Evidence of the statistical usability of the data generated on whatsapp

Lucainson RAYMOND

June 15, 2021

Abstract

In this paper, we propose to make a quantitative analysis of the data generated on WhatsApp, in other words, of the messages exchanged between the members of a WhatsApp group. We do the work with the chat streams of a WhatsApp group of Haitian developers whose name is "ML Haiti". The objective pursued by said group is the popularization of knowledge in Artificial Intelligence (AI), particularly Machine Learning (ML). In our work, we use Natural Language Processing (NLP) techniques to extract insights from the unstructured data (WhatsApp messages). The NLP techniques, branch of artificial intelligence which combines with Linguistics, allows us to characterize the psycholinguistic schemes of users of the group chat from quantitative models based on theorems and theories (Zip's Law, tf-idf, word embedding ...). From, among other things, the structure of their vocabulary, the N-gram networks of their conversations, the frequency of certain specific words, the use of emoticons, and so forth, we are able to draw up the psychological portrait of the users (on a few aspects), highlighting the historical patterns of their virtual behaviors… And then, we proceeded to an unsupervised classification technique (clustering) to better understand and summarize in some ways the topics discussed in their conversations. Thus, we have employed two (2) well known Machine Learning (ML) techniques. These are the t-Student Stochastic Neighbor Embedding (t-SNE) and the topic modeling based on the Latent Dirichlet Allocation model (LDA). And finally, we used a time series model to be able to predict the number of messages that will be exchanged in the next 10 days. It is therefore an ARIMA(p, d, q) bagged model built around the Box-Cox and Loess-based decomposition (BLD) bootstrap.

Keywords: Natural Language Processing (NLP), Zipf's Law, tf-idf, word embedding, Time series, unsupervised learning, n-gram, Clustering

1. Introduction

The number of users of the Whatsapp network around the world has already reached more than 2 billion (Whatsapp, 2019). Over 5 million businesses use Whatsapp Business services (PYMNTS, 2019). And we exchange sixty-five (65) billion messages on Whatsapp on average per day worldwide (Statista, 2019). These statistics provide no doubt that WhatsApp is a treasure trove of data that can be used to create value and explore our past, understand our present and plan for the future. In addition, the dazzling technological advances in the field of Natural Language Processing (NLP) provide us with cutting-edge tools for analyzing this kind of data (unstructured data) and to extract insights from it. Fascinated by NLP, we therefore have a Data Science (DS) project relating to the manipulation of data of this nature (WhatsApp Data). So we took the opportunity offered by this HarvardX program to work on.

There are many of us on the planet where virtuality is prevalent in our daily life. We hold business meetings online; we store our archives in the cloud; we inform ourselves online; we have fun online; we meet online … In the end, our civilization is at the point of giving shape in the virtual. Around this state of affairs, we never stop developing technological tools to understand life in the virtual world. And disciplines from various backgrounds benefit from it or are directly involved in this process.

The WhatsApp network, of interest to us, is the subject of our present analysis. As the title of our little work indicates, we want to show the analytical possibilities that we have with the tools of Natural Language Processing (NLP) to search the data generated on said network. With this analysis, we will have a good perspective on the movements of the group; the pace and tone of the exchanges …

To carry out these explorations and analyzes, we will mainly use the R language (through R Studio) with all the appropriate packages. We will adopt a step-by-step approach where operations will be carried out sequentially.

Our first step is to assume that the corpus (the set of all messages) is a collection of words where the order does not influence our analysis. We mainly consider unigrams (the bag-of-wors approach); for some analyzes we also use bigrams to provide more meaningful results. And, as a matter of precision, our analysis is recorded in the period from 2019-09-21 to 2021-06-06. Also note that the group's developers mainly use English as the language of communication, sometimes with a mixture of haitian Creole words. Either way, we assume that English is dominant in conversions and that Creole is negligible and that it will not influence linguistic analyzes.

2. Methodology

As mentioned earlier, we have a modest job related to the data generated on WhatsApp. The objective pursued here is to illustrate some of the possibilities of apprehending behavioral schemes by working on the data of a public WhatsApp group of Haitian developers. The goal is to apply the pair (Computational Linguistics & Statistics/ML) justly for detecting some psycholinguistic patterns. Our work revolves around four (4) axes. A first where we do an exploratory analysis of the data set. A second where we perform a behavioral analysis, a third where we do a linguistic analysis and finally, a fourth where we specify a model for forecasting the number of messages arriving daily on the group.

3. Exploration of the dataset (file exported from WhatsApp)

```r
#Loading packages
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(ggthemes))
suppressPackageStartupMessages(library(ggrepel))
suppressPackageStartupMessages(library(gganimate))
suppressPackageStartupMessages(library(ggimage))
suppressPackageStartupMessages(library(plotly))
suppressPackageStartupMessages(library(RColorBrewer))
suppressPackageStartupMessages(library(factoextra))
suppressPackageStartupMessages(library(scales))
suppressPackageStartupMessages(library(rwhatsapp))
suppressPackageStartupMessages(library(stopwords))
suppressPackageStartupMessages(library(NLP))
suppressPackageStartupMessages(library(tidytext))
suppressPackageStartupMessages(library(tidylo))
suppressPackageStartupMessages(library(stringr))
suppressPackageStartupMessages(library(tm))
suppressPackageStartupMessages(library(topicmodels))
suppressPackageStartupMessages(library(Rmpfr))
suppressPackageStartupMessages(library(tidyr))
suppressPackageStartupMessages(library(widyr))
suppressPackageStartupMessages(library(Metrics))
suppressPackageStartupMessages(library(tseries))
suppressPackageStartupMessages(library(forecast))
suppressPackageStartupMessages(library(lubridate))
suppressPackageStartupMessages(library(corrr))
suppressPackageStartupMessages(library(cluster))
suppressPackageStartupMessages(library(Rtsne))
suppressPackageStartupMessages(library(tsne))
suppressPackageStartupMessages(library(ggraph))
suppressPackageStartupMessages(library(igraph))
suppressPackageStartupMessages(library(wordcloud2))
suppressPackageStartupMessages(library(stringr))
suppressPackageStartupMessages(library(urltools))
suppressPackageStartupMessages(library(Rgraphviz))
suppressPackageStartupMessages(library(visNetwork))
suppressPackageStartupMessages(library(DT))
suppressPackageStartupMessages(library(data.table))
```

3.1. Importing the dataset

To read the dataset in the R environment, we call the rwa_read() function of the rwhatsapp package created by Johannes GRUBER. This saves us a bit of "string manipulation" which is a very time-consuming task.

```r
#Reading
dataset<-rwa_read("_chat.txt")
```

3.2 Description of the data

9,929 messages (observations) are exchanged between the developpers over 485 days, i.e. from 2019-09-21 to 2021-06-06.

```r
dataset%>%
  glimpse()

## Rows: 9,929
## Columns: 7
## $ time       <dttm> 2019-09-21 01:45:35, 2019-09-21 01:46:46, 2019-09-21 0
```

```
1:47~
## $ author     <fct> NA, NA, "PierreRobentz Cassion,ENG", "Doertley TELFORT"
, "T~
## $ text       <chr> "<U+200E>Madsen Servius added you", "<U+200E>You're now
an admin", "Thank~
## $ source     <chr> "_chat.txt", "_chat.txt", "_chat.txt", "_chat.txt", "_c
hat.~
## $ id         <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
17, ~
## $ emoji      <list> [NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NUL~
## $ emoji_name <list> [NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NUL~
```

3.3 Some basic statistics

Theses number are for the period up to `2019-09-21`, `01:45:35` (to be more precise). It is assumed that there are rather dynamic over time.

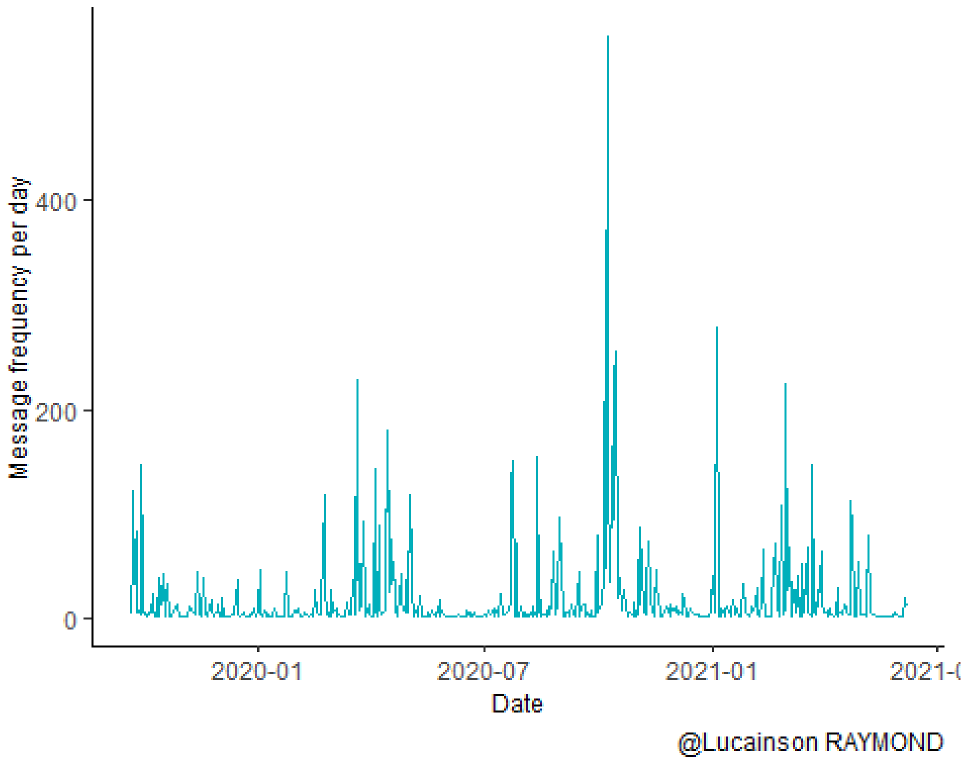
```
#Active users
n_author<-nlevels(dataset$author)

#Total number of messages
n_messages<-length(dataset$text)

#Number of suppressed messages
msgdeleted="This message was deleted"
msg_you="You deleted this message"
msg_supp1=grep(msgdeleted, dataset$text,value=T)
msg_supp2=grep(msg_you, dataset$text,value=T)
n_supp_mess<-print(length(msg_supp1)+length(msg_supp2))

#Number of emojis
n_emojis<-length((unlist(dataset$emoji_name)))

#Dataframe
df_stat<-data.frame(var=c("Active users","Total number of messages","Total nu
mber of deleted messages","Total number of used emojis"),
                    frequency=c(n_author,n_messages,n_supp_mess,n_emojis))

df_stat %>%
  knitr::kable()%>%
    kable_styling(bootstrap_options = "striped" , full_width = F , position =
"center") %>%
  kable_styling(bootstrap_options = "bordered", full_width = F , position ="c
enter") %>%
  column_spec(1,bold = T ) %>%
  column_spec(2,bold =T ,color = "white" , background ="blue")
```

variable	frequency
Active users	143
Total number of messages	9929
Total number of deleted messages	182
Total number of used emojis	1655

3.4 Chronological series of messages

In this part, the aim is to visualize the series of exchanges going from 2019-09-21 to 2021-06-06. This will be an opportunity to graphically detect possible patterns (trend, seasonality) in the temporal flows of said exchanges.

```r
dataset%>%
  mutate(day = as.Date.factor(time)) %>%
  mutate(year=year(day))%>%
  filter(!is.na(year))%>%
  count(day) %>%
  ggplot(aes(x = day, y = n),fill="#00AFBB")+
  geom_line(stat = "identity",
                color= "#00AFBB",size=0.5) +
  ylab("Message frequency per day") +
  xlab("Date") +
  labs(caption = "@Lucainson RAYMOND")+
  theme_classic()+
  theme(
      axis.title.x = element_text(family="Cambria",size=9),
      axis.title.y = element_text(family="Cambria",size=9))
```

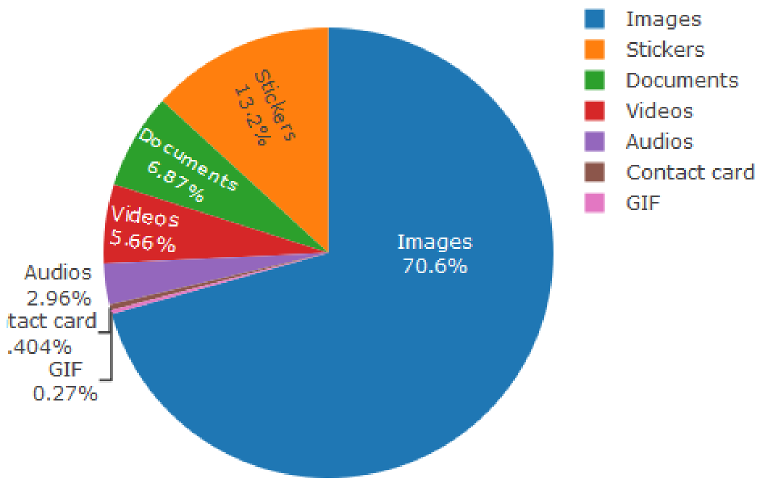


3.5 Statistics of media files shared on the group

For the period submitted to our analysis, here is the distribution of multimedia files shared on the group. The circular diagram (Camembert) below illustrates the distribution of image files, videos, documents, stickers, contact cards & GIFs, in the exchanges of the developers.

```
#We are going to do some string manipulation in this part using RegeX techniq
ues.
# First, we will extract the traces from the media files by highlighting thei
r patterns in the dataset
image=".*image omitted$"
video=".*video omitted$"
document=".*document omitted$"
contact=".*Contact card omitted$"
audio=".*audio omitted$"
gif=".*GIF omitted$"
sticker=".*sticker omitted$"


#Now using the function grep () which is a function of the "base" package of
R, we will extract all of said traces as follows
im=grep(image, dataset$text)
vid=grep(video, dataset$text)
doc=grep(document, dataset$text)
cont=grep(contact, dataset$text)
aud=grep(audio, dataset$text)
```

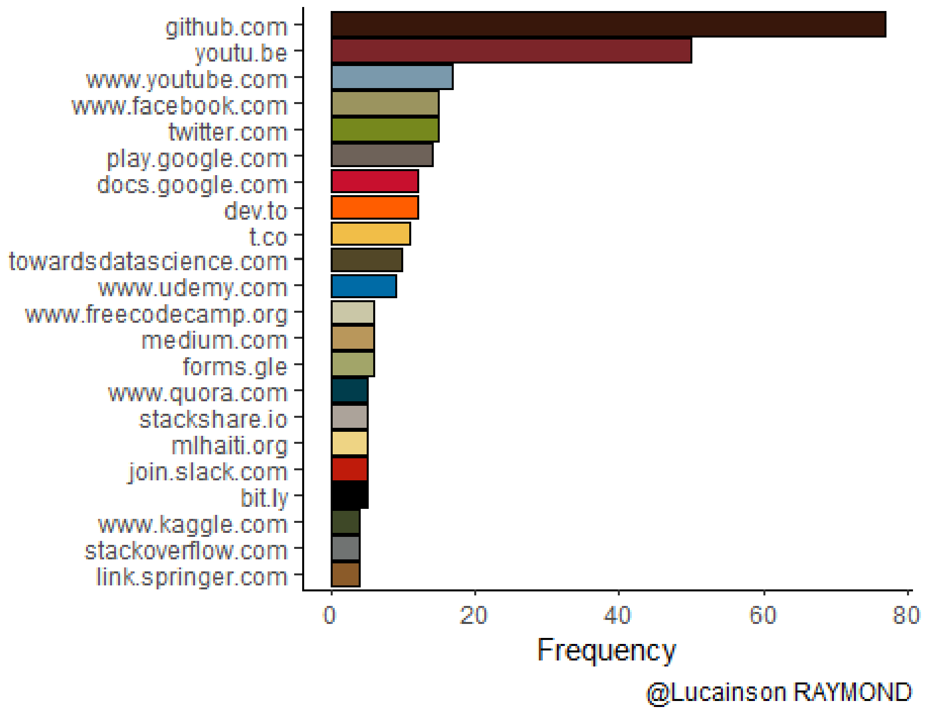
```
gf=grep(gif, dataset$text)
stick=grep(sticker, dataset$text)

#Then we build a vector of data with the frequency of sharing each category o
f files while formally labeling them for convenience of reading
media<-c(length(im),length(vid),
         length(doc),length(cont),length(aud),
         length(gf),length(stick))

labels<-c("Images","Videos",
          "Documents","Contact card",
          "Audios","GIF","Stickers")

med1<-tibble(media,labels)

fig <- plot_ly(med1, labels = ~labels, values = ~media, type = 'pie',textinfo
='label+percent',
                                 insidetextorientation='radial')

fig
```



3.6 Top_20 websites shared on the group

```
#We are going to do string manipulation again
#We are therefore going to define the regular pattern (Regular expression) of
websites with the syntax R. Then, we are going to create two (2) additional c
```

```
olumns (url_pattern & url) in the original dataset to link them to the series
of exchanges; question of capturing all messages containing a url pattern and
extracting it over time

url_pattern <- "http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-
fA-F][0-9a-fA-F]))+"
dataset$ContentURL <- str_extract(dataset$text, url_pattern)
dataset$url<-suffix_extract(domain(dataset$ContentURL))$host

#Producing a graph with the ggplot2 package
            dataset %>%
                mutate(day = as.Date.factor(time)) %>%
                count(url)%>%
                filter(!is.na(url))%>%
                top_n(n = 20, n) %>%
                ggplot(aes(fill=url,x = reorder( url, n), y = n)) +
                geom_bar(stat = "identity",color="black",show.legend = F) +
            scale_fill_manual(values=c("black","#FF5D00","#C8102E",
                                       "#A2A569","#38170B","#BF1B0B",
                                       "#8B5B29", "#B9975B", "#EED484",
                                       "#6E6259", "#707372", "#ACA39A",
                                       "#F1BE48", "#524727", "#76881D",
                                       "#9B945F", "#CAC7A7","#3E4827",
                                       "#003D4C", "#006BA6", "#7A99AC",
                                       "#7C2529", "#9A3324", "#BE531C",
                                       "#999999", "#E69F00", "#56B4E9",
                                       "#FFC465", "#66ADE5", "#252A52"))+
                ylab("Frequency") + xlab("") +
                coord_flip() +
                ggtitle("")+
                labs(caption = "@Lucainson RAYMOND")+
                theme(
                    axis.title.x = element_text(family="Cambria",size=9),
                    axis.title.y = element_text(family="Cambria",size=9))+
            theme_classic()
```



