TEXT ANALYSIS

Kariuki Reuben Mwangi

3/9/2022

Introduction

- Objecctives of the course
- data wrangling
- visualization
- Perform Sentimental analysis
- Run and interpret the topic model

```
# AIRLINE TWEET DATASET
ch_1_twitter_data <- readRDS("C:/Users/kariuki
Reuben/Downloads/ch_1_twitter_data.rds")
# loading the required library
library(tidyverse)

# printing the twitter_data

ch_1_twitter_data
```

```
## # A tibble: 7,044 x 6
##    tweet_id date                complaint_label tweet_text
usr_followers_c~
##       <dbl> <dttm>              <chr>           <chr>
<dbl>
##  1  4.77e17 2014-06-12 00:07:25 Non-Complaint   "1. Haneda. 2.~
152
##  2  4.77e17 2014-06-12 00:12:30 Non-Complaint   "My plane to G~
184
##  3  4.77e17 2014-06-12 00:13:56 Complaint       "So apparently~
136
##  4  4.77e17 2014-06-12 00:16:09 Non-Complaint   "Je supporte l~
1
##  5  4.77e17 2014-06-12 00:17:37 Non-Complaint   "Dear @CebuPac~
67
##  6  4.77e17 2014-06-12 00:18:49 Complaint       "Boo @Delta ju~
138
```

```
##  7  4.77e17 2014-06-12 00:26:42 Non-Complaint   "#PALFliesHane~
21
##  8  4.77e17 2014-06-12 00:31:08 Complaint       "@JetBlue you ~
133
##  9  4.77e17 2014-06-12 00:35:27 Non-Complaint   "Celebrating @~
607
## 10  4.77e17 2014-06-12 00:46:47 Non-Complaint   "Don't do this~
165
## # ... with 7,034 more rows, and 1 more variable: usr_verified <lgl>
```

```
# print just the complaints in twitter_data

ch_1_twitter_data %>%
  filter(complaint_label== "Complaint")
```

```
## # A tibble: 1,676 x 6
##    tweet_id date                complaint_label tweet_text
usr_followers_c~
##       <dbl> <dttm>              <chr>           <chr>
<dbl>
##  1  4.77e17 2014-06-12 00:13:56 Complaint       "So apparently~
136
##  2  4.77e17 2014-06-12 00:18:49 Complaint       "Boo @Delta ju~
138
##  3  4.77e17 2014-06-12 00:31:08 Complaint       "@JetBlue you ~
133
##  4  4.77e17 2014-06-12 00:49:18 Complaint       "@TheRealKerst~
221
##  5  4.77e17 2014-06-12 00:54:32 Complaint       "@AmericanAir ~
10
##  6  4.77e17 2014-06-12 00:58:36 Complaint       "I strongly ad~
158
##  7  4.77e17 2014-06-12 01:08:40 Complaint       "@donclifford ~
55
##  8  4.77e17 2014-06-12 01:27:36 Complaint       "@USAirways fl~
995
##  9  4.77e17 2014-06-12 02:17:21 Complaint       "Just asked @D~
7005
## 10  4.77e17 2014-06-12 02:18:16 Complaint       "@migs647 @Vir~
919
## # ... with 1,666 more rows, and 1 more variable: usr_verified <lgl>
```

```
# start with the data frame and group the data by whether or not tweet is a
complait,compute the mean,min and max follower counts

ch_1_twitter_data %>%
  group_by(complaint_label)%>%
  summarize(avg_followers = mean(usr_followers_count),min_followers=
min(usr_followers_count),max_followers = max(usr_followers_count))
```

```
## # A tibble: 2 x 4
##   complaint_label avg_followers min_followers max_followers
##   <chr>                   <dbl>         <dbl>         <dbl>
## 1 Complaint               3234.             0       1259803
## 2 Non-Complaint           4487.             0       2200851
```

Counting Categorical data

```
# Count the number of verified and non-verified users
ch_1_twitter_data%>%
filter(complaint_label == "Complaint")%>%
count(usr_verified)
```

```
## # A tibble: 2 x 2
##   usr_verified     n
##   <lgl>        <int>
## 1 FALSE         1650
## 2 TRUE            26
```

```
# group by whether or not user is verified
ch_1_twitter_data%>%
group_by(usr_verified)%>%
  summarize(avg_followers =mean(usr_followers_count),n=n())
```

```
## # A tibble: 2 x 3
##   usr_verified avg_followers     n
##   <lgl>                <dbl> <int>
## 1 FALSE                1999.  6927
## 2 TRUE               133849.   117
```

Tokenizing and Cleaning

Explore the content of the airline tweets in twitter_data through word counts. The content of each tweet is in the tweet_text column.

```
# load the tidytext packages
#install.packages("tidytext")
library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 4.0.5
```

```r
# Tokenize the twitter data
tidy_twitter <- ch_1_twitter_data%>%
  unnest_tokens(word,tweet_text)
# compute word counts and arrange the count in desc
tidy_twitter %>%
  count(word)%>%
  arrange(desc(n))
```

```
## # A tibble: 18,601 x 2
##    word      n
##    <chr> <int>
##  1 to     2834
##  2 the    2212
##  3 a      1989
##  4 i      1752
##  5 t.co   1405
##  6 http   1361
##  7 for    1356
##  8 you    1345
##  9 on     1289
## 10 and    1153
## # ... with 18,591 more rows
```

```r
# cleaning and counting-Remove stop words to explore the content of just the
airline tweets classified as complaints in twitter_data.Remove the stop
words,filter to keep complaints and compute word counts and arrange in desc
 tidy_twitter <- ch_1_twitter_data%>%
   unnest_tokens(word,tweet_text)%>%
   anti_join(stop_words)
```

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

```r
 # filtering the complaint label
tidy_twitter%>%
filter(complaint_label=="Complaint")%>%
   count(word)%>%
   arrange(desc(n))
```

```
## # A tibble: 3,863 x 2
##    word                n
##    <chr>           <int>
##  1 flight            459
##  2 united            362
##  3 americanair       294
##  4 usairways         207
##  5 time              167
##  6 delta             141
##  7 service           137
##  8 2                 129
##  9 delayed           123
```

```
## 10 british_airways   121
## # ... with 3,853 more rows
```

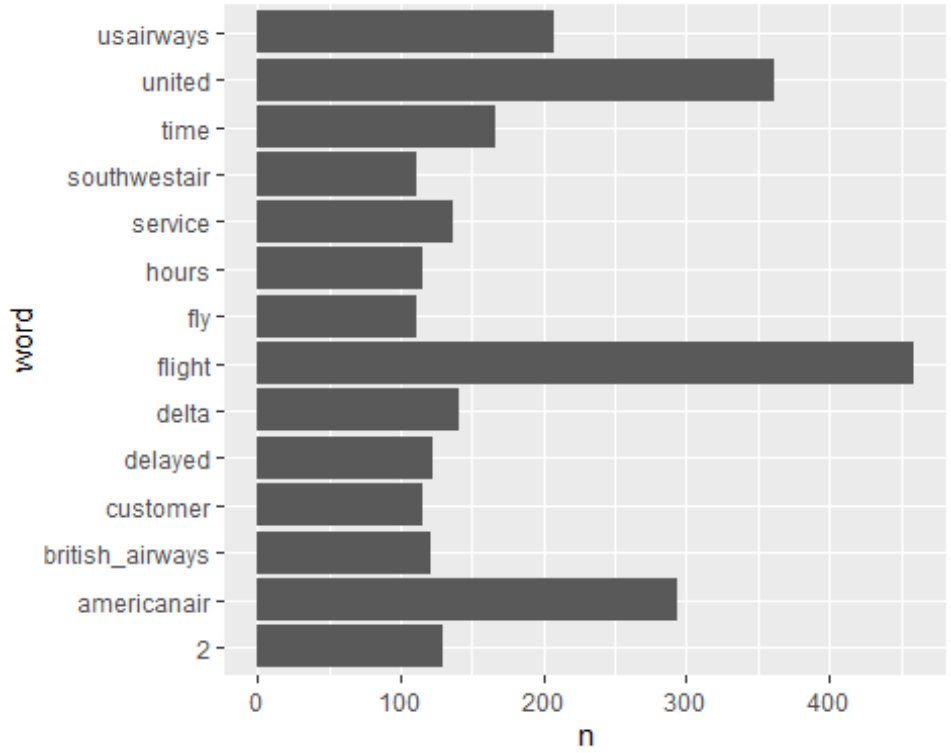
Plotting word counts

Visualizing complaints that ended the last chapter with complaint word counts. Now let's visualize those word counts with a bar plot.The tidyverse and tidytext packages have been loaded. twitter_data has been tokenized and the standard stop words have been removed.

```
# keep words with count greater than 100
word_counts<- tidy_twitter%>%
  filter(complaint_label== "Complaint")%>%
  count(word)%>%
  filter(n>100)%>%
  arrange(desc(n))

word_counts
```

```
## # A tibble: 14 x 2
##    word                n
##    <chr>           <int>
##  1 flight            459
##  2 united            362
##  3 americanair       294
##  4 usairways         207
##  5 time              167
##  6 delta             141
##  7 service           137
##  8 2                 129
##  9 delayed           123
## 10 british_airways   121
## 11 customer          116
## 12 hours             115
## 13 fly               112
## 14 southwestair      112
```

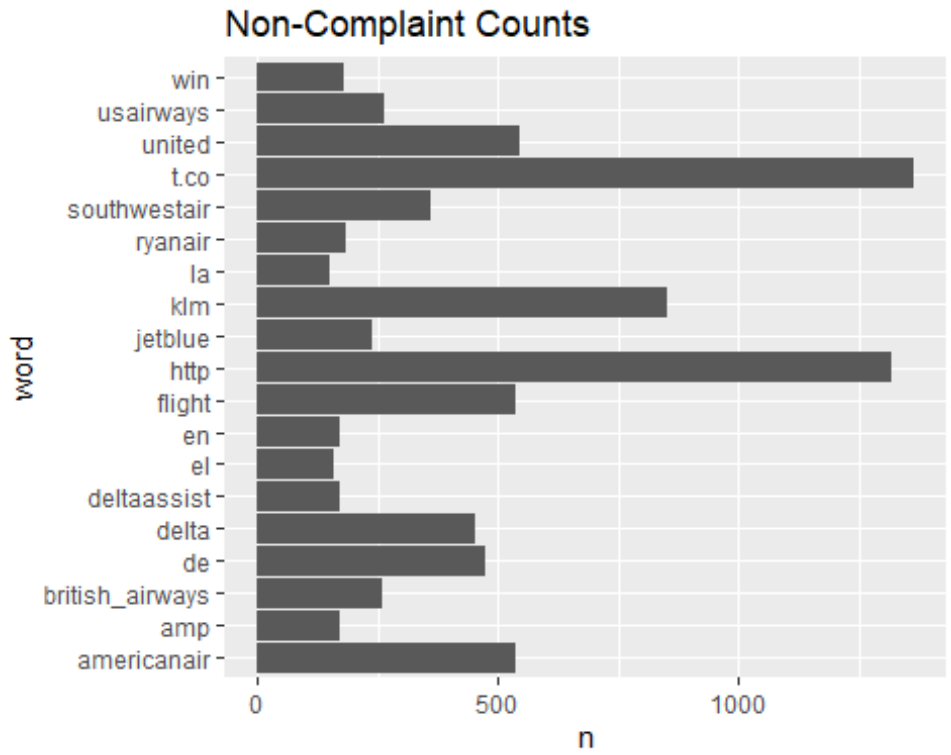
```
# create a bar plot using word_counts with x =word and flip the plot
coordinates
ggplot(word_counts,aes(x=word,y =n))+geom_col()+ coord_flip()
```



Visualizing non-complaints

Now let's visualize the word counts associated with non-complaints.

```
# Only keep the non -complaints
word_counts <- tidy_twitter%>%
  filter(complaint_label=="Non-Complaint")%>%
  count(word)%>%
  filter(n>150)

# creating a bar plot using the new word_counts

ggplot(word_counts,aes(x=word,y= n))+
  geom_col()+
  coord_flip()+
  ggtitle("Non-Complaint Counts")
```



Improving word count plots-Adding custom stop words

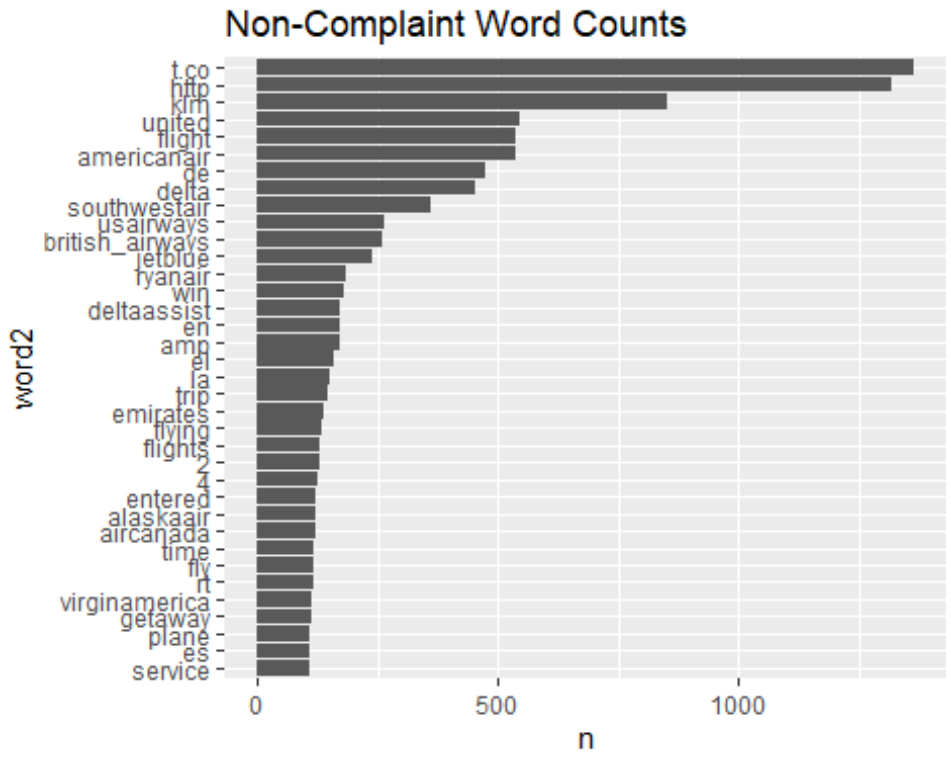
We've seen a number of words in twitter_data that aren't informative and should be removed from your final list of words. In this exercise, you will add a few words to your custom_stop_words data frame.

```
# Columns should match stops words,add http,win and t.co as custom stop words
custom_stop_words <-
tribble(~words,~lexicon,"http","CUSTOM","win","CUSTOM","t.co","CUSTOM")
custom_stop_words

## # A tibble: 3 x 2
##   words lexicon
##   <chr> <chr>
## 1 http  CUSTOM
## 2 win   CUSTOM
## 3 t.co  CUSTOM

# bind the custom stop words to stop_words

stop_words2<- stop_words%>%
  bind_rows(custom_stop_words)
```

Visualizing word counts using factors

I've added a number of other custom stop words (including the airline names) and tidied the data for you. Now you will create an improved visualization and plot the words arranged in descending order by word count.

```
# keep terma that occur more than 100 times,reorder word as an ordered factor
by word counts
word_counts <- tidy_twitter %>%
  filter(complaint_label == "Non-Complaint")%>%
  count(word)%>%
  filter(n > 100)%>%
  mutate(word2 = fct_reorder(word,n))

# plot the new word column with the type factor
ggplot(word_counts,aes(x=word2,y =n))+
  geom_col()+
  coord_flip()+
  ggtitle("Non-Complaint Word Counts")
```



Counting by product and reordering

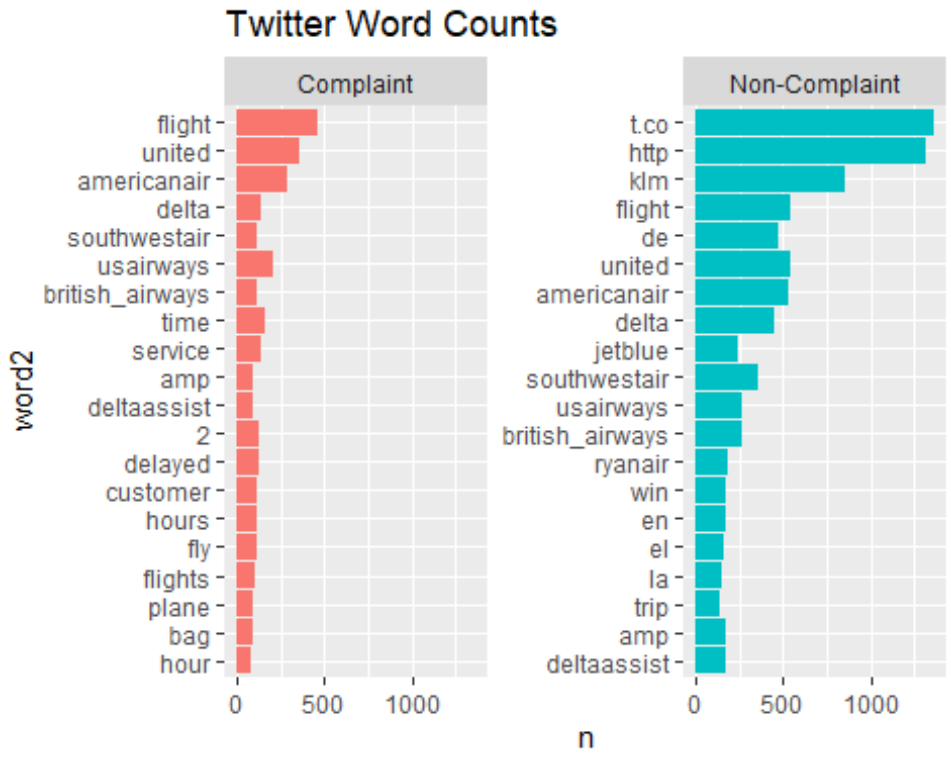
tidy_twitter has been tokenized and stop words, including custom stop words, have been removed. You would like to visualize the differences in word counts based on complaints and non-complaints.

```
# count words by whether or not its a complaint,group by whether or not its a complaint,keep the top 20 words,ungroup before reordering word as a factor by count
word_counts<- tidy_twitter %>%
  count(word,complaint_label)%>%
  group_by(complaint_label)%>%
  top_n(20,n)%>%
  ungroup()%>%
  mutate(word2 = fct_reorder(word,n))
```

Visualizing word counts with facets

The word_counts from the previous exercise have been loaded. Let's visualize the word counts for the Twitter data with separate facets for complaints and non-complaints.

```
# include a color aesthetic tied to whether or not its a complaint
ggplot(word_counts,aes(x=word2,y = n, fill= complaint_label))+
  geom_col(show.legend = FALSE)+
  facet_wrap(~complaint_label,scales = "free_y")+
  coord_flip()+
  ggtitle("Twitter Word Counts")
```

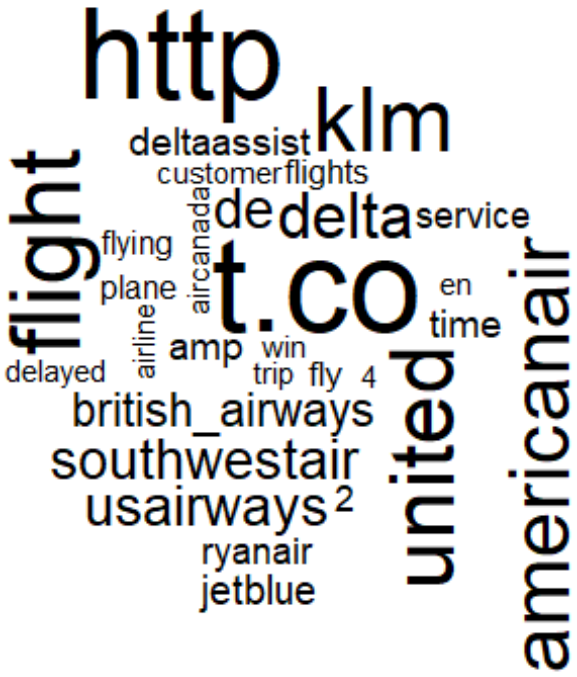


Creating a word cloud

We've seen bar plots, now let's visualize word counts with word clouds! tidy_twitter has already been loaded, tokenized, and cleaned.

```
# load the wordcloud library
#install.packages("wordcloud")
library(wordcloud)

## Warning: package 'wordcloud' was built under R version 4.0.5

## Loading required package: RColorBrewer

# compute word countsand assign to word_couts
word_counts <- tidy_twitter %>%
  count(word)
word_counts

## # A tibble: 17,998 x 2
##    word                n
##    <chr>           <int>
##  1 _adowaa_            1
##  2 _arzar              1
##  3 _austrian          11
##  4 _bbbb_              1
```

```
##  5 _cierratindall     1
##  6 _confucksia        1
##  7 _for_              1
##  8 _hkhodary          1
##  9 _jchapman_         1
## 10 _kellydale_        1
## # ... with 17,988 more rows

#assign the word column to word and count column to freq
wordcloud(word = word_counts$word,freq = word_counts$n,max.words =  30)
```



Adding a splash of color

What about just the complaints? And let's add some color. Red seems appropriate. The wordcloud package has been loaded along with tidy_twitter.

```
# compute complaint word counts and assign to word_counts
word_counts <- tidy_twitter%>%
  filter(complaint_label == "Complaint")%>%
  count(word)
# create a complaint word cloud of the top 50 terms colored red
```

```
wordcloud(words =  word_counts$word,freq =  word_counts$n,max.words =
50,colors =  "red")
```

END