# Week 3 – Assignment 2 (Project)

By: Zach Adair

MSDS 680: Machine Learning

Regis University

5/22/2019

## Introduction

For the second assignment of this week we will take a data set of messages and classify if the messages are spam or not spam. The technique that will be used to classify the SMS message will be Naïve Bayes and there will be some data manipulation involved like changing all the letters to lowercase, removing numbers, removing punctuation and more. I will also be creating some matrices to take a deeper look at the words in the dataset as well as doing a text analysis by doing word counts and creating visualizations based on the dataset. After running my analysis I will summarize my results and check the accuracy of the model.

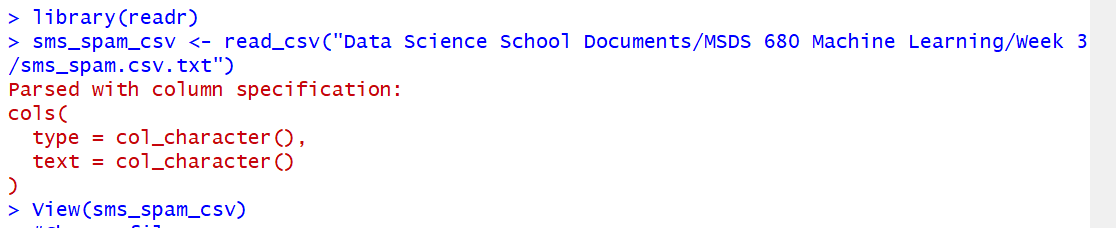
## Packages to Use

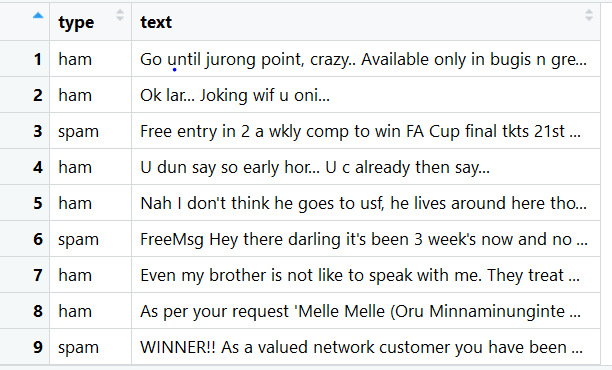
The RStudio packages I will be using for this assignment are listed below:

* e1071
* klaR
* tm
* SnowballC
* wordclouds
* RColorBrewer

## Upload Dataset into RStudio

Like any typical analysis, can’t do anything if we don’t have any data, so the first step here is to bring in the SMS dataset that was needed to be downloaded. I downloaded a text file version of it to my computer so that is how I brought it into RStudio.

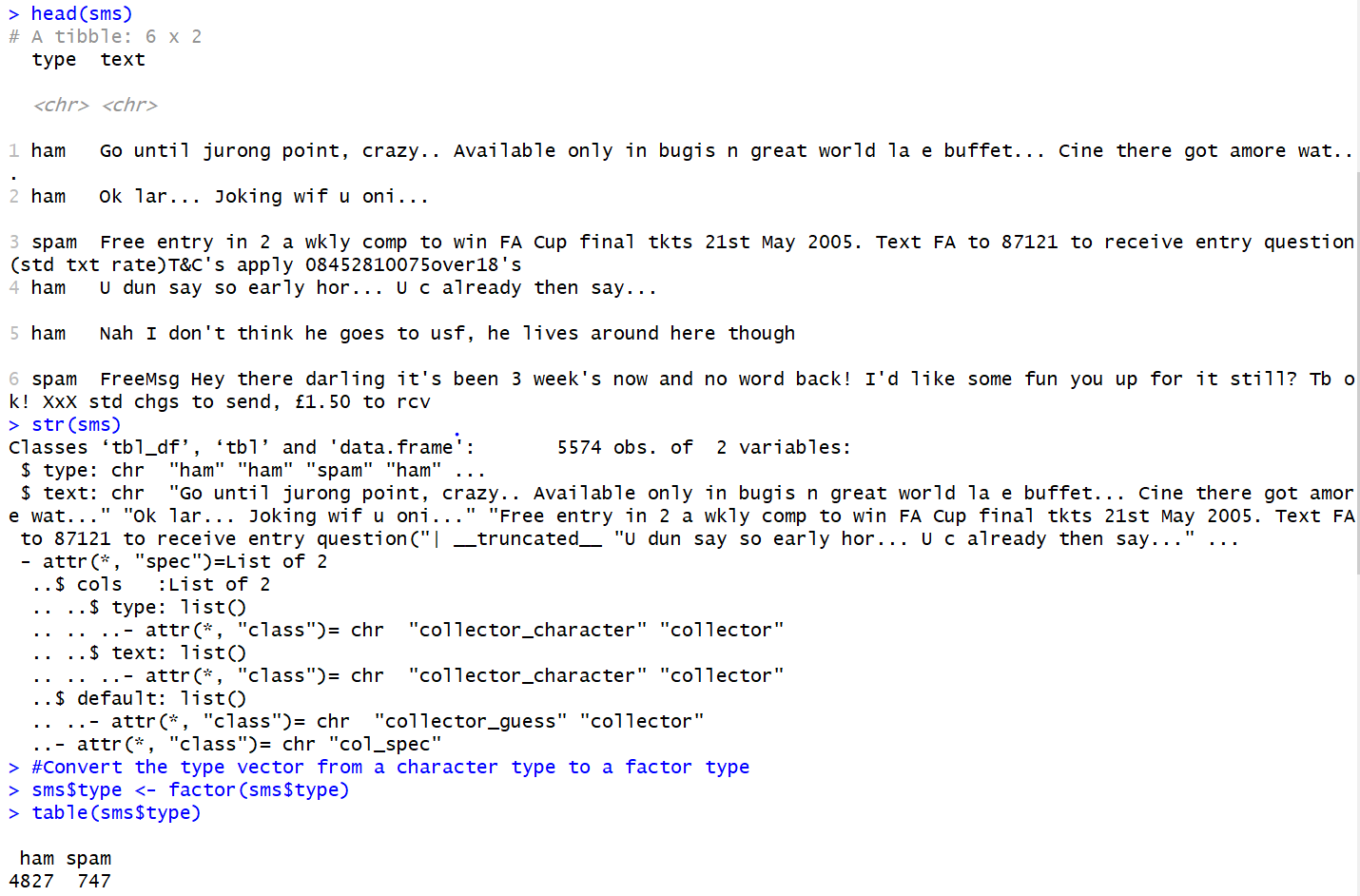




I also brought in the data to view it, the data has two fields, type and text. Type has the indication if the text is spam or not and the text field just has the messages within it.

