Tejal Bhangale

R : Hands-on exercises

1. Download the news articles dataset named “20news-18828.tar.gz”, from an online textual dataset repository: http://qwone.com/%7Ejason/20Newsgroups/. The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup articles, partitioned (nearly) evenly across 20 different newsgroups.

2. Convert them to a term frequency (TF) matrix.
(each row is an article, each column is a unique term, and each entry of this TF matrix is term frequency). In this step, it is important to preprocess the news articles using proper data preprocessing techniques such as

a. Remove semantically insignificant words such as prepositions, pronouns, adverbs, articles,etc.

- load the corpus from the directory
  all <- Corpus(DirSource("D:/Sem 2/Temporal and spatial data/20news-18828", encoding = "UTF-8", recursive=TRUE), readerControl=list(reader=readPlain,language="en"))

- remove punctuations
  all.p <- tm_map (all, removePunctuation)

- change text to lower cases
  all.p <- tm_map (all.p , content_transformer(tolower))

- remove english stop words
  all.p <- tm_map(all.p, content_transformer(removeWords),stopwords("english"))

- apply SMART words choice technique
  all.p <- tm_map(all.p, content_transformer(removeWords),stopwords("SMART"))

- apply porter stemmer
  all.p <- tm_map (all.p, stemDocument)

- remove numbers
  all.p <- tm_map (all.p, removeNumbers)

- remove extra whitespace
  all.p <- tm_map (all.p, stripWhitespace)

b. Turn the corpus into Document Term Matrix (DTM)
   dtm <- DocumentTermMatrix (all.p)

c. Remove rows/columns with many missing values. Remove sparse words with the factor 0.99.
   dtm <- removeSparseTerms(dtm, 0.99)

```
#load these libraries
install.packages('tm')
require('tm')
require('SnowballC')

#data pre-processing
all <- Corpus(DirSource("D:/Sem 2/Temporal and spatial data/20news-18828", +
      encoding = "UTF-8", recursive=TRUE), +
      readerControl=list(reader=readPlain,language="en"))

all.p <- tm_map (all, removePunctuation)
all.p  <- tm_map (all.p , content_transformer(tolower))
all.p <- tm_map(all.p, content_transformer(removeWords),stopwords("english"))
all.p <- tm_map(all.p, content_transformer(removeWords),stopwords("SMART"))
all.p <- tm_map (all.p, stemDocument)
all.p <- tm_map (all.p, removeNumbers)
all.p <- tm_map (all.p, stripWhitespace)

#turn into Document-Term Matrix
dtm <- DocumentTermMatrix (all.p)

#remove sparse terms
dtm <- removeSparseTerms(dtm, 0.99)

#conversion to data frame
library(tidytext)
DF <- tidy(dtm)
```

3. Feature selection: Select top 100 best features from the data matrix.
Select the top-100 most frequent terms as the top 100 best features.

```
#Feature Selection
#find top 100 most frequent words
#convert document-term matrix to matrix
m <- as.matrix(dtm)
#convert matrix to data frame
dataframe_m = tidy(m)
#sort matrix by sum of frequencies
sortedMatrix <- sort(colSums(m), decreasing=TRUE)
#select top 100 words from sorted matrix
sorted100 <- head(sortedMatrix, 100)
```

OutPut:

```
> sorted100
  subject     write    articl      dont     peopl      time      make      work      good 
    20635     14828     11859      9638      9515      7973      7180      6489      5973 
     year    system     thing   problem       god      file  question    window      call 
     5864      5787      5376      5299      5222      4857      4552      4372      4347 
    point      post   program       run      read     state     drive    number      find 
     4170      4157      4105      4023      3854      3849      3620      3605      3550 
     back      game       day    includ    inform    person       ive    univers    govern 
     3513      3495      3486      3414      3325      3261      3239      3236      3212 
      bit christian     email      part     start    reason   support      case       law 
     3205      3204      3199      3186      3179      3151      2979      2949      2919 
      set  interest    comput       car     power     group      fact      imag      give 
     2890      2862      2857      2777      2766      2750      2749      2738      2715 
      key      made    doesnt       lot   control       put      line      list     david 
     2684      2633      2630      2609      2605      2605      2600      2585      2567 
     live      data      word     great     world      book     exist      card     space 
     2508      2472      2453      2434      2418      2383      2366      2362      2337 
    didnt   softwar      long      show      team      play     chang   opinion   general 
     2330      2308      2288      2268      2252      2245      2229      2212      2207 
     kill     claim      john     place       gun      true      idea      base      mean 
     2202      2196      2192      2176      2154      2138      2122      2119      2104 
      end      talk      chip     order    public   version  armenian    nation      issu 
     2099      2093      2088      2086      2071      2052      2019      2009      1996 
     note 
     1971 
```