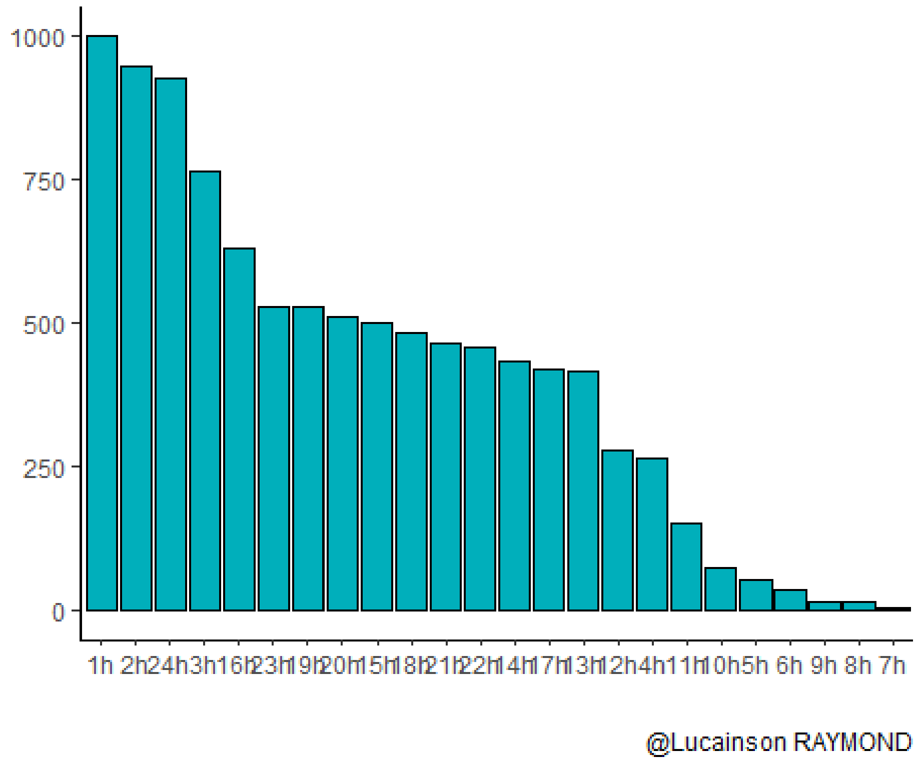
4. Behavioral Analysis

4.1 Members' activity on the group according to the time of day

Members have historically sent more messages during 3 peak hours of the day (1h, 2h, & 24h). And between 7h and 8h, members are obviously not active. At least, a message sent during this time interval will hardly generate commitments (conversations). Which makes sense, because during these hours people are usually on road for work.

```
d<-dataset %>%
   mutate(hour = hour(time)) %>%
   group_by(hour)%>%
   count(hour)

          d$hour[d$hour=="0"]<-"24h"
          d$hour[d$hour=="1"]<-"1h"
          d$hour[d$hour=="2"]<-"2h"
          d$hour[d$hour=="3"]<-"3h"
          d$hour[d$hour=="4"]<-"4h"
          d$hour[d$hour=="5"]<-"5h"
          d$hour[d$hour=="6"]<-"6h"
          d$hour[d$hour=="7"]<-"7h"
          d$hour[d$hour=="8"]<-"8h"
          d$hour[d$hour=="9"]<-"9h"
          d$hour[d$hour=="10"]<-"10h"
          d$hour[d$hour=="11"]<-"11h"
```

```
        d$hour[d$hour=="12"]<-"12h"
        d$hour[d$hour=="13"]<-"13h"
        d$hour[d$hour=="14"]<-"14h"
        d$hour[d$hour=="15"]<-"15h"
        d$hour[d$hour=="16"]<-"16h"
        d$hour[d$hour=="17"]<-"17h"
        d$hour[d$hour=="18"]<-"18h"
        d$hour[d$hour=="19"]<-"19h"
        d$hour[d$hour=="20"]<-"20h"
        d$hour[d$hour=="21"]<-"21h"
        d$hour[d$hour=="22"]<-"22h"
        d$hour[d$hour=="23"]<-"23h"

         d%>%
             ggplot(aes(x = reorder(hour, -n),y=n))+
             geom_bar(stat = "identity", color="black",
                      fill="#00AFBB") +
           labs(caption = "@Lucainson RAYMOND")+
             ylab("") +
             xlab("") +
             theme(plot.title = element_text((hjust=0.5)))+
         theme_classic()
```

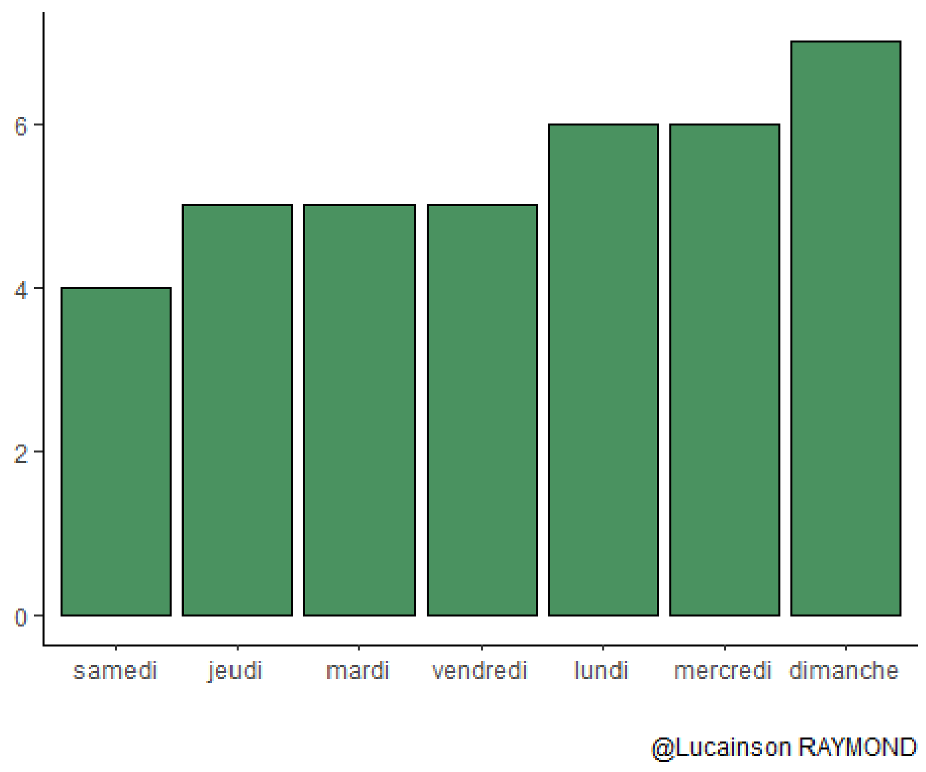


4.2 Members' activity on the group according to the day of the week

We find that members of the chat group are generally more active in the group during Sunday (Dimanche[1], in French). It's normal because Sunday is generally free of work. It's weekend. And note that their activities on the group drop in Saturday, notwithstanding that it's weekend. Now let's take sequential order: we see, between Sunday and Saturday, the flows of messages are rather oscillatory. There isn't any monotony order.

```r
#Distribution of messages by day of the week (median number)
           dataset %>%
               mutate(x = as.Date(time)) %>%
               select(x)%>%
               group_by(x)%>%
               count()%>%
               mutate(day=weekdays(x))%>%
               mutate(dayx=weekdays(x))%>%
               ungroup()%>%
               select(-x)%>%
               group_by(day)%>%
               summarise(n=round(median(n),0))%>%
               ggplot(aes(x = reorder(day, n), y = n))+
               geom_bar(stat = "identity", color="black",
                        fill="#4A9260") +
               labs(caption = "@Lucainson RAYMOND")+
               ylab("") +
               xlab("") +
               theme(plot.title = element_text((hjust=0.5)))+
               theme_classic()
```

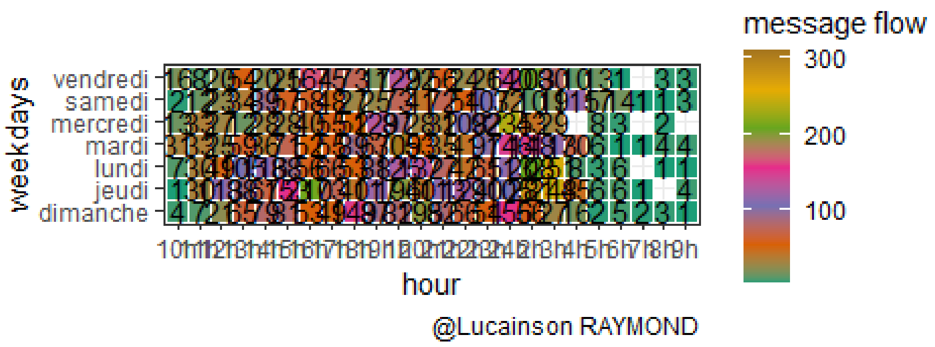
[1] This appears like that on the plot because our computer is in french.



4.3 Bivariate distribution (activity depending on day & time)

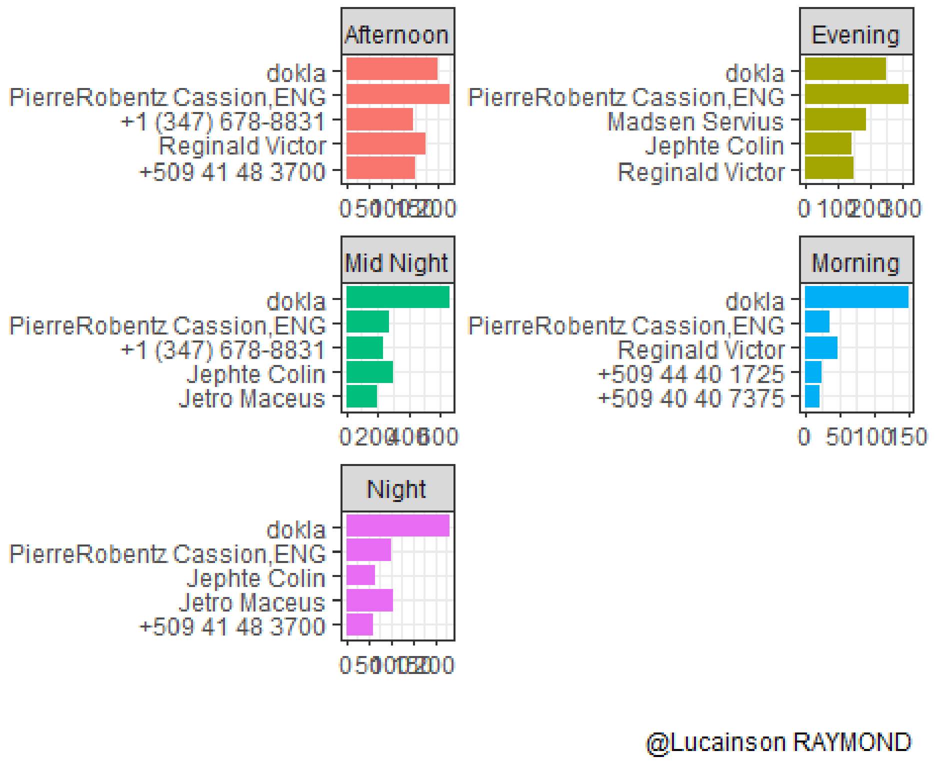
```
 dataset$hour<-hour(dataset$time)
           dataset$weekdays<-weekdays(dataset$time, abbreviate=F)

           df <- dataset %>%
                   group_by(weekdays, hour) %>%
                   summarise(n=n()) %>%
                   rename(`message flow`=n)

#Transforming numbers in hour format
           df$hour[df$hour=="0"]<-"24h"
           df$hour[df$hour=="1"]<-"1h"
           df$hour[df$hour=="2"]<-"2h"
           df$hour[df$hour=="3"]<-"3h"
           df$hour[df$hour=="4"]<-"4h"
           df$hour[df$hour=="5"]<-"5h"
           df$hour[df$hour=="6"]<-"6h"
           df$hour[df$hour=="7"]<-"7h"
           df$hour[df$hour=="8"]<-"8h"
           df$hour[df$hour=="9"]<-"9h"
           df$hour[df$hour=="10"]<-"10h"
           df$hour[df$hour=="11"]<-"11h"
           df$hour[df$hour=="12"]<-"12h"
           df$hour[df$hour=="13"]<-"13h"
```

```
            df$hour[df$hour=="14"]<-"14h"
            df$hour[df$hour=="15"]<-"15h"
            df$hour[df$hour=="16"]<-"16h"
            df$hour[df$hour=="17"]<-"17h"
            df$hour[df$hour=="18"]<-"18h"
            df$hour[df$hour=="19"]<-"19h"
            df$hour[df$hour=="20"]<-"20h"
            df$hour[df$hour=="21"]<-"21h"
            df$hour[df$hour=="22"]<-"22h"
            df$hour[df$hour=="23"]<-"23h"

            # So we display the data using the graphical grammar from the ggp
lot2 package
            ggplot(df, aes(hour,weekdays)) +
                geom_tile(aes(fill = `message flow`),
                          colour = "white") +
                geom_text(aes(label=`message flow`),show.legend = F)+
                scale_fill_distiller(palette = "Dark2",
                                     direction = 1) +
                scale_x_discrete(breaks = df$hour,
                                 labels = df$hour) +
                labs(caption = "@Lucainson RAYMOND")+
                theme(legend.position = "None",
                      panel.grid = element_blank()) +
                coord_equal()+
                theme_bw()
```



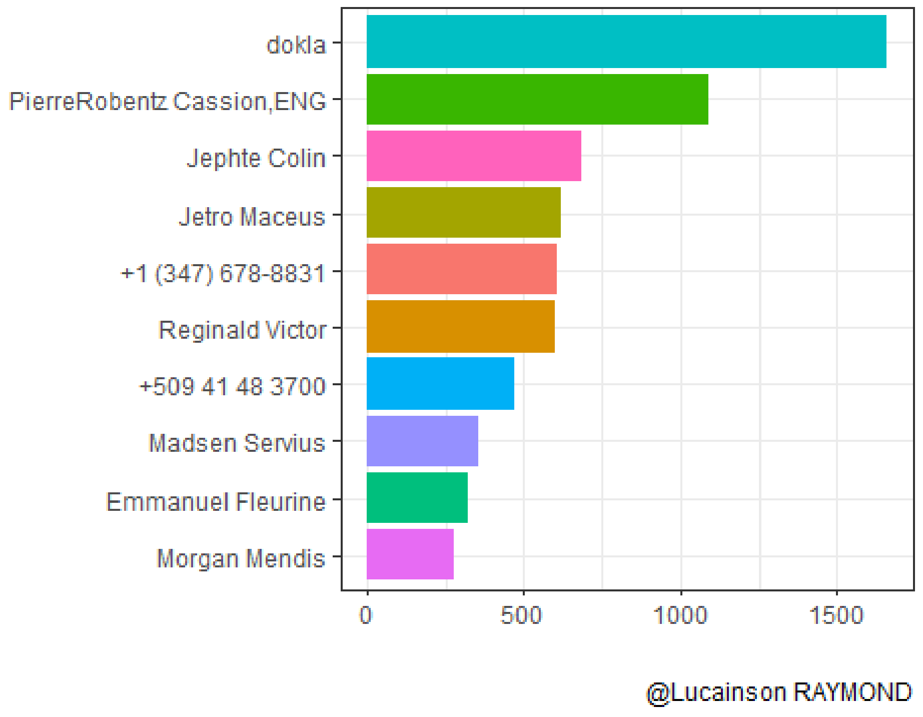
4.4 Top 5 most active according to the day's moment

```
        #By a simple transformation of the "time" column of the original data
set,
        #we are going to create the 4 moments, in this case,
        #Morning (6h-12h), Afternoon (12h-16h), Evening (Night) & Mid-night (
0h-6h)
        #Next, we will highlight which of the Users
        #are most active during said moments

            times=dataset%>%mutate(time=hour(time))
            times=times %>% mutate(Moment = case_when(
                time > 6 & time <= 12~ 'Morning',
                time > 12 & time <= 16 ~ 'Afternoon',
                time > 16 & time <= 21 ~ 'Evening',
                time > 21 & time <= 24 ~ 'Night',
                time > 0 & time <= 6~ 'Mid Night'))

            #Next, we run the algorithm by combining the functions in
            #several packages of R, including dplyr [count (), filter (), ung
roup (), top_n ()],
            # then the layers of ggplot2 [ggplot (), geom_bar (), facet_wrap
() ...]

            times%>%count(author,Moment)%>%

                filter(!is.na(author))%>%
                group_by(Moment)%>%filter(!is.na(Moment))%>%
                top_n(n = 5, n) %>%
                ggplot(aes(x=reorder(author,n),y=n,fill=Moment))+
                geom_bar(stat="identity",show.legend = F)+
                facet_wrap(~Moment,ncol=2,scales="free")+
                coord_flip()+
                labs(x="",y="",caption = "@Lucainson RAYMOND")+
                theme(plot.title = element_text((hjust=0)))+
                theme_bw()
```



4.5 Top most active users of the chat group

```
                getpalette=colorRampPalette(brewer.pal(8,'Dark2'))

                dataset%>%
                    mutate(day = as.Date.factor(time)) %>%
                    count(author)%>%
                    filter(!is.na(author))%>%
                    top_n(n = 10, n) %>%
                    ggplot(aes(x = reorder(author, n), y = n,fill=getpalette(
10))) +
                    geom_bar(stat = "identity",show.legend = F) +
                    ylab("") +
                    xlab("") +
                    coord_flip() +
                    ggtitle("")+
                    labs(caption = "@Lucainson RAYMOND")+
                    theme(plot.title = element_text((hjust=0)))+
                    theme_bw()
```

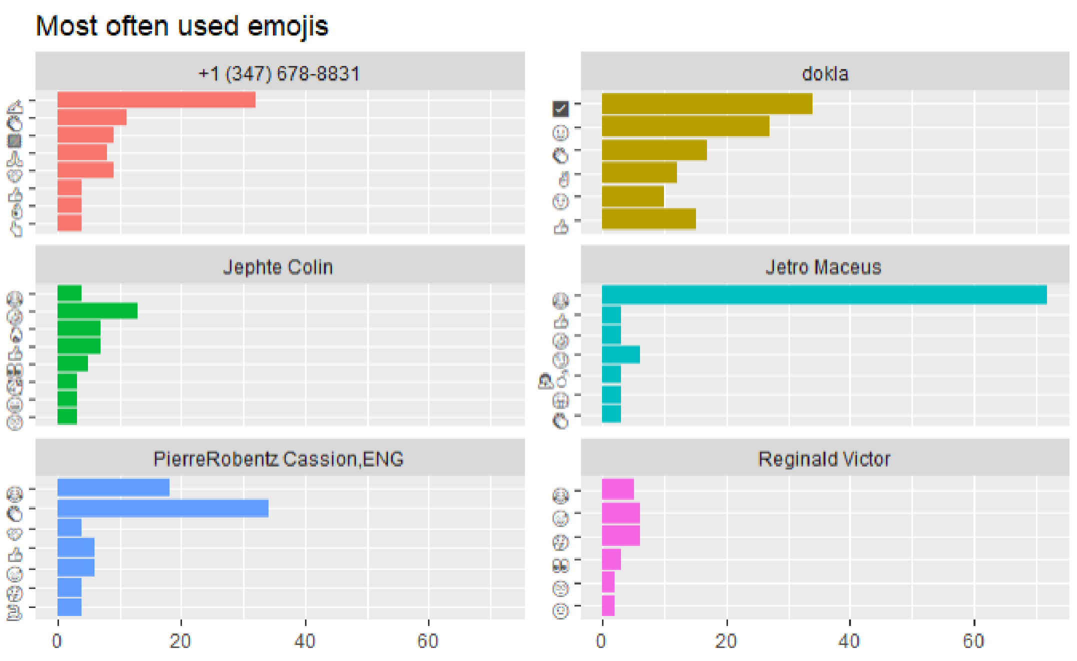


5. Psycho-linguistic analysis

5.1 Top emojis used by top active users

Social networks offer us the opportunity to economize on language with emoticon technology. The latter, as we say, make it possible to convey or simulate a thought in the most expressive way possible without using words. We remember the old adage: “a picture is worth a thousand words”. In addition, emoticons can also be seen as a kind of barometry of the emotionality of a group, of a virtual space (social network, precisely), in many ways. From the count of emojis (by category), we can already get an idea of the general mood of the space. In short! In our case, we are going to focus on the leading animators (alpha animators). We are therefore going to prioritize the categories of emoticons they use in the group throughout the period submitted to our analysis. We assume that they have a significant propensity to use emoticons.

```
#The top 6 active authors
auth<- dataset %>%
  mutate(day = as.Date.factor(time)) %>%
  count(author)%>%
  filter(!is.na(author))%>%
  arrange(-n)%>%
  top_n(n = 6, n)



dataset %>%
  filter(author %in% auth$author)%>%
  unnest(emoji) %>%
  count(author, emoji, sort = TRUE) %>%
  group_by(author) %>%
  top_n(n = 6, n) %>%
  ggplot(aes(x = reorder(emoji, n), y = n, fill = author)) +
  geom_col(show.legend = FALSE) +
  ylab("") +
  xlab("") +
  coord_flip() +
  facet_wrap(~author, ncol = 2, scales = "free_y")  +
  ggtitle("Most often used emojis")
```



5.2 Favorite words of the most active developpers

We are going to highlight all the words most used by each of the most active developers throughout the period under study.

