Cover Sheet

By including this statement, we, all the students listed in the table below, declare that:

- We hold a copy of this assignment if the original is lost or damaged.
- We hereby certify that no part of this assignment 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 the assignment has been written for us by any other person except where collaboration has been authorised by the unit coordinator.
- We are aware that this work may be reproduced and submitted to plagiarism detection software programs for the purpose of detecting possible plagiarism; this software may retain a copy on its database for future plagiarism checking.
- We hereby certify that no part of this assignment or product has been submitted by any of us in another (previous or current) assessment, except where appropriately referenced, and with prior permission from the unit coordinator for this unit.
- We hereby certify that we have read and understand what the University considers to be academic misconduct, and that we are aware of the penalties that may be imposed for academic misconduct.

Name	**Student Number**	**Contribution (%)**
Rachel Hardie	18820821	25%
Dylan Wang	18998014	25%
Dylan Yoo	18640377	25%
Sreenath Ramachandran	18878716	25%

Introduction

This report examines and analyses the twitter language used by popular author J.K. Rowling and twitter users who interact with or mention her.

Analysis of Twitter language about J.K. Rowling

In this section, we want to examine the language used in tweets. The rtweet package was used to to download 1000 tweets that contain the terms 'J.K. Rowling', rowling, and jk_rowling then save them as a .csv so as to not repeatedly call the api and because of the fast changing nature of twitter the frequent terms could change significantly. The terms we are analysing were captured on 28th September 2018.

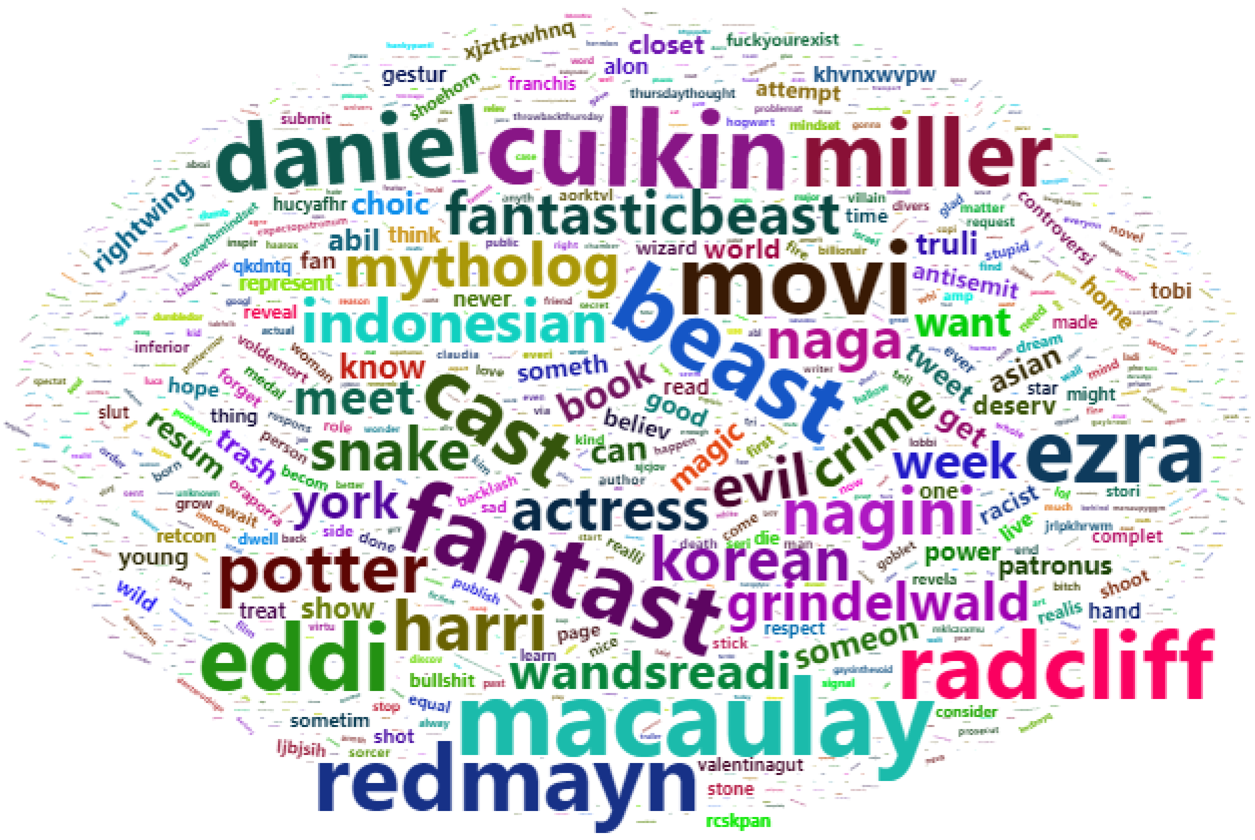
```
library("ROAuth")
library("tm")
library("wordcloud")
library("rtweet")
library("twitteR")
library("wordcloud2")
#install.packages("dendextend")
library("dendextend")
library("knitr")

##Code below was used to create the SWA project.csv file,
## which is used for the work to have a consistent set of tweets for analysis.

#key="3g0mNQkwVxe0JKiRfSyyS0rvu"
#secret="Wu38XdVVJFPWpqWVBQDwyl2Reqhmg8XhDWjTJl3XwOqNlwYEcx"
#access_token="243587433-BPMCMRHSyb3pSJS3raa0w9qPCU3RAnB3aPZtjVEM"
#access_secret="g8iN8UChw7jAnybFKfMxPWiWiS8hZw1lVeKzmipnnu3zR"
#app="WSU300958"
#setup_twitter_oauth(key,secret,access_token,access_secret)
#tweets=search_tweets(q="'J.K.Rowling' OR Rowling OR jk_rowling",n=1000,lang="en")
#save_as_csv(tweets,"SWAProject.csv")
tweets=read.csv("SWAProject.csv")
```