4. **Compute the similarity between each pair of articles.**

   Select 1000 sample documents from all the documents and compute Euclidean Similarity, Cosine similarity and Jaccard similarity between each pair of articles.

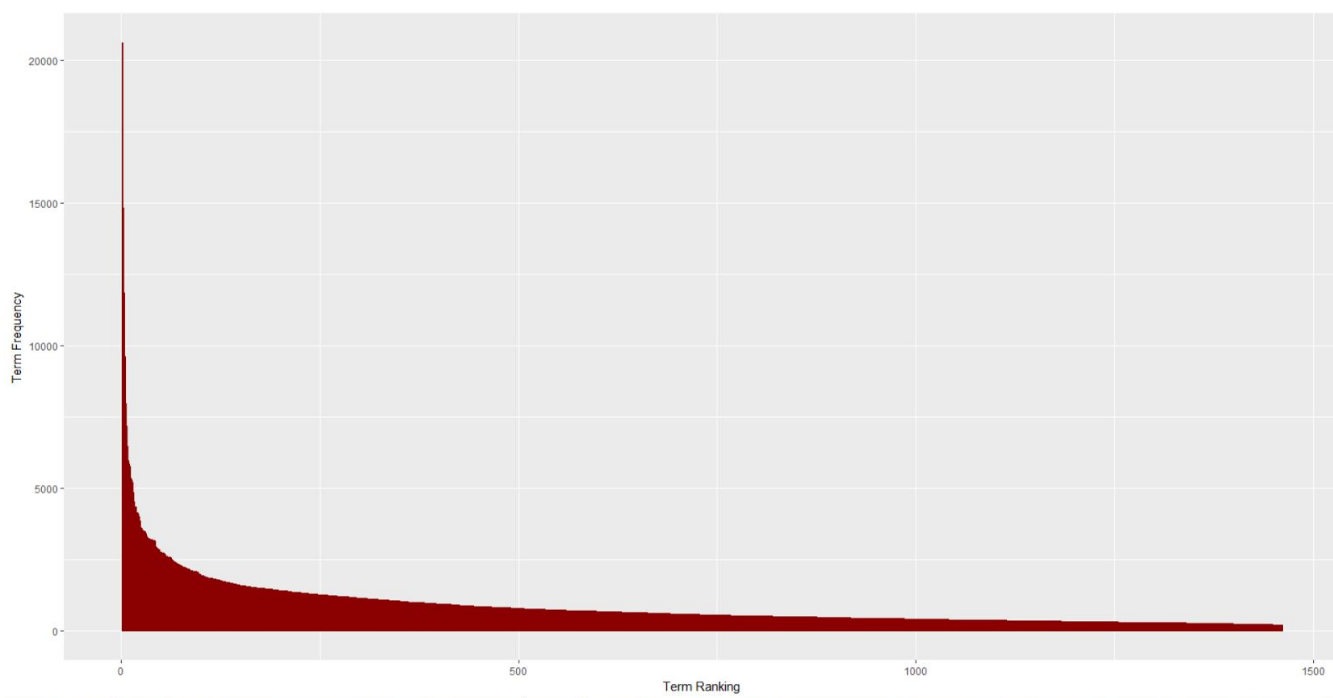
```
#find similarities
install.packages('proxy')
require('proxy')

#Sample 1000 docs
#-----------------
samplem <- head(as.matrix(dtm),1000)
dataframe_samplem = tidy(samplem)

#Cosine similarity
#-------------------
m_cosdis <- dist(samplem, method="cosine")

#jaccard similarity
#-------------------
m_jacdis <- dist(samplem, method="Jaccard")

#Eucleadian similarity
#---------------------
m_eucdis <- dist(samplem, method="euclidean")
```

The resulting matrices m_cosdis, m_jacdis, m_eucdis hold each similarity matrix.

