Project 4 DATA 607 (CODE USING MODEL NAIVE BAYES)

Team Project Members: Banu Boopalan, Samuel Kigamba, James Mundy, Alain T Kuiete

11/16/2019

```
library(tidyverse)
library(tidyr)
library(dplyr)
library(stringr)
library(tidytext)
library(tm)
library(SnowballC)
library(ggplot2)
library(wordcloud)
library(caret)
library(gbm)
library(e1071)
library(SparseM)
library(caTools)
library(randomForest)
#library(tree)
library(ipred)
#library(glmnet)
library(tau)
library(devtools)
#install.packages('quanteda')
library(quanteda)
```

PART 1 CODE (Find in this Part1 RPUBS link below):

Our Project Team 4 above (Banu Boopalan, Samuel Kigamba, James Mundy, Alain T Kuiete), we will submit 2 RPUB documents (RPUBS LINK PROVIDED BY EACH TEAM MEMBER). This is the first document representing the first model. In this code, we have performed data transformations, exploratory data analysis, visualizations using wordclouds, frequency plots on words, and performed Naive Bayes Model and reported the Confusion Matrix results for the Naive Bayes Model. We tried to plot the prediction model using plot and mosaicplot but we were not able draw the plot for to show the plot of the model which requires further understanding. Within the model we are able to create document term matrix, segment the train and test data and then run the model to report summary model statistics. Each team member will report a different accuracy due to the files read in.

PART 2 CODE (Submitted part of a separate RPUBS link)

Our Project Team 4 above (Banu Boopalan, Samuel Kigamba, James Mundy, Alain T Kuiete), we will submit 2 separate RPUB documents. The 2nd document link to RPUBS, we have performed data transformations, exploratory data analysis, visualizations using wordclouds, frequency plots on words, and performed SVM model and reported the Confusion Matrix results for the SVM model. We tried to plot the model using plot but we were not successful in representing a way to plot the model, The support vector #'s are high range

so we have to dive deeper into how to represent and plot the model through plot or Kernlab pacakge or Kernfit. Within the model we are able to create document term matrix and term document matrix, segment the train and test data and then run the model to report summary model. The SVM reported an accuracy for each of our teammates will be different as we are reading in our own files from the directory. The SVM reported higher accuracy than the Naive Bayes upon first review.

Collaboration via POWERPOINT, GITHUB, GOTO MEETING along with weekly meetings on Tuesday, Friday.

Section : Ham files

Downloading the Dataset for Ham

Creating Ham Data Frame

```
#ham.dir="C:\\DATA607\\Project4\\spamHam\\20021010_easy_ham (1).tar\\easy_ham"
#ham.dir="C://Users//Banu//Documents//RScriptfiles//Project4//SpamHam//easyham//20030228_easy_ham//easy
ham.dir="easy_ham"
ham.file.names = list.files(ham.dir)

str(ham.file.names)
```

```
##  chr [1:4045] "00001 (1).7c53336b37003a9286aba55d2945844c" ...
```

```
ham.file.names[1:15]
```

```
##  [1] "00001 (1).7c53336b37003a9286aba55d2945844c"
##  [2] "00001.7c53336b37003a9286aba55d2945844c"
##  [3] "00002 (1).9c4069e25e1ef370c078db7ee85ff9ac"
##  [4] "00002.9c4069e25e1ef370c078db7ee85ff9ac"
##  [5] "00003 (1).860e3c3cee1b42ead714c5c874fe25f7"
##  [6] "00003.860e3c3cee1b42ead714c5c874fe25f7"
##  [7] "00004 (1).864220c5b6930b209cc287c361c99af1"
##  [8] "00004.864220c5b6930b209cc287c361c99af1"
##  [9] "00005.bf27cdeaf0b8c4647ecd61b1d09da613"
## [10] "00006 (1).253ea2f9a9cc36fa0b1129b04b806608"
## [11] "00006.253ea2f9a9cc36fa0b1129b04b806608"
## [12] "00007 (1).37a8af848caae585af4fe35779656d55"
## [13] "00007.37a8af848caae585af4fe35779656d55"
## [14] "00008 (1).5891548d921601906337dcf1ed8543cb"
## [15] "00008.5891548d921601906337dcf1ed8543cb"
```

```
ham_files = list.files(path = ham.dir, full.names = TRUE)
no_of_ham_files = length(list.files(ham.dir, all.files = "FALSE", full.names = "TRUE"))
print(paste("There are",no_of_ham_files,"spam files in the easy_ham folder."))
```

```
## [1] "There are 4045 spam files in the easy_ham folder."
```

```
#ham_files

# List of docs
ham.docs <- ham.file.names[1]
for(i in 2:length(ham.file.names))
{
  filepath<-paste0(ham.dir, "/", ham.file.names[i])
  text <-readLines(filepath)
  list1<- list(paste(text, collapse="\n"))
  ham.docs = c(ham.docs,list1)
}
#head(ham.docs, 2)
```

Extracting the Ham senders emails

```
senders <- unlist(str_extract(ham.docs[2], "(?<name>[\\w.-]+)\\@(?<domain>[-\\w+\\.\\w+]+)(\\.\\w+)?"))
for (i  in 3:length(ham.docs)) {
  s <- unlist(str_extract(ham.docs[i],"(?<name>[\\w.-]+)\\@(?<domain>[-\\w+\\.\\w+]+)(\\.\\w+)?"))
  senders <- c(senders, s)
}
summary(senders)
```

```
##    Length     Class      Mode
##      4044 character character
```

```
head(senders, 2)
```

```
## [1] "exmh-workers-admin@redhat.com" "Steve_Burt@cursor-system.com"
```

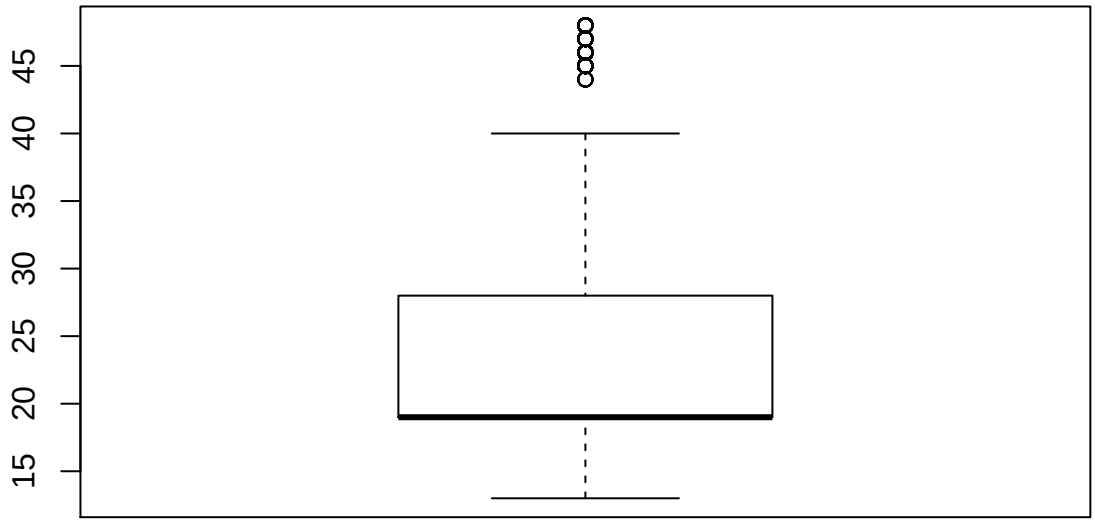
Creating a Ham Sender' Email Data Frame

```
email.length <- nchar(senders[1])
for (i in 2:length(senders)) {
  email.length <-c(email.length,nchar(senders[i]))
}
sender.df <- tibble(email=senders, length=email.length)
head(sender.df, 2)
```

```
## # A tibble: 2 x 2
##   email                         length
##   <chr>                          <int>
## 1 exmh-workers-admin@redhat.com     29
## 2 Steve_Burt@cursor-system.com      28
```

vizualizing the Length of Different Senders' Emails

```r
boxplot(sender.df$length)
```



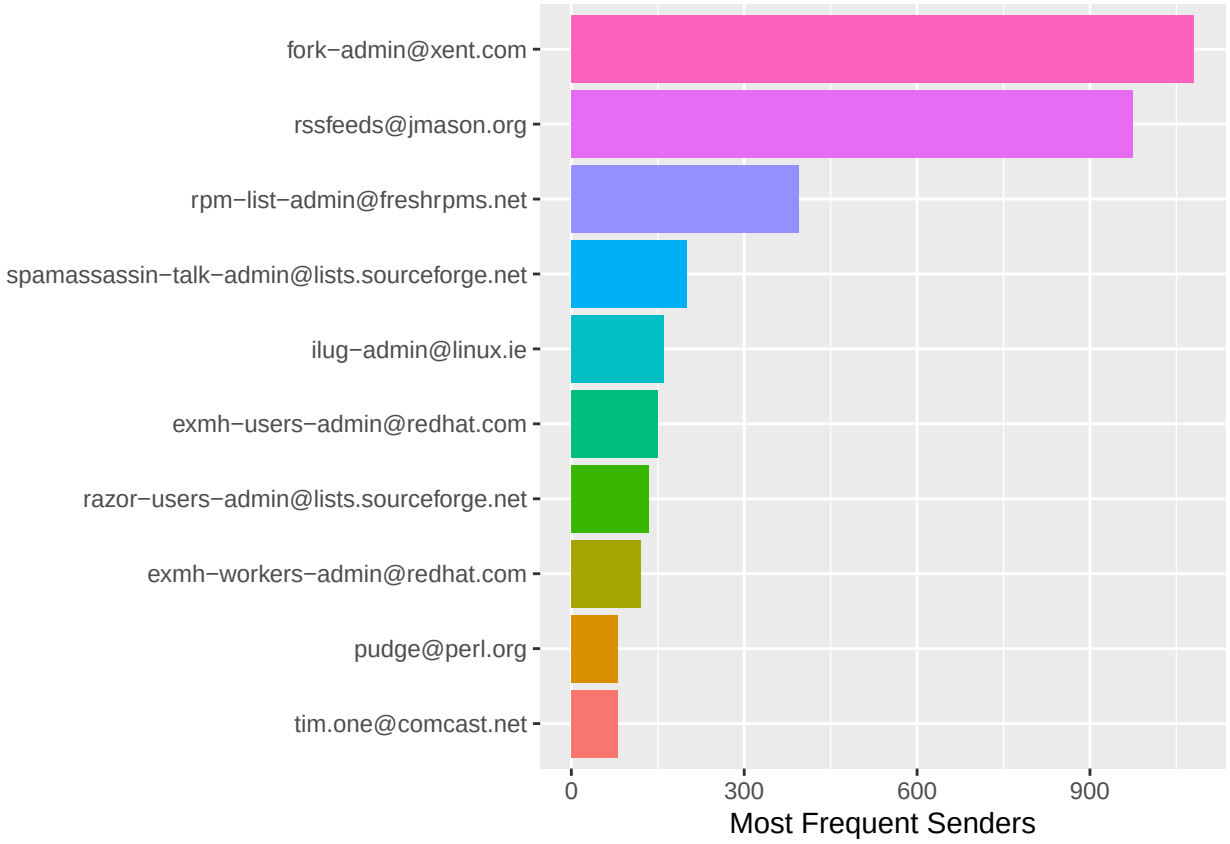
Grouping the Senders' emails by email address

```r
sen.email <- sender.df %>%
  group_by( new.email =email, length)%>%
  summarise(n=n())%>%
  arrange(desc(n))
```

visualizing the 10 most frequent Emails Ham

```r
sender.df %>%
  group_by(email) %>%
  summarise(n=n())%>%
  top_n(10)%>%
  mutate(email = reorder(email, n)) %>%
  ggplot(aes(email, n, fill = email)) +
  geom_col(show.legend = FALSE)  +
  labs(y = "Most Frequent Senders",
       x = NULL) +
  coord_flip()
```

```
## Selecting by n
```



