SWA Group Assignment

Group 27

9/10/2021

Declaration

Student Name | Student Number | Contribution (%)

Noah Grant | 19347992 | 0.20

Daniel Longo | 19288651 | 0.20

Bradley Roberts | 19971416 | 0.20

Raymond Sujono | 19805304 | 0.20

Brennan Orlando Zuniga Terrey | 20029786 | 0.20

By including this statement, we the authors of this work, verify that:

We hold a copy of this assignment that we can produce if the original is lost or damaged.

We hereby certify that no part of this assignment/product has been copied from any other student's work or from any other source except where due acknowledgement is made in the assignment.

No part of this assignment/product has been written/produced for us by another person except where such collaboration has been authorized by the subject lecturer/tutor concerned.

We are aware that this work may be reproduced and submitted to plagiarism detection software programs to detect possible plagiarism (which may retain a copy on its database for future plagiarism checking).

We hereby certify that we have read and understood what the School of Computing and Mathematics defines as minor and substantial breaches of misconduct as outlined in the learning guide for this unit.

Question 8.1 - Analyzing tweets

```
#libraries
library(rtweet)
library(tm)

library(wordcloud)

library(dendextend)

library(igraph)

library(twitteR)

library(stopwords)

#user authorization details (8.1,8.2,8.3)
app="DataAquisition"
key="dmvXsCWU9judhx5rBou45ShRX"
secret="DUvg045HPAS5NJtr87Owyf3ct4zd5Zdlr8MshnFSCEqWefHUyG"
access_token="1416912216536719362-bb3rzIvvWneIpOI92Q6gX9kmdYGTCk"
access_secret="af2rof3FEGf9DWAlqk8nIUqZmPnMnFt8gMc6zhlhoaZBf"
twitter_token = create_token(app,key,secret,access_token,access_secret,set_re
nv = FALSE)

#authentication for twitter (8.4)
setup_twitter_oauth(key,secret,access_token, access_secret)

#function that generates the tweet source matrix
makeMatrix = function(tweets,random, names)
{

  #Get the raw data from the Source
  data1 = table(tweets$source)
  data2 = table(random$source)

  #make the matrix
  tweet.source.table = matrix(data = rep(0,length(names)) ,nrow = 2,ncol = le
ngth(names), byrow = FALSE)
  row.names(tweet.source.table) = c("tweets","random tweets")
  colnames(tweet.source.table) = names

  #get largest data set
  if(length(data1) > length(data2))
  {
    largest = length(data1)
  } else {
    largest = length(data2)
  }
```

```
  #fill the matrix with data
  for (i in 1:length(names))
  {
      for (j in 1:largest)
      {
          #only allow if not NA and column names match
          if(!is.na(data1[j]))
          {
            if(row.names(data1)[j] == names[i])
            {
              #put data in row 1
              tweet.source.table[1,i] = as.numeric(data1[j])
            }
          }

          if(!is.na(data2[j]))
          {
            if(row.names(data2)[j] == names[i])
            {
              #Put data in row 2
              tweet.source.table[2,i] = as.numeric(data2[j])
            }
          }
      }
  }
  return(tweet.source.table)
}

#Question 1 - get 1000 tweets about Danny Devito
#check if the data is saved, otherwise this means this is first time the code
is running OR that the data is missing.
if(file.exists("tweets.rds")){
  tweets = readRDS("tweets.rds")
}else{
  user = "Danny Devito"
  tweets = search_tweets(user, n = 1000, type = "recent", token = twitter_tok
en, include_rts = FALSE, lang="en")

  #save tweets
  saveRDS(tweets, "tweets.rds")
}

#question 2- get 1000 tweets containing "the"
if(file.exists("random.rds")){
  random = readRDS("random.rds")
}else{
   random = search_tweets("the", n = 1000, type = "recent", token = twitter_t
oken, include_rts = FALSE, lang="en")
```

```
  #save random
  saveRDS(random, "random.rds")
}

#question 3- make a matrix of tweets source
#get column names for matrix
names = sort(unique(c(tweets$source,random$source)), decreasing = FALSE)

#generate the source table
tweet.source.table = makeMatrix(tweets,random, names)

#display the table
print(tweet.source.table)
```

```
                autopo.st -@artefaktorradio Bitly BlockBible BotCulture Buffer cappertekTwitterAPI
tweets                     0    1       0       1     7           0
random tweets              1    0       1       0     1           1
                Cheap Bots, Done Quick! cm_botlove ContentStudio.io CoSchedule Crowdfire App
tweets                    44       8          1       2         1
random tweets             21       0          0       0         0
                Crypto Volume Details Devdiscourse News Desk dlvr.it Dream Journal Bot Echobox
Fabrik.fm
tweets                     0            1    13          2     4      0
random tweets              1            0     0          0     0      2
                Feedspot Scheduler financialjuice Freshdesk FS Poster GameLosingBot Hootsuite In
c. IFTTT
tweets                  1         0      0      10         1         11   13
random tweets           0         1      1      0          0         0    3
                Instagram JamesHicksMLM Publishing App Joker's BB Updates Kai-Reports LaterMe
dia Locobuzz CX
tweets            0                 0           0       1       3       0
random tweets     1                 1           1       0       0       1
                Loomly Missinglettr MovieGifBot Movietitles myloveisJoJo newsly-tech Nintendo Sw
itch Share
tweets          1        0       1       0       1       2           1
```

