Sentiment Analysis

Samyak Agarwal

Whatsapp Sentiment Analysis

Let's proceed with sentiment analysis on my WhatsApp chat data and map the sentiment over time on a daily basis. We'll follow these steps:

```
1.  Load and preprocess the chat data.
2.  Perform sentiment analysis.
3.  Aggregate the sentiment scores by day.
4.  Visualize the results.
```

```r
# Load necessary libraries
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

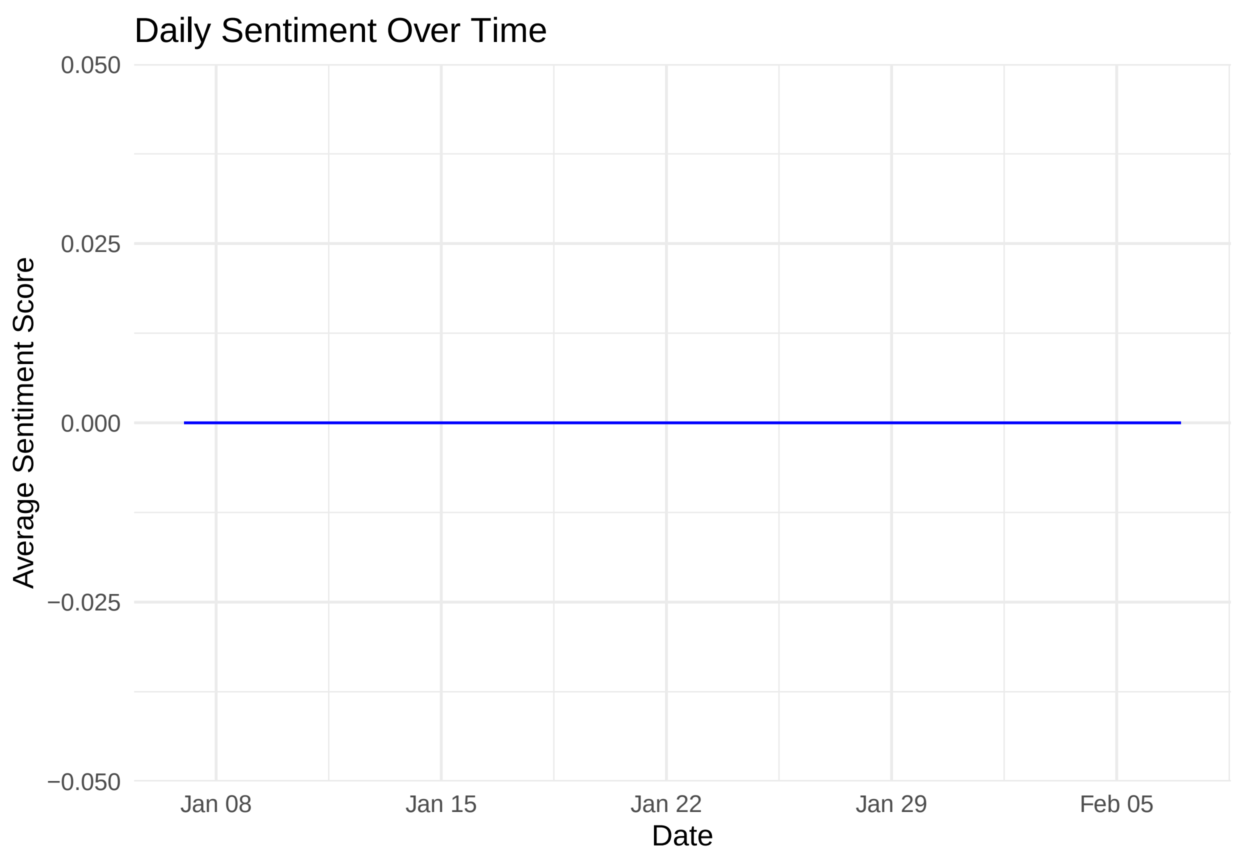
```r
library(syuzhet)
library(ggplot2)

# Load chat data
```

```r
filepath <- "_chat.txt"
chat_data <- readLines(filepath, encoding = "UTF-8")

# Preprocess the data
process_chat <- function(chat_data) {
  # Remove system messages
  chat_data <- chat_data[!grepl("Messages and calls are end-to-end encrypted|sticker omitted|image omit

  # Extract datetime, author, and message
  chat_df <- data.frame(text = chat_data, stringsAsFactors = FALSE) %>%
    separate(text, into = c("datetime", "author_message"), sep = " - ", extra = "merge", fill = "right")
    separate(author_message, into = c("author", "message"), sep = ": ", extra = "merge", fill = "right")
    mutate(datetime = dmy_hms(datetime),
           date = date(datetime))

  return(chat_df)
}

chat_df <- process_chat(chat_data)
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `datetime = dmy_hms(datetime)`.
## Caused by warning:
## !  1458 failed to parse.
```

```r
# Perform sentiment analysis
chat_df$sentiment <- get_sentiment(chat_df$message, method = "syuzhet")

# Classify sentiment as positive, negative, or neutral
chat_df <- chat_df %>%
  mutate(sentiment_type = case_when(
    sentiment > 0 ~ "positive",
    sentiment < 0 ~ "negative",
    TRUE ~ "neutral"
  ))

# Aggregate sentiment by day
daily_sentiment <- chat_df %>%
  group_by(date) %>%
  summarize(daily_sentiment = mean(sentiment, na.rm = TRUE))

