M7L3 Homework Assignment

Joshua Conte

November 12, 2017

1 M7L3 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)
library(ggplot2)
library(cluster)
library(amap)
library(useful)
library(qdap)
```

1.1 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:

1.1.1 Extract and rank a list of the important hashtags (using td-idf or word entropy).

This was a difficult assignment, the algorithms I used could not handle a lot of data. To get around this I had to use the first 50,000 lines of the CSV file, otherwise R would timeout and give me a memory error. I am not sure if it is the program or my computer, but I do have 16 GB of RAM.

Below is my code:

```
# get the first 500,000 lines of the file
hashLines<-twitter$V3[1:50000]
hashLines<-hashLines[!is.na(hashLines)]
```

```
# First I use str_extract_all and put all hashtags in a list for V1,V2, and V3
hashList<-str_extract_all(hashLines, "#\\S+")

# I  change the list to a vector
hashTag<-unlist(hashList, use.names=FALSE)

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

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

# 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: 7272, documents: 70977)>>
## Non-/sparse entries: 70453/516074291
## Sparsity           : 100%
## Maximal term length: 76
## Weighting          : term frequency (tf)
```

```
# inspect most popular words
findFreqTerms(tweets.tdm, lowfreq=80)
```

```
##  [1] "#ai"                   "#analytics"
##  [3] "#analytics,"           "#analytics."
##  [5] "#analytics:"           "#analytics?"
##  [7] "#bi"                   "#bigdata"
##  [9] "#bigdata,"             "#bluemix"
## [11] "#blurreal"             "#business"
## [13] "#businessintelligence" "#clickcounter"
## [15] "#cloud"                "#content"
## [17] "#crowdfunding"         "#data"
## [19] "#data15"               "#dataanalytics"
## [21] "#datamining"           "#datascience"
## [23] "#datastories"          "#dataviz"
## [25] "#deeplearning"         "#digital"
## [27] "#digitalmarketing"     "#entrepreneurs:"
## [29] "#etl"                  "#finance"
## [31] "#fintech"              "#getanalyticsdone"
## [33] "#google"               "#googleanalytics"
## [35] "#groovy"               "#hadoop"
## [37] "#healthcare"           "#hiring"
## [39] "#hr"                   "#hrtechconf"
## [41] "#ibm"                  "#ibminsight"
## [43] "#ibmz"                 "#indiegogo"
## [45] "#infographic"          "#innovation"
## [47] "#intell"               "#intentionalliving"
```

```
## [49] "#iot"                  "#java"
## [51] "#job"                  "#job:"
## [53] "#jobs"                 "#kdn"
## [55] "#kickstarter"          "#leadership"
## [57] "#leadfromwithin"       "#lovethydata"
## [59] "#machinelearning"      "#management"
## [61] "#marketing"            "#mobile"
## [63] "#news"                 "#newwaytowork"
## [65] "#oow15"                "#prediction"
## [67] "#predictive"           "#predictiveanalytics"
## [69] "#pushawardskathniels"  "#pushawardslizquens"
## [71] "#python"               "#resilience"
## [73] "#retail"               "#ronr"
## [75] "#rstats"               "#rt"
## [77] "#saas"                 "#sap"
## [79] "#saptd"                "#senior"
## [81] "#seo"                  "#socialmedia"
## [83] "#spark"                "#sqlserver"
## [85] "#startup"              "#statistics"
## [87] "#stocks"               "#strategy"
## [89] "#tdpartners15"         "#tech"
## [91] "#technology"           "#twitter"
## [93] "#ux"                   "#web"
## [95] "#website"
```

1.1.2 Cluster the tweets using these hashtags.

This works with the tdm file created above:

```
# convert the sparse term-document matrix to a standard data frame
dfTweets <- as.data.frame(inspect(tweets.tdm))
```

