# R for Social Media Analysis

Saurabh Dhawan[*] & Simon Hegelich

Bavarian School of Public Policy, Technical University of Munich, Germany

[*] Corresponding Author. Email: saurabh.dhawan@tum.de

Final draft for:

L. Sloan & A. Quan-Haase (Eds.), *The SAGE Handbook of Social Media Research Methods* (pp. xxx). SAGE Publications Ltd.

## Abstract

Analyzing social media data requires tools that are both sophisticated enough to handle its volume and velocity, and customizable enough to fit an evolving set of features and use cases. R fits the bill on both counts. In this chapter, with Twitter as our platform of choice, we will first show how to use R to collect and explore social media data. Then utilizing the rich variety of domain-specific packages available in R, we will illustrate the preliminary steps for a range of analytical possibilities that are typical to social media data – analyzing the tweets for what they are about (quantitative text analysis), the sentiment therein (sentiment analysis), changes in trends over time (time series analysis), and drawing a map of their locations (geospatial mapping).

*Keywords:* R, Tutorial, Twitter, social media, text analysis, sentiment analysis, time series, geospatial mapping

## Introduction

Social media data is often characterized by the 3 V's of volume, variety and velocity. Twitter alone has 199 million daily users (Twitter Inc., 2021) who send hundreds of millions of tweets a day. At the same time, social media data is, by its very nature, dynamic and defies set templates – existing platforms keep introducing new features and new platforms keep rising to the fore every few years. In addition, the extent and terms of access to social media data granted to researchers remain somewhat limited and subject to (often arbitrary) changes by the private corporations that own it (Hegelich, 2020). These novelties and limitations constantly bring with them opportunities for novel use cases and methodological challenges for researchers. Social media analysis thus requires tools that are, at the same time, sophisticated enough to handle its volume and velocity, customizable enough to fit its evolving set of features and use cases, and dynamic enough to respond to the unreliable nature of terms of access granted by the platforms.

R fits the bill on all counts. It is a mature and sophisticated programming language that was built from the ground up for statistical analysis, which makes it especially suitable for data science. It is customizable, open-source and freely available under the GNU General Public License in source code form. Thus, not only is it non-proprietary and free of cost but it can also be inspected, modified and redistributed as per the user's need. It compiles and runs on a wide variety of operating systems and can be used with several well-regarded IDEs (integrated development environment), such as RStudio, the R extension pack for Visual Studio Code, or Jupyter notebooks.

Along with Python, it has emerged as one of the two programming languages of choice for data science (Muenchen, 2019). Especially in academic research, it is immensely popular and holds an edge over all other competitors. This prevalence in academia constitutes a major advantage in itself as this results in many of the cutting-edge analytical tools getting first developed in R. Another major advantage fueling its popularity is its comprehensive collection of extension libraries called as 'packages', which extend the general functionality of the base R language and offer features customized for domain-specific needs. A variety of such R packages are available to cover the whole pipeline of social media analysis from scraping data to text mining.

In this chapter, we will show how to use R to collect, analyze and visualize social media data with Twitter as our platform of choice and political communication by British politicians on Twitter as our subject of interest. Political communication has been one of the central themes on Twitter since its inception and the platform has emerged as a frequent medium of study in political science research, for instance, to study fake news (Grinberg et al., 2019), political discourse

(Papakyriakopoulos et al., 2020), estimating political orientation of users (Shahrezaye et al., 2019; please note that the current Twitter terms and conditions advise against it, see *Developer terms, 2021*) and election forecasts (Tumasjan et al., 2011). We will extract tweets posted by leaders of the five most prominent political parties in the UK (Boris Johnson, Keir Starmer, Nicola Sturgeon, Edward Davey and Nigel Farage) and utilize the variety of subject-specific packages available in R to analyze what their tweets are about, the sentiments therein, changes in trends over time and finally draw a map of their locations.

