Team 4 : Antonin Kadi, Mathieu François et Garance Faure

HACKING HEALTH REPORT

I- Project definition

In this special period of quarantine that is so frantic, the hacking health hackathon comes just in time : we all have loads of information on the Covid 19 and have all talked and thought about the management of available information.

In our team we thought that Natural language processing and Sentiment analysis are one of the most impactful analysis so we started from here. After discussions and technical details -that we will detail afterwards- we closed our attention on the sentimental analysis of words under the hashtag #COVID19 on Twitter. After removing stop words and cleaning our data, we took the top words and divided them under positive and negative terms. We decided that this analysis would have more impact if the data collected was represented on the UK map. Thus helping realising the division of how the handling of the crisis is received in different areas.

II - Our environment

In this paragraph we will go over the technical environment we have worked with, both the software we have used and the libraries or our choices before programming.

First of all, we programmed all the code in the R language.
We put our files in Github. The link to the repository of our project is: https://github.com/warint/covid-19/tree/master/team4
The main file is "mappingUK.R", it contains all the steps of our code (connection to the twitter API, data cleaning, sentiment analysis, all the graphics and the map).

In order to be able to perform all our analyses and graphs, we used many R libraries. Here is the list of the main libraries and packages:

- Rtweet
- Leaflet and map
- Tidytext and dplyr
- Get_sentiment with the lexicon "afinn"
- Ggplot2

Rtweet is the library that allowed us to connect with the twitter API via tokens. In addition, to get the token key we had to request access to a twitter developer account. However we had a problem with the amount of data we had access to. Indeed, there are several access rights to the API and we only have basic access. So we had access to 17000 tweet initially but for many of them the location was missing. After all the cleaning and selection of the data we wanted (the tweets only coming from the "United Kingdom") we only had 190 tweets left which poses a slight problem of interpretation.
We chose to analyze the tweets from the "United Kingdom" because the tweets are written in English and the library we were using for the sentiment analysis is in English. We didn't have time to add a step in our program to translate the words from French to English for example. We preferred to focus on analysis and visualization.

Leaflet and map are the two libraries that allowed us to create the map. The code of the map was made by the Skema AI lab who kindly lent it to us. We left all the parts even if we didn't use them so that the code can be easily reused and modified for those who wish to do so.

The **tidytext** and **dplyr** libraries allowed us to clean up the data mainly.

In order to perform the sentiment analysis, we used the **package get_sentiment()** and the **lexicon "afinn"**. The technique we decided to use is the “bag of word” technique. In addition to that, we decided to use the lexicon "afinn" and not "bing" for example because "afinn" gives a score of "power" of the word. That is to say that the more negative the word is, the lower its score will be (for example -6) and the more positive it is, the higher its score will be (for example 5).

III - Technical work

In order to realize this project, we used as a basis a code already developed by the laboratory named “Mapping UK Tweets”. We add to this our own sentiment analysis and we adapted the result map to fit with.

Here you can find the general structure of the R code and after the detail of the important parts.

