Whatsapp Messages Analysis

1.Dataset

Dataset is from when I chatted with my friend on whatsapp. Choosing ‘export chat’ can get the chat transcript. The download is a zip file, which is unzipped in txt format. The dataset includes time, author and message.

2.Data Cleaning

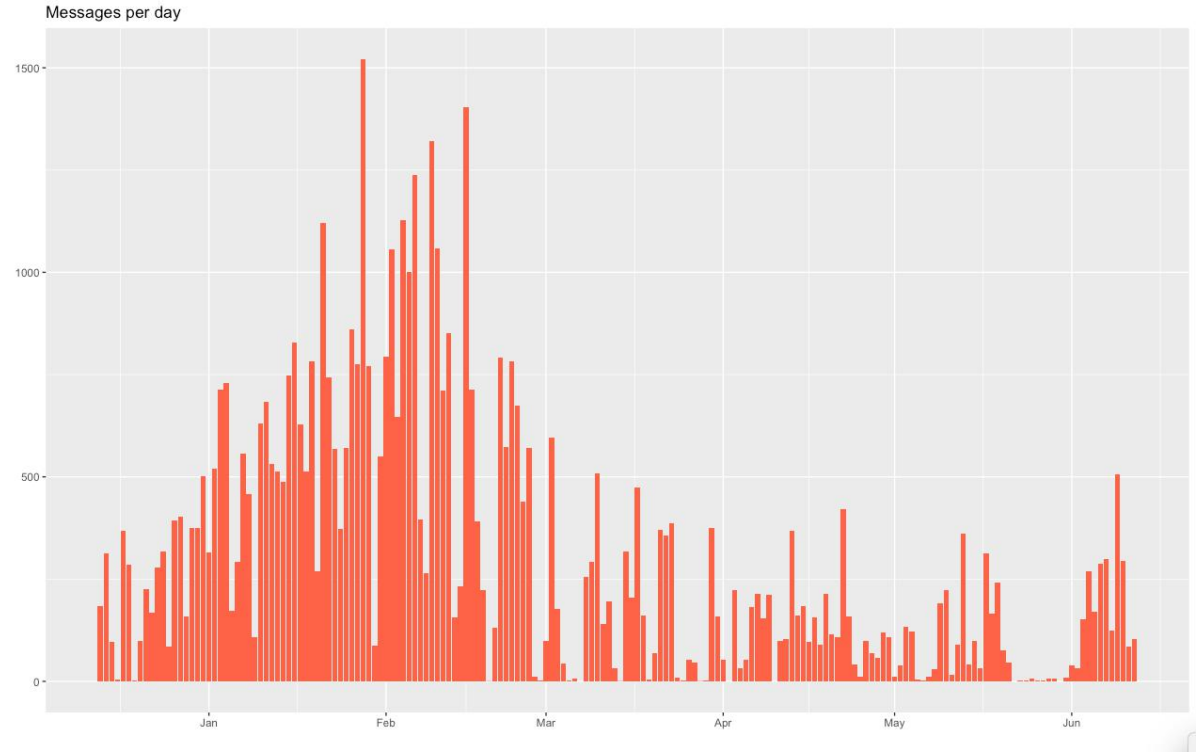
Deleted some unimportant words in the chat transcript. This step accords with everyone's habits.

```
to_remove = c(stopwords(language = "en"), "media","message","deleted","https","www",
              "omitted","ref","Yeah","Yea","yeah","ahahaha","yea","ok","No","no","yes","Ok",
              "Yes","u","hahahaha","ahah","youtube","m.youtube","ahahah","hah","okkkkk",
              "whatttttt","whattttt","yay","mm","nah","d1","oooh","ohh","hhhh","hhhhh",
              "hhhhhh","haha","hhh","oh","hhhhhhh","hhhhhhhh","OKAY","okay")
```