5. Term Frequency analysis.

N denote the number of terms, provide term frequency histogram plot where X-axis (x=[1, 2, 3, 4, …, N]) represents the rankings of term frequencies in a descending order from left to right. For instance, 1 denotes the rank-1 term with the highest frequency; 2 denotes the rank-2 term with the 2nd highest frequency Y-axis represents corresponding term frequencies.

```
#plot frequency
install.packages('reshape2')
require('reshape2')
install.packages('ggplot2')
require('ggplot2')

dfplot <- as.data.frame(melt(sortedMatrix))
dfplot$word <- dimnames(dfplot)[[1]]
dfplot$word <- factor(dfplot$word,
                      levels=dfplot$word[order(dfplot$value,
                                               decreasing=TRUE)])
dfplot$word <- as.numeric(dfplot$word)
fig <- ggplot(dfplot, aes(x=word, y=value)) +
       geom_bar(stat="identity", fill="grey", colour="darkred")
fig <- fig + xlab("Term Ranking")
fig <- fig + ylab("Term Frequency")
print(fig)
```



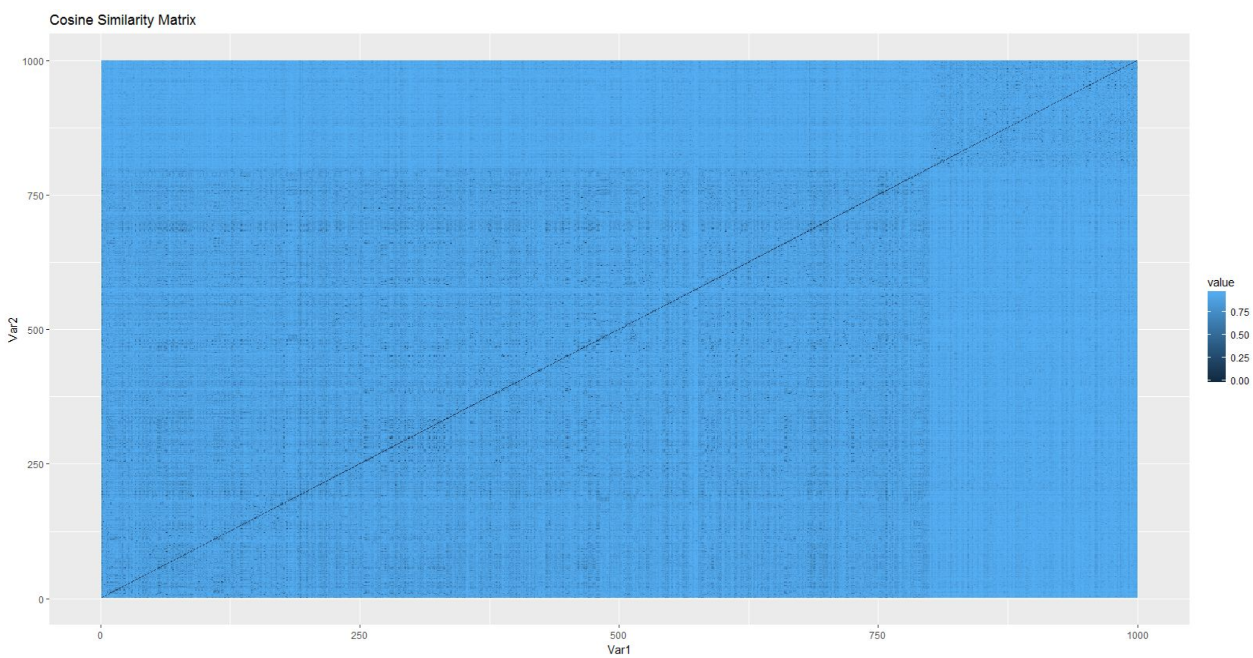
6. Heatmap based similarity analysis of document pairs:

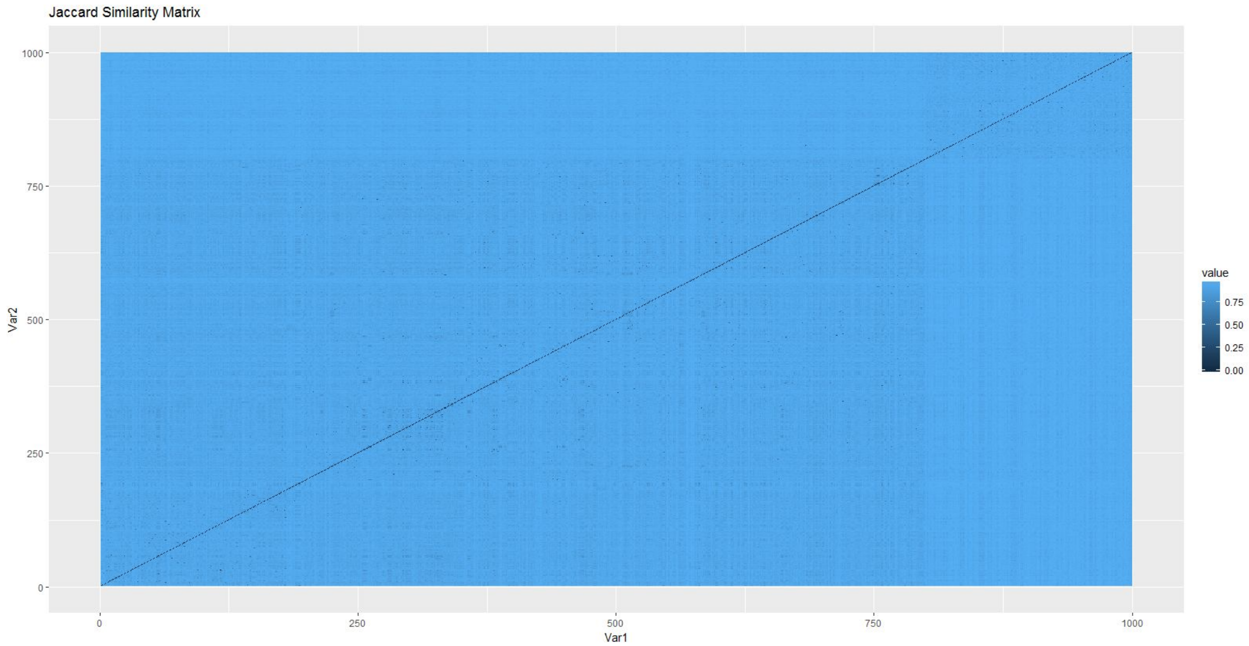
Let M denote the number of articles,

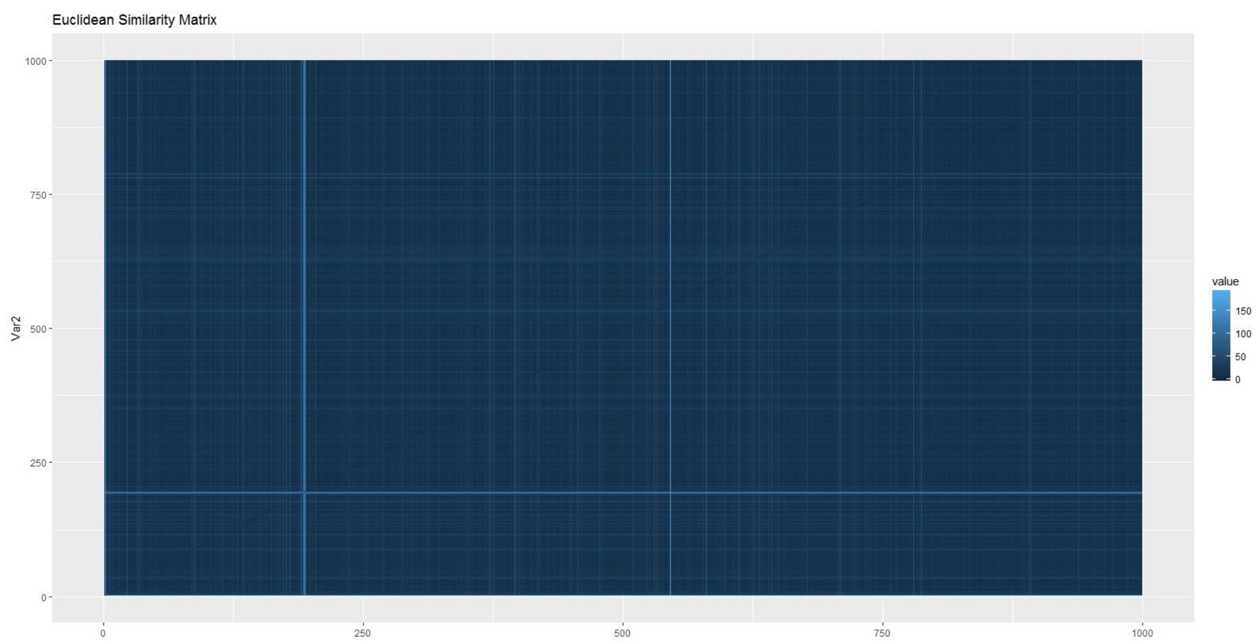
compute three pairwise similarity matrices of M articles with respect to Euclidean, Cosine, and

Jarcard. The size of this matrix is M*M. Then, plot the heatmaps of three similarity matrices. Finally, analyze whether the distances among articles are consistent.

```
#heatmap generation ggplot
ggplot(data = melt(as.matrix(m_cosdis)), aes(x=Var1, y=Var2, fill=value))+
  geom_tile() + ggtitle("Cosine Similarity Matrix")

ggplot(data = melt(as.matrix(m_jacdis)), aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()+ ggtitle("Jaccard Similarity Matrix")

ggplot(data = melt(as.matrix(m_eucdis)), aes(x=Var1, y=Var2, fill=value)) +
  geom_tile()+ ggtitle("Euclidean Similarity Matrix")
```







7. **Analyze the correlations (degree of similarity)** of Cosine-Euclidean, Euclidean-Jaccard, Jaccard - Cosine:

First, compute the Pearson correlation coefficients of each similarity pair.

```
#Analyze the correlations (degree of similarity)
Cosine_Euclidean = cor(m_eucdis,m_cosdis,method = "pearson")
Euclidean_Jarcard = cor(m_jacdis,m_eucdis,method = "pearson")
Jarcard_Cosine = cor(m_cosdis,m_jacdis,method = "pearson")
```

Output:

```
> Cosine_Euclidean = cor(m_eucdis,m_cosdis,method = "pearson")
> Cosine_Euclidean
[1] -0.04345195
> Euclidean_Jarcard = cor(m_jacdis,m_eucdis,method = "pearson")
> Euclidean_Jarcard
[1] 0.0007552901
> Jarcard_Cosine = cor(m_cosdis,m_jacdis,method = "pearson")
> Jarcard_Cosine
[1] 0.7469218
>
```

