Whats-app Chat Analysis

1/19/2024

Introduction

In this chapter I am going to deal with Whats-app data analysis and visualization.Whats-app is a means of communicating in the modern world using phones and computers. Whats-app is used in modern world as a mean of online help desk by most service companies. Most companies and individuals uses Whats-app for purposes like project management , product advertisement , group communications, etc.The case study is for one of the "____" club group in Nairobi-Kenya.

- I am going to restrict this topic to only text messages exchange using Whats-app for our analysis.
- Dow-loading Whats-app data from Android phones
- The structure of Whats-app chat data
- Reading Whats-app chat data using R
- Visualizing Whats-app chat data.
- Message per day
- Message per weekday
- Radar charts
- Messages per hour
- Heat Map
- Messages per Author
- Emoji analysis
- Word analysis

Objectives

One should be able to download data from Whats-app, Visualize the data download from whats-app.

Whats-app uses a customized version of Extension Messaging and Presence Protocol , Whats-app has more than 2 billion active users in more than 180 countries as of 2020.

Whats-app store all data in servers. It is possible to access this data using API s exposed by whats-app.

Reading Whats-app chat Data in R

One of the ways to read Whats-app chat data using R is with the help of the library . This library provides a function-rwa_read() to read the TXT file containing the Whats-app chat data. The usage is shown as follow.

```
#necessary  libarries
#install.packages("rwhatsapp")
library(rwhatsapp)

v_chats<-rwa_read("C:/Users/User/OneDrive/Desktop/Whatsapp Data Chat Analysis/WhatsApp Chat with Friends of RCNC (1).txt")
# We can view the read data using the head() function as shown below

head(v_chats,5)

## Messages and calls are end -to-end encrypted . No one outside of this chat , not even Whatsapp, can read or listen to them.
```

Visualizing Whatsapp chat data

Before we dive deeper into visualizations, we discuss how to extract some information from the data. For this code to work we need some extra libraries such as lubridate and dplyr as shown below.

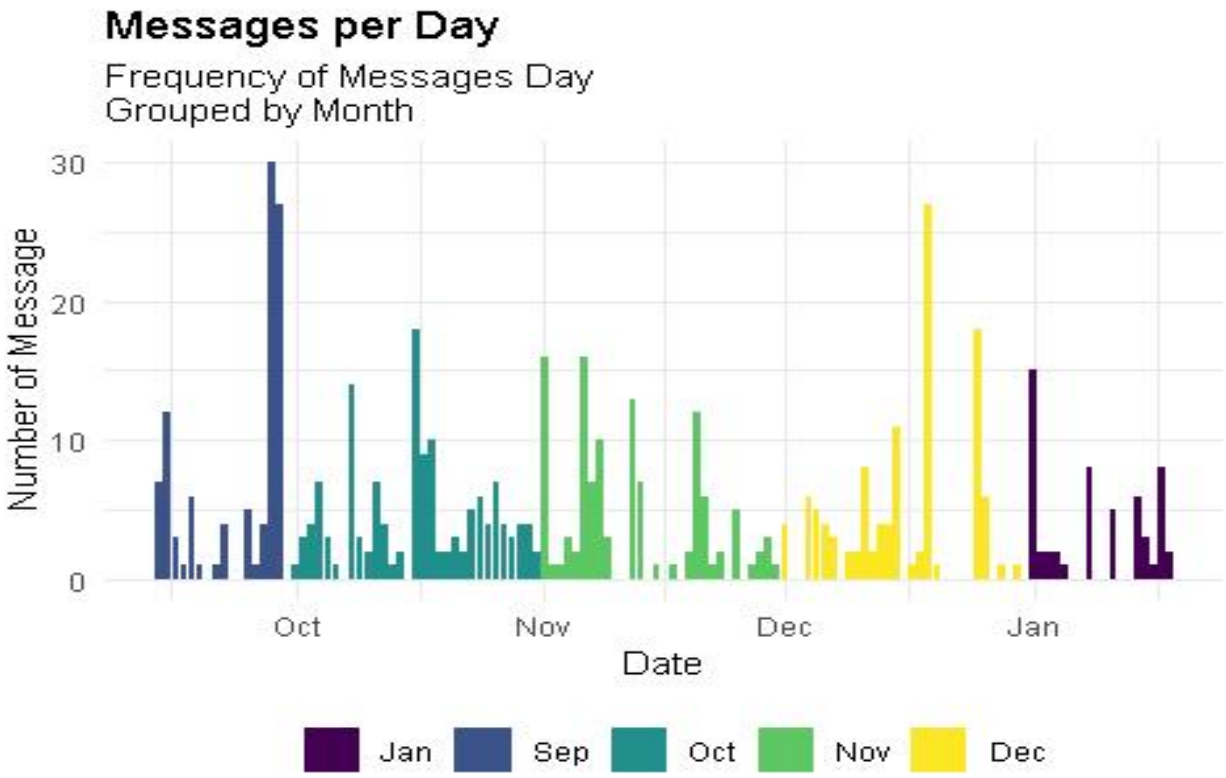
```
#install.packages("lubridate")
#install.packages("tidyverse")
#install.packages("viridis")

#Extracting some important information from the data available
```

```
library(lubridate)

library(tidyverse)

library(viridis)



v_chats<- v_chats %>%
  mutate(message_date= date(time))%>%
  mutate(message_month= month(time,label = TRUE))%>%
  mutate(message_month= factor(message_month))%>%
  mutate(message_weekday_number=wday(message_date))%>%
  mutate(message_weekday_name= weekdays(message_date))%>%
  mutate(message_weekday_name=factor(message_weekday_nam
e))%>%
  mutate(message_hour=hour(time))%>%
  filter(!is.na(author))
```