## Explore the Data

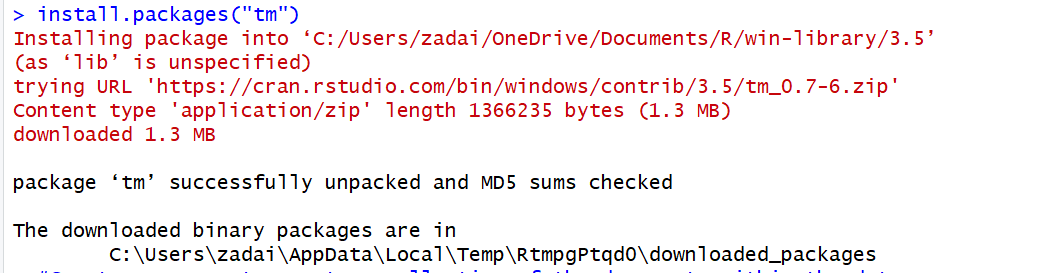
Now that the data is loaded it’s time to explore the data a little bit.

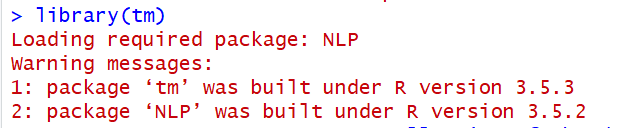


As you can see, I first looked at some of the very first data rows, then looked at the structure of the table fields. Converted the type vector from a character type to a factor type and ended it by creating a table which sums up which records are spam and which are ham.

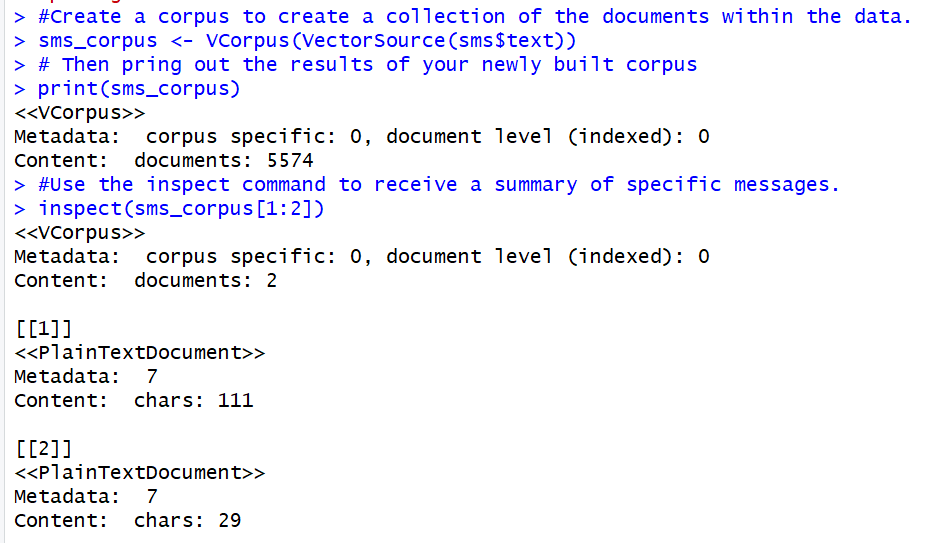
## Data Preparation

To prepare the data, I will bring in the tm package and then prepare the data from there.



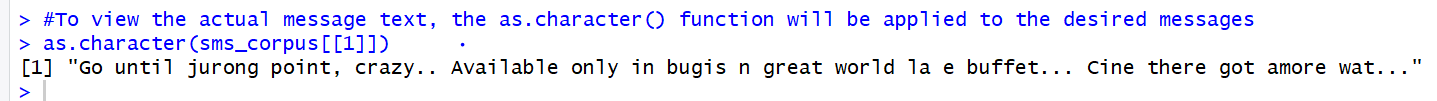


Now that the tm package is in the fold, I will create a corpus for my data. A corpus is a collection of the documents within the dataset.

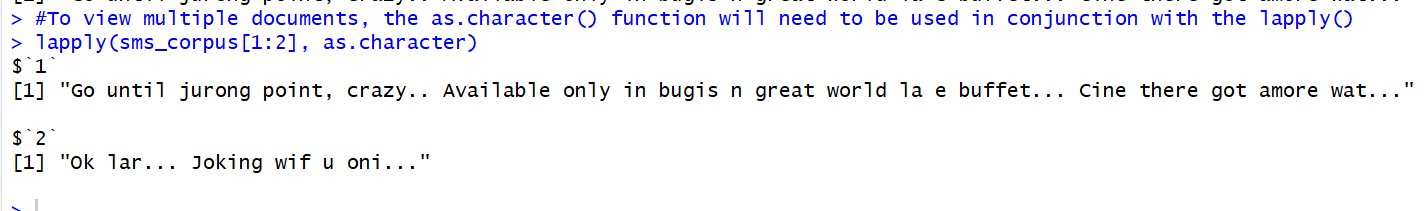


Then the inspect function which I used after the corpus is made to summarize specific messages.

