Text mining on email data and predicting spam

true

Step 1 Load requried packages

```
suppressWarnings(library(RTextTools))
```

```
## Loading required package: SparseM
```

```
##
## Attaching package: 'SparseM'
```

```
## The following object is masked from 'package:base':
##
##     backsolve
```

```
suppressWarnings(library(tm))
```

```
## Loading required package: NLP
```

```
suppressWarnings(library(wordcloud))
```

```
## Loading required package: RColorBrewer
```

```
suppressWarnings(library(e1071))
```

Step 2 Load Email data from ham and spam folder to a data frame

```
# create ham data frame

dir="C:/Users/Arindam/Documents/Data Science/Cuny/Data 607/Assignments/Data files/easy_ham"
filename = list.files(dir)

docs<-NA
for(i in 1:length(filename))
{
  filepath<-paste0("C:/Users/Arindam/Documents/Data Science/Cuny/Data 607/Assignments/Data fi
les/easy_ham/",filename[1])
  text <-readLines(filepath)
 list1<- list(paste(text, collapse="\n"))
 docs = c(docs,list1)

}
ham<-data.frame()
ham<-as.data.frame(unlist(docs),stringsAsFactors = FALSE)
ham$score<-1
colnames(ham)<-c("text","score")


# create spam data frame

dir_spam="C:/Users/Arindam/Documents/Data Science/Cuny/Data 607/Assignments/Data files/spam"
filename_spam = list.files(dir_spam)

docs_spam<-NA

for(i in 1:length(filename_spam))
{
  filepath_spam<-paste0("C:/Users/Arindam/Documents/Data Science/Cuny/Data 607/Assignments/Da
ta files/spam/",filename_spam[1])
  text_spam <-readLines(filepath_spam)
  list1_spam<- list(paste(text_spam, collapse="\n"))
  docs_spam = c(docs_spam,list1_spam)

}
spam<-data.frame()
spam<-as.data.frame(unlist(docs_spam),stringsAsFactors = FALSE)
spam$score<-0
colnames(spam)<-c("text","score")


# creating combined data frame of spam and ham

spam_ham<-rbind(spam,ham)
```

Step 3 Create email corpus

```
 email_corpus <- Corpus(VectorSource(spam_ham$text))
#clean_corpus <- tm_map(email_corpus, tolower)
clean_corpus <- tm_map(email_corpus, removeNumbers)
clean_corpus <- tm_map(clean_corpus, removePunctuation)
clean_corpus <- tm_map(clean_corpus, removeWords, stopwords())
clean_corpus <- tm_map(clean_corpus, stripWhitespace)
```