Message Per Day

The first message we are going to check is the number of messages per day.

```
v_chats %>%
  group_by(message_month)%>%
  count(message_date)%>%
  ggplot(aes(x=message_date, y =n , fill= message_month))+ geom_ba
r(stat = "identity")+
  scale_fill_viridis(discrete=TRUE)+
  labs(x="Date", y ="Number of Message", fill = "Month")+
  ggtitle("Messages per Day","Frequency of Messages Day\nGrouped b
y Month")+
  theme_minimal()+
  theme(legend.title = element_blank(),
        legend.position = "bottom")
```



September 2023 emerged as the month with the highest daily message count, closely trailed by December of the same year. The elevated activity in September can be attributed to the execution of various plans and events. In contrast, the festive season in December contributed to increased interactions among individuals.

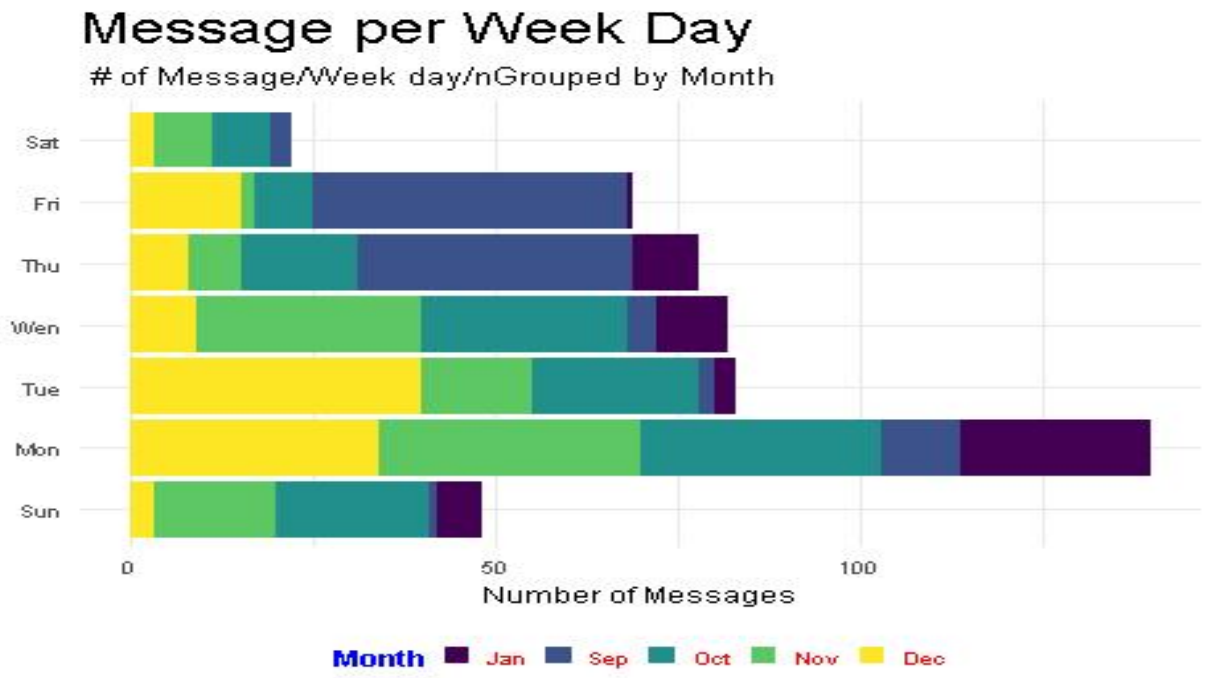
As we transitioned into January 2024, a decline in daily message frequency became evident. However, there is a palpable anticipation of a resurgence in chat activity as the club gears up for its initiatives in the new year. The lull in messages at the beginning of January is expected to be temporary, with the prospect of heightened communication on the horizon.

