","NA")
```

```
  )
}
```

Complete the following: ####Extract a list of the top 9 users (e.g. @NikBearBrown) I got all the usernames in all three columns using str_extract_al(). Then I changed them to vectors and put them together. Then I put them into a dataframe table that has the username and quantity. I sorted the dataframe and printed the first 9.

```
# First I use str_extract_all and put all users in a list for V1,V2, and V3
userList1<-str_extract_all(twitter$V1, "@\\S+")
userList2<-str_extract_all(twitter$V2, "@\\S+")
userList3<-str_extract_all(twitter$V3, "@\\S+")

# I  change the list to a vector
userTag1<-unlist(userList1, use.names=FALSE)
userTag2<-unlist(userList2, use.names=FALSE)
userTag3<-unlist(userList3, use.names=FALSE)

# I put all three vectors together
userTag <- c(userTag1, userTag2, userTag3)

# I put it into a dataframe
dfUser <-
  as.data.frame(table(userTag))
names(dfUser) <- c("User", "Quantity")

# This sorts the data from highest quantity to least.
dfUserSort <-
  dfUser[order(dfUser$Quantity, decreasing = TRUE),]

# I print out the first 9
head(dfUserSort,9)
```

```
##                    User Quantity
## 51258      @jose_garde:    19068
## 11698     @BigDataBlogs     7702
## 19435        @CloudExpo     6113
## 54193      @KirkDBorne:     5865
## 96732 @TungstenBigData:     5500
## 11703    @BigDataBlogs:     5325
## 94366       @ThingsExpo     4790
## 19134       @ClearGrip:     4760
## 81063  @Ronald_vanLoon:     4543
```

1.1.6.1 Extract a list of the top 9 hashtags (e.g. #Bear)

This is the same as above but I replaced @ with #.

```
# First I use str_extract_all and put all hashtags in a list for V1,V2, and V3
hashList1<-str_extract_all(twitter$V1, "#\\S+")
hashList2<-str_extract_all(twitter$V2, "#\\S+")
hashList3<-str_extract_all(twitter$V3, "#\\S+")

# I  change the list to a vector
hashTag1<-unlist(hashList1, use.names=FALSE)
```

```
hashTag2<-unlist(hashList2, use.names=FALSE)
hashTag3<-unlist(hashList3, use.names=FALSE)

# I put all three vectors together
hashTag <- c(hashTag1, hashTag2, hashTag3)

# I put it into a dataframe
dfHash <-
  as.data.frame(table(hashTag))
names(dfHash) <- c("Hashtag", "Quantity")

# This sorts the data from highest quantity to least.
dfHashSort <-
  dfHash[order(dfHash$Quantity, decreasing = TRUE),]

# I print out the first 9
head(dfHashSort,9)
```

```
##          Hashtag Quantity
## 8204    #BigData   331249
## 8200    #bigdata   247258
## 3471  #analytics    56168
## 37054       #IoT    55743
## 47653      #news    48319
## 3472  #Analytics    47560
## 44514      #MGWV    43108
## 27392  #followme    42998
## 25858       #F4F    42918
```

1.1.6.2 Find the top 5 most positve tweets and the top 5 most negative tweets

I wish I would have wrote this on my own, but I got help from: https://analyzecore.com/2014/04/28/twitter-sentiment-analysis/

I found the negative and positive word list (around 6800 words) from: http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

I used the score.sentiment function with the word lists after I loaded them. Then I put the result into a dataframe and sorted it, then used head and tail to get the top 5 and bottom 5.

```
score.sentiment = function(sentences, pos.words, neg.words, .progress='none')
{
  require(plyr)
  require(stringr)

  # we got a vector of sentences. plyr will handle a list or a vector as an "l" for us
  # we want a simple array of scores back, so we use "l" + "a" + "ply" = laply:
  scores = laply(sentences, function(sentence, pos.words, neg.words) {

    # clean up sentences with R's regex-driven global substitute, gsub():
    sentence = gsub('[[:punct:]]', '', sentence)
    sentence = gsub('[[:cntrl:]]', '', sentence)
    sentence = gsub('\\d+', '', sentence)
    # and convert to lower case:
    sentence = tolower(sentence)
```

```
    # split into words. str_split is in the stringr package
    word.list = str_split(sentence, '\\s+')
    # sometimes a list() is one level of hierarchy too much
    words = unlist(word.list)

    # compare our words to the dictionaries of positive & negative terms
    pos.matches = match(words, pos.words)
    neg.matches = match(words, neg.words)

    # match() returns the position of the matched term or NA
    # we just want a TRUE/FALSE:
    pos.matches = !is.na(pos.matches)
    neg.matches = !is.na(neg.matches)