# Plot daily sentiment over time
ggplot(daily_sentiment, aes(x = date, y = daily_sentiment)) +
  geom_line(color = "blue") +
  labs(title = "Daily Sentiment Over Time",
       x = "Date",
       y = "Average Sentiment Score") +
  theme_minimal()
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```



As we can see from the results, my chats contains words which are not in english, rather in india we use Hinglish, which are hindi words written using english alphabets, to make sure our sentiment analysis works properly i will attempt to tackle this problem,

Preprocessing the chat data and extracting unique words

```r
# Load necessary libraries
library(dplyr)
library(tidyr)
library(lubridate)
library(stringr)

# Load chat data
filepath <- "_chat.txt"
chat_data <- readLines(filepath, encoding = "UTF-8")

# Preprocess the data with flexible datetime parsing
process_chat <- function(chat_data) {
  # Remove system messages
  chat_data <- chat_data[!grepl("Messages and calls are end-to-end encrypted|sticker omitted|image omit

  # Extract datetime, author, and message
  chat_df <- data.frame(text = chat_data, stringsAsFactors = FALSE) %>%
    separate(text, into = c("datetime", "author_message"), sep = "] ", extra = "merge", fill = "right")
    mutate(datetime = str_remove(datetime, "\\["),
           datetime = mdy_hms(datetime),
           author_message = str_trim(author_message)) %>%
    separate(author_message, into = c("author", "message"), sep = ": ", extra = "merge", fill = "right"

  return(chat_df)
}

chat_df <- process_chat(chat_data)
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `datetime = mdy_hms(datetime)`.
## Caused by warning:
## !  5 failed to parse.
```

```r
head(chat_df)
```

```
##              datetime         author                                    message
## 1 2024-06-26 00:20:41         Blubul      Missed voice call, Click to call back
## 2 2024-06-26 00:36:22         Blubul       Voice call, Answered on other device
## 3 2024-06-26 01:11:50 Samyak Agarwal                                  Hey hotie
## 4 2024-06-26 01:12:21         Blubul                                Waddup cunt
## 5 2024-06-26 01:13:29 Samyak Agarwal           Waps add krna hai kya insta par?
## 6 2024-06-26 01:13:51         Blubul tune insta se bhi hatadiya????
```

```r
# Extract unique words from chat data
unique_words <- unique(unlist(strsplit(tolower(paste(chat_df$message, collapse = " ")), "\\s+")))

# Remove any empty strings
unique_words <- unique_words[unique_words != ""]

# Display the first few unique words
head(unique_words)
```

```
## [1] "missed" "voice"  "call,"  "click"  "to"     "call"
```

Updating the data frame with meanings of the hinglish words

```r
# Recreate the hinglish_to_english data frame
hinglish_to_english <- data.frame(
  hinglish = c("achaa", "haann", "masti", "dosti", "pyaar", "dard", "kutta", "sach", "yaar", "kamina",
               "waps", "krna", "hatadiya", "haas", "rhi", "tu", "ka", "sahi", "vele", "machar", "karna"
               "log", "chal", "haan", "kyu", "fir", "tum", "ye", "kuch", "hai", "gya", "maza", "bta", "
               "kya", "se", "ho", "dekha", "wala", "par", "liya", "hota", "kuch", "ki", "raha", "hai",
               "mein", "ab", "aaj", "kal", "jab", "sirf", "socha", "tujhe", "meri", "tumhari", "use", "
               "dekh", "ja", "pehle", "abhi", "kyunki", "unka", "sab", "se", "bada", "chota", "sabse",
               "tab", "kab", "kaise", "kaha", "koi", "nahi", "haan", "nahi", "vo", "jo", "jaisi", "aisa
               "ek", "do", "teen", "char", "panch", "chhe", "saat", "aath", "nau", "das"),
  English = c("good", "yes", "fun", "friendship", "love", "pain", "dog", "truth", "friend", "rascal",
              "back", "do", "removed", "laugh", "doing", "you", "of", "right", "idle", "mosquito",
              "do", "people", "move", "yes", "why", "then", "you", "this", "some", "is", "went", "fun",
              "tell", "hit", "what", "from", "be", "see", "one who", "on", "taken", "was", "some", "of"
              "doing", "is", "not", "in", "now", "today", "tomorrow", "when", "only", "thought", "you",
              "my", "your", "that", "his", "see", "go", "before", "now", "because", "his", "all", "from
              "big", "small", "biggest", "since", "then", "when", "how", "where", "someone", "no", "yes
              "no", "he", "who", "like", "like this", "one", "two", "three", "four", "five", "six", "se
              "eight", "nine", "ten"),
  stringsAsFactors = FALSE
)

# Verify the structure and content
str(hinglish_to_english)
```

```
## 'data.frame':    93 obs. of  2 variables:
##  $ hinglish: chr  "achaa" "haann" "masti" "dosti" ...
##  $ English : chr  "good" "yes" "fun" "friendship" ...
```

```
print(head(hinglish_to_english))
```

```
##   hinglish    English
## 1    achaa       good
## 2    haann        yes
## 3    masti        fun
## 4    dosti friendship
## 5    pyaar       love
## 6     dard       pain
```

```
# Ensure the English column is a character vector
hinglish_to_english$English <- as.character(hinglish_to_english$English)

# Verify conversion
str(hinglish_to_english$English)
```

```
##  chr [1:93] "good" "yes" "fun" "friendship" "love" "pain" "dog" "truth" ...
```

```
print(head(hinglish_to_english$English))
```

```
## [1] "good"       "yes"        "fun"        "friendship" "love"
## [6] "pain"
```

```
# Test get_nrc_sentiment with a simple character vector
simple_vector <- c("good", "bad", "happy", "sad", "neutral")
simple_sentiment <- get_nrc_sentiment(simple_vector)
print(simple_sentiment)
```

```
##   anger anticipation disgust fear joy sadness surprise trust negative positive
## 1     0            1       0    0   1       0        1     1        0        1
## 2     1            0       1    1   0       1        0     0        1        0
## 3     0            1       0    0   1       0        0     1        0        1
## 4     0            0       0    0   0       0        0     0        0        0
## 5     0            1       0    0   0       0        0     1        0        0
```