Example of a Ham File

```
ham.docs[4]
```

```
## [[1]]
## [1] "From Steve_Burt@cursor-system.com  Thu Aug 22 12:46:39 2002\nReturn-Path: <Steve_Burt@cursor-sys
```

Using Regular Expressions to extract all the emails in the Ham Files

```
emails <- unlist(str_extract_all(ham.docs[2],"(?<name>[\\w.-]+)\\@(?<domain>[-\\w+\\.\\w+]+)(\\.\\w+)?"
for (i  in 3:length(ham.docs)) {
  s <- unlist(str_extract_all(ham.docs[i],"(?<name>[\\w.-]+)\\@(?<domain>[-\\w+\\.\\w+]+)(\\.\\w+)?"))
  emails <- c(emails, s)
}
summary(emails)
```

```
##    Length     Class      Mode
##     73597 character character
```

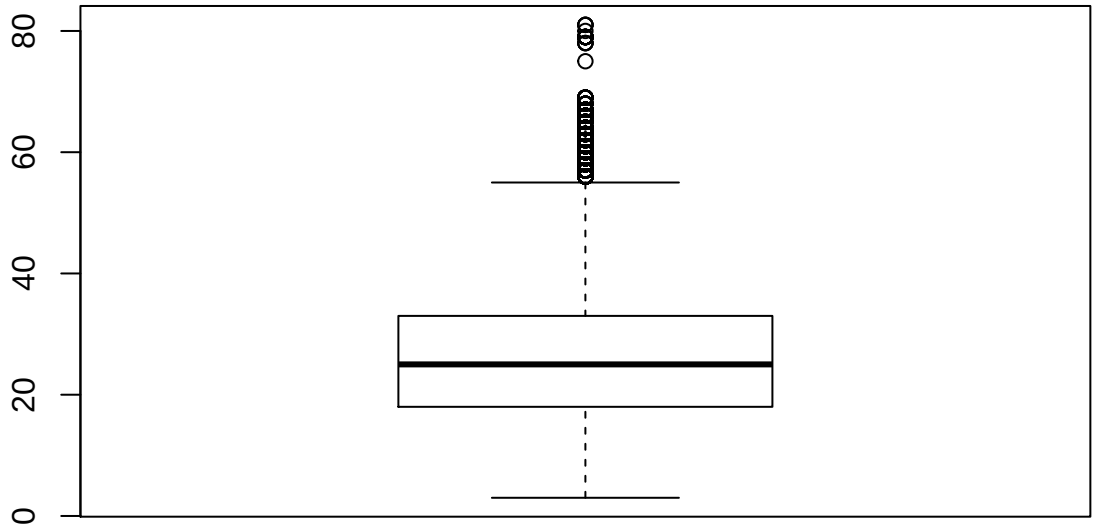
Turning These Ham Emails to a Data Frame

```
len <- nchar(emails[1])
for (i in 2:length(emails)) {
  len <-c(len, nchar(emails[i]))
}
ham.emails <- tibble(mail = 1:length(emails), emails, len)
head(ham.emails, 2)
```

```
## # A tibble: 2 x 3
##    mail emails                                        len
##   <int> <chr>                                       <int>
## 1     1 exmh-workers-admin@redhat.com                  29
## 2     2 exmh-workers-admin@spamassassin.taint.org      41
```

visualizing the length of all Emails

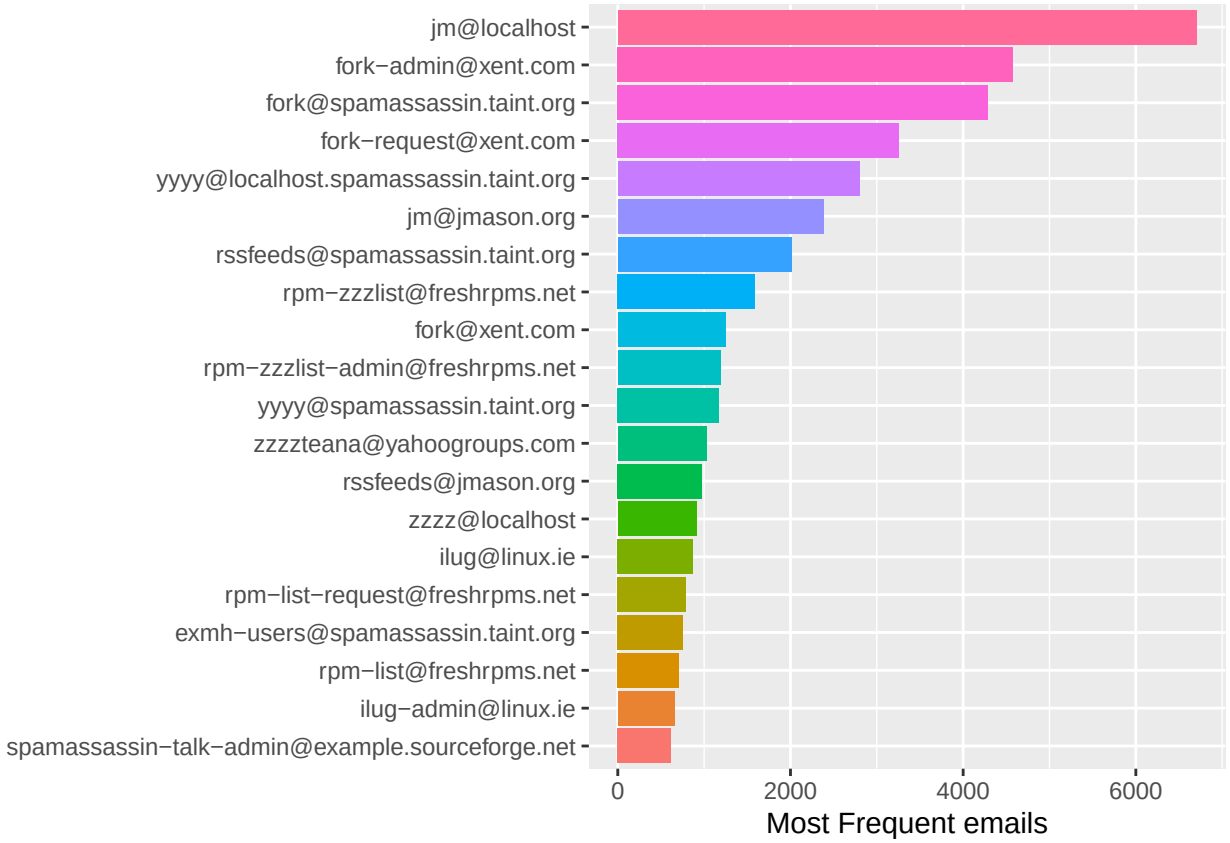
```
boxplot(ham.emails$len)
```



visualizing the 20 Most Frequent Emails

```
ham.emails %>%
  group_by(emails) %>%
  summarise(n=n())%>%
  top_n(20)%>%
  mutate(emails = reorder(emails, n)) %>%
  ggplot(aes(emails, n, fill = emails)) +
  geom_col(show.legend = FALSE)  +
  labs(y = "Most Frequent emails",
       x = NULL) +
  coord_flip()
```

```
## Selecting by n
```



Body of the Email

Extracting words in the Bodies of All Emails

```
#ham.emails <- ham.emails %>%
  #unnest_tokens(word, text)%>%
  #group_by(text) %>%
```

```
  #mutate(n= n()) %>%
  #ungroup()
#ham.emails
```

Creating a Data Frame containing the words

```
ham.list <- tibble(files = 1:length(ham.docs),
                   text = ham.docs)
```

Adding the Frequency of Words to the Data frame

```
ham.block <- ham.list %>%
  unnest_tokens(word, text)%>%
  group_by(files) %>%
  mutate(n= n()) %>%
  ungroup()
head(ham.block, 2)
```

```
## # A tibble: 2 x 3
##   files word      n
##   <int> <chr> <int>
## 1     1 00001     3
## 2     1 1         3
```

Organizing the Data frame and adding the Term Frequency(tf), Inverse Document Frequency of a term(idf), and the combining of two term(tf_idf)

```
ham.block <- ham.block %>%
  bind_tf_idf(word, files, n)
```

```
## Warning in bind_tf_idf.data.frame(., word, files, n): A value for tf_idf is negative:
## Input should have exactly one row per document-term combination.
```

```
ham.block <- ham.block %>%
  arrange(desc(tf_idf))
head(ham.block, 2)
```

```
## # A tibble: 2 x 6
##   files word                                n    tf   idf tf_idf
##   <int> <chr>                           <int> <dbl> <dbl>  <dbl>
## 1     1 00001                               3 0.333  8.31   2.77
## 2     1 7c53336b37003a9286aba55d2945844c     3 0.333  8.31   2.77
```

Cleaning the Data Frame,

We select only words with IDF greater than 0 and we remove words containing numbers

```
ham.block2 <- ham.block %>%
  filter(idf>0,str_detect(word,"([^\\d.+\\w.+\\.\\,.+]+?)")) %>%
  arrange(desc(tf_idf))
head(ham.block2, 2)
```

```
## # A tibble: 2 x 6
##   files word         n     tf   idf tf_idf
##   <int> <chr>    <int>  <dbl> <dbl>  <dbl>
## 1  2743 laptop's    60 0.0167  6.36  0.106
## 2  2744 laptop's    60 0.0167  6.36  0.106
```