We aim for this to be an introductory hands-on tutorial specifically aimed at the preparatory steps and preliminary analyses in several different domains using Twitter data. The intended scope of each section will be set to take the reader to a level of analysis in the respective domain beyond which analysis is not specific to Twitter data but becomes a problem like any other in that domain. We, however, assume familiarity with the basics of R, especially the packages *dplyr* and *ggplot2* from the tidyverse. For those who are still to start using R or need to brush up on the basics, we whole-heartedly recommend Wickham & Grolemund (2016)'s introductory textbook, R for Data Science, which is available for free online.

## Getting Data from Twitter

Where available and accessible, an API (Application Programming Interface) is often the easiest way to collect well-structured data from the web. An API acts as the mediator for exchange of data between the user's program and the platform servers and also provides a list of possible requests and commands that can be executed. The analogy often given is that of a restaurant waiter communicating between customers and the kitchen staff – not only conveying orders to the kitchen and food back to the right customers but first providing the customers with a clear menu of what is available to be ordered. In this section, we will show how to get access to the Twitter API from within R and collect tweets and the associated metadata using the *rtweet* package.

## Loading Essential Packages

We will first load the essential libraries that will be used throughout the session while other libraries specific to a section will be introduced when needed. Please note that the following code was written and tested with R version 4.0.2 (Published: 2020-06-22). Several of the packages listed below might not work with older versions of R. We have stated the package version number and minimum R version required for each individual package as comments within the code.

```
#Installing required packages
#Insert the name of a package below to install it

install.packages("insert_package_name")

#IMPORTANT# From here onwards, the rest of the analysis assumes that
#the following libraries/packages are installed and loaded
#and will not be mentioned again in each section

library(rtweet) # v0.7.0 Requires R (>= 3.1.0)
library(tidyverse) # v1.3.1 Requires R (>= 3.3)
library(tidytext) # v0.3.1 Requires R (>= 2.10)
library(ggthemes)# v4.2.4 Requires R (>= 3.3.0)
```

## The *rtweet* Package

*rtweet* (Kearney, 2019) is a comprehensive and mature R package, developed by Michael W. Kearney, that provides a range of functions designed to simplify access and analysis of Twitter data. First, it simplifies the process of extracting data from Twitter's API. Second, it wrangles the .json files retrieved from the Twitter API into a tidy structure – a tibble object, which is the R tidyverse equivalent of a data frame and is relatively easier to work on within R.

## Getting Access to the Twitter API

Twitter requires users to authenticate themselves and calls made to the Twitter API are restricted by access tokens. This allows Twitter to set limits to who can download what information based on the type of subscription: Standard, Premium, or Enterprise. We will make use of the Standard API v1.1 which is completely free of cost and has a rich set of features that are more than sufficient for anyone starting out with analyzing Twitter data. Authorization to use it can be gained in one of the following two ways:

The quick but less rigorous way is through *rtweet*'s embedded rstats2twitter app via a web browser. First, install the R packages *rtweet* and *httpuv*. Second, make sure that you are logged in to your twitter account in your web browser. Third, run any of the functions from the rtweet package mentioned below that make a request to twitter API. This should open a window in your web browser with a message, "Authorize the rstats2twitter app by logging into Twitter, or selecting 'Authorize app'". Select 'Authorize app' and you can immediately start collecting twitter data.

```
### Authentication Method I: Quick but less robust
# Using rtweet's embedded rstats2twitter app

#1# Log in to your twitter account in your web browser

#2# Load required libraries

library(rtweet)

#To enable browser-based authentication;
library(httpuv) # v 1.6.1 Requires R (>= 2.15.1)

#3# Initialize by running any rtweet function that makes an api call

search_tweets('#nhs', n = 100)

#4# Check your browser window and select 'Authorize app' when prompted.
```