Assign Sentiment Scores Using the Correct Column Name:

```r
library(syuzhet)

# Get sentiment scores for English words using NRC lexicon
english_sentiment <- get_nrc_sentiment(hinglish_to_english$English)
hinglish_to_english$sentiment <- english_sentiment$positive - english_sentiment$negative

# Combine with the existing Hinglish lexicon
hinglish_lexicon <- data.frame(
  word = hinglish_to_english$hinglish,
  sentiment = hinglish_to_english$sentiment
)

# Display the updated lexicon
head(hinglish_lexicon)
```

```
##    word sentiment
## 1 achaa         1
## 2 haann         0
## 3 masti         1
## 4 dosti         1
## 5 pyaar         1
## 6  dard        -1
```

Perform Sentiment Analysis Using the Updated Lexicon

```r
# Function to calculate sentiment using Hinglish lexicon
get_hinglish_sentiment <- function(text) {
  words <- unlist(strsplit(tolower(text), "\\s+"))
  sentiments <- hinglish_lexicon$sentiment[match(words, hinglish_lexicon$word)]
  sentiments <- sentiments[!is.na(sentiments)]
  if(length(sentiments) == 0) return(0)
  return(mean(sentiments))
}

# Perform sentiment analysis on each message
chat_df$sentiment <- sapply(chat_df$message, get_hinglish_sentiment)

# Display the first few rows of the processed chat data with sentiment scores
head(chat_df)
```

```
##              datetime         author                                    message
## 1 2024-06-26 00:20:41         Blubul      Missed voice call, Click to call back
## 2 2024-06-26 00:36:22         Blubul       Voice call, Answered on other device
## 3 2024-06-26 01:11:50 Samyak Agarwal                                  Hey hotie
## 4 2024-06-26 01:12:21         Blubul                                Waddup cunt
## 5 2024-06-26 01:13:29 Samyak Agarwal           Waps add krna hai kya insta par?
## 6 2024-06-26 01:13:51         Blubul tune insta se bhi hatadiya????
##   sentiment
## 1          0
## 2          0
## 3          0
## 4          0
## 5          0
## 6          0
```

Aggregate Sentiment Scores in 15-Minute Intervals

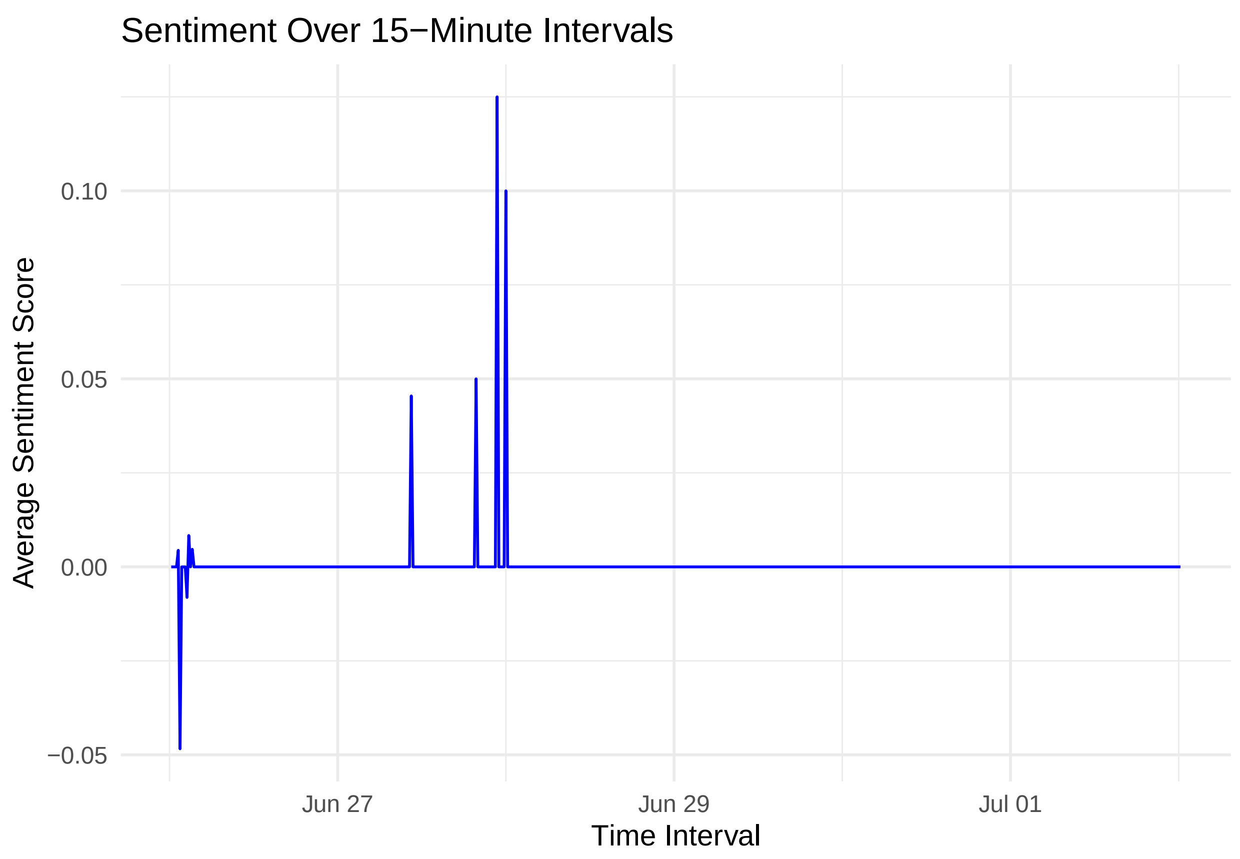
```r
# Aggregate sentiment in 15-minute intervals
chat_df <- chat_df %>%
  mutate(interval = floor_date(datetime, "15 minutes"))

interval_sentiment <- chat_df %>%
  group_by(interval) %>%
  summarize(interval_sentiment = mean(sentiment, na.rm = TRUE))

# Display the aggregated sentiment scores
head(interval_sentiment)
```

```
## # A tibble: 6 x 2
##   interval            interval_sentiment
##   <dttm>                           <dbl>
## 1 2024-06-26 00:15:00            0
## 2 2024-06-26 00:30:00            0
## 3 2024-06-26 01:00:00            0
## 4 2024-06-26 01:15:00            0.00440
## 5 2024-06-26 01:30:00           -0.0484
## 6 2024-06-26 01:45:00            0
```