Datas district retrieval

⇩

Datas Tweets retrieval

⇩

Data Tweets cleansing & Mining

⇩

Sentiment Analysis

⇩

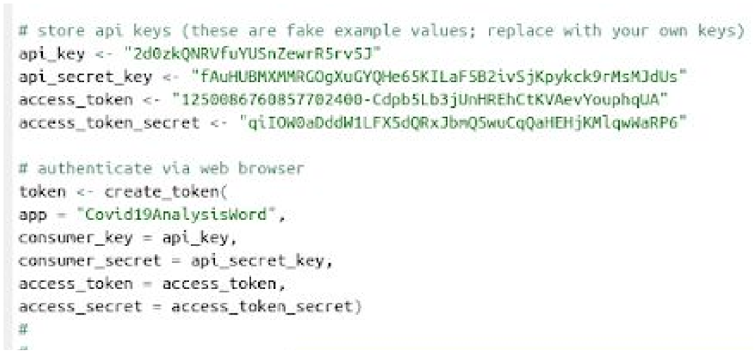
Load district data on the map

⇩

Print on the map result of sentiment analysis

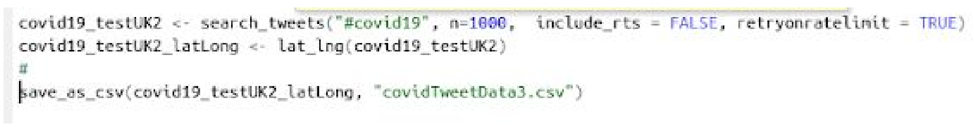
The first part of the code is the retrieval of the data for the district. In this version of the code, we don’t use this datas because our computers were not powerful enough to support interactive map display. We have left the code so you can use to adapt our solution to your version.

After we created a token in order to have access to the twitter’s API. For this, we created an app from a Twitter developer account and it gave us keys in order to request the API as you can see below.



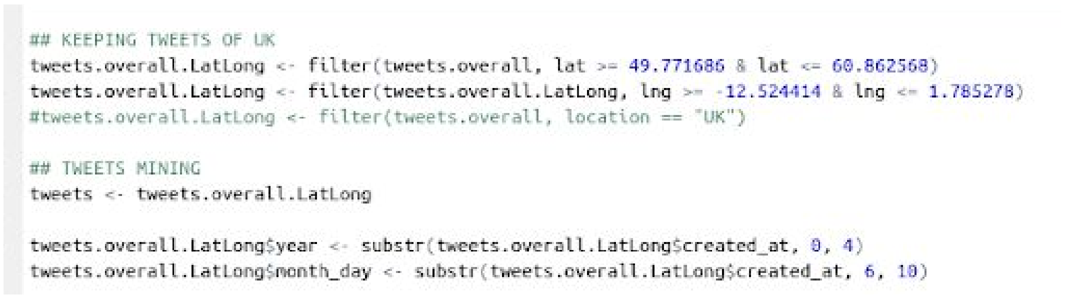
```r
# store api keys (these are fake example values; replace with your own keys)
api_key <- "2d0zkQNRVfuYUSnZewrR5rv5J"
api_secret_key <- "fAuHUBMXMMRGOgXuGYQHe65KILaF5B2ivSjKpykck9rMsMJdUs"
access_token <- "1250086760857702400-Cdpb5Lb3jUnHREhCtKVAevYouphqUA"
access_token_secret <- "qiIOW0aDddW1LFX5dQRxJbmQ5wuCqQaHEHjKMlqwWaRP6"

# authenticate via web browser
token <- create_token(
app = "Covid19AnalysisWord",
consumer_key = api_key,
consumer_secret = api_secret_key,
access_token = access_token,
access_secret = access_token_secret)
#
#
```

With this access and the package “Rtweet”, we were able to download tweets with the parameters we wanted. For this, we looked for tweets with the hasthag “#Covid19”. We also added to our datas the longitude and the latitude (thanks to the function lat_lng()) of the tweets which will be necessary for the next part of the code. And we saved this datas in a CSV file in order to save them.



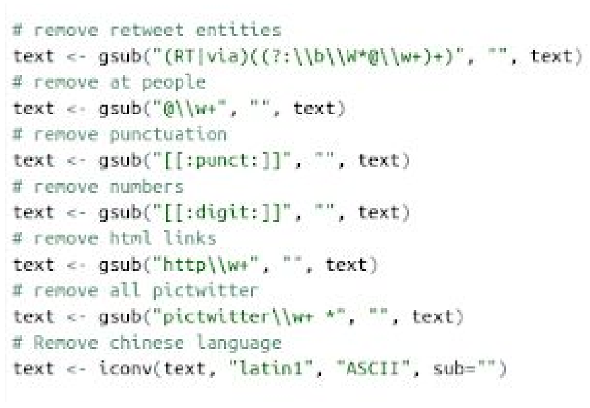
```r
covid19_testUK2 <- search_tweets("#covid19", n=1000,  include_rts = FALSE, retryonratelimit = TRUE)
covid19_testUK2_latLong <- lat_lng(covid19_testUK2)
#
save_as_csv(covid19_testUK2_latLong, "covidTweetData3.csv")
```

Here we add some specifications to our datas. First, we kept only the datas from UK tweets and we kept the date of the publication of the tweet in order to make comparison of the sentiment analysis depending on the different events.