```
## <<TermDocumentMatrix (terms: 7272, documents: 70977)>>
## Non-/sparse entries: 70453/516074291
## Sparsity           : 100%
## Maximal term length: 76
## Weighting          : term frequency (tf)
## Sample             :
##                Docs
## Terms           1 10 2 3 4 5 6 7 8 9
##   #analytics    0  1 0 0 0 0 0 0 0 0
##   #bigdata      0  0 0 0 0 0 0 1 0 0
##   #data         0  0 0 0 0 0 0 0 0 0
##   #datascience  0  0 0 0 0 0 0 0 0 0
##   #dataviz      0  0 0 0 0 0 0 0 0 0
##   #ibminsight   0  0 0 0 0 0 0 0 0 0
##   #iot          0  0 0 0 0 0 0 0 0 0
##   #job          0  0 0 0 0 0 0 0 0 0
##   #jobs         0  0 0 0 0 0 0 0 0 0
##   #marketing    0  0 0 0 0 0 0 0 0 0
```

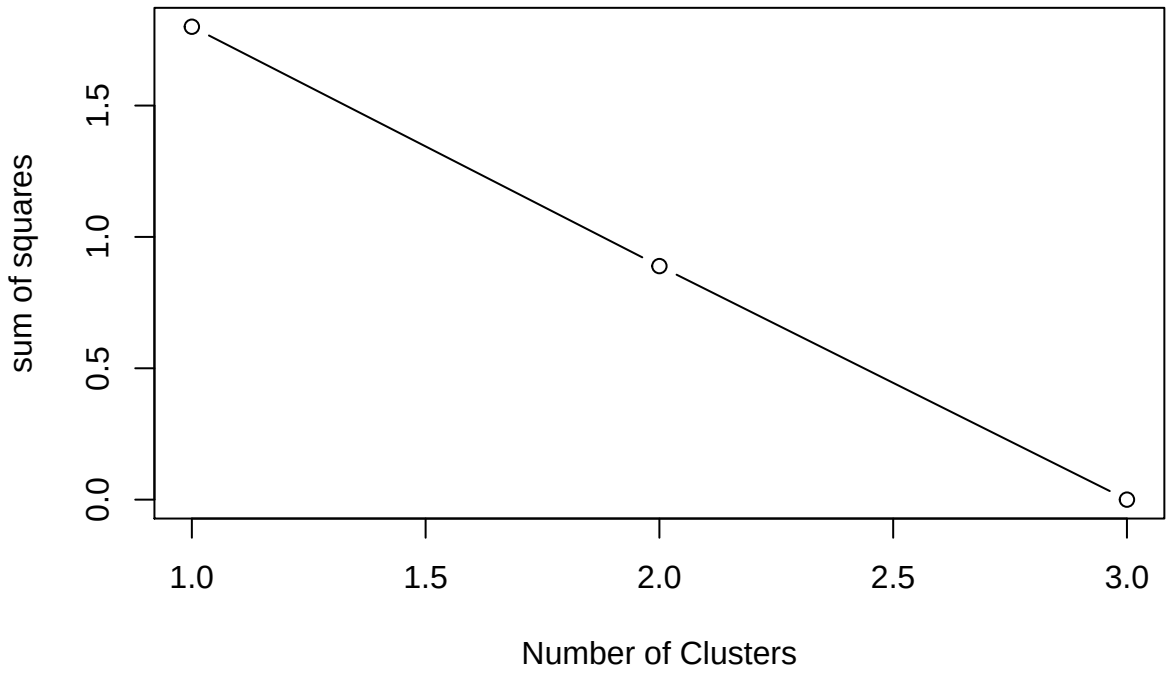
```
# inspect dimensions of the data frame
nrow(dfTweets)
```