3.Data Visualization

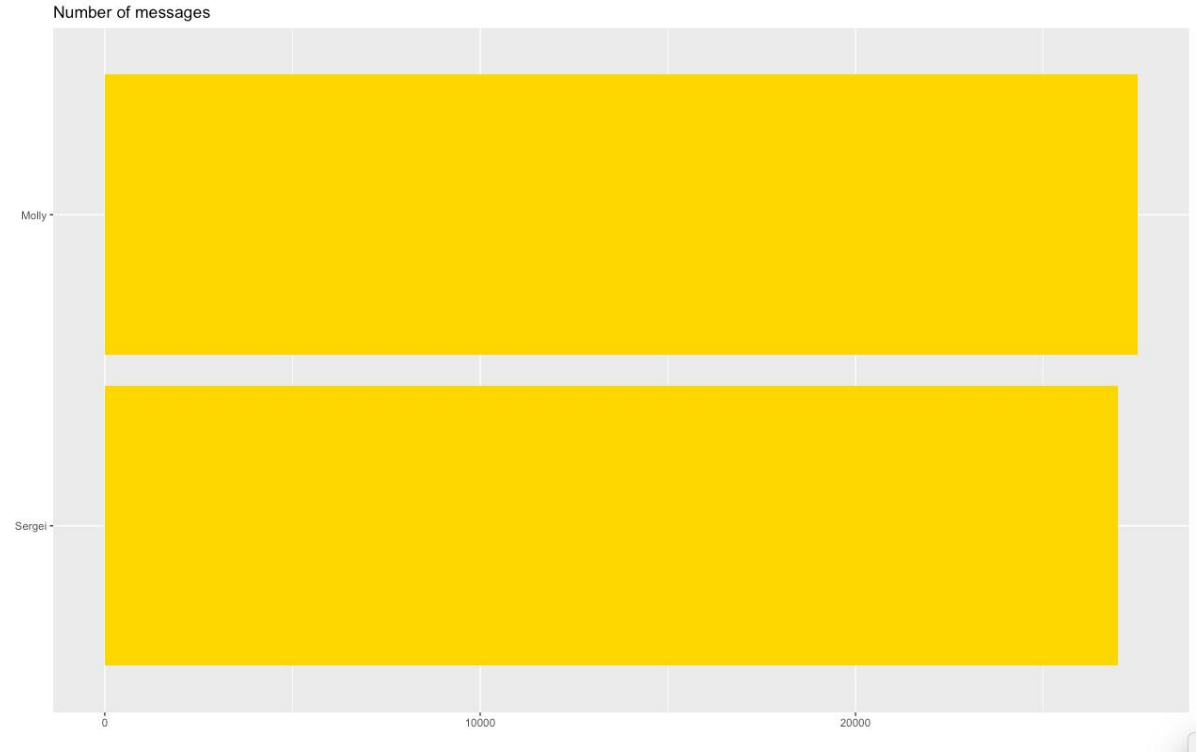
The 3-1 shows the frequency of messages. The chat transcript is from 12/2020 to 06/2021, in total, it has 7 months. From the graph, we can see that the most frequent chat in February, with over 1500 messages in one day. As time went on, contacts gradually decreased.

```
chat %>%
  mutate(day = date(time)) %>%
  count(day) %>%
  ggplot(aes(x = day, y = n)) +
  geom_bar(stat = "identity", fill="#FF6347") +
  ylab("") + xlab("") +
  ggtitle("Messages per day")
```