Message per Weekday

Visualizing the messages per week days using bar-chart.

```
v_chats%>%
  group_by(message_month,message_weekday_number,message_weekday
_name)%>%
  count()%>%
  ggplot(aes(x=reorder(message_weekday_name,message_weekday_num
ber),y=n,fill=message_month))+
```

```
  geom_bar(stat = "identity")+
  scale_x_discrete(labels=c("Monday"="Mon",
                            "Tuesday"="Tue",
                            "Wednesday"="Wen",
                            "Thursday"="Thu",
                            "Friday"="Fri",
                            "Saturday"="Sat",
                            "Sunday"="Sun"))+
  scale_color_viridis(discrete = TRUE)+
  labs(x="", y="Number of Messages", fill="Month")+
  coord_flip()+
  ggtitle("Message per Week Day"," # of Message/Week day/nGroupe
d by Month")+ theme_minimal()+
  theme(legend.title = element_text(color= "blue", size=9, face="bold"),
        legend.text = element_text(color="red",size = 7),
        legend.position = "bottom",
        legend.key.size = unit(0.3,"cm"),
        legend.key.width = unit(0.3,"cm"),
        axis.text.x = element_text(size = 7),
        axis.text.y = element_text(size = 7),
        axis.title = element_text(size = 9),
        plot.title = element_text(size = 18),
        plot.subtitle = element_text(size = 10))
```



Observing the data, it becomes apparent that during September, Thursday and Friday stood out as the days with the highest volume of chats and messages. In October, the majority of conversations occurred on Mondays. Moving into November, Wednesdays and Mondays emerged as the peak days for interactions. December presented a distinctive pattern, with Tuesday being the day of most significant conversation.

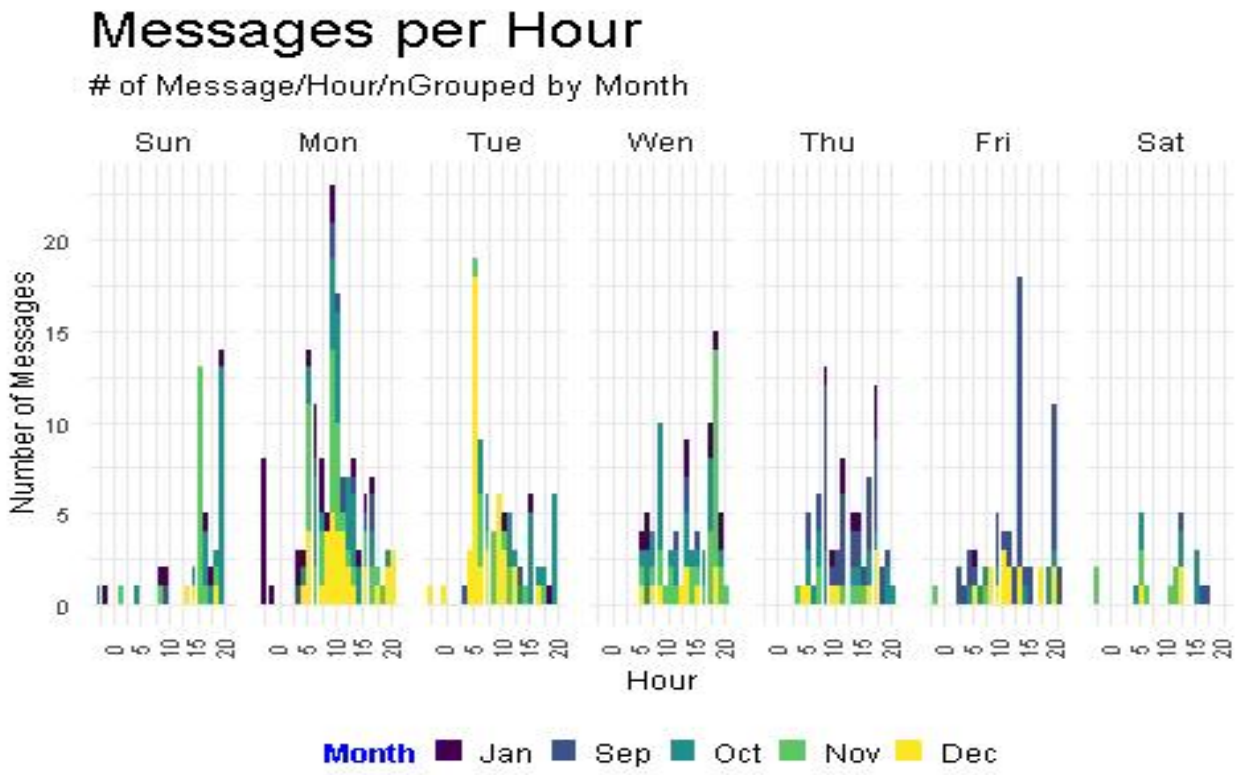
As we transition into the new year, 2024, the trend continues with a notable concentration of conversations on Mondays. It's noteworthy that this observation comes with an acknowledgment that we are still within the first week of the year, leaving ample time for patterns to evolve as we progress through the initial month of 2024.