```
## [1] 10
```

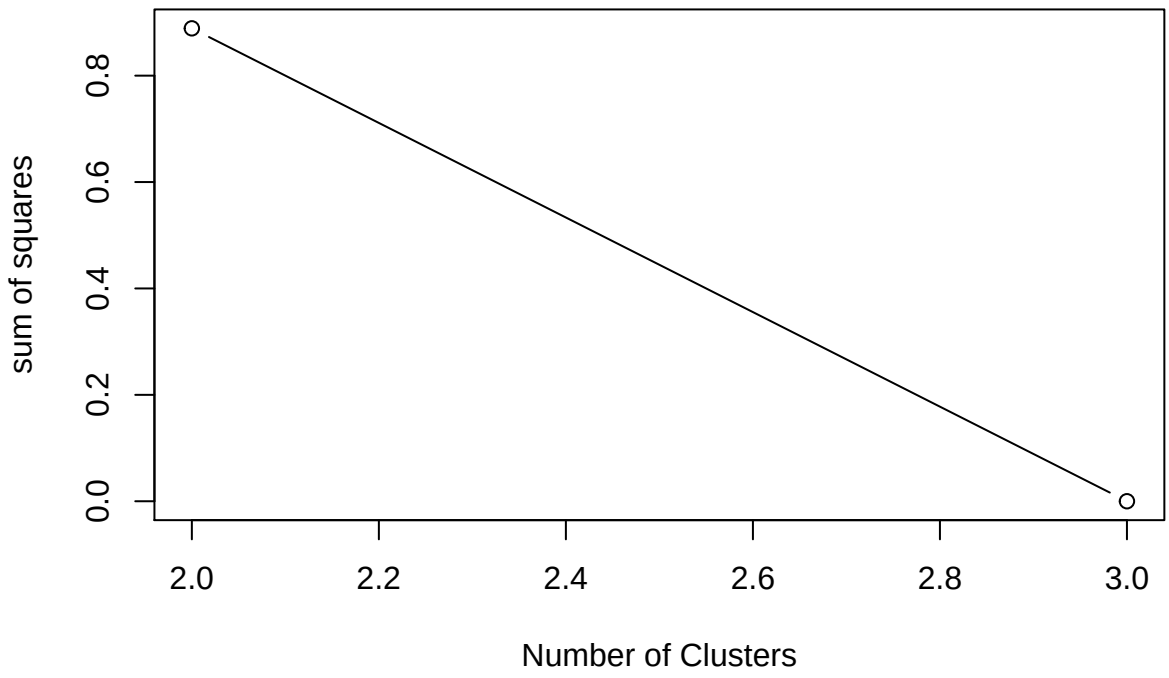
```r
ncol(dfTweets)
```