We then constructed a document-term matrix using TFIDF weighting to describe the frequency of terms in the tweets we had collected.

```
corpus=Corpus(VectorSource(tweets$text))
corpus = tm_map(corpus, function(x) iconv(x, to = 'UTF8', sub = 'byte'))
corpus = tm_map(corpus, function(x) iconv(x, to = 'ASCII', sub = ' '))
corpus = tm_map(corpus,removeNumbers)
corpus = tm_map(corpus, removeWords,c(stopwords(),"J.K. Rowling","https", "t.co"))
corpus = tm_map(corpus,removePunctuation)
corpus = tm_map(corpus,stripWhitespace)
corpus = tm_map(corpus, tolower)
corpus = tm_map(corpus, stemDocument)
corpus = tm_map(corpus, removeWords,c(stopwords(),"jk","https", "t.co","rowl","jkrowl","will"
,"like","just","doe","far","quot","look","take","make","jaqkcpmtg","ygorfremo","ask","peopl",
"next","twitter","peopl","new","write","charact","got","ohhgmljqrm","say","defend","respond",
 "vrmgsnrsa","bbepgeik","nkuwxgsn","novrijqr","nlrbgguhxt"))

dtm = DocumentTermMatrix(corpus)
tweet.wdtm = weightTfIdf(dtm)
tweet.matrix = as.matrix(tweet.wdtm)
```

After removing non-informative words we sum the frequencies of each terms in all documents to obtain a vector of term frequencies summed over all tweets then visualised the frequency of words in a word cloud.

```
##term frequencies are stored in the vector 'freqs'
freqs=colSums(tweet.matrix)
data = data.frame(names(freqs), freqs)
wordcloud2(data, size = 0.3, shape = "circle")
```

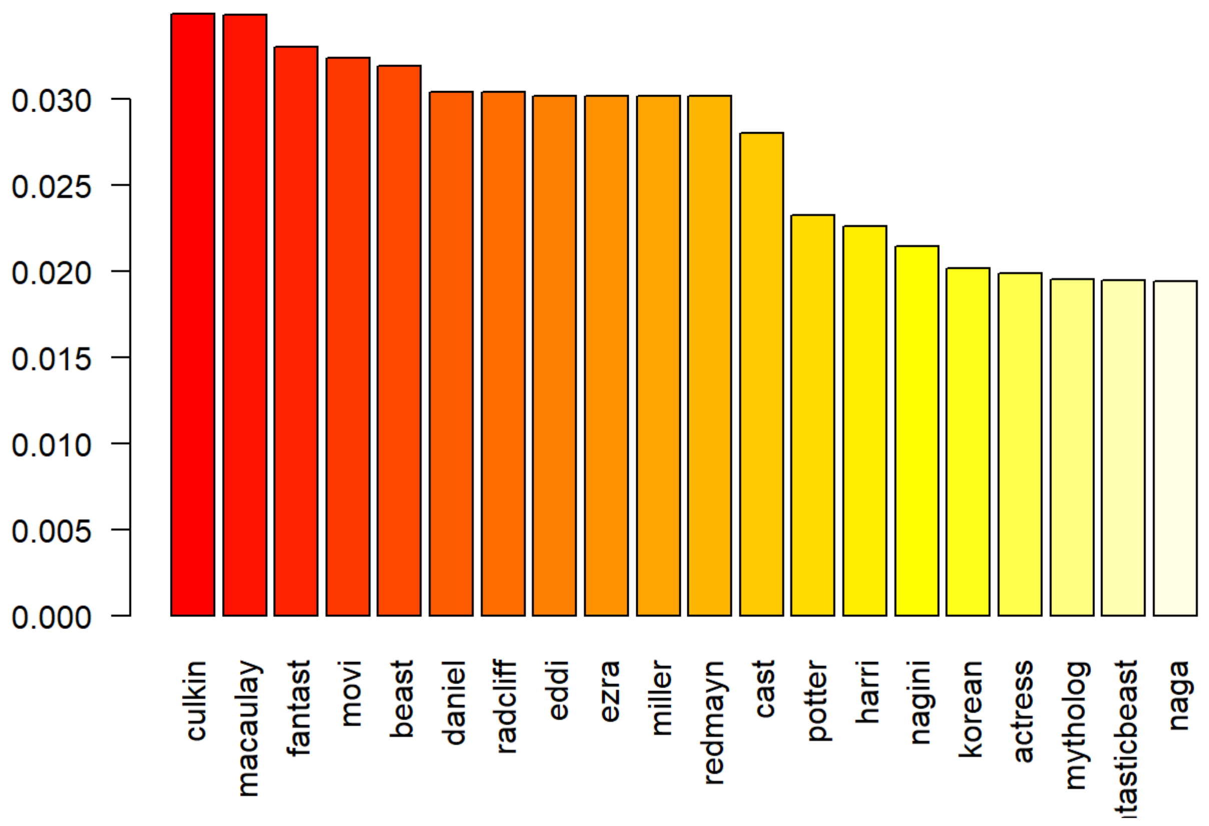


```
head(freqs)
```

```
##    believ      book       can      good    happen      kind
## 13.516580 23.166611 19.616821 14.278003  6.780191  8.157560
```