random tweets 1 1 0 1 0 0 0

OneDirect Suite - P Orient Twitter Monster PickNava Pipedream, Inc Plume for Andr oid

tweets 0 1 0 1 1

random tweets 1 0 1 0 1

Racing Snail Post Service Radio.co now playing Raspberry Arduino Reddit Official R oshi_Ebooks

tweets 1 0 0 1 1

random tweets 0 1 1 0 0

SAM Broadcaster Song Info Shop Matrix (UK) Application Simplify360 SoCast Digital SocialFlow

tweets 1 1 0 1 7

random tweets 1 0 1 0 1

SocialNewsDesk Sprinklr Sprout Social Squarespace Stories Travel Talon Android T empsAndSuch

tweets 4 0 2 0 0 0 0

random tweets 0 2 2 1 1 1 1

The Social Jukebox Tiana1 Treecko Bot Scream Tumblr Tweet our Streams Tweetbo t for Mac TweetDeck

tweets 0 0 15 2 0 1 22

random tweets 5 1 0 0 1 0 5

Tweetlogix Twibble.io twiteradious TwitPane for Android Twittascope twittbot.net

tweets 1 0 1 1 4 2

random tweets 0 1 0 0 0 2

Twitter Auto Post - @HotHitsUK24 Twitter Auto Post - @stradiost11 Twitter for An droid