Visualize the Results

```r
# Plot sentiment over 15-minute intervals
library(ggplot2)

ggplot(interval_sentiment, aes(x = interval, y = interval_sentiment)) +
  geom_line(color = "blue") +
  labs(title = "Sentiment Over 15-Minute Intervals",
       x = "Time Interval",
       y = "Average Sentiment Score") +
  theme_minimal()
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```



Apply Emotion Detection to Each Message, Summarize Emotion Scores Over 15-Minute Intervals and Visualize Emotions Over Time

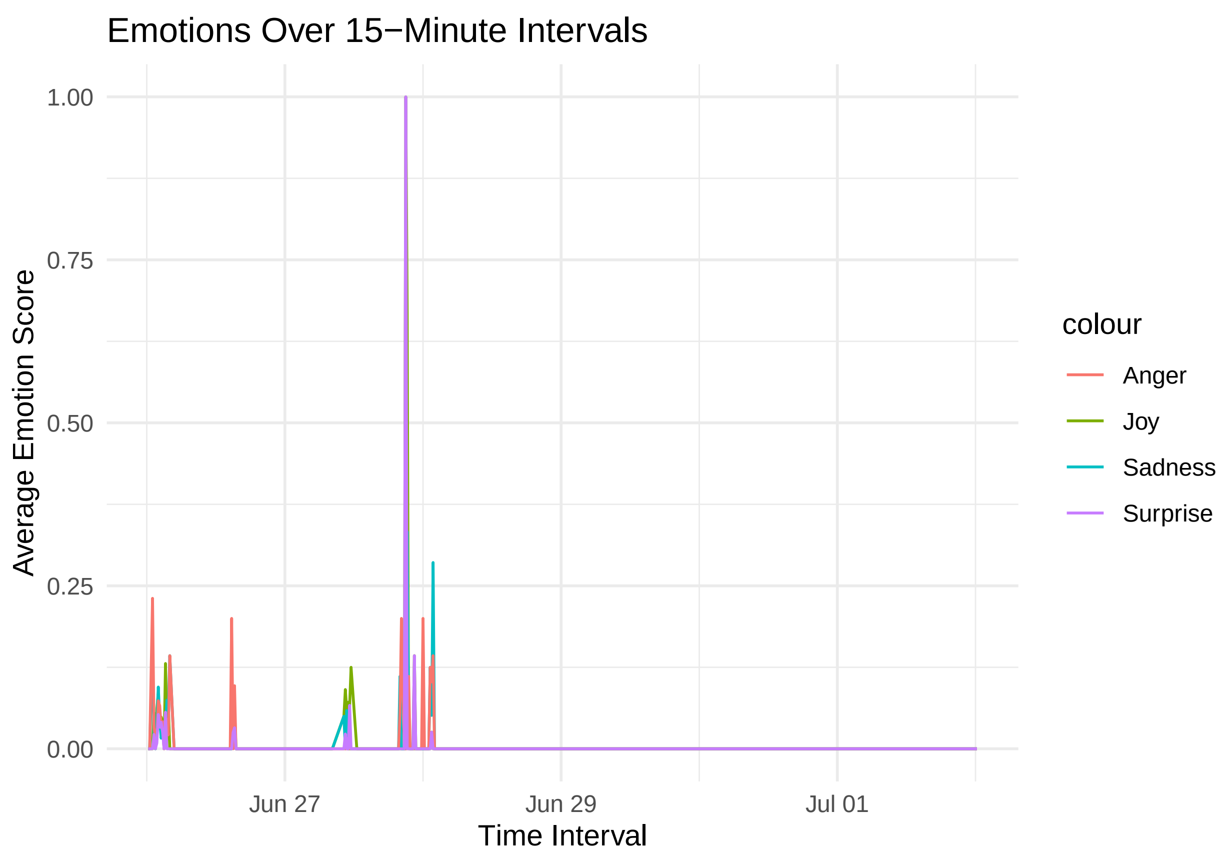
```r
library(syuzhet)
library(dplyr)
library(ggplot2)
library(lubridate)