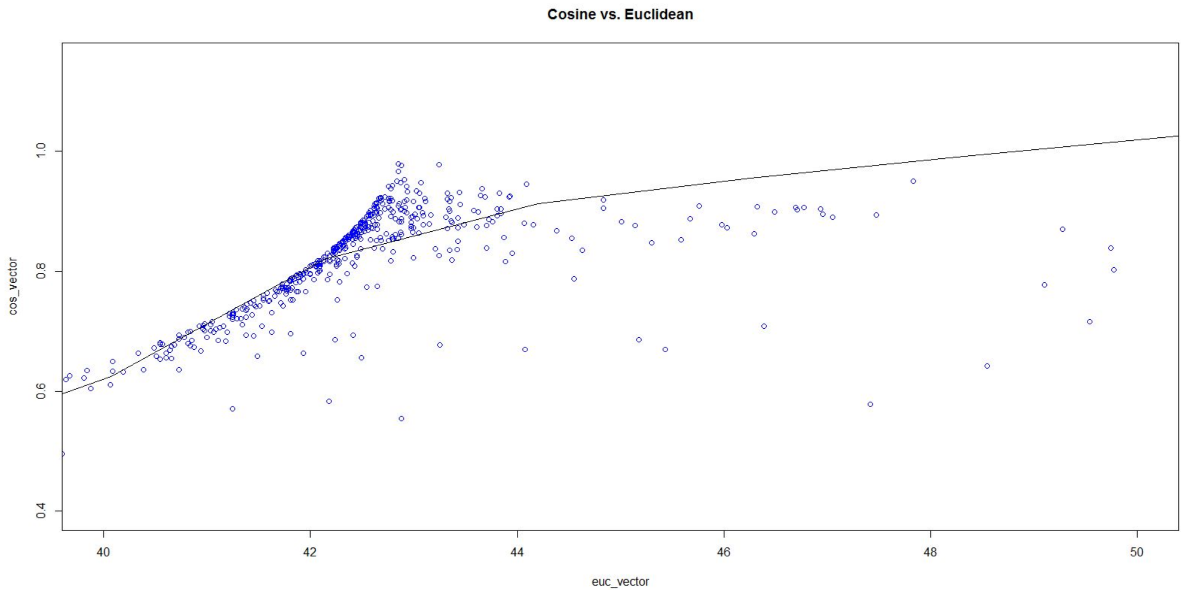
Then, use the linear regression y=ax+b to fit the three similarity pairs

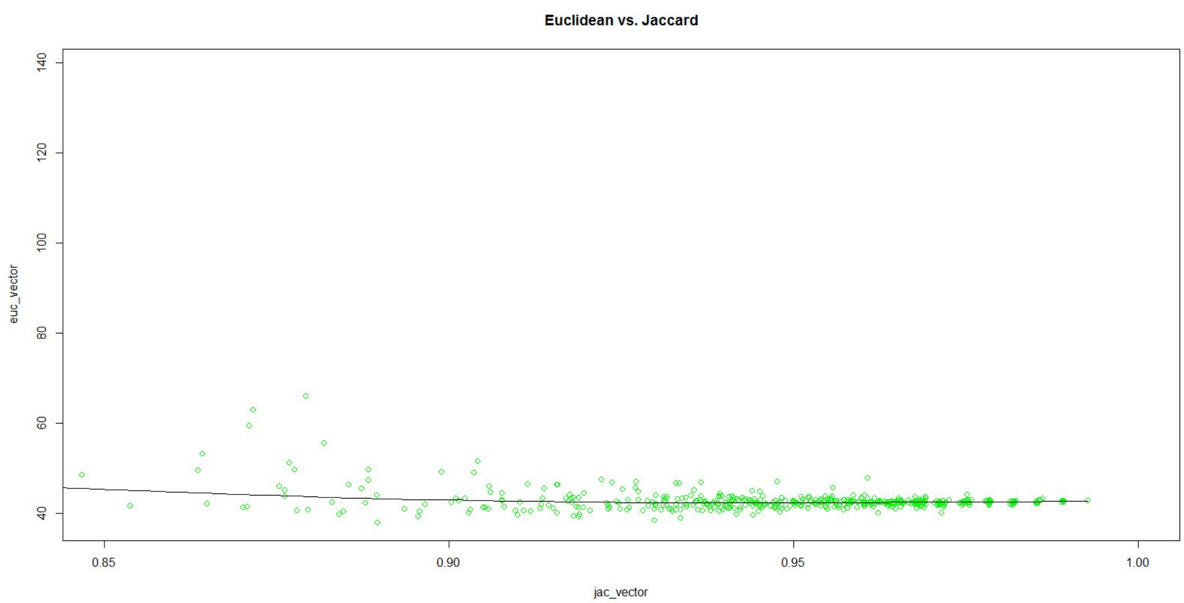
- Cosine=a*Euclidean+b,
- Euclidean=a*Jarcard+b, and
- Jarcard=a*Cosine+b.