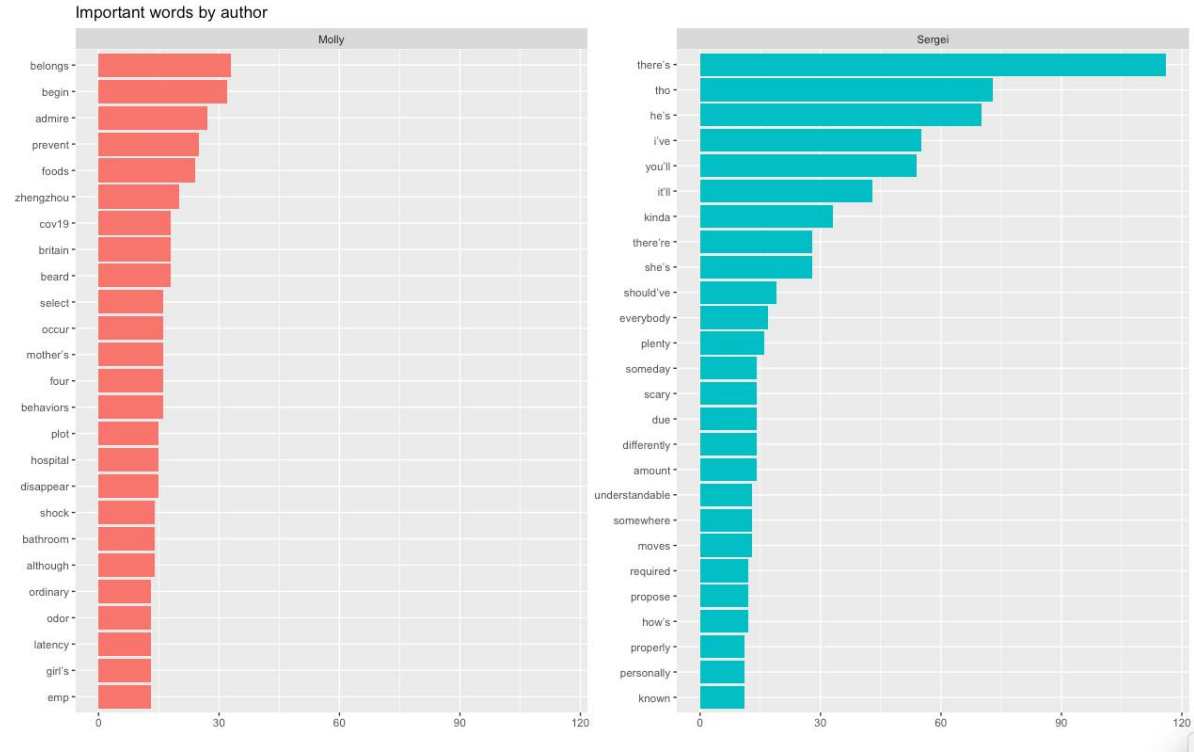
The 3-2 indicates the number of everyone’s messages.

```
chat %>%
  mutate(day = date(time)) %>%
  count(author) %>%
  ggplot(aes(x = reorder(author, n), y=n)) +
  geom_bar(stat = "identity", fill="#FFD700") +
  ylab("") + xlab("") +
  coord_flip() +
  ggtitle("Number of messages")
```