tweets 0 0 208

random tweets 1 1 319

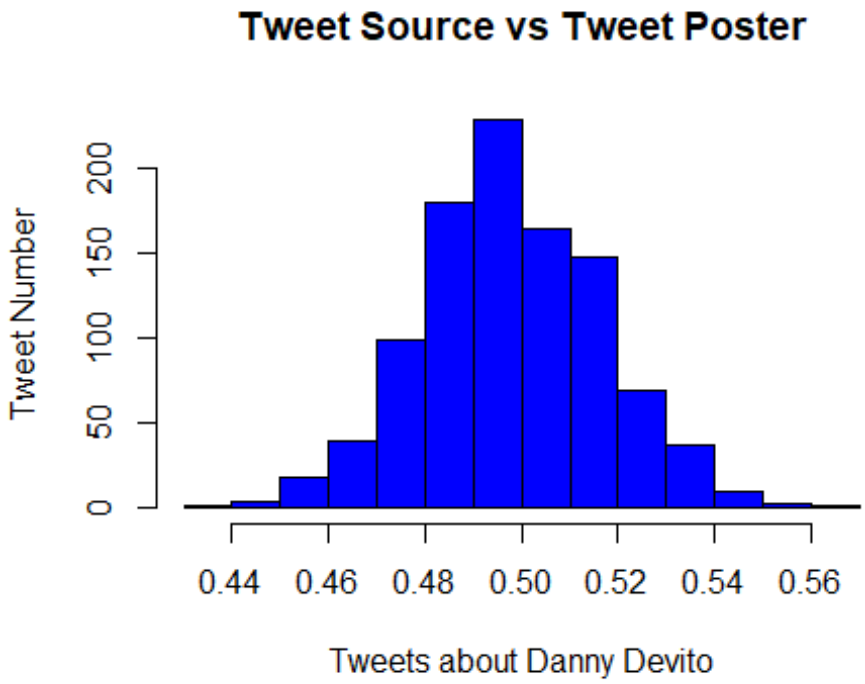
Twitter for iPad Twitter for iPhone Twitter for Mac Twitter Media Studio Twitter W eb App

```
tweets                13          342          0            1         223

random tweets         26          345          1            0         229

          Typefully Ugin WatrCoolr WeSmirch WhazupNaija WordPress.com

tweets          0   0      1      1       1        8

random tweets       4   1      0      0       0        2
```

```
#question 4- perform a chi square test
result = chisq.test(tweet.source.table, simulate.p.value = TRUE)

#null hypothesis:   No relation ship between the tweets and words
#alternative hypothesis: there is a relationship between the tweets and words

print(result$p.value)
```

```
## [1] 0.0004997501
```

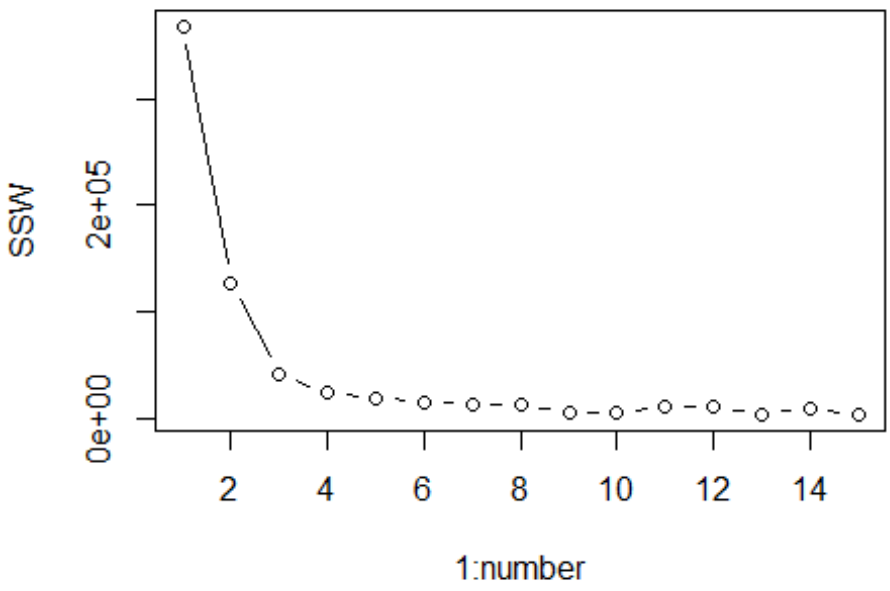
```
#with the P value being less then 0.05 we reject the null hypothesis. Therefo
re there is a relationship
#between the different tweet sources.


#question 5 - make a bootstrap distribution for "twitter for iPhone"
source = which(names == "Twitter for iPhone")     #retrieve index of "Twitter
for iPhone"
N = sum(tweet.source.table[,source])              #calculate total for "Twitt
er for iPhone"
phat1 = tweet.source.table[1,source]/N            #percentage of tweets
phat2 = tweet.source.table[2,source]/N            #percentage of random

boot.dist = replicate(1000, mean(sample(x= c("Tweet","Random"),size=N, prob =
c(phat1,phat2), replace=TRUE) =="Tweet"))
hist(boot.dist, main = "Tweet Source vs Tweet Poster", xlab = "Tweets about D
anny Devito", ylab = "Tweet Number", col = "blue")
```




```
#both mean tweets and random percentages of using a iPhone are about 0.50%

#question 6 - compute the 95% confidence interval
print( quantile(boot.dist, c(0.025,0.975)))

##      2.5%     97.5%
## 0.4614265 0.5356623
```