Messages per Hour

one can visualize these messages per hour and grouping by month and hour.Want to see how messages were exchanged in every month for every weekday in every hour.

```
v_weekdays<-c("Sun","Mon","Tue","Wen","Thu","Fri","Sat")
names(v_weekdays)<-1:7
v_chats%>%
  group_by(message_month,message_weekday_number,message_weekday
_name,message_hour)%>%
  count()%>%
  ggplot(aes(x=message_hour,y=n,fill=message_month))+
  geom_bar(stat = "Identity")+
  scale_fill_viridis(discrete = TRUE)+
  labs(x="Hour", y = "Number of Messages",fill="Month")+
  facet_wrap(~message_weekday_number,ncol = 7,
             labeller = labeller(message_weekday_number=v_weekday
s))+
  ggtitle("Messages per Hour","# of Message/Hour/nGrouped by Month
")+
  theme_minimal()+
  theme(legend.title = element_text(color="blue",size =9,face = "bold"),
        legend.position = "bottom",
        legend.key.size = unit(0.3,"cm"),
        legend.key.width = unit(0.3,"cm"),
        axis.text.x = element_text(size = 7,angle = 90),
```

```
        axis.text.y = element_text(size = 7),
        axis.title = element_text(size = 9),
        plot.title = element_text(size = 18),
        plot.subtitle = element_text(size = 10))
```



It is evident that the majority of conversations are concentrated within the 5-10 hour time-frame, considering the total span of 24 hours in a day. Interestingly, Saturday stands out with the lowest number of messages per hour, as well as the lowest overall daily message count.

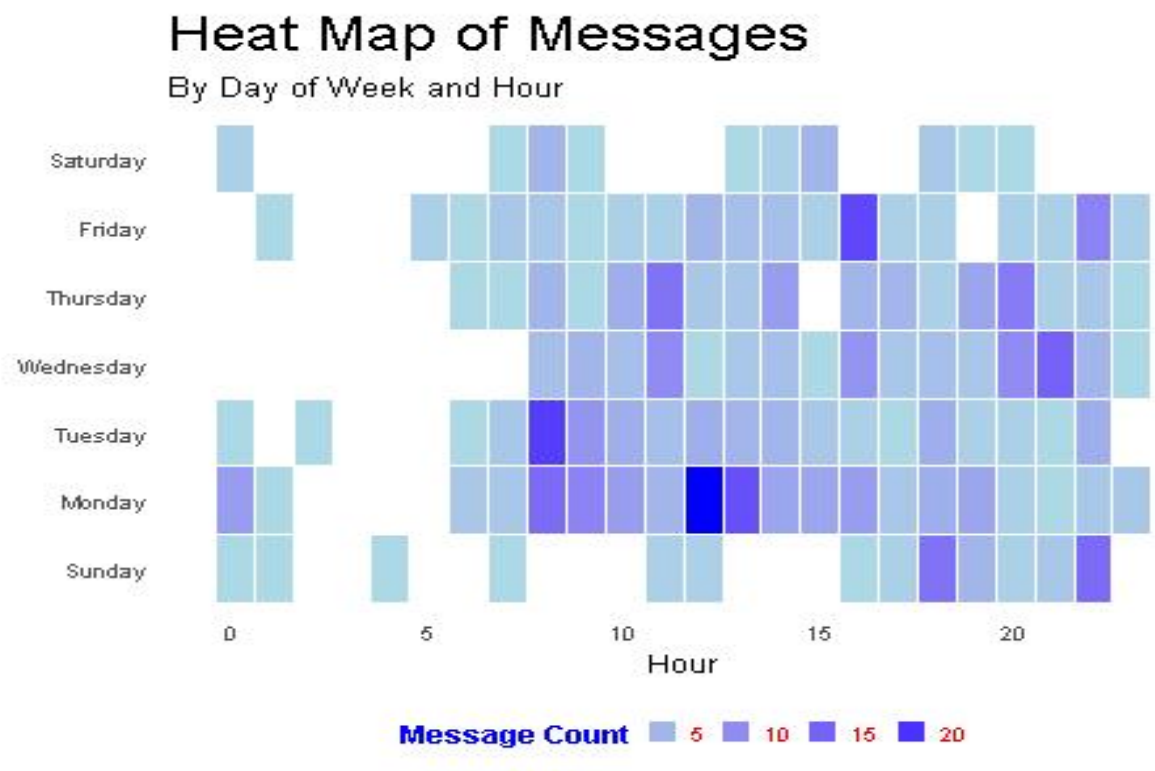
In contrast, Monday, Tuesday, and Friday emerge as the standout days, boasting the highest number of messages per hour. Specifically focusing on January, a notable trend surfaces, with most conversations occurring on Mondays within the 0-10 hour window. This pattern highlights the significance of early-week interactions during the initial month of the year.