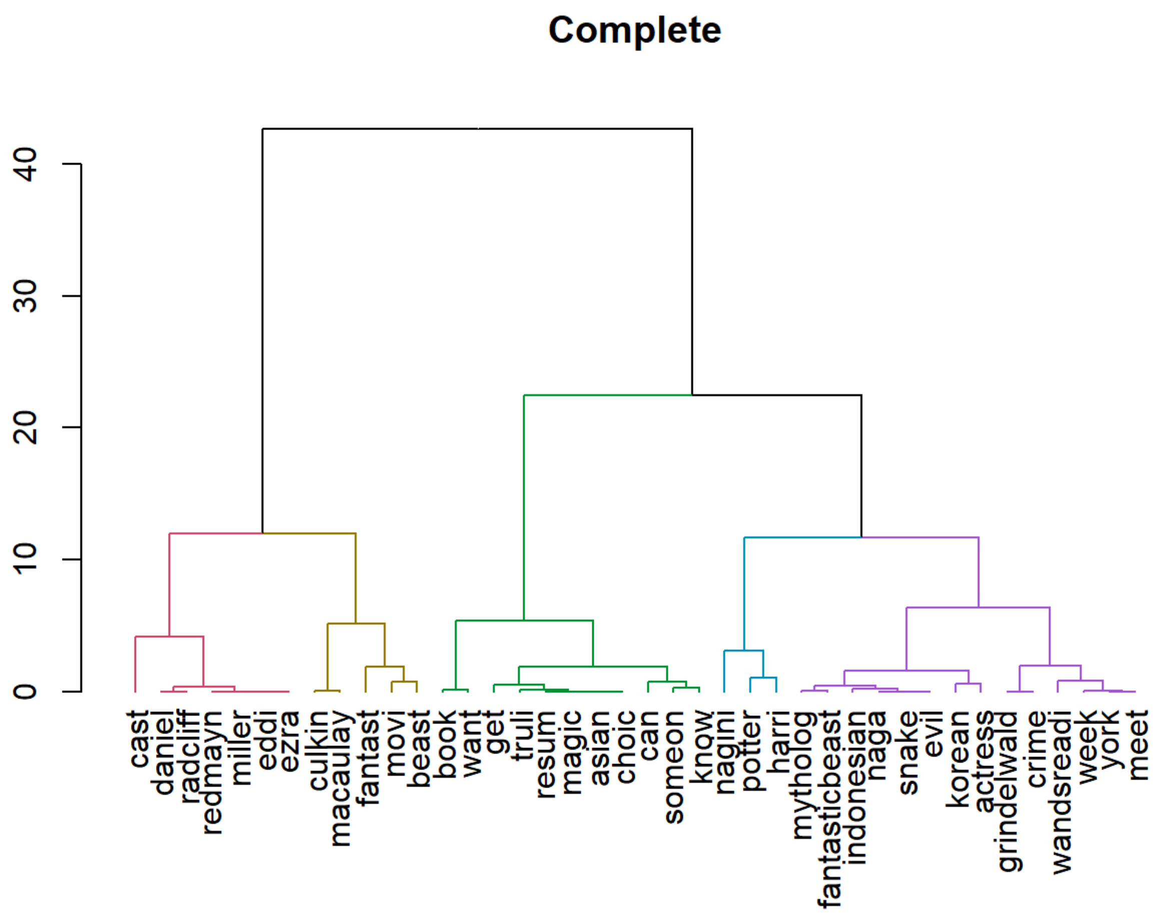
Then we Compute the proportion of each term in the tweets from the vector of term frequencies and Visualize the top 20 words and their proportion by using a bar plot.

```
id=order(freqs,decreasing = TRUE)[1:20]
topWords=colnames(tweet.matrix)[id]
N = length(freqs)
words.proportion = freqs/N
barplot(words.proportion[id], col=heat.colors(20),las=2)
```