The more robust but somewhat tedious way is to gain access through your own access keys. For this, first, go to https://developer.twitter.com/en/apply-for-access, click on 'Apply for a developer account' and follow the instructions (see Janetzko, this volume, for a more detailed discussion on accessing APIs). Second, once you are signed in to your developer account, navigate to *Developer Portal* > *Projects and Apps* > *Overview* > + *Create App*. Give a unique name to your app (this might take a few tries) and on the next screen, you would find your API Key and API Secret Key. Save this information. Third, within the same page click on *Developer Portal* > *Projects and Apps* > *Overview* > YourAppName > Settings, enable 3-legged OAuth. Then *Authentication settings* > *Edit* and under *CALLBACK URLS* enter http://127.0.0.1:1410. Under *WEBSITE URL*, if you don't have another personal website, you can enter your Twitter profile URL. Finally, next to the *Settings* button, click on *Keys and tokens* and generate and save your *Access Token* and *Access Secret.*

Enter the information you saved into an R script file as follows and pass them along to the *create_token()* function:

```
### Authentication Method II: Laborious but more robust
## Create and save your own Twitter credentials and authentication
token

#1# Load required libraries

library(rtweet)

#2# Replace the quoted text below with the unique identification
elements that you saved from your twitter developer account

twitter_token <- create_token(
  app = "your_app_name",
  consumer_key = "your_api_key",
  consumer_secret = "your_api_secret_key",
  access_token = "your_access_token",
  access_secret = "your_access_token_secret")

#3# check to see if the token is loaded
```

```
get_token()
View(twitter_token)
```

Note that, at the time of writing Twitter is slowly rolling out its API v2.0. Some of its functionality has been made available as ‘Early Access’, while other planned features are listed as ‘incubating’ and ‘nesting’ (Guide to the future of the Twitter API, 2021). Twitter’s API v2 will introduce multiple product tracks – Standard, Business and Academic Research. Within each track, there will be multiple access levels available – Basic, Elevated or Managed. The academic track would be available for free to students and researchers affiliated to accredited universities and research institutions. It would offer access to all the Twitter API v2 endpoints and a higher monthly volume cap. It would also offer access to the full-archive search endpoint and more precise filtering capabilities across all endpoints to enable researchers to better limit data collection to what is relevant for a particular study (see Tornes & Trujillo, 2021 for further details).

However, the API v2.0 is not yet compatible with many of the required tools (most importantly, for our purpose, the package *rtweet*) and the benefits of the ‘Early Access’ features accrue mostly to advanced users. Until the transition is complete and the relevant packages are updated, in our opinion, a beginner would be better served by the Standard API v1.1, which is more robust, easier to find online help for, and compatible with a wider range of tools and packages.

## Collecting Data

The *rtweet* package provides several different functions that allow you to collect tweets and user data from the Twitter API based on different criteria.

```
#Collecting Tweets

search_tweets() #Get tweets matching a user provided search query from
the past 6-9 days.

stream_tweets() # Returns tweets from the live stream

get_timeline() #Returns tweets from specific Twitter users

# Collecting User Data

search_users() # Returns data for up to 1,000 users matched by user
provided search query

lookup_users() # Returns data on specific Twitter users

get_friends() #Get user IDs of accounts followed by a user

get_followers() #Get user IDs of accounts following a user

get_favorites() # Get statuses or tweets favorited by a user
```

We will start by downloading a dataset of tweets from the accounts of leaders of 5 prominent political parties in the UK. We can do so with the *get_timeline* function, which returns up to 3,200 statuses posted to the timelines of each of the specified Twitter users.

```
#1# Collect 3000 Tweets posted to the timelines of each user

tweets <- get_timeline(c("BorisJohnson", "Keir_Starmer",
                         "EdwardJDavey", "NicolaSturgeon",
                         "Nigel_Farage"),
                       n = 3000)

#2# See all the variables in the retrieved data frame

names(tweets)
```

## Exploring and Characterizing the Data

The *tweets* data frame now has ~3000 tweets each for 5 different politicians or a total of about 15000 tweets. Data thus retrieved includes not only the tweet text but also a range of over 90 metadata variables ranging from user identifiers and time stamps to friends and followers count. See here for a data dictionary: https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet. These variables can be used to characterize both tweets and users in a number of ways, especially the extent of their activity and influence. This is an important step as social media activity, in general, is very unevenly distributed; a few users are hyperactive (and/or influential in terms of number of followers) and tweet a lot and get many more likes and retweets while a typical user is passive most of the time (Papakyriakopoulos et al., 2020). It is thus generally advisable to examine the data for users with inordinate amounts of activity or influence.