Heat Map

One can visualize this data regarding messages per hour in the form of heat map.

```
v_chats%>%
  group_by(message_weekday_number,message_weekday_name,message
```

```
_hour)%>%
  count()%>%
  ggplot(aes(x=message_hour,y=reorder(message_weekday_name,messag
e_weekday_number)))+
  geom_tile(aes(fill= n), color= "white",na.rm = TRUE)+
  scale_fill_gradient(low="lightblue",high="blue")+
  guides(fill=guide_legend(title = "Message Count"))+
  labs(title = "Heat Map of Messages", subtitle = "By Day of Week
and Hour",
       x="Hour",y= "")+
  theme_minimal()+
  theme(legend.title = element_text(colour = "blue",size = 9,face = "b
old"),
        legend.text = element_text(color = "red",size = 7),
        legend.position = "bottom",
        legend.key.size = unit(0.3,"cm"),
        legend.key.width = unit(0.3,"cm"),
        axis.text.x = element_text(size = 7),
        axis.text.y = element_text(size = 7),
        axis.title = element_text(size = 9),
        plot.title = element_text(size = 18),
        plot.subtitle = element_text(size = 10),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())
```



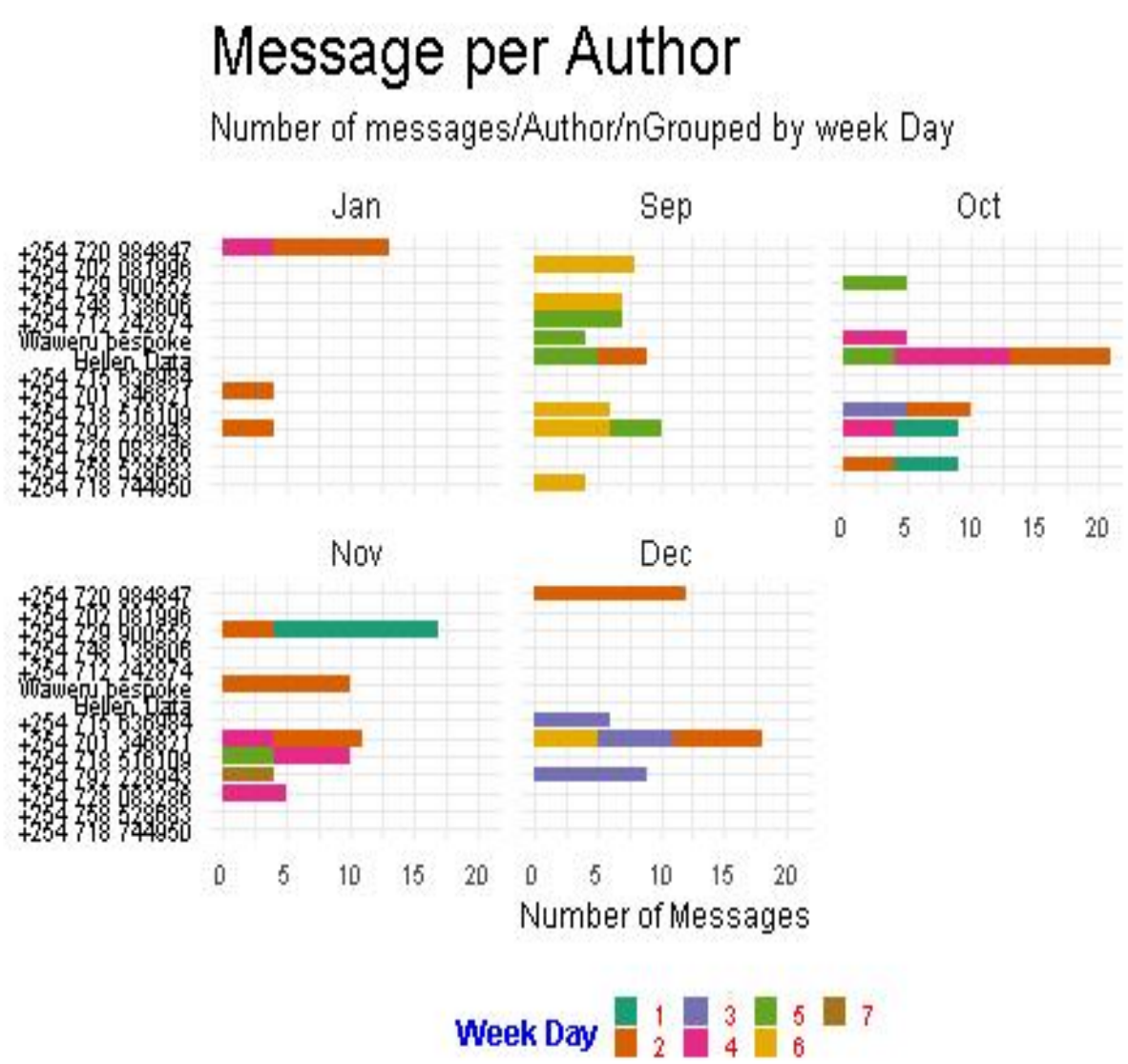
It is evident that Monday, Friday, and Tuesday are the weekdays that consistently exhibit the highest message counts, particularly around the 12-hour mark. These specific days of the week stand out as periods of heightened communication activity, reflecting a noteworthy pattern in the messaging behavior.

