**ASSIGNMENT-8 (CS-553 PROJECT)**

**Twitter Sentiment Analysis :-**

The increasing popularity of the micro-blogging sites like Twitter, which facilitates users to exchange short messages (aka *tweets*) is an impetus for data analytics tasks for varied purposes, ranging from business intelligence to nation security. Twitter is being used by a large number of users for events update and sentiment expression. Since tweets are generally unstructured in nature and do not follow grammatical structures, parsing techniques generally do not work well due to incorrect parts-of-speech assignment to individual words.

For this project I propose to use Twitter API along with R’s **twitteR** library to extract tweets from twitter and then apply Text-Mining on the extracted twitter data using R’s **tm** library.

The plan to use the ETL flow model to proceed with the project.

**Extraction:-**

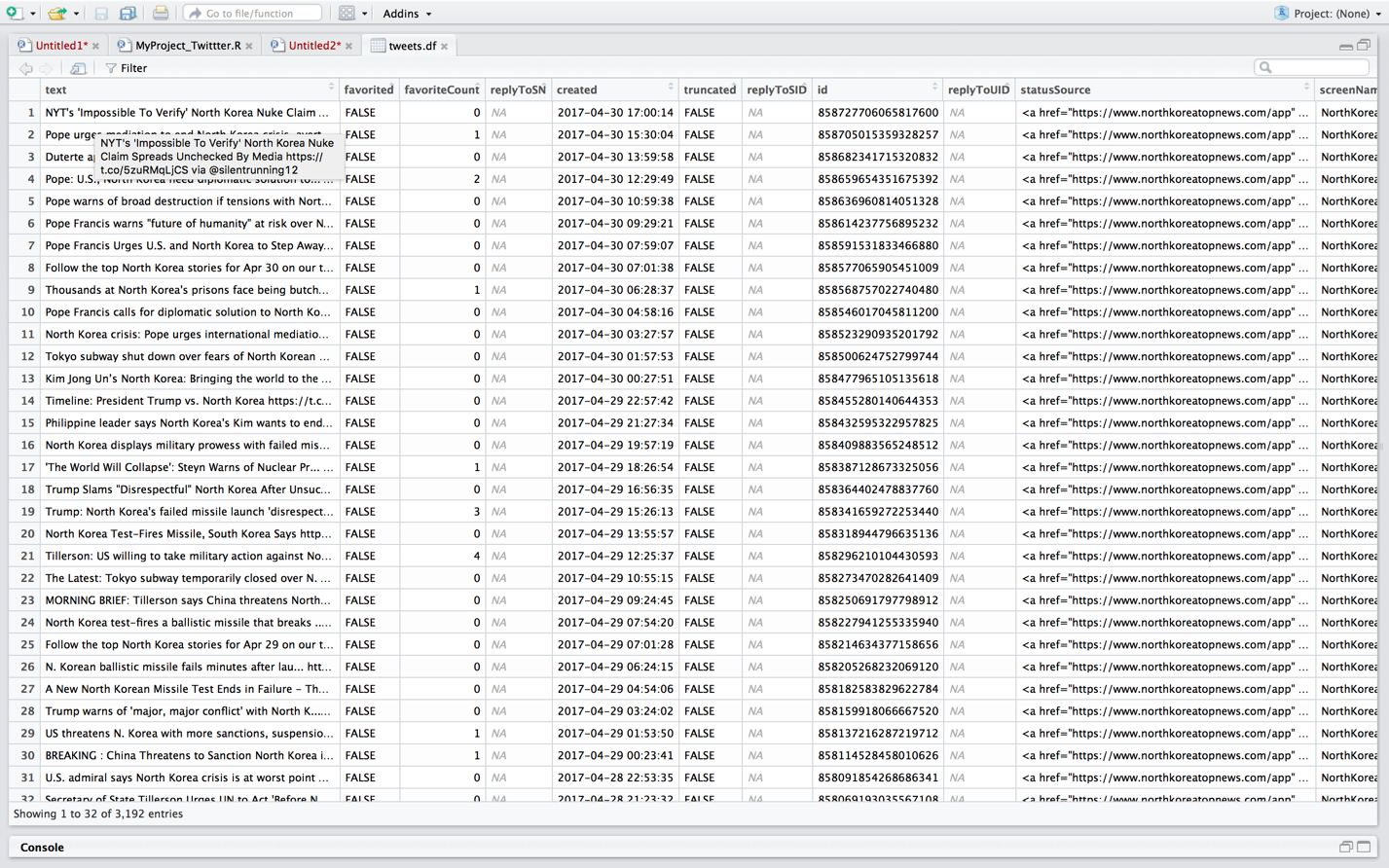
****

The above code basically extracts tweets from Twitter using the Twitter API. For extracting the tweets we need to provide some Access Token which I have provided in the **setup\_twitter\_oauth()** function.

Then I have extracted 3200 tweets from twitter (3200) is the max limit for extracting tweets from twitter at a given time.

By default, the tweets are in **list** format but for analysis purpose we need to first convert it into **dataframe** and then select only the column that contains text data for creating the text corpus.

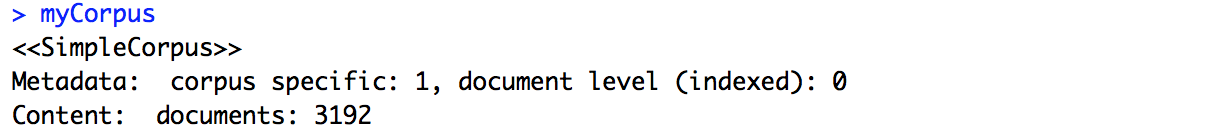
The following is snippet of the dataset that is created which contains 3192 tweets.