Example of the sparcity of a word

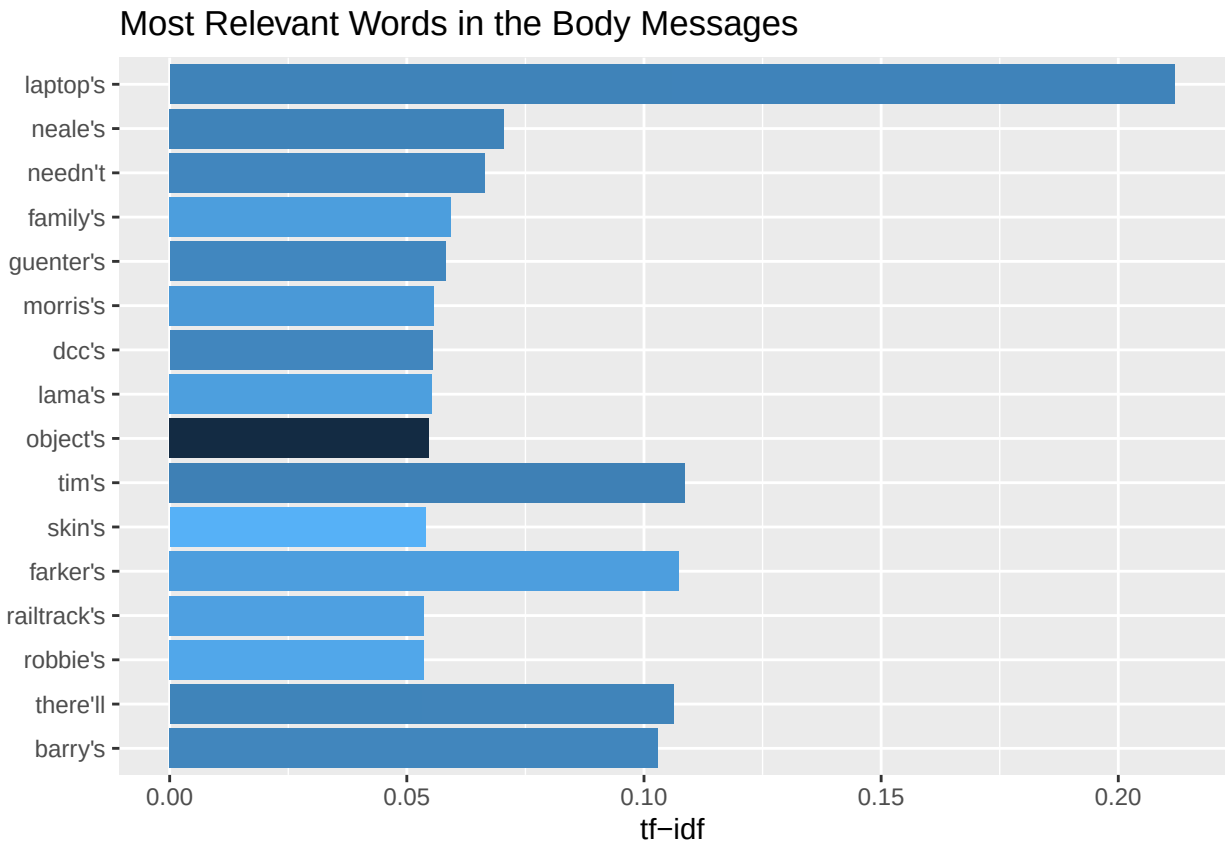
```
filter(ham.block2, word=="laptop's")
```

```
## # A tibble: 7 x 6
##   files word         n      tf   idf  tf_idf
##   <int> <chr>    <int>   <dbl> <dbl>   <dbl>
## 1  2743 laptop's    60 0.0167   6.36 0.106
## 2  2744 laptop's    60 0.0167   6.36 0.106
## 3  1966 laptop's   603 0.00166  6.36 0.0105
## 4  1967 laptop's   603 0.00166  6.36 0.0105
## 5  2026 laptop's   627 0.00159  6.36 0.0101
## 6  1968 laptop's   805 0.00124  6.36 0.00790
## 7  1969 laptop's   805 0.00124  6.36 0.00790
```

Visualization of the 20 Most Relevant Words in the Bodies of Emails

```
ham.block2%>%
  arrange(desc(tf_idf)) %>%
  top_n(20)%>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  ggplot(aes(word, tf_idf, fill = files)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf", title = "Most Relevant Words in the Body Messages") +
  coord_flip()
```

```
## Selecting by tf_idf
```


Most Relevant Words in the Body Messages
laptop's
neale's
needn't
family's
guenter's
morris's
dcc's
lama's
object's
tim's
skin's
farker's
railtrack's
robbie's
there'll
barry's
0.00
0.05
0.10
0.15
0.20
tf−idf

Section 2: Spam Files

Loading the Spam files

```
#spam.dir="C:\\DATA607\\Project4\\spamHam\\20021010_spam.tar\\spam"
#spam.dir="C://Users//Banu//Documents//RScriptfiles//Project4//SpamHam//20050311_spam_2.tar//spam_2"
spam.dir="spam_2"
spam.file.names = list.files(spam.dir)

spam_files = list.files(path = ham.dir, full.names = TRUE)
no_of_spam_files = length(list.files(spam.dir, all.files = "FALSE", full.names = "TRUE"))
print(paste("There are",no_of_spam_files,"spam emails in the spam_2 folder"))
```

```
## [1] "There are 1397 spam emails in the spam_2 folder"
```

```
#spam_files

# List of docs
spam.docs <- spam.file.names[1]
for(i in 2:length(spam.file.names))
{
  filepath<-paste0(spam.dir, "\\", spam.file.names[i])
  text <-readLines(filepath)
  l<- list(paste(text, collapse="\n"))
  spam.docs = c(spam.docs,l)
}
```

Example of a Spam Document

```
spam.docs[7]
```

```
## [[1]]
## [1] "From sales@outsrc-em.com  Mon Jun 24 17:53:15 2002\nReturn-Path: sales@outsrc-em.com\nDelivery-
```

Creating Spam Dataframe

```
spam.list <- tibble(block = 1:length(spam.docs),
                    text = spam.docs)
```

Extracting Word from The Bodies of Spam Files

```
spam.block <- spam.list %>%
  unnest_tokens(word, text)%>%
  group_by(block) %>%
  mutate(n= n()) %>%
  ungroup()
```

Selecting the Most Frequent Words with TF_IDF

```
spam.block <- spam.block %>%
  bind_tf_idf(word, block, n)
```

```
## Warning in bind_tf_idf.data.frame(., word, block, n): A value for tf_idf is negative:
## Input should have exactly one row per document-term combination.
```

```
spam.block <- spam.block %>%
  arrange(desc(tf_idf))
head(spam.block)
```

```
## # A tibble: 6 x 6
##   block word                                      n      tf   idf tf_idf
##   <int> <chr>                                 <int>   <dbl> <dbl>  <dbl>
## 1     1 00001.317e78fa8ee2f54cd4890fdc09ba8176     1 1        6.14 6.14
## 2   805 4.21.157.32                            109 0.00917  7.24 0.0664
## 3   805 g6l6w9415993                           109 0.00917  7.24 0.0664
## 4   805 1027225826.1122                        109 0.00917  7.24 0.0664
## 5   805 winnereritmugu                         109 0.00917  7.24 0.0664
## 6   805 winnergkrsvyyyyl                       109 0.00917  7.24 0.0664
```

Cleaning The Spam List of Words

```
spam.block2 <- spam.block %>%
  filter(idf>0,str_detect(word,"([^\\d.+\\w.+\\.\\,.+]+?)")) %>%
  arrange(desc(tf_idf))
head(spam.block2)
```

```
## # A tibble: 6 x 6
##   block word                    n      tf   idf tf_idf
##   <int> <chr>               <int>   <dbl> <dbl>  <dbl>
## 1   743 luke's                127 0.00787  7.24 0.0570
## 2    58 mailto:angie_pepi     192 0.00521  7.24 0.0377
## 3   382 car's                 195 0.00513  7.24 0.0371
## 4   996 mailto:remove_me123   196 0.00510  7.24 0.0369
## 5   536 ident:nobody          125 0.008    4.53 0.0363
## 6   362 mailto:bluejo         202 0.00495  7.24 0.0359
```

Creating a Spam Sender' Email Data Frame

```
spam.senders <- unlist(str_extract(spam.docs[2], "(?<name>[\\w.-]+)\\@(?<domain>[-\\w+\\.\\w+]+)(\\.\\w
for (i  in 3:length(spam.docs)) {
  s <- unlist(str_extract(spam.docs[i],"(?<name>[\\w.-]+)\\@(?<domain>[-\\w+\\.\\w+]+)(\\.\\w+)?"))
  spam.senders <- c(spam.senders, s)
}
summary(spam.senders)
```

```
##    Length     Class      Mode
##      1396 character character
```

```
head(spam.senders)
```

```
## [1] "lmrn@mailexcite.com"               "amknight@mailexcite.com"
## [3] "jordan23@mailexcite.com"           "merchantsworld2001@juno.com"
## [5] "cypherpunks-forward@ds.pro-ns.net" "sales@outsrc-em.com"
```

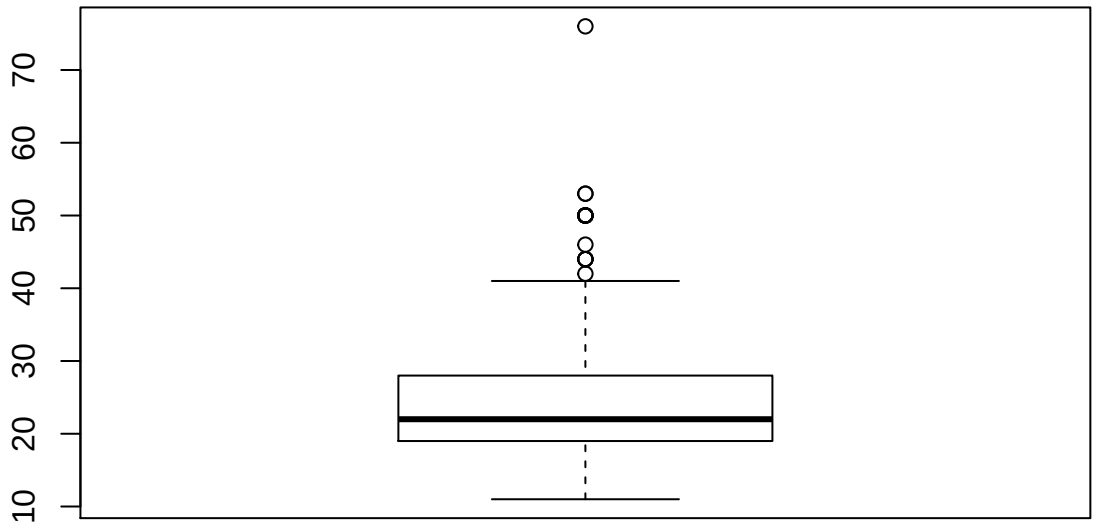
Creating a Spam Senders' Email Data Frame

```
spam.email.len <- nchar(spam.senders[1])
for (i in 2:length(spam.senders)) {
  spam.email.len <-c(spam.email.len,nchar(spam.senders[i]))
}
spam.sender.df <- tibble(email=spam.senders, len=spam.email.len)
head(spam.sender.df)
```

```
## # A tibble: 6 x 2
##   email                              len
##   <chr>                            <int>
## 1 lmrn@mailexcite.com                 19
## 2 amknight@mailexcite.com             23
## 3 jordan23@mailexcite.com             23
## 4 merchantsworld2001@juno.com         27
## 5 cypherpunks-forward@ds.pro-ns.net   33
## 6 sales@outsrc-em.com                 19
```

visualizing the Length of Different Spam Senders' Emails

```
boxplot(spam.sender.df$len)
```