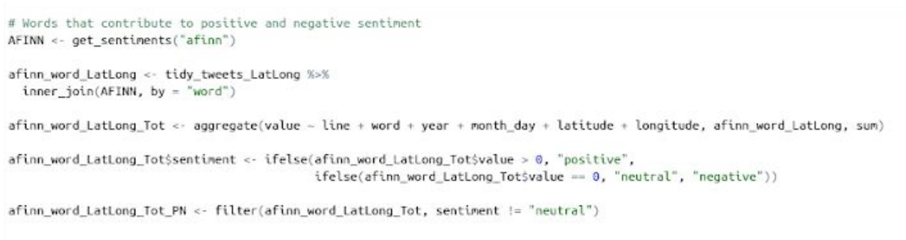
```r
## KEEPING TWEETS OF UK
tweets.overall.LatLong <- filter(tweets.overall, lat >= 49.771686 & lat <= 60.862568)
tweets.overall.LatLong <- filter(tweets.overall.LatLong, lng >= -12.524414 & lng <= 1.785278)
#tweets.overall.LatLong <- filter(tweets.overall, location == "UK")

## TWEETS MINING
tweets <- tweets.overall.LatLong

tweets.overall.LatLong$year <- substr(tweets.overall.LatLong$created_at, 0, 4)
tweets.overall.LatLong$month_day <- substr(tweets.overall.LatLong$created_at, 6, 10)
```

After that we cleaned the text from the tweets. We took of the retweet entities, punctuations, html links..



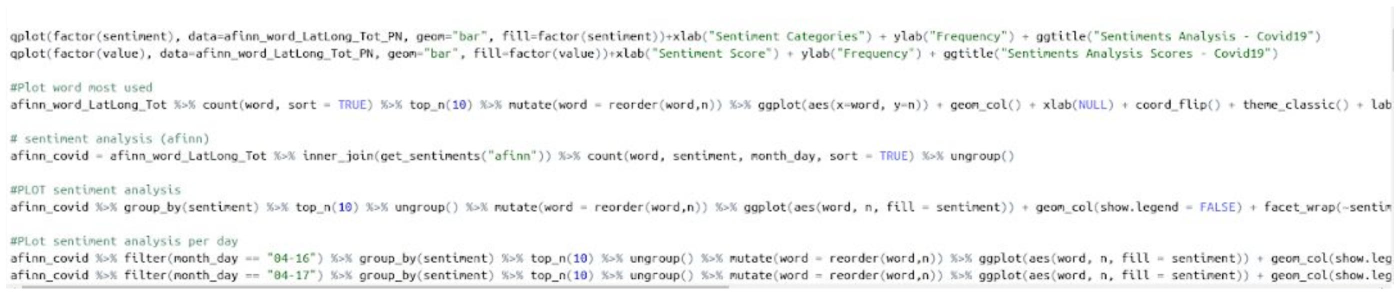
```
# remove retweet entities
text <- gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", text)
# remove at people
text <- gsub("@\\w+", "", text)
# remove punctuation
text <- gsub("[[:punct:]]", "", text)
# remove numbers
text <- gsub("[[:digit:]]", "", text)
# remove html links
text <- gsub("http\\w+", "", text)
# remove all pictwitter
text <- gsub("pictwitter\\w+ *", "", text)
# Remove chinese language
text <- iconv(text, "latin1", "ASCII", sub="")
```

So the datas were finally ready to our analysis. So we use the lexicon “Afinn” for our sentiment analysis. We have chosen this lexicon because this one returns a sentiment score of each word while the lexicon “Bing” does not.