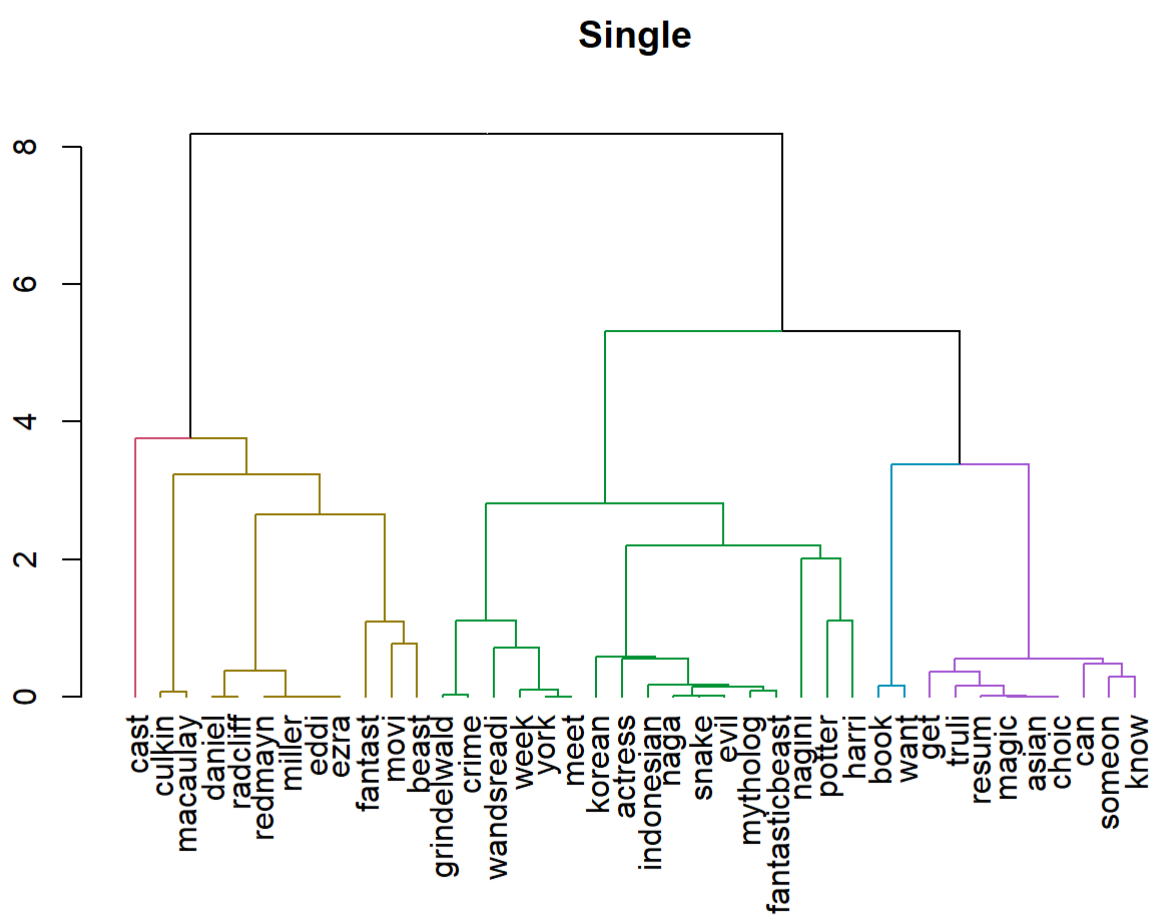


In order to visualise the clustering of words we created a a single linkage and complete linkage dendrogram of the 40 most frequent terms.

```
id2 = order(freqs, decreasing = TRUE)[1:40]
topWords.clus = colnames(tweet.matrix)[id2]
 D = dist(freqs[id2])

##Complete Linkage
h.complete = hclust(D)
h.complete <- color_branches(h.complete, k = 5)
plot(h.complete,main = "Complete")
```



```
##Single Linkage
h.single = hclust(D, method="single")
h.single <- color_branches(h.single, k = 5)
plot(h.single,main = "Single")
```



After review, we concluded that Complete Linkage is the better option as the clusters are better defined and it does not suffer from clusters being forced together due to single elements which, specifically for these tweets, is important because some terms such as the titles of books and movies written by or based on the works of J.K. Rowling can cause there to be different clusters linked together. This is known as chaining phenomena and can cause clusters to have a large diameter, wheras complete linkage will give us clusters with a more equal diameter.

What do these words tell us about J.K. Rowling? Comment on what people are saying about this person?