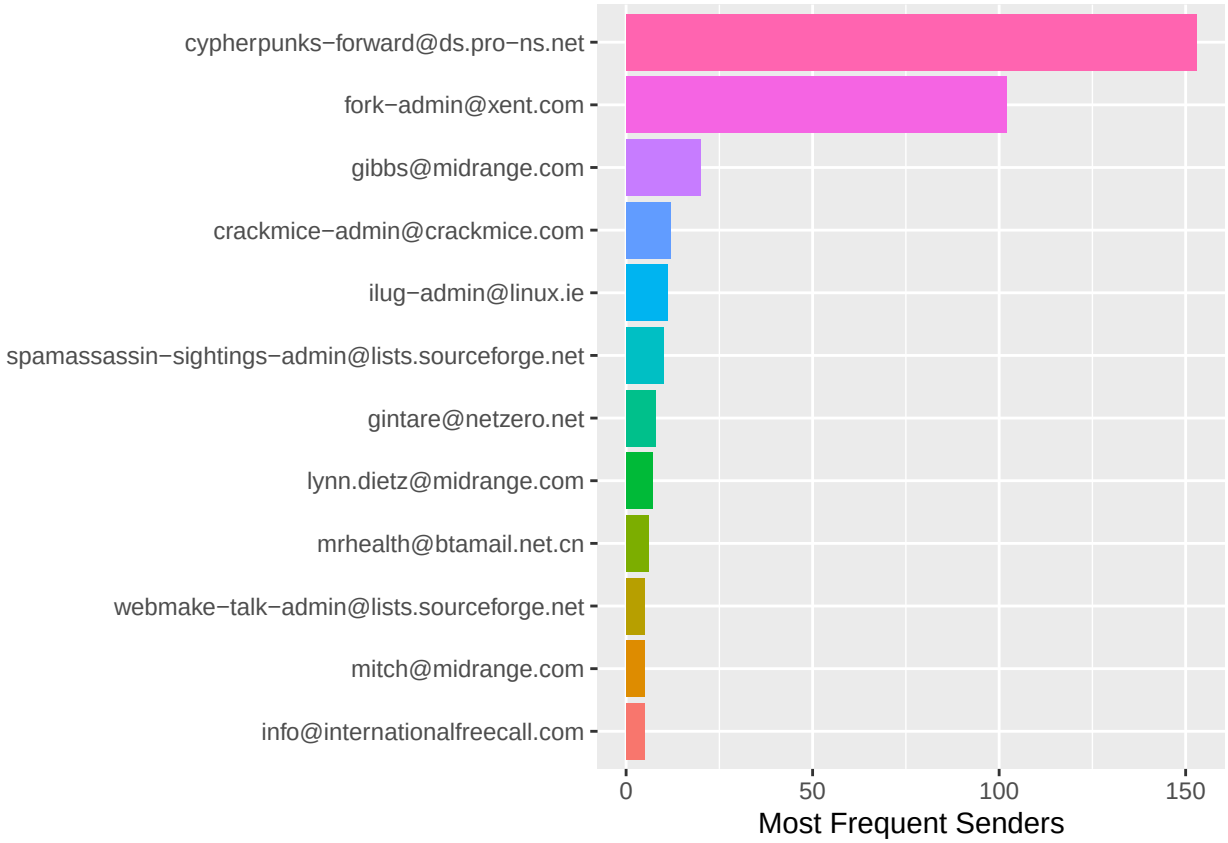
Grouping the Spam Senders' emails by email address

```
spam.sen.email <- spam.sender.df %>%
  group_by( new.email =email, len)%>%
  summarise(n=n())%>%
  arrange(desc(n))
```

visualizing the 10 Most Relevant Spam Senders' Emails

```
spam.sender.df %>%
  group_by(email) %>%
  summarise(n=n())%>%
  top_n(10)%>%
  mutate(email = reorder(email, n)) %>%
  ggplot(aes(email, n, fill = email)) +
  geom_col(show.legend = FALSE)  +
  labs(y = "Most Frequent Senders",
       x = NULL) +
  coord_flip()
```

```
## Selecting by n
```



Example of Spam Document

```
spam.docs[2]
```

```
## [[1]]
## [1] "From lmrn@mailexcite.com  Mon Jun 24 17:03:24 2002\nReturn-Path: merchantsworld2001@juno.com\nD
```

```
spam.emails <- unlist(str_extract_all(spam.docs[2],"(?<name>[\\w.-]+)\\@(?<domain>[-\\w+\\.\\w+]+)(\\.\\
for (i  in 3:length(spam.docs)) {
  s <- unlist(str_extract_all(spam.docs[i],"(?<name>[\\w.-]+)\\@(?<domain>[-\\w+\\.\\w+]+)(\\.\\w+)?"))
  spam.emails <- c(spam.emails, s)
}
summary(spam.emails)
```

```
##    Length     Class      Mode
##     22103 character character
```

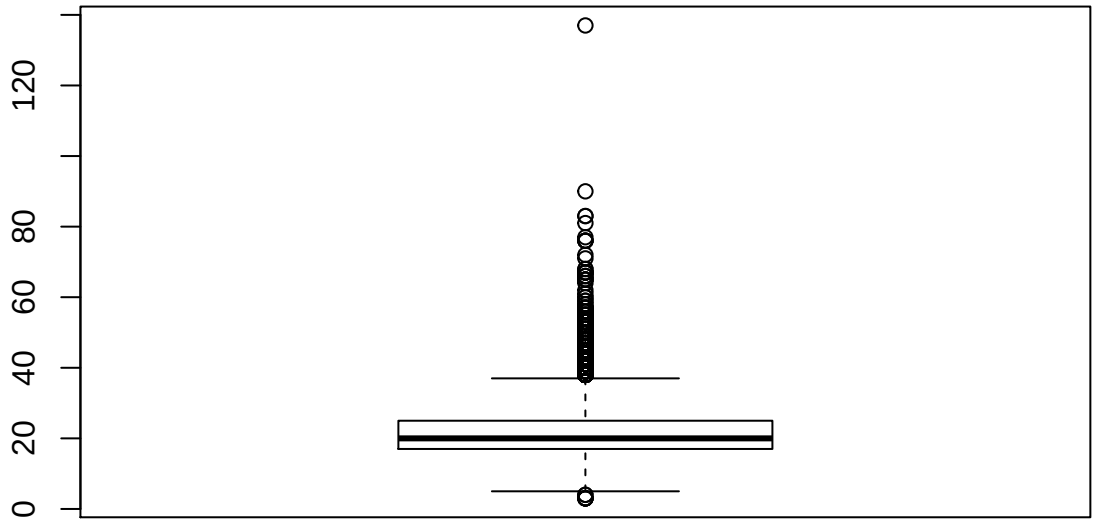
visualizing the Length of Different Senders' Emails

```
len <- nchar(spam.emails[1])
for (i in 2:length(spam.emails)) {
```

```
  len <-c(len, nchar(spam.emails[i]))
}
spam.emails <- tibble(mail = 1:length(spam.emails), spam.emails, len)
head(spam.emails)
```

```
## # A tibble: 6 x 3
##    mail spam.emails                                              len
##   <int> <chr>                                                  <int>
## 1     1 lmrn@mailexcite.com                                       19
## 2     2 merchantsworld2001@juno.com                               27
## 3     3 jm@jmason.org                                             13
## 4     4 jm@netnoteinc.com                                         17
## 5     5 B0000178595@203.129.205.5.205.129.203.in-addr.arpa        50
## 6     6 B0000178595@203.129.205.5.205.129.203.in-addr.arpa        50
```

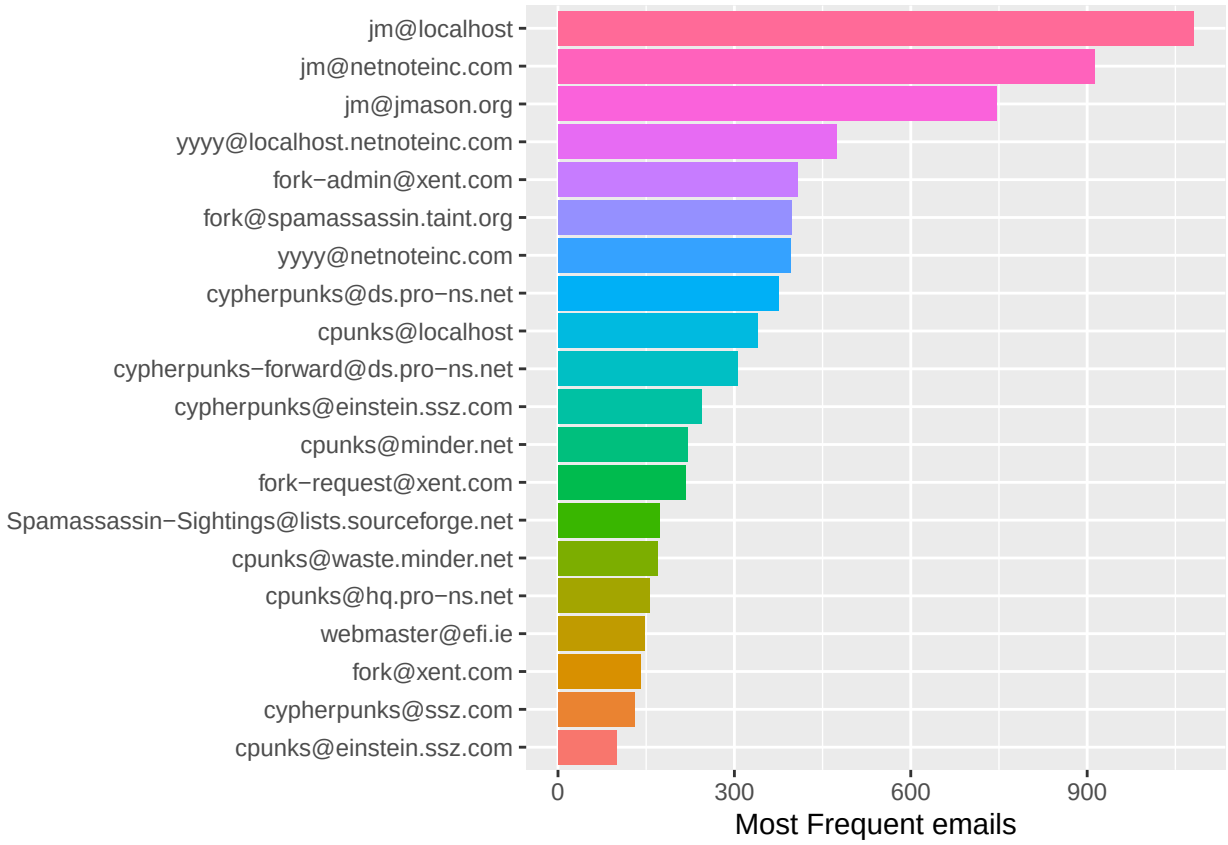
```
boxplot(spam.emails$len)
```




```
spam.emails %>%
  group_by(spam.emails) %>%
  summarise(n=n())%>%
  top_n(20)%>%
  mutate(spam.emails = reorder(spam.emails, n)) %>%
  ggplot(aes(spam.emails, n, fill = spam.emails)) +
  geom_col(show.legend = FALSE)  +
```

```
  labs(y = "Most Frequent emails",
       x = NULL) +
  coord_flip()
```