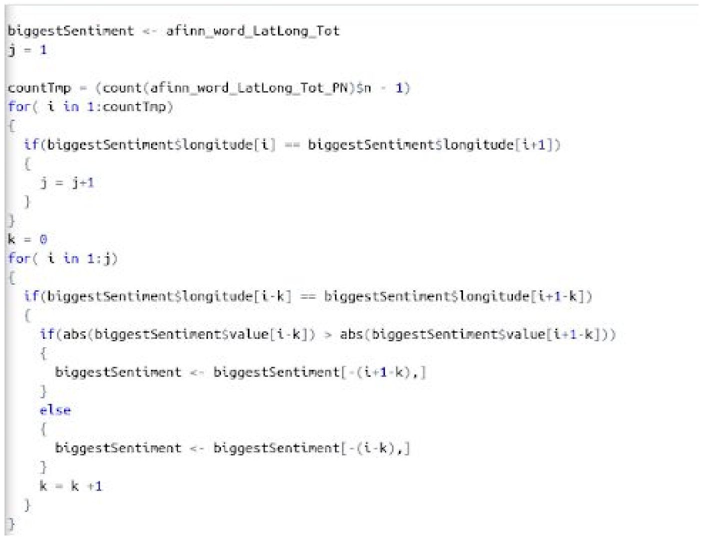
```
# Words that contribute to positive and negative sentiment
AFINN <- get_sentiments("afinn")

afinn_word_LatLong <- tidy_tweets_LatLong %>%
  inner_join(AFINN, by = "word")

afinn_word_LatLong_Tot <- aggregate(value ~ line + word + year + month_day + latitude + longitude, afinn_word_LatLong, sum)

afinn_word_LatLong_Tot$sentiment <- ifelse(afinn_word_LatLong_Tot$value > 0, "positive",
                                           ifelse(afinn_word_LatLong_Tot$value == 0, "neutral", "negative"))

afinn_word_LatLong_Tot_PN <- filter(afinn_word_LatLong_Tot, sentiment != "neutral")
```

And with this sentiment analysis, we generated 4 plots which gives us a better way to see the results of this sentiment analysis than just the scores themselves. You will find these plots in the next part.



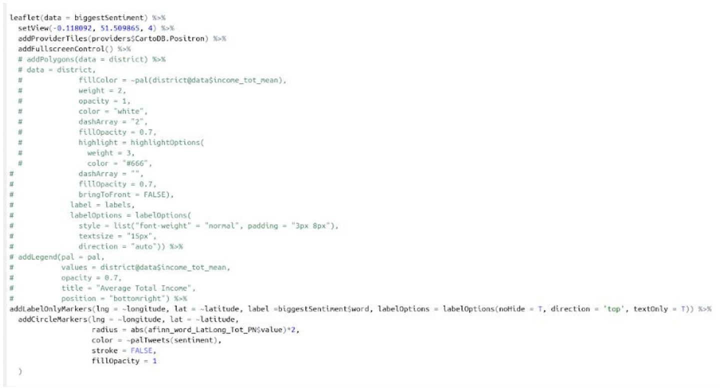
```
qplot(factor(sentiment), data=afinn_word_LatLong_Tot_PN, geom="bar", fill=factor(sentiment))+xlab("Sentiment Categories") + ylab("Frequency") + ggtitle("Sentiments Analysis - Covid19")
qplot(factor(value), data=afinn_word_LatLong_Tot_PN, geom="bar", fill=factor(value))+xlab("Sentiment Score") + ylab("Frequency") + ggtitle("Sentiments Analysis Scores - Covid19")

#Plot word most used
afinn_word_LatLong_Tot %>% count(word, sort = TRUE) %>% top_n(10) %>% mutate(word = reorder(word,n)) %>% ggplot(aes(x=word, y=n)) + geom_col() + xlab(NULL) + coord_flip() + theme_classic() + lab

# sentiment analysis (afinn)
afinn_covid = afinn_word_LatLong_Tot %>% inner_join(get_sentiments("afinn")) %>% count(word, sentiment, month_day, sort = TRUE) %>% ungroup()

#PLOT sentiment analysis
afinn_covid %>% group_by(sentiment) %>% top_n(10) %>% ungroup() %>% mutate(word = reorder(word,n)) %>% ggplot(aes(word, n, fill = sentiment)) + geom_col(show.legend = FALSE) + facet_wrap(~sentim

#PLot sentiment analysis per day
afinn_covid %>% filter(month_day == "04-16") %>% group_by(sentiment) %>% top_n(10) %>% ungroup() %>% mutate(word = reorder(word,n)) %>% ggplot(aes(word, n, fill = sentiment)) + geom_col(show.leg
afinn_covid %>% filter(month_day == "04-17") %>% group_by(sentiment) %>% top_n(10) %>% ungroup() %>% mutate(word = reorder(word,n)) %>% ggplot(aes(word, n, fill = sentiment)) + geom_col(show.leg
```