**Transformation:-**

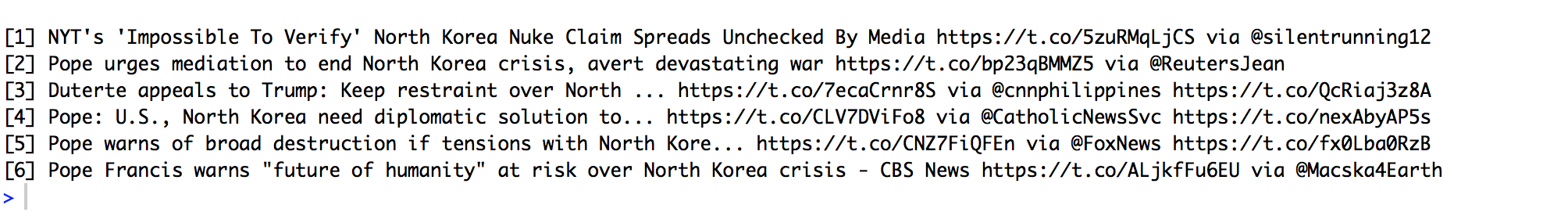
In the transformation part I will be performing all the data cleaning process, (since the data is textual data) the cleaning process involves the following:

* Converting the data to lower case
* Removing the urls
* Removing the punctuation marks
* Removing Stop words
* Removing English language space
* Removing extra white space
* I will include the code for stemming but I don’t find it of much relevance for my project so will be avoiding its use.

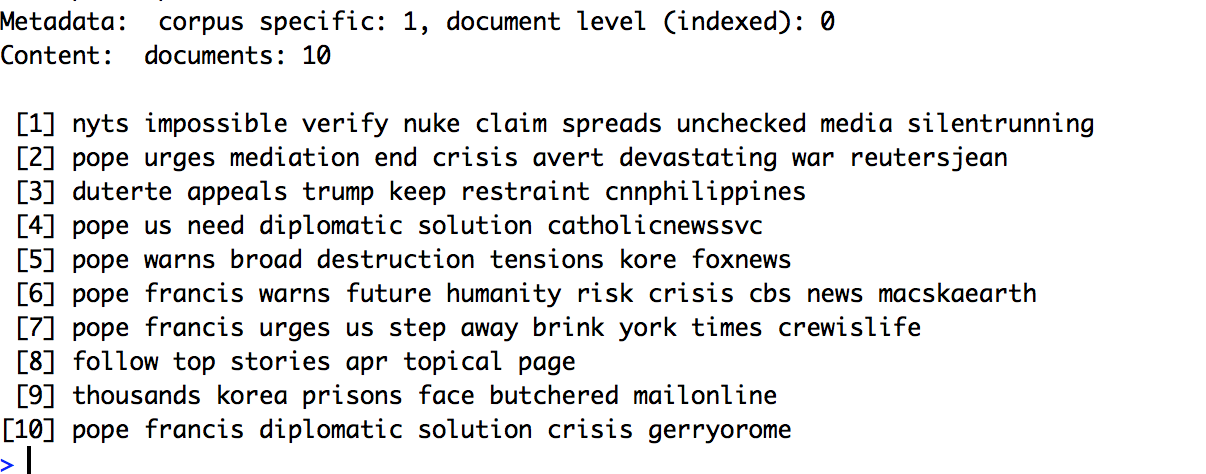


You can see that your corpus consists of 3192 documents.

**The flowing snippet shows the first 6 rows of the original un-cleaned data.**



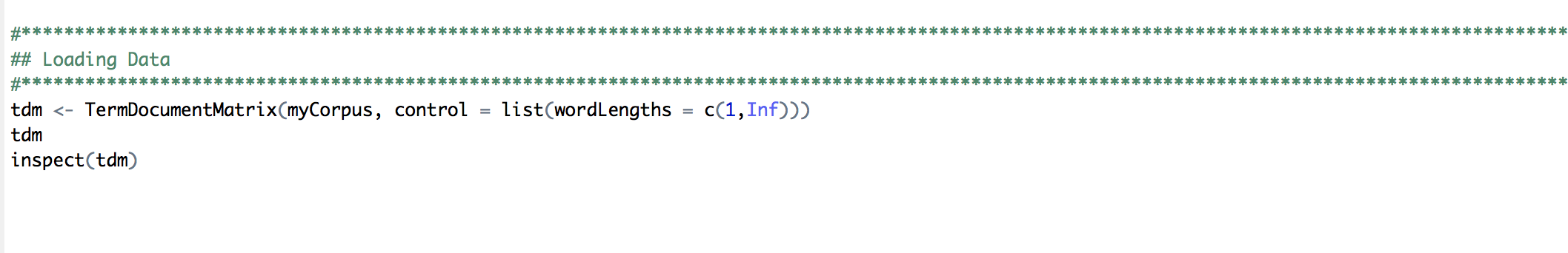
**The flowing snippet shows the first 10 rows of the cleaned data.**

****

We can see how the corpus has been converted into lower case we have got rid of all the un-wanted punctuations and urls.

**Note:- I have also removed north korea from the corpus because it was obvious that the term will occur in almost all the tweets and it might hamper the prediction or could be too over fitted.**

**Loading:-**

****

In the loading part I am creating a **TermDocumentMatrix**

Term-Document Matrix

**Description**

Constructs or coerces to a term-document matrix or a document-term matrix.