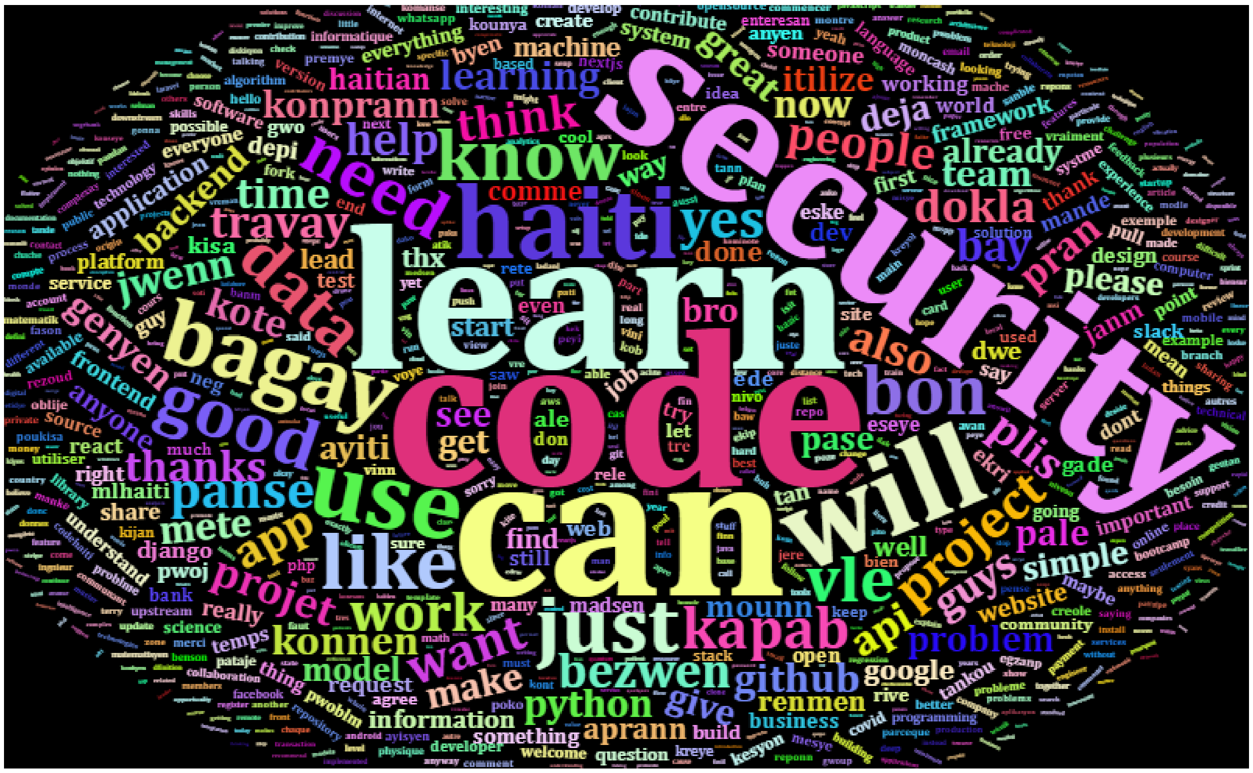
```
#So now, we are going the transform our text in a corpus
#Let's start by cleaning it, by drop some useless words (stopwords) in french
, english and haitian creole (our native language)
to_remove1 <- c(stopwords(kind = "fr"), stopwords(kind="en"),
                "media",
                "omitted","image omitted",
                "ref","manquant","vidéo","remplacé","amp","groupe","group","t
rès",
                "dass","lap","appuyez","intégré","téléphone","numéro","bout",
"en",
                "schon","me","dak","met","fok","aprs","anne","chiffrés","chif
fré","changé","ajouter",
                "mal", "janvier","marchs", "age","send","pages","participants
","Participants",
                "android.s.wt","message","document","image","nn","Oserlecrire
", "messages",
                "sticker","yo", "de","pa","la","se","ki","c","gen","mw","nou"
,"add","envoyer",
                "ou","li","pou","w","nan","yon","nou","k","a","men","f","ke",
"retiré",
                "ak","an","sou","tout","menm","si","e","https","ap","tou","pa
ske","settings","dirk","omis",
                "epi","epwi","lan","mpa","non","mwen","ka","we","fè","tt","fe
","stp","gon","gen","moun","ion",
                "konn","jan","s'on","anpil","oui","wap","épi","plus","pi","c'
est","ion","yon","youn","sa",
                "lot","lè","oubyen","tap","oswa","ds","too","ave","etre","ui"
,"mte","pouw","t","absente","sent",
                "tj","u","jus","jis","ete","w'ap","oserlecrire","g'on","lol",
'poun',"infos","sak","fè","supprim",
                "video","deleted","add","changed","pdf","page","phone","numbe
r","take","new","waiting",
                "que","qui","pap","pat","nan","toujours","it","i","of",'you',
"very","and","donk","san","son",
                "chak","ns","etc","fé","saaa","laa","laaa","mgen","nap","htg"
,"usd","2350","2019",
                "kap",'al',"this","the","supprim","supprimé","for","l","while
","may","peut","peuvent","yap","pour","toujou",
                "mar","to","ti","di","wi","ye","2","have","are","dil","masse"
,"fel","map","ok","fon","anh","svp",                    "dim","2500","my","fo
rms.gle","that","siw","d'i", "meme","oserlecrire","2020","25","2","5093397602
7","o",
                "greg","in","sonw","sak","deux","deuxime","deuxième","faire",
'fait','fait faire',"wimax",
                "wifi", "Marchs","allow","allows","Clavens","clavens","formul
aire","formulaires",
                "group","Wilbens","wilbens","association","Association","atwo
od","salutations","left","Marchs","Administration","administration",
                "added","untitled","annuel","collects", "p","io","ui","uii","
uiii","pral","509","poum","poun",
                "konsa","bn","GIF","gif","audio","Audio","bout","en","chiffre
ment","ann","hein","via","one","two","using","link","invite","joined",
                "mage","Omitted","video","Video","dirk","2400")

#Let's clean up the text again, removing other non-informative "regex" (url,
numbers, useless white spaces, punctuation ...)
whatsapp <- data.frame(text=dataset$text)
whatsapp$text <- as.character(whatsapp$text)
whatsapp$text <- gsub('\\p{So}|\\p{Cn}', '', whatsapp$text, perl = TRUE)
whatsapp$text <- gsub('http\\S+\\s*', '', whatsapp$text)
whatsapp$text <- gsub("[[:digit:]]", '', whatsapp$text)
whatsapp$text <- gsub('\\b+RT', '', whatsapp$text)
whatsapp$text <- gsub('#\\S+', '', whatsapp$text)
```

```r
whatsapp$text <- gsub('@\\S+', '', whatsapp$text)
whatsapp$text <- gsub("+509\\S+", '', whatsapp$text)
whatsapp$text <- gsub('[[:cntrl:]]', '', whatsapp$text)
whatsapp$text <- gsub("\\d", '', whatsapp$text)
whatsapp$text <- gsub("[:graph:]]", '', whatsapp$text)
whatsapp$text <- gsub("<(.+)>", '', whatsapp$text)
whatsapp$text <- gsub('<p{So}|>p{Cn}', '', whatsapp$text, perl = TRUE)
whatsapp$text <- gsub("\uFFFD", "", whatsapp$text, fixed=TRUE)

# Specification of a function to lower case all tokens (1-gram) of the text
tryTolower <- function(x){
  # return NA when there is an error
  y=NA
  # tryCatch error
  try_error= tryCatch(tolower(x),error=function(e) e)
  if (!inherits(try_error, 'error'))
    y=tolower(x)

  return(y)
}

#Cleaning (continued)
custom.stopwords <- c(stopwords('english'),to_remove1)
clean.corpus <- function(corpus) {
  corpus <- tm_map(corpus,
                   content_transformer(tryTolower))
  corpus <- tm_map(corpus,removeWords,
                   custom.stopwords)
  corpus<-tm_map(corpus,removePunctuation)
  corpus <- tm_map(corpus,stripWhitespace)
  corpus <- tm_map(corpus,removeNumbers)
  return(corpus)
}
corpus <- Corpus(VectorSource(whatsapp$text))
corpus <- clean.corpus(corpus)
corpus<-tm_map(corpus, function(x) iconv(x, "latin1", "ASCII", sub=""))



#Conversion now of the corpus into a Document-Term-Matrix object
dtm <- DocumentTermMatrix(corpus)

wh_td <- tidy(dtm)
```

5.2.1 Wordcloud

Understand the dynamics of our wordcloud. This is not to highlight the networking of words, but rather to show the order of frequency of words in the corpus through the relative size of said words in the graph. It's a frequentist approach.

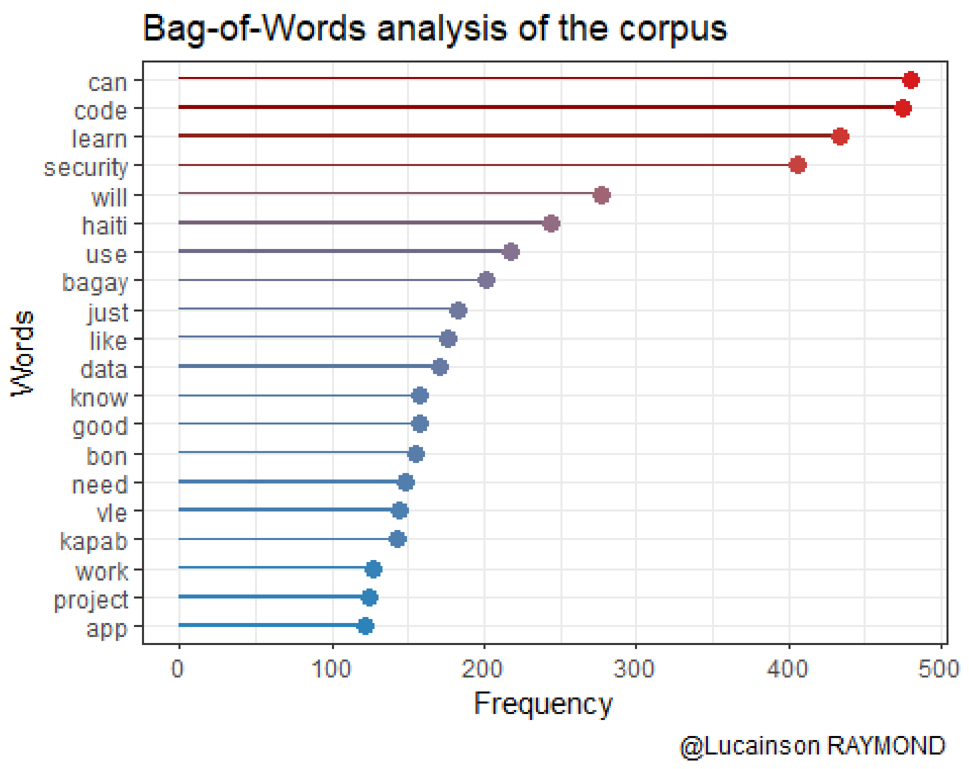
```r
wh_words <- wh_td %>%
  count(term,sort=TRUE) %>%
  filter(n >= 10)

wordcloud2(data=wh_words,size=0.8,fontFamily = "Cambria",
                                color = "random-light",backgroundColor = "black")
```



5.3 NGRAM ANALYSIS

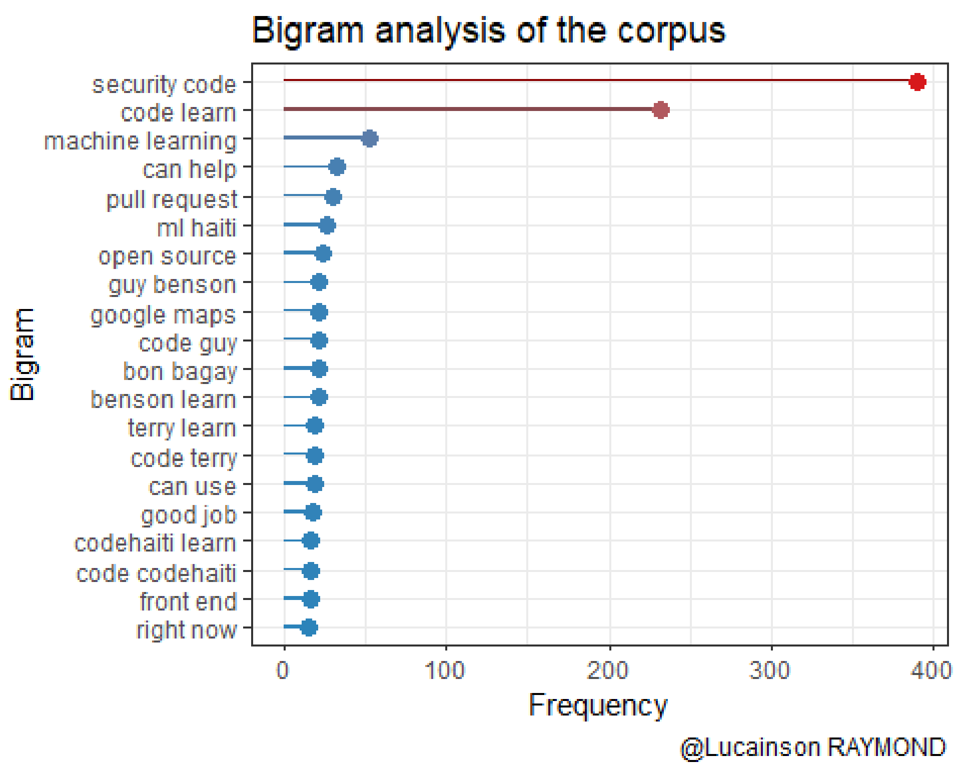
5.3.1 Bag-of-word barplot

```r
data.frame(text = sapply(corpus, as.character), stringsAsFactors = FALSE) %>%
  unnest_tokens(output = ngram, input = text, token="ngrams",n=1) %>%
  count(ngram, sort = TRUE) %>%
  arrange(desc(n))%>%
  filter(!is.na(ngram))%>%
  slice(1:20) %>%
  ggplot(aes(fct_reorder(ngram, n), n)) +
  geom_col(aes(fill=n), show.legend = FALSE, width = .1) +
  geom_point(aes(color=n), show.legend = FALSE, size = 3)+
  coord_flip()+
  theme_minimal() +
  labs(x="Words", y="Frequency",title = "Bag-of-Words anlysis of the corpus",
        caption = "@Lucainson RAYMOND")+
  scale_fill_gradient(low="#2b83ba",high="darkred") +
  scale_color_gradient(low="#2b83ba",high="#d7191c") +
  theme_bw()
```



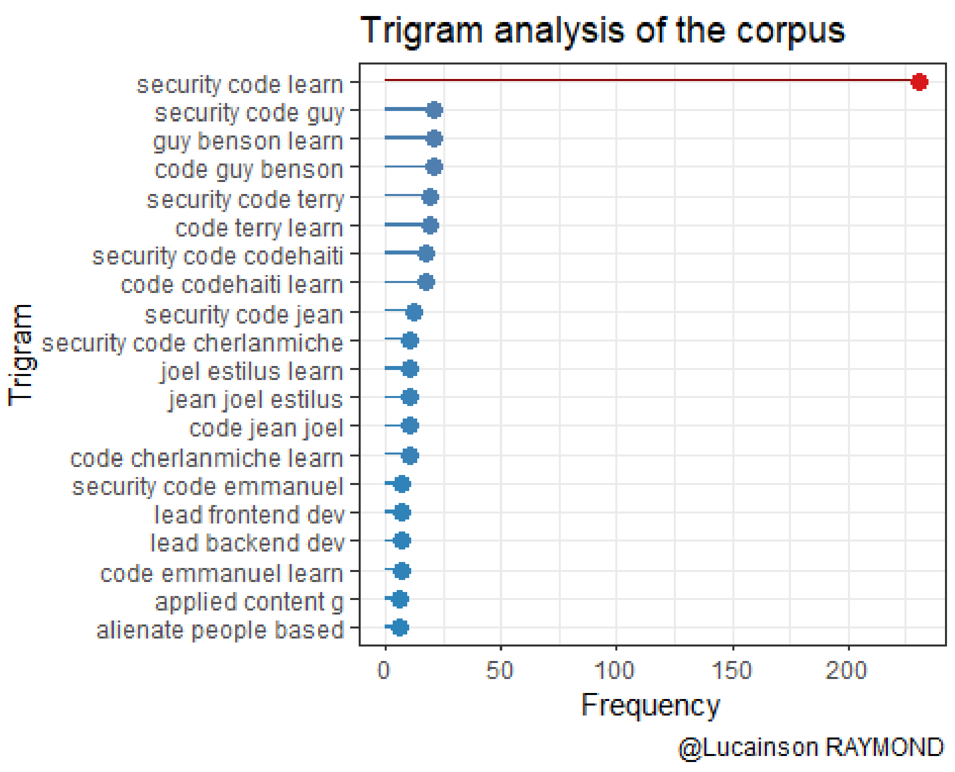
5.3.2 Bi-gram barplot

```
data.frame(text = sapply(corpus, as.character), stringsAsFactors = FALSE) %>%
  unnest_tokens(output = ngram, input = text, token="ngrams",n=2) %>%
  count(ngram, sort = TRUE) %>%
  arrange(desc(n))%>%
  filter(!is.na(ngram))%>%
  slice(1:20) %>%
  ggplot(aes(fct_reorder(ngram, n), n)) +
  geom_col(aes(fill=n), show.legend = FALSE, width = .1) +
  geom_point(aes(color=n), show.legend = FALSE, size = 3)+
  coord_flip()+
  theme_minimal() +
  labs(x="Bigram", y="Frequency",title = "Bigram anlysis of the corpus",
       caption = "@Lucainson RAYMOND")+
  scale_fill_gradient(low="#2b83ba",high="darkred") +
  scale_color_gradient(low="#2b83ba",high="#d7191c") +
  theme_bw()
```



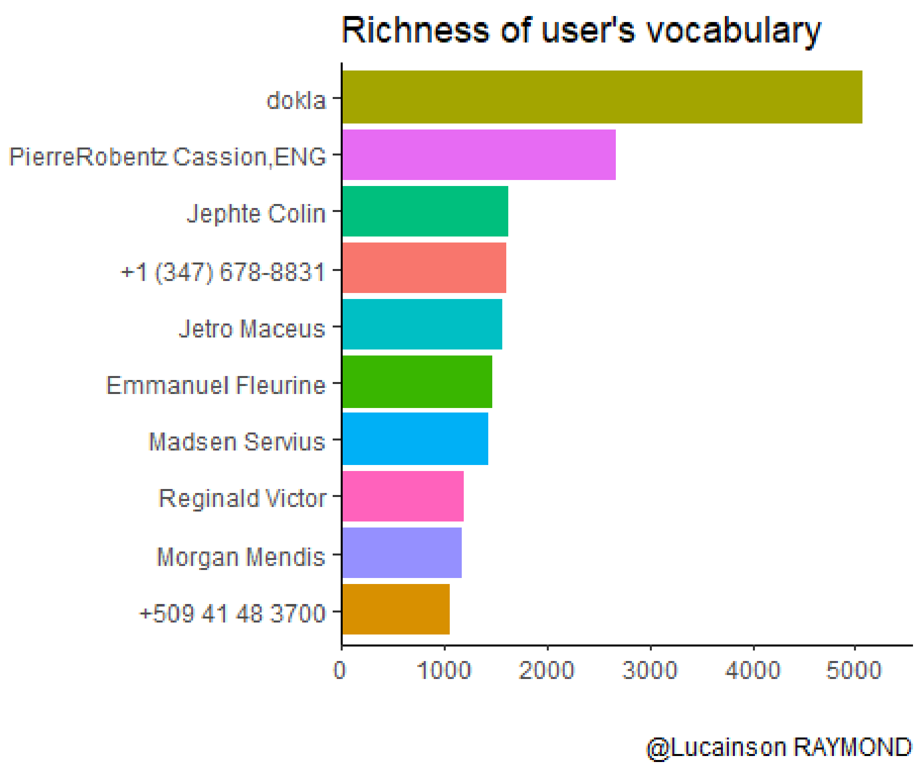
5.3.3 Tri-gram barplot

```
data.frame(text = sapply(corpus, as.character), stringsAsFactors = FALSE) %>%
  unnest_tokens(output = ngram, input = text, token="ngrams",n=3) %>%
  count(ngram, sort = TRUE) %>%
  arrange(desc(n))%>%
  filter(!is.na(ngram))%>%
  slice(1:20) %>%
  ggplot(aes(fct_reorder(ngram, n), n)) +
  geom_col(aes(fill=n), show.legend = FALSE, width = .1) +
  geom_point(aes(color=n), show.legend = FALSE, size = 3)+
  coord_flip()+
  theme_minimal() +
  labs(x="Trigram", y="Frequency",title = "Trigram anlysis of the corpus",
       caption = "@Lucainson RAYMOND")+
  scale_fill_gradient(low="#2b83ba",high="darkred") +
  scale_color_gradient(low="#2b83ba",high="#d7191c") +
  theme_bw()
```