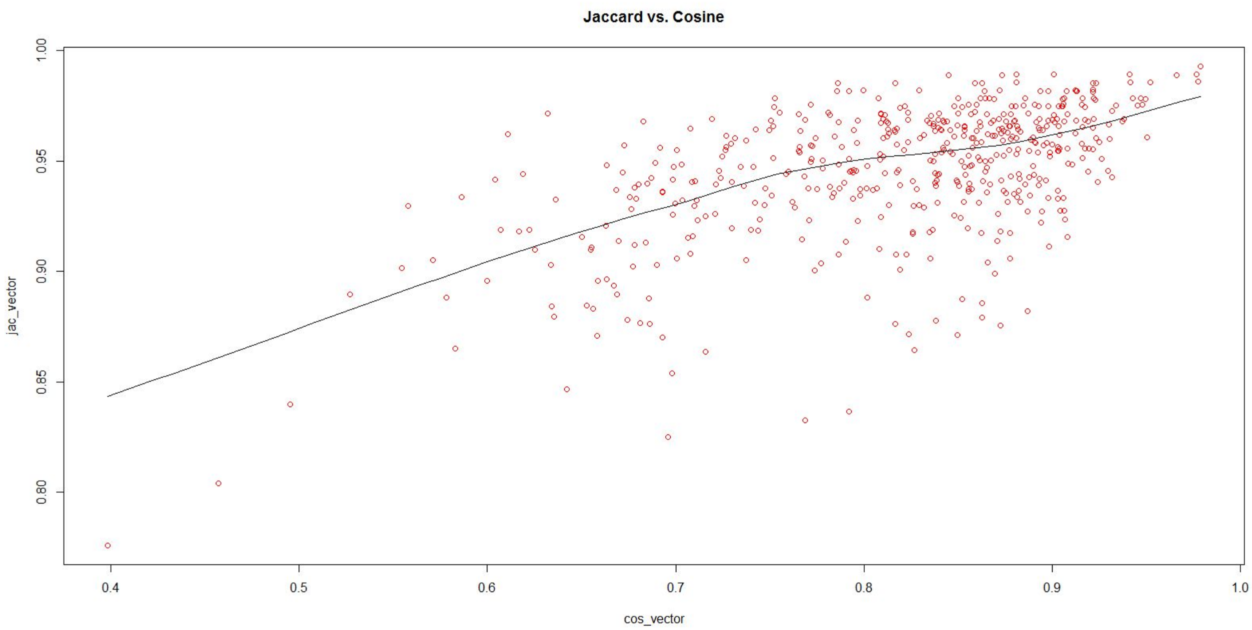
```
#linear regression
cos_vector <- as.vector(head(m_cosdis,500))
euc_vector <- as.vector(head(m_eucdis,500))
jac_vector <- as.vector(head(m_jacdis,500))

fit1 <- lm(formula=cos_vector ~ euc_vector, data=dataframe_m)
fit2 <- lm(formula=euc_vector ~ jac_vector, data=dataframe_m)
fit3 <- lm(formula=jac_vector ~ cos_vector, data=dataframe_m)
```

Finally, plot the scatter plots and the fitted lines of the three similarity pairs.

```
scatter.smooth(x=euc_vector, y=cos_vector,xlim=c(40,50),+
               main="Cosine vs. Euclidean", col='blue')+
scatter.smooth(x=jac_vector, y=euc_vector,xlim=c(0.85,1),+
              main="Euclidean vs. Jaccard", col='green')+
scatter.smooth(x=cos_vector, y=jac_vector, main="Jaccard vs. Cosine", col='red')
```