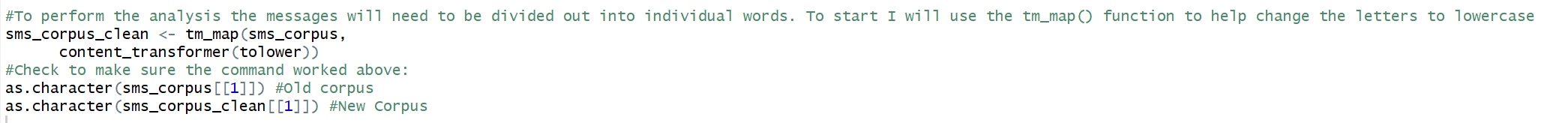
To see the messages, use the as.character() function to the corpus and see the list of the messages.

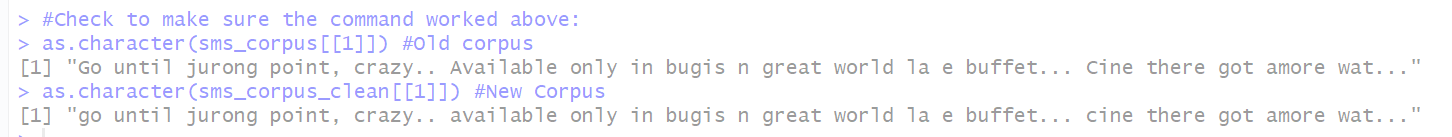


To see multiple documents the as.character() and lapply() functions will need to be used simultaneously.



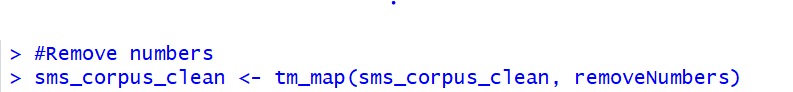
From here, I’ll change the data to lowercase and then test the old corpus against the new one.



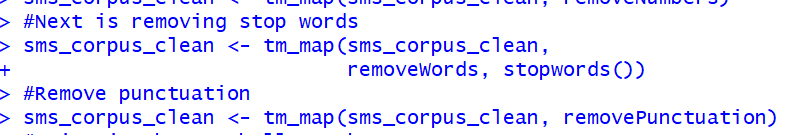


From the results, it is easy to notice that now all of the letters are now lowercase.

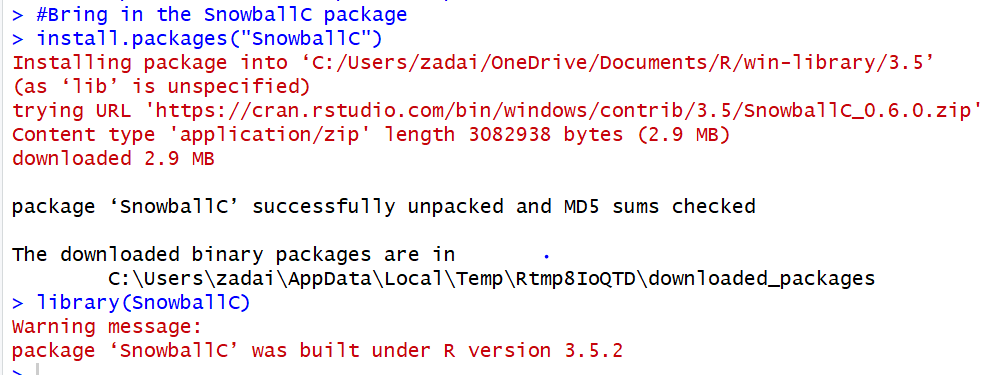
The next move is to now remove the numbers from the data, this will take a similar step but instead of tolower, the command will be removeNumbers.



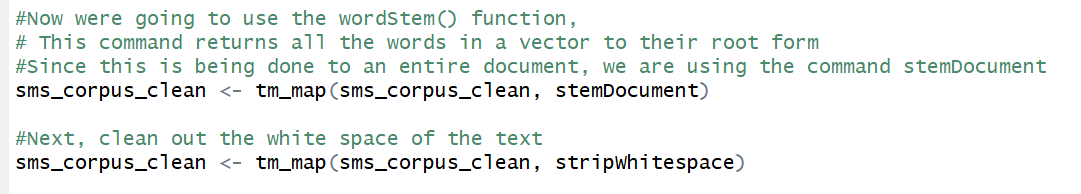
The remaining data manipulated commands that need to be done to this text are to remove stop words and remove punctuation.



With those last text manipulation steps taken its time to move onto some different commands that clean up our text for analysis but need a different R package, the SnowballC package.