Question 8.2 - Clustering the Tweets

```
#question 7 - preprocess the text and make a document term matrix
#Pre process the Tweets
wordlist = c("danny devito","danny","devito","Danny Devito","devitos","dannyd
evito", stopwords::data_stopwords_smart, stopwords())

tweet.corpus = Corpus(VectorSource(tweets$text))
corpus = tm_map(tweet.corpus, function(x) iconv(x, to = 'ASCII'))

corpus = tm_map(corpus,content_transformer(function(x) gsub("(f|ht)tp(s?)://\
\S+","",x)))

corpus = tm_map(corpus,content_transformer(function(x) gsub("@\\w+","",x)))

tweet.tdm = TermDocumentMatrix(corpus, control = list(removePunctuation = TRU
E, stopwords = wordlist, removeNumbers = TRUE, tolower = TRUE, stemming = FAL
SE))


#make matrix and remove empty values
tweet.matrix = as.matrix(tweet.tdm)
empty = which(colSums(tweet.matrix)==0)
tweet.matrix = tweet.matrix[,-empty]

#question 8 - use the elbow method to find the best number of clusters
#use cosine rule and MDS to construct a weighted tweet matrix
D = dist(tweet.matrix, method = "euclidian")^2/2
mds.tweet.matrix = cmdscale(D, k=2)

#generate SSW - find the number of appropriate clusters
SSW = c()
number = 15
for (i in c(1:number))
{
  SSW[i] = kmeans(mds.tweet.matrix,i,nstart = 20)$tot.withinss
}

plot(1:number, SSW, type = "b")
```




```
#question 9 - Cluster the tweets using K means
clusterNumber = 3 #change this result manually when new data is used
K = kmeans(mds.tweet.matrix, clusterNumber, nstart = 20)
mds2.tweet.matrix = cmdscale(D,2)

#question 10 - Display the cluster graph
plot(mds2.tweet.matrix, col = K$cluster, main = "Tweet Clusters", xlab = "X D
istance", ylab = "Y Distance",
     type = "p")
```


```
#question 11 - find the size of the clusters and find the biggest
print(c("The Cluster sizes are: ", K$size))
```

```
## [1] "The Cluster sizes are: " "1"
## [3] "2"                       "2461"
```

```
print(c("The biggest Cluster contains: ",K$size[which.max(K$size)]))
```