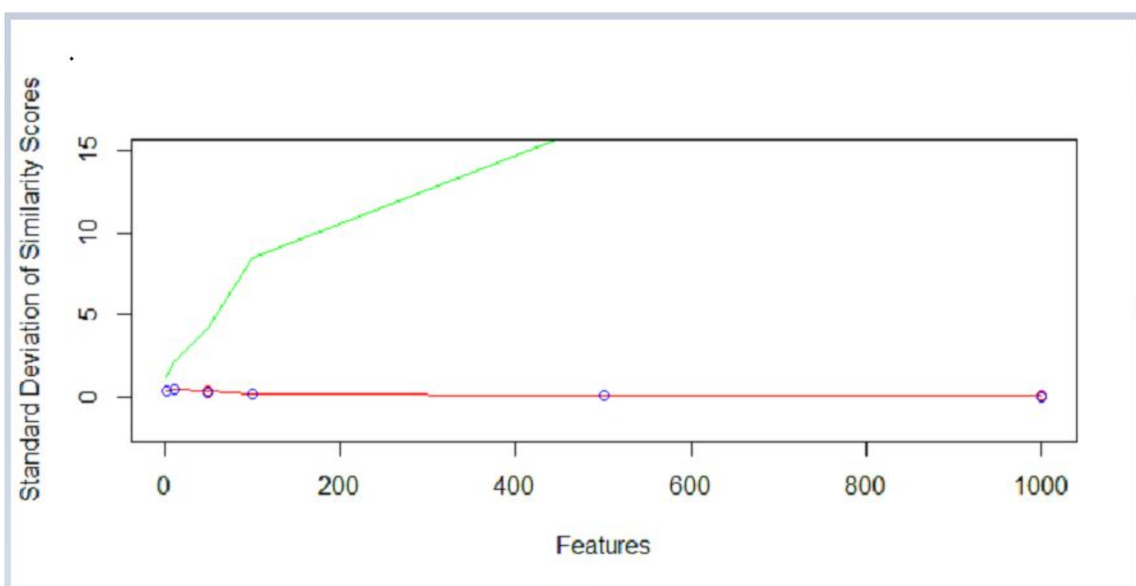

Cosine vs. Euclidean
cos_vector
euc_vector
1.0
0.8
0.6
0.4
40
42
44
46
48
50


Euclidean vs. Jaccard
euc_vector
jac_vector
140
120
100
80
60
40
0.85
0.90
0.95
1.00



8. Standard Deviation : Let N denote the number of features (i.e., terms). Analyze how the standard deviation of similarity scores changes/varies when the number of features (i.e., terms) increases from 2 to N with respect to Euclidean, Cosine, and Jaccard.

Green - Euclidean, Red - Cosine, Blue - Jaccard



9. Rank all the articles pairs in a decreasing order of similarity with respect to each metric (Euclidean, Cosine, and Jaccard) and select top 3 article pair with respect to each metric. Compare the 9 article pairs.

```
#Rank article pairs

jac_melt <- melt(as.matrix(m_jacdis))
tail(jac_melt[order(jac_melt$value),])

euc_melt <- melt(as.matrix(m_eucdis))
tail(euc_melt[order(euc_melt$value),])

cos_melt <- melt(as.matrix(m_cosdis))
tail(cos_melt[order(cos_melt$value),])
```

Output:

```
> jac_melt <- melt(as.matrix(m_jacdis))
> tail(jac_melt[order(jac_melt$value),])
       Var1 Var2     value
545814  814  546 0.9981481
545871  871  546 0.9981481
813546  546  814 0.9981481
870546  546  871 0.9981481
545822  822  546 0.9981651
821546  546  822 0.9981651
> euc_melt <- melt(as.matrix(m_eucdis))
> tail(euc_melt[order(euc_melt$value),])
       Var1 Var2    value
192546  546  193 172.0814
545193  193  546 172.0814
193546  546  194 185.1378
545194  194  546 185.1378
1546    546    2 198.2675
545002    2  546 198.2675
> cos_melt <- melt(as.matrix(m_cosdis))
> tail(cos_melt[order(cos_melt$value),])
       Var1 Var2     value
545822  822  546 0.9972595
821546  546  822 0.9972595
22873   873   23 0.9972650
872023   23  873 0.9972650
463788  788  464 0.9974851
787464  464  788 0.9974851
```

Jaccard and Cosine matrices show similar article pairs. So, we can consider them as accurate compared to Euclidean matrix.