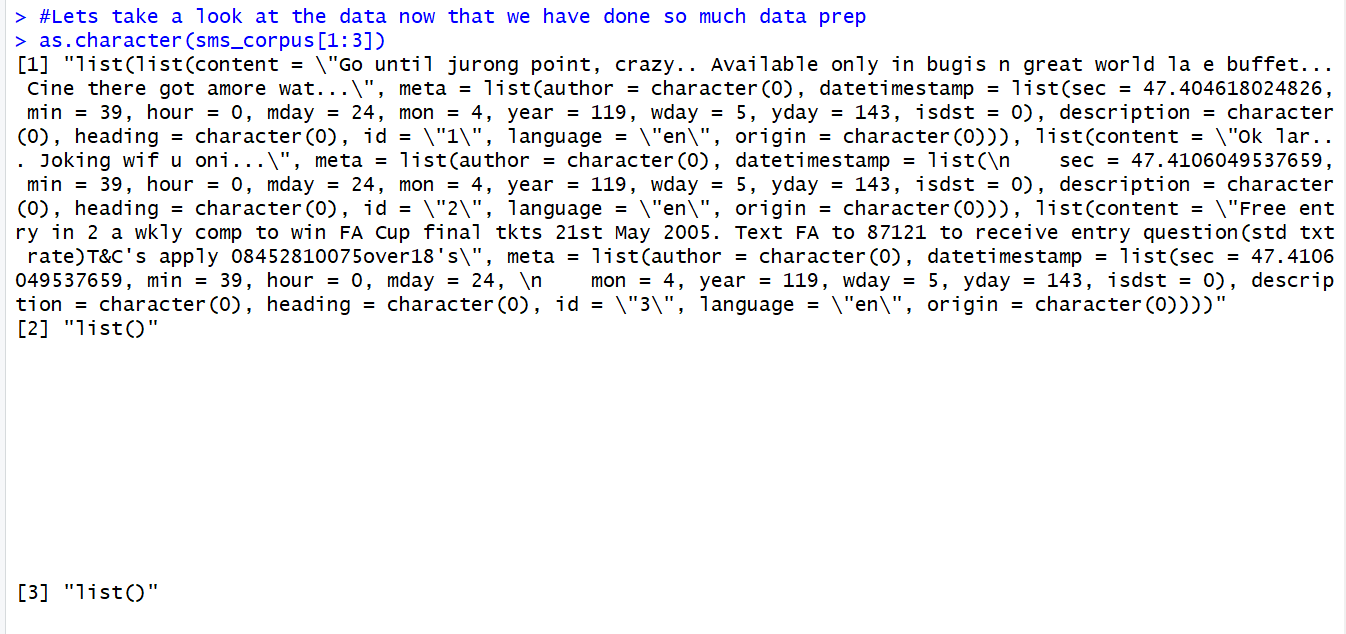


The commands were going to use now are the stemDocument() command and stripWhitespace command. The first will take any word down to its root word, the other will get rid of the white space between words.

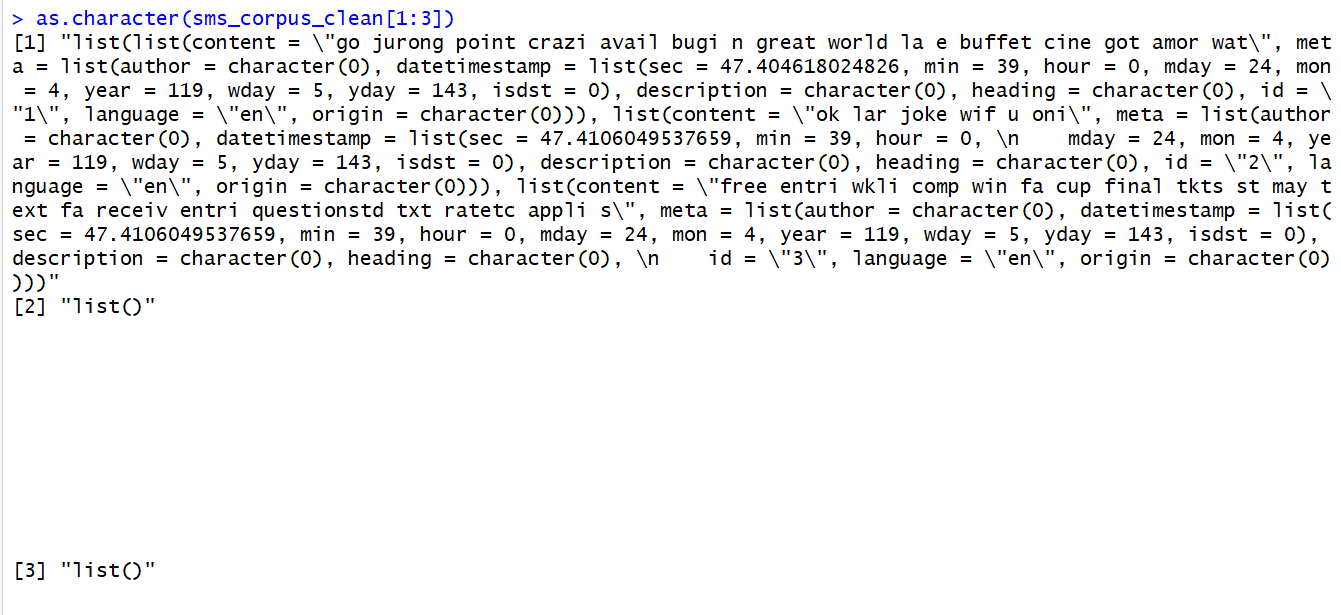


Now that the data is cleaned let’s look at a before and after of our cleaned SMS data.

Before cleaning process:

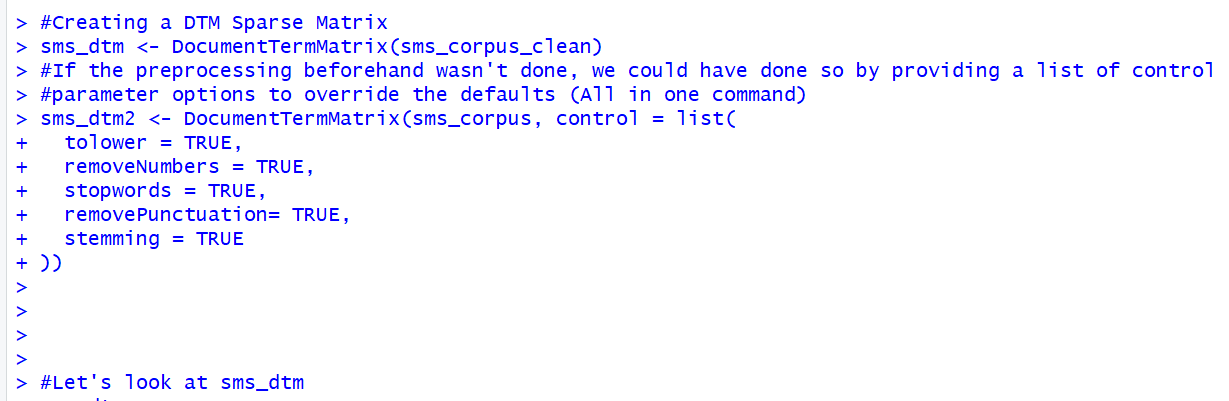


After the data has been cleaned up.