```
## [1] 10
```

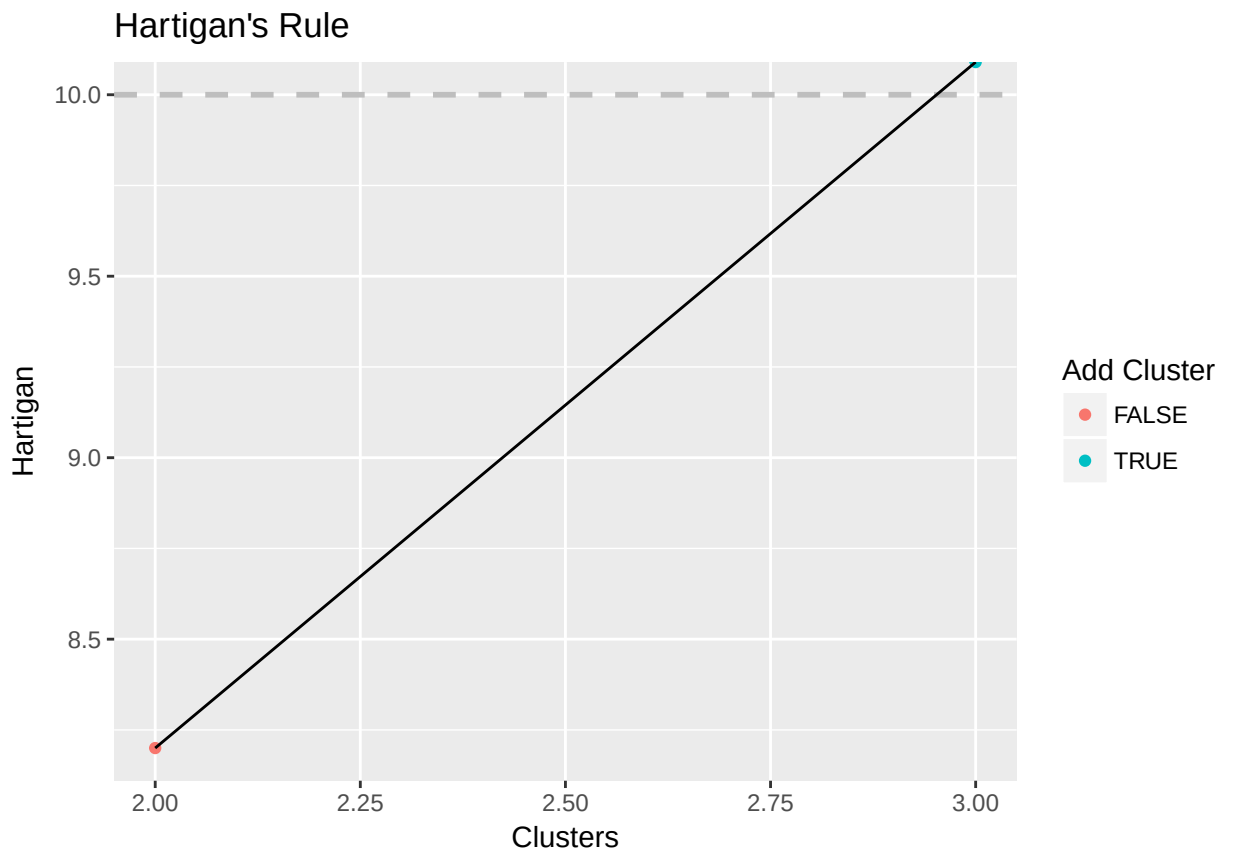
```r
# Determining number of clusters
sos <- (nrow(dfTweets) - 1) * sum(apply(dfTweets, 2, var))
for (i in 2:3)
  sos[i] <- sum(kmeans(dfTweets, centers = i)$withinss)
plot(1:3,
     sos,
     type = "b",
     xlab = "Number of Clusters",
     ylab = "sum of squares")
```