**Usage**

TermDocumentMatrix(x, control = list())

DocumentTermMatrix(x, control = list())

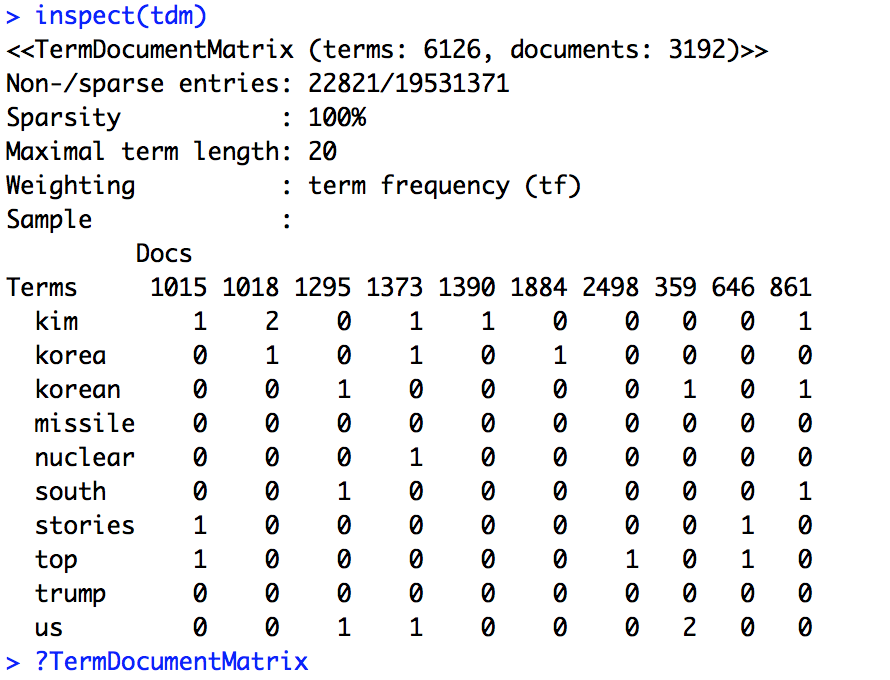
as.TermDocumentMatrix(x, ...)

as.DocumentTermMatrix(x, ...)

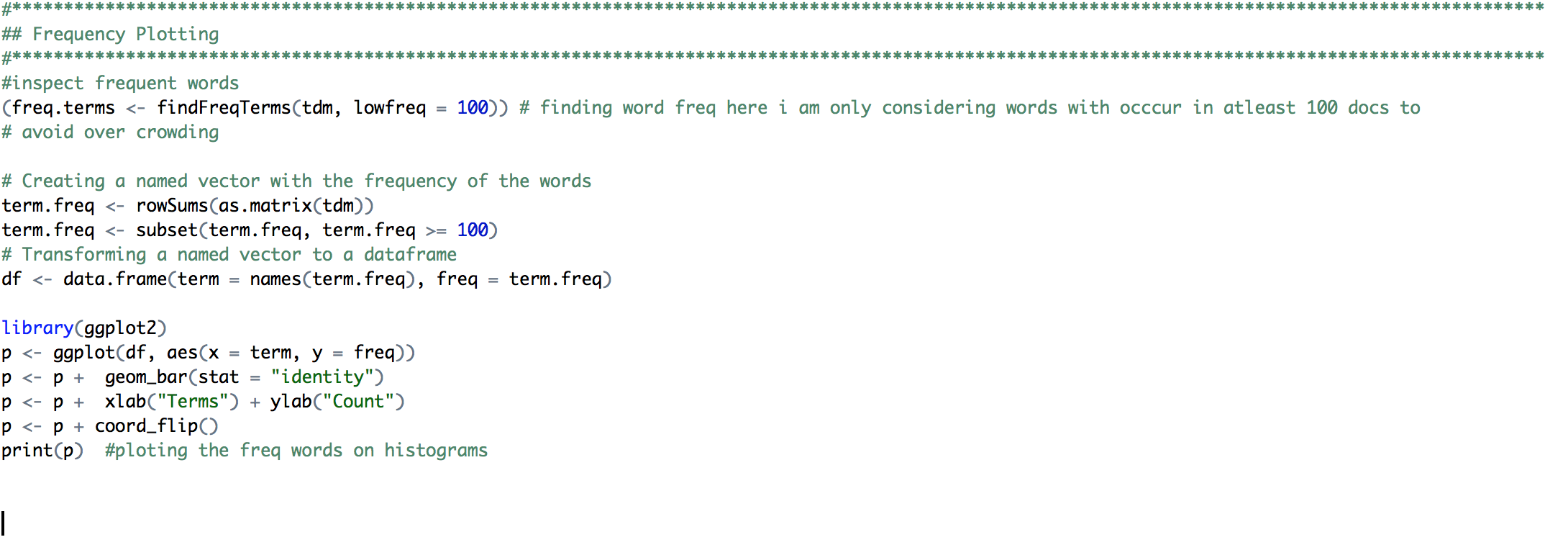
**Arguments**

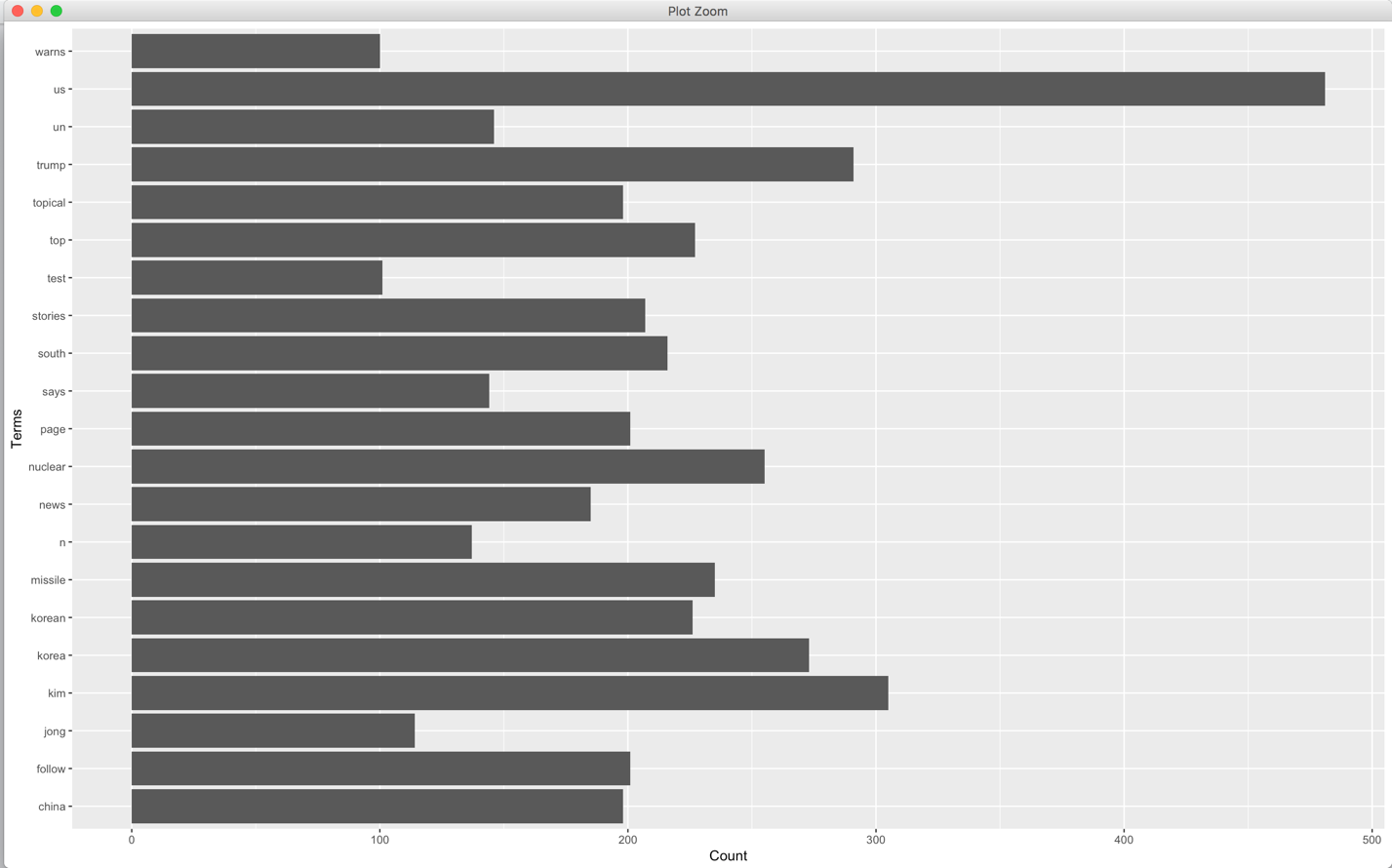
|  |  |
| --- | --- |
| x | a corpus for the constructors and either a term-document matrix or a document-term matrix or a [simple triplet matrix](http://127.0.0.1:9495/help/library/slam/html/matrix.html) (package **slam**) or a [term frequency vector](http://127.0.0.1:9495/help/library/tm/help/termFreq) for the coercing functions. |
| control | a named list of control options. There are local options which are evaluated for each document and global options which are evaluated once for the constructed matrix. Available local options are documented in [termFreq](http://127.0.0.1:9495/help/library/tm/help/termFreq) and are internally delegated to a [termFreq](http://127.0.0.1:9495/help/library/tm/help/termFreq) call.  This is different for a [SimpleCorpus](http://127.0.0.1:9495/help/library/tm/help/SimpleCorpus). In this case all options are processed in a fixed order in one pass to improve performance. It always uses the Boost Tokenizer (via **Rcpp**) and takes no custom functions as option arguments.  Available global options are:  bounds  A list with a tag global whose value must be an integer vector of length 2. Terms that appear in less documents than the lower bound bounds$global[1] or in more documents than the upper bound bounds$global[2] are discarded. Defaults to list(global = c(1, Inf))(i.e., every term will be used).  weighting  A weighting function capable of handling a TermDocumentMatrix. It defaults to weightTffor term frequency weighting. Available weighting functions shipped with the **tm** package are [weightTf](http://127.0.0.1:9495/help/library/tm/help/weightTf), [weightTfIdf](http://127.0.0.1:9495/help/library/tm/help/weightTfIdf), [weightBin](http://127.0.0.1:9495/help/library/tm/help/weightBin), and [weightSMART](http://127.0.0.1:9495/help/library/tm/help/weightSMART). |
| ... | the additional argument weighting (typically a [WeightFunction](http://127.0.0.1:9495/help/library/tm/help/WeightFunction)) is allowed when coercing a simple triplet matrix to a term-document or document-term matrix. |

The following snippet shows the numers of terms = **6126** in documents= **3192** also it shows the Non-/sparse enteries **22821/19531371** along with the weighing term used i.e **term freq** and other info like sparsity, max length along with the sample of 10 terms.



**Frequency Plotting:-**

****

****

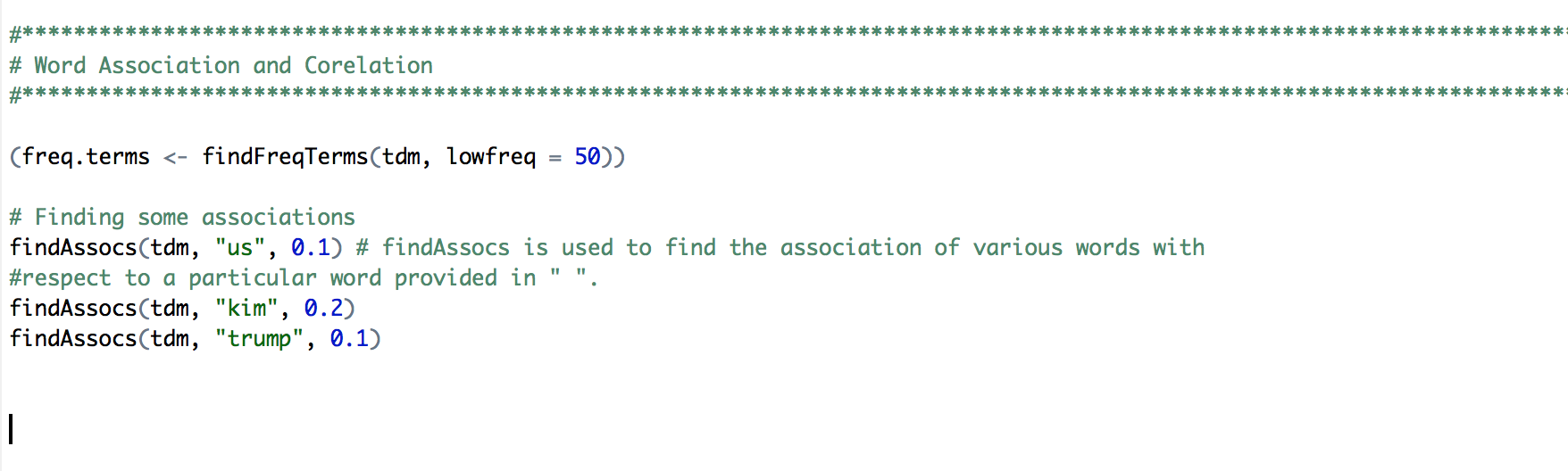
The frequency plot basically plots the term with frequency more than 100 i.e the terms occurring in atleast 100 document ( I kept the lower threshold as 100 coz I wanted to the list of significant terms which are currently occurring in users tweets in order to predict or conclude on some topic)

From my analysis I can see that term **us** has a significant place in tweets and is twitted in 481 tweets followed by terms like **trump**, **nuclear**, **kim**, **china**, etc. And It does make sense because these are the most prominent terms which I assumed to occur considering the current scenarios in “North Korea”.

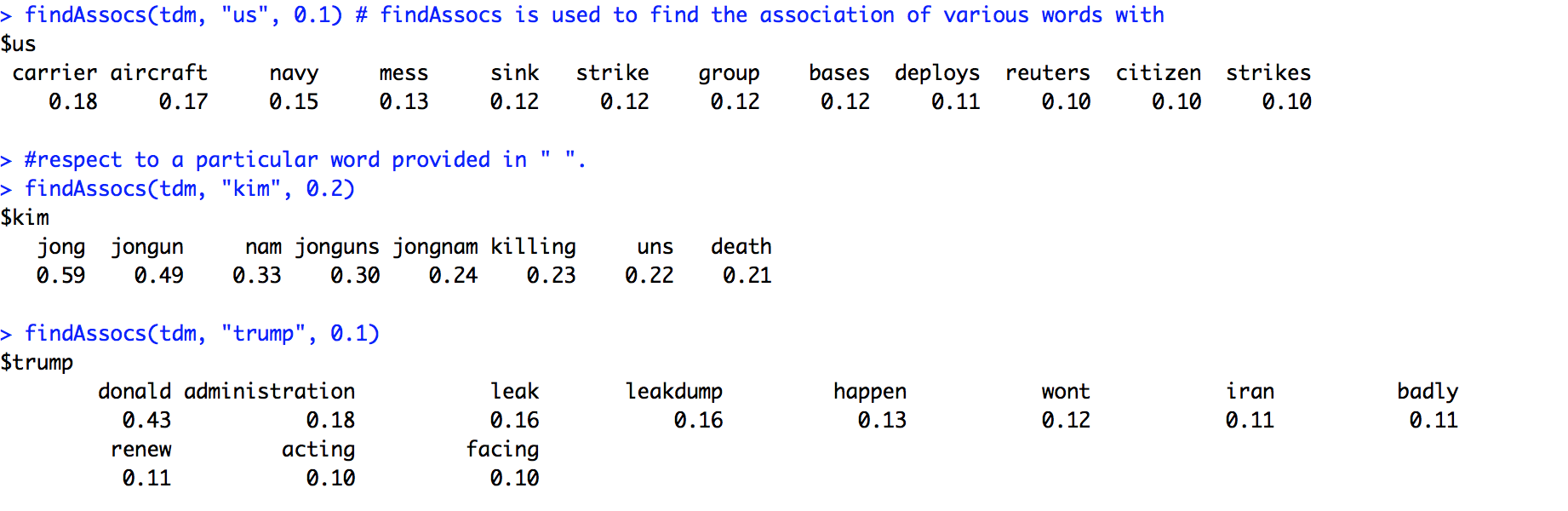
**Word Association: -**

Word Association is basically used to see how closely two terms are associated with each other.

It also plays a significant role in text analysis because it provides us with a list of closely associated words and by learning the association we can learn about the realtion between two terms.



From the above snippet I tried to find the association of the terms us, kim, trump with other terms



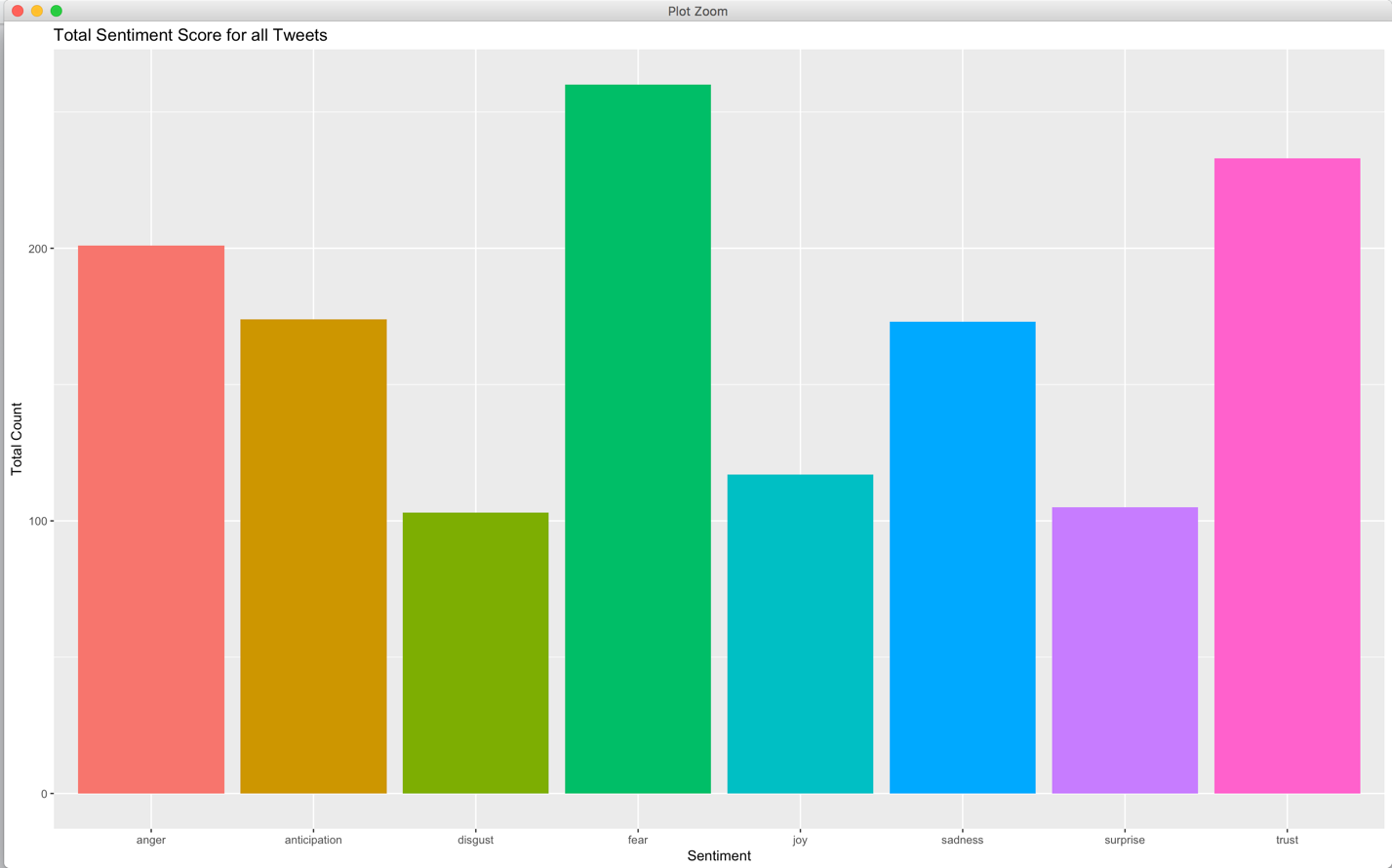
Here is the output that depicts that the term us is associated with terms like carrier, aircraft , strike navy and it makes sense coz us had recently made air strikes in Syria and also sent its aircraft cariers to Syrian coast, but its interesting to see how these topic correlates with “north korea”.

**Sentiment Analysis :-**

It is basically the process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc., is positive, negative, or neutral.

For prediting the sentiment analysis I am using R’s syuzhet, lubridate, ggplot2 libraries.

What R does is basically learns from the terms in our corpus and classifies it into 7-10 categories namely, positive, negative, fear, anger, trust, surprise, sadness, joy, disgust.



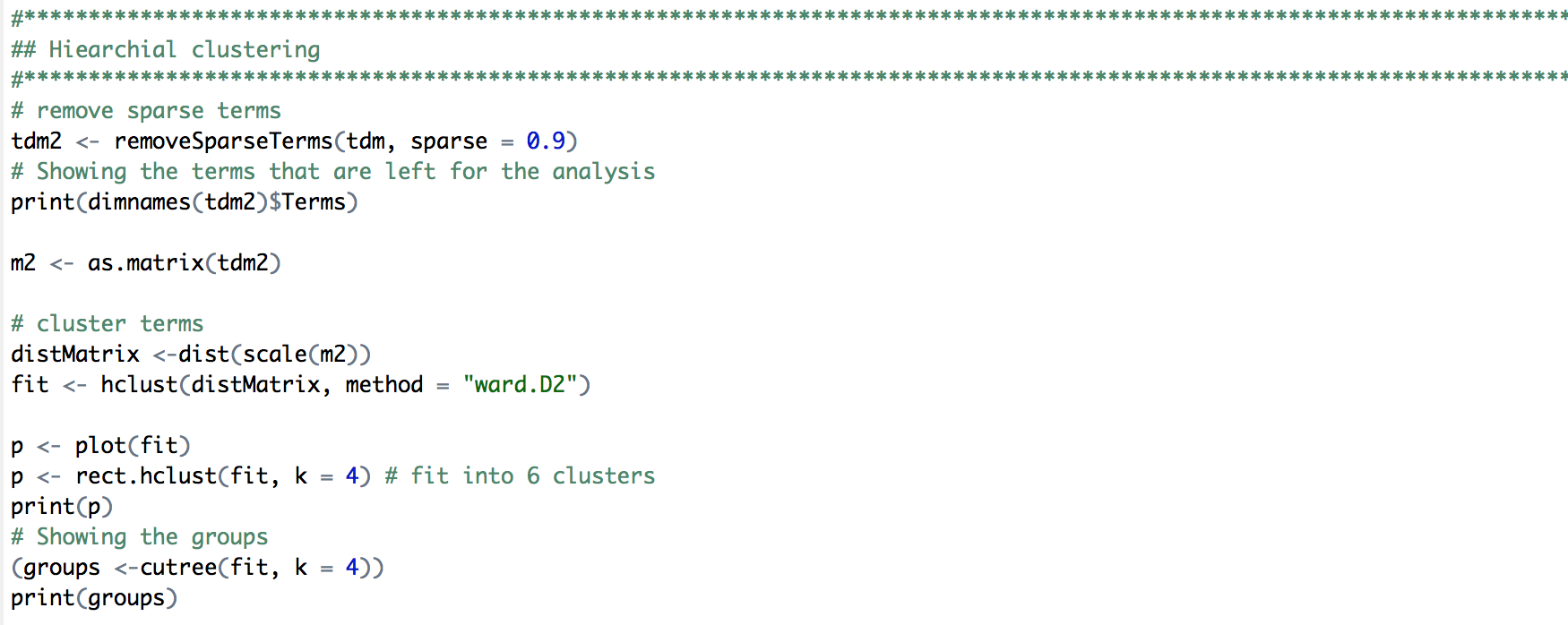
Now in the above plot we can see peoples sentiments on the topic “north korea”, majority of the crowd is in fear of a world war 3 , people are angry, there is sadness but it is strange to see that there is also a high percent of trust factor involved in the peoples sentiments .

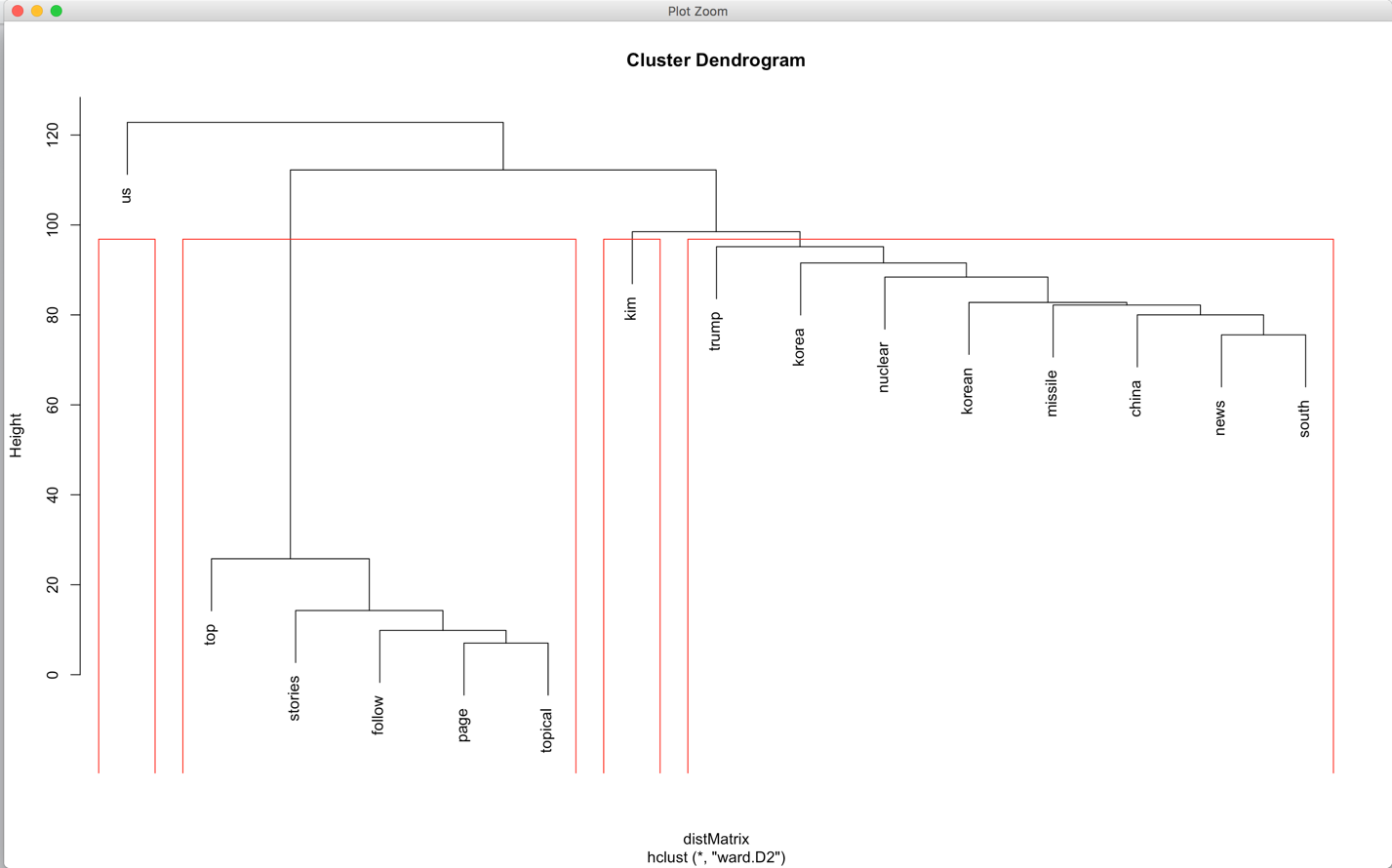
**Hierchial Clustering:-**

In [data mining](https://en.wikipedia.org/wiki/Data_mining) and [statistics](https://en.wikipedia.org/wiki/Statistics), **hierarchical clustering** (also called **hierarchical cluster analysis** or **HCA**) is a method of [cluster analysis](https://en.wikipedia.org/wiki/Cluster_analysis) which seeks to build a [hierarchy](https://en.wikipedia.org/wiki/Hierarchy) of clusters. Strategies for hierarchical clustering generally fall into two types:[[1]](https://en.wikipedia.org/wiki/Hierarchical_clustering#cite_note-clusteringMethods-1)

* **Agglomerative**: This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
* **Divisive**: This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

**In our case we are using divisive clustering using dendograms.**

****

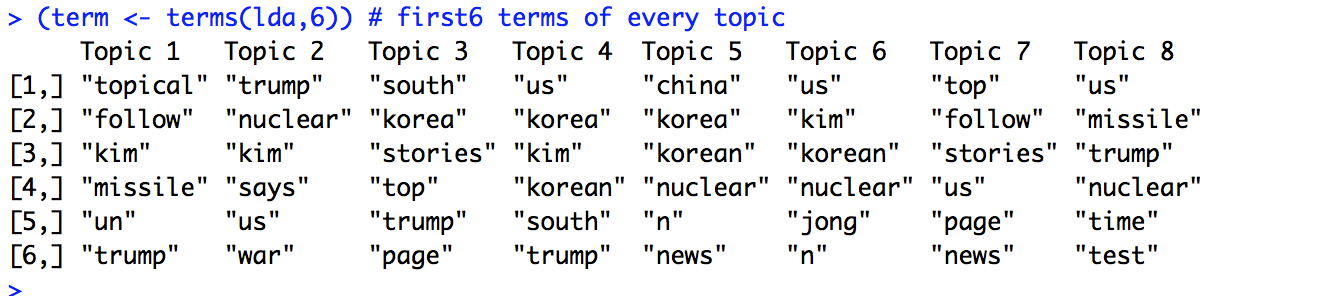
****

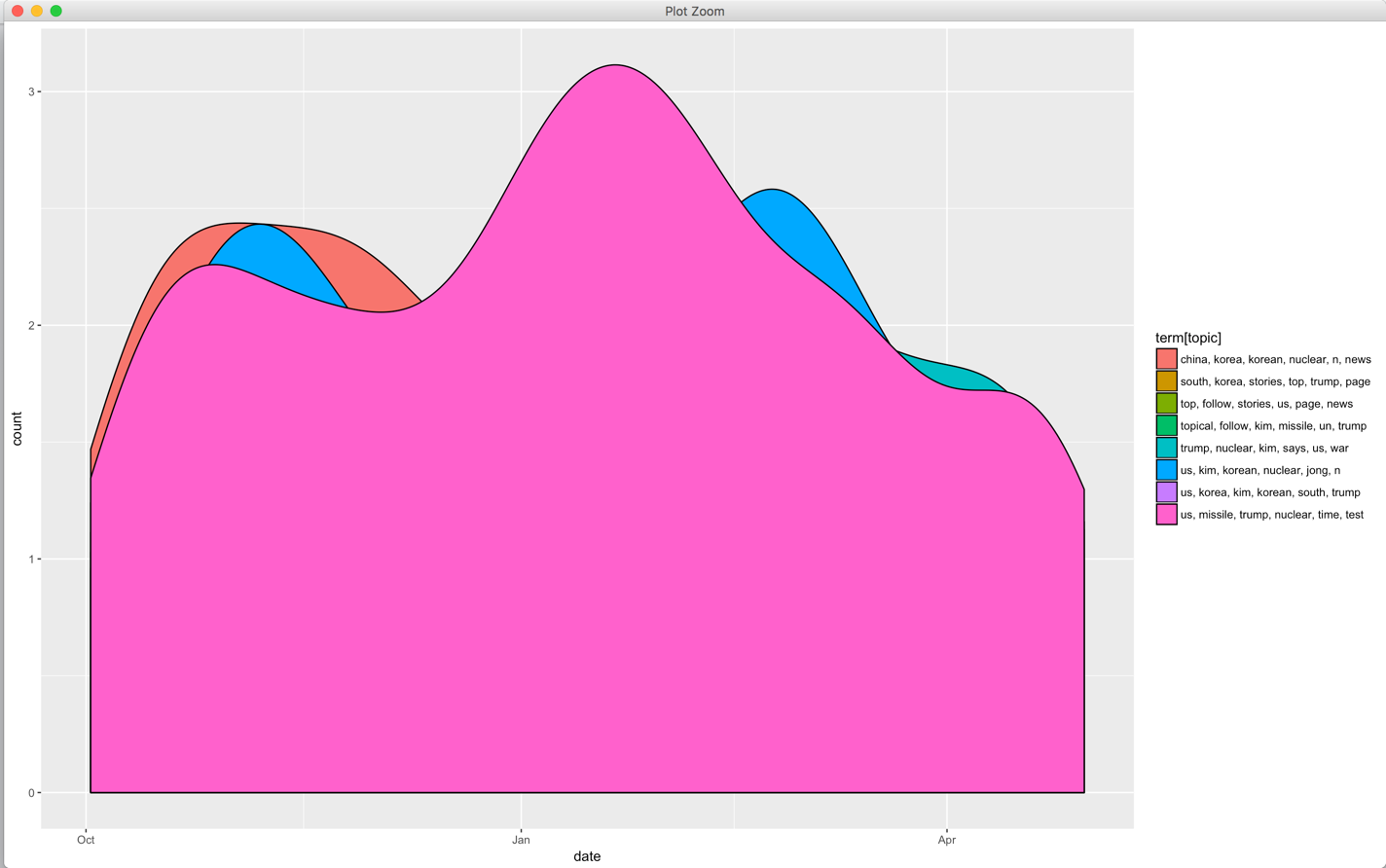
From the above snippit you can see how the terms kim, trump, korea, nuclear, missile , china are closely related to each other and thus clustered in kind of common cluster.

**Topic Modeling :-**

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning) and [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing), a **topic model** is a type of [statistical model](https://en.wikipedia.org/wiki/Statistical_model) for discovering the abstract "topics" that occur in a collection of documents. Topic modeling is a frequently used text-mining tool for discovery of hidden semantic structures in a text body. Intuitively, given that a document is about a particular topic, one would expect particular words to appear in the document more or less frequently: "dog" and "bone" will appear more often in documents about dogs, "cat" and "meow" will appear in documents about cats, and "the" and "is" will appear equally in both. A document typically concerns multiple topics in different proportions; thus, in a document that is 10% about cats and 90% about dogs, there would probably be about 9 times more dog words than cat words. The "topics" produced by topic modeling techniques are clusters of similar words. A topic model captures this intuition in a mathematical framework, which allows examining a set of documents and discovering, based on the statistics of the words in each, what the topics might be and what each document's balance of topics is.

For our project I have used LDA i.e Latent Dirichlet Association which basically takes the most frequent occurring terms from the documents and then creates topic models based on those terms.





The plot shows that the LDA function has created 8 topic models.