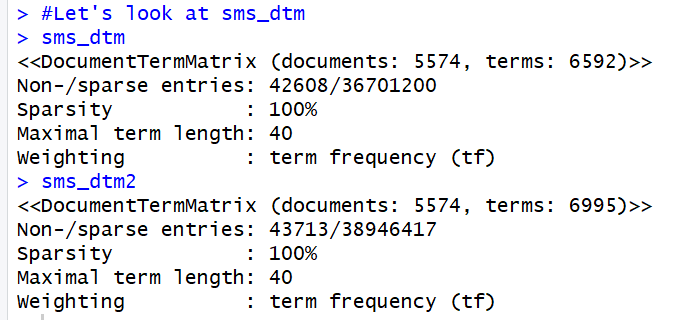


From the looks of our before it looks like all the steps that I went through in the cleaning process has shown in the outputs, there is only lowercase letters, punctuation is gone and whitespace is noticeably taken away from the after cleaned data.

The next step in the data preparation process will be spitting text documents into separate words. The process of doing this is called tokenization.

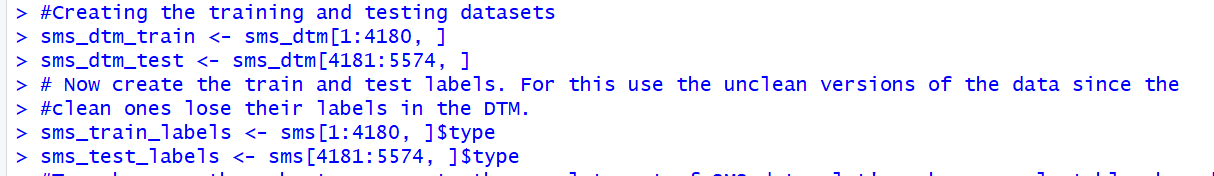


I did a DTM for both the clean and uncleaned version. Essentially sms\_dtm2 is just all the processing data steps rolled into one command execution. Let’s take a look at both of them to see the differences.

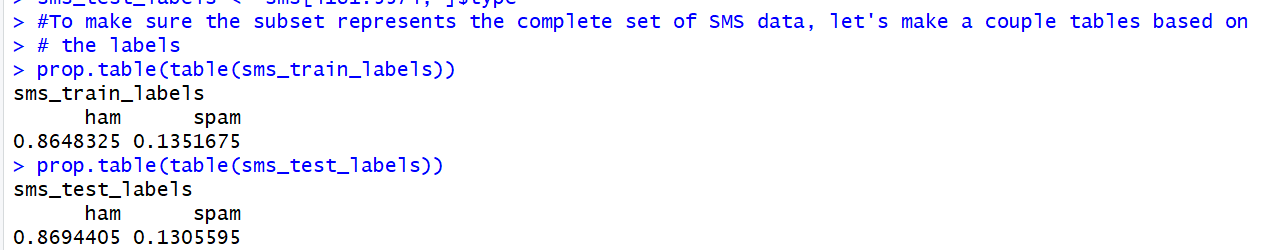


From the looks of it there seems to be only some small differences, within the terms there is a difference of just around 400 documents and the entries there appears to be a difference there but neither are big enough to really write home about so we will continue on.

With the data now parsed out, the next step is creating training and test datasets to train and test our spam filter once it is built. I will also be creating training and testing labels for my data. For my training/testing split I will have my data split into 75% training and 25% testing.



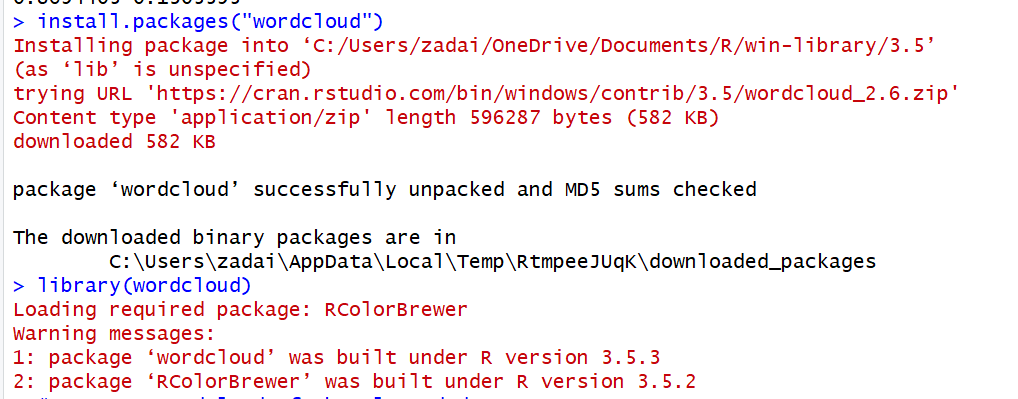
Now that the datasets and labels are made lets move into creating tables for the labels.



With the datasets all set up now I will move onto visualizing and analyzing the datasets.