Within these 15000 tweets we retrieved, the code below shows how to find the following by using basic data wrangling functions from the *dplyr* package (part of the tidyverse):

1. the most popular tweets,
2. the most favorited tweets,
3. the most influential users, and
4. the most popular hashtags

For this, we make use of the *slice_max* and *select* function from *dplyr* among others. The *select* function helps us winnow down the 90 variables present in the retrieved tweets data by letting us subset only the variables of interest using their names. The *slice_max* function sorts the values of a variable in descending order and displays the specified number of values from the top. For users with older versions of R, please note that slice_max() has superseded the similar older function top_n() (for more details see *dplyr documentation, 2021*). In the current analysis, while we downloaded the same number of tweets (~3000) per user, the 5 politicians vary widely in terms of their number of followers, which biases the engagement metrics (number of retweets and likes) of their tweets.

```
#1# Find the most popular tweets (in terms of number of retweets)

tweets %>%
  slice_max(retweet_count, n = 1) %>%
  select(screen_name, retweet_count, text)

>> A tibble: 1 x 3
  screen_name  retweet_count text
  <chr>                <int> <chr>
1 BorisJohnson        117448 "Over the last 24 hours I have developed
mild symptoms and tested positiv…

#2# Find the most favorited (liked) tweets

tweets %>%
  slice_max(favorite_count, n = 1) %>%
  select(screen_name, favorite_count, text)

>> A tibble: 1 x 3
  screen_name  favorite_count text
  <chr>                 <int> <chr>
1 BorisJohnson         349346 "Over the last 24 hours I have developed
mild symptoms and tested positiv…

#3# Find the most influential users (in terms of number of followers)

tweets %>%
  distinct(user_id, .keep_all = TRUE) %>%
  slice_max(followers_count, n = 5) %>%
  select(screen_name, followers_count)

>> A tibble: 5 x 2
  screen_name    followers_count
  <chr>                    <int>
1 BorisJohnson           3538545
2 Nigel_Farage           1618007
3 NicolaSturgeon         1455727
4 Keir_Starmer           1111777
5 EdwardJDavey             65020

#4# Find the most popular hashtags
#See the next section on Text analysis
#for details on unnest_tokens()

tweets %>%
  unnest_tokens(hashtag, text, "tweets", to_lower = FALSE) %>%
  filter(str_detect(hashtag, "^#")) %>%
  count(hashtag, sort = TRUE)
```

```
>> A tibble: 1,628 x 2
   hashtag                      n
   <chr>                    <int>
 1 #coronavirus               309
 2 #VoteEd                    306
 3 #PMQs                      189
 4 #GetBrexitDone             166
 5 #BothVotesSNP              120…
```

## Quantitative Text Analysis

The heart of Twitter is the tweet, a short text message that might sometimes include pictures, emoticons etc. but is for the most part text written in a natural human language. Indeed, much of social media content is textual data, whether that's tweets, Facebook posts, or user comments on YouTube videos. Seen together with emails, blogs, messaging apps, and the plethora of other textual digital trace data that's generated every day, some have even termed our age as the age of scripturience—possessed by an obsessive desire to write incessantly (Seymour, 2020). Accordingly, the first point in analyzing social media data is often 'text as data' or quantitative text analysis (Grimmer & Stewart, 2013; Welbers et al., 2017), which has lately become a mainstay in computational social sciences (Lazer et al., 2021) in fields ranging from sociology to communication and political science.

Text written as social media posts, however, requires extensive preprocessing and conversion to data formats that are compatible with the required analytical tools. Below, we cover the preparatory steps required before text data can be analyzed.