# Function to detect emotions using the NRC lexicon
get_emotion <- function(text) {
  emotion <- get_nrc_sentiment(text)
  return(emotion)
}

# Apply the emotion detection to each message
emotions <- lapply(chat_df$message, get_emotion)

# Convert the list of emotions to a data frame
emotion_df <- do.call(rbind, emotions)
emotion_df <- cbind(chat_df, emotion_df)

# Summarize emotion scores over 15-minute intervals
emotion_summary <- emotion_df %>%
  mutate(interval = floor_date(datetime, "15 minutes")) %>%
  group_by(interval) %>%
  summarise(across(anger:positive, mean, na.rm = TRUE))
```

```
## Warning: There was 1 warning in `summarise()`.
## i In argument: `across(anger:positive, mean, na.rm = TRUE)`.
## i In group 1: `interval = 2024-06-26 00:15:00`.
## Caused by warning:
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
## Supply arguments directly to `.fns` through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))
```

```r
# Plot emotions over 15-minute intervals
ggplot(emotion_summary, aes(x = interval)) +
  geom_line(aes(y = joy, color = "Joy")) +
  geom_line(aes(y = sadness, color = "Sadness")) +
  geom_line(aes(y = anger, color = "Anger")) +
  geom_line(aes(y = surprise, color = "Surprise")) +
  labs(title = "Emotions Over 15-Minute Intervals",
       x = "Time Interval",
       y = "Average Emotion Score") +
  theme_minimal()
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```



High Surprise Score:

- There is a noticeable spike in the purple line around June 27, indicating a moment of high surprise in the chat messages.

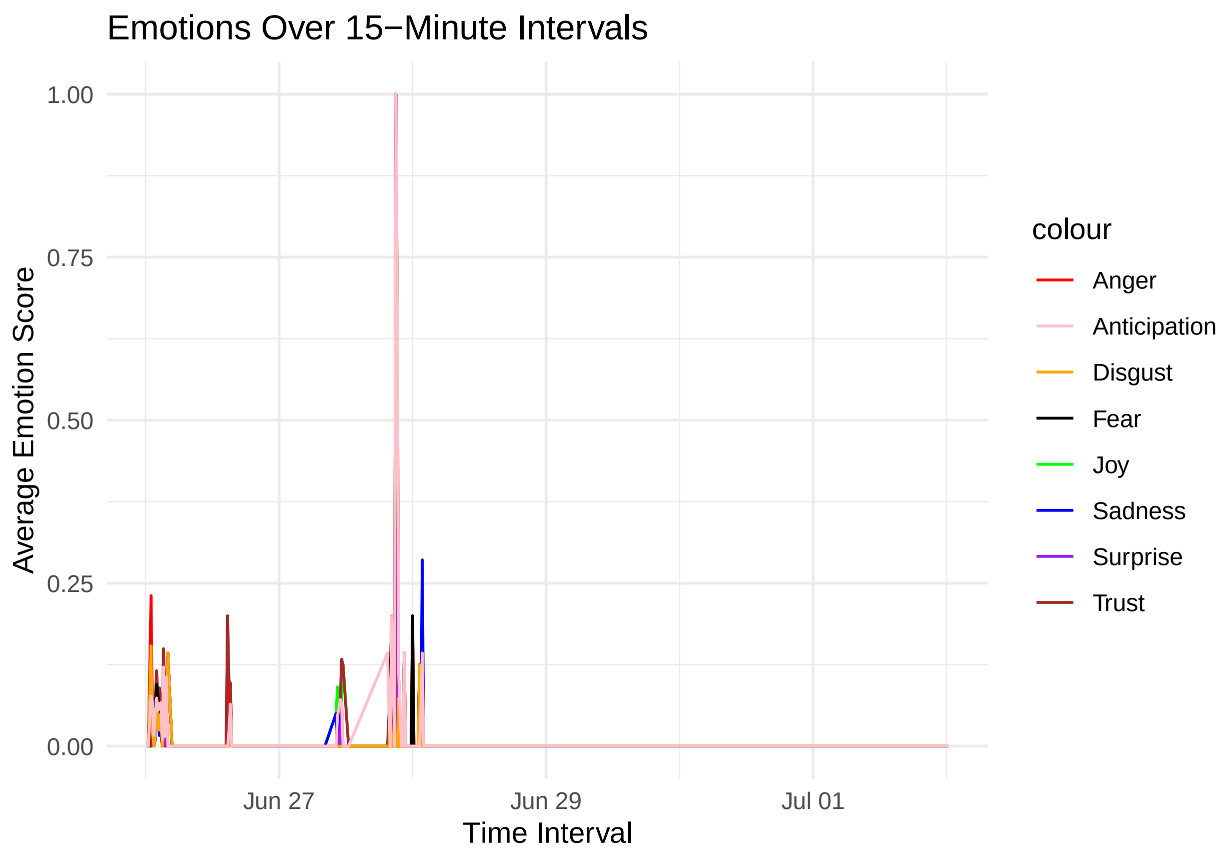
Fluctuating Anger and Joy Scores:

- The red and green lines show fluctuations around June 27, suggesting alternating expressions of anger and joy.