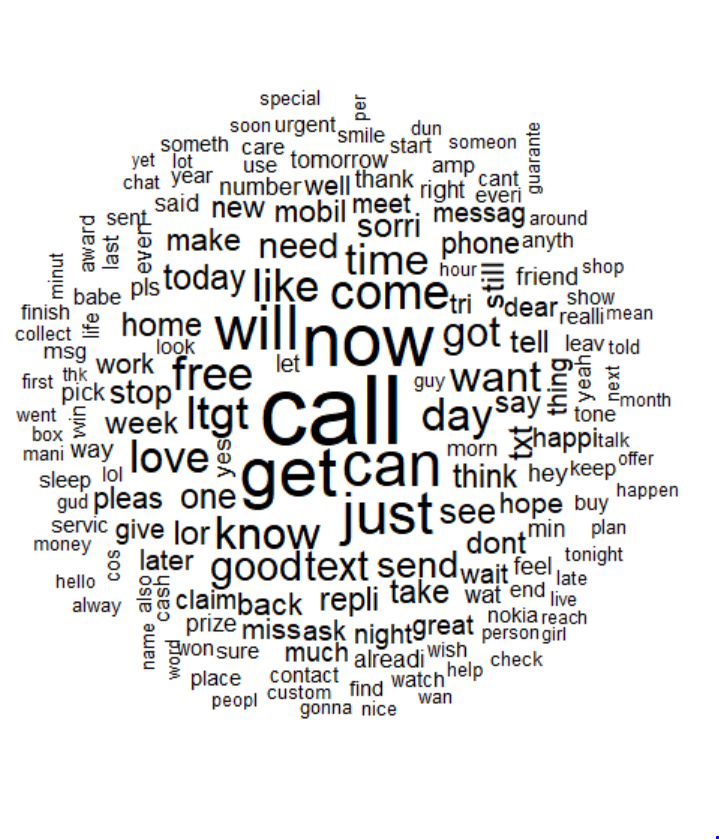
## Visualizing the Text Data

I will start with visualizing the data, finding out which words were used most frequently. The visualization I’ll start with is the wordcloud, to do this I will need ot bring in the wordcloud package into my RStudio.



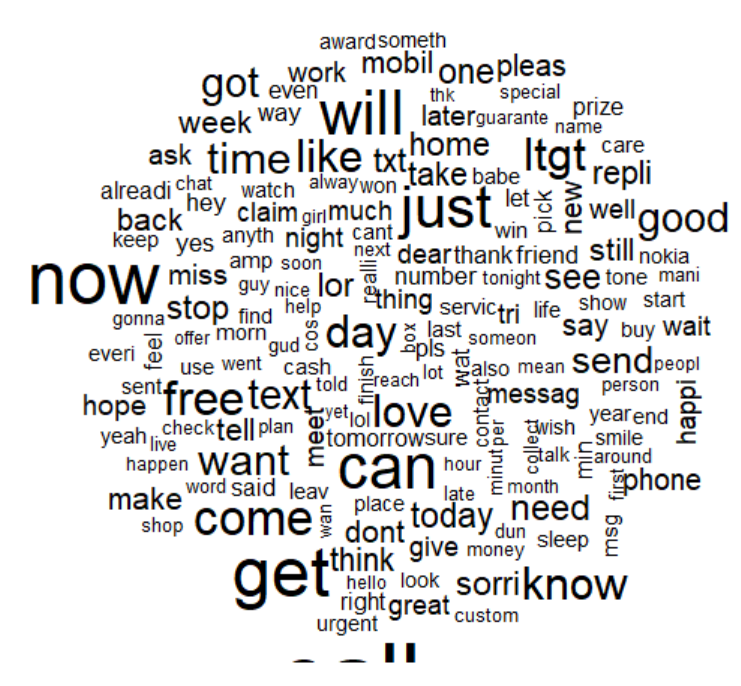
From there I can start using the wordcloud command:



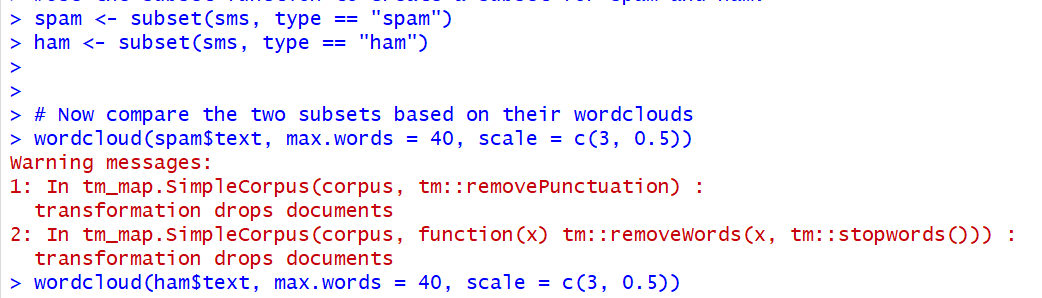


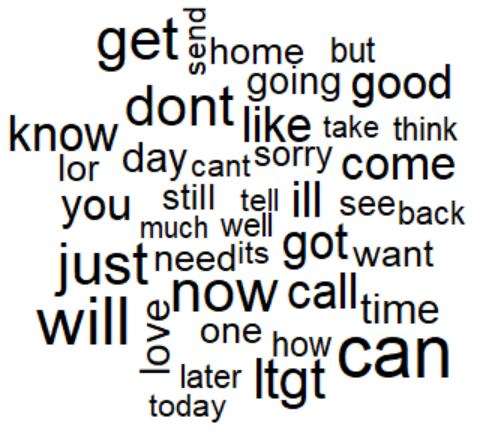
I wanted to see it randomized so I switched it to random.order = TRUE as well:





Now I will create a subset each for spam and ham and compare the wordclouds that they construct.