We have chosen to keep only one word/sentiment by each tweet because our map is not interactive. For this, we kept only the strongest sentiment and we saved it in a new dataset. It was very hard to have a clean result with several words/sentiments to display on the map. However, if you take this code and you have an interactive map, we think this part of the code will be useless.



```
biggestSentiment <- afinn_word_LatLong_Tot
j = 1

countTmp = (count(afinn_word_LatLong_Tot_PN)$n - 1)
for( i in 1:countTmp)
{
  if(biggestSentiment$longitude[i] == biggestSentiment$longitude[i+1])
  {
    j = j+1
  }
}
k = 0
for( i in 1:j)
{
  if(biggestSentiment$longitude[i-k] == biggestSentiment$longitude[i+1-k])
  {
    if(abs(biggestSentiment$value[i-k]) > abs(biggestSentiment$value[i+1-k]))
    {
      biggestSentiment <- biggestSentiment[-(i+1-k),]
    }
    else
    {
      biggestSentiment <- biggestSentiment[-(i-k),]
    }
    k = k +1
  }
}
```

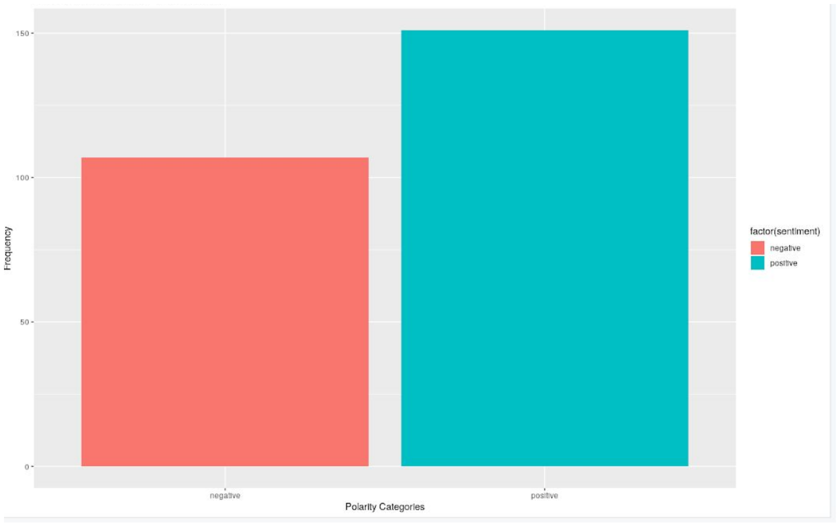
Finally, we displayed our result on the map thanks to the function leaflet(). We draw circles green or red according of the sentiment is positive or negative. We also updated the circle radius according to the score of the sentiment.And after that, we displayed the word according to the sentiment on the top of the circles. You will find the result in the next part.