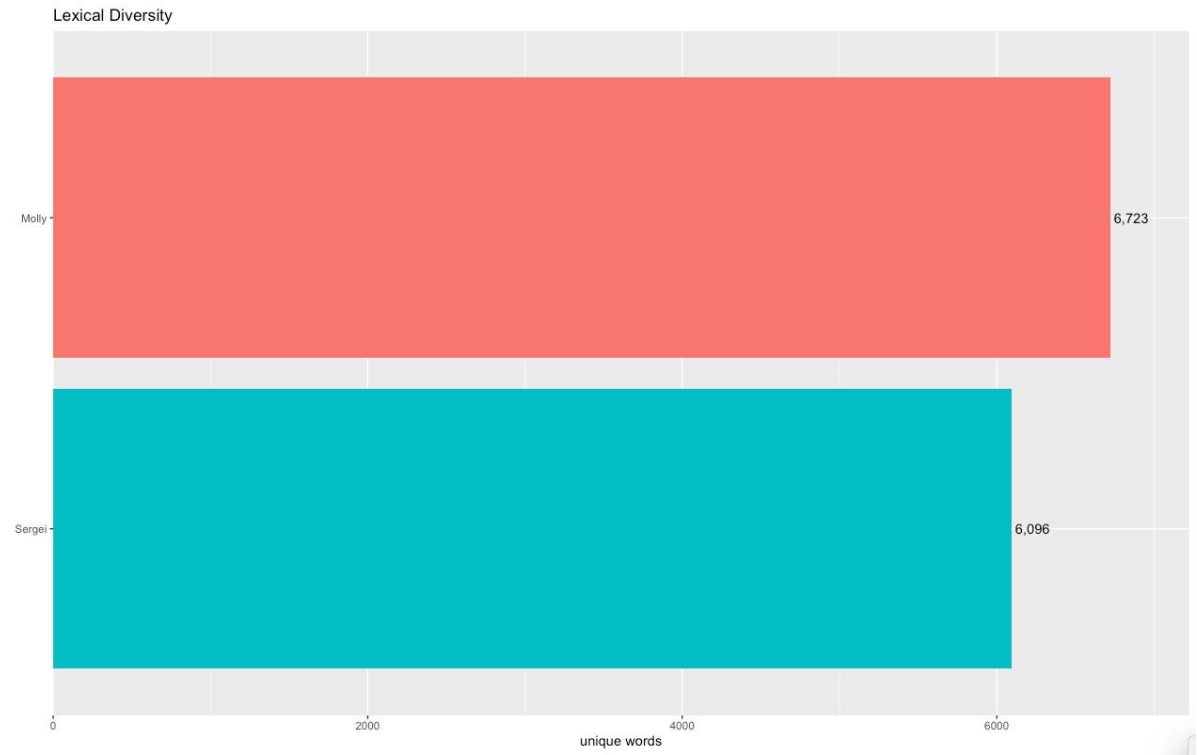
The 3-3 refers to the 25 most common words used by two people, it can be set to show more common words.

```
chatdata=chat
chatdata %>%
  unnest_tokens(input = text,
                output = word) %>%
  select(word, author) %>%
  filter(!word %in% to_remove) %>%
  mutate(word = gsub(".com", "", word)) %>%
  mutate(word = gsub("m.youtube", "", word)) %>%
  mutate(word = gsub("b23", "", word)) %>%
  mutate(word = gsub("d0", "", word)) %>%
  count(author, word, sort = TRUE) %>%
  bind_tf_idf(term = word, document = author, n = n) %>%
  filter(n > 10) %>%
  group_by(author) %>%
  top_n(n = 25, tf_idf) %>%
  ggplot(aes(x = reorder_within(word, n, author), y = n, fill = author)) +
  geom_col(show.legend = FALSE) +
  ylab("") +
  xlab("") +
  coord_flip() +
  facet_wrap(~author, ncol = 2, scales = "free_y") +
  scale_x_reordered() +
  ggtitle("Important words by author")
```