5.4 Lexical diversity of top users

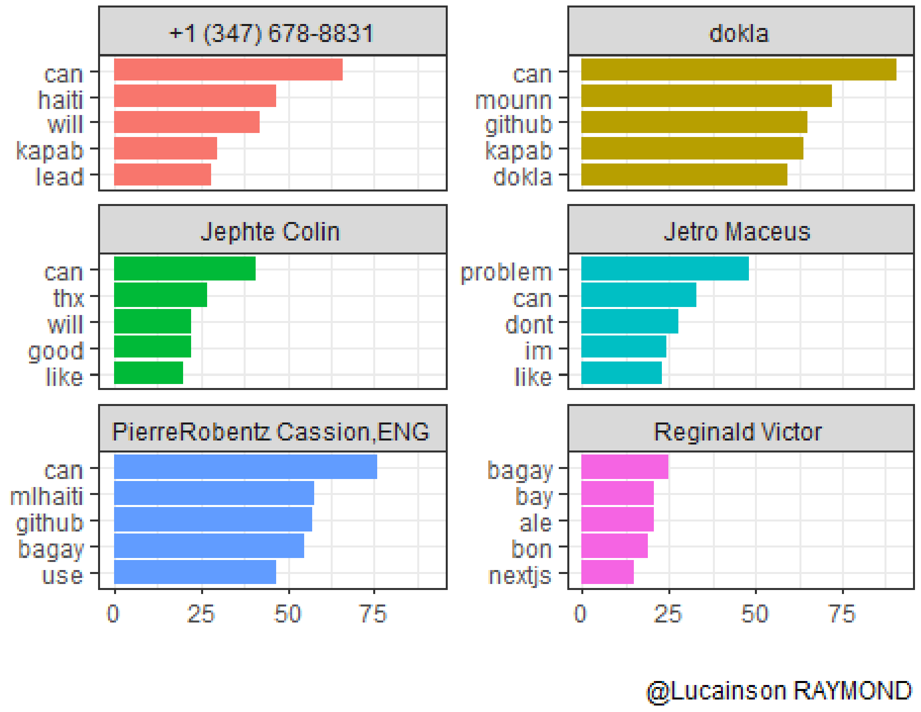
```
dataset %>%
  unnest_tokens(input = text,
                output = word) %>%
  filter(!word %in% to_remove1) %>%
  group_by(author) %>%
  filter(!is.na(author))%>%
  summarise(lex_diversity = n_distinct(word)) %>%
  arrange(desc(lex_diversity)) %>%
  top_n(n = 10, lex_diversity)%>%
  ggplot(aes(x = 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("") +
  xlab("") +
  labs(title="Richness of user's vocabulary",caption="@Lucainson RAYMOND") +
  theme(plot.title = element_text((hjust=0.5)))+
  coord_flip()+
  theme_classic()
```



5.5 Predilection words of top users

5.5.1 Unweighted frequency

```
auth<- dataset %>%
  mutate(day = as.Date.factor(time)) %>%
  count(author)%>%
  filter(!is.na(author))%>%
  arrange(-n)%>%
  top_n(n = 6, n)

dataset %>%
  unnest_tokens(input = text,
                output = word) %>%
  filter(!word %in% to_remove1) %>%
  mutate(word = gsub(".com", "", word)) %>%
  mutate(word = gsub("^gag", "9gag", word)) %>%
  mutate(word = gsub("[[:digit:]]", "", word))%>%
  filter(!word %in% "")%>%
  count(author, word, sort = TRUE) %>%
  group_by(author) %>%
  top_n(n = 5, n) %>%
  filter(!is.na(author))%>%
  filter(author %in% auth$author)%>%
  ggplot(aes(x = reorder_within(word, n, author), y = n, fill = author)) +
  geom_col(show.legend = FALSE) +
  ylab("") +
  xlab("") +
  labs(caption = "@Lucainson RAYMOND")+
  coord_flip() +
  facet_wrap(~author, ncol = 2, scales = "free_y") +
  scale_x_reordered() +
  ggtitle("")+
  theme_bw()
```

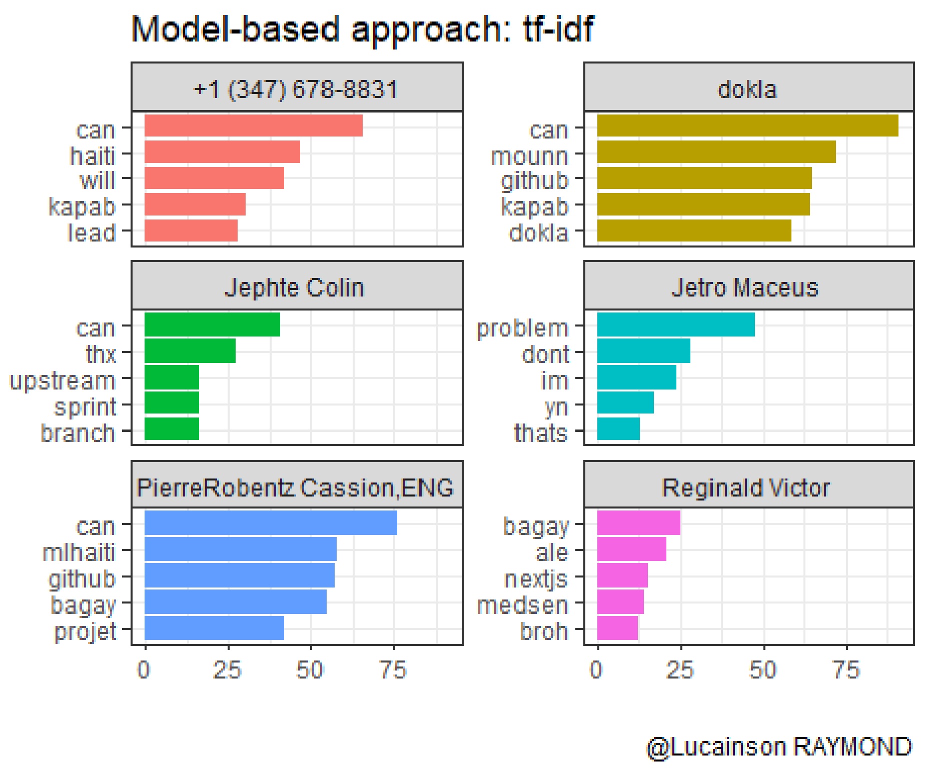


5.5.2 Model-based approach: tf-idf[2]

```
auth<- dataset %>%
  mutate(day = as.Date.factor(time)) %>%
  count(author)%>%
  filter(!is.na(author))%>%
  arrange(-n)%>%
  top_n(n = 6, n)


dataset %>%
  unnest_tokens(input = text,
                output = word) %>%
```

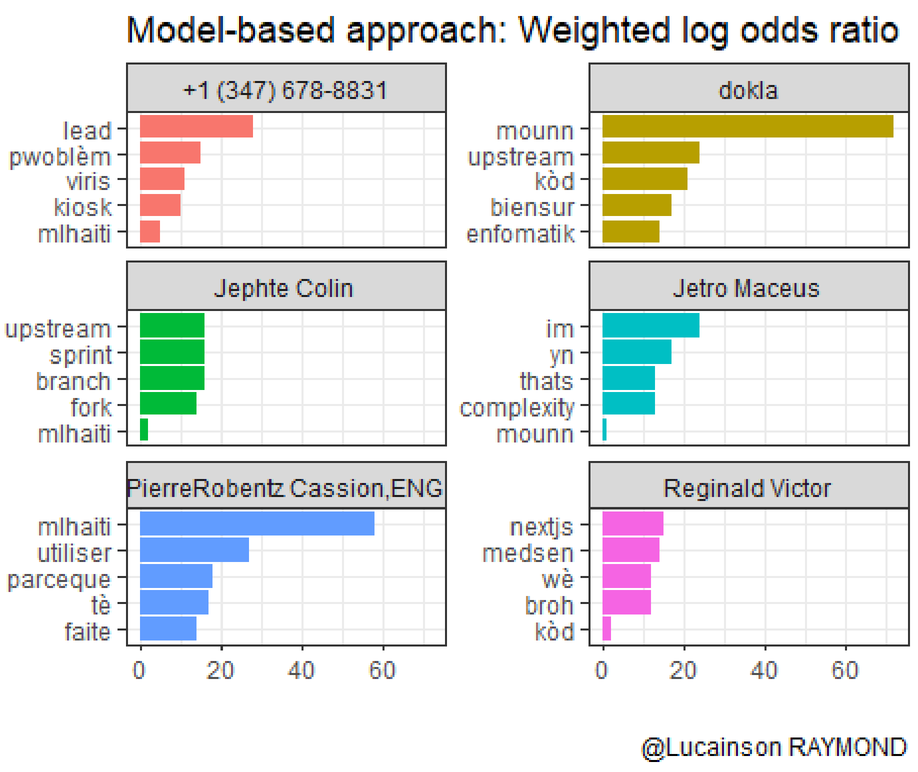
[2] Term Frequency – Inverse Document Matrix

```
  select(word, author) %>%
  filter(!word %in% to_remove1) %>%
  mutate(word = gsub(".com", "", word)) %>%
  mutate(word = gsub("^gag", "9gag", word)) %>%
  mutate(word = gsub("[[:digit:]]", "", word))%>%
  filter(!word %in% "")%>%
  count(author, word, sort = TRUE) %>%
  filter(!is.na(author))%>%
  bind_tf_idf(term = word, document = author, n = n) %>%
  group_by(author) %>%
  filter(author %in% auth$author) %>%
  top_n(n = 5, tf_idf) %>%
  ggplot(aes(x = reorder_within(word, n, author), y = n, fill = author)) +
  geom_col(show.legend = FALSE) +
  ylab("") +
  xlab("") +
  labs(caption = "@Lucainson RAYMOND")+
  coord_flip() +
  facet_wrap(~author, ncol = 2, scales = "free_y") +
  scale_x_reordered() +
  ggtitle("Model-based approach: tf-idf")+
  theme(plot.title = element_text((hjust=0.5)))+
  theme_bw()
```



5.5.3 Model-based approach: Weighted log odds ratio

```r
auth<- dataset %>%
  mutate(day = as.Date.factor(time)) %>%
  count(author)%>%
  filter(!is.na(author))%>%
  arrange(-n)%>%
  top_n(n = 6, n)


dataset %>%
  unnest_tokens(input = text,
                output = word) %>%
  select(word, author) %>%
  filter(!word %in% to_remove1) %>%
  mutate(word = gsub(".com", "", word)) %>%
  mutate(word = gsub("^gag", "9gag", word)) %>%
  mutate(word = gsub("[[:digit:]]", "", word))%>%
  filter(!word %in% "")%>%
  count(author, word, sort = TRUE) %>%
  filter(!is.na(author))%>%
  bind_log_odds(feature = word, set = author, n = n) %>%
  group_by(author) %>%
  filter(author %in% auth$author) %>%
  top_n(n = 5, log_odds_weighted) %>%
  ggplot(aes(x = reorder_within(word, n, author), y = n, fill = author)) +
  geom_col(show.legend = FALSE) +
  ylab("") +
  xlab("") +
  labs(caption = "@Lucainson RAYMOND")+
  coord_flip() +
  facet_wrap(~author, ncol = 2, scales = "free_y") +
  scale_x_reordered() +
  ggtitle("Model-based approach: Weighted log odds ratio")+
  theme(plot.title = element_text((hjust=0.5)))+
  theme_bw()
```



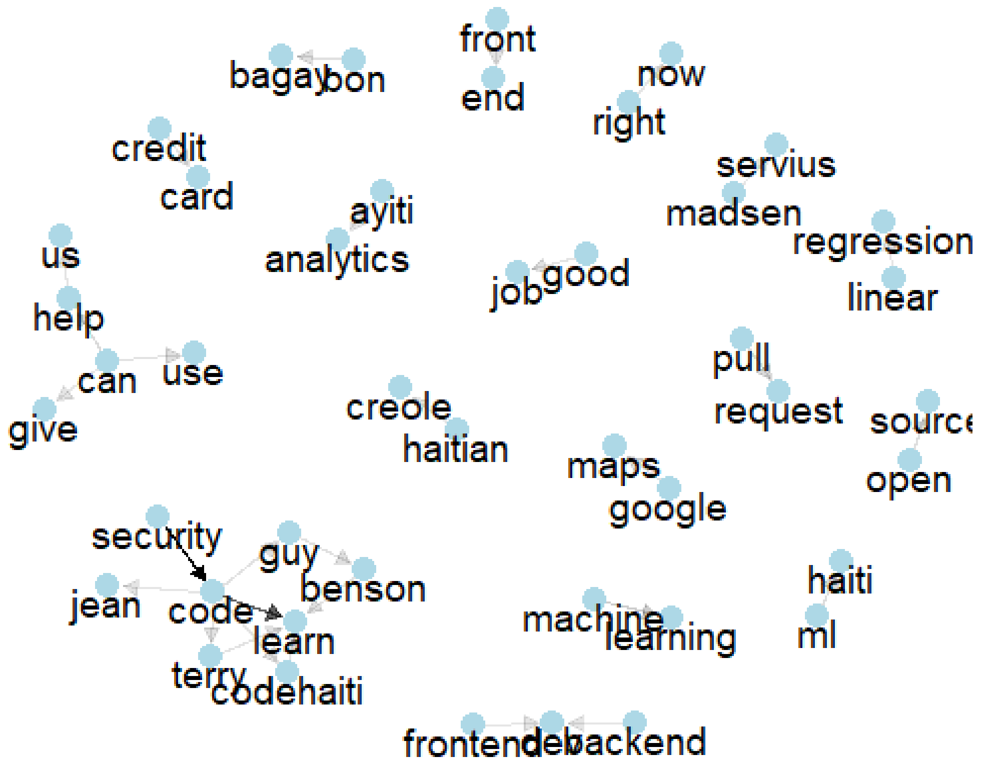
5.6 Word Network Plot

```
#Conversion of the corpus into a Term-Document-Matrix object
tdm<-TermDocumentMatrix(corpus)
print(tdm)

## <<TermDocumentMatrix (terms: 11373, documents: 9929)>>
## Non-/sparse entries: 47059/112875458
## Sparsity           : 100%
## Maximal term length: 52
## Weighting          : term frequency (tf)
```

Statistics from our Term-Document-Matrix show that we have 11,373 terms and 9,929 documents (messages), as well as a very high Sparsity. The latter is therefore indicative of an astronomical number of gaps (or even frequency of zeros) in the matrix of terminological documents.

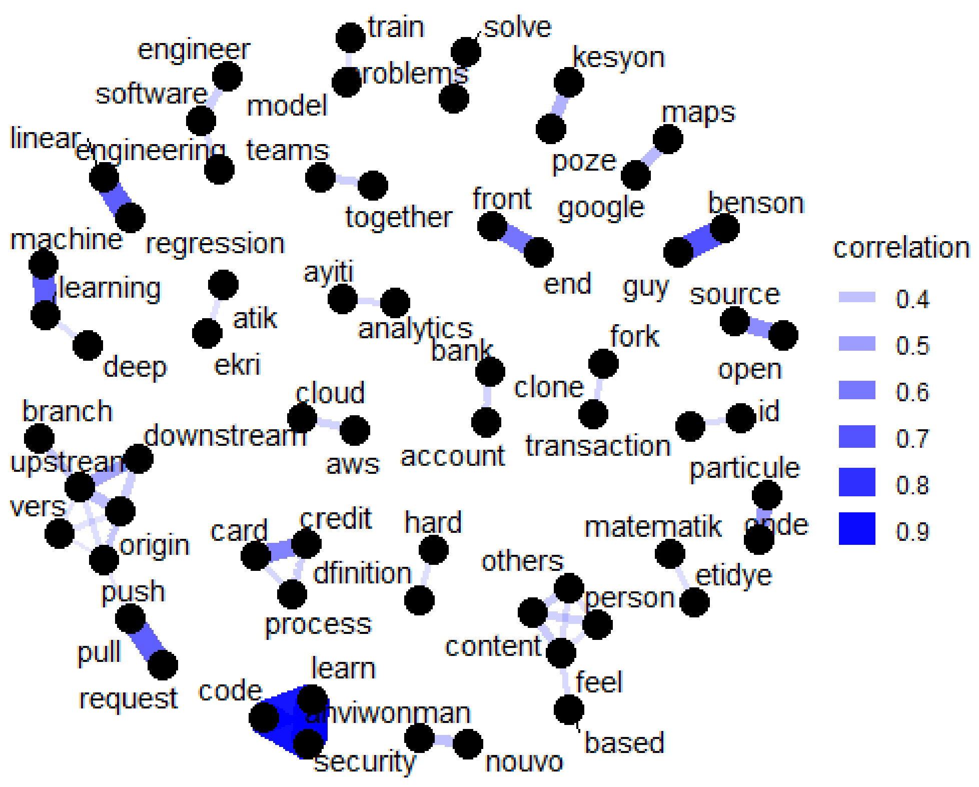
```
getConditionedDataFrame <- function(Corpus) {
  # calculate the frequency of words and sort it by frequency
  word.freq <- sort(rowSums(as.matrix(tdm)), decreasing = T)
  word.freq <- subset(word.freq, word.freq >=1)
  df <- data.frame(term = names(word.freq), freq = word.freq)
  return(df)
}
```

```r
what.df <- getConditionedDataFrame(Corpus)
what.df.t <- what.df[,-2]

corpus1<-corpus

bigr <- data.frame(text = sapply(corpus1, as.character), stringsAsFactors = F
ALSE) %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)

two_word <- bigr %>% count(bigram, sort = TRUE)%>%
  drop_na()

bigrams_separated <- two_word %>%
  separate(bigram, c("word1", "word2"), sep = " ")

bigram_graph <- head(bigrams_separated %>%
                       arrange(desc(n)),30) %>%
  graph_from_data_frame()

#bigram_graph

set.seed(1996)
ggraph(bigram_graph, layout = "stress") +
  geom_edge_link() +
  geom_node_point() +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1)

a <- grid::arrow(type = "closed", length = unit(.08, "inches"))
ggraph(bigram_graph, layout = "fr") +
  geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                 arrow = a, end_cap = circle(.08, 'inches')) +
  geom_node_point(color = "lightblue", size = 4) +
  geom_node_text(aes(label = name), vjust = 1, hjust = 0.5, size=5) +
  theme_void()
```