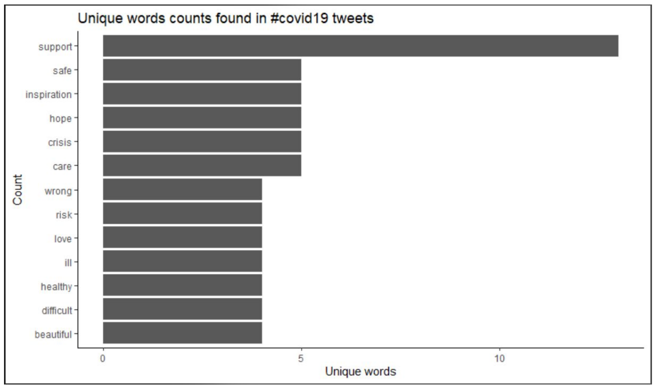
```
leaflet(data = biggestSentiment) %>%
  setView(-0.118092, 51.509865, 4) %>%
  addProviderTiles(providers$CartoDB.Positron) %>%
  addFullscreenControl() %>%
  # addPolygons(data = district) %>%
  # data = district,
  #             fillColor = ~pal(district@data$income_tot_mean),
  #             weight = 2,
  #             opacity = 1,
  #             color = "white",
  #             dashArray = "2",
  #             fillOpacity = 0.7,
  #             highlight = highlightOptions(
  #               weight = 3,
  #               color = "#666",
#               dashArray = "",
#               fillOpacity = 0.7,
#               bringToFront = FALSE),
#             label = labels,
#             labelOptions = labelOptions(
#               style = list("font-weight" = "normal", padding = "3px 8px"),
#               textsize = "15px",
#               direction = "auto")) %>%
# addLegend(pal = pal,
#           values = district@data$income_tot_mean,
#           opacity = 0.7,
#           title = "Average Total Income",
#           position = "bottomright") %>%
addLabelOnlyMarkers(lng = ~longitude, lat = ~latitude, label =biggestSentiment$word, labelOptions = labelOptions(noHide = T, direction = 'top', textOnly = T)) %>%
  addCircleMarkers(lng = ~longitude, lat = ~latitude,
                   radius = abs(afinn_word_LatLong_Tot_PN$value)*2,
                   color = ~palTweets(sentiment),
                   stroke = FALSE,
                   fillOpacity = 1
  )
```

IV- Results and plot explanations.

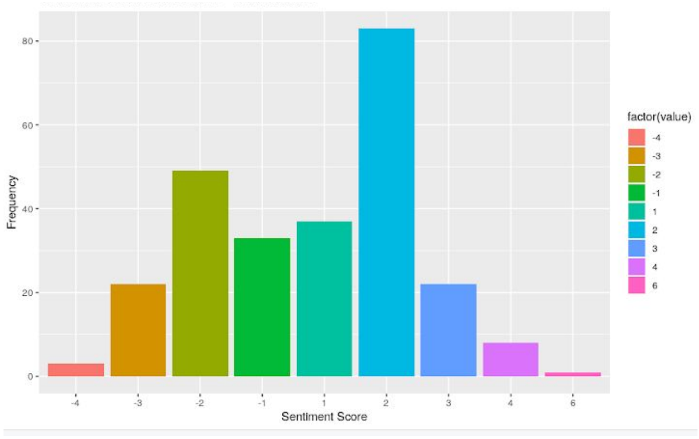
- This chart shows the frequency of negative and positive words. Surprisingly the frequency of positive comments is higher than the negative one's. Notice that here the frequency is number of non-unique words.



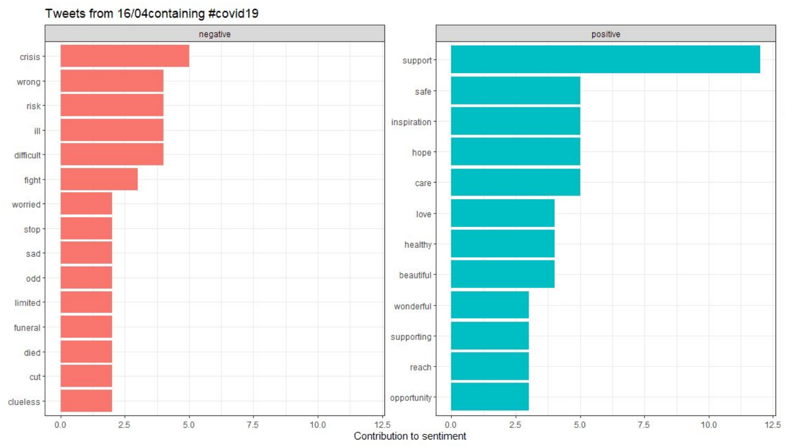
- This bar chart here represents the unique words count found in the #Covid19 on Twitter. Here the split of negative or positive comment isn't done yet. We can see that support is the highest represented with more than 15 occurrences, second is safe and third is inspiration. From the second to the 6th, the occurrence of unique words are similar, this is due to the fact that we used a predefined set of datas that isn't representative of the result.