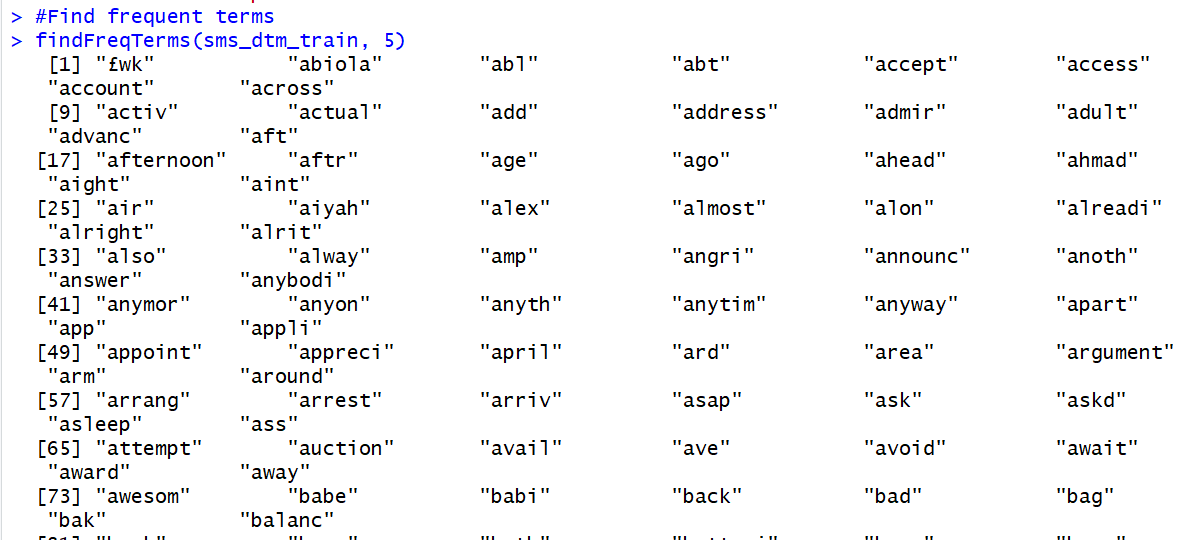




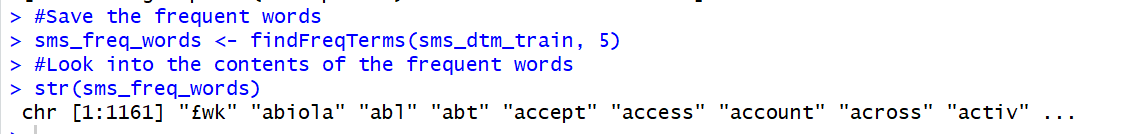
Spam is on the left, the ham wordcloud is on the right. It apparent that the two word clouds have very different words associated within them. Words like free, call, or now are prominent in the spam cloud while words like don’t, get, or will are prominent in the ham cloud.

The next step in this analysis will be to create indicator features for frequent words within the data. This is the final step in the data preparation process.

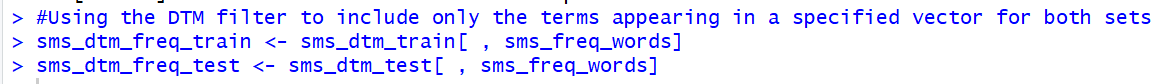
Start by finding all the frequently used words in the dataset.



Next save the frequent words.

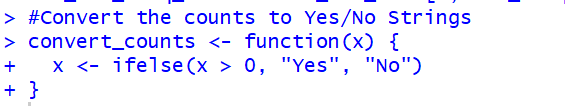


Next we set the DTM filter to only have the terms we expect to see in the specified vector.

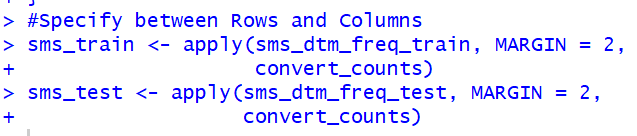


Both sets of data have 1161 features that correspond with words appearing in at least 5 messages.

Since Naïve Bayes classifier is typically trained on data with categorical features we have a bit of a problem. The reason is the cells are in the sparse matrix are numeric and measure the number of times a word appears in a message, so that will need to be changed to a categorical variable that simply indicates Y/N for if the word appears.



Now we will bring in the MARGIN command to specify between rows and columns.



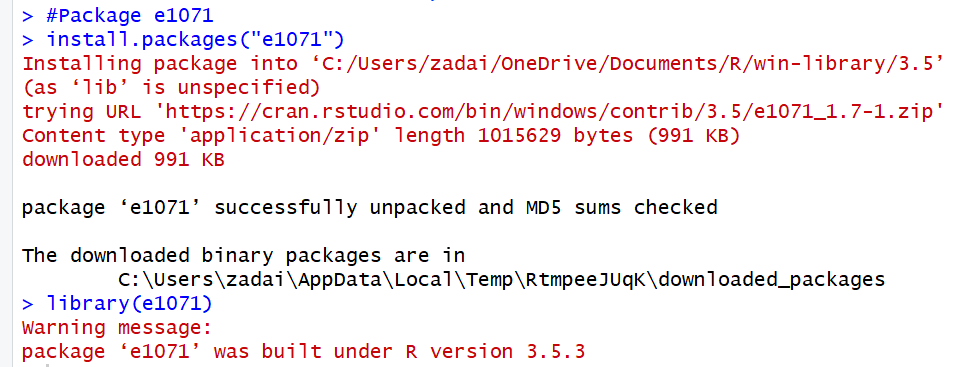
Our result is now we have two character matrixes that are populated with cells stating either “Yes” or “No”.

With the data transformations complete is now time to train the model on our data.

# Training and Evaluating a Model

It is time to train the Naïve Bayesian model to the SMS data! The algorithm will use either the presence or absence of words to estimate the probability a given SMS message is either spam or ham.

Before getting started, I need to grab the e1071 library package for RStudio.



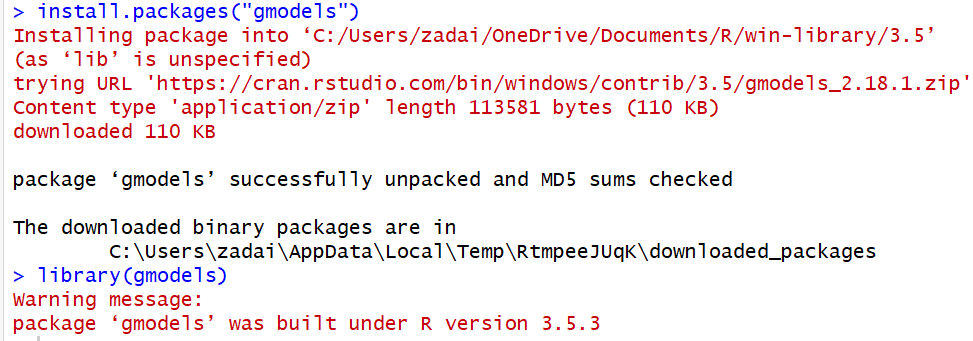
Now let’s set up our classifier on the training matrix.