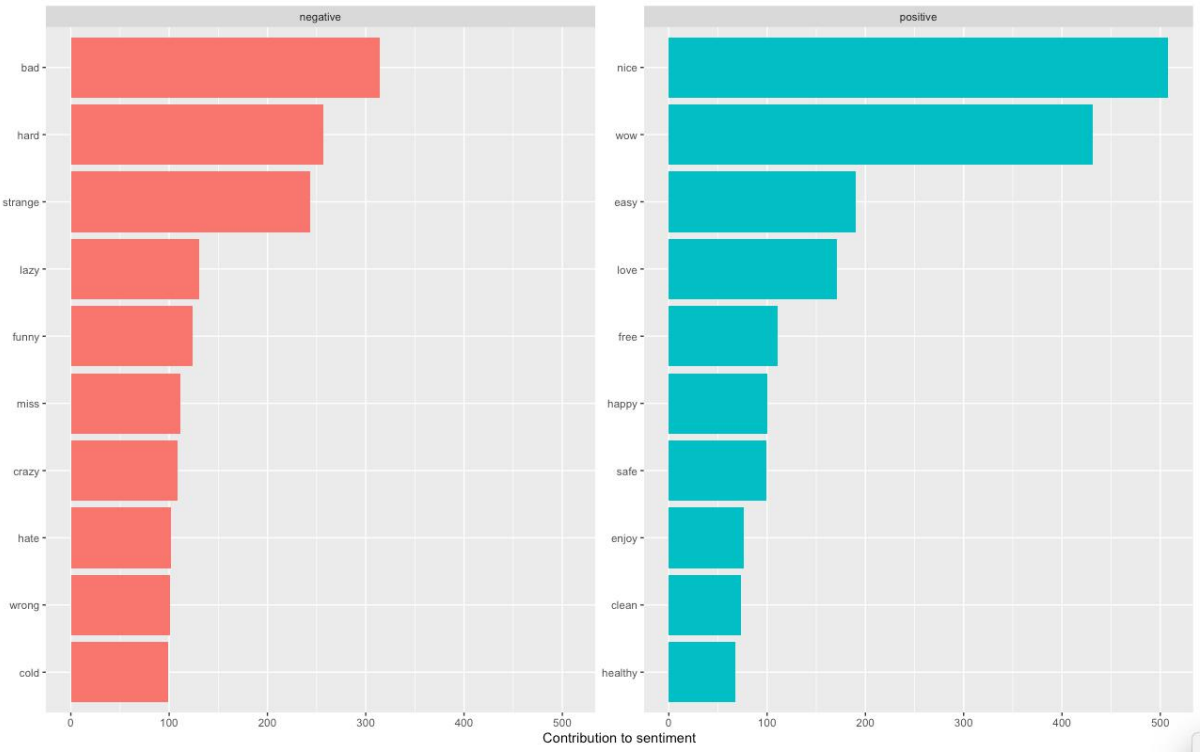


The 3-4 shows everyone's lexical diversity. In chatting, Molly uses a little more vocabulary.

```
chat %>%
  unnest_tokens(input = text,
                output = word) %>%
  filter(!word %in% to_remove) %>%
  group_by(author) %>%
  summarise(lex_diversity = n_distinct(word)) %>%
  arrange(desc(lex_diversity)) %>%
  ggplot(aes(x = reorder(author, lex_diversity),
             y = lex_diversity,
             fill = author)) +
  geom_col(show.legend = FALSE) +
  scale_y_continuous(expand = (mult = c(0, 0, 0, 500))) +
  geom_text(aes(label = scales::comma(lex_diversity)), hjust = -0.1) +
  ylab("unique words") +
  xlab("") +
  ggtitle("Lexical Diversity") +
  coord_flip()
```



The 3-5 shows frequent positive and negative words in chat logs, Bing lexicon includes some sentiment words, which categorises words in a binary fashion into positive and negative categories. So chooses to use the Bing lexicon to categorize words. Using the Bing lexicon needs to install textdata at first.



4.Wordcloud

Using the wordcloud package needs to install wordcloud at first. Then create a new data.frame including words and words' frequent.

```
install.packages("wordcloud")
install.packages("RColorBrewer")
```

```
library(wordcloud)
library(RColorBrewer)
```

```
wordcloud(words = df$word, freq = df$n, min.freq = 5,
          max.words=50, random.order=FALSE, rot.per=0,
          colors=brewer.pal(8, "Dark2"))
```

5.Wordcloud2

Wordcloud2 is an advanced wordcloud, install wordcloud2 package needs to use this code. The wordcloud2 can set a custom image and change the wordcloud's shape. Meanwhile, it can use different fonts to draw the wordcloud. It is more flexible than wordcloud.

```
install.packages("devtools")
devtools::install_github("lchiffon/wordcloud2")
```

```
library(RColorBrewer)
library(wordcloud2)
```

```
495       questions   53
496            shy    53
497        anymore    52
498        arrived    52
499         hungry    52
500        install    52
 [ reached 'max' / getOption("max.print") -- omitted 8762 rows ]
```

```
newdf=df[1:300,]
```

Here, only the most frequently occurring 300 words are selected as a thesaurus.

```
wordcloud2(data=newdf,size = 0.3,
           fontFamily='HanziPen SC',fontWeight='bold',
           color='random-light',backgroundColor="white", shape = 'star')
```


```
wordcloud2(data=newdf, size = 0.7, fontFamily='Phosphate',fontWeight='normal',
           color='random-light',backgroundColor="white",minRotation = -pi/4, maxRotation = pi/4,
            rotateRatio = 0.4,figPath ="couple.png", ellipticity = 0.65, widgetsize = NULL)
```

here, using a custom picture, please make sure your image is PNG. And the color is dark. For example, the image's name is heartt. When using a custom picture, it needs to refresh a lot of times then the R studio Viewer can show the result.

```
wordcloud2(data=newdf, size = 1.3, fontFamily='Snell Roundhand',fontWeight='bold',
           color='random-light',backgroundColor="white",figPath ="heartt.png")
```