5.7 Word Correlation Plot

```
corpus2<-corpus
raym <- data.frame(text = sapply(corpus2, as.character), stringsAsFactors = F
ALSE)
raym$what_nbr <- 1:nrow(raym)
what_word <- raym %>% unnest_tokens(word, text)
what_word1 <- raym %>% unnest_tokens(word, text)%>%
  group_by(word)%>%
  count(sort=T)


what_word_correl <- what_word %>%
  group_by(word) %>%
  filter(n() >= 15) %>%
  pairwise_cor(word, what_nbr, sort = TRUE, upper = FALSE)


set.seed(1996)
what_word_correl %>%
  filter(correlation > 0.3) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = correlation, edge_width = correlation), edg
e_colour = "blue") +
  geom_node_point(size = 5) +
  geom_node_text(aes(label = name), repel = TRUE,
                 point.padding = unit(0.2, "lines"))+
  theme_void()
```



5.8 Word Association

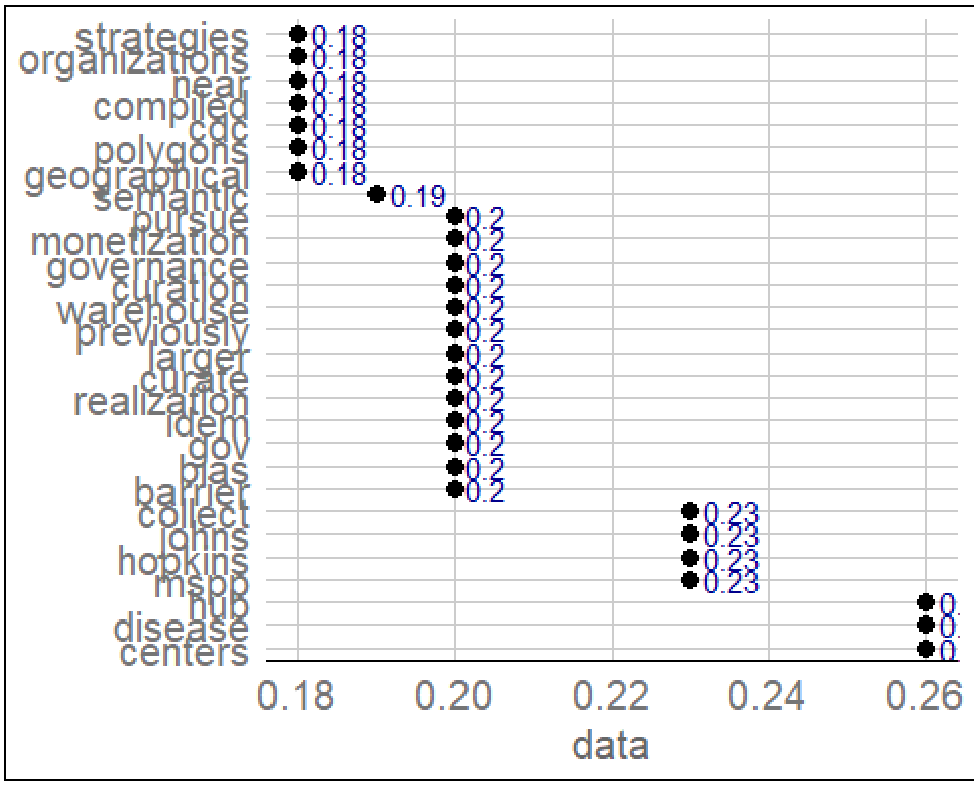
Now we are interested in associating certain targeted words with others throughout the corpus. We will therefore look at, at a given threshold of correlation, which words appear with the words in question (co-occurrence). Thus, we are interested in 3 tokens (words): “data”, “haiti” & “people”.

To begin with, when developers of the chat group mention “data” in their messages, what other words of informative value also appear in at least 18% of these messages? What about the words “Haiti” at a threshold correlation of 12%? “People” at 25%? The three (3) graphs below provide the answers.

Note: We mainly use the findAssocs () function of the “tm” package to do the search on the whole corpus.

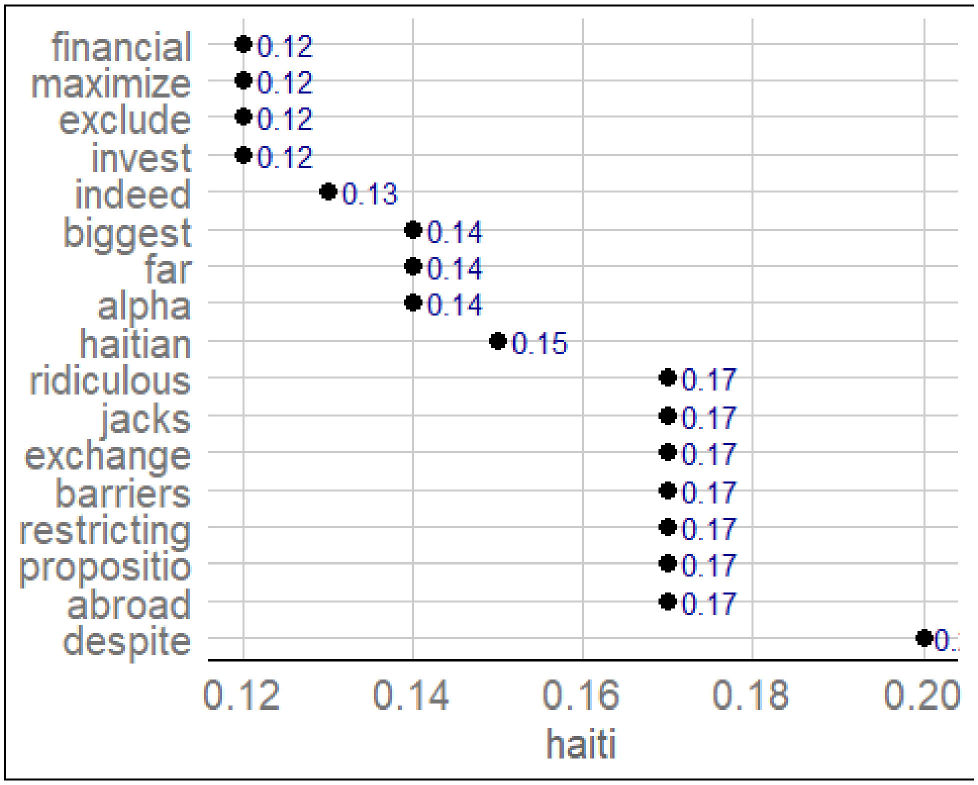
5.8.1 Word association with “data”

```
associations <- findAssocs(tdm, "data", 0.18)
associations <- as.data.frame(associations)
associations$terms <- row.names(associations)
associations$terms <- factor(associations$terms,
                             levels=associations$terms)
ggplot(associations,aes(y=terms)) +
  geom_point(aes(x=data),data=associations,size=3) +
  theme_gdocs() +
  geom_text(aes(x=data,label=data),
            colour="darkblue",hjust=-0.25,size=4) +
  theme(text=element_text(size=15),
        axis.title.y=element_blank())
```



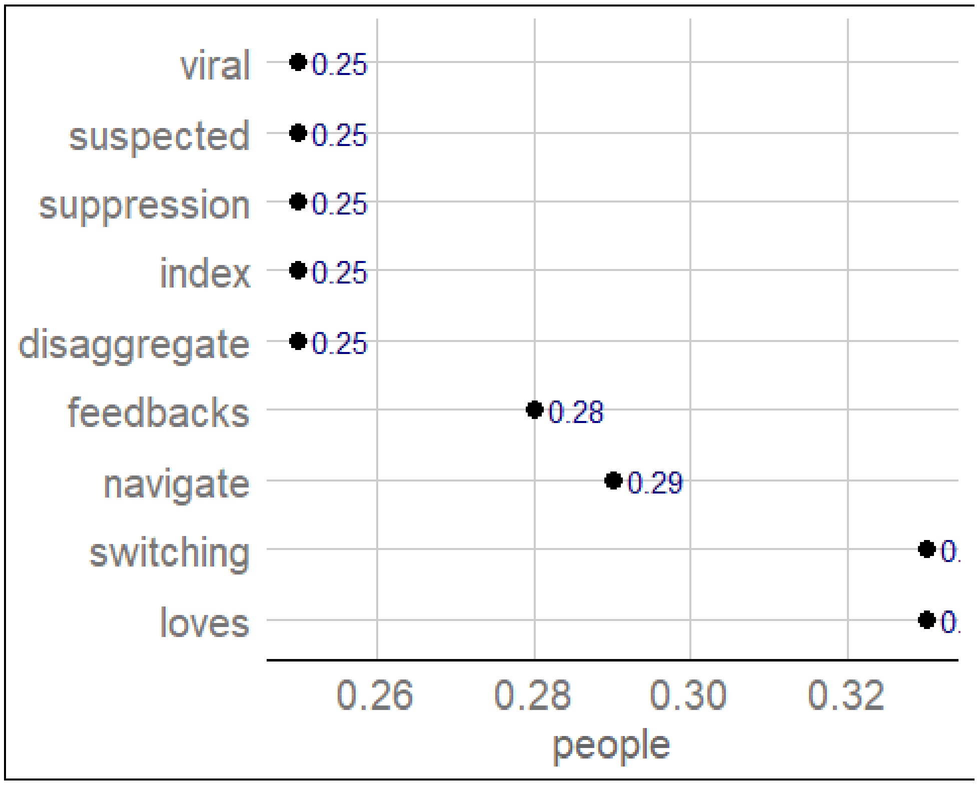
5.8.2 Word association with "Haiti"

```
associations <- findAssocs(tdm, "haiti", 0.12)
associations <- as.data.frame(associations)
associations$terms <- row.names(associations)
associations$terms <- factor(associations$terms,
                             levels=associations$terms)
ggplot(associations,aes(y=terms)) +
  geom_point(aes(x=haiti),haiti=associations,size=3) +
  theme_gdocs() +
  geom_text(aes(x=haiti,label=haiti),
            colour="darkblue",hjust=-0.25,size=4) +
  theme(text=element_text(size=15),
        axis.title.y=element_blank())
```



5.8.3 Word association with "people"

```
associations <- findAssocs(tdm, "people", 0.25)
associations <- as.data.frame(associations)
associations$terms <- row.names(associations)
associations$terms <- factor(associations$terms,
                             levels=associations$terms)
ggplot(associations,aes(y=terms)) +
  geom_point(aes(x=people),people=associations,size=3) +
  theme_gdocs() +
  geom_text(aes(x=people,label=people),
            colour="darkblue",hjust=-0.25,size=4) +
  theme(text=element_text(size=15),
                axis.title.y=element_blank())
```



5.9 Detection of Zipf's law in user exchanges

Zipf's Law is an interesting phenomenon that can be applied universally in many contexts, such as social science, cognitive science, and linguistics. When we consider a variety of data sets, there will generally be an unequal distribution of the objects. Zipf's law asserts that the frequency f of a word w, ie f(w), appears as a nonlinear decreasing function of the rank of the word, ie r(w), in a corpus (Kumar & Paul, 2016). This law is a power law: the frequency is a function of the negative power of the rank.

Given a collection of words, we can estimate the frequency of each unique word, which is nothing more than the number of times the word appears in the collection. If we sort the words in descending order of their frequency of occurrence in the collection and calculate their rank, the product of their frequency and their associated rank reveals a very interesting pattern. It is this pattern that we will try to highlight with the following code snippets.

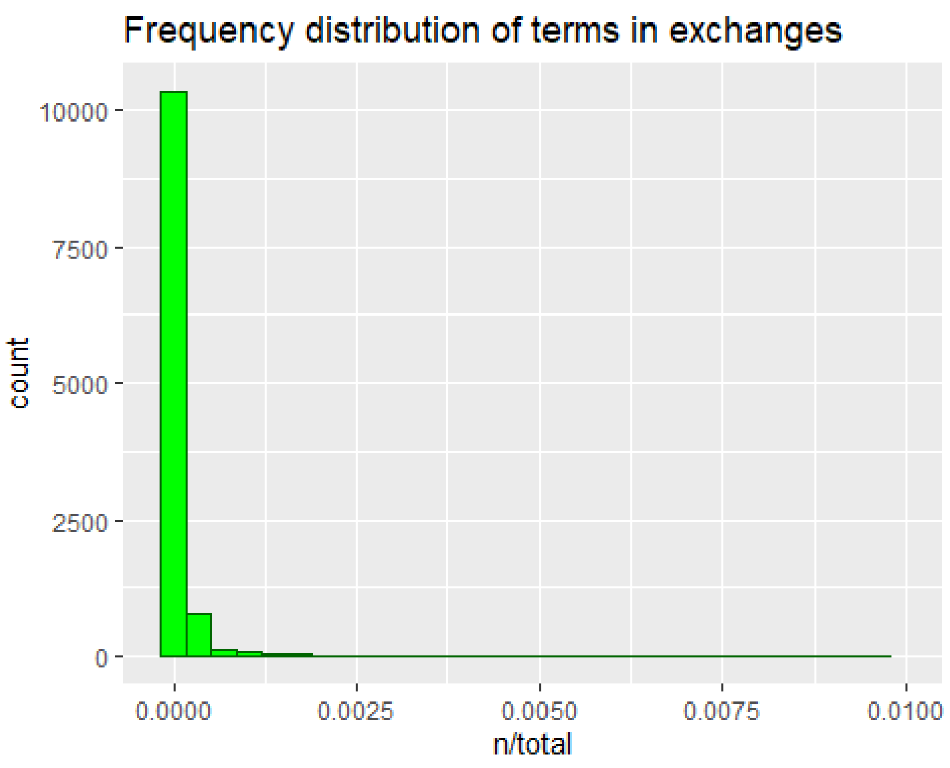
To highlight Zipf's law in users' exchanges, we will first look at the frequency of unique terms (term frequency) in descending order. Then, we'll calculate the frequency of each term in each document and then concatenate them.

```
terms <- wh_td %>%
  count(term,sort=T)
print(terms)

## # A tibble: 11,373 x 2
##    term         n
##    <chr>    <int>
##  1 code       467
##  2 can        438
##  3 learn      431
##  4 security   404
##  5 will       250
##  6 haiti      221
##  7 use        203
##  8 bagay      186
##  9 just       172
## 10 like       165
## # ... with 11,363 more rows
```

```
terms$var <- 1
totalterms <- terms %>%
  group_by(var) %>%
  summarize(total=sum(n))
terms_total <- left_join(terms,totalterms)

print(terms_total)
```

```
## # A tibble: 11,373 x 4
##    term          n   var total
##    <chr>    <int> <dbl> <int>
##  1 code       467     1 47059
##  2 can        438     1 47059
##  3 learn      431     1 47059
##  4 security   404     1 47059
##  5 will       250     1 47059
##  6 haiti      221     1 47059
##  7 use        203     1 47059
##  8 bagay      186     1 47059
##  9 just       172     1 47059
## 10 like       165     1 47059
## # ... with 11,363 more rows
```

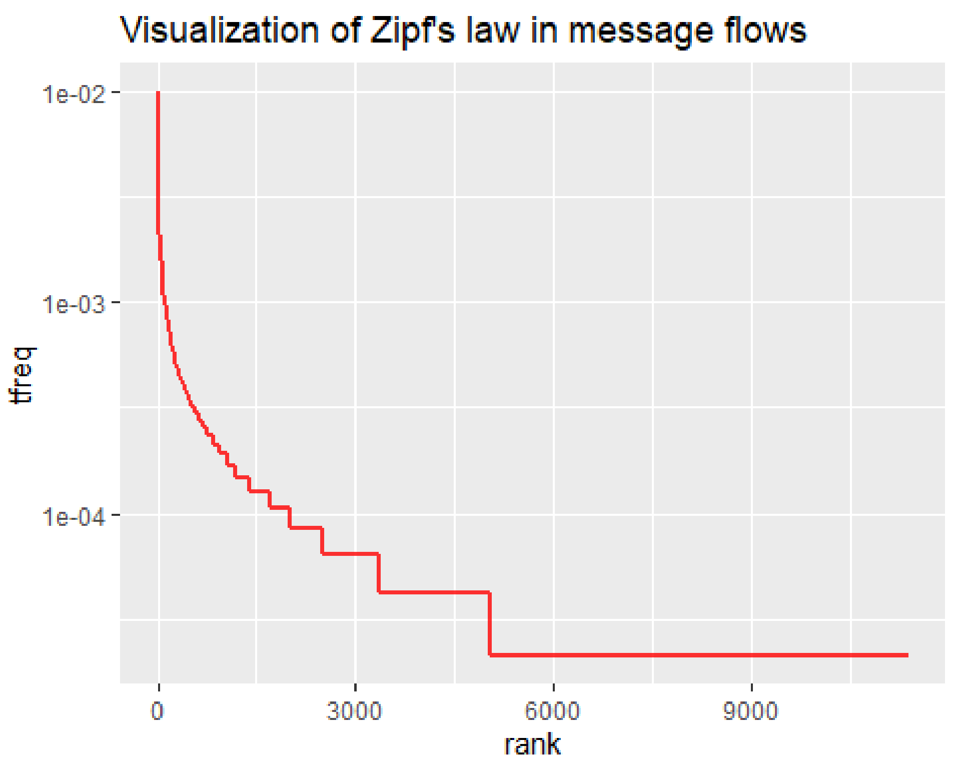
```
ggplot(terms_total,aes(n/total)) +
  geom_histogram(fill="green",color="darkgreen",show.legend=FALSE) +
  xlim(NA,0.010) +
  ggtitle("Frequency distribution of terms in exchanges")
```



```
freq_by_rank <- terms_total %>%
  mutate(rank=row_number()) %>%
  mutate(tfreq=n/total)
print(freq_by_rank)
```

```
## # A tibble: 11,373 x 6
##    term         n   var total  rank   tfreq
##    <chr>    <int> <dbl> <int> <int>   <dbl>
##  1 code       467     1 47059     1 0.00992
##  2 can        438     1 47059     2 0.00931
##  3 learn      431     1 47059     3 0.00916
##  4 security   404     1 47059     4 0.00858
##  5 will       250     1 47059     5 0.00531
##  6 haiti      221     1 47059     6 0.00470
##  7 use        203     1 47059     7 0.00431
##  8 bagay      186     1 47059     8 0.00395
##  9 just       172     1 47059     9 0.00365
## 10 like       165     1 47059    10 0.00351
## # ... with 11,363 more rows
```

```
p <- ggplot(freq_by_rank,aes(x=rank,y=tfreq)) +
  geom_line(size=1,col="red",alpha=.8,show.legend=FALSE) +
  scale_y_log10() +
  ggtitle("Visualization of Zipf's law in message flows")
print(p)
```



5.10 Clustering analysis

5.10.1 Word Embedding

In Natural Language Processing (NLP), *word embedding* is the numerical representation of textual data. Its main objective is to capture as much as possible the context, hierarchical and morphological information concerning the sentences, words or letters in question.

Word embedding can be categorized in two (2) ways:

- Frequency-based embedding
- Prediction-based embedding

The first paradigm, in this case, Frequency-based embedding, uses a word counting mechanism, while the second, a probability distribution mechanism. Frequency-based embedding can be implemented in several ways: using a token count vector, a tf-idf vector or a co-occurrence vector / matrix. Regarding the second paradigm, the two (2) most popular techniques are word2vec and GLoVe. But this is not our concern in this present work, since we will opt for the first technique: Frequency-based embedding. For that, we will use the t-SNE algorithm. To implement the t-SNE[3] (a probabilistic dimensionality reduction algorithm), we will therefore create a vector space with the vectors of tf-idf.

[3] t-Student Stochastic Neighbor Embedding

Thus, we will start by creating with the following code a Document-Term-Matrix object weighted by the tf-idf metric (very important!!!). Next, we will convert said object into a Matrix object to facilitate calculations on vectors of tf-idf.