Messages per Author

Just diving deeper to see messages per author, per weekdays across all the months.

```
#install.packages("RColorBrewer")
library(RColorBrewer)
library(dplyr)
library(ggplot2)

threshold <- 4   # Set your minimum threshold here

v_chats %>%
  group_by(message_month, message_weekday_number, message_weekday_name, author) %>%
  count() %>%
```

```
  filter(n >= threshold) %>%  # Exclude authors with fewer than 'th
reshold' messages
  ggplot(aes(x = reorder(author, n),
                 y = n,
                 fill = reorder(message_weekday_number, message_week
day_name))) +
  geom_bar(stat = "Identity") +
  scale_fill_manual(labels = c("Monday" = "Mon",
                                        "Tuesday" = "Tue",
                                        "Wednesday" = "Wen",
                                        "Thursday" = "Thus",
                                        "Friday" = "Fri",
                                        "Saturday" = "Sat",
                                        "Sunday" = "Sun"),
                           values = brewer.pal(8, "Dark2")) +
  facet_wrap(~message_month) +
  labs(x = "", y = "Number of Messages", fill = "Week Day") +
  coord_flip() +
  ggtitle("Message per Author", "Number of messages/Author/nGrouped
 by week Day") +
  theme_minimal() +
  theme(legend.title = element_text(colour = "blue", size = 9, face =
"bold"),
          legend.text = element_text(color = "red", size = 7),
          legend.position = "bottom",
          legend.key.size = unit(0.3, "cm"),
          legend.key.width = unit(0.3, "cm"),
          axis.text.x = element_text(size = 7),
          axis.text.y = element_text(size = 7, color = "black"),
          axis.title = element_text(size = 9),
          plot.title = element_text(size = 18),
          plot.subtitle = element_text(size = 10))
```