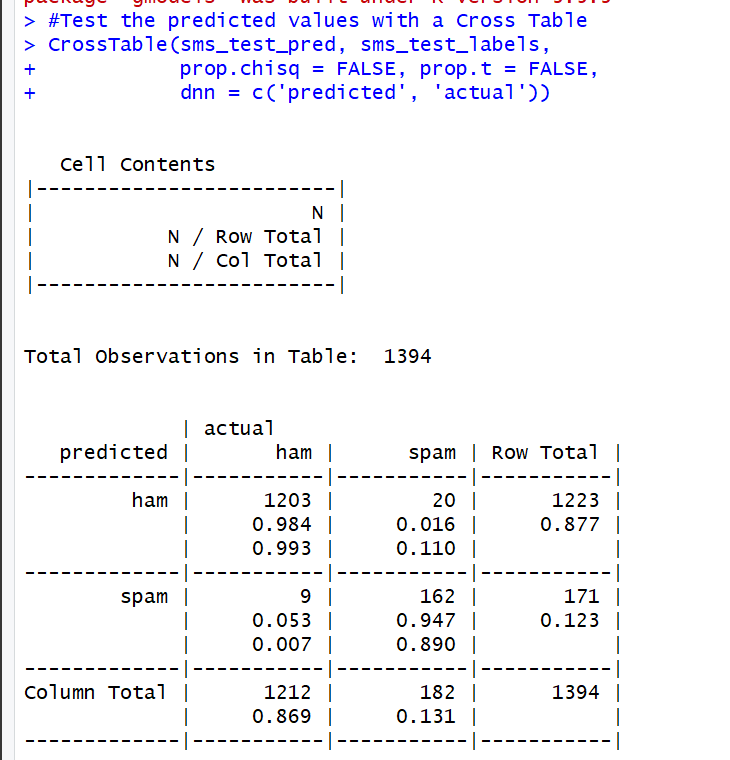
Next we will use the predict function to make predictions and then store those predicted values in sms\_test\_pred.



Using a Cross Table I will compare the values of the predicted but first I need to bring in the package gmodels.



Now using the CrossTable command, I will test the predicted values.

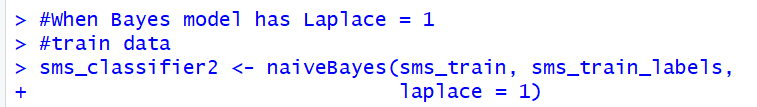


From the look of our cross table, I can see that only 29 out of 1394 were incorrectly classified, 20 of those spam when they were ham and 9 vice-versa. That gives me a percentage of 97.9% accuracy with this model in total. Out of the spam SMS messages I was 89% accurate and from the ham SMS messages I was 99.2% accurate.

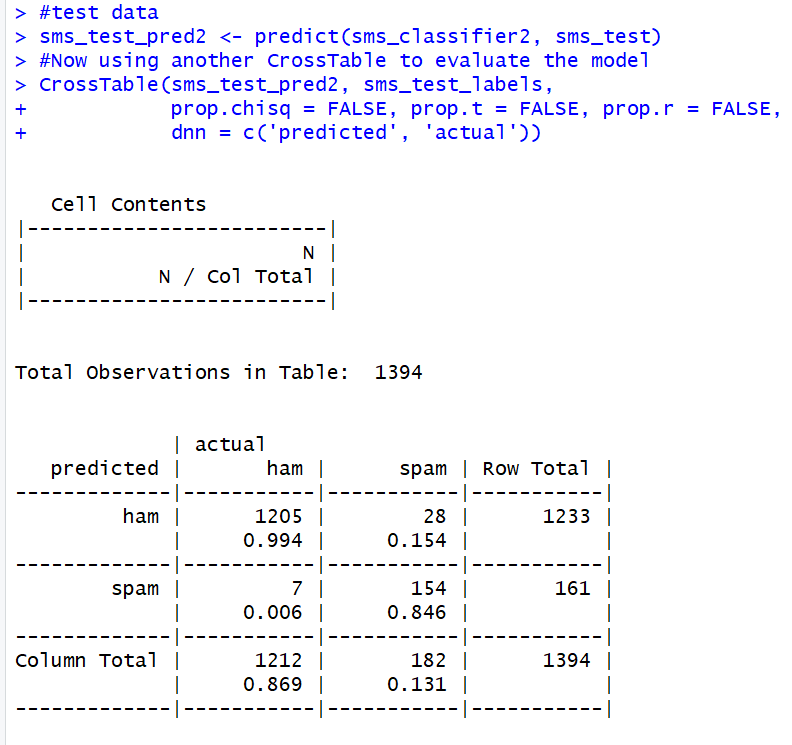
## Improving the Model

Every model has some room for improvement and just because one message has a trigger word to predict as a spam SMS doesn’t always mean that is the case.

Building another Naïve Bayes model but this time set laplace = 1.



Now that laplace = 1 is established we can test the model again and measure its accuracy with another cross table.



When adding the Laplace estimator and having it equal 1 there was a smaller number of false positives that occurred. Though only two less that is better than the first test which had 9 so it is decent improvement by the model to get rid of false positives. You never want to tweak a good fitting model too much in risk of the model than becoming either too aggressive or too conservative and since this is a good fitting model it would be tough to make anymore changes to it.

## Summary

In this assignment for MSDS 680, I got to go through a problem using Naïve Bayes theorem in RStudio and apply it onto some SMS message data which needed to be determined if it was a message that was spam or ham. The process throughout the assignment was long because there was a lot of text manipulation to be done to the data but it was fun to do that as well as set up the model and have it do a good job predicting values.