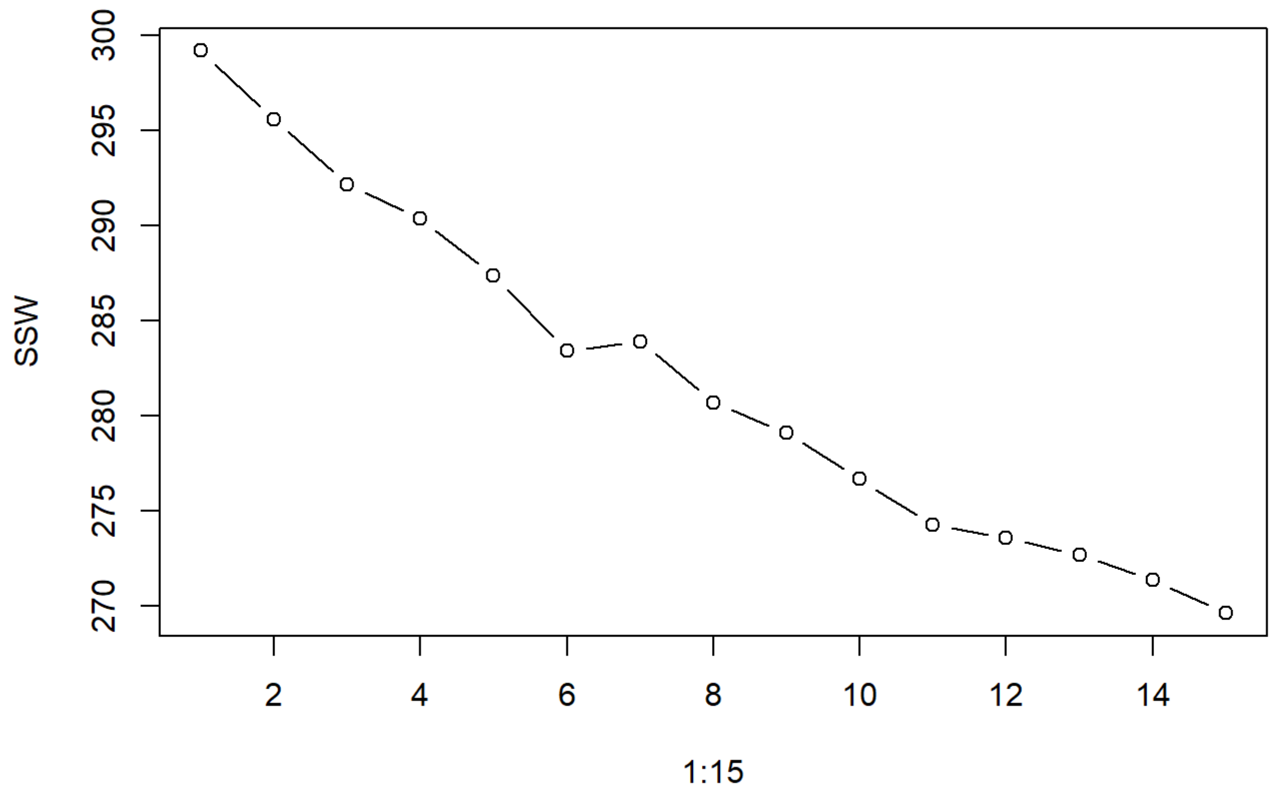
The majority of terms are related more to the creations of J.K. Rowling than about her personally. Actors are frequently mentioned in relation to her, most likely because she is involved with the production of major motion pictures based on her written works. There is also a significant cluster regarding the upcoming film, ‘Fantastic Beasts and Where To Find Them: The Crimes of Grindelwald’ and a Korean Actress.

Clustering the Users Who Posted Tweets About the Public Figure

We want to categorize (cluster) the users of the tweets about J.K. Rowling based on the descriptions provided in their Twitter account to figure out what kind of users are tweeting about them. Descriptions in the users’ Twitter profiles give a short piece of information about the Twitter handle. To cluster users we built a new document term matrix by using the user descriptions of the tweets.

We used K-means clustering with cosine distance for better clustering. First a Distance matrix was required to find cosine distance then we performed MDS using 300 dimensions