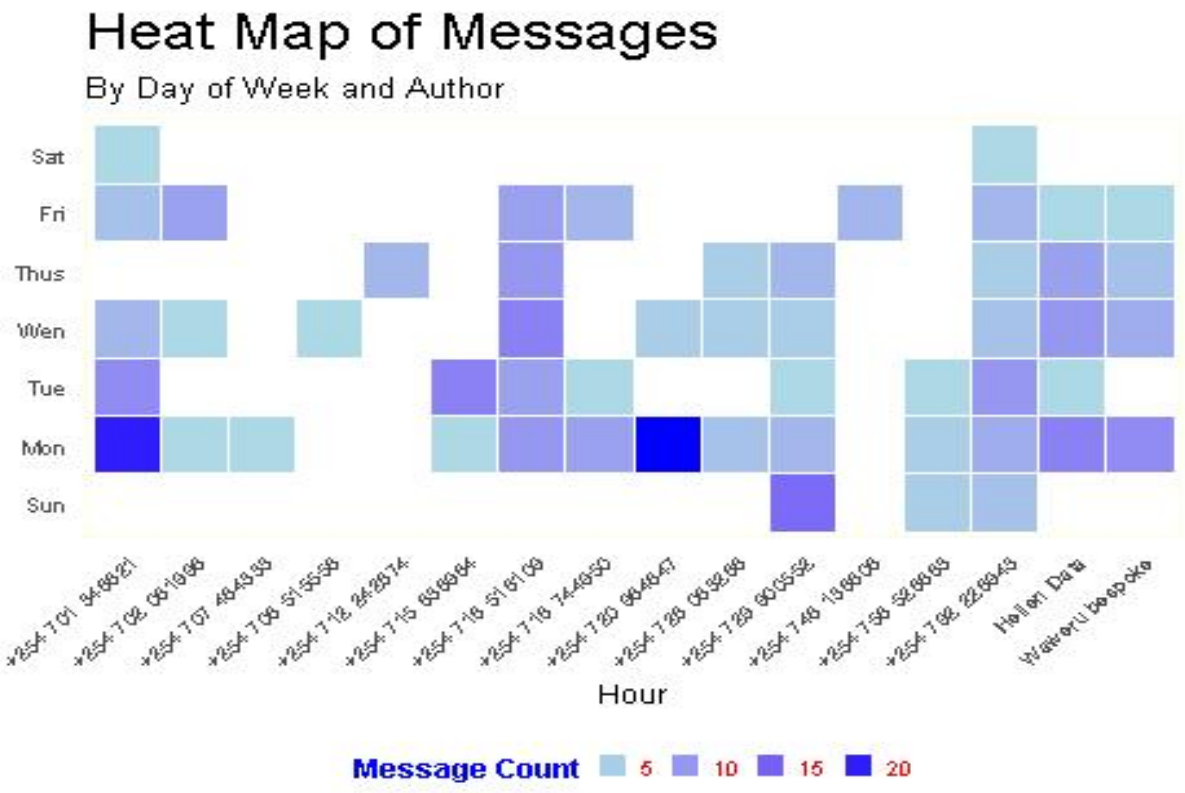


What is apparent from the data is the representation of the number of messages per author. The established threshold filters authors who have contributed more than 4 messages over the past 5 months. This criterion suggests that the individuals surpassing this threshold are likely to be leaders or prominent figures within their respective clubs.

Creating similar thing but now using a Heat map- Messages per author per weekday.

```
# Set your minimum threshold here
threshold <- 4

v_chats %>%
  group_by(message_weekday_number, message_weekday_name, author)
 %>%
  count() %>%
  filter(n >= threshold) %>%  # Exclude authors with fewer than 'th
```

```
reshold' messages
  ggplot(aes(x = author, y = reorder(message_weekday_name, messag
e_weekday_number))) +
  geom_tile(aes(fill = n), color = "white", na.rm = TRUE) +
  scale_fill_gradient(low = "lightblue", high = "blue") +
  scale_y_discrete(labels = c("Monday" = "Mon",
                              "Tuesday" = "Tue",
                              "Wednesday" = "Wen",
                              "Thursday" = "Thus",
                              "Friday" = "Fri",
                              "Saturday" = "Sat",
                              "Sunday" = "Sun")) +
  guides(fill = guide_legend(title = "Message Count")) +
  labs(title = "Heat Map of Messages", subtitle = "By Day of Week
and Author",
       x = "Hour", y = "") +
  theme_minimal() +
  theme(legend.title = element_text(colour = "blue", size = 9, face =
"bold"),
        legend.text = element_text(color = "red", size = 7),
        legend.position = "bottom",
        legend.key.size = unit(0.3, "cm"),
        legend.key.width = unit(0.3, "cm"),
        axis.text.x = element_text(size = 7, angle = 45, hjust = 1),
 # Adjust angle and hjust
        axis.text.y = element_text(size = 7),
        axis.title = element_text(size = 9),
        plot.title = element_text(size = 18),
        plot.subtitle = element_text(size = 10),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(colour = "lightyellow"))
```



On Mondays, there are two authors who stand out with the highest number of chats or messages.

Emoji Analysis

Whats-app data provides two fields with information's on the emojis used in the chats. These fields are emoji and so a lot of analysis is possible on the emoji used in the chats.

The main purpose of this data preparation is to get images of the emojis as the emojis are coded in whats-app chat data.