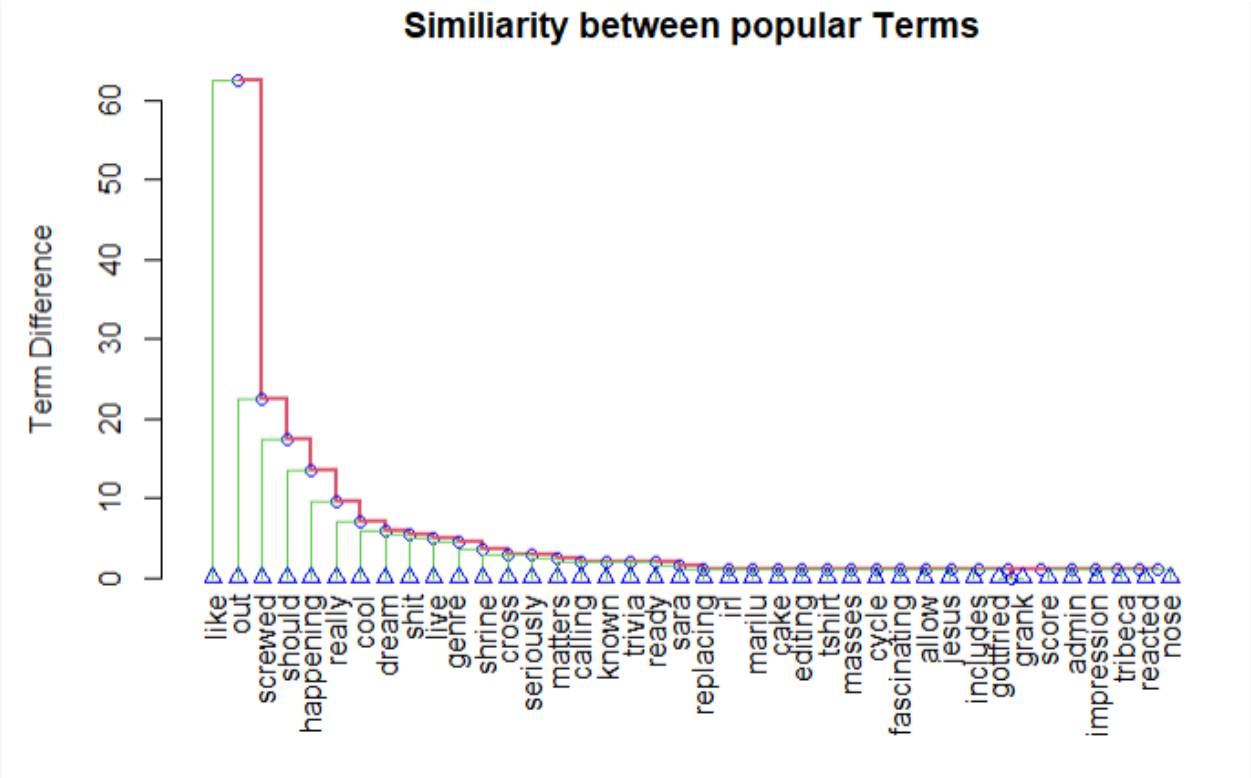
```
## [1] "The biggest Cluster contains: " "2461"
```

Question 8.3 - Identifying topics

```
#find the Biggest Cluster and retrieve terms contains in it
positions = unique(which(K$cluster == which.max(K$size)))
Popular.Tweets =  tweet.matrix[sample(positions,1500),]

#question 12 - make a Word cloud
weights = rowSums(as.matrix(Popular.Tweets))
words = names(weights)
suppressWarnings(wordcloud(words,weights,min.freq = 3, colors = colours(disti
nct = TRUE), random.color = TRUE))
```

Most Common Terms contained in Tweets

```r
#question 13- make a Dendogram

#to limit number of words get 40 random terms from the biggest cluster
Popular.Tweets =  tweet.matrix[sample(positions,40),]

#make the dendogram
D = dist(Popular.Tweets, method = "euclidian")^2/2 #cosine
h = hclust(D)
dend = as.dendrogram(h)
nodePar <- list( col = "blue")
plot(dend, main="Similiarity between popular Terms", ylab = "Term Difference"
,
     nodePar = nodePar, edgePar = list(col = 2:3, lwd = 2:1))
```



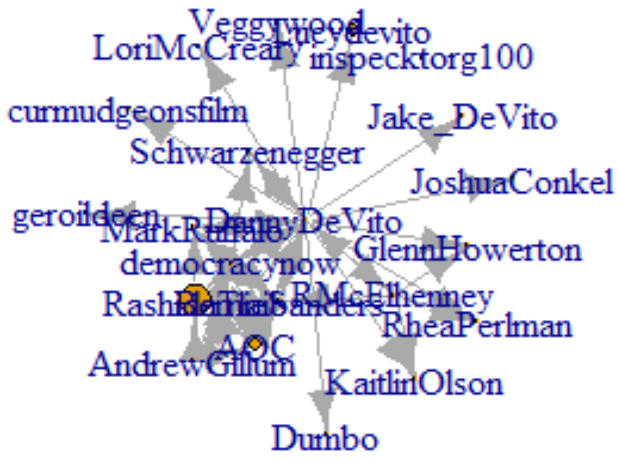
```
#question 14- interpret the results


The Word Cloud lacks context as it has many general words so results must be
gathered from specific words based on multiple runs of the word cloud algorit
hm Danny devitos most common tweets are about events relating to politics, ce
lebrity news, or his personal life

The dendrogram is not the best way to find a trend as due to the limit of 40
terms to make it readable only a small amount of insight can be gained per a
run. The similarity measures the dendogram shows tells us politics and celebr
ity life, personal life as well but the words dont have a lot of context
```