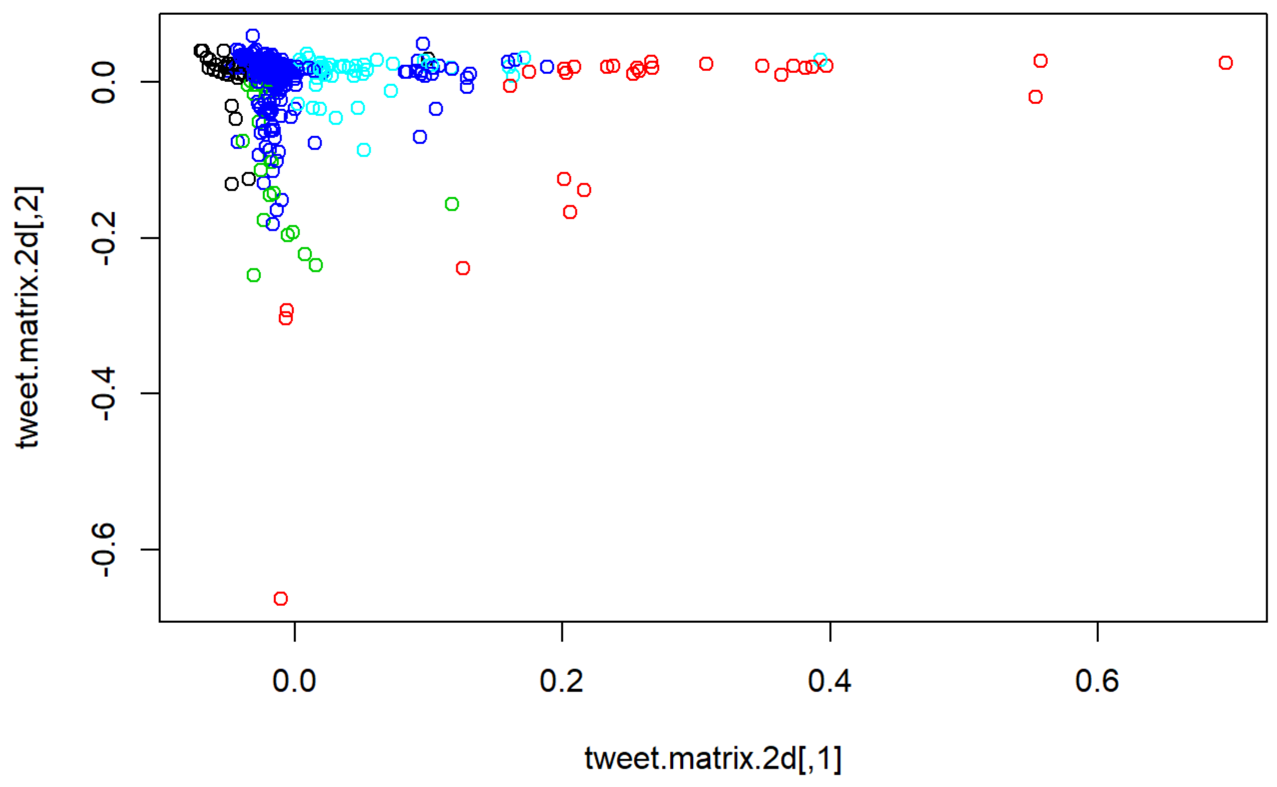
```r
corpus1=Corpus(VectorSource(tweets$description))
corpus1 = tm_map(corpus1, function(x) iconv(x, to = 'UTF8', sub = 'byte'))
corpus1 = tm_map(corpus1, function(x) iconv(x, to = 'ASCII', sub = ' '))
corpus1 = tm_map(corpus1,removeNumbers)
corpus1 = tm_map(corpus1, removeWords,c(stopwords(),"youuuuuuuufufufuufefuufefuufefufufuf","u
fbuf"))
corpus1 = tm_map(corpus1,removePunctuation)
corpus1 = tm_map(corpus1,stripWhitespace)
corpus1 = tm_map(corpus1, tolower)
corpus1 = tm_map(corpus1, stemDocument)
corpus1 = tm_map(corpus1, removeWords,c(stopwords(),"youuuuuuuufufufuufefuufefuufefufufuf","u
fbuf","deppuufuduuucuucuaubaucbufcuubudueauubufcueuuub","ufcuuuubudaucucufudfububucuducbuaaua
uacueuuedubuuuuauduuufcuuuueuufu"))
descriptionDtm = DocumentTermMatrix(corpus1)
tweet.wdescriptionDtm = weightTfIdf(descriptionDtm)
tweet.descriptionmatrix = as.matrix(tweet.wdescriptionDtm)
#removing profiles with no description.
empties = which(rowSums(abs(tweet.descriptionmatrix)) == 0)
tweet.descriptionmatrix = tweet.descriptionmatrix[-empties, ]

#We are using Kmeans clustering with cosine distance for better clustering.
#Distance matrix is required to find cosine distance.
norm.tweet.matrix = diag(1/sqrt(rowSums(tweet.descriptionmatrix^2))) %*% tweet.descriptionmat
rix
D = dist(norm.tweet.matrix, method = "euclidean")^2/2
D[is.na(D)]=0# Convert na's in the distance matrix to 0.
## perform MDS using 300 dimensions
mds.tweet.matrix <- cmdscale(D, k=300)
##Elbow
SSW = rep(0,15)
for (i in 1:15) {
  number.of.clusters=i
  k=kmeans(mds.tweet.matrix,number.of.clusters,nstart = 10)
  SSW[i]=k$tot.withinss
}
plot(1:15,SSW,type="b")
```



After inspecting the plot we used the elbow method to selected the number of clusers to be five.