    # and conveniently enough, TRUE/FALSE will be treated as 1/0 by sum():
    score = sum(pos.matches) - sum(neg.matches)

    return(score)
  }, pos.words, neg.words, .progress=.progress )

  scores.df = data.frame(score=scores, text=sentences)
  return(scores.df)
}
# This imports the positive word list without the headers
posWords<-read.table("positive-words.txt", skip = 35,stringsAsFactors = FALSE, header = FALSE)

# This puts it into vector form
posWords<-unlist(posWords)


# This imports the negative word list without the headers
negWords<-read.table("negative-words.txt", skip = 35,stringsAsFactors = FALSE, header = FALSE)

# This puts it into vector form
negWords<-unlist(negWords)


# This scores all of the tweets
dfTweet<-score.sentiment(twitter$V3, posWords, negWords, .progress='none')
```

```
## Loading required package: plyr
```

```
# This sorts the list
dfTweetSort <-
  dfTweet[order(dfTweet$score, decreasing = TRUE),]

# This is the top 5
head(dfTweetSort, 5)
```

```
##        score
## 669352     8
## 163999     6
## 223294     6
## 559418     6
```

```
## 561669     6
##
## 669352 #BigData is good. #BigData is good #BigData is good #BigData is good #BigData is good #BigData
## 163999  SQLite Free - Datum 6.4.1 - Fast modern database viewer. (Free): Datum is a fast modern datab
## 223294                                                    PLUS PLUS #goodness, #kindness, #GRACE, #humil
## 559418     #Bigdata leads to smarter #tech. Smarter tech leads to more transparency, which equals mor
## 561669     #Bigdata leads to smarter #tech. Smarter tech leads to more transparency, which equals mor
```

```
# This is the last 5
tail(dfTweetSort, 5)
```

```
##         score
## 774333     -5
## 803737     -5
## 961928     -5
## 1015370    -5
## 770303     -6
##
## 774333      RT @guidonld: Sterling: "Anybody who lacks a cloud for big data is in desperate peril" @b
## 803737  Experienced #BigData and #DataScience resources are not cheap. If you want cheap, prepare to
## 961928            Drones are the anonymous murder machines of the hegemon struggling to stave off its
## 1015370 #bigdata #iot I am tired of the #cloud THERE IS NO CLOUD. There is a lot of hardware and cabl
## 770303                                      Before, it was, "Lies, damned lies, and #Statistics" now
```

1.1.6.3 Create a world cloud of 100 related tweets

I modified the score function to pnly use the list of words, this way it will match anything from a list. So the output will only be good as the list provided. I used a christmas word list that matches the most common christmas tweets. I take the top 100 and use it for the word cloud.

```
score.single.sentiment = function(sentences, pos.words, neg.words, .progress='none')
{
  require(plyr)
  require(stringr)

  # we got a vector of sentences. plyr will handle a list or a vector as an "l" for
  # us we want a simple array of scores back, so we use "l" + "a" + "ply" = laply:
  scores = laply(sentences, function(sentence, pos.words) {

    # clean up sentences with R's regex-driven global substitute, gsub():
    sentence = gsub('[[:punct:]]', '', sentence)
    sentence = gsub('[[:cntrl:]]', '', sentence)
    sentence = gsub('\\d+', '', sentence)
    # and convert to lower case:
    sentence = tolower(sentence)

    # split into words. str_split is in the stringr package
    word.list = str_split(sentence, '\\s+')
    # sometimes a list() is one level of hierarchy too much
    words = unlist(word.list)

    # compare our words to the dictionaries of positive & negative terms
    pos.matches = match(words, pos.words)
    #neg.matches = match(words, neg.words)
```

```
    # match() returns the position of the matched term or NA
    # we just want a TRUE/FALSE:
    pos.matches = !is.na(pos.matches)
    #neg.matches = !is.na(neg.matches)

    # and conveniently enough, TRUE/FALSE will be treated as 1/0 by sum():
    score = sum(pos.matches)

    return(score)
  }, pos.words, .progress=.progress )