Question 8.4 - Building Networks

```
#returns the number of followers  for each friend
count.followers = function(friends)
{
  follower.count = c()
  for (i in 1:length(friends))
  {
      follower.count[i] = friends[[i]]$getFollowersCount()
  }
  return(follower.count)
}

#returns an edgelist for the graph of friends
user.to.edgelist = function(user,friends)
{
    friend.names = c()
    for (i in c(1:length(friends)))
    {
        friend.names[i] = friends[[i]]$getScreenName()
    }

    #build edge list
    user.name = rep(user$getScreenName(), length(friends))
    e1 = cbind(user.name, friend.names)
    return(e1)
}

#get degree 1.5 and degree 1 edges
getEdges = function(topfriends,user)
{
   connections = c()
if(file.exists("connections.rds")){
  connections = readRDS("connections.rds")
}else{
    #Add Danny devito if not already present
    if( length(topfriends) == 20)
    topfriends = c(topfriends,user)

    #create empty matrix and retrieve edges
    connections = matrix(data = rep(0,length(topfriends)^2*2), ncol = 2, byro
w = TRUE)
    for (i in 1:length(topfriends))
    {
      #user 1
      name1 = screenName(getUser(topfriends[i]))
```

```
      for (j in 1:length(topfriends))
      {
         #user 2
          name2 = screenName(getUser(topfriends[j]))
          if(name1 != name2)
          {
              #check if friends are following each other
              link = lookup_friendships(name1, name2)
              if(length(link) != 0)
              {
                  if(link$value[4])
                  {
                    #save the results
                    connections[i*j,1] = name1
                    connections[i*j,2] = name2
                  }
              }
          }
      }
    }

  #save connections between friends
  saveRDS(connections, "connections.rds")
}
  #filter out empty rows
  empties = which(connections[,1] == 0)
  connections = connections[-empties,]

  return(connections)
}

#Question 15 - find the 20 most popular friends of Danny devito

#Get Danny Devito's friends
user = getUser("DannyDevito")
friends = user$getFriends(100)

#find 20 of the most popular friends
friendfollowercount = count.followers(friends)
friendposition = order(friendfollowercount,decreasing = TRUE)[1:20]
topfriends = c()
topfriends = friends[friendposition]

#Question 16 - Get Number of Tweets from most popular friends
totalTweets = c()
for (i in 1:length(topfriends))
{
    name = getUser(topfriends[i])
```

```
    totalTweets[i] = statusesCount(name)
}


#Question 17 - Make a degree 1.5 egocentric graph about Danny Devito and his
followers

#combine the 1 degree edges and 1.5 degree edges and plot them
#1.5 connections
connections = getEdges(topfriends,user)
edges = user.to.edgelist(user, topfriends)

#1 degree connections
edges = unique(rbind(edges,connections))
gd = graph.edgelist(edges)
suppressWarnings(plot(gd, layout = layout.fruchterman.reingold, vertex.size =
40*(totalTweets/sum(totalTweets))))
```




```
#Question 18 - compute the centrality using betweenness
sort(betweenness(gd), decreasing = TRUE)[1:3]

##  DannyDeVito democracynow RashidaTlaib
##          132            5            4
```

```
#question 19 - interpret the results


Danny DeVito is the most central vertex which is to be expected as he is user
of interest. Other popular vertex’s are a democracy based group which he a is
likely frequencts if not a part and RashidaTlaib is a US senator who is like
holds similar policital views to himself and many of his friends leading to m
any 1.5 degree connections.

vertex size is based on the tweets making the user with the most tweets the b
iggest vertex. Danny Devito Does not post the most tweets but instead Rashida
Tlaib
```