```r
plot(2:3,
     sos[c(2:3)],
     type = "b",
     xlab = "Number of Clusters",
     ylab = "sum of squares")
```



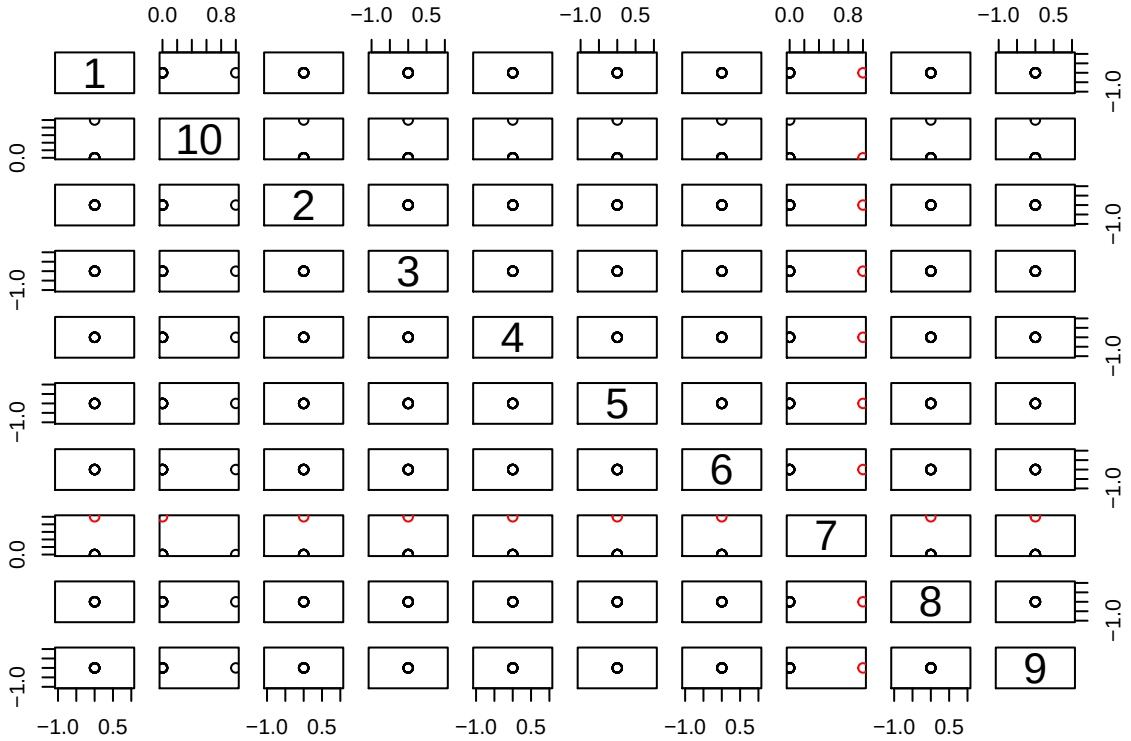

```
# Hartigans's rule FitKMean (similarity)
# require(useful)
best<-FitKMeans(dfTweets,max.clusters=3, seed=111)
PlotHartigan(best)
```




```
k <- 2
trails<-1000
dfTweets.2.cluster <- kmeans(dfTweets,k, nstart = trails)
dfTweets.2.cluster
```

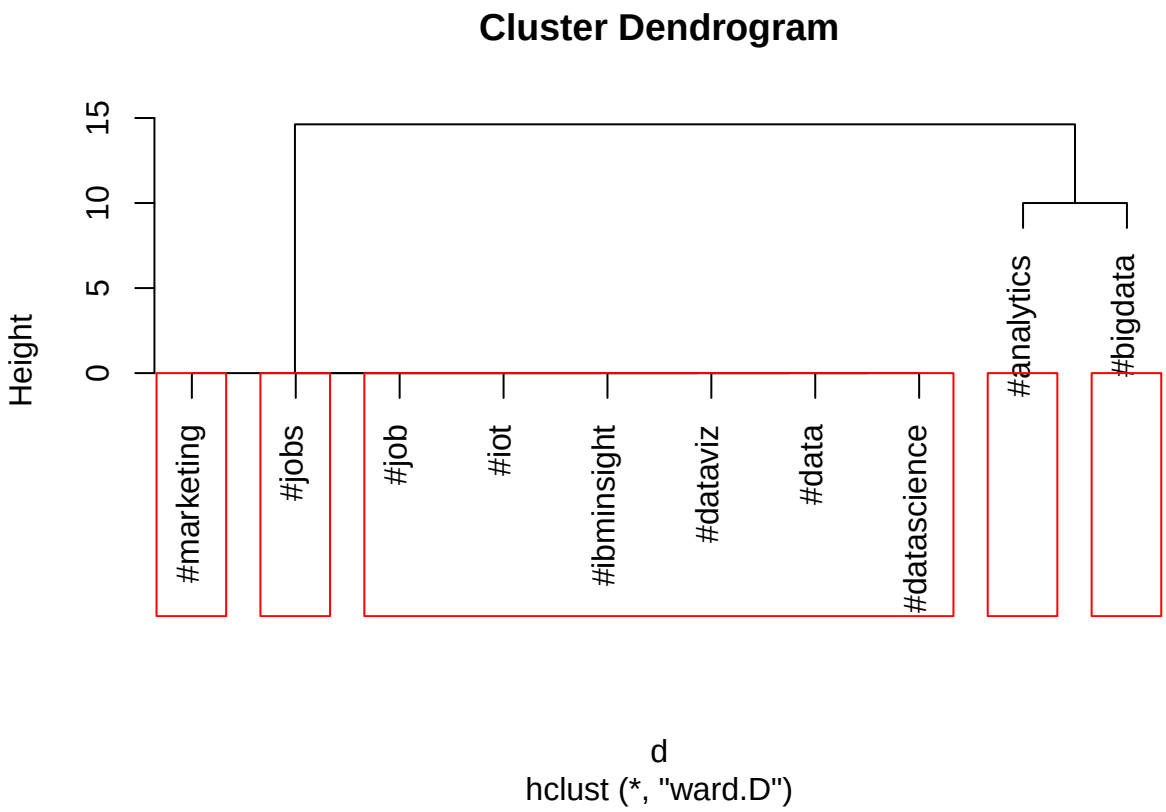
```
## K-means clustering with 2 clusters of sizes 9, 1
##
## Cluster means:
##   1        10 2 3 4 5 6 7 8 9
## 1 0 0.1111111 0 0 0 0 0 0 0 0
## 2 0 0.0000000 0 0 0 0 0 1 0 0
##
## Clustering vector:
##   #analytics     #bigdata        #data #datascience     #dataviz
##            1            2            1            1            1
##  #ibminsight         #iot         #job        #jobs   #marketing
##            1            1            1            1            1
##
## Within cluster sum of squares by cluster:
## [1] 0.8888889 0.0000000
##  (between_SS / total_SS =  50.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
plot(dfTweets,col=dfTweets.2.cluster$cluster)      # Plot Clusters
```