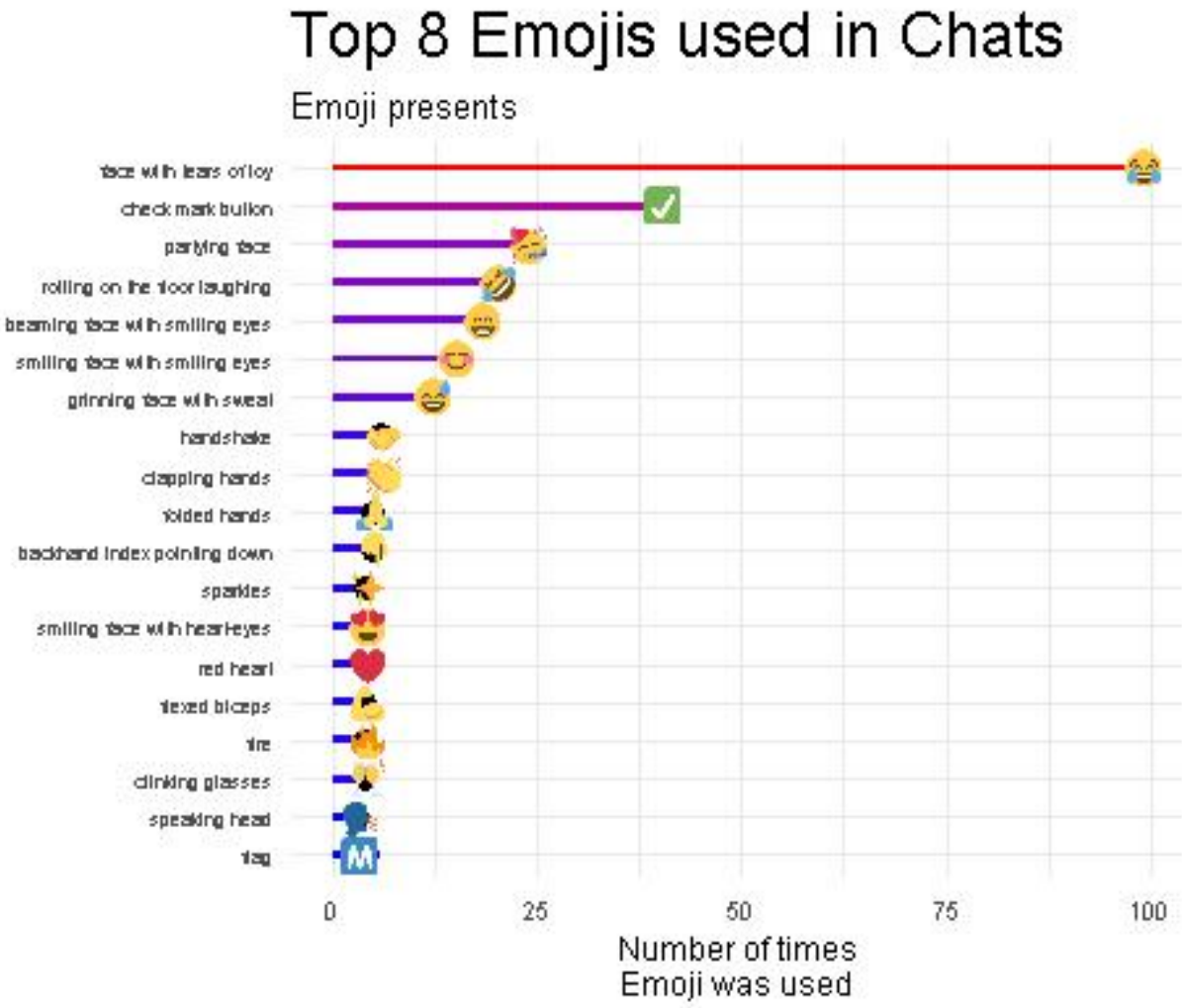
```
head(v_chats[,c("emoji","emoji_name")]%>%
        filter(!is.null(emoji_name))%>%
        filter(emoji_name  !="NULL"))

## clean  data

v_Emojis <- v_chats  %>%
unnest(c(emoji,  emoji_name))  %>%
mutate(emoji  =  str_sub(emoji,  end  =  1))  %>%
mutate(emoji_name  =  str_remove(emoji_name,  ":.*"))  %>%
mutate(emoji_url  =  map_chr(emoji,
```

```
~paste0("https://abs.twimg.com/emoji/v2/72x72/",
as.hexmode(utf8ToInt(.x)), ".png"))
) %>%
filter(!is.null(emoji_name)) %>%
filter(emoji_name != "NULL")

head(v_Emojis)

# time author text source

head(v_Emojis$emoji_url)
```

Emoji Visualizations

```
# Install the ggimage package if you haven't already
#install.packages("ggimage")

# Load the ggimage package
library(ggimage)
library(ggplot2)
library(ggtext)

# Set your threshold here
threshold <- 20

v_Emojis %>%
  count(emoji, emoji_name, emoji_url) %>%
  arrange(desc(n)) %>%
  filter(row_number() <= threshold) %>%  # Keep only the top 8 e
mojis
  ggplot(aes(x = reorder(emoji_name, n), y = n)) +
  geom_col(aes(fill = n), show.legend = FALSE, width = 0.2) +
  geom_point(aes(color = n), show.legend = FALSE, size = 3) +
  geom_image(aes(image = emoji_url), size = 0.05) +
  scale_fill_gradient(low = "blue", high = "red") +
  scale_color_gradient(low = "black", high = "brown") +
  coord_flip() +
  labs(title = "Top 8 Emojis used in Chats", subtitle = "Emoji present
s",
       x = "", y = "Number of times\nEmoji was used") +
  theme_minimal() +
  theme(
```

```
    axis.text.x = element_text(size = 7),
    axis.text.y = element_text(size = 5),
    axis.title = element_text(size = 9),
    plot.title = element_text(size = 18),
    plot.subtitle = element_text(size = 10)
)
```



Here are the top 5 emojis used by authors in this group

- Face with tears of joy
- Check mark button
- Partying face
- Rolling on floor laughing
- Beaming face with smiling face

Conclusion

The analysis of Whats-app data provides a wealth of information.There are hardly any products available in the world as on date (end of 2024) for analysis of Whats-app data. So, this is an area where useful products could be created, and businesses could reformed around the same.