```
install.packages("tm")
install.packages("wordcloud")
install.packages("e1071")
library(tm)
library(wordcloud)
library(e1071)

#creating a data frame using CSV files
#With base R functions, to avoid conversion of strings to factors you
would do, for example:

sms_spam_df <-
read.csv(file="C:\\Users\\MVD\\Desktop\\sms_spam.csv",stringsAsFactors
=F)

str(sms_spam_df)
#head(sms_spam_df)
#table(sms_spam_df$category)

#Data preparation - cleaning and standardizing text data

#creating a corpus
sms_corpus <- VCorpus(VectorSource(sms_spam_df$text))
#sms_corpus
print(sms_corpus)

#iew a summary of the first and second SMS messages in the corpus:
inspect(sms_corpus[1:2])
#sms_corpus[1:2]

#data pre-processing

#translate all letters to lower case
clean_corpus <- tm_map(sms_corpus, content_transformer(tolower))
#clean_corpus <- tm_map(sms_corpus, tolower)
#as.character(clean_corpus[[1]])

# remove numbers
clean_corpus <- tm_map(clean_corpus, removeNumbers)

# remove punctuation
clean_corpus <- tm_map(clean_corpus, removePunctuation)

# remove stopwords
stopwords()[1:15]
clean_corpus <- tm_map(clean_corpus, removeWords, stopwords())
#as.character(clean_corpus[[1]])

# remove whitespaces
clean_corpus <- tm_map(clean_corpus, stripWhitespace)

#inspect, i.e., display detailed information on a corpus, a term-
document matrix, or a text document.
inspect(clean_corpus[1:3])

#tokenize each message into words to build the key structure for the
```

analysis, a sparse matrix comprising:

```
sms_dtm <- DocumentTermMatrix(clean_corpus)

str(sms_dtm)

#The which() function will return the position of the elements
#(i.e., row number/column number/array index) in a logical vector
which are TRUE.

spam_indices <- which(sms_spam_df$category == "spam")
ham_indices <- which(sms_spam_df$category == "ham")

#spam_indices
#ham_indices

wordcloud(clean_corpus[ham_indices], min.freq=40)  # look at the 40
most common words
wordcloud(clean_corpus[spam_indices], min.freq=40)

#Split out training and test sets

# split the raw data
sms_raw_train <- sms_spam_df[1:4169,]
sms_raw_test <- sms_spam_df[4170:5559,]

# then split the document-term matrix
sms_dtm_train <- sms_dtm[1:4169,]
sms_dtm_test <- sms_dtm[4170:5559,]

# and finally the corpus
sms_corpus_train <- clean_corpus[1:4169]
sms_corpus_test <- clean_corpus[4170:5559]

#create separate corpuses for spam and ham
spam <- subset(sms_raw_train, category == "spam")
ham <- subset(sms_raw_train, category == "ham")

#reducing the DTM

five_times_words <- findFreqTerms(sms_dtm_train, 5)

sms_train <- DocumentTermMatrix(sms_corpus_train,
control=list(dictionary = five_times_words))
sms_test <- DocumentTermMatrix(sms_corpus_test,
control=list(dictionary = five_times_words))

#Remember that NB works on factors, but our DTM only has numerics.
#Let's define a function which converts counts to yes/no factor, and
apply it to our reduced matrices.
convert_count <- function(x) {
  y <- ifelse(x > 0, 1,0)
  y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
  y
}
sms_train <- apply(sms_train, 2, convert_count)
sms_test <- apply(sms_test, 2, convert_count)
```

```
sms_classifier <- naiveBayes(sms_train,
factor(sms_raw_train$category))
sms_test_pred <- predict(sms_classifier, newdata=sms_test)
k=table(sms_test_pred, sms_raw_test$category)
k
accuracy = sum(diag(k))/sum(k)*100
accuracy
```