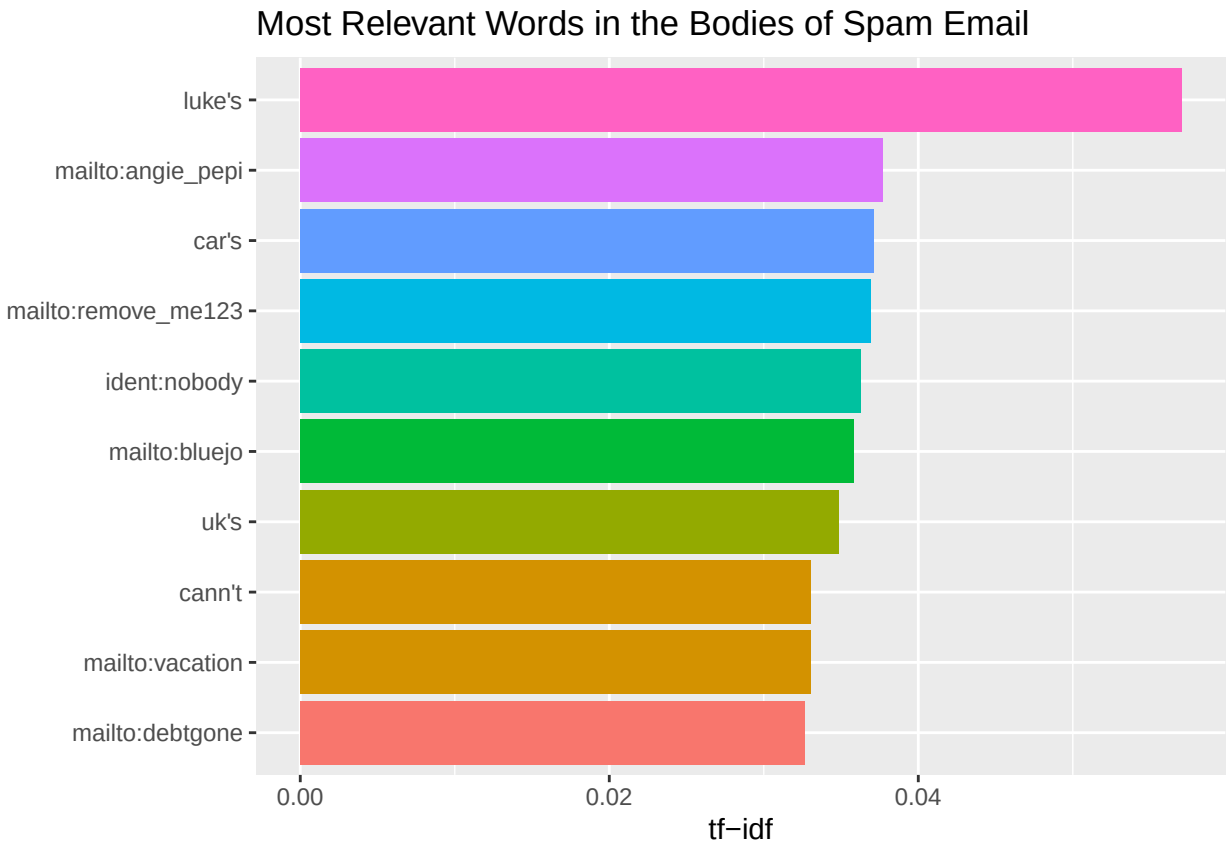
```
## Selecting by n
```



Visualization of the 10 Most Relevant Words in the Bodies of Spam Emails

```
spam.block2%>%
  top_n(10)%>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  mutate(block = reorder(block, tf_idf)) %>%
  arrange(desc(tf_idf)) %>%
  ggplot(aes(word, tf_idf, fill = block)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf", title = "Most Relevant Words in the Bodies of Spam Email") +
  coord_flip()
```

```
## Selecting by tf_idf
```


Most Relevant Words in the Bodies of Spam Email
luke's
mailto:angie_pepi
car's
mailto:remove_me123
ident:nobody
mailto:bluejo
uk's
cann't
mailto:vacation
mailto:debtgone
0.00
0.02
0.04
tf−idf

Spam Ham classification using Naivebayes Classifier

We create an object/model that can loop through any list of documents and create a corpus for each. This way we avoid duplicating this code for each and every set of documents that we need to loop through.

```
to_VCorpus <- function(file_path) {
  corpus <- file_path %>%
    paste(., list.files(.), sep = "/") %>%
    lapply(readLines) %>%
    VectorSource() %>%
    VCorpus()
}

docmnt_clean <- function(corpus) {
  corpus <- corpus %>%
    tm_map(removeNumbers) %>%
    tm_map(removePunctuation) %>%
    tm_map(tolower) %>%
    tm_map(PlainTextDocument) %>%
    tm_map(removeWords, stopwords("en")) %>%
    tm_map(stripWhitespace) %>%
    tm_map(stemDocument)
return(corpus)
}

addTag <- function(corpus, tag, value) {
  for (i in 1:length(corpus)){
    meta(corpus[[i]], tag) <- value
  }
  return(corpus)
}
```

Create a corpus for each of the two email classification using the object model above

```
#Ham
Ham_Corpus <- ham.dir %>%
  to_VCorpus %>%
  docmnt_clean %>%
  addTag(tag = "emails", value = "ham")

inspect(Ham_Corpus[1:5])
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 5
##
## [[1]]
## <<PlainTextDocument>>
## Metadata:  8
## Content:  chars: 3048
##
```

```
## [[2]]
## <<PlainTextDocument>>
## Metadata:  8
## Content:  chars: 3048
##
## [[3]]
## <<PlainTextDocument>>
## Metadata:  8
## Content:  chars: 1945
##
## [[4]]
## <<PlainTextDocument>>
## Metadata:  8
## Content:  chars: 1945
##
## [[5]]
## <<PlainTextDocument>>
## Metadata:  8
## Content:  chars: 2234
```

```
head(Ham_Corpus)
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 6
```

```
#Spam
Spam_Corpus <- spam.dir %>%
  to_VCorpus %>%
  docmnt_clean %>%
  addTag(tag = "emails", value = "spam")

inspect(Spam_Corpus[1:5])
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 5
##
## [[1]]
## <<PlainTextDocument>>
## Metadata:  8
## Content:  chars: 2334
##
## [[2]]
## <<PlainTextDocument>>
## Metadata:  8
## Content:  chars: 2926
##
## [[3]]
## <<PlainTextDocument>>
## Metadata:  8
## Content:  chars: 3602
##
```

```
## [[4]]
## <<PlainTextDocument>>
## Metadata:  8
## Content:  chars: 3675
##
## [[5]]
## <<PlainTextDocument>>
## Metadata:  8
## Content:  chars: 2183
```

```
writeLines(as.character(Ham_Corpus[[2]]))
```

```
## exmhworkersadminredhatcom thu aug
## returnpath exmhworkersadminspamassassintaintorg
## deliveredto zzzzlocalhostnetnoteinccom
## receiv localhost localhost
## phoboslabsnetnoteinccom postfix esmtp id dec
## zzzzlocalhost thu aug edt
## receiv phobo
## localhost imap fetchmail
## zzzzlocalhost singledrop thu aug ist
## receiv listmanspamassassintaintorg listmanspamassassintaintorg
## dogmaslashnullorg esmtp id gmbyrz
## zzzzexmhspamassassintaintorg thu aug
## receiv listmanspamassassintaintorg localhostlocaldomain
## listmanredhatcom postfix esmtp id thu aug
## edt
## deliveredto exmhworkerslistmanspamassassintaintorg
## receiv intmxcorpspamassassintaintorg intmxcorpspamassassintaintorg
## listmanredhatcom postfix esmtp id cfd
## exmhworkerslistmanredhatcom thu aug
## edt
## receiv maillocalhost intmxcorpspamassassintaintorg
## id gmbyg exmhworkerslistmanredhatcom thu aug
##
## receiv mxspamassassintaintorg mxspamassassintaintorg
## intmxcorpredhatcom smtp id gmbyy
## exmhworkersredhatcom thu aug
## receiv ratreepsuacth mxspamassassintaintorg
## smtp id gmbihl exmhworkersredhatcom
## thu aug
## receiv deltacsmuozau deltacoepsuacth
## ratreepsuacth esmtp id gmbwel
## thu aug ict
## receiv munnariozau localhost deltacsmuozau
## esmtp id gmbqpw thu aug
## ict
## robert elz kremunnariozau
## chris garrigu cwgdatedfaddeepeddycom
## cc exmhworkersspamassassintaintorg
## subject re new sequenc window
## inreplyto tmdadeepeddyvirciocom
## refer tmdadeepeddyvirciocom
## tmdadeepeddyvirciocom munnariozau
```

```
## tmdadeepeddyvirciocom
## tmdadeepeddyvirciocom
## mimevers
## contenttyp textplain charsetusascii
## messageid munnariozau
## xloop exmhworkersspamassassintaintorg
## sender exmhworkersadminspamassassintaintorg
## errorsto exmhworkersadminspamassassintaintorg
## xbeenther exmhworkersspamassassintaintorg
## xmailmanvers
## preced bulk
## listhelp mailtoexmhworkersrequestspamassassintaintorgsubjecthelp
## listpost mailtoexmhworkersspamassassintaintorg
## listsubscrib httpslistmanspamassassintaintorgmailmanlistinfoexmhwork
## mailtoexmhworkersrequestredhatcomsubjectsubscrib
## listid discuss list exmh develop exmhworkersspamassassintaintorg
## listunsubscrib httpslistmanspamassassintaintorgmailmanlistinfoexmhwork
## mailtoexmhworkersrequestredhatcomsubjectunsubscrib
## listarch httpslistmanspamassassintaintorgmailmanprivateexmhwork
## date thu aug
##
## date wed aug
## chris garrigu cwgdatedfaddeepeddycom
## messageid tmdadeepeddyvirciocom
##
##
## cant reproduc error
##
## repeat like everi time without fail
##
## debug log pick happen
##
## pickit exec pick inbox list lbrace lbrace subject ftp rbrace rbrace sequenc mercuri
## exec pick inbox list lbrace lbrace subject ftp rbrace rbrace sequenc mercuri
## ftocpickmsg hit
## mark hit
## tkerror syntax error express int
##
## note run pick command hand
##
## delta pick inbox list lbrace lbrace subject ftp rbrace rbrace sequenc mercuri
## hit
##
## that hit come obvious version nmh im
## use
##
## delta pick version
## pick nmh compil fuchsiacsmuozau sun mar ict
##
## relev part mhprofil
##
## delta mhparam pick
## seq sel list
##
```

```
##
## sinc pick command work sequenc actual
## one that explicit command line search popup
## one come mhprofil get creat
##
## kre
##
## ps still use version code form day ago havent
## abl reach cvs repositori today local rout issu think
##
##
##
##
## exmhwork mail list
## exmhworkersredhatcom
## httpslistmanredhatcommailmanlistinfoexmhwork
```

```
writeLines(as.character(Ham_Corpus[[8]]))
```

```
## irregularsadmintbtf thu aug
## returnpath irregularsadmintbtf
## deliveredto zzzzlocalhostnetnoteinccom
## receiv localhost localhost
## phoboslabsnetnoteinccom postfix esmtp id daec
## zzzzlocalhost thu aug edt
## receiv phobo
## localhost imap fetchmail
## zzzzlocalhost singledrop thu aug ist
## receiv webtbtf routetelocitycom
## dogmaslashnullorg esmtp id
## gmdgoz zzzzirrspamassassintaintorg thu aug
## receiv webtbtf localhostlocaldomain webtbtf
## esmtp id gmdpi thu aug
##
## receiv redharveehom red may forg
## webtbtf esmtp id gmdoi
## irregularstbtf thu aug
## receiv prservnet outprservnet
## redharveehom esmtp id gmdfbd
## irregularstbtf thu aug
## receiv
## slipmausprservnet prservnet
## esmtp id qujc thu aug
## mimevers
## xsender unverifi
## messageid pbaca
## undisclosedrecipi
## monti solomon montyroscomcom
## contenttyp textplain charsetusascii
## subject irr klez virus wont die
## sender irregularsadmintbtf
## errorsto irregularsadmintbtf
## xbeenther irregularstbtf
## xmailmanvers
```

```
## preced bulk
## listhelp mailtoirregularsrequesttbtfsubjecthelp
## listpost mailtoirregularstbtf
## listsubscrib httptbtfmailmanlistinfoirregular
## mailtoirregularsrequesttbtfsubjectsubscrib
## listid new home tbtf irregular mail list irregularstbtf
## listunsubscrib httptbtfmailmanlistinfoirregular
## mailtoirregularsrequesttbtfsubjectunsubscrib
## listarch httptbtfmailmanprivateirregular
## date thu aug
##
## klez virus wont die
##
## alreadi prolif virus ever klez continu wreak havoc
##
## andrew brandt
## septemb issu pc world magazin
## post thursday august
##
##
## klez worm approach seventh month wriggl across
## web make one persist virus ever
## expert warn may harbing new virus use
## combin pernici approach go pc pc
##
## antivirus softwar maker symantec mcafe report
## new infect daili sign letup press time
## british secur firm messagelab estim everi
## email messag hold variat klez virus say
## klez alreadi surpass last summer sircam prolif
## virus ever
##
## newer klez variant arent mere nuisancesthey can carri
## virus corrupt data
##
##
##
## httpwwwpcworldcomnewsarticleaidasp
##
## irregular mail list
## irregularstbtf
## httptbtfmailmanlistinfoirregular
```