Note: A Document-Term-Matrix object, implemented in R with the DocumentTermMatrix () function of the tm package, is a Sparse matrix where documents are in rows and terms (words) are in columns. Not to be confused with the Term-Document-Matrix object we worked on above.

```
#We use the TF-IDF metric to section the corpus into clusters
dtm_tfidf <-DocumentTermMatrix(corpus,control = list(weighting = function(x)
weightTfIdf(x, normalize = FALSE)))
str(dtm_tfidf)

## List of 6
##  $ i       : int [1:47059] 1 1 2 2 3 5 5 5 5 6 ...
##  $ j       : int [1:47059] 1 2 3 4 5 6 7 8 9 6 ...
##  $ v       : num [1:47059] 7.82 9.69 10.28 6.89 6.72 ...
##  $ nrow    : int 9929
##  $ ncol    : int 11373
##  $ dimnames:List of 2
##   ..$ Docs : chr [1:9929] "1" "2" "3" "4" ...
##   ..$ Terms: chr [1:11373] "madsen" "servius" "admin" "now" ...
##  - attr(*, "class")= chr [1:2] "DocumentTermMatrix" "simple_triplet_matrix
"
##  - attr(*, "weighting")= chr [1:2] "term frequency - inverse document freq
uency" "tf-idf"

inspect(dtm_tfidf)

## <<DocumentTermMatrix (documents: 9929, terms: 11373)>>
## Non-/sparse entries: 47059/112875458
## Sparsity           : 100%
## Maximal term length: 52
## Weighting          : term frequency - inverse document frequency (tf-idf)
## Sample             :
##       Terms
## Docs      bagay      can     code      data haiti      just     learn secu
rity
##   1569  0.00000 18.01058 0.000000  0.000000     0  0.000000  0.000000
0
##   3356  0.00000  0.00000 0.000000  0.000000     0  0.000000  0.000000
0
##   3767  0.00000  0.00000 0.000000  0.000000     0  0.000000  0.000000
0
##   4409  0.00000  0.00000 0.000000  0.000000     0  0.000000  0.000000
0
##   491  17.21482  0.00000 0.000000  0.000000     0  5.851168  0.000000
0
##   7457  0.00000  0.00000 0.000000  0.000000     0  0.000000  0.000000
0
```

```
##   7792  0.00000 13.50794 4.410154  0.000000     0 11.702336  4.525889
0
##   7877  0.00000  0.00000 0.000000 12.195047     0  0.000000  4.525889
0
##   7881  0.00000  0.00000 0.000000  0.000000     0  0.000000  0.000000
0
##   7883  0.00000  0.00000 0.000000  6.097524     0  0.000000 13.577666
0
##       Terms
## Docs        use will
##   1569  0.00000    0
##   3356  0.00000    0
##   3767  0.00000    0
##   4409  0.00000    0
##   491   0.00000    0
##   7457  0.00000    0
##   7792 16.83629    0
##   7877  0.00000    0
##   7881  0.00000    0
##   7883  0.00000    0
```

Before converting the object to a matrix, we remove a large amount of Sparse from it in order to optimize the computation time and refine our analysis. Because terms with a lot of zero in the corpus really don't add anything.

```
dtm_tfidf2<-removeSparseTerms(dtm_tfidf,sparse=0.994)
print(dtm_tfidf2)

## <<DocumentTermMatrix (documents: 9929, terms: 77)>>
## Non-/sparse entries: 8941/755592
## Sparsity           : 99%
## Maximal term length: 8
## Weighting          : term frequency - inverse document frequency (tf-idf)
```

The document-term-matrix object weighted by the tf-idf metric is then converted into a matrix for computational convenience.

```
#Conversion of dtm object to matrix
m2 <- as.matrix(dtm_tfidf2)
```

5.10.2 t-distributed Stochastic Neighbor Embedding (t-SNE)

The t-SNE was introduced by Laurens Van Der MAATEN & Geoff HINTON. It is therefore a variant of the Stochastic Neighbor Embedding (Hinton & Roweis, 2002) which relies on the Gaussian distribution for the probabilistic modeling of very high dimensional data. As for the t-SNE, it pivots on the t-Student distribution, as its name suggests.

T-SNE (t-distributed Stochastic Neighbor Embedding) is an unsupervised machine learning algorithm. At the same time, it is a non-linear technique for reducing dimensionality and visualizing very large data in reduced dimensions.

The t-SNE algorithm converts the similarity between the data points (in our case, the tokens) into joint probabilities and tries to minimize the Kullback-Leiber divergence between the joint probabilities of the vector space of the vectors of tf-idf.

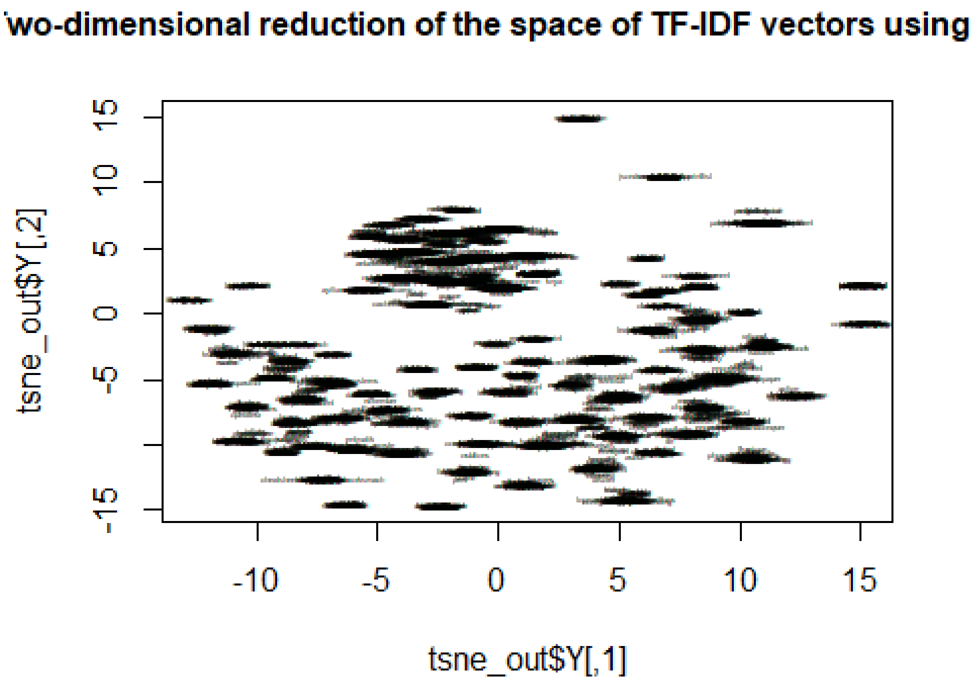
By changing the placement of words in the embedding to minimize the Kullback-Leibler divergence between these two probability distributions: qij and pij, we create a map that focuses on the small-scale structure, due to the asymmetry of the Kullback-Leibler divergence. The t-Student distribution is chosen to avoid the overcrowding problem: in the original high-dimensional space, there are potentially many equidistant word at moderate distance from a particular word, more than what can be taken into account in the low-dimensional representation (Krijthe, 2018). The t-distribution ensures that these words are well distributed in the new representation.

Since we will have dense data after reducing the sparsity of the matrix, we will therefore perform a Principal Component Analysis (PCA) on the data matrix well before running the t-SNE algorithm. The PCA will therefore reduce the dimension of the matrix, keeping only the factorial axes maximizing the inertia (n-dimensional variance). Thus, the t-sne will now be carried out on a matrix of format 9,929 x 50. Because we have 50 principal components (inertial axes) retained (by default) by the algorithm …

Notice that we specify in the function syntax: num_threads = 0. this is a way of telling the algorithm to do parallel calculations using all the available threads of our processor. Thus, the calculations are completed after 51.46 seconds. The perplexity argument (Perplexity = 30) is the number of closest neighbors each term (word) has. From high dimensional space to low dimensional (Plan) space, the position of a word on the map will always be determined with its thirty (30) closest neighbors. We also have max_iter = 500. This means that the model repeats its training five hundred (500) times. And we have a very low error (error = `2.974648`) after the 500th iteration …

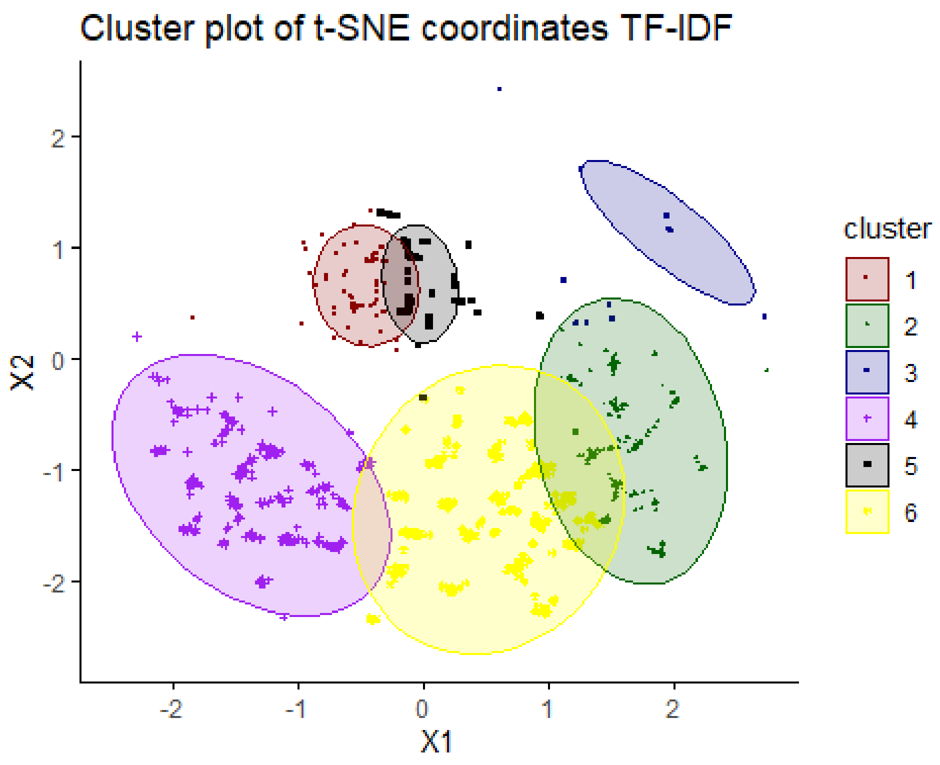
```
#Training of the model
tsne_out <- Rtsne(m2,dims=2,check_duplicates=F,perplexity=30,theta=0.5,pca=T,
verbose=T,max_iter = 500, num_threads = 0)

## Performing PCA
## Read the 9929 x 50 data matrix successfully!
## OpenMP is working. 2 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 13.58 seconds (sparsity = 0.016423)!
## Learning embedding...
## Iteration 50: error is 84.913262 (50 iterations in 6.19 seconds)
## Iteration 100: error is 76.918218 (50 iterations in 7.37 seconds)
## Iteration 150: error is 75.327921 (50 iterations in 4.22 seconds)
## Iteration 200: error is 74.953387 (50 iterations in 3.75 seconds)
## Iteration 250: error is 74.772877 (50 iterations in 3.87 seconds)
## Iteration 300: error is 3.609356 (50 iterations in 4.28 seconds)
## Iteration 350: error is 3.087361 (50 iterations in 4.82 seconds)
## Iteration 400: error is 2.998196 (50 iterations in 5.61 seconds)
## Iteration 450: error is 2.974648 (50 iterations in 5.57 seconds)
## Iteration 500: error is 2.997057 (50 iterations in 5.77 seconds)
## Fitting performed in 51.46 seconds.
```

```
m <- as.matrix(tdm)
plot(tsne_out$Y,t="n",main="Two-dimensional reduction of the space of TF-IDF
vectors using t-SNE",cex.main=1)
text(tsne_out$Y,labels=rownames(m),cex=0.25)
```



The t-SNE separates & groups a set of words into clusters that are sematically different or similar. You can see this clustering from the graph below. We therefore have six (6) clusters of words. It's obviously very efficient. This algorithm generally generates much more precise results than other traditional partitioning algorithms, such as: K-means, K-modes or even standard PCR (applied alone).

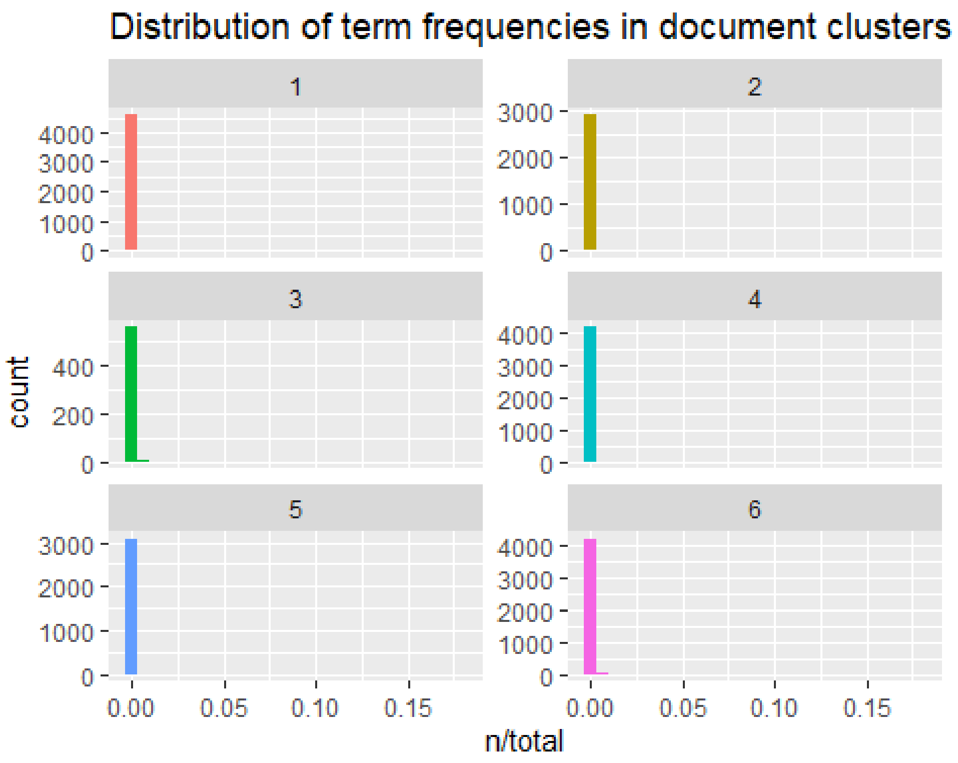
```
#The t-SNE coordinates will then be adjusted to CLARA (clustering for Large A
pplications), which will result from the extraction of 6 clusters
X1 <- tsne_out$Y[,1]
X2 <- tsne_out$Y[,2]
# clara clustering
df <- data.frame(X1,X2)

clara.res <- clara(df,6,samples=40,pamLike=TRUE)
fviz_cluster(clara.res,
             palette = c("darkred","darkgreen","darkblue","purple","black","y
ellow"), # color palette
             ellipse.type = "t", # Ellipse de Concentration
             geom = "point", pointsize = 0.5,title="Cluster plot of t-SNE coo
rdinates TF-IDF",
             ggtheme = theme_classic())
```



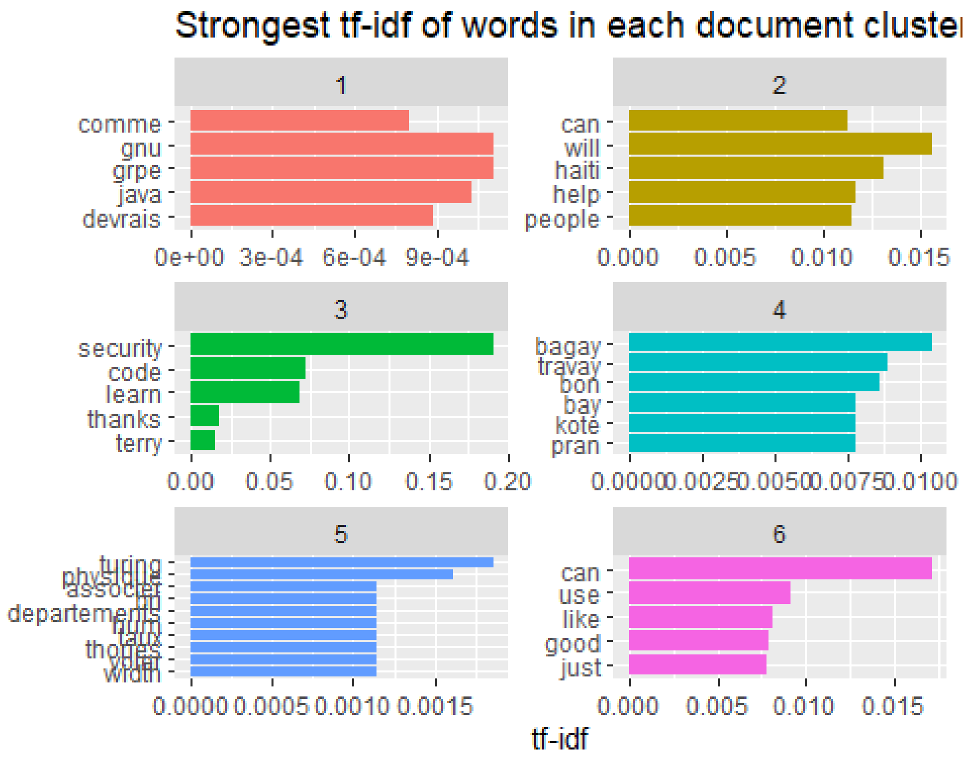
To contextualize the clusters built by the algorithm, we examine which words are important for documents within each of the clusters. We quantify each term within each document by giving weights based on the tf-idf metric. Remember that the goal of the tf-idf is to find the important words by increasing the weight of words that are not widely used in a body of documents and by decreasing the weight of commonly used words. To do this, we first created an orderly format of the messages and cluster members that we gathered earlier.

```
#Word importance with the bind_tf_idf() function
dd <- cbind(df,cluster=clara.res$cluster)
dd$document <- as.numeric(rownames(dd))
dd$document <- as.character(dd$document)
dd$cluster <- as.factor(dd$cluster)
final <- left_join(wh_td,dd)
final2 <- final %>%
  group_by(cluster) %>%
  count(cluster,term,sort=TRUE) %>%
  ungroup()
totals <- final2 %>%
  group_by(cluster) %>%
  summarize(total=sum(n))
cluster_words <- left_join(final2,totals)
print(cluster_words)
```

```
## # A tibble: 19,629 x 4
##    cluster term         n total
##    <fct>   <chr>    <int> <int>
##  1 3       code       410  2290
##  2 3       security   398  2290
##  3 3       learn      392  2290
##  4 6       can        293 11869
##  5 2       will       186  8250
##  6 4       bagay      178 11855
##  7 6       use        157 11869
##  8 2       haiti      156  8250
##  9 4       bon        147 11855
## 10 6       like       140 11869
## # ... with 19,619 more rows

#Distribution of term frequencies in document clusters
ggplot(cluster_words,aes(n/total,fill=cluster)) +
  geom_histogram(show.legend=FALSE) +
  facet_wrap(~ cluster,ncol=2,scales="free_y") +
  ggtitle("Distribution of term frequencies in document clusters")
```



```
cluster_words <- cluster_words %>%
  bind_tf_idf(term,cluster,n)
print(cluster_words)

## # A tibble: 19,629 x 7
##    cluster term         n total     tf   idf  tf_idf
##    <fct>   <chr>     <int> <int>  <dbl> <dbl>   <dbl>
##   1 3       code        410  2290 0.179  0.405 0.0726
##   2 3       security    398  2290 0.174  1.10  0.191
##   3 3       learn       392  2290 0.171  0.405 0.0694
##   4 6       can         293 11869 0.0247 0.693 0.0171
##   5 2       will        186  8250 0.0225 0.693 0.0156
##   6 4       bagay       178 11855 0.0150 0.693 0.0104
##   7 6       use         157 11869 0.0132 0.693 0.00917
##   8 2       haiti       156  8250 0.0189 0.693 0.0131
##   9 4       bon         147 11855 0.0124 0.693 0.00859
##  10 6       like        140 11869 0.0118 0.693 0.00818
## # ... with 19,619 more rows

# The idf & the tf-idf are zero (0) for common words appearing in all cluster
s
cluster_words %>%
  select(-total) %>%
  arrange(desc(tf_idf)) %>%
  mutate(term=factor(term,levels=rev(unique(term)))) %>%
  group_by(cluster) %>%
  top_n(5) %>%
  ungroup %>%
  ggplot(aes(term,tf_idf,fill=cluster)) +
  geom_col(show.legend=FALSE) +
  labs(x=NULL,y="tf-idf") +
  facet_wrap(~ cluster,ncol=2,scales="free") +
  coord_flip() +
  ggtitle("Strongest tf-idf of words in each document cluster")
```



