# MS4S09 Coursework 1 - 2020/21

17076749

Mark Baber

06/01/2021

## 1 - Introduction

This report will be my assignment for MS4S09 where the exploration of data mining, sentiment analysis and topic modelling will be used to analyse a dataset for an airline.

Before anything, lets put any of the packages used to carry out the analysis below.

Now load the packages which will be used.

Start by reading in the data.

### Get a better picture

It is good practice to understand the data, this can be done by looking at a summary, head, class and structure of the data. of the data

```
##     tweet_id sentiment        airline retweet_count
## 1 Tr_tweet_1   neutral Virgin America             0
## 2 Tr_tweet_2  positive Virgin America             0
## 3 Tr_tweet_3   neutral Virgin America             0
## 4 Tr_tweet_4  negative Virgin America             0
## 5 Tr_tweet_5  negative Virgin America             0
##
## 1                                                                                            @Virgin
## 2                                                       @VirginAmerica plus you’ve added commercials
## 3                                                        @VirginAmerica I didn’t today... Must mean
## 4 @VirginAmerica it’s really aggressive to blast obnoxious "entertainment" in your guests’ faces &am
## 5                                                                        @VirginAmerica and it’s a r
##               tweet_created tweet_location              user_timezone
## 1 2015-02-24 11:35:52 -0800                Eastern Time (US & Canada)
## 2 2015-02-24 11:15:59 -0800                Pacific Time (US & Canada)
## 3 2015-02-24 11:15:48 -0800      Lets Play Central Time (US & Canada)
## 4 2015-02-24 11:15:36 -0800                Pacific Time (US & Canada)
## 5 2015-02-24 11:14:45 -0800                Pacific Time (US & Canada)
```

Most of this dataset is made up of characters which can’t really be plotted yet. So lets instead look at the different types of airlines within the dataset.

```
## [1] "Virgin America" "United"         ""
## [1] 1
```

```
## [1] 0
```

```
## [1] "Virgin America" "United"
```

Within the dataset there were 3 airlines, albeit one is empty. So above the empty dataset was found and removed from the dataset with a built in function called drop_na. After removing the na value, the unique airlines are then counted again as a sanity check.

## 2 - Text Mining

To ensure the next few steps are easier, lets copy the dataframe into a corpus, which will allow for manipulating with the function getTransformations() which is from the tm package..

```
## [1] "removeNumbers"     "removePunctuation" "removeWords"
## [4] "stemDocument"      "stripWhitespace"
```

Next lets look at cleaning up the data a bit by: 1 - changing it to lowercase 2 - remove numbers 3 - remove punctuation 4 - remove common stop words 5 - stem the document has been left out due to changing united to unit 6 - strip white space

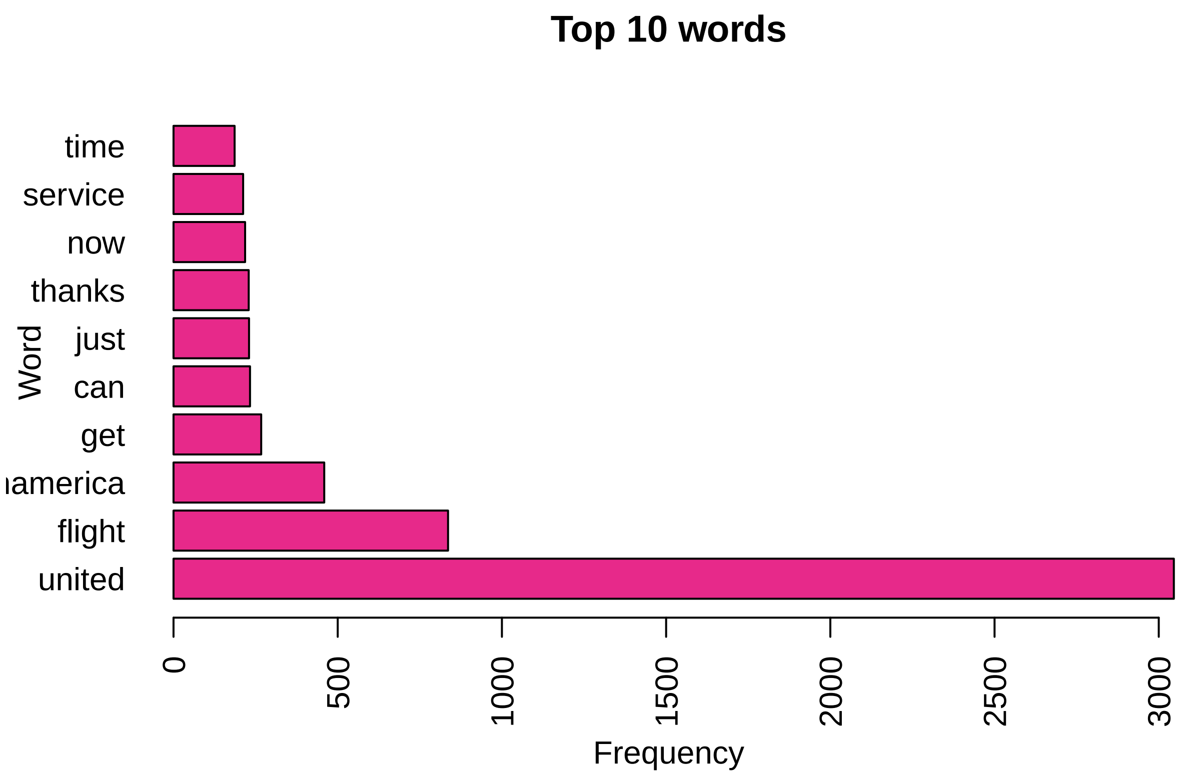
Now to create a function to transform the text, this will be a custom function which we learnt from lecture. This function toSpace() will transform/manipulate the content which is given to the function, and replace the content within ' " " ' to a blank space.

- example "/" will change to " "

Next lets create a term document matrix which is a matrix of all the terms within the document (terms being words).

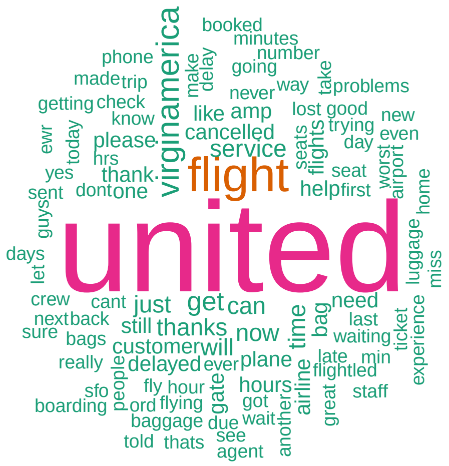
```
##                        word freq
## united               united 3046
## flight               flight  836
## virginamerica virginamerica  459
## get                     get  267
## can                     can  233
## just                   just  230
## thanks               thanks  229
## now                     now  218
## service             service  212
## time                   time  186
```

Now lets plot these top ten words and have a look at them within a bar plot.



Whilst showing this in a barplot can be good, it would look like some words would need to be pre-processed more when moving to the next step, sentiment analysis. The words such as get, can, just don't really add any value, but we can clearly see people are tweeting to united a lot more than virginamerica.

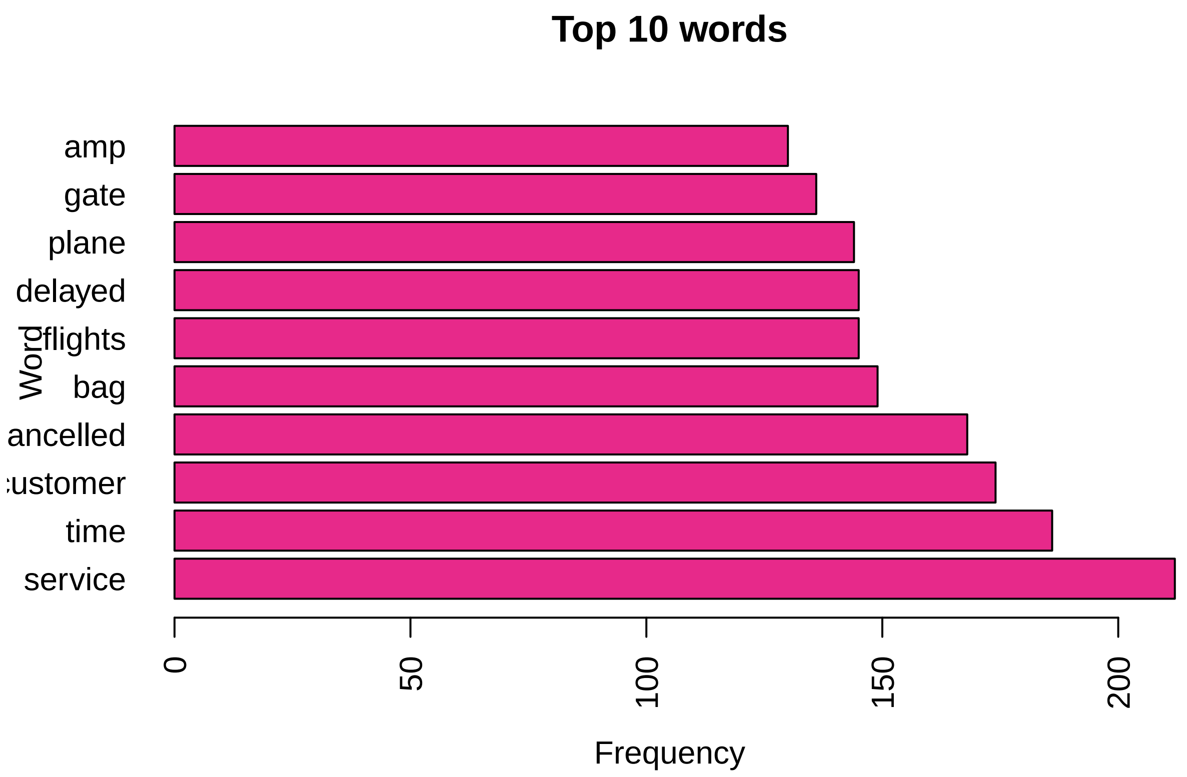
For a more interesting chart, lets look at making a wordcloud, including all of the words with the frequency representing the words size.



Now that we have seen the top words and a lovely chart, lets see how it changes when removing the 2 companies names and flight from the dataset. As these 3 are very much skewing the charts.

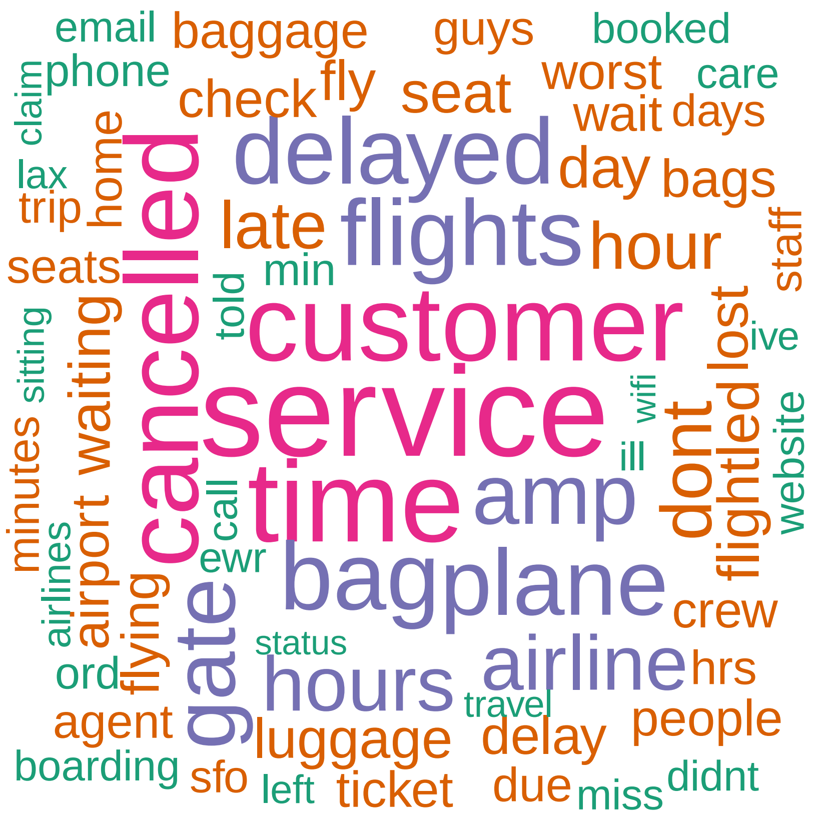
```
## Joining, by = "word"
```

In the section above, there was a tibble which contained 3 custom stopwords, which was then used to remove these stop words from the dataframe d. Now to plot the dataset again, to see how it has been affected with the removal of the custom stopwords.



This new chart shows a more evenly distributed frequency for the words, with server being the top word. This could easily be good server, or bad service.

Next lets take a look at the wordcloud again.



This word cloud doesn't look at nice as the first one, but here we can see the words are similar in size which shows how much the custom stop words were throwing off the charts.

## 3 - Sentiment Analysis

This section will look at using Sentiment Analysis techniques to measure the sentiment per word/sentence within the dataset.

```
##     tweet_id sentiment        airline retweet_count
## 1 Tr_tweet_1   neutral Virgin America             0
## 2 Tr_tweet_2  positive Virgin America             0
## 3 Tr_tweet_3   neutral Virgin America             0
## 4 Tr_tweet_4  negative Virgin America             0
## 5 Tr_tweet_5  negative Virgin America             0
##
## 1                                                                                             @Virgin
## 2                                                           @VirginAmerica plus you've added commercials
## 3                                                            @VirginAmerica I didn't today... Must mean 
## 4 @VirginAmerica it's really aggressive to blast obnoxious "entertainment" in your guests' faces &am
## 5                                                                             @VirginAmerica and it's a r
##              tweet_created tweet_location             user_timezone
## 1 2015-02-24 11:35:52 -0800                Eastern Time (US & Canada)
## 2 2015-02-24 11:15:59 -0800                Pacific Time (US & Canada)
## 3 2015-02-24 11:15:48 -0800      Lets Play Central Time (US & Canada)
## 4 2015-02-24 11:15:36 -0800                Pacific Time (US & Canada)
## 5 2015-02-24 11:14:45 -0800               Pacific Time (US & Canada)
```

Next will be to tokenize the texts column into a new column.

```
##     tweet_id sentiment        airline retweet_count             tweet_created
## 1 Tr_tweet_1   neutral Virgin America             0 2015-02-24 11:35:52 -0800
## 2 Tr_tweet_1   neutral Virgin America             0 2015-02-24 11:35:52 -0800
## 3 Tr_tweet_1   neutral Virgin America             0 2015-02-24 11:35:52 -0800
## 4 Tr_tweet_1   neutral Virgin America             0 2015-02-24 11:35:52 -0800
## 5 Tr_tweet_2  positive Virgin America             0 2015-02-24 11:15:59 -0800
##   tweet_location              user_timezone          word
## 1                Eastern Time (US & Canada) virginamerica
## 2                Eastern Time (US & Canada)          what
## 3                Eastern Time (US & Canada)      dhepburn
## 4                Eastern Time (US & Canada)          said
## 5                Pacific Time (US & Canada) virginamerica
```

Now there is a new column called word which has our tokenized text. Next we will continue to remove the stopwords.

```
## Joining, by = "word"
```

```
##     tweet_id sentiment        airline retweet_count             tweet_created
## 1 Tr_tweet_1   neutral Virgin America             0 2015-02-24 11:35:52 -0800
## 2 Tr_tweet_1   neutral Virgin America             0 2015-02-24 11:35:52 -0800
## 3 Tr_tweet_2  positive Virgin America             0 2015-02-24 11:15:59 -0800
## 4 Tr_tweet_2  positive Virgin America             0 2015-02-24 11:15:59 -0800
## 5 Tr_tweet_2  positive Virgin America             0 2015-02-24 11:15:59 -0800
##   tweet_location              user_timezone          word
## 1                Eastern Time (US & Canada) virginamerica
## 2                Eastern Time (US & Canada)      dhepburn
## 3                Pacific Time (US & Canada) virginamerica
## 4                Pacific Time (US & Canada)         added
## 5                Pacific Time (US & Canada)   commercials
```

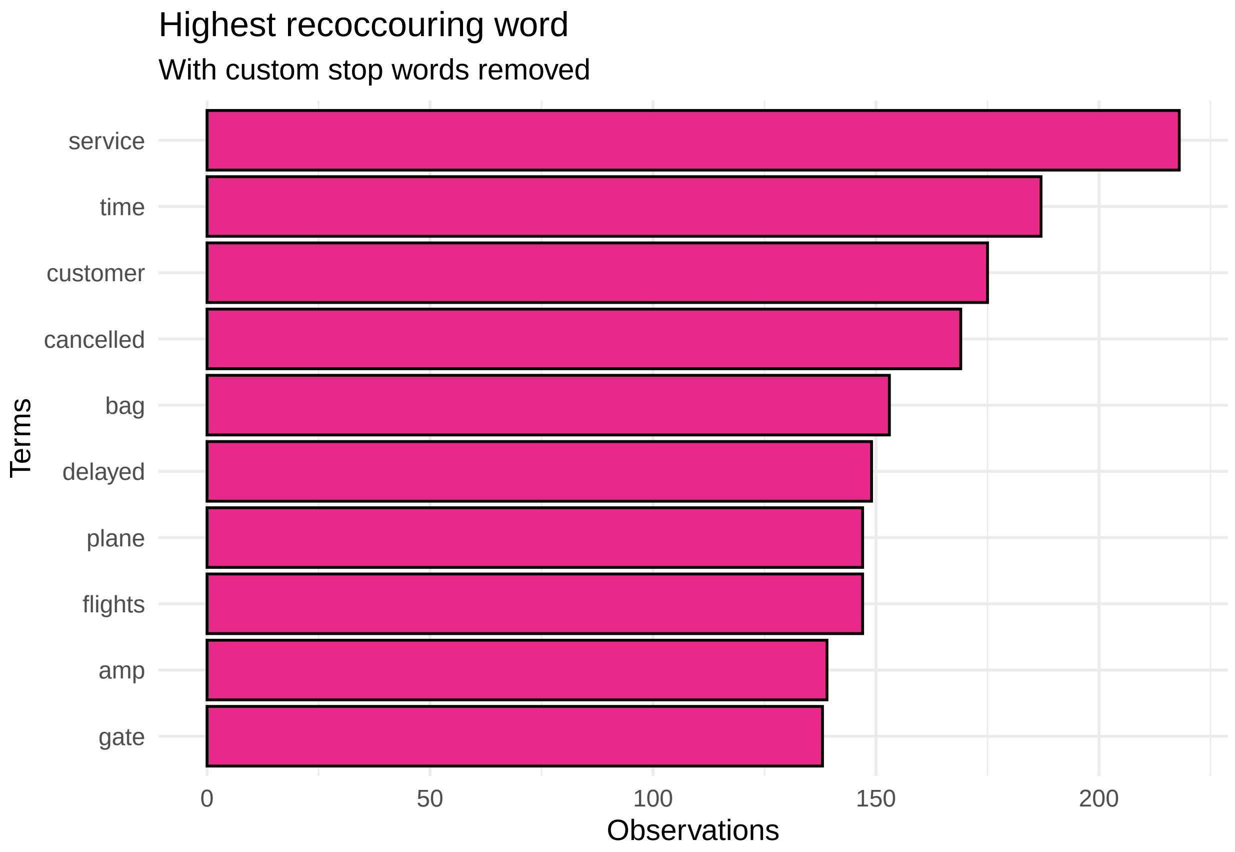
After removing the stop words, lets see what are the most popular words again.

```
##            word    n
## 1        united 3074
## 2        flight  845
## 3 virginamerica  463
## 4          t.co  263
## 5          http  251
```

Above we can see there 3 most popular words are united, flight and virginamerica. Whilst not surprising they could also be omitted to try and get a more evenly distributed dataset. To copy the steps above, lets remove the custom stop words and some other words which don't really make sense. ("t.co" ,"http" "2")

```
## Joining, by = "word"
##        word   n
## 1   service 218
## 2      time 187
## 3  customer 175
## 4 cancelled 169
## 5       bag 153
```

this could also be plotted for a better visual using ggplot.



The plot above shows the top ten most popular words with more than 125 observations, a lot of these words could go either way really. They could be positive or negative, so in the next steps lets look at investigating these words.

```
##        word   n
## 1   service 218
## 2      time 187
## 3  customer 175
## 4 cancelled 169
## 5       bag 153
```

Now lets figure out the actual sentiment for the dataset.

There are many lexicon libraries which are out there, but for this report lets focus on 2 and maybe adding a third party library later. These libraries are ‘bing’ and ‘nrc’, lets check the sentiment for the dataset using these 2 libraries.

## bing

```
## Joining, by = c("sentiment", "word")
```

```
##      word   n
## 1 delayed 143
## 2    lost  69
## 3   delay  68
## 4   worst  67
## 5    miss  44
```

The most popular words within our dataset against the bing library.

## nrc

```
## Joining, by = c("sentiment", "word")
```

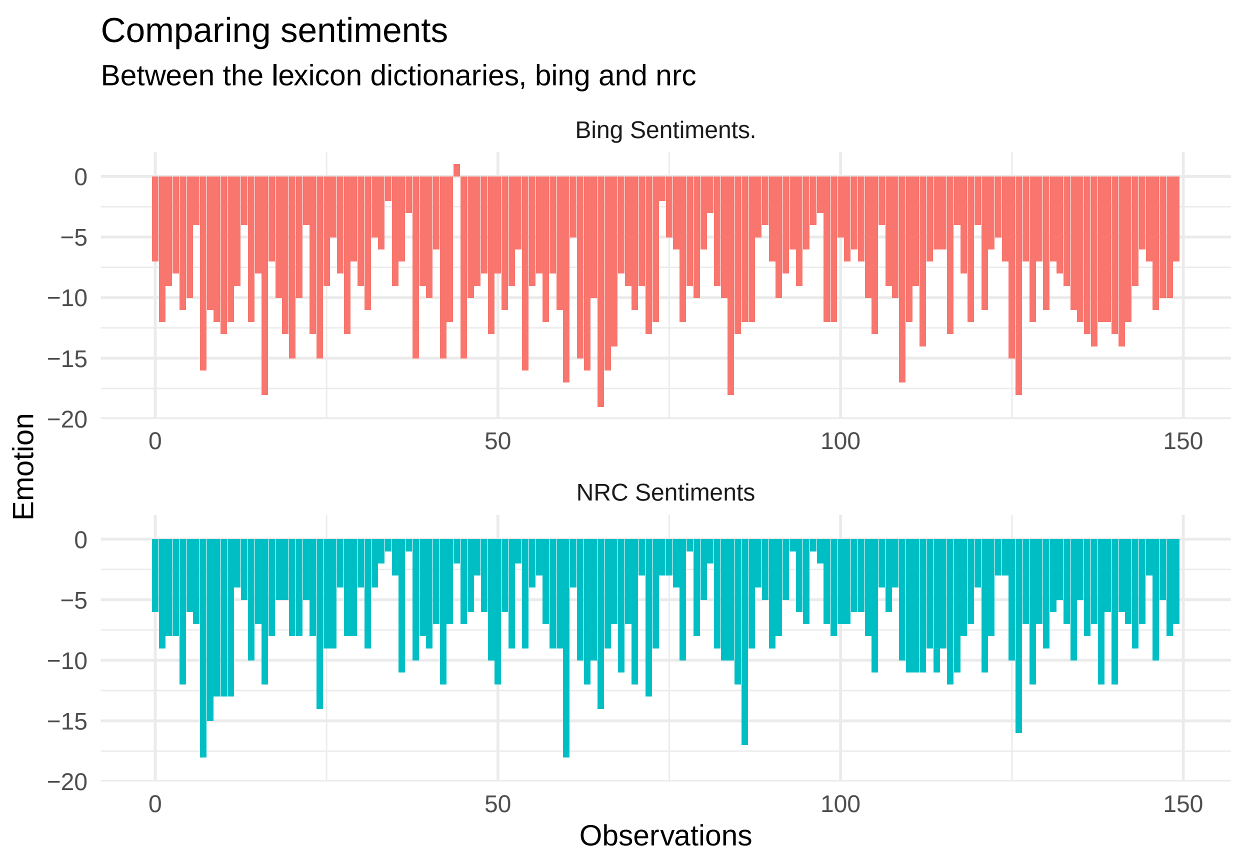
```
##      word   n
## 1 delayed 143
## 2    late  90
## 3    lost  69
## 4   delay  68
## 5    wait  56
```

The most popular words within our dataset against the nrc library.

Lets plot both of these, to get a better visual of the differences.

```
## Joining, by = c("sentiment", "word")
## Joining, by = c("sentiment", "word")
```

Now lets plot it to get an idea of how the sentiment is per 150 lines.



Now this lovely negative graph shows us the overall sentiment total per 150 lines, bing measures some of the words a lot harsher, with what looks like 0 positives. Yet NRC shows not as much negative scores (but very close) with even at least 1 positive sentiment.

```
## # A tibble: 5 x 2
##   word       sentiment
##   <chr>      <chr>
## 1 2-faces    negative
## 2 abnormal   negative
## 3 abolish    negative
## 4 abominable negative
## 5 abominably negative
```

```
## # A tibble: 5 x 2
##   word      sentiment
##   <chr>     <chr>
## 1 abacus    trust
## 2 abandon   fear
## 3 abandon   negative
## 4 abandon   sadness
## 5 abandoned anger
```

As mentioned above, there are also other lexicon dictionaries, some which are community driven and open source, with others being behind a pay wall. Lets look at the dictionary afinn.

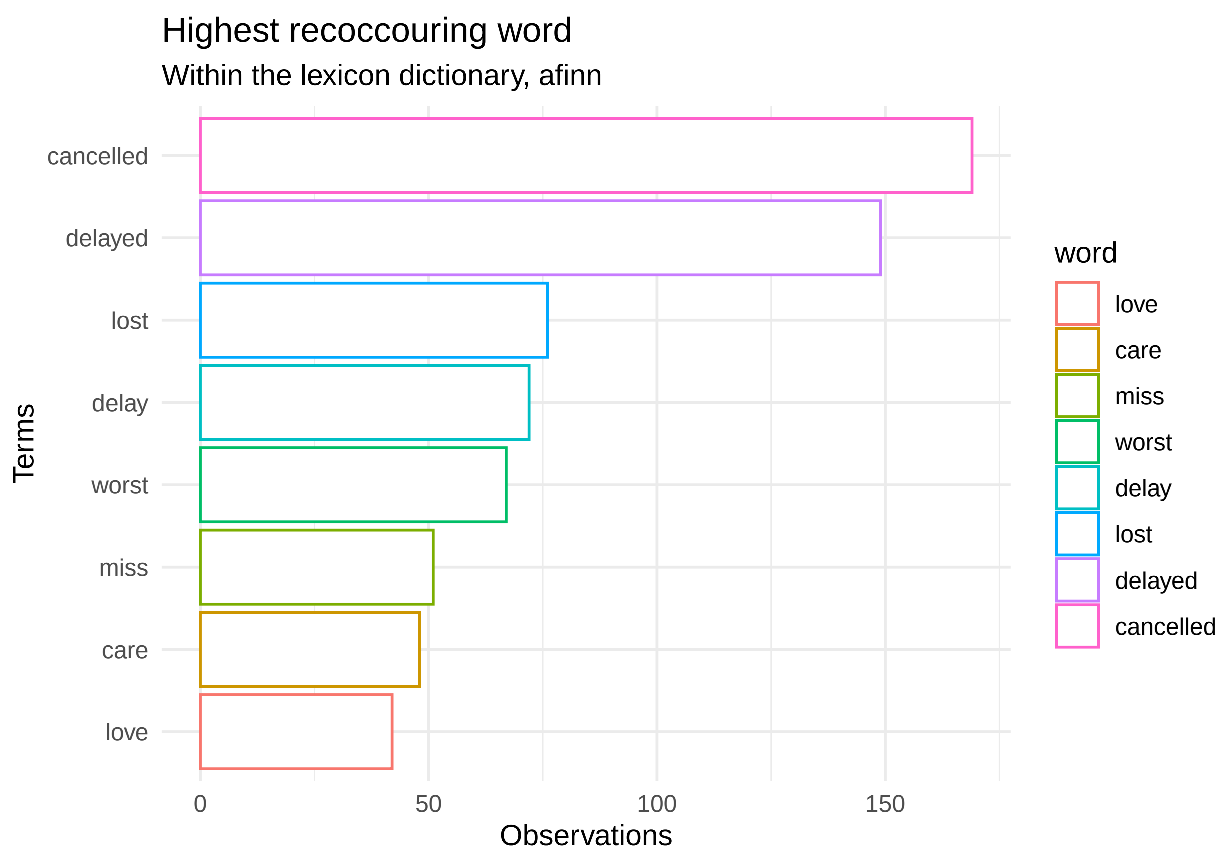
```
## # A tibble: 5 x 2
##   word      value
##   <chr>     <dbl>
## 1 abandon      -2
## 2 abandoned    -2
## 3 abandons     -2
## 4 abducted     -2
## 5 abduction    -2
```

The good thing about afinn is that it ranks the words with a value, which for example, could be used to find the most negative episode of a tv show. For now lets use a similar method for afinn as we did for bing and nrc.

```
## Joining, by = "word"
```

```
##         word value   n
## 1  cancelled    -1 169
## 2    delayed    -1 149
## 3       lost    -3  76
## 4      delay    -1  72
## 5      worst    -3  67
## 6       miss    -2  51
## 7       care     2  48
## 8       love     3  42
## 9     missed    -2  37
## 10     stuck    -2  36
```

This output shows the word which, the value of the word (This can range from -5 to +5 (Neg to Pos)) with the number of observations within the dataset.



## 4 - Topic Modelling

Lets start by getting a copy of our cleaned sentiment analysis data (to speed things up.)

```
##        word   n
## 1   service 218
## 2      time 187
## 3  customer 175
## 4 cancelled 169
## 5       bag 153
```

Next the data frame would need to be changed to a Document Term Matrix

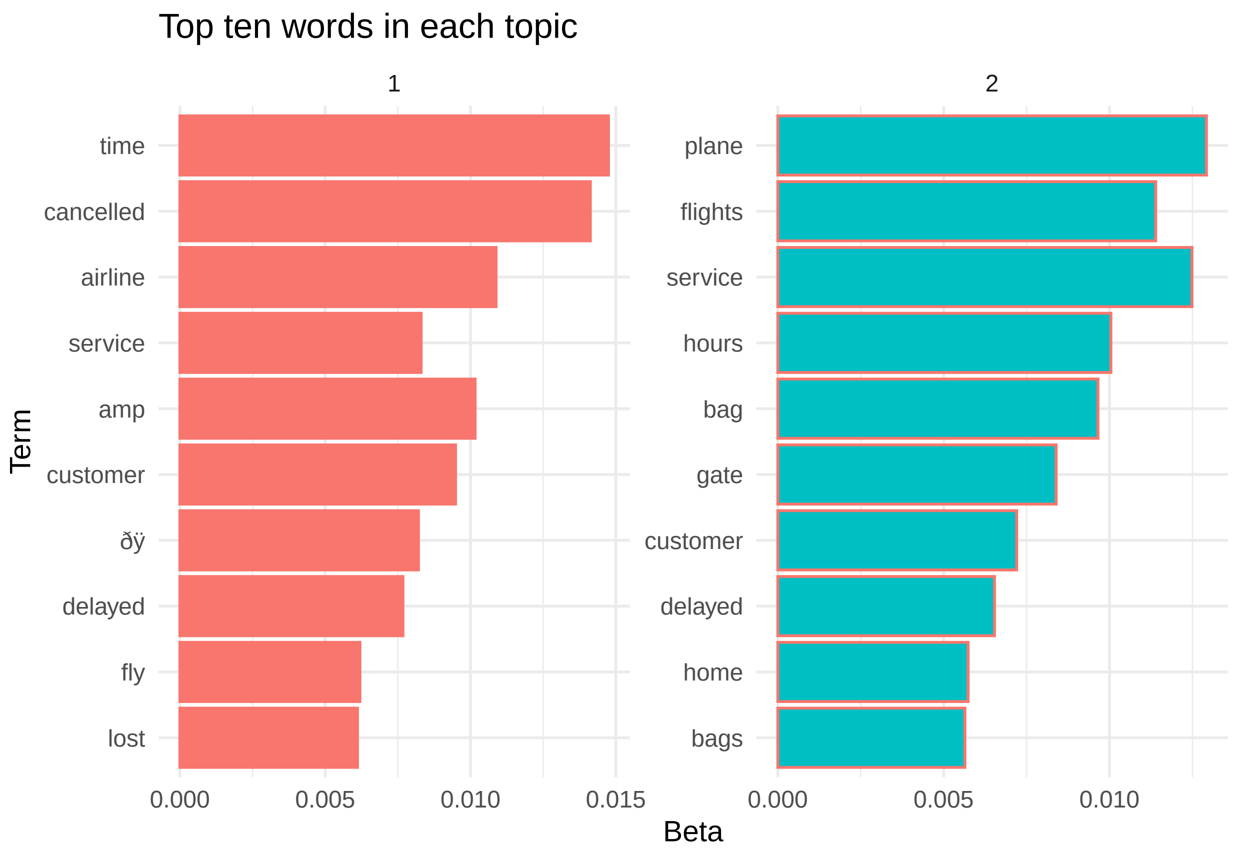
After converting to a corpus, then to a DTM, this dataset can now be used with the LDA model which is part of the topic models package. Lets start by assuming there are 2 topics and see how the topics fit. This could also be changed later.

With the LDA created, the next steps would be to extract how the data falls into 2 topics.

```
## # A tibble: 11,450 x 3
##    topic term              beta
##    <int> <chr>            <dbl>
##  1     1 dhepburn    0.0000590
##  2     2 dhepburn    0.0000363
##   3     1 added       0.000535
##   4     2 added       0.000227
##   5     1 commercials 0.0000596
##   6     2 commercials 0.0000358
##   7     1 experience  0.00373
##   8     2 experience  0.00123
##   9     1 tacky       0.0000886
##  10     2 tacky       0.00000679
## # ... with 11,440 more rows
```

The output above shows each topic within the document, against the term/word with the beta which is the probability/% of accuracy.

```
## Selecting by beta
```



The two topics now show the top 10 words for both topic 1 and topic 2. These words are the highest contributors to the weight of each topic.

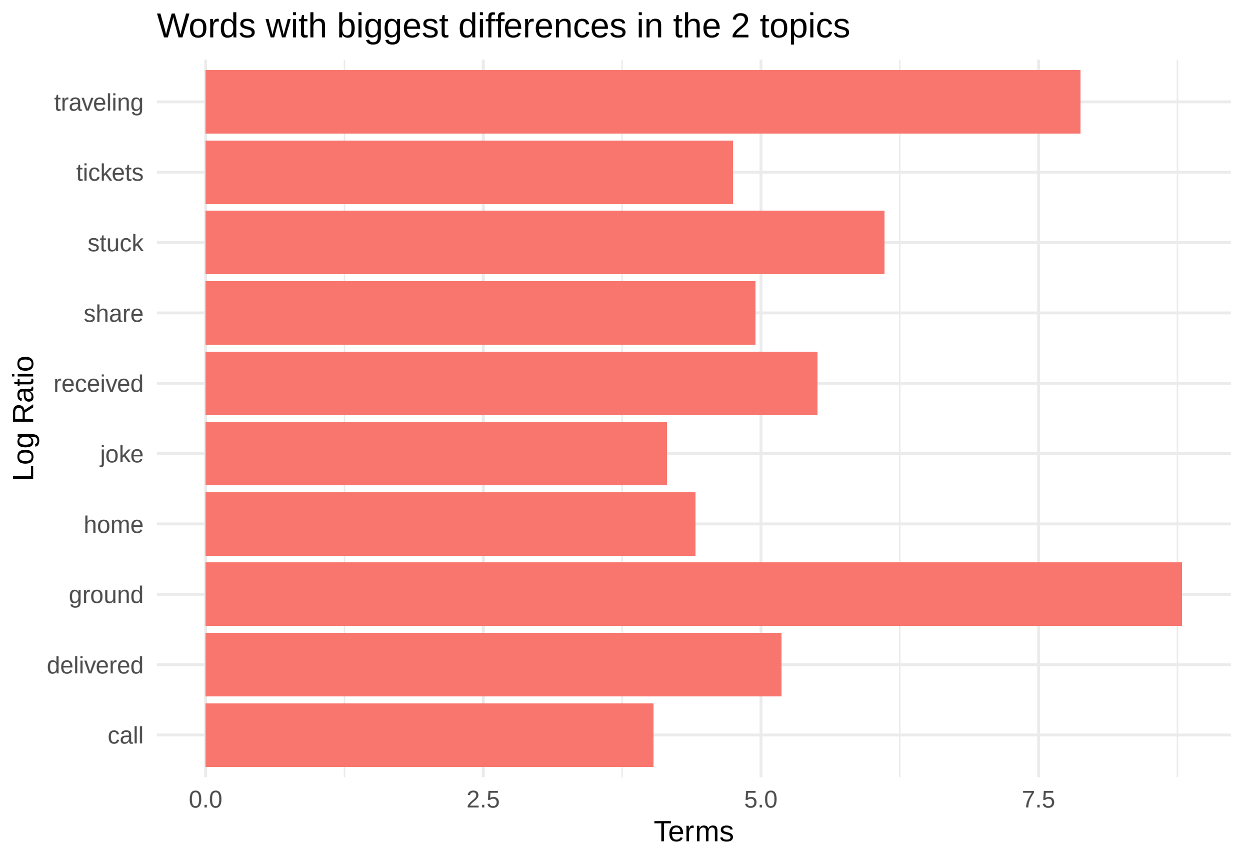
With the top 3 words being different in both of these topics, we can assume it has worked and notice the difference. With topic 1 showing, time, cancelled and airline, and topic 2 showing, planes, flight, service.

Due to this being quite difficult to interpret, lets look at comparing the biggest differences in words, between all topics.

```
## # A tibble: 237 x 4
##    term       topic1   topic2 log_ratio
##    <chr>        <dbl>    <dbl>     <dbl>
##  1 100      0.000668 0.00114     0.777
##  2 1st      0.00103  0.00183     0.833
##  3 add      0.00106  0.00113     0.0964
##  4 agent    0.00231  0.00331     0.519
##  5 agents   0.00257  0.000861   -1.58
##  6 ago      0.000270 0.00259     3.27
##  7 air      0.00125  0.00104    -0.275
##  8 airline  0.0109   0.000370   -4.88
##  9 airlines 0.00156  0.00264     0.762
## 10 airport  0.00363  0.00334    -0.120
## # ... with 227 more rows
```

Comparing the two betas between both topics is a good way to see how much the words can vary from the two topics using the LDA Model. This was done with a log2 ratio to make the differences more symmetrical with negative and positive values.

Whilst this shows a lot of information about the differences between the words and the topics, it is hard to visualise. So the plot below will look at the top 10 words with the biggest differences.



These top ten words shows the words which have the biggest differences between the two topics, whilst all of the dataset is around airlines, by suggesting there is only two topics the LDA model has managed to split them as best as it could.

## 5 - Further Exploration

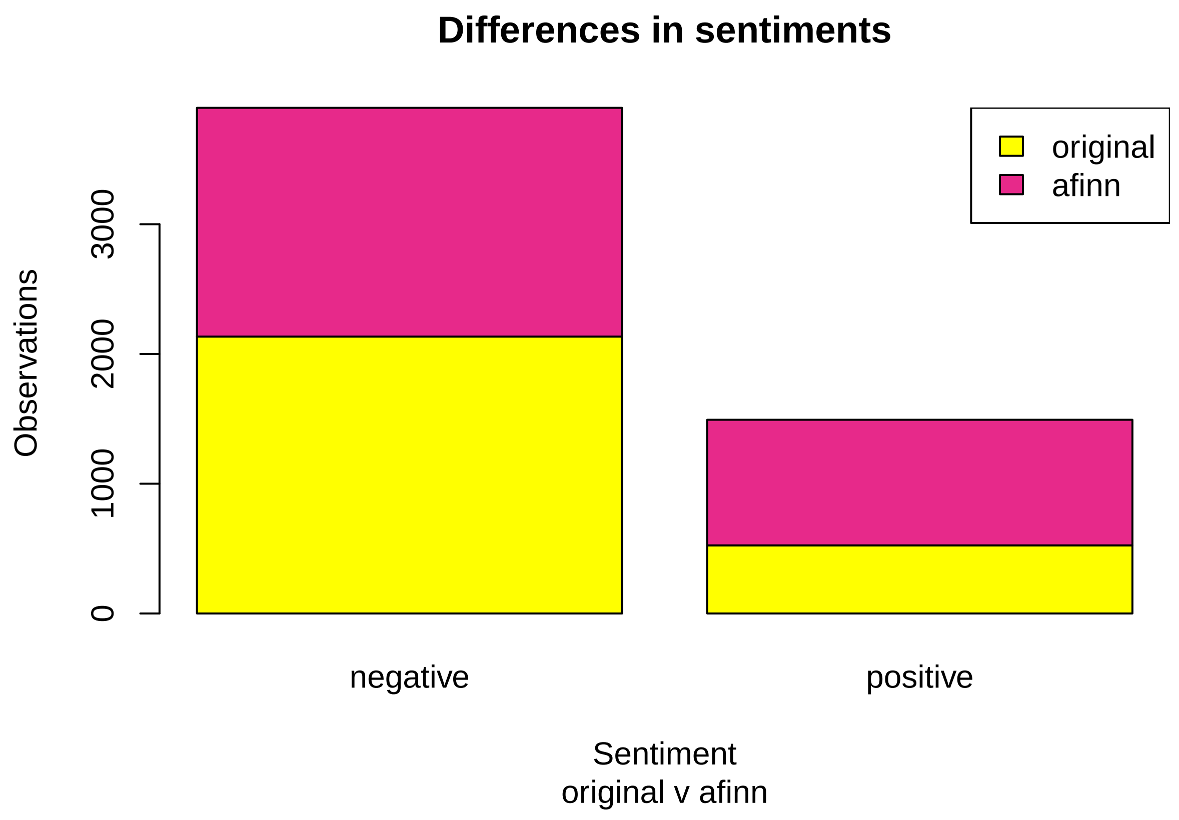
### 5.1 - Compare original Sentiment column

This section, lets have a look at the original sentiment column of the dataset.

Here is the original sentiments within the dataset counted, lets compare these to different lexicons.

```
## Joining, by = "word"
## Joining, by = "word"
```

Now that there is a positive and negative count for afinn and the original sentiments within the dataset, lets plot them and see the differences.



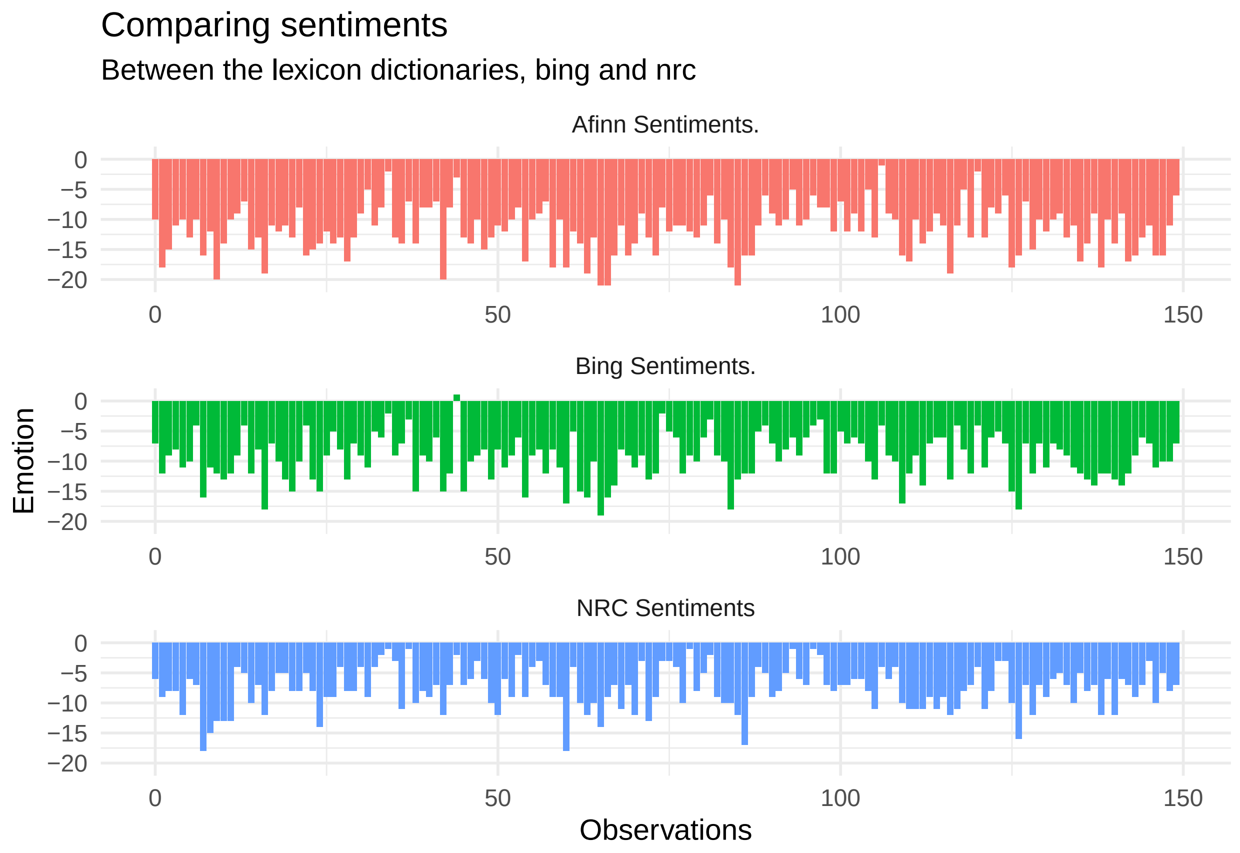
### 5.2 - Comparing lexicon dictionaries

This section will look at the differences with the 3 dictionaries explored within this report.

```
## Joining, by = "word"

## Joining, by = c("sentiment", "word")
## Joining, by = c("sentiment", "word")
```

Now lets plot it to get an idea of how the sentiment is per 150 lines.



Whilst looking at the comparisons between the 3 lexicon dictionaries, here there are a few things which stand out. Whilst this is calculating the sentiment for 150 terms, all 3 lexicon dictionaries look similar with a few differences here and there.

This could be a good or a bad thing when looking at sentiment analysis and using these lexicons, but with there being multiple choices it is good to see how they differ

## 6 - References

### 6.1 - Books

Silge, J. and Robinson, D., 2017. Text mining with R: A tidy approach. Available at: https://www.tidytextmining.com (Accessed 05/12/2020)

Winters, R. (2021), Practical Prediction Analysis, Available at: https://www.oreilly.com/library/view/practical-predictive-analytics/9781785886188/ba5bd5c5-31d7-4502-a626-fe5aa1194ca9.xhtml (Accessed 23/01/2021)

### 6.2 - Web

Silge, J. (2018), The game is afoot! Topic modeling of Sherlock Holmes stories, Available at: https://juliasilge.com/blog/sherlock-holmes-stm/ (Accessed 24/01/2021)

## 6.3 - Packages

Emil Hvitfeldt (2020). textdata: Download and Load Various Text Datasets. R package version 0.4.1. https://CRAN.R-project.org/package=textdata

Erich Neuwirth (2014). RColorBrewer: ColorBrewer Palettes. R package version 1.1-2. https://CRAN.R-project.org/package=RColorBrewer

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Hadley Wickham (2020). tidyr: Tidy Messy Data. R package version 1.1.2. https://CRAN.R-project.org/package=tidyr

Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2021). dplyr: A Grammar of Data Manipulation. R package version 1.0.3. https://CRAN.R-project.org/package=dplyr

Ian Fellows (2018). wordcloud: Word Clouds. R package version 2.6. https://CRAN.R-project.org/package=wordcloud

Ingo Feinerer and Kurt Hornik (2020). tm: Text Mining Package. R package version 0.7-8. https://CRAN.R-project.org/package=tm

Milan Bouchet-Valat (2020). SnowballC: Snowball Stemmers Based on the C 'libstemmer' UTF-8 Library. R package version 0.7.0. https://CRAN.R-project.org/package=SnowballC

RStudio Team (2020). RStudio: Integrated Development for R. RStudio, PBC, Boston, MA URL http://www.rstudio.com/.

Silge J, Robinson D (2016). "tidytext: Text Mining and Analysis Using Tidy Data Principles in R." *JOSS*, *1*(3). doi: 10.21105/joss.00037 (URL: https://doi.org/10.21105/joss.00037), <URL: http://dx.doi.org/10.21105/joss.00037>.

## 6.4 - Software

Windows 10 - 20H2

Ubuntu-20.04-LTS

R - 4.0.2

RStudio - version 1.3.1093