## Convert to Tidytext

Tidytext (Silge & Robinson, 2017) has emerged as one of the preferred formats that facilitate text analysis in R. Text once converted to the tidytext format contains one observation (in this case, one word) per row. At the same time, each row retains information about the source document (in this case, tweet metadata such as user name and time stamp) while the ordering of the rows mirrors the order in which the words originally appeared. The text data is then considered *tidy*, in the sense of being compatible with the rest of the tools and packages from the R *tidyverse*. The major advantage being that now the entire gamut of regular R functions can be brought to bear on our tweets text data.

## Tokenize

Quantitative text analysis extracts information from statistical distributions of meaningful units of text in a collection of documents. Tokenization refers to the process by which the text to be analyzed is split into such units (referred to as tokens), which are most commonly just single words but can also be characters, n-grams, specific regex patterns, or full sentences.

In step 1 in the code below, we use Tidytext's *unnest_tokens()* function to both tidy and tokenize the tweets text at once, while retaining the *screen_name* and *created_at* information with each token. We use the inbuilt 'tweets' tokenizer, which tokenizes the text as single 'words' but is specialized for dealing with text from social media, messaging services and online forums, as it preserves usernames, hashtags, and URLs.

## Remove stop-words and clean the text

Automatic text analysis often makes use of the frequency of words as a feature to mine information from text. Words that appear very often or ones that are unique to some documents might signify different aspects of meaning inherent in text. However, many common words such as ‘the’, ‘and’, ‘is’ appear with very high frequency but carry little information on their own (termed Stopwords) and are hence filtered out before analyzing text data. The *tidytext* package includes English stop words from three different lexicons as a single data frame called *stop_words*. Since we have already converted our data to the tidy format, we can apply the function *anti_join()* from the *dplyr* package in tidyverse to remove the stopwords. *anti_join* keeps only the rows from *tidy_tweets* that don’t have a match in *stop_words*.

We can also define a custom stop list (of stop words specific to our data that we know to be irrelevant to the analysis) as a character vector and remove them using *anti_join*. Using this, we remove the remnants of URLs such as “https” or “t.co” or ‘&amp;’ as well as other words that are not meaningful to our analysis such as ‘rt’ (short for retweet). A custom stop list can also be used to remove other words that are judged to be redundant and not useful for the question at hand.

There are several other ways in which text needs to be preprocessed. It is a common practice to change all letters to the lower case (otherwise People and people would count as separate words). Also, punctuation is commonly removed. Fortunately, the *unnest_tokens* function in *tidytext* that we used took care of both these steps already. Note that, changing all text to lower case will not preserve URLs if they are an element of interest in the analysis, which is not the case here. In any case, with strip_url set to TRUE, we are also using unnest_tokens to remove web links as they are not of interest here. Additionally, we remove whitespaces from the data as otherwise they seem just like any other ‘word’ to an algorithm and complicate the analysis.

```
#1# One Step tidytext - Tokenize, Remove punctuation, and Convert to
lowercase

tidy_tweets <- tweets %>%
  select(screen_name, created_at, text) %>%
  unnest_tokens(word, text, token = "tweets",
                strip_url = TRUE,
                strip_punct = TRUE,
                to_lower = TRUE)

#2# Remove stop-words and other cleaning steps

#Remove stop words
data("stop_words")
tidy_tweets <- tidy_tweets %>%
  anti_join(stop_words)

#Define and remove custom stop words
custom_stop_words <- tibble(
  word = c("https", "t.co", "amp", "rt"))
tidy_tweets <- tidy_tweets %>%
  anti_join(custom_stop_words)

#Remove Whitespaces
tidy_tweets$word <- gsub("\\s+","",tidy_tweets$word)

#3# Plot comparative language use in tweets

tidy_tweets %>%
  #Keep only the data from selected accounts
  filter(screen_name == c("BorisJohnson",
                          "Keir_Starmer")) %>%
  #Count word frequencies
  count(screen_name, word) %>%
  group_by(word) %>%
  #Keep only words with total occurrence in both > 100
  filter(sum(n) > 100) %>%
  ungroup() %>%
  #Split the screen_name variable into separate columns
  pivot_wider(names_from = "screen_name",
              values_from = "n", values_fill = 0) %>%
  ggplot(aes(BorisJohnson, Keir_Starmer)) +
  #Plot the 45 degree line
  geom_abline(color = "gray", size = 0.3, lty = 2) +
  #Plot data as points
  geom_jitter(alpha = 0.2, size = 1) +
  #Plot data as text
  geom_text(aes(label = word), check_overlap = TRUE,
            vjust = 1, size = 3) +
  coord_equal() +
  theme_few()
```