  scores.df = data.frame(score=scores, text=sentences)
  return(scores.df)
}

# This is a list I found on wikipedia relating to video games
christmasWords<-read.table("christmas.txt",header = FALSE, sep="\n")
```

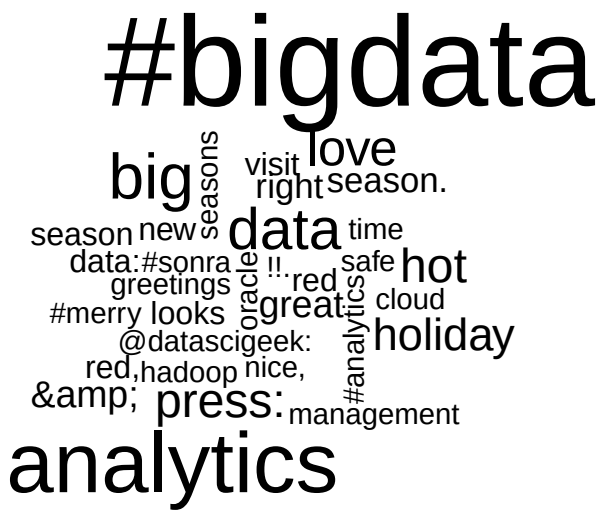
```
## Warning in scan(file = file, what = what, sep = sep, quote = quote, dec =
## dec, : EOF within quoted string
```

```
# This puts it into vector form
christmasWords<-unlist(as.character(christmasWords$V1))

# This removes new lines
christmasWords<-gsub("[\n]", " ", christmasWords)

# This splits up the words
christmasWords<-strsplit(christmasWords, " ")

# This puts it back into a vecote
christmasWords<-unlist(christmasWords)

# This scores all of the tweets
dfchristmas<-score.single.sentiment(twitter$V3, christmasWords)

# This sorts the list
dfchristmasSort <-
  dfchristmas[order(dfchristmas$score, decreasing = TRUE),]

dfchristmasTweets<-dfchristmasSort[1:100,]
BestTweets<-unlist(as.character(dfchristmasTweets$text))


tweets.corpus <- Corpus(DataframeSource(data.frame(BestTweets)))

# Eliminating Extra Whitespace
tweets.clean<-tm_map(tweets.corpus, stripWhitespace)

# stemDocument
tweets.clean.stem<-tm_map(tweets.clean, stemDocument)

# Convert to Lower Case
tweets.clean.lc <- tm_map(tweets.clean, content_transformer(tolower))
```

```
# Remove Stopwords
tweets.clean <- tm_map(tweets.clean.lc, removeWords, stopwords("english"))

# Building a Document-Term Matrix
tweets.tdm <- TermDocumentMatrix(tweets.clean, control = list(minWordLength = 1))
tweets.tdm
```

```
## <<TermDocumentMatrix (terms: 696, documents: 100)>>
## Non-/sparse entries: 1127/68473
## Sparsity           : 98%
## Maximal term length: 23
## Weighting          : term frequency (tf)
```

```
# Word Cloud
m <- as.matrix(tweets.tdm)
# calculate the frequency of words
v <- sort(rowSums(m), decreasing=TRUE)
myNames <- names(v)
d <- data.frame(word=myNames, freq=v)
wordcloud(d$word, d$freq, min.freq=5)
```



1.1.6.4 Which tweets could be classified as game development?

In this case I used a list of video game terms from wikipedia and used the function above.

```
# This is a list I found on wikipedia relating to video games
gameWords<-read.table("game.txt",header = FALSE, sep="\n")
```

```
## Warning in scan(file = file, what = what, sep = sep, quote = quote, dec =
## dec, : EOF within quoted string
```

```
# This puts it into vector form
gameWords<-unlist(as.character(gameWords$V1))

# This removes new lines
gameWords<-gsub("[\n]", " ", gameWords)

# This splits up the words
gameWords<-strsplit(gameWords, " ")

# This puts it back into a vecote
gameWords<-unlist(gameWords)

# This scores all of the tweets
dfGame<-score.single.sentiment(twitter$V3, gameWords)

# This sorts the list
dfGameSort <-
  dfGame[order(dfGame$score, decreasing = TRUE),]

# This is the top 10
head(dfGameSort, 10)
```

```
##        score
## 93880      8
## 187754     8
## 188562     8
## 196413     8
## 196587     8
## 196588     8
## 196590     8
## 196645     8
## 196653     8
## 196742     8
##
## 93880           Finding the time to look closer at Primo Analytics. Will take a while to get used to
## 187754 @mcuban Getting paid millions to play a child's game, take the time to learn how to make an u
## 188562  @MarinersHockey lose to @Valley_Wildcats 8-4 in Game 2. Mariners need to regroup and get som
## 196413        The Power of Data to Know the World, to Improve the World, and to Change the World, w
## 196587                            The Power of Data to Know the World, to Improve the World, and to C
## 196588                            The Power of Data to Know the World, to Improve the World, and to C
## 196590                            The Power of Data to Know the World, to Improve the World, and to C
## 196645          RT @BigData_Fr: The Power of Data to Know the World, to Improve the World, and to C
## 196653          RT @BigData_Fr: The Power of Data to Know the World, to Improve the World, and to C
## 196742          RT @BigData_Fr: The Power of Data to Know the World, to Improve the World, and to C
```