Create wordcloud for Ham and Spam corpus before cleanup using bing lexicon

```
#TermDocumentMatrix
docs <- Corpus(VectorSource(Ham_Corpus))
dtm1 <- TermDocumentMatrix(docs)
m <- as.matrix(dtm1)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)
head(d, 10)
```

```
##                         word  freq
## "",                      "", 52958
## character(0), character(0), 24270
## "receiv             "receiv 22723
## esmtp                 esmtp 10228
## mon                     mon  9561
## sep",                 sep",  8454
## ist",                 ist",  6799
## sep                     sep  6721
## "jmlocalhost   "jmlocalhost  6702
## localhost         localhost  6043
```

```
mydtm4 <- tidy(dtm1)
str(mydtm4)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    54318 obs. of  3 variables:
##  $ term    : chr  "\"\")," "\"\"," "\"\\023c\\024" "\"aa" ...
##  $ document: chr  "1" "1" "1" "1" ...
##  $ count   : num  3938 52958 1 19 2 ...
```

```
mydtm_sentiments4 <- slice(mydtm4 , 1:60000) %>% inner_join(get_sentiments("bing"), by = c(term = "word
str(mydtm_sentiments4)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    908 obs. of  4 variables:
##  $ term     : chr  "abolish" "abort" "abound" "absurd" ...
##  $ document : chr  "1" "1" "1" "1" ...
##  $ count    : num  4 10 5 28 13 18 2 2 5 20 ...
##  $ sentiment: chr  "negative" "negative" "positive" "negative" ...
```

```
docs4 <- Corpus(VectorSource(Spam_Corpus))
dtm5 <- TermDocumentMatrix(docs4)
m5 <- as.matrix(dtm5)
v5 <- sort(rowSums(m5),decreasing=TRUE)
d5 <- data.frame(word = names(v5),freq=v5)
head(d5, 10)
```

```
##                         word  freq
## "",                      "", 28837
## character(0), character(0),  8382
## "tr",                 "tr",  6841
## "receiv             "receiv  6116
## "td                     "td  5496
## mon                     mon  3230
## size                   size  3049
## "br",                 "br",  2833
## esmtp                 esmtp  2605
## jul",                 jul",  2572
```

```
#HamCorpus
mydtm4 <- tidy(dtm1)
str(mydtm4)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    54318 obs. of  3 variables:
##  $ term    : chr  "\"\")," "\"\"," "\"\\023c\\024" "\"aa" ...
##  $ document: chr  "1" "1" "1" "1" ...
##  $ count   : num  3938 52958 1 19 2 ...
```

```
mydtm_sentiments4 <- slice(mydtm4 , 1:100000) %>% inner_join(get_sentiments("bing"), by = c(term = "wor
str(mydtm_sentiments4)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    908 obs. of  4 variables:
##  $ term     : chr  "abolish" "abort" "abound" "absurd" ...
##  $ document : chr  "1" "1" "1" "1" ...
##  $ count    : num  4 10 5 28 13 18 2 2 5 20 ...
##  $ sentiment: chr  "negative" "negative" "positive" "negative" ...
```

```
#SpamCorpus
mydtm5 <- tidy(dtm5)
str(mydtm5)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    77347 obs. of  3 variables:
##  $ term    : chr  "\"\")," "\"\"," "\"aa" "\"aa\"," ...
##  $ document: chr  "1" "1" "1" "1" ...
##  $ count   : num  1186 28837 5 4 2 ...
```

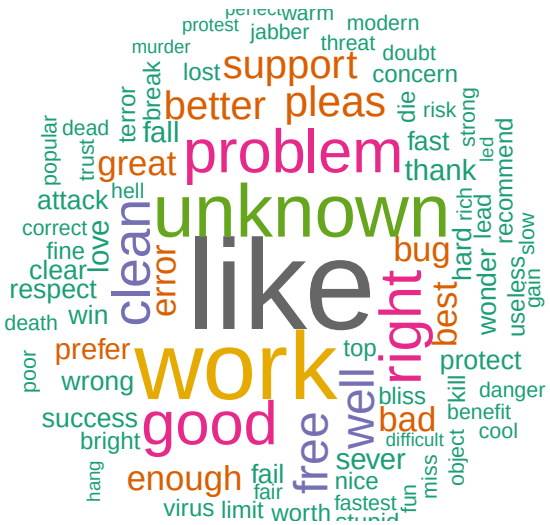
```
mydtm_sentiments5 <- slice(mydtm5 , 1:100000) %>% inner_join(get_sentiments("bing"), by = c(term = "wor
str(mydtm_sentiments5)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    541 obs. of  4 variables:
##  $ term     : chr  "abort" "abscond" "acclaim" "accomplish" ...
##  $ document : chr  "1" "1" "1" "1" ...
##  $ count    : num  1 1 3 3 1 2 7 27 4 1 ...
##  $ sentiment: chr  "negative" "negative" "positive" "positive" ...
```

```
#Side By Side
#Create two panels to add the word clouds to
#par(mfrow=c(1,2))
#set.seed(1234)
plot.new()
text(x=0.5, y=0.5, "Wordcloud using Bing Lexicon for Ham corpus")
```

Wordcloud using Bing Lexicon for Ham corpus

```
wordcloud(words = mydtm_sentiments4$term, freq = mydtm_sentiments4$count, min.freq = 50,          max.w
          colors=brewer.pal(8, "Dark2"))
```




```
plot.new()
text(x=0.5, y=0.5, "Wordcloud using Bing Lexicon for Spam corpus")
```

Wordcloud using Bing Lexicon for Spam corpus

```
wordcloud(words = mydtm_sentiments5$term, freq = mydtm_sentiments5$count, min.freq = 50,          max.w
          colors=brewer.pal(8, "Dark2"))
```

Combine the two cleaned up corpus data into a single data frame

```
ham_DtFr = as.data.frame(unlist(Ham_Corpus), stringsAsFactors = FALSE)
ham_DtFr$type = "ham"
colnames(ham_DtFr) = c("text", "type")

spam_DtFr = as.data.frame(unlist(Spam_Corpus), stringsAsFactors = FALSE)
spam_DtFr$type = "spam"
colnames(spam_DtFr) = c("text", "type")

combined_DtFr = rbind(ham_DtFr[1:1000,], spam_DtFr[1:1000,]) # Combined dataframe of both corpuses
head(combined_DtFr, 10)
```

```
##                                                          text type
## 1                              exmhworkersadminredhatcom thu aug  ham
## 2                 returnpath exmhworkersadminspamassassintaintorg  ham
## 3                          deliveredto zzzzlocalhostnetnoteinccom  ham
## 4                                    receiv localhost localhost  ham
## 5                    phoboslabsnetnoteinccom postfix esmtp id dec  ham
## 6                                     zzzzlocalhost thu aug edt  ham
## 7                                                 receiv phobo  ham
## 8                                   localhost imap fetchmail  ham
## 9                            zzzzlocalhost singledrop thu aug ist  ham
## 10 receiv listmanspamassassintaintorg listmanspamassassintaintorg  ham
```

```
final_corpus = c(Ham_Corpus, Spam_Corpus) # Combined Corpus
inspect(final_corpus[1:5])
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 5
##
## [[1]]
## <<PlainTextDocument>>
## Metadata:  8
## Content:  chars: 3048
##
## [[2]]
## <<PlainTextDocument>>
## Metadata:  8
## Content:  chars: 3048
##
## [[3]]
## <<PlainTextDocument>>
## Metadata:  8
## Content:  chars: 1945
##
## [[4]]
## <<PlainTextDocument>>
## Metadata:  8
## Content:  chars: 1945
##
## [[5]]
## <<PlainTextDocument>>
## Metadata:  8
## Content:  chars: 2234
```