Include More Emotions in the Plot

```r
# Plot more emotions over 15-minute intervals
ggplot(emotion_summary, aes(x = interval)) +
  geom_line(aes(y = joy, color = "Joy")) +
  geom_line(aes(y = sadness, color = "Sadness")) +
  geom_line(aes(y = anger, color = "Anger")) +
  geom_line(aes(y = surprise, color = "Surprise")) +
  geom_line(aes(y = trust, color = "Trust")) +
  geom_line(aes(y = fear, color = "Fear")) +
  geom_line(aes(y = disgust, color = "Disgust")) +
  geom_line(aes(y = anticipation, color = "Anticipation")) +
  labs(title = "Emotions Over 15-Minute Intervals",
       x = "Time Interval",
       y = "Average Emotion Score") +
  theme_minimal() +
  scale_color_manual(values = c("Joy" = "green", "Sadness" = "blue", "Anger" = "red", "Surprise" = "purp
                               "Trust" = "brown", "Fear" = "black", "Disgust" = "orange", "Anticipatio
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```



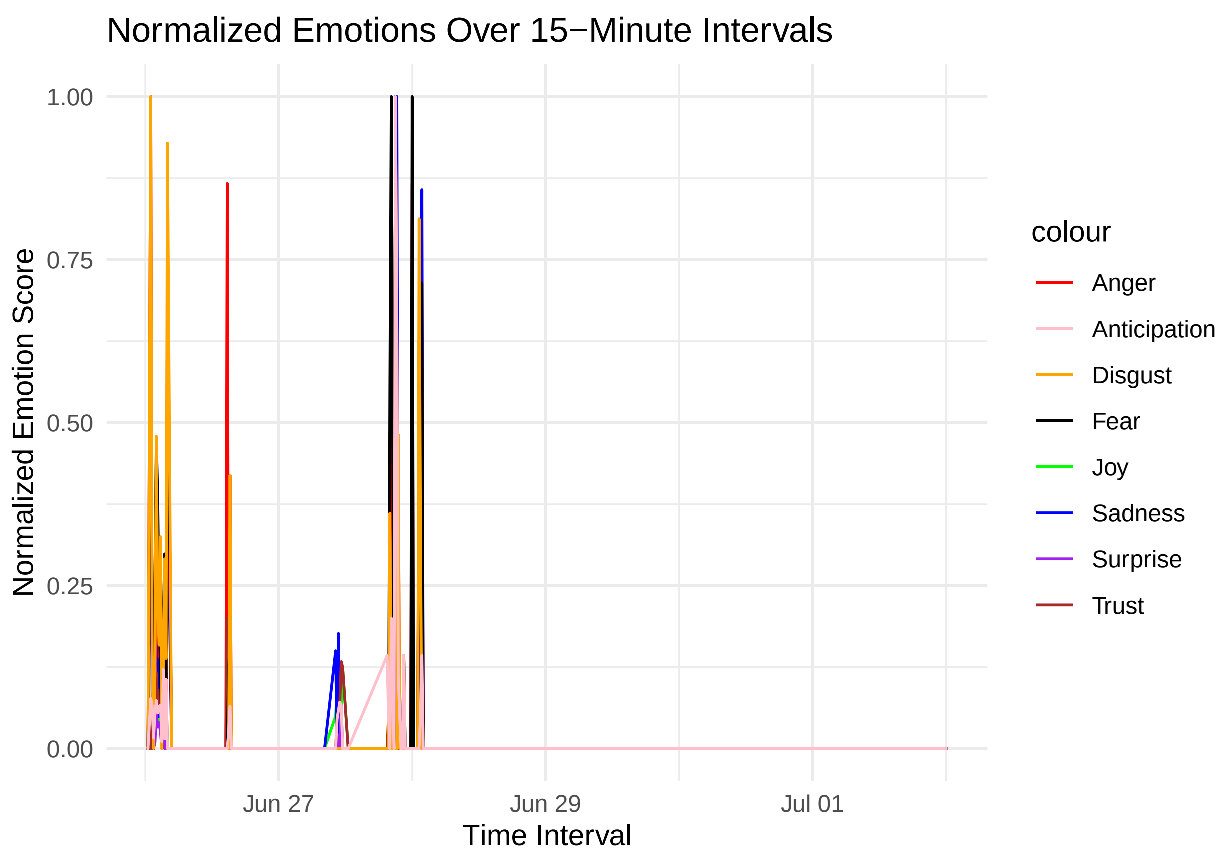
Emotion Distribution:

- Similar to the first plot, this graph shows the distribution and intensity of various emotions over
- Peaks in specific lines indicate higher intensity of that emotion during the respective time interva
- Comparative Analysis:
- Allows for a detailed comparison of the fluctuations of different emotions.

Normalize Emotion Scores

```r
# Normalize emotion scores
emotion_summary_norm <- emotion_summary %>%
  mutate(across(anger:positive, ~ . / max(., na.rm = TRUE)))

# Plot normalized emotions over 15-minute intervals
ggplot(emotion_summary_norm, aes(x = interval)) +
  geom_line(aes(y = joy, color = "Joy")) +
  geom_line(aes(y = sadness, color = "Sadness")) +
  geom_line(aes(y = anger, color = "Anger")) +
  geom_line(aes(y = surprise, color = "Surprise")) +
  geom_line(aes(y = trust, color = "Trust")) +
  geom_line(aes(y = fear, color = "Fear")) +
  geom_line(aes(y = disgust, color = "Disgust")) +
  geom_line(aes(y = anticipation, color = "Anticipation")) +
  labs(title = "Normalized Emotions Over 15-Minute Intervals",
       x = "Time Interval",
       y = "Normalized Emotion Score") +
  theme_minimal() +
  scale_color_manual(values = c("Joy" = "green", "Sadness" = "blue", "Anger" = "red", "Surprise" = "pur
                                "Trust" = "brown", "Fear" = "black", "Disgust" = "orange", "Anticipatio
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```



Normalization:

- Scores are adjusted to a common scale, allowing for a direct comparison of the intensity of different emotions.