## Boris v. Keir: Comparative Language Use in Tweets

Now, that we have cleaned our data and have converted it to the tidy format, we have access to the whole range of tools and analytical possibilities available in *tidyverse* and generally in R. For instance, we can manipulate the converted data frame *tidy_tweets* using *dplyr* and plot it using *ggplot2*. We can also further preprocess this data frame to a document-term matrix or tf-idf scores which might serve as input to other advanced text analysis methods (see Silge & Robinson, 2017) such as topic modeling.

To stay within the scope of this tutorial, we will do a comparative analysis of language used in the tweets by the current Prime Minister Boris Johnson (Conservative Party; Center-Right) and the leader of the opposition Keir Starmer (Labour Party; Center-Left). Specifically, we will examine comparative word counts in tweets by each leader to see which words were used more often in political communication and also which ones were common to both or distinct to each. We do this by plotting word counts (Fig. 1) for each leader (with Boris Johnson and Keir Starmer on the x-axis and y-axis, respectively) for words that have been used at least 100 times in total. Words that turn up on or close to the 45º line (that passes through the point (0,0) and has a slope of 1) would be the words that were used nearly equally by both, while words that turn up closer to either axis would be the words used somewhat exclusively by one leader.

This simple visualization (Fig. 1) hints at several interesting features of contemporary British politics. Close to the 45º line, common words such as British, world and time are used equally, and people and country roughly equally by both leaders.  Other words are somewhat exclusive to each. Boris Johnson tends to tweet a lot about Brexit and #getbrexitdone, but more tellingly, Keir Starmer nearly never brings it up. However, true to form, the center-left Labour leader tends to tweet a lot more about jobs, government and workers, which are surprisingly scarce in tweets by Prime Minister Johnson.