Users were then clustered and visualised in two-dimensional vector space with K-means clustering using cosine distance and number of clusters obtained from the elbow method.

```
K = kmeans(mds.tweet.matrix,5, nstart = 10 )
tweet.matrix.2d = cmdscale(D,k=2)
plot(tweet.matrix.2d, col = K$cluster)
```



We listed the top 10 words associated with each user cluster then manually determined the type of user profiles for each cluster (e.g. journalist, organisation, personal).

```
## [1] "Cluster 1:"
```

```
##       news     latest        hot  entertain     celebr      trust 
## 0.73434291 0.19130630 0.15619185 0.14603676 0.11520210 0.11166849 
##      break       buzz      share      world 
## 0.10903829 0.09770377 0.09648133 0.09223339 
```

```
## [1] "Cluster 2:"
```

```
##          fan      account         star         movi          war 
##   0.71330125   0.58246232   0.22360585   0.13727035   0.12981003 
##         idea mollybofficí  accountufau          big         trek 
##   0.12831708   0.08741916   0.08741916   0.08632540   0.07494149 
```

```
## [1] "Cluster 3:"
##                      know                    potter
##                 0.5659281                 0.3399740
##                     harri                    jkrowl
##                 0.3362529                 0.1273976
##                      depp                    devote
##                 0.1062105                 0.1062105
## johnnydeppismygrindelwald         proudofjohnnydepp
##                 0.1062105                 0.1062105
##    wearewithyoujohnnydepp   youcantkeepagoodmandown
##                 0.1062105                 0.1062105
```

```
## [1] "Cluster 4:"
```

```
##       just       live       life     debora       like     writer 
## 0.02845849 0.02709392 0.02619351 0.02423185 0.02273985 0.02184494 
##       book     reason       girl        die 
## 0.02114287 0.01983203 0.01938908 0.01902200
```

```
## [1] "Cluster 5:"
```

```
##       love       book  tsutaetai      allah       sake       sinc 
## 0.79865228 0.08220614 0.07382062 0.07382062 0.07382062 0.07229281 
##        lot  instagram       game     follow 
## 0.07094780 0.06673683 0.06467856 0.06353174
```

Clusters 2,3,4 and 5 all appear to be fan accounts, while cluster 1 seems more likely to be news and entertainment accounts based on the terms in each cluster.

Retweet Analysis of the User Clusters

We want to examine if the number of retweeted tweets is independent of which user cluster posted the tweet.

Retweets from each cluster are identified and their IDs are stored for later use.

```
retweetsId=which(tweets[clusterTweetsId,]$is_retweet==TRUE)
retweetsId2=which(tweets[clusterTweetsId2,]$is_retweet==TRUE)
retweetsId3=which(tweets[clusterTweetsId3,]$is_retweet==TRUE)
retweetsId4=which(tweets[clusterTweetsId4,]$is_retweet==TRUE)
retweetsId5=which(tweets[clusterTweetsId5,]$is_retweet==TRUE)
```

Next, we constructed a 2xM table where M is the number of user clusters we found previously. Each row represents the total number of retweeted tweets and non-retweeted tweets in each cluster.

```
#Number of retweeted tweets in each cluster.
R=length(retweetsId)
R2=length(retweetsId2)
R3=length(retweetsId3)
R4=length(retweetsId4)
R5=length(retweetsId5)
## Number of tweets which are not retweeted in each cluster
NR=length(clusterTweetsId)-R
NR2=length(clusterTweetsId2)-R2
NR3=length(clusterTweetsId3)-R3
NR4=length(clusterTweetsId4)-R4
NR5=length(clusterTweetsId5)-R5

comparison.table = matrix(c(R,R2,R3,R4,R5,NR,NR2,NR3,NR4,NR5),ncol = 5,byrow = TRUE)
colnames(comparison.table)=c("Cluster1","Cluster2","Cluster3","Cluster4","Cluster5")
rownames(comparison.table)=c("Retweets","Not Retweets")
comparison.table=as.table(comparison.table)
comparison.table
```