Comparative Analysis:

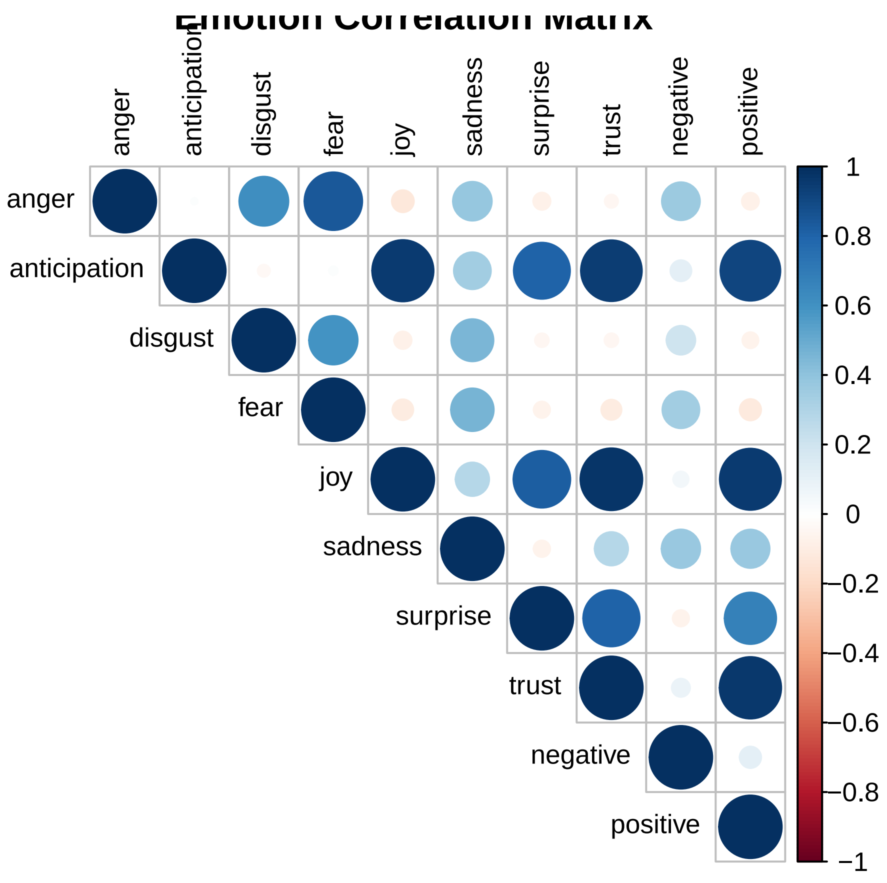
- Easier to identify which emotions are more dominant at specific times.
- For instance, anticipation and fear show significant spikes at certain intervals.

Emotion Correlation Analysis

```r
# Calculate correlation between different emotions
emotion_cor <- cor(emotion_summary %>% select(-interval), use = "complete.obs")

# Plot the correlation matrix
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
corrplot(emotion_cor, method = "circle", type = "upper", tl.col = "black", tl.cex = 0.8, title = "Emoti
```



Key Features:

Diagonal Elements:

- The diagonal elements represent the correlation of each emotion with itself, which is always 1 (perfect correlation).

Off-Diagonal Elements:

- The off-diagonal elements show the correlation between different emotions.
- The color scale ranges from blue (strong positive correlation) to red (strong negative correlation)
- The size of the circles indicates the magnitude of the correlation, with larger circles representing stronger correlations.

Interpretation:

Positive Correlation (Blue Circles):

- Indicates that the two emotions tend to increase and decrease together. For example, if the correlation between "joy" and "trust" is positive, it suggests that when messages express more joy, they also tend to express more trust.

Negative Correlation (Red Circles):

• Indicates that the two emotions tend to move in opposite directions. For example, if the correlation between "anger" and "joy" is negative, it suggests that when messages express more anger, they tend to express less joy.

Strength of Correlation:

• Correlation coefficients closer to 1 or -1 indicate stronger relationships, while those closer to 0 indicate weaker relationships.

Examples from the Plot:

Joy and Trust:

• The correlation between joy and trust is strong and positive, suggesting that messages expressing joy often also express trust.

Anger and Fear:

• The correlation between anger and fear is positive, suggesting that messages expressing anger often also express fear.

Positive and Negative Sentiment:

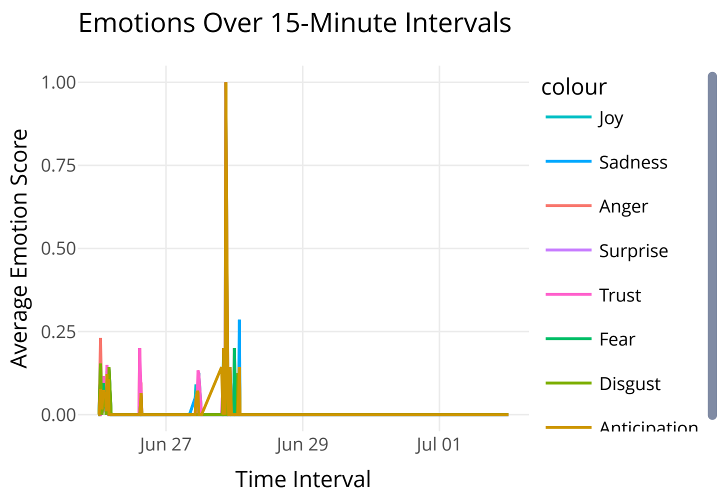
• The correlation between positive and negative sentiment is strong and negative, as expected since positive and negative sentiments typically move in opposite directions.