5.11 Topic modeling over time

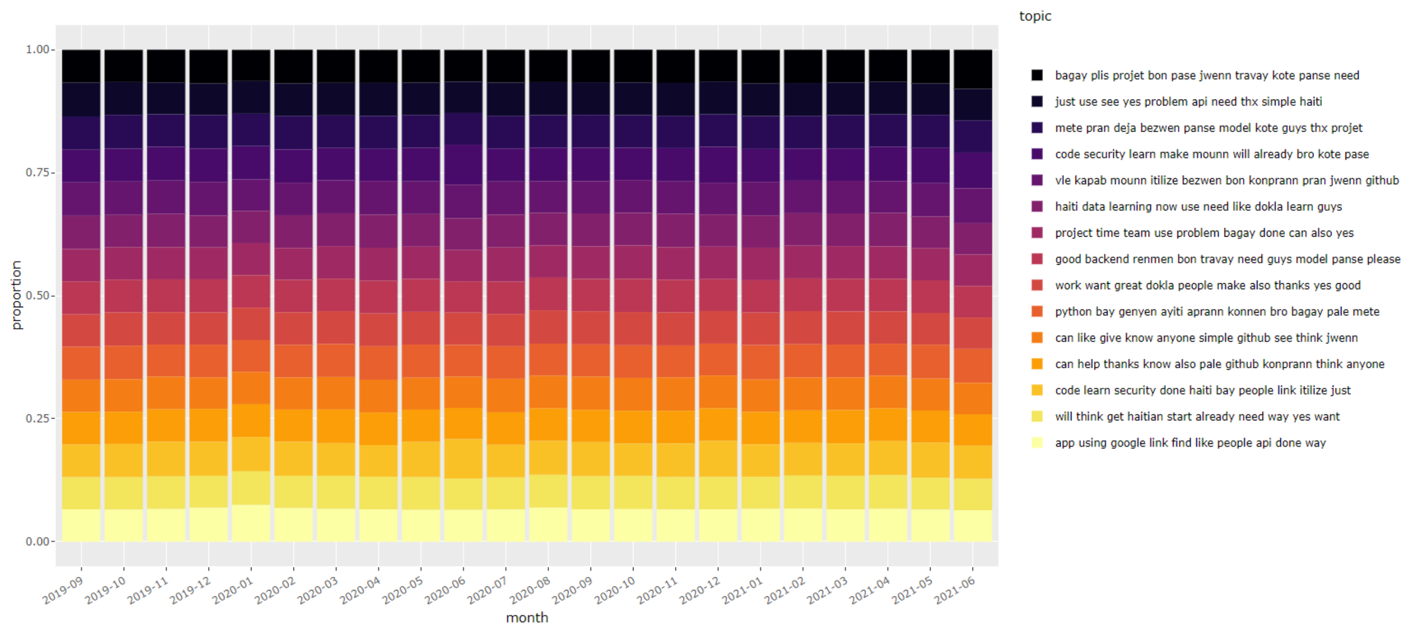
The topic modeling here, as an unsupervised learning paradigm, is built with the LDA (Latent Dirichlet Allocation) model. A fundamental aspect of this modeling is the message generation system is based on a Markov chain, itself based on strings of n-grams (or sequences of words of length n) of the corpus. The LDA model here is based on the idea that each document in the corpus (or WhatsApp message, in this case) belongs to a number of subjects, the subjects themselves being composed of a small number of words which are used most often in each of them. The model is trained multiple times to determine the maximum likelihood that each document (WhatsApp message) belongs to each subject, which is based on the probability that each subject/ word appears in the document given its frequency.

The optimal number of topics determined by the LDA model itself is 15. And note that topics such as coding, collaboration work, computer security, difficulties related to the technological ecosystem in Haiti, software engineering, etc., are dominant in conversions. Please look at the graph below to get a clearer idea. Note that in topic modeling, which is moreover unsupervised learning, labeling requires a cognitive effort on the part of the reader to be effective. It is up to the reader, therefore, to analyze the main themes of each equation in order to assign a theme that can be seen as the label.

```
#a kind of average to judge log-liklihood by
harmonicMean <- function(logLikelihoods, precision = 2000L) {
  llMed <- median(logLikelihoods)
  as.double(llMed - log(mean(exp(-mpfr(logLikelihoods,
                                       prec = precision) + llMed))))
}
```

```r
corpus_x<-tm_map(corpus, function(x) iconv(x, "latin1", "ASCII", sub=""))
dtm<-DocumentTermMatrix(corpus_x)
dtm<-removeSparseTerms(dtm,sparse=0.994)

#TDM/DTM for topic modeling
#remove empty values from DTM and original dataframe copy
rowTotals <- apply(dtm , 1, sum)
dtm<- dtm[rowTotals> 0, ]
dataset <- dataset[rowTotals> 0,]


#run LDA model n times
seqk <- seq(2, 15, 1)
burnin <- 500
iter <- 500
keep <- 50

fitted_many <- lapply(seqk, function(k) LDA(dtm, k = k, method = "Gibbs",
                                            control = list(burnin = burnin, i
ter = iter, keep = keep)))

#Extract logliks from each topic
logLiks_many <- lapply(fitted_many, function(L)  L@logLiks[-c(1:(burnin/keep)
)])

#Compute harmonic means
hm_many <- sapply(logLiks_many, function(h) harmonicMean(h))


#Create model with the GIBBS sampling method
lda.mod <- LDA(dtm, seqk[which.max(hm_many)], method = "Gibbs", control = lis
t(iter=2000))

#Gather topics
topics <- topics(lda.mod, 1)

#HEATMAT TOPIC MODELING

toptermsPerTopic <- terms(lda.mod, 10)
topicNames <- apply(toptermsPerTopic, 2, paste, collapse=" ")


tmResult <- posterior(lda.mod)
theta <- tmResult$topics

countsOfPrimaryTopics <- rep(0,seqk[which.max(hm_many)])
names(countsOfPrimaryTopics) <- topicNames
```

```
for (i in 1:nDocs(dtm)) {
  topicsPerDoc <- theta[i, ] # select topic distribution for document i
  # get first element position from ordered list
  primaryTopic <- order(topicsPerDoc, decreasing = TRUE)[1]
  countsOfPrimaryTopics[primaryTopic] <- countsOfPrimaryTopics[primaryTopic]
+ 1
}

#Topic proportions over time
#In a last step, we provide a distant view on the topics in the data over tim
e.
#For this, we aggregate mean topic proportions per month of all SOTU speeches
.
#These aggregated topic proportions can then be visualized as a bar plot.

#Append month information for aggregation
dataset$month <- substr(dataset$time, 0, 7)

#Get mean topic proportions per month
topic_proportion_per_month <- aggregate(theta, by = list(month = dataset$mont
h), mean)

#Set topic names to aggregated columns
colnames(topic_proportion_per_month)[2:(seqk[which.max(hm_many)]+1)] <- topic
Names

#Reshape the dataframe
suppressPackageStartupMessages(library(reshape))
vizDataFrame <- melt(topic_proportion_per_month, id.vars = "month")%>%
  dplyr::rename(topic=variable)
vizDataFrame$value<-round(vizDataFrame$value,4)

#Plotting topic proportions per month as bar plot
g<-ggplot(vizDataFrame, aes(x=month, y=value, fill=topic)) +
  geom_bar(stat = "identity") + ylab("proportion") +
  scale_fill_viridis(discrete = T,option="B")+
  theme(axis.text.x = element_text(angle = 30, hjust = 1))

ggplotly(g)
```



5.12 Sentiment analysis with the tidytext package

One way to analyze the sentiment of a text is to think of the text as a combination of its individual words and the sentimental content of the entire text as the sum of the sentimental content of each word. We point out that this is not the only existing paradigm. There are many more. But it is the most used. This is what we, on our side, are going to use as well. And this approach takes advantage of the tools of the tidy ecosystem. The tidytext package offers three (3) sentimental analysis lexicons:

- Afinn (developed by Finn Ârup NIELSEN)
- Bing (developed by Bing LIU et al.)
- NRC Emotion Lexicon (developed by Saif MOHAMMAD and Peter TURNEY)

These three lexicons are based on unigrams (single words). In the course of our work, we will make use of two (2) of these lexicons: bing (for the polarity of feelings) and NRC Emotion Lexicon (for the calculation of emotional charges). The bing lexicon assigns “positive” or “negative” to a set of words targeted by Scientists (Bing LIU et al.). As for the NRC Emotion Lexicon, it is a list of words (English) and their associations with eight (8) basic emotions (anger, fear, anticipation, confidence, surprise, sadness, joy and disgust) and two feelings (negative and positive).

5.12.1 Textual data pre-processing

In order to be able to carry out the sentiment analysis of the feelings released in the text (opinion mining), we used a method that was not entirely orthodox in linguistics. Despite the fact that developers use haitian Creole a few times in their messages, we assume that the corpus is essential English; and indeed it is. Thus, the sentiment analysis will be done only on English words. This way, we are able to easily use the frameworks of Bing (et al.) And Mohammad (and Turney).

Here, we put the dtm object (so named by us) in tidy format, a way of making it homogeneous with the textual format of the lexicons that we will use for sentiment analysis.

```
#Tidy format
pol_tidy<-tidy(dtm)
pol_tidy[7907:7920,]

## # A tibble: 14 x 3
##    document term    count
##    <chr>    <chr>   <dbl>
##  1 8601     panse       1
##  2 8602     bon         1
##  3 8604     bon         2
##  4 8606     api         1
##  5 8607     bon         1
##  6 8610     google      1
##  7 8610     haitian     1
##  8 8610     like        1
##  9 8611     api         1
## 10 8614     just        1
## 11 8614     pase        1
## 12 8615     get         1
## 13 8616     get         1
## 14 8617     pase        1

colnames(pol_tidy)<-c('line_number','word','count')
pol_tidy$line_number<-as.numeric(pol_tidy$line_number)
```

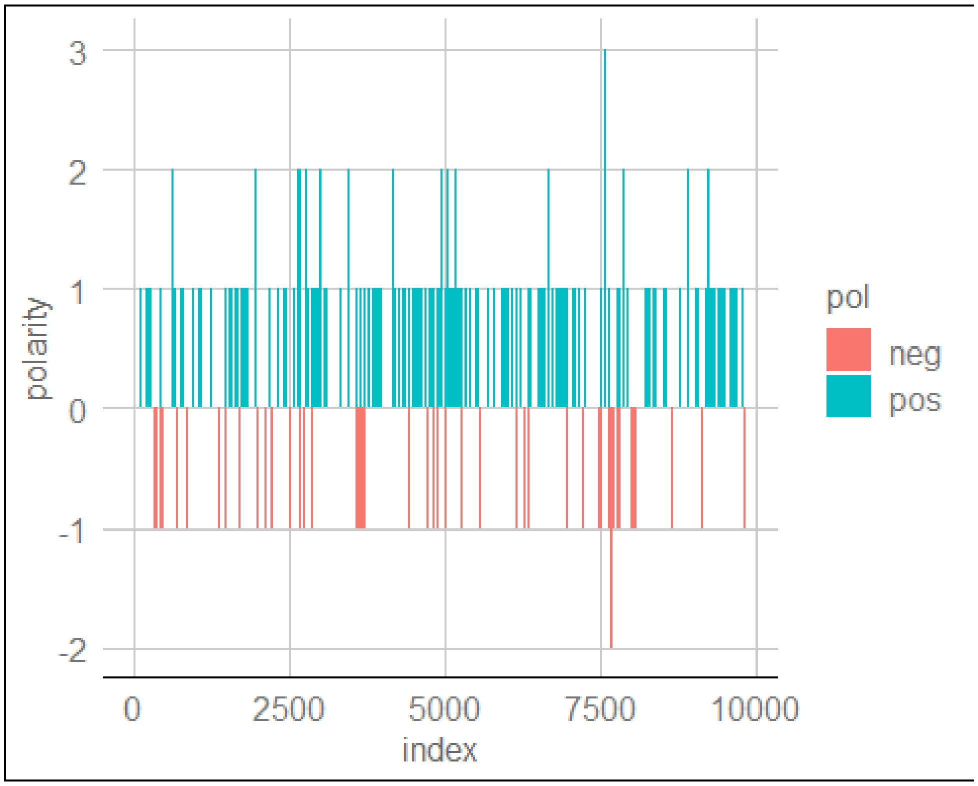
Here, thanks to the inner_join () function, we concatenate the set of polarized words from the bing lexicon with the unigrams of our object in tidy format. Thus, the technique counts all occurrences of words from the bing lexicon in the body of the tidy object. This quantifies the polarity of feelings by an occurrence counting operation.

```
# Polarity of feelings with the bing lexicon
bing<-get_sentiments("bing")
pol_sentiment<-inner_join(pol_tidy,bing)

pol_sentiment<-count(pol_sentiment,sentiment,
                     index=line_number)

pol_sentiment<-spread(pol_sentiment,sentiment,n, fill=0)
pol_sentiment[15:20,]

## # A tibble: 6 x 3
##   index negative positive
##   <dbl>    <dbl>    <dbl>
## 1   227        1        0
## 2   228        0        1
## 3   242        0        1
## 4   249        0        1
## 5   267        0        1
## 6   269        0        1
```

```
pol_sentiment$polarity<-pol_sentiment$positive-pol_sentiment$negative
pol_sentiment$pol<-ifelse(pol_sentiment$polarity>=0,
                              "pos", "neg")
```

5.12.2 Graphic representation of sentimental polarity with the Bing lexicon

```
#Plot with ggplot2
ggplot(pol_sentiment, aes(x=index, y=polarity,
                              fill=pol)) +
  geom_bar(stat="identity", position="identity",width=
           15)+theme_gdocs()
```

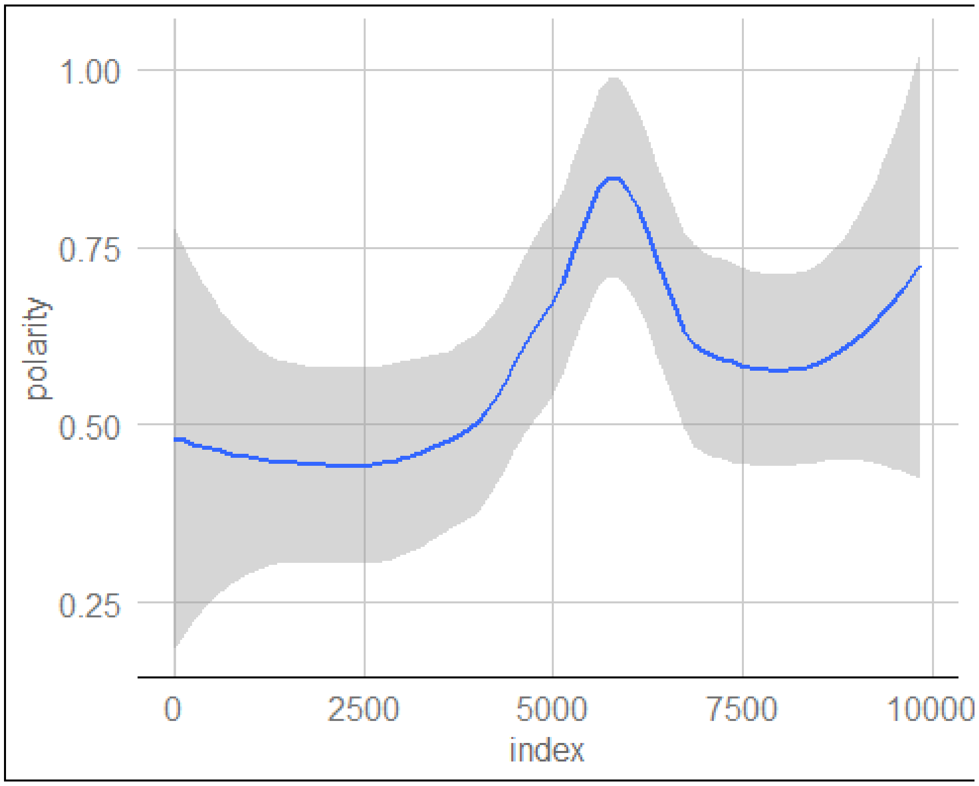


5.12.3 Application of the GAM (General Additive Modeling) algorithm to evaluate the trajectory of polarity over time

With the previous graph, it turns out to be a bit difficult to grasp the general polarity over time, as the bars are visually jagged. In contrast, a smooth visual can be more insightful than the previous choppy bar chart.

To create a smoothed line, we use the geom_smooth() layer when building a time series. The geom_smooth() layer adds a "smoothed conditional average" to the plot. The smoothed line helps identify patterns in the midst of busy visuals. In this case, the fitted line is created with a Generalized Additive Model (GAM). The GAM fits a linear model to the polarity values in pol_sentiment. Confidence intervals are estimated based on the fitted model and are also plotted with the GAM line. Adding the smoothing function to the ggplot () function is simple. We just have to add the stat_smooth() and theme_gdocs() layer for aesthetics (see Text mining in Practice with R, 2017).

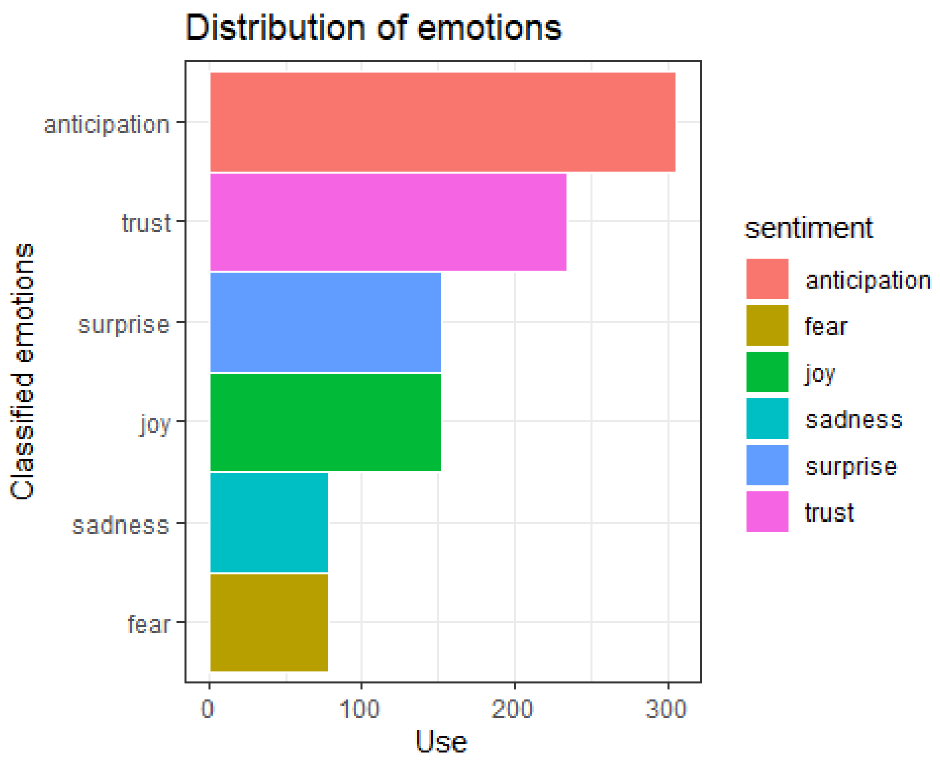
```
#smoothing
pol_smooth<- ggplot(pol_sentiment, aes(index, polarity))
print(pol_smooth + stat_smooth()+theme_gdocs())
```



5.12.4 Emotions released in the discussions

Now we make use of the “NRC Emotion Lexicon” to highlight emotions.