```
dfTweets.scale <- scale(dfTweets)
d <- dist(dfTweets.scale, method = "euclidean") # distance matrix
fit <- hclust(d, method="ward.D")
plot(fit) # display dendogram?

groups <- cutree(fit, k=5) # cut tree into 5 clusters
# draw dendogram with red borders around the 5 clusters
rect.hclust(fit, k=5, border="red")
```



1.1.3 Use the qdap polarity function to score the polarity of the tweets in ML.Tweets.csv.

This is from an example I found from: http://michaellevy.name/blog/conference-twitter/

This takes the polarity of the tweets after a little reformating:

```
pol =
    lapply(hashLines, function(txt) {
        # strip sentence enders so each tweet is analyzed as a sentence,
        # and +'s which muck up regex
        gsub('(\\.|!|\\?)\\s+|(\\++)', ' ', txt) %>%
            # strip URLs
            gsub(' http[^[:blank:]]+', '', .) %>%
            # calculate polarity
            polarity()
    })

head(pol)
```

```
## [[1]]
##   all total.sentences total.words ave.polarity sd.polarity stan.mean.polarity
## 1 all               1          19        0.229          NA                 NA
##
## [[2]]
##   all total.sentences total.words ave.polarity sd.polarity stan.mean.polarity
## 1 all               1          10            0          NA                 NA
```