Interactive Visualization

```r
library(plotly)
```

```
##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```r
# Create an interactive plot
p <- ggplot(emotion_summary, aes(x = interval)) +
  geom_line(aes(y = joy, color = "Joy")) +
  geom_line(aes(y = sadness, color = "Sadness")) +
  geom_line(aes(y = anger, color = "Anger")) +
  geom_line(aes(y = surprise, color = "Surprise")) +
  geom_line(aes(y = trust, color = "Trust")) +
  geom_line(aes(y = fear, color = "Fear")) +
  geom_line(aes(y = disgust, color = "Disgust")) +
  geom_line(aes(y = anticipation, color = "Anticipation")) +
  labs(title = "Emotions Over 15-Minute Intervals",
       x = "Time Interval",
       y = "Average Emotion Score") +
  theme_minimal()

# Convert to interactive plotly plot
ggplotly(p)
```



Emotions Over 15-Minute Intervals
Average Emotion Score
1.00
0.75
0.50
0.25
0.00
Jun 27
Jun 29
Jul 01
Time Interval
colour
Joy
Sadness
Anger
Surprise
Trust
Fear
Disgust
Anticipation

Peaks and Trends:

- Peaks in specific lines indicate higher intensity of that emotion during the respective time intervals.

- The highest peak in anticipation around June 27 indicates a significant spike in messages expressing anticipation.

Comparative Analysis:

- The plot shows how different emotions fluctuate over time, allowing for comparison of their intensity and occurrence.

Sentiment and Emotion Summary Statistics

```r
# Summary statistics for sentiment
sentiment_summary <- chat_df %>%
  summarize(mean_sentiment = mean(sentiment, na.rm = TRUE),
            sd_sentiment = sd(sentiment, na.rm = TRUE))

print(sentiment_summary)
```

```
##   mean_sentiment sd_sentiment
## 1    0.001640465   0.08537382
```

```r
# Summary statistics for emotions
emotion_summary_stats <- emotion_summary %>%
  summarize(across(anger:positive, list(mean = mean, sd = sd), na.rm = TRUE))

print(emotion_summary_stats)
```

```
## # A tibble: 1 x 20
##   anger_mean anger_sd anticipation_mean anticipation_sd disgust_mean disgust_sd
##        <dbl>    <dbl>             <dbl>           <dbl>        <dbl>      <dbl>
## 1     0.0330   0.0606            0.0603           0.155       0.0162     0.0351
## # i 14 more variables: fear_mean <dbl>, fear_sd <dbl>, joy_mean <dbl>,
## #   joy_sd <dbl>, sadness_mean <dbl>, sadness_sd <dbl>, surprise_mean <dbl>,
## #   surprise_sd <dbl>, trust_mean <dbl>, trust_sd <dbl>, negative_mean <dbl>,
## #   negative_sd <dbl>, positive_mean <dbl>, positive_sd <dbl>
```

Detailed Explanation of the Summary Statistics

1. Sentiment Summary Statistics **Description:** The first table displays the summary statistics for the sentiment scores in the WhatsApp chat messages. The statistics include the mean and standard deviation (SD) of the sentiment scores.

Key Features: - **Columns:** - **mean_sentiment:** The average sentiment score across all time intervals. - **sd_sentiment:** The standard deviation of the sentiment scores, indicating the variability in sentiment expression.

Interpretation: - **Mean Sentiment:** - The mean sentiment score of 0.001640465 indicates a neutral average sentiment in the messages, as it is close to zero.

- **Standard Deviation:**
  - The standard deviation of 0.08537382 suggests moderate variability in the sentiment scores, indicating that while the average sentiment is neutral, there are fluctuations in the sentiment intensity.

Applications: - Sentiment Analysis: - These statistics provide an overview of the general sentiment and its variability in the chat data. - Useful for understanding the overall emotional tone and identifying periods of significant sentiment changes.

2. Emotion Summary Statistics Description: The second table displays the summary statistics for the detected emotions in the WhatsApp chat messages. The statistics include the mean and standard deviation (SD) of each emotion.

Key Features: - Columns: - anger_mean: The average score for anger across all time intervals. - **anger_sd:** The standard deviation of the anger scores, indicating the variability in anger expression. - **anticipation_mean:** The average score for anticipation. - **anticipation_sd:** The standard deviation of the anticipation scores. - **disgust_mean:** The average score for disgust. - **disgust_sd:** The standard deviation of the disgust scores. - **fear_mean:** The average score for fear. - The table continues with similar statistics for other emotions such as joy, sadness, surprise, trust, etc.

Interpretation: - Mean Scores: - The mean score represents the average intensity of each emotion across the entire chat data. - For example, an anger_mean of 0.033 indicates a low average intensity of anger in the messages.

- **Standard Deviation:**
  - The standard deviation indicates the variability or spread of the emotion scores.
  - For instance, an anticipation_sd of 0.155 suggests higher variability in anticipation expressions compared to an anger_sd of 0.060.

Applications: - Emotion Analysis: - These statistics provide an overview of the general emotional tone and its variability in the chat data. - Useful for identifying emotions with high intensity and variability, which might indicate key emotional moments.

Summary

The summary statistics tables provide a quantitative overview of the sentiments and emotions in the WhatsApp chat messages. They highlight the average intensity and variability of different sentiments and emotions, offering valuable insights into the emotional dynamics and overall tone of the conversation. These insights can be used for further analysis and to identify key emotional moments within the chat data.