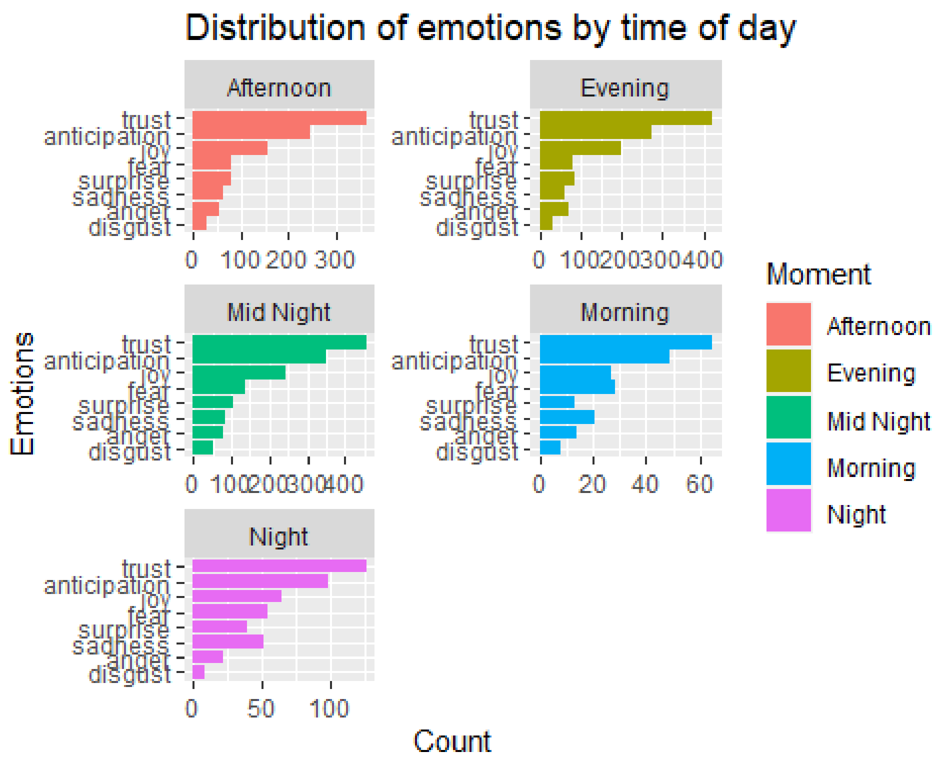
```
data=pol_tidy%>%
  inner_join(get_sentiments("nrc"),by="word")
data[c(1:10),]

## # A tibble: 10 x 4
##    line_number word     count sentiment
##          <dbl> <chr>    <dbl> <chr>
##  1           5 learn        1 positive
##  2           6 learn        1 positive
##  3           7 learn        1 positive
##  4          24 learning     1 positive
##  5          29 learning     1 positive
##  6          33 learning     1 positive
##  7          50 learning     1 positive
##  8          50 model        1 positive
##  9          51 learning     2 positive
## 10          62 learn        1 positive

sentimentcount=data%>%
  group_by(sentiment)%>%
  filter(sentiment %in% c("joy","anger",
                          "disgust","fear","sadness",
                          "surprise","anticipation",
                          "trust"))%>%
                                  count()%>%arrange(desc(n))
ggplot(sentimentcount, aes(x=reorder(sentiment,n),
                           y=n,fill=sentiment)) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_flip()+theme_bw()+labs(x='Classified emotions',y='Use',title = "Distr
ibution of emotions")
```



5.12.5 Emotions according to the time of day

We find that the distributions of emotional charges in the exchanges according to the time of day are don't have the same structures as the overall one (above). On the overall plot, we see that "anticipation" feeling takes over. However, the "trust" feeling dominates when it comes to analyzes according to the time of the day. Anyway, note that the scales are not identical (very important!!!).

```
data1=dataset%>%unnest_tokens(word,text)
textsentiment1=data1%>%
  inner_join(get_sentiments("nrc"))

  times=textsentiment1%>%mutate(time=hour(time))
times=times %>% mutate(Moment = case_when(
  time > 6 & time <= 12~ 'Morning',
  time > 12 & time <= 16 ~ 'Afternoon',
  time > 16 & time <= 21 ~ 'Evening',
  time > 21 & time <= 24 ~ 'Night',
  time > 0 & time <= 6~ 'Mid Night'))
times%>%count(sentiment,Moment)%>%
  filter(sentiment %in% c("joy","anger","disgust","fear",
                        "sadness","surprise","anticipation","trust"))%>%
  group_by(Moment)%>%filter(!is.na(Moment))%>%ungroup()%>%
  ggplot(aes(x=reorder(sentiment,n),y=n,fill=Moment))+
  geom_bar(stat="identity")+facet_wrap(~Moment,ncol=2,scales="free")+
  coord_flip()+
  labs(x="Emotions",y="Count",title = "Distribution of emotions by time of da
y")
```

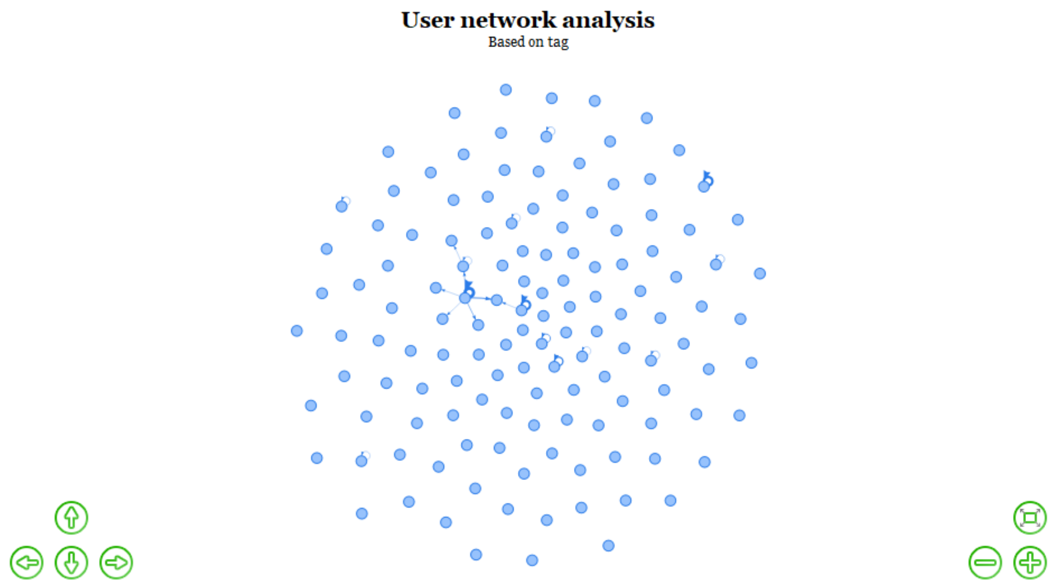


6. Network Analysis for the chat group

This network analysis is implemented using the visNetwork package in R. It is about taking into account the network formed by the participants of a group in the conversations. The limit of this analysis remains the fact that the “edges” of the said network are built around the tags in the messages. In other words, the limit lies in the fact that community detection is done for users who tag themselves in conversations. And note that the Zooming option in the graph isn’t unfortunately available because the document here doesn’t support html embedding.

```
nodes <- dataset %>%
  dplyr::group_by(id = author) %>%
  summarise() %>%
  mutate(label = id)
```

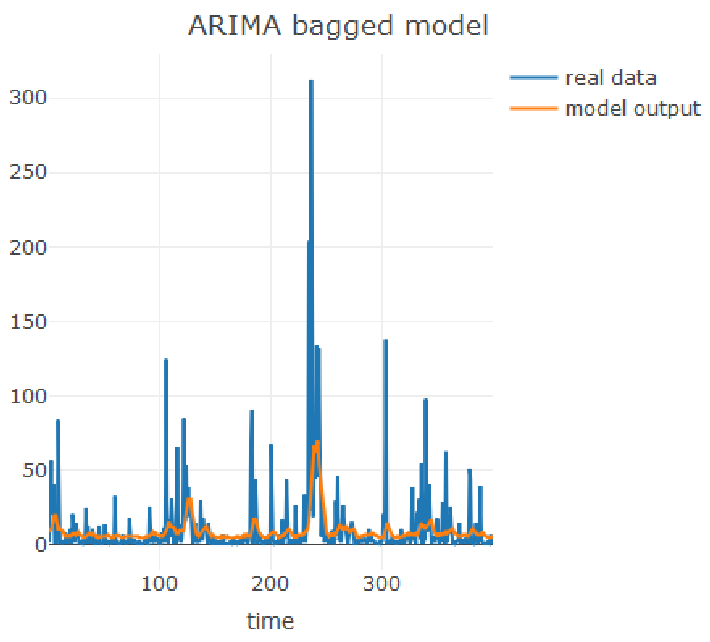
```
edges <- dataset %>%
  select(author, text) %>%
  dplyr::rename(from = author) %>%
  mutate(to = str_extract(text, paste(unique(dataset$author), collapse = "|")
)) %>%
  select(from, to) %>%
  na.omit() %>%
  dplyr::group_by(from,to) %>%
  summarise(value = n())

visNetwork(nodes, edges, main="User network analysis",
           submain = "Based on tag", font="white") %>%
  visOptions(highlightNearest = TRUE)%>%
  visEdges(arrows = 'from',font='white')%>%
  visInteraction(navigationButtons = TRUE,tooltipStyle = 'position: fixed;vis
ibility:hidden;padding: 5px;white-space: nowrap;
 font-family: cursive;font-size:18px;font-color:purple;background-color: red'
)%>%
  visNodes(font= '14px arial white')
```



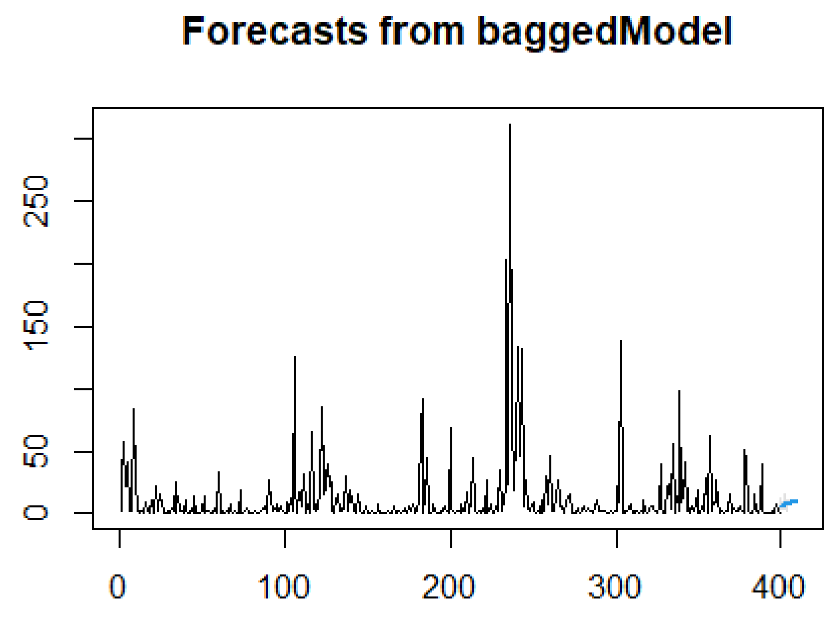
7. Forecasting

In this section, the concern is to specify a model in order to make forecasts on the flow of messages arriving daily on the group. We implement an ARIMA (Auto Regressive Integrated Moving Average) bagged-model described in Bergmeir et al. (2016). In this case, considering the original series (number of messages per day), a bootstrapped series is calculated with the Box-Cox and Loess-based decomposition (BLD) bootstrap. This is in order to escape the constraint of the probability distribution of said series. It is very difficult to describe the law of probability of Whatsapp message flow. So, with the bootstrap method, there is no need to test the normality of the series. In short, the optimal model is selected and estimated automatically using the Hyndman-Khandakar (2008) algorithm to select p and q and the Haslett and Raftery (1989) algorithm to estimate the d parameter. The Hyndman-Khandakar algorithm uses,among others, a combination of (KPSS) Kwiatkowski-Phillips-Schmidt-Shin test (unit root test), AKAIKE criterion minimization (AICc) & Maximum likelihood estimator (MLE) to obtain an optimal ARIMA model among all candidates (models). Note that one of the limitations of our model is that it gives certain prediction intervals (100%). But anyway, that's one of the eventual flaws in any boostrap process.

```
# Extraction of a sub-dataset from the original dataset including the distrib
ution of the quantities of messages per day

time_series<-dataset %>%
  mutate(day = as.Date.factor(time)) %>%
  count(day)

#Print the first 10 values
head(time_series,10)

##           day  n
## 1  2019-09-21  2
## 2  2019-09-22 30
## 3  2019-09-23 57
## 4  2019-09-24 21
## 5  2019-09-25 41
## 6  2019-09-26  2
## 7  2019-09-27  2
## 8  2019-09-28  2
## 9  2019-09-29 84
## 10 2019-09-30 27

#Training the bagged model by a boostrap process
set.seed(1996)
sarima_model <- baggedModel(time_series$n,bootstrapped_series = bld.mbb.boots
trap(time_series$n, 50),fn = auto.arima,trace=F)
arima_fitted <- round(as.vector(sarima_model$fitted),0)

time<-seq_along(time_series$n)
sarimaFitted <- as.data.frame(cbind(time,arima_fitted, time_series$n))

plot_ly(sarimaFitted, x = ~time) %>%
  add_lines(y = ~time_series$n, mode = "lines", text=~time_series$day, name =
"real data") %>%
  add_lines(y = ~arima_fitted, mode = "lines", text=~time_series$day, name =
"model output") %>%
  layout(title = "ARIMA bagged model",
         xaxis=list(autorange=TRUE),
         yaxis=list(title="",autorange=TRUE))
```



```
fcast <- forecast(sarima_model,h=10)
fcast1<-as_tibble(fcast)
fcast1$`Point Forecast`<-round(fcast1$`Point Forecast`,0)
head(fcast1,10)

## # A tibble: 10 x 3
##    `Point Forecast` `Lo 100` `Hi 100`
##               <dbl>    <dbl>    <dbl>
##  1                6    3.13     13.9
##  2                6    2.16      9.36
##  3                7    4.46     17.7
##  4                7    0.833    10.2
##  5                8    5.44     10.2
##  6                8    5.89     12.3
##  7                8    6.82     10.6
##  8                9    6.51     10.8
##  9                9    6.92     10.9
## 10                9    6.98     11.3

#Plot
plot(fcast)
```



```
sarima_tab<- as_tibble(cbind(arima_fitted, time_series$n))
sarima_tab<-cbind(sarima_tab,as.Date(time_series$day))%>%
  as_tibble()%>%
  rename(Day=`as.Date(time_series$day)`)%>%
  rename(real_data=V2)

#Arrange the tibble data by decreasing order
sarima_tab<-sarima_tab[order(sarima_tab$Day,decreasing = T),]

head(sarima_tab,10)

## # A tibble: 10 x 3
##    arima_fitted real_data Day
##           <dbl>     <dbl> <date>
##  1            6         1 2021-06-06
##  2            6         3 2021-06-05
##  3            5         7 2021-06-04
##  4            5         1 2021-05-28
##  5            5         1 2021-05-27
##  6            5         2 2021-05-25
##  7            6         1 2021-05-23
##  8            7         1 2021-05-20
##  9            7         1 2021-05-18
## 10            9         1 2021-05-08
```

8. Conclusion

In this work, we were able to demonstrate how statistically usable data generated on WhatsApp is. We were able to demonstrate, through a behavioral analysis, the hours of the day as well as the days of the week where the members of the WhatsApp group that were the subject of our study are the most active. Further on, thanks to a psycholinguistic analysis, we were able to highlight the general mood of the group through the emojis used by the leaders of the said groups, in other words the most active. In this same section, two (2) Machine Learning (ML) algorithms were used to be able to classify the volume of texts written by the members of the group into a certain number of topics. From there, we were able to realize that the dominant topics of conversations over time generally revolve around coding, collaborative projects, the difficulties of the technological ecosystem in Haiti, and so forth. And we used a kind of Heatmap to represent the probability measures of these subjects over time. Note that we have also made a rather static classification over time. For this, the t-SNE algorithm was used. Clusters of texts have been highlighted. And the six (6) clusters, albeit outnumbered compared to the optimal number self-chosen by the LDA model itself of 15, return virtually the same conclusion with regard to the topics discussed by the developers. And finally, we move towards forecasting by using an ARIMA bagged-model. The model was optimized with an algorithm explained above by retaining the appropriate parameters as well as by satisfying the various fundamental statistical tests. With this, we were able to forecast the number of messages that will be exchanged between developers within a ten (10) day horizon.

References

- CROSTON, J. (1972) “Forecasting and stock control for intermittent demands”, Operational Research Quarterly, 23(3), 289-303.
- SHENSTONE, L., and HYNDMAN, R.J. (2005) “Stochastic models underlying Croston’s method for intermittent demand forecasting”. Journal of Forecasting, 24, 389-402.
- POON, S. Huang, and GRANGER, Clive W. J. (2003), Forecasting Volatility in Financial MArket: A Review, Journal of Economic Litterature, Vol. 41, No.2, pp. 478-539
- RAVINDRA, S. Kumar, and VIKRAM, Garg(2015), Mastering Social Media Mining with R, Packt Publishing Ltd.
- KUMAR, Ashish, and PAUL, Avinash(2016), Mastering Text Mining with R, Packt Publishing Ltd.
- KULKARNI, Akshay, and SHIVANANDA, Adarsha(2019), Natural Language Processing Recipes, Apress*
- MOHAMMAD, Saif, and TURNEY, Peter(2013), Crowdsourcing a Word-Emotion Association Lexicon, Computational Intelligence, 29 (3), 436-465
- MOHAMMAD, Saif, and TURNEY, Peter(2010), Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create an Emotion Lexicon, In Proceedings of the NAACL-HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text, LA, California.
- SILGE, Julia, and ROBINSON, David(2017), Text Mining with R, A Tidy Approach,O’REILLY
- MAKHABEL, Bater, MISHRA, Pradeepta, DANNEMAN, Nathan, and HEIMANN, Richard(2016), R: Mining Spatial, Text, Web, and Social Media data, Learning Path, Packt Publishing Ltd.
- DOUC, Randal, MOULINES, Eric, and STOFFER, S. David(2014), Non-linear time series, Theory, Methods, and Applications with R Examples, CRC Press
- KWARTLER,Ted(2017), Text Mining in Practice with R, Wiley
- GHALNOS, Alexios(2020), Introduction to the rugarch package
- MATTHEWS, Clive(1998), An Introduction to Natural Language Processing Through Prolog, Routledge
- https://cran.r-project.org/web/packages/Rtsne/Rtsne.pdf
- https://github.com/JBGruber/rwhatsapp
- https://github.com/robjhyndman/forecast
- https://rpubs.com/RatherBit/90267
- https://rdrr.io/cran/astsa/man/sarima.html
- https://datascienceplus.com/time-series-analysis-using-arima-model-in-r/
- https://rpubs.com/andreasme/keras-lstm-notebook
- https://www.r-bloggers.com/time-series-deep-learning-forecasting-sunspots-with-keras-stateful-lstm-in-r/
- https://www.r-bloggers.com/generalized-additive-models/

- https://qmplus.qmul.ac.uk/pluginfile.php/1547776/mod_resource/content/2/tutorial5-solution.pdf
- https://www.machinegurning.com/rstats/tsne/
- http://freerangestats.info/blog/2016/11/27/ets-friends
- https://machinelearningmastery.com/how-to-grid-search-deep-learning-models-for-time-series-forecasting/