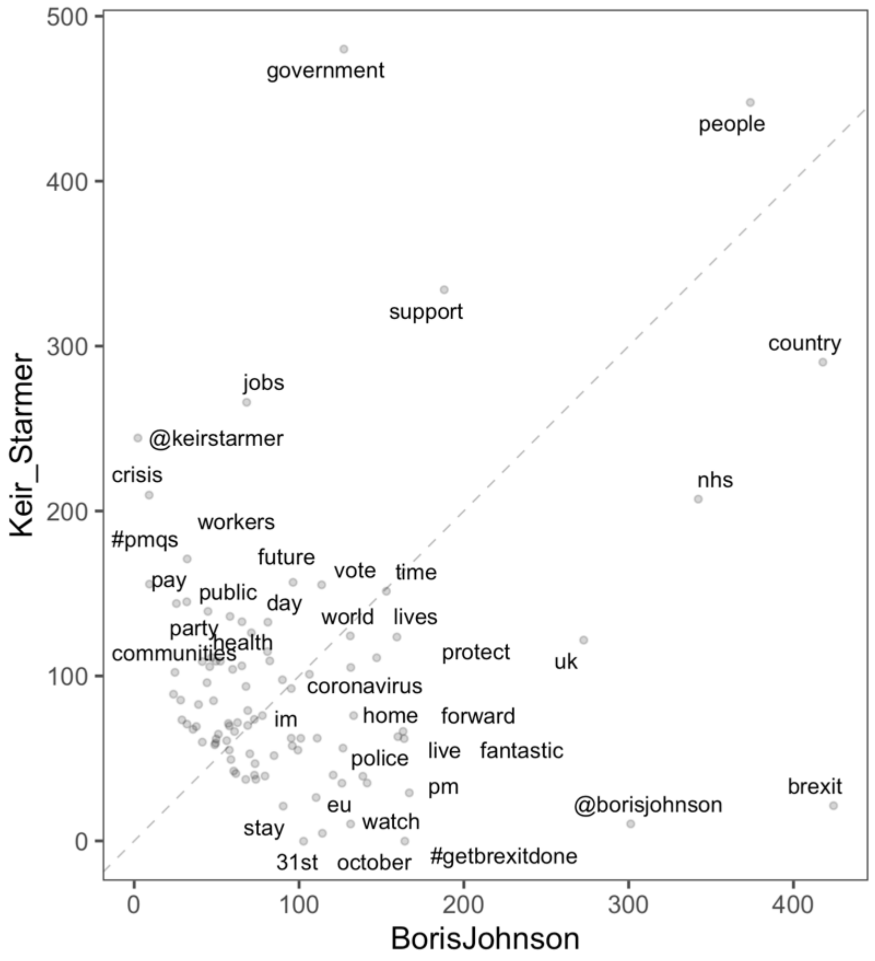

***Figure 1. Comparative language use in tweets from the twitter accounts of Boris Johnson and Keir Starmer****. Closeness of a word to either axis denotes more frequent use by the corresponding leader and less by the other. Note that while Mr. Johnson tweets a lot about Brexit, Mr. Starmer is nearly silent on the issue. The Labour leader instead tweets a lot more about jobs and workers, words relatively scarce in tweets by Mr. Johnson.*

## Sentiment Analysis

Sentiment analysis is a special case of dictionary-based quantitative text analysis approaches that involve tagging words in the text being analyzed with pre-defined meaning or values using a dictionary. It can be a useful way to quantify affective content, attitudes and opinions inherent to any textual data, especially to measure change in their values across time or categories. See, for example, Tumasjan et al. (2011)'s use of sentiment analysis of Twitter data for election forecasts.

Sentiment dictionaries or lexicons contain lists of all words with affective content and an evaluation of their valence (e.g., as sentiments: positive/negative; or emotions: anger/disgust/joy/trust/surprise etc.). Needless to say, the quality of sentiment analysis is dependent on the quality and suitability of the dictionary used. The tidytext package includes several of these sentiment dictionaries. In this analysis, we will utilize the 'bing' dictionary (Hu and Liu, 2004), which assigns all sentiment-laden words in English to either a positive or negative category. We can load specific dictionaries using the *get_sentiments()* function. The dictionary thus loaded is a data frame with one word and its associated sentiment per row. Here, the advantages of having converted the twitter text to the tidy data format become apparent – we can now juxtapose our *tidy_tweets* data set with the bing lexicon using the *dplyr* function *inner_join()*, which returns only the words that are present in both. It must be noted that the absolute number of positive and negative words thus inferred from a given set of texts is interpretable only in a relative sense and when juxtaposed to comparable texts in the same domain.

```
#Sentiment Analysis#
#Comparing Sentiment in UK leaders' tweets

tidy_tweets %>%
  inner_join(get_sentiments("bing")) %>%
  ggplot() +
  geom_bar(mapping = aes(x = screen_name, fill = sentiment),
           position = "fill", width = 0.6) +
  labs(x="Twitter Handle",
       y = "Proportional Sentiment Score") +
  scale_fill_grey() +
  theme_few()
```