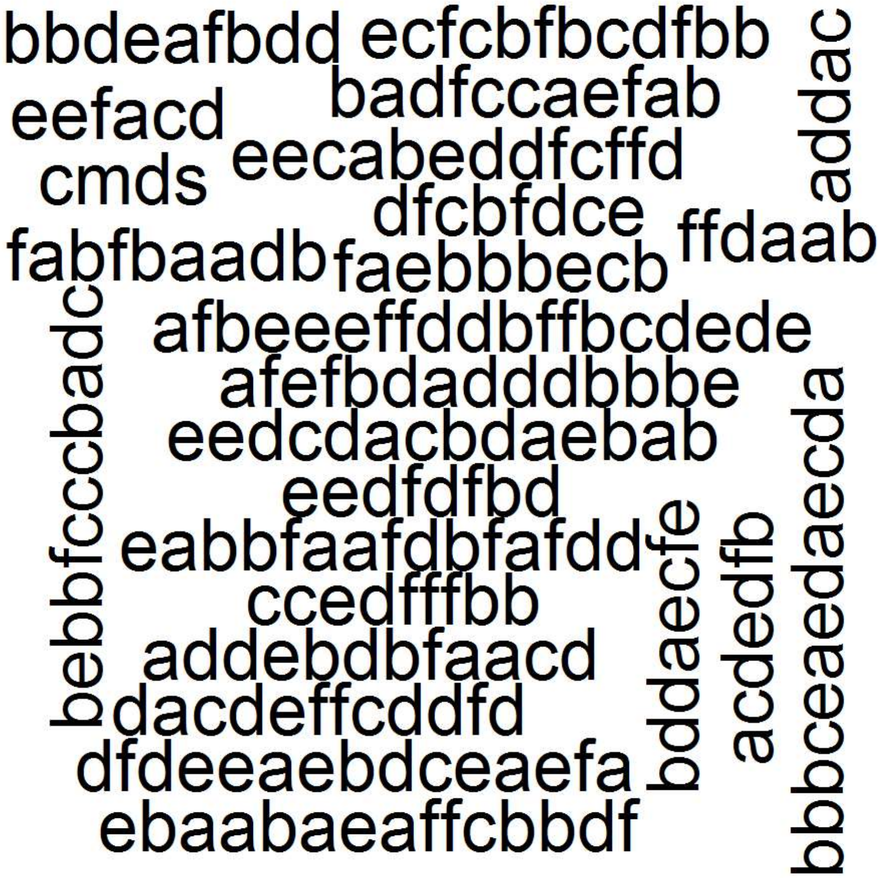
Step 4

create documenterm matrix and wordcloud for ham and spam

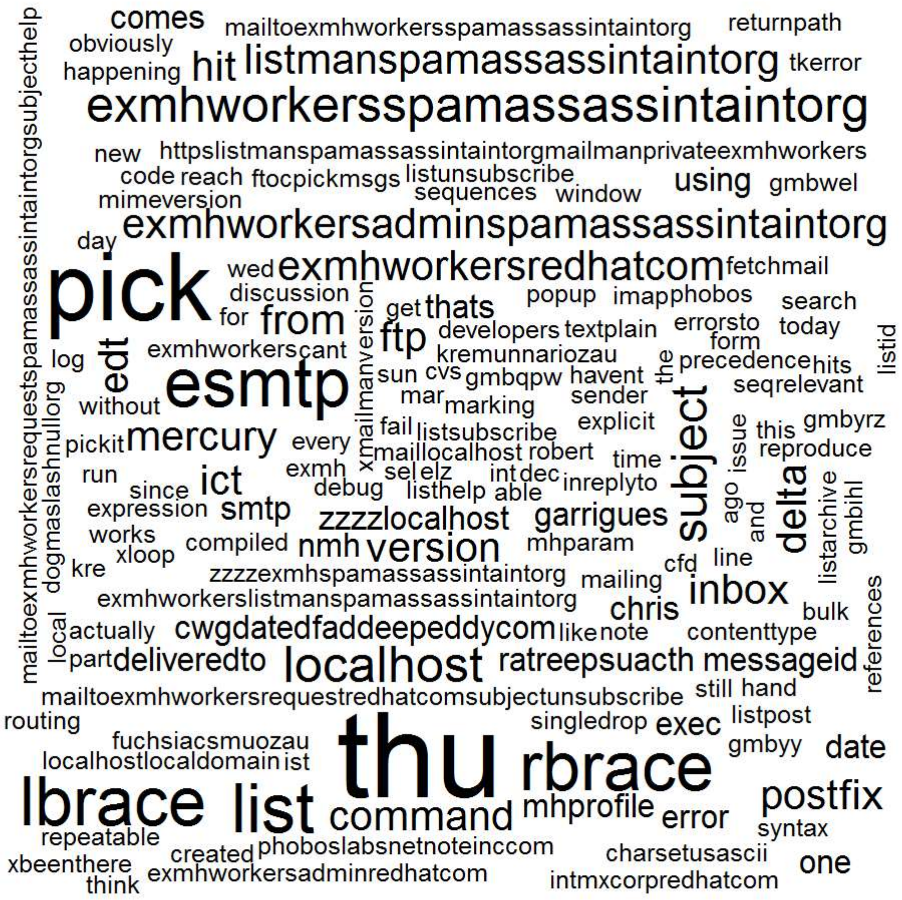
```
email_dtm <- DocumentTermMatrix(clean_corpus)

# spam word cloud


spam_indices <- which(spam_ham$score == 0)
suppressWarnings(wordcloud(clean_corpus[spam_indices], min.freq=40))
```



```
ham_indices <- which(spam_ham$score == 1)
suppressWarnings(wordcloud(clean_corpus[ham_indices], min.freq=50))
```