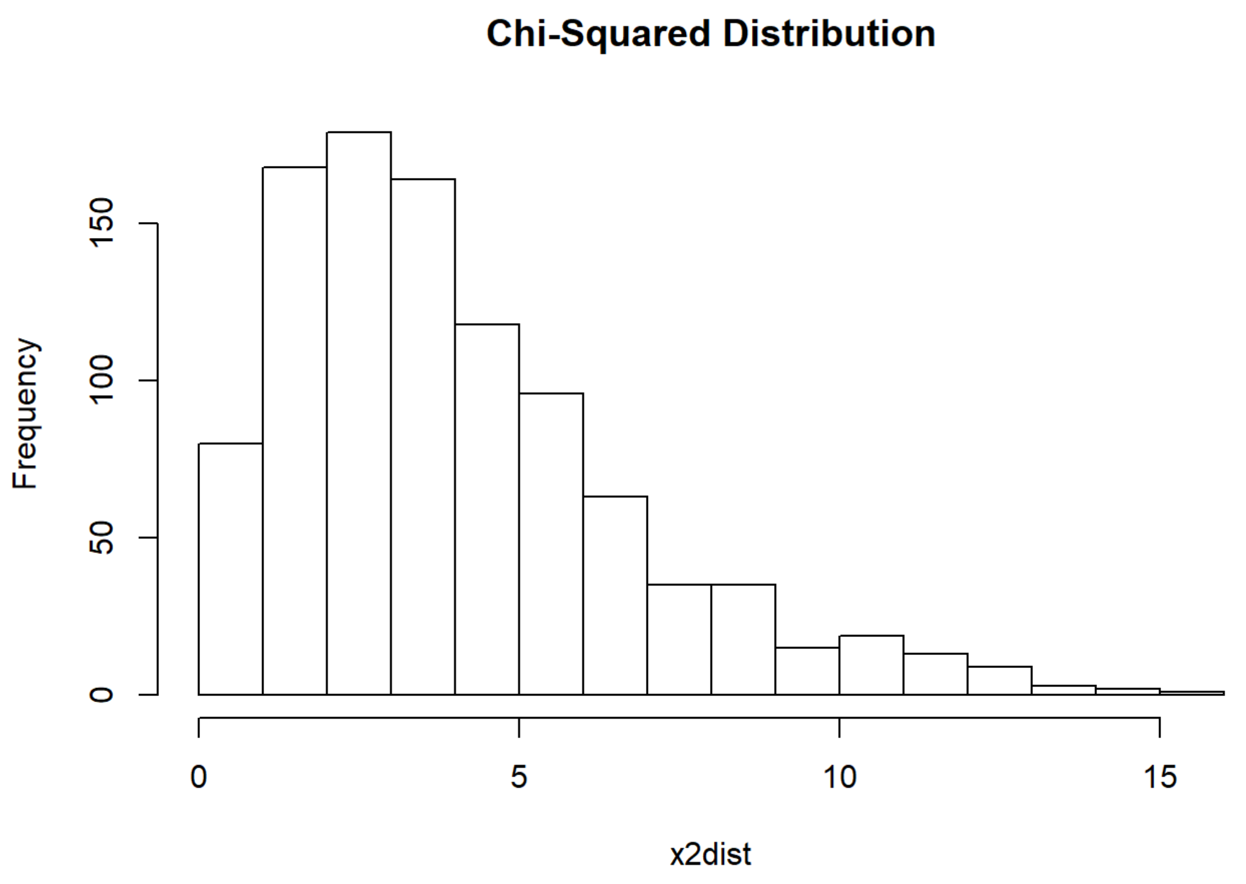
```
##              Cluster1 Cluster2 Cluster3 Cluster4 Cluster5
## Retweets           22       26       18      492       32
## Not Retweets       12       12        7      248       13
```

Finally, we tested if retweeting is independent of user groups by performing a Chi-Square test for Independence on the data seen in comparison table above

```
#Chi-sq test for Independence
table.matrix=as.matrix(comparison.table)
stretchTable=function(tab,variableNames){
  tabx=rep(rownames(tab),rowSums(tab))
  l=ncol(tab)
  m=nrow(tab)
  cn=colnames(tab)
  taby=c()
  for(a in 1:m){
    for(b in 1:l){
      taby=c(taby,rep(cn[b],tab[a,b]))
    }
  }
  d=data.frame(x=tabx,y=taby)
  colnames(d)=variableNames
  return(d)
}
tab2= stretchTable(table.matrix,c("Tweets","Cluster"))
table(tab2)
```

```
##                Cluster
## Tweets           Cluster1 Cluster2 Cluster3 Cluster4 Cluster5
##   Not Retweets         12       12        7      248       13
##   Retweets             22       26       18      492       32
```

```
 expectedIndependent = function(X) {
    n = sum(X)
    p = rowSums(X)/sum(X)
   q = colSums(X)/sum(X)
    return(p %o% q * n) # outer product creates table
 }
 chiSquaredStatistic = function(X, E) {
     return(sum((X-E)^2/E))
 }

  ## compute the expected table
 E = expectedIndependent(table(tab2)) # compute expected counts

## compute the randomisation distribution - this was commented out after running to stop rand
omisation of clusters during analysis
x2dist = replicate(1000, { # compute 1000 randomised chi-squared statistics
   clusterShuffle = sample(tab2$Cluster)
  tweetShuffle = sample(tab2$Tweets)
  Xindep = table(tweetShuffle, clusterShuffle)
   chiSquaredStatistic(Xindep, E)
 })
 hist(x2dist,main = "Chi-Squared Distribution")
```



```
 x2 = chiSquaredStatistic(table(tab2), E)
 sprintf("Chi-Sq statistic is: %f",x2)
```

```
## [1] "Chi-Sq statistic is: 0.824682"
```

Hypothesis testing the independence of retweets and clusters

Ho: retweets and clusters are independent Ha: retweets and clusters are not independent

```
# calculate pvalue
p.value=mean(x2dist>x2)
p.value
```

```
## [1] 0.945
```

We Do not reject null nypothesis as p-value > alpha(0.05)

As the p-value is greater than the significance level do not reject the null hypothesis.

This means that we do not have enough evidence to suggest that the retweets and clusters are not independent.

Conclusion

When we consider all of our findings together we can say that J.K Rowling is intertwined with her creations from the Harry Potter novels and the subsequent film adaptations. Many of the frequent terms are actors from the film adaptions and also the film “Fantastic Beasts and Where To Find Them: The Crimes of Grindewald” which is due to be released.

We also note that because J.K. Rowling is a very well known author and has a very active twitter account there can be a huge shift of conversation topics in a short period of time and our analysis is based on only 1000 tweets from the 28th September 2018.