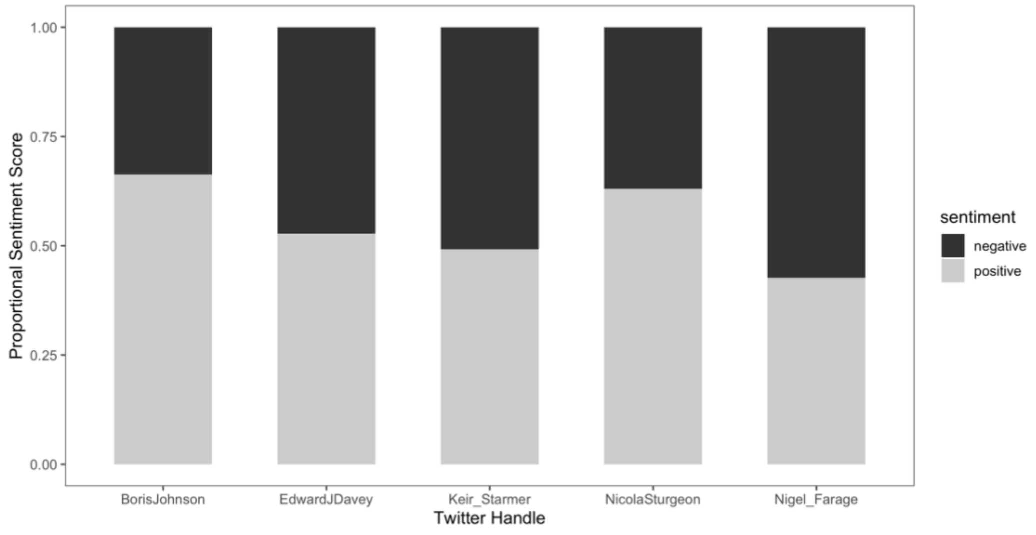

***Figure 2. Proportional Sentiment Score for tweets by each of the five leaders****. It can be noted that the leaders currently in power (Boris Johnson and Nicola Sturgeon) use more positive language in their tweets as compared to the ones who are in opposition.*

The output has the list of all the affective words according to the '*bing'* dictionary from each leader's tweets along with a positive/negative categorization. Finally, we plot a count of positive/negative words as a proportion of all affective words in the tweets of each leader (Fig. 2). Interestingly, it can be noted that the two leaders who use more positive language in their tweets – Boris Johnson and Nicola Sturgeon – are the ones currently in power, while the leaders currently out of power, in line with their democratic role as opposition and critics of the incumbent governments, use relatively more negative language.

## Mapping Twitter Data

In this section, we will describe how to extract location information from Twitter data and draw a geographical map depicting those locations. Geospatial mapping of Twitter data can be a useful tool in understanding localized trends or reactions to localized events (Mai & Hranac, 2013) and even disaster management (Singh et al., 2019). A notable caveat here is that since users only seldom provide location information, only a small fraction of any tweets corpus can be used for mapping – Sloan et al. (2013), for instance, found that in a sample of 113 million tweets only 0.85% (966,082) were geo-tagged while Sloan and Morgan (2015) showed that the estimate varies depending on the demographic characteristics of Twitter users.

There are four sources of geographical information in Twitter data. First, the tweet text itself might refer to specific places or local events. Second, the location entered by users in their account profile and available in the variable *location* in the data we extracted. Third, sometimes Twitter users add a place to their tweet to signify where a specific tweet is being sent from. This information is available in the variable *place_full_name*. Fourth, the precise coordinates added by GPS-enabled devices in geotagged tweets. The *rtweet* package provides a useful *lat_lng()* function that parses tweets data and appends single-point latitude and longitude variables to it.

The code below shows how to use geotagged tweets to draw geographical maps with the location they were tweeted from. First, we extract the numerical geographical coordinates (as latitudes and longitudes) from the Twitter data using the *lat_lng()* function from *rtweet* and remove the tweets which don't have this information. Second, we extract geo data from the *maps* package and convert it into a data frame compatible with *ggplot2* and use that to draw an empty world map using *ggplot2 geom_polygon*. This map can then be limited to any area of interest, which in this case is the United Kingdom. Finally, we overlay this map with location coordinates extracