Milestone 4 - Twitter Sensitivity Analysis

1. Goal

The main goal/objective of my project is to analyse the sentiment of various tweets from Twitter, and observe trends in the tweets' sentiments. I decided to focus on tweets around the topic of Corona Virus across 6 specified countries.

2. Data Source

The main source of data came from Twitter using the package `rtweet`. This required me to create a developers account on Twitter and obtain a token key to access the data. The dataframe returned from Twitter is not very good for obtaining the location of a tweet - this is because very few people allow this feature, or they can put a custom location (e.g. "NZ" or "KiwiLand" etc) into their profile which is very messy.

As a result, I decided to search for tweets by including a country name as a keyword to obtain tweets on corona virus as well as the specified country. Overall, I obtained roughly 2000 tweets for each of the 6 countries: New Zealand, USA, Australia, Canada, UK/England and China. Note this does not mean the tweet came from the same country, but it will allow me to analyse the sentiment of a tweet involving that country. The data gethering code can be seen in Appendix 2.1.

Furthermore, I also need to use sentiment data. Initially in milestone 2, I decided to use the *get_sentiment* along with some downloaded lexicon packages. However, I have since found a better package, namely `sentimentr`, which calculates the sentiment of text in a smarter way. This will be further discussed in the next section.

3. Data Processing and Exploration

Tidying main data frame

The main form of data tidying required is to reduce the number of columuns. As seen below, there are initially 90 attributes, a lot of which are not going to be of any use to me. So the first step is to remove many irrelevant columns.

```
library(tidyverse)
library(rtweet)

load("sample.RData")
length(names(sample_data)) # number of col names of a twitter dataframe
```

```
## [1] 90
```

By selecting some useful parameters, I reduced the columns to relevant ones only and added a new column to each country's dataframe called 'specified_country' - this is the country I have specifically searched for tweets on. Then I join each country's dataset into one. The new column names can be seen below, and the code is in Appendix 3.1.

```
load("allDFs.Rdata")
names(coronatweets.df) # potentially useful column names
```

```
##  [1] "screen_name"        "text"               "created_at"
##  [4] "source"             "specified_country"  "location"
##  [7] "favorite_count"     "retweet_count"      "followers_count"
## [10] "statuses_count"     "verified"           "account_created_at"
## [13] "description"        "tidytext"           "wordcount"
## [16] "av_sentiment"
```

Note: `tidytext`, `wordcount` and `av_sentiment` are attributes I have added which are explained below - they have just been saved to the dataframe already.

Next, I combine each of the individual country datasets into one dataframe - although the individual sets may still be used for independent analysis. This can also be seen in Appendix 3.1

Tidy text attribute

I also tidy the text in the next section below (main code in Appendix 3.1) by removing unnecessary elements, such as http strings (weblinks), new line characters and I convert to lower case for simplicity. Note that I originally (in milestone 2)

removed punctuation as well to tidy the text. I have decided to keep punctuation in for now as this impacts the sentiment analysis.

An example of an original tweet text, and the cleaned text is output below the code segment.

```
coronatweets.df$text[14] # original text
```

```
## [1] "Coronavirus: taking Parliament onto the holodeck https://t.co/RUCI0aAT0J"
```

```
coronatweets.df$tidytext[14]
```

```
## [1] "coronavirus: taking parliament onto the holodeck "
```

Adding sentiment with `sentimentr`

```
library(sentimentr)
library(tidyverse)
#use sentimentr package to calculate average sentiment
sentiments <- coronatweets.df$tidytext %>%
  get_sentences() %>% sentiment_by()
```

The `sentiment_by` function calculates the average sentiment, word count and some other attributes of each text input. I have chosen this analytical tool for the sentiment because it uses a standard dictionary lookup (where positive and negative words are scored +1 and -1 repectively), but also takes *valence shifters* into account.

Valence shifters are words in text that may alter the overall sentiment of a word - e.g. the word "really" in "I **really** like it" amplifies the sentiment of the word "like". Alternatively, they can flip the sentiment entirely - e.g. "I do **not** like it". This makes calculating the sentiment much more accurate compared to treating every word independently, and avoids misinterpretation.

The function breaks the text into sentences/clauses and can also deal with punctuation. Furthermore, removing punctuation manually beforehand could lead to the function miscalculating the sentiment. For example consider the text: "Not really. Good thing I don't care". The function calculates the sentiment of each sentence seperately and gives an average sentiment. However, if I remove punctuation, it will read "Not really Good..." all at once, which we can see is an example of a valence shifter (as "not really" changes the sentiment of "good"). This is why my tidy text has kept its punctuation.

Adding to `coronatweets.df`:

```
coronatweets.df <- coronatweets.df %>%
  mutate(wordcount = sentiments$word_count) %>% # word count in tweet
  mutate(av_sentiment = sentiments$ave_sentiment) %>%  # sentiment av value
  mutate(dscrt_sentiment = ifelse(av_sentiment>=0.05, "Positive", # discrete sentiment (with )
                                  ifelse(av_sentiment<= -0.05, "Negative", "Neutral")))

# example output
coronatweets.df$tidytext[50]
```

```
## [1] "coronavirus: charity auction for maccas free delivery heading skyward  "
```

```
coronatweets.df$dscrt_sentiment[50]
```

```
## [1] "Positive"
```

An example of a tweet and its discrete sentiment is seen above.

I converted the calculated *average sentiment* into a *discrete sentiment*. The sentiment was classified as positive if `av_sentiment >= 0.05`, negative if `av_sentiment <= -0.05` and neutral otherwise (I gave a very small bound either side of zero to account for neutral sentiment). An output table of positivity/negativity can be seen below the following code for each country.

```
tab <- table(coronatweets.df$dscrt_sentiment, coronatweets.df$specified_country)
tab <- rbind(tab, Total=colSums(tab))
tab <- cbind(tab, Total=rowSums(tab))
tab
```

```
##           Aus Canada China   NZ   UK  USA Total
```

```
## Negative  831    772   994  593  738  881  4809
## Neutral   459    401   438  307  487  405  2497
## Positive  708    720   565 1099  775  712  4579
## Total    1998   1893  1997 1999 2000 1998 11885
```

It is interesting to see from the table that NZ has the most amount of positive tweets, and the least amount of negative tweets (based on the calculations), while China and the USA both have largest number of negative tweets. I personally think that this seems plausible given how NZ has been praised for their response. Furthermore, as the virus began in China and USA has the highest death toll, it is expected that these countries have more negative sentiment association.

Creating a document term matrix

In addition to the above dataset, I decided to create a document term matrix to represent the number of occurances of words in tweet text to see how the words themselves impact the sentiment. The main chunk of code for this can be seen in Appendix 3.2.

The general procedure was to reformat the text suitable for a corpus ("documents" of each text attribute) - ideally I would have only tidied the text once above, but different variations in text formatting were necessary in different cases. Then I create the corpus with the `tm` package and remove common *stopwords* (such as "I", "you" etc.), remove *keywords* (the search terms I used to find tweets - as these will occur very often) and I stem each document (this shortens each word to its stem word - e.g. "happiness" to "happy").

From the corpus I create the document term matrix which can be seen in the code below. I demonstrate the most frequent occuring words across all the tweets (occuring more than 750 times).

```
##  [1] "news"  "trump" "peopl" "case"  "virus" "will"  "test"  "amp"   "death"
## [10] "via"
```

```
DTM <- DocumentTermMatrix(corpus) # create DTM from the corpus

# reduce sparse entries
reduce_DTM <- removeSparseTerms(DTM, 0.99)

# get frequent words (> 750)
freqWords <- findFreqTerms(DTM, lowfreq = 750)
freqWords
```

```
##  [1] "news"  "trump" "peopl" "case"  "virus" "will"  "test"  "amp"   "death"
## [10] "via"
```

```
# create word freq matrix dataframe and add sentiment column
word_matrix.df <- as.data.frame(as.matrix(reduce_DTM))
word_matrix.df <- word_matrix.df %>% mutate(sentiment = coronatweets.df$dscrt_sentiment)
```

The word matrix is now a dataframe where each row is a tweet object, and the columns represent a word and its occurances in the current tweet. I add the discrete sentiment on to the word matrix for furture modelling.

Some more data exploration

See appendix 3.3 for the main code to generate the top word tables and word clouds below.

```
knitr::kable(head(nz_words, 7))
```

word	n
coronavirus	930
covid	783
zealand	758
australia	339
nz	274
vaccine	240
lockdown	198

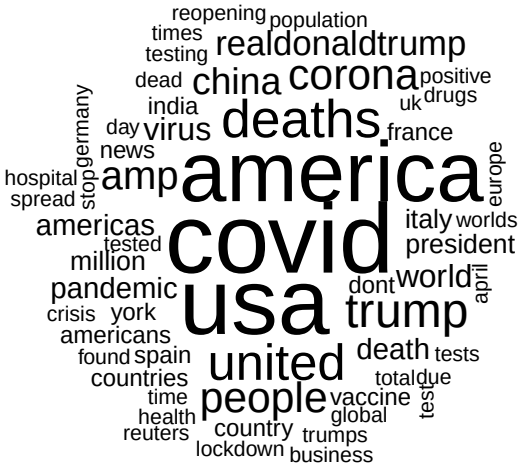
```
knitr::kable(head(us_words, 7))
```

word	n
coronavirus	993
covid	727
usa	677
america	613
deaths	320
united	312
trump	273

Above we can see the top 10 used words in the NZ and US set of tweets respectively. It is interesting to see words like "deaths" (giving negative conotations) as the 5th most used word in the Corona-based tweets for USA, and "vaccine" (possibly hopeful conotations) in the NZ tweets.

As seen below, I have used a word cloud to represent the words used in each country's set of tweets. Below are the word clouds generated for NZ and USA, for example. These word clouds give an idea of the most common words used across the tweets from a given country - bare in mind some of these popular words will likely be the keywords I used to generate the data (such as "corona" and the name of the country). This helps me explore different sentiments (just by reading the words) between countries, as well as on the given topic of coronavirus. In fact, we can already see some contrasting sentiments across NZ and USA words, such as "leader", "praised", "money", "save" in NZ versus "deaths" and "dead" appearing more frequently in USA - although NZ also contains "death" there still seems to be a contrasting sentiment.

```
par(mfrow=c(1,2))
par(mar=c(0.5, 0.5, 0.5, 0.5))
wordcloud(words=nz_words$word, freq=nz_words$n, min.freq = 20, random.order = FALSE, max.words = 60)
wordcloud(words=us_words$word, freq=us_words$n, min.freq = 20, random.order = FALSE, max.words = 60)
```



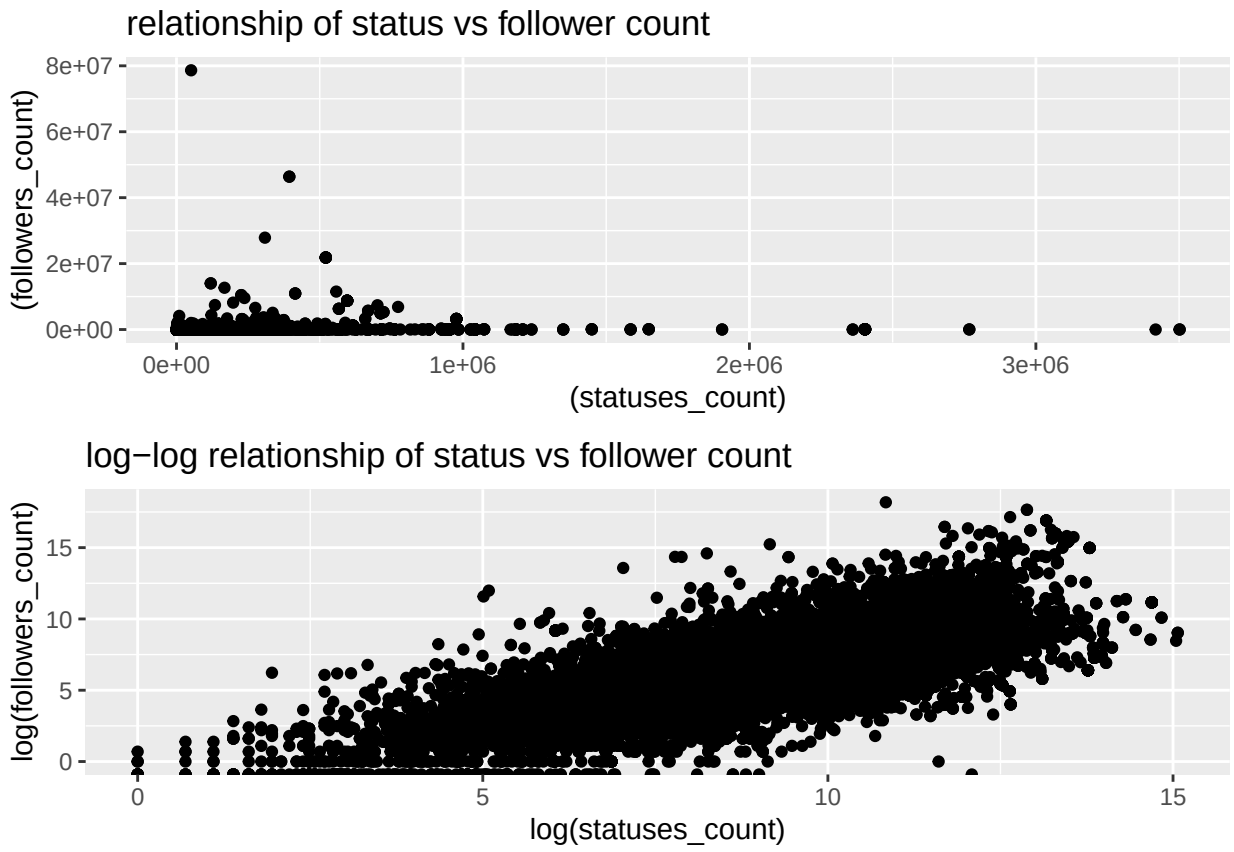
I also decided to look at some other attributes and how they interact. Below is the code and plots of *statuses_count* (the number of statuses a user has made) against *followers_count* (the number of followers a user has).

```
library(ggplot2)
library(gridExtra)

plot1 <- coronatweets.df %>%
  ggplot(aes(x=(statuses_count), y=(followers_count))) +
  geom_point() + labs(title="relationship of status vs follower count")

plot2 <- coronatweets.df %>%
  ggplot(aes(x=log(statuses_count), y=log(followers_count))) +
  geom_point() + labs(title="log-log relationship of status vs follower count")

grid.arrange(plot1,plot2)
```



From the graphs above we can see that it is initially difficult to see a relationship between these two attributes, however on the log-log model we can see the linearly increasing relationship. It will be interesting to see in future analysis if these attributes are important when determining the sentiment of a tweet.

4. Analytical Plan

(Implementation is shown in the results section - 5.)

International sentiment comparison - box plot

Firstly, I do some exploratory/visual analysis of the average sentiment scores based on each country. I have used a box plot which can be seen in the results section. I decided to use a box plot to portray the data in a clearer, more visual way. This will allow me to guage how the sentiment appears to change between countries and observe the spread of sentiment within each country too.

Tree (using `word_matrix.df`)

Next, I decided to use a regression tree model to predict sentiment. This involved splitting the data into train and test sets. I wanted to use a tree to observe the tree structure visually and because trees are often used in classification. For this model, I used the word matrix data frame (word frequencies).

Random forest (using `word_matrix.df`)

The results of the tree model were not particulary good (as outline in the results section further down). So I decided to use a random forest as these use many more trees to improve the fit. The first forest is a model based on the word matrix data again. By using the `ranger` package for the forest, I am also able to observe the importance (I used impurity) of the variables (i.e. words) used in prediction.

Random forest (using `coronatweets.df`)

Finally, I use the original coronatweets.df data to fit another forest. I wanted to see how both forest models compare using the different data frames.

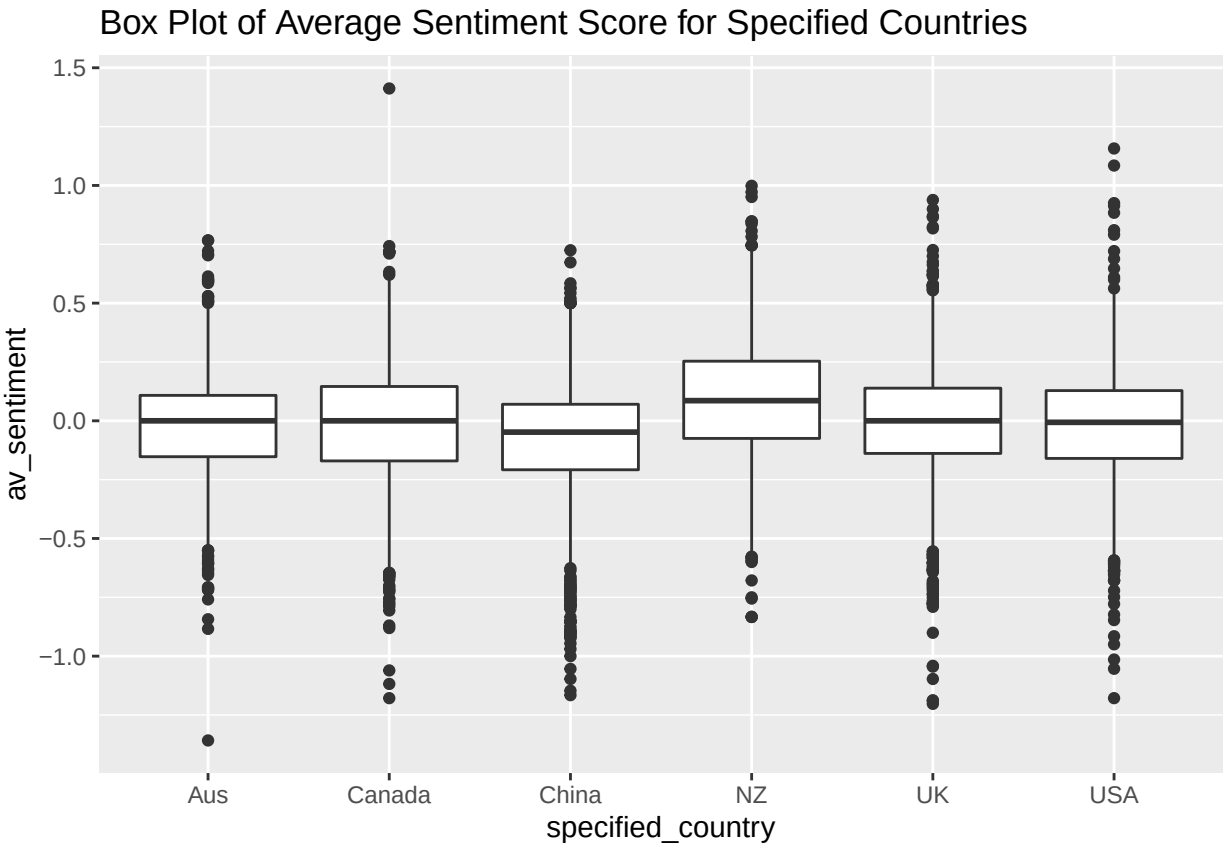
5. Implementation and Results of above Analysis Methods

International sentiment comparison - box plot

```
library(ggplot2)

# Load data and convert specified_country to a factor variable
coronatweets.df$specified_country <- as.factor(coronatweets.df$specified_country)

# Box plot
plot <- ggplot(coronatweets.df, aes(x = specified_country, y = av_sentiment)) +
  geom_boxplot() +
  ggtitle("Box Plot of Average Sentiment Score for Specified Countries")
plot
```



NZ related tweets can be seen to have the most positive (highest average sentiment) tweets on average, while China related tweets appear the most negative on average. There is a larger spread of sentiment scores seen for USA related tweets,

relative to other countries like NZ. This could imply more controversy in this country than in others.

Tree (using `word_matrix.df`)

I split the word matrix data set (from the document term matrix) into train and test sets, and implement a tree and random forest model below (the full code can be seen in Appendix 4.1) Split wordmatrix into train/test sets

```
set.seed(710381321)
# split data
colnames(word_matrix.df) <- make.names(colnames(word_matrix.df))

train_idx <- sample(nrow(word_matrix.df), 0.8*nrow(word_matrix.df))
train_words <- word_matrix.df[train_idx,]
test_words <- word_matrix.df[-train_idx,]
```

Fitting the tree:

```
library(rpart)

# use rpart to fit tree to predict sentiment from word matrix
tree <- rpart(sentiment~., data=train_words, method="class", cp=0.01)
# plotcp(tree)
# plot(tree)

pred <- predict(tree, test_words, type="class")
pred_tab <- table(actual=test_words$sentiment, pred=pred)
pred_tab
```

```
##           pred
## actual     Negative Neutral Positive
##   Negative      900       0       41
##   Neutral       453       0       49
##   Positive      727       0      207
```

At the moment the tree model is mostly just predicting tweets to be negative. Hence, I decided to try a forest next.

Random forest (using `word_matrix.df`)

```
library(ranger)
set.seed(710381321)
forest <- ranger(sentiment~., data=train_words, num.trees = 50,
                 verbose = FALSE,
                 importance="impurity", classification = TRUE)
forest
```

```
## Ranger result
##
## Call:
##  ranger(sentiment ~ ., data = train_words, num.trees = 50, verbose = FALSE,      importance = "impurity",
##
## Type:                             Classification
## Number of trees:                  50
## Sample size:                      9508
## Number of independent variables:  221
## Mtry:                             14
## Target node size:                 1
## Variable importance mode:         impurity
## Splitrule:                        gini
## OOB prediction error:             37.44 %
```

The forest model is fit above. The OOB error is around 40%. This corresponds to a r^2 of 60%. This is not an amazing model, but I show the prediction table below.

```
forest_pred <- predict(forest, test_words)
forest_tab <- table(actual=test_words$sentiment,pred=forest_pred$predictions)
forest_tab
```

```
##           pred
## actual     Negative Neutral Positive
##   Negative      704      49      188
##   Neutral       210     163      129
##   Positive      228      57      649
```

The forest model is much better than the tree, however there are still many incorrectly predicted sentiments. Overall, we can see that the diagonal terms do make up the majority of each row/column.

```
tail(sort(forest$variable.importance))
```

```
##    peopl      amp    death     case    virus   pandem
## 40.08710 40.10622 44.02810 45.85472 49.91929 50.95929
```

The important words can be seen above. It is interesting to see that the model uses words such as “death,”pandem" (stem word from pandemic), and “virus” as the some of the most important words to determine the sentiment. This makes sense, as these are associated with having a strong sentiment.

Random forest (using `coronatweets.df`)

```
set.seed(710381321)

colnames(coronatweets.df) <- make.names(colnames(coronatweets.df)) # get suitable col names

X <- coronatweets.df %>% select(-av_sentiment) %>% select(-text) %>% select(-tidytext)
y <- coronatweets.df$dscrt_sentiment

train_idx2 <- sample(nrow(X), 0.8*nrow(X)) # get train indices for coronatweets.df
# forest model
#
forest2 <- ranger(dscrt_sentiment~specified_country+verified+statuses_count+favorite_count+retweet_count+foll
                  verbose = FALSE,
                  importance="impurity", classification = TRUE)
# predict
forest_pred2 <- predict(forest2, X[-train_idx2,])
forest_tab2 <- table(actual=X[-train_idx2,]$dscrt_sentiment,
                     pred=forest_pred2$predictions)
forest_tab2
```

```
##           pred
## actual     Negative Neutral Positive
##   Negative      497     128      316
##   Neutral       213     108      181
##   Positive      351     124      459
```

The original tweets dataframe is not any better than the word matrix. In fact it is particularly bad at predicting neutral sentiment, and negative and positive sentiments are not predicted as well as the previous forest.

```
tail(sort(forest2$variable.importance))
```

```
##          verified specified_country     retweet_count    favorite_count
##          45.58642         268.65948         327.12526         518.54373
##   followers_count    statuses_count
##        2193.59490        2439.91358
```

Above are the most important variables for the coronatweets.df dataset.

Discussion

General strengths and weaknesses of data

The data I have available has some limitations, which could impact my results. Firstly, the text attribute of a tweet is limited to a relatively small amount of characters. This makes it harder to perform analysis on because the average sentiment calculation does not have much input text to work with, so any errors or misinterpreted sentiments impact the average sentiment more.

Another limitation when calculating the sentiment of a tweet is the (current) inability to analyse emojis/emoticons. These can often strongly indicate, or add to, the sentiment of a message - especially when there are limited characters already. At this stage, I have been unable to make use of any emoticons in the text, however in the future the emojis could be a strength of the text data by contributing to the sentiment score.

The data also contains a mixture of different types/structures of attributes such as discrete and continuous, as well as string and numeric type elements. This may make it more difficult to create the model (perhaps just due to my level of understanding in R).

Despite the mix of attribute types, a plus is that there is a large number of attributes to observe. It gives me a range of different variables that I can analyse with respect to the various tweet sentiments.

Another strength of my data that I have fonud very useful so far is that I have a (roughly) equal amount of data for each of the 6 countries chosen. This is obviously because I searched for data using the relevant key words, nonetheless, it means that I can easily compare results or summaries between countries.

Strengths and weaknesses of analysis

Through data exploration and analysis, I was able to acomplish my main goal - to analyse the sentiment of corona virus tweets and observe trends in the sentiment, specifically between selected countries. I am happy with the sentiment calculations I performed as the sentiment scores seem realistic for most tweets and the general trends between countries (demonstrated in tables and box plot) seem plausible. It could be interesting and beneficial to use more countries in any future analysis, although 6 was not an issue for me.

My other main goal was to not only analyse the sentiment, but also to try to predict it. I attempted to do this using a tree model, and 2 forest models (using a different set of data for each).

The tree model was certainly very bad - it just predicted most tweets to be negative. This was not very useful overall.

The forest model using the `coronatweets.df` dataframe was better than the tree, in that more of the predictions were right, however there were still many incorrect preditions (especially for neutral sentiments). Overall, it makes sense that this forest was not particularly good, as the important attributes in this model (such as the number of statuses the user has or the number of followers) are not as representative of sentiment as I first thought - as people will often post a mix of tweets irrespective of their popularity or number of posts.

Finally, the best model was a forest to the `word_matrix.df` dataframe. This makes more sense to predict the sentiment of a tweet as looking at the common words in text will often reflect more about its overall emotion. This predictions for the 3 discrete sentiments are mostly right, however there are still plenty of incorrectly predicted sentiments.

Conclusion

Overall, my project succeeded in analysing sentiments between various countries, and I was able to observe how different words influence the sentiment more than others. My project was not particularly successful at predicting sentiment (especially neutral sentiment), however, I believe I have a good start that could be improved upon in the future. I would look at using penalised regression models or perhaps neural networks next time in an attempt to improve the prediction of the sentiments.

Appendices

2.1 Obtaining the Twitter data based on Corona Virus for 6 countries:

```
library(rtweet)
library(tidytext)
library(textdata)
library(tidyverse)

nz_corona <- search_tweets(
  q = "corona OR coronavirus OR covid19 nz OR zealand", # query (key words) - nz
  n = 2000, # number of tweets to return
  type = "mixed", # "recent" (default), "mixed", "popular" etc. tweets
  include_rts = FALSE, # include retweets
  parse = TRUE, # gets data as tidier df
  lang = "en" # limit tweets to english
)

us_corona <- search_tweets(
  q = "corona OR coronavirus OR covid19 usa OR \"united states\" OR america", # usa
  n = 2000,
  type = "mixed",
  include_rts = FALSE,
  parse = TRUE,
  lang = "en"
)

uk_corona <- search_tweets(
  q = "corona OR coronavirus OR covid19 uk OR \"united kingdom\" OR england", # uk
  n = 2000,
  type = "mixed",
  include_rts = FALSE,
  parse = TRUE,
  lang = "en",
)

aus_corona <- search_tweets(
  q = "corona OR coronavirus OR covid19 aus OR australia OR aussie", # australia
  n = 2000,
  type = "mixed",
  include_rts = FALSE,
  parse = TRUE,
  lang = "en"
)

ca_corona <- search_tweets(
  q = "corona OR coronavirus OR covid19 ca OR canada", # canada
  type = "mixed",
  include_rts = FALSE,
  parse = TRUE,
  lang = "en"
)

ch_corona <- search_tweets(
  q = "corona OR coronavirus OR covid19 china OR ch OR ROC", # china
  type = "mixed",
  include_rts = FALSE, s
  parse = TRUE,
  lang = "en"
)
```

3.1 Reducing columns and tidying data and text:

```
keep_cols <- as.vector(c("screen_name","text","created_at","source", "specified_country",
                         "location","favorite_count","retweet_count",
                         "followers_count","statuses_count","verified",
                         "account_created_at",'description'))
nz_corona <- nz_corona %>% mutate(specified_country = "NZ") %>% select(keep_cols)
us_corona <- us_corona %>% mutate(specified_country = "USA") %>% select(keep_cols)
uk_corona <- uk_corona %>% mutate(specified_country = "UK") %>% select(keep_cols)
aus_corona <- aus_corona %>% mutate(specified_country = "Aus") %>% select(keep_cols)
ca_corona <- ca_corona %>% mutate(specified_country = "Canada") %>% select(keep_cols)
ch_corona <- ch_corona %>% mutate(specified_country = "China") %>% select(keep_cols)

# merge the data on the kept columns for all countries
corona.df <- full_join(nz_corona, us_corona, by=keep_cols)
corona.df <- full_join(corona.df, uk_corona, by=keep_cols)
corona.df <- full_join(corona.df, aus_corona, by=keep_cols)
corona.df <- full_join(corona.df, ch_corona, by=keep_cols)
coronatweets.df <- full_join(corona.df, ca_corona, by=keep_cols) # full dataset
```

Tidying the text attribute:

```
# remove http elements, punctuation and new lines etc, convert to lower case
coronatweets.df$tidytext <- gsub("https\\S+", "", coronatweets.df$text) # http elements
# coronatweets.df$tidytext <- gsub("[^[:alpha:][:space:]]*", "", coronatweets.df$tidytext) # punctuation
coronatweets.df$tidytext <- gsub("[\r\n]", "", coronatweets.df$tidytext) # newline chars
coronatweets.df$tidytext <- tolower(coronatweets.df$tidytext) # lower case

coronatweets.df$text[1]
coronatweets.df$tidytext[1]

save(nz_corona, us_corona, uk_corona, aus_corona, ca_corona, ch_corona, coronatweets.df, file="allDFs.Rdata")
```

3.2 Creating the Document Term Matrix

```
library(tm)
library(SnowballC)

keywords <- c("corona", "coronavirus", "covid19","covid", "nz", "new", "zealand", "usa", "united", "states",

text <- coronatweets.df$text # prepare text for corpus
text <- gsub("https\\S+", "", text) # http elements
text <- gsub("[^[:alpha:][:space:]]*", "", text) # punctuation
text <- gsub("[\r\n]", "", text) # newline chars


corpus <- Corpus(VectorSource(text)) # create corpus ("documents" of each tweet text)
corpus <- corpus %>%
  tm_map(tolower) %>% #lower case
  # tm_map(removePunctuation) %>% # remove punc (duplicate?)
  tm_map(removeNumbers) %>%
  tm_map(removeWords, stopwords("english")) %>% # remove stopwords
  tm_map(removeWords, keywords) %>% # remove keywords used in searches
  tm_map(stemDocument)


DTM <- DocumentTermMatrix(corpus)

# reduce sparse entries
reduce_DTM <- removeSparseTerms(DTM, 0.99)
```

```
# get frequent words (> 750)
freqWords <- findFreqTerms(DTM, lowfreq = 750)
freqWords

# create word freq matrix dataframe and add sentiment column
word_matrix.df <- as.data.frame(as.matrix(reduce_DTM))
word_matrix.df <- word_matrix.df %>% mutate(sentiment = coronatweets.df$dscrt_sentiment)
```

3.3 Generating the word clouds:

This was achieved by cleaning the text, and then creating a word frequency dataframe for each country set.

```
library(wordcloud)
library(stopwords)
library(tidytext)


gen_wordcloud <- function(tweets.df){
  # remove http elements, punctuation, and \n chars
  tweets.df$text <- gsub("https\\S+", "", tweets.df$text)
  tweets.df$text <- gsub("[^[:alpha:][:space:]]*", "", tweets.df$text)
  tweets.df$text <- gsub("[\r\n]", " ", tweets.df$text)

  tweettext <- tweets.df %>%
    select(text) %>%
    unnest_tokens(word, text)# separate all words in the df

  wordcount <- tweettext %>% anti_join(stop_words) # remove common english words
  wordcount <- wordcount %>% count(word, sort=TRUE) # count and sort word freq in order

  return(wordcount)
}

nz_words <- gen_wordcloud(nz_corona)
us_words <- gen_wordcloud(us_corona)
uk_words <- gen_wordcloud(uk_corona)
ca_words <- gen_wordcloud(ca_corona)
aus_words <- gen_wordcloud(aus_corona)
ch_words <- gen_wordcloud(ch_corona)

head(nz_words,10)
head(us_words,10)

wordcloud(words=nz_words$word, freq=nz_words$n, min.freq = 20, random.order = FALSE)
wordcloud(words=us_words$word, freq=us_words$n, min.freq = 20, random.order = FALSE)
```

Chunk 8.1: Split wordmatrix into train/test sets

```
set.seed(710381321)
# split data
colnames(word_matrix.df) <- make.names(colnames(word_matrix.df))

train_idx <- sample(nrow(word_matrix.df), 0.8*nrow(word_matrix.df))
train_words <- word_matrix.df[train_idx,]
test_words <- word_matrix.df[-train_idx,]
```

Chunk 8.2: Fit Tree

```
library(rpart)
```

```
# use rpart to fit tree to predict sentiment from word matrix
tree <- rpart(sentiment~., data=train_words, method="class", cp=0.001)
# summary(tree)
# plotcp(tree)
# plot(tree)

pred <- predict(tree, test_words, type="class")
pred_tab <- table(actual=test_words$sentiment, pred=pred)
pred_tab
```

```
##           pred
## actual     Negative Neutral Positive
##   Negative      531       5      405
##   Neutral       177      25      300
##   Positive      183      16      735
```

```
# head(pred)
# head(test_words$sentiment)
```

Chunk 8.3: Forest for word matrix

```
library(ranger)
set.seed(710381321)
forest <- ranger(sentiment~., data=train_words, num.trees = 50,
                 verbose = FALSE,
                 importance="impurity", classification = TRUE)
forest
```

```
## Ranger result
##
## Call:
##  ranger(sentiment ~ ., data = train_words, num.trees = 50, verbose = FALSE,      importance = "impurity",
##
## Type:                             Classification
## Number of trees:                  50
## Sample size:                      9508
## Number of independent variables:  221
## Mtry:                             14
## Target node size:                 1
## Variable importance mode:         impurity
## Splitrule:                        gini
## OOB prediction error:             37.44 %
```

```
forest_pred <- predict(forest, test_words)
length(forest_pred$predictions)
```

```
## [1] 2377
```

```
forest_tab <- table(actual=test_words$sentiment,pred=forest_pred$predictions)
forest_tab
```

```
##           pred
## actual     Negative Neutral Positive
##   Negative      704      49      188
##   Neutral       210     163      129
##   Positive      228      57      649
```

```
tail(sort(forest$variable.importance))
```

```
##    peopl      amp    death     case    virus   pandem
## 40.08710 40.10622 44.02810 45.85472 49.91929 50.95929
```

Forest for coronatweets.df dataset

```r
set.seed(710381321)
# split data
colnames(coronatweets.df) <- make.names(colnames(coronatweets.df))

X <- coronatweets.df %>% select(-av_sentiment) %>% select(-text) %>% select(-tidytext)
y <- coronatweets.df$dscrt_sentiment

train_idx2 <- sample(nrow(X), 0.8*nrow(X))
# train_twts <- coronatweets.df[train_idx2,]
# test_twts <- coronatweets.df[-train_idx2,]


forest2 <- ranger(dscrt_sentiment~specified_country+verified+statuses_count+favorite_count+retweet_count+foll
                  verbose = FALSE,
                  importance="impurity", classification = TRUE)
# forest

forest_pred2 <- predict(forest2, X[-train_idx2,])

forest_tab2 <- table(actual=X[-train_idx2,]$dscrt_sentiment,
                     pred=forest_pred2$predictions)
forest_tab2
```

```
##           pred
## actual     Negative Neutral Positive
##   Negative      637      24      280
##   Neutral       338      15      149
##   Positive      483      20      431
```

```r
tail(sort(forest2$variable.importance))
```

```
##          verified specified_country     retweet_count    favorite_count
##          29.68502         232.08335         239.80505         315.12801
##   followers_count    statuses_count
##         832.25778         902.02295
```

```r
names(X)
```

```
##  [1] "screen_name"        "created_at"         "source"
##  [4] "specified_country"  "location"           "favorite_count"
##  [7] "retweet_count"      "followers_count"    "statuses_count"
## [10] "verified"           "account_created_at" "description"
## [13] "wordcount"          "dscrt_sentiment"
```

EOF