```
##
## [[3]]
##   all total.sentences total.words ave.polarity sd.polarity stan.mean.polarity
## 1 all               1          14       -0.267          NA                 NA
##
## [[4]]
##   all total.sentences total.words ave.polarity sd.polarity stan.mean.polarity
## 1 all               1          10            0          NA                 NA
##
## [[5]]
##   all total.sentences total.words ave.polarity sd.polarity stan.mean.polarity
## 1 all               1          12            0          NA                 NA
##
## [[6]]
##   all total.sentences total.words ave.polarity sd.polarity stan.mean.polarity
## 1 all               1           3            0          NA                 NA
```

1.1.4 Would creating a custom polarity.frame - A dataframe or environment containing a dataframe of positive/negative words and weights - based on the tags and words in these tweets improve the polarity score? Try it.

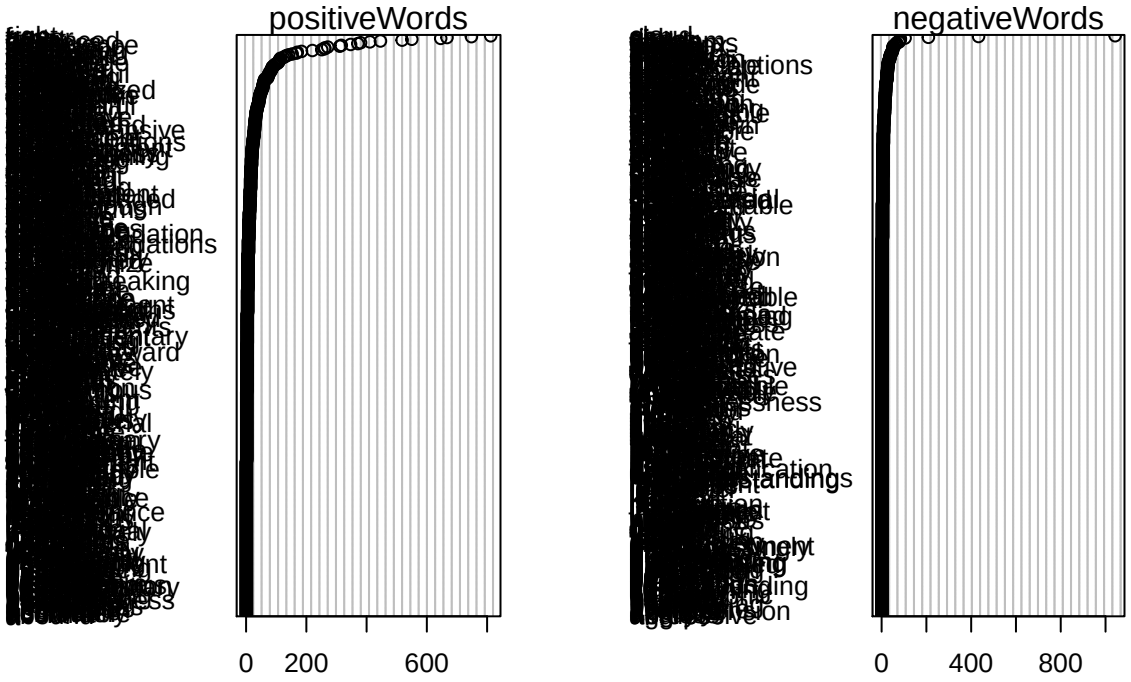
This is using qdap's polarity function straight out of the box to examine the emotional valence of each tweet. The frequency of usage in the conference tweets of words that qdap identifies as positively or negatively valenced.

```
polWordTables =
  sapply(pol, function(p) {
    words = c(positiveWords = paste(p[[1]]$pos.words[[1]], collapse = ' '),
              negativeWords = paste(p[[1]]$neg.words[[1]], collapse = ' '))
    gsub('-', '', words)  # Get rid of nothing found's "-"
  }) %>%
  apply(1, paste, collapse = ' ') %>%
  stripWhitespace() %>%
  strsplit(' ') %>%
  sapply(table)

par(mfrow = c(1, 2))
invisible(
  lapply(1:2, function(i) {
    dotchart(sort(polWordTables[[i]]), cex = .8)
    mtext(names(polWordTables)[i])
  }))
```

```
## Warning in dotchart(sort(polWordTables[[i]]), cex = 0.8): 'x' is neither a
## vector nor a matrix: using as.numeric(x)

## Warning in dotchart(sort(polWordTables[[i]]), cex = 0.8): 'x' is neither a
## vector nor a matrix: using as.numeric(x)
```


positiveWords
0
200
600
negativeWords
0
400
800