Partition data into training set and test set in ratio of 70:30

```
set.seed(100)
combined_DtFr$text[combined_DtFr$text == ""] = "NaN"
train_index = createDataPartition(combined_DtFr$type, p = 0.70, list = FALSE)
corpus_train = combined_DtFr[train_index,]
head(corpus_train)
```

```
##                                                    text type
## 1                        exmhworkersadminredhatcom thu aug  ham
## 2          returnpath exmhworkersadminspamassassintaintorg  ham
## 3                   deliveredto zzzzlocalhostnetnoteinccom  ham
## 11                       dogmaslashnullorg esmtp id gmbyrz  ham
## 12                    zzzzexmhspamassassintaintorg thu aug  ham
## 13 receiv listmanspamassassintaintorg localhostlocaldomain  ham
```

```
corpus_test = combined_DtFr[-train_index,]
head(corpus_test, 10)
```

```
##                                                          text type
## 4                                  receiv localhost localhost  ham
## 5                  phoboslabsnetnoteinccom postfix esmtp id dec  ham
## 6                                  zzzzlocalhost thu aug edt  ham
## 7                                              receiv phobo  ham
## 8                                   localhost imap fetchmail  ham
## 9                          zzzzlocalhost singledrop thu aug ist  ham
## 10 receiv listmanspamassassintaintorg listmanspamassassintaintorg  ham
## 16            deliveredto exmhworkerslistmanspamassassintaintorg  ham
## 18                         listmanredhatcom postfix esmtp id cfd  ham
## 20                                                         edt  ham
```

Create a Document Term Matrix

```
trainCorpus = Corpus(VectorSource(corpus_train$text))
testCorpus = Corpus(VectorSource(corpus_test$text))

train_clean_corpus <- tm_map(trainCorpus, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(trainCorpus, removeNumbers): transformation
## drops documents
```

```
test_clean_corpus <- tm_map(testCorpus, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(testCorpus, removeNumbers): transformation
## drops documents
```

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

```
## Warning in tm_map.SimpleCorpus(train_clean_corpus, removePunctuation):
## transformation drops documents
```

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

```
## Warning in tm_map.SimpleCorpus(test_clean_corpus, removePunctuation):
## transformation drops documents
```

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

```
## Warning in tm_map.SimpleCorpus(train_clean_corpus, removeWords,
## stopwords()): transformation drops documents
```

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

```
## Warning in tm_map.SimpleCorpus(test_clean_corpus, removeWords,
## stopwords()): transformation drops documents
```

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

```
## Warning in tm_map.SimpleCorpus(train_clean_corpus, stripWhitespace):
## transformation drops documents
```

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

```
## Warning in tm_map.SimpleCorpus(test_clean_corpus, stripWhitespace):
## transformation drops documents
```

```
corpus_train_dtm = DocumentTermMatrix(train_clean_corpus)
corpus_test_dtm = DocumentTermMatrix(test_clean_corpus)
```

Create Term Document Matrix and Plot wordcloud and sentiment

```
#Wordcloud for train_clean_Corpus
docs1 <- Corpus(VectorSource(train_clean_corpus))
dtm2 <- TermDocumentMatrix(docs1)
m <- as.matrix(dtm2)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)
head(d, 10)
```

```
##            word freq
## "nan",   "nan",  199
## "",        "",   99
## "receiv "receiv   92
## "br",     "br",   65
## aug",     aug",   65
## thu         thu   58
## "brbr", "brbr",   57
## esmtp     esmtp   30
## br",       br",   21
## mail       mail   21
```

```
mydtm <- tidy(dtm2)
str(mydtm)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    1305 obs. of  3 variables:
##  $ term    : chr  "\"\"," "\"aabaabhaceadbdc\"," "\"abl" "\"absorb" ...
##  $ document: chr  "1" "1" "1" "1" ...
##  $ count   : num  99 1 2 2 1 1 2 2 1 1 ...
```

```
head(mydtm, 100)
```

```
## # A tibble: 100 x 3
##    term                   document count
```

```
##    <chr>                  <chr>    <dbl>
##  1 "\"\","                1           99
##  2 "\"aabaabhaceadbdc\"," 1            1
##  3 "\"abl"                1            2
##  4 "\"absorb"             1            2
##  5 "\"act"                1            1
##  6 "\"add"                1            1
##  7 "\"addressbr\","       1            2
##  8 "\"agre"               1            2
##  9 "\"aid"                1            1
## 10 "\"alreadi"            1            1
## # ... with 90 more rows
```

```
#slice sentiments of 1000 rows

mydtm_sentiments <- slice(mydtm , 1:100000) %>% inner_join(get_sentiments("bing"), by = c(term = "word")
mydtm_sentiments
```

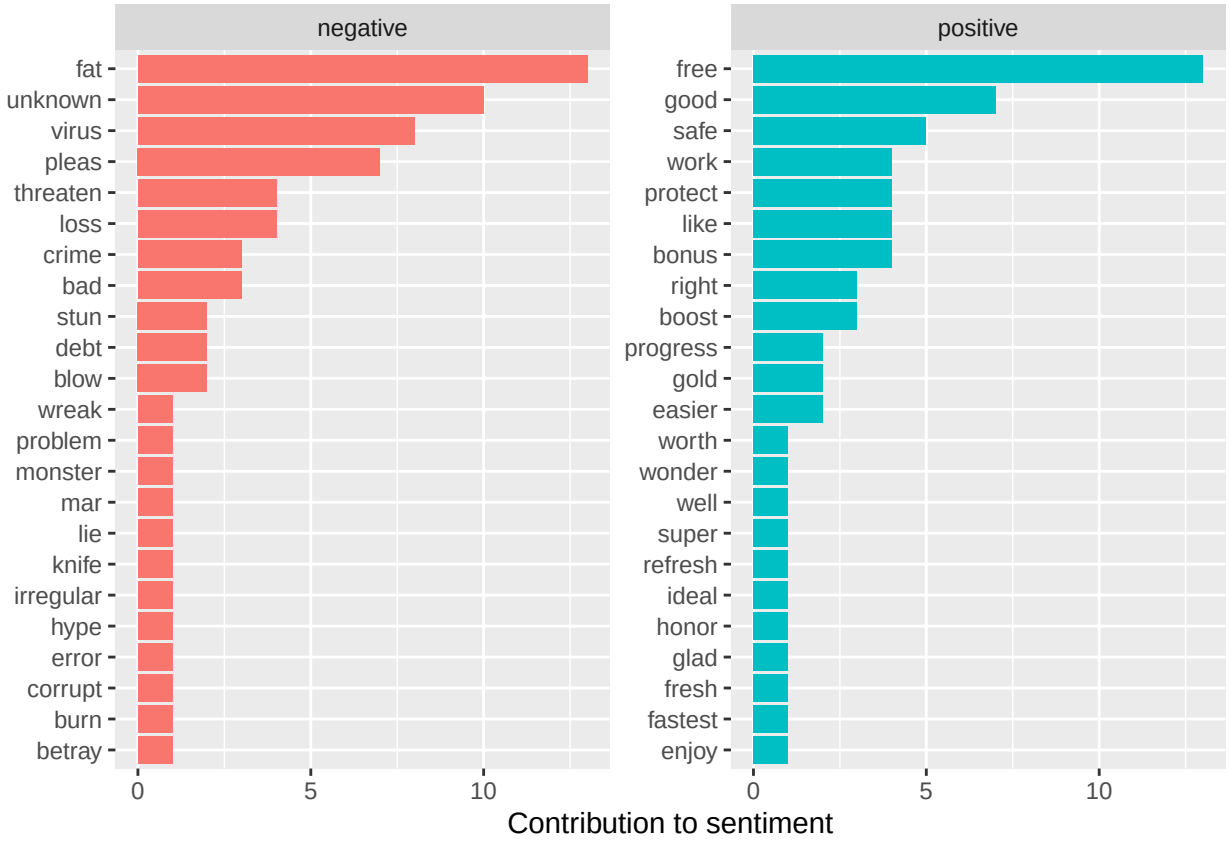
```
## # A tibble: 46 x 4
##    term    document count sentiment
##    <chr>   <chr>    <dbl> <chr>
##  1 bad     1            3 negative
##  2 betray  1            1 negative
##  3 blow    1            2 negative
##  4 bonus   1            4 positive
##  5 boost   1            3 positive
##  6 burn    1            1 negative
##  7 corrupt 1            1 negative
##  8 crime   1            3 negative
##  9 debt    1            2 negative
## 10 easier  1            2 positive
## # ... with 36 more rows
```

```
str(mydtm_sentiments)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    46 obs. of  4 variables:
##  $ term     : chr  "bad" "betray" "blow" "bonus" ...
##  $ document : chr  "1" "1" "1" "1" ...
##  $ count    : num  3 1 2 4 3 1 1 3 2 2 ...
##  $ sentiment: chr  "negative" "negative" "negative" "positive" ...
```

```
mydtm_sentiments %>%
  count(sentiment, term, wt = count) %>%
  top_n(50) %>%
  ungroup() %>%
  mutate(term = reorder(term, n)) %>%
  ggplot(aes(term, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Contribution to sentiment",
       x = NULL) +
  coord_flip()
```

```
## Selecting by n
```




```
#Wordcloud for test_clean_Corpus
docs2 <- Corpus(VectorSource(test_clean_corpus))
dtm3 <- TermDocumentMatrix(docs2)
m3 <- as.matrix(dtm3)
v3 <- sort(rowSums(m3),decreasing=TRUE)
d3 <- data.frame(word = names(v3),freq=v3)
head(d3, 10)
```

```
##            word freq
## "nan",   "nan",   80
## "receiv "receiv   41
## "",         "",   36
## "brbr", "brbr",   30
## thu         thu   29
## aug",     aug",   28
## "br",     "br",   23
## aug         aug   14
## esmtp     esmtp   14
## mail       mail   13
```

```
mydtm3 <- tidy(dtm3)
str(mydtm3)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    826 obs. of  3 variables:
##  $ term    : chr  "\"\"," "\"abandon" "\"act" "\"ad" ...
##  $ document: chr  "1" "1" "1" "1" ...
##  $ count   : num  36 1 1 1 2 2 3 1 1 1 ...
```

```
head(mydtm3, 100)
```

```
## # A tibble: 100 x 3
##    term                          document count
##    <chr>                         <chr>    <dbl>
##  1 "\"\","                       1           36
##  2 "\"abandon"                   1            1
##  3 "\"act"                       1            1
##  4 "\"ad"                        1            1
##  5 "\"addressbr\","              1            2
##  6 "\"agenc"                     1            2
##  7 "\"aid"                       1            3
##  8 "\"alreadi"                   1            1
##  9 "\"americanexpressbr\","      1            1
## 10 "\"amknightmailexcitecom\"," 1            1
## # ... with 90 more rows
```

```
#slice sentiments of 1000 rows

mydtm_sentiments3 <- slice(mydtm3 , 1:100000) %>% inner_join(get_sentiments("bing"), by = c(term = "wor
mydtm_sentiments3
```

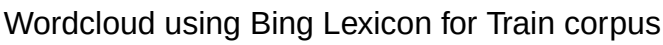
```
## # A tibble: 28 x 4
##    term    document count sentiment
##    <chr>   <chr>    <dbl> <chr>
##  1 boost   1            1 positive
##  2 corrupt 1            1 negative
##  3 enjoy   1            1 positive
##  4 error   1            1 negative
##  5 fail    1            1 negative
##  6 fastest 1            1 positive
##  7 fat     1            4 negative
##  8 free    1            4 positive
##  9 good    1            5 positive
## 10 great   1            2 positive
## # ... with 18 more rows
```

```
str(mydtm_sentiments3)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    28 obs. of  4 variables:
##  $ term     : chr  "boost" "corrupt" "enjoy" "error" ...
##  $ document : chr  "1" "1" "1" "1" ...
##  $ count    : num  1 1 1 1 1 1 4 4 5 2 ...
##  $ sentiment: chr  "positive" "negative" "positive" "negative" ...
```

```r
#Side By Side
#Create two panels to add the word clouds to
#par(mfrow=c(1,2))

plot.new()
text(x=0.5, y=0.5, "Wordcloud using Bing Lexicon for Train corpus")
```




```r
wordcloud(words = mydtm_sentiments$term, freq = mydtm_sentiments$count, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```


```
plot.new()
text(x=0.5, y=0.5, "Wordcloud using Bing Lexicon for Test corpus")
```

Wordcloud using Bing Lexicon for Test corpus

```
wordcloud(words = mydtm_sentiments3$term, freq = mydtm_sentiments3$count, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```

Define input variables 0 and 1 from string to integer

```
convert_count = function(x) {
  y = ifelse(x > 0, 1, 0)
  y = factor(y, levels = c(0,1), labels = c(0,1))
  y
}
```

Train the model and predict the outcome

```
train = apply(corpus_train_dtm, 2, convert_count)
test = apply(corpus_test_dtm, 2, convert_count)

str(train)
```

```
##  chr [1:1400, 1:1021] "1" "0" "0" "0" "1" "0" "1" "0" "0" "1" "0" "1" ...
##  - attr(*, "dimnames")=List of 2
##   ..$ Docs : chr [1:1400] "1" "2" "3" "4" ...
##   ..$ Terms: chr [1:1021] "aug" "exmhworkersadminredhatcom" "thu" "exmhworkersadminspamassassintaint
```

```r
str(test)
```

```
##  chr [1:600, 1:697] "1" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" ...
##  - attr(*, "dimnames")=List of 2
##   ..$ Docs : chr [1:600] "1" "2" "3" "4" ...
##   ..$ Terms: chr [1:697] "localhost" "receiv" "dec" "esmtp" ...
```

Use NaiveBayes Model to train and test/predict the test data set

```
classifier = naiveBayes(train, factor(corpus_train$type))
pred = predict(classifier, newdata = test)

classifier$apriori
```

```
## factor(corpus_train$type)
##  ham spam
##  700  700
```

```
classifier$tables[1:15]
```

```
## $aug
##                          aug
## factor(corpus_train$type)           0           1
##                       ham  0.871428571 0.128571429
##                       spam 0.991428571 0.008571429
##
## $exmhworkersadminredhatcom
##                          exmhworkersadminredhatcom
## factor(corpus_train$type)           0           1
##                       ham  0.997142857 0.002857143
##                       spam 1.000000000 0.000000000
##
## $thu
##                          thu
## factor(corpus_train$type)          0          1
##                       ham  0.91428571 0.08571429
##                       spam 1.00000000 0.00000000
##
## $exmhworkersadminspamassassintaintorg
##                          exmhworkersadminspamassassintaintorg
## factor(corpus_train$type)           0           1
##                       ham  0.994285714 0.005714286
##                       spam 1.000000000 0.000000000
##
## $returnpath
##                          returnpath
## factor(corpus_train$type)           0           1
##                       ham  0.988571429 0.011428571
##                       spam 0.994285714 0.005714286
##
## $deliveredto
##                          deliveredto
## factor(corpus_train$type)           0           1
##                       ham  0.981428571 0.018571429
##                       spam 0.997142857 0.002857143
##
## $zzzzlocalhostnetnoteinccom
##                          zzzzlocalhostnetnoteinccom
```

```
## factor(corpus_train$type)    0    1
##                       ham  0.99 0.01
##                       spam 1.00 0.00
##
## $dogmaslashnullorg
##                           dogmaslashnullorg
## factor(corpus_train$type)           0           1
##                       ham  0.988571429 0.011428571
##                       spam 0.997142857 0.002857143
##
## $esmtp
##                           esmtp
## factor(corpus_train$type)          0          1
##                       ham  0.96285714 0.03714286
##                       spam 0.98428571 0.01571429
##
## $gmbyrz
##                           gmbyrz
## factor(corpus_train$type)           0           1
##                       ham  0.997142857 0.002857143
##                       spam 1.000000000 0.000000000
##
## $zzzzexmhspamassassintaintorg
##                           zzzzexmhspamassassintaintorg
## factor(corpus_train$type)           0           1
##                       ham  0.998571429 0.001428571
##                       spam 1.000000000 0.000000000
##
## $listmanspamassassintaintorg
##                           listmanspamassassintaintorg
## factor(corpus_train$type)           0           1
##                       ham  0.997142857 0.002857143
##                       spam 1.000000000 0.000000000
##
## $localhostlocaldomain
##                           localhostlocaldomain
## factor(corpus_train$type)           0           1
##                       ham  0.995714286 0.004285714
##                       spam 1.000000000 0.000000000
##
## $receiv
##                           receiv
## factor(corpus_train$type)          0          1
##                       ham  0.89571429 0.10428571
##                       spam 0.96571429 0.03428571
##
## $listmanredhatcom
##                           listmanredhatcom
## factor(corpus_train$type)           0           1
##                       ham  0.997142857 0.002857143
##                       spam 1.000000000 0.000000000
```

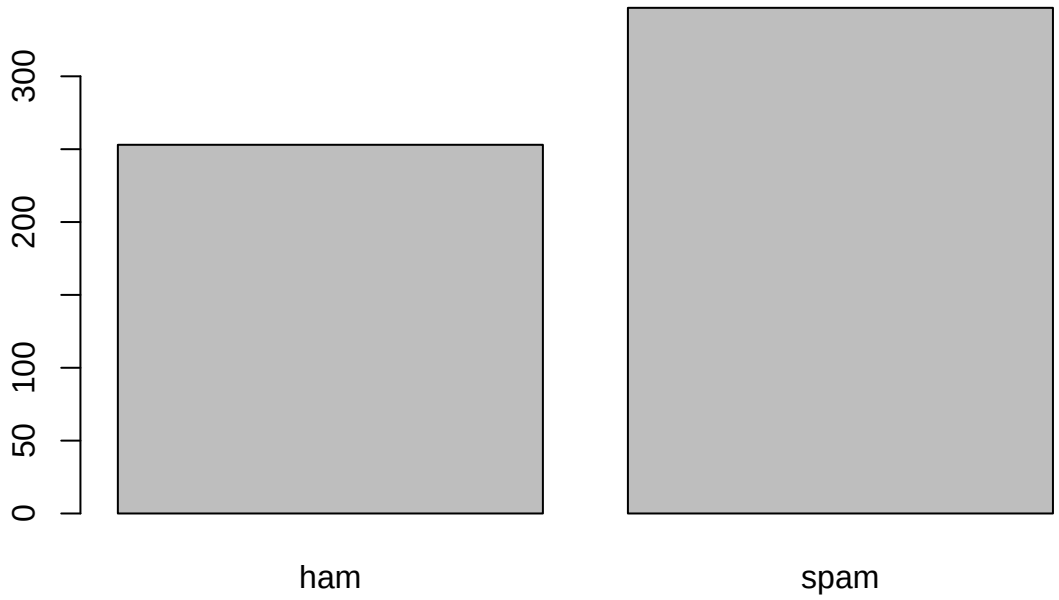
```
classifier$levels
```

```
## [1] "ham"  "spam"
```

```
classifier$call
```

```
## naiveBayes.default(x = train, y = factor(corpus_train$type))
```

```
plot(pred)
```




```
#Using SVM to classify emails

#classifier_svm = svm(formula = type ~ .,
                      # data = train,
                      # type = 'C-classification',
                      # kernel = 'linear')

#model_svm = svm(type ~ ., data = train)

#pred_svm = predict(classifier_svm, newdata = train)

#confusionMatrix(predsvm, corpus_train$type)
```

Output in the form of a confusion matrix

```
confusion_matrix = table(pred, corpus_test$type)
confusion_matrix
```

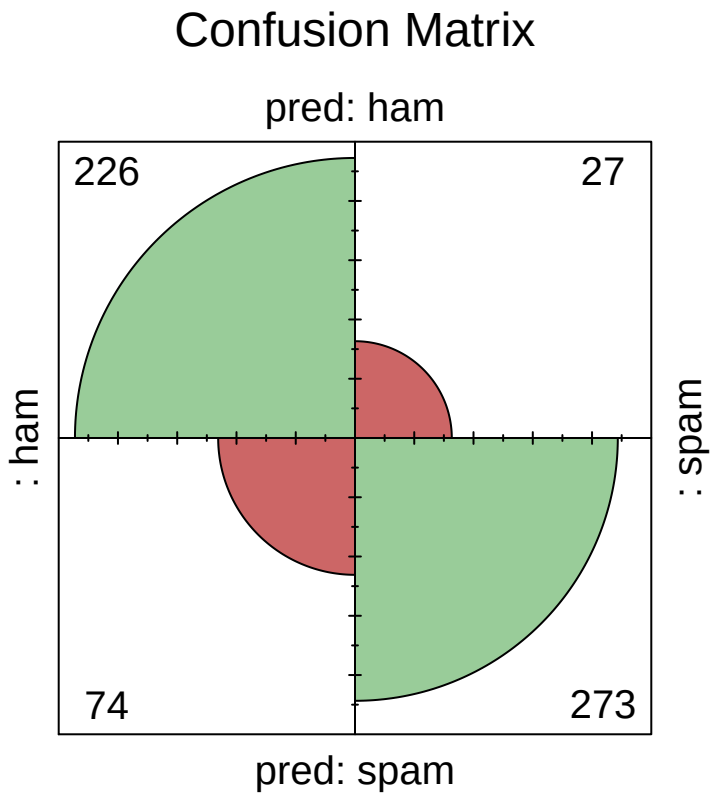
```
##
## pred   ham spam
##   ham  226   27
##   spam  74  273
```

```
confMatrix1 <- confusionMatrix(pred, as.factor(corpus_test$type))
confMatrix1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction ham spam
##       ham  226   27
##       spam  74  273
##
##                Accuracy : 0.8317
##                  95% CI : (0.7993, 0.8607)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6633
##
##  Mcnemar's Test P-Value : 4.713e-06
##
##             Sensitivity : 0.7533
##             Specificity : 0.9100
##          Pos Pred Value : 0.8933
##          Neg Pred Value : 0.7867
##              Prevalence : 0.5000
##          Detection Rate : 0.3767
##    Detection Prevalence : 0.4217
##       Balanced Accuracy : 0.8317
##
##        'Positive' Class : ham
##
```

A visual plot of the confusion matrix

```
fourfoldplot(confusion_matrix, color = c("#CC6666", "#99CC99"),
             conf.level = 0, margin = 1, main = "Confusion Matrix")
```


Confusion Matrix
pred: ham
226
27
: ham
: spam
74
273
pred: spam