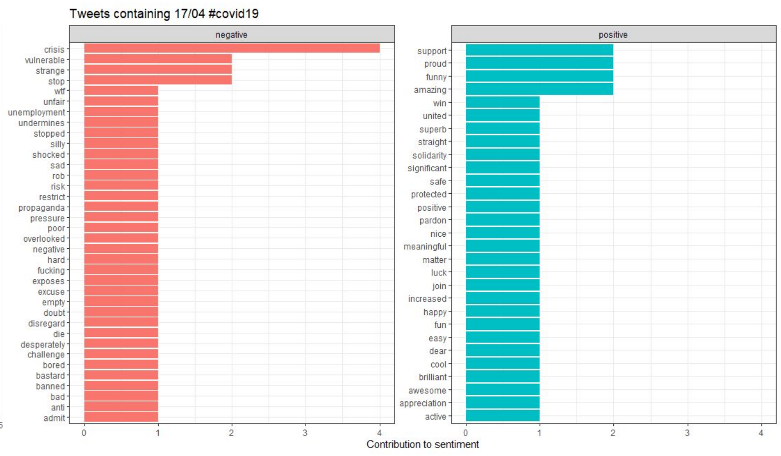


- This plot represents the frequency of words according to their sentiment score (from -4 to 6). The sentiment score helps to built up afterwards the positiveness or negativeness of a bag of words. Here we go deeper in the meaning of positive and negative : indeed, we can see that even thought the frequency of positive words is higher than the negative ones there “positiveness” is to be relativised: it is of 2 on a scale of 6.

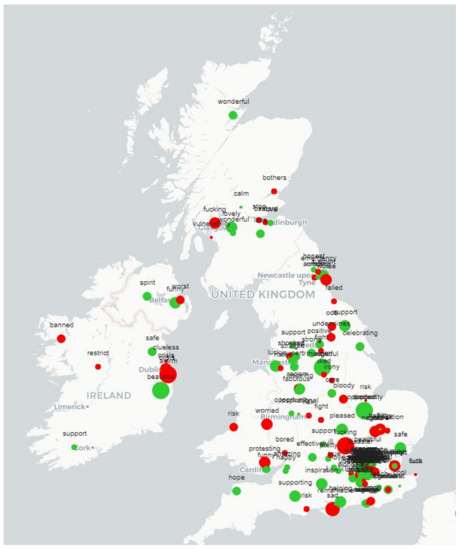


- Theses plots represent the split of positive and negative terms on the 16th and 17th under the #covid19 (as explained beneath, this is not ideal and depends on how data is collected). On the 17th, we see that the negative word criss is the one appearing the most. After that in the negative rank appears vulnerable and strange. Showing that people mostly feel lost and disoriented. Occurring as often but in positive terms we see support, proud and funny. This could lead us to think that some people can still feel positive about the situation and find a sense of purpose (suport, proud).





- Here you can see our final result : the map of the United Kingdom with for each appearance of a wording that is considered as positive : a green dot and for each wording that is considered negative, a red dot. The size of the dot varies according to the number of occurrences.



V - Conclusion

We are very glad to have done this project. We feel like we have learned new R skills and sentiment analysis. Also, it allowed to work on a subject that we cared about, so this project is complete success to our eyes.

As we said above, a part of the code (the map in particular) was given to us by the AI lab of Skema. We left all the part even if we did used them in order for the code to be easily modified.

If someone want to continue this project, we will advise him to have access to the enterprise premium API of twitter in order to have lots of data available and avoid our problem of lack of tweet to analyze. Or find a way to collect tweets without using rtweet or without using the twitter API.