Step 5 creating model to assess spam and ham

```
# sample 70% data traning and 30 % for prediction

smp_size <- floor(0.75 * nrow(spam_ham))

## set the seed to make your partition reproductible
set.seed(123)
train_ind <- sample(seq_len(nrow(spam_ham)), size = smp_size)

train_spam_ham <- spam_ham[train_ind, ]
test_spam_ham <- spam_ham[-train_ind, ]


# count of spam and ham in train data set

spam<-subset(train_spam_ham,train_spam_ham$score==0)
ham<-subset(train_spam_ham,train_spam_ham$score==1)
```

Step 6 Create corpus for training and test data

```
# create corpus for train and test data set

train_email_corpus <- Corpus(VectorSource(train_spam_ham$text))
test_email_corpus <- Corpus(VectorSource(test_spam_ham$text))


#train_clean_corpus <- tm_map(train_email_corpus, tolower)
#test_clean_corpus <- tm_map(test_email_corpus, tolower)


train_clean_corpus <- tm_map(train_email_corpus ,removeNumbers)
test_clean_corpus <- tm_map(test_email_corpus, removeNumbers)



train_clean_corpus <- tm_map(train_clean_corpus, removePunctuation)
test_clean_corpus <- tm_map(test_clean_corpus, removePunctuation)


train_clean_corpus <- tm_map(train_clean_corpus, removeWords, stopwords())
test_clean_corpus  <- tm_map(test_clean_corpus, removeWords, stopwords())


train_clean_corpus<- tm_map(train_clean_corpus, stripWhitespace)
test_clean_corpus<- tm_map(test_clean_corpus, stripWhitespace)


train_sms_dtm <- DocumentTermMatrix(train_clean_corpus)
test_sms_dtm <- DocumentTermMatrix(test_clean_corpus)

# frequently used words

#five_times_words <- findFreqTerms(train_sms_dtm, 5)
#length(five_times_words)

# modify documentum matrix with frequency

#train_sms_dtm1 <- DocumentTermMatrix(train_sms_dtm, control=list(dictionary = five_times_words))

#test_sms_dtm1 <- DocumentTermMatrix(test_sms_dtm, control=list(dictionary = five_times_words))

# count function

convert_count <- function(x) {
  y <- ifelse(x > 0, 1,0)
  y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
  y
}
```

Step 7 Email classification of spam and ham emails

```
# using that to documentum matrix

train_sms <- apply(train_sms_dtm, 2, convert_count)
test_sms <- apply(test_sms_dtm, 2, convert_count)

# classification of email

classifier <- naiveBayes(train_sms, factor(train_spam_ham$score))
class(classifier)
```

```
## [1] "naiveBayes"
```

```
test_pred <- predict(classifier, newdata=test_sms)

table(test_pred, test_spam_ham$score)
```

```
##          
## test_pred   0   1
##         0 119   0
##         1   1 631
```

```
#looking at the table we can see model is very accurate and only one scenario is there where
 email was mis classified
```

reference:Following source was used as guideline for this assignment

http://www3.nd.edu/~steve/computing_with_data/20_text_mining/text_mining_example.html# (http://www3.nd.edu/~steve/computing_with_data/20_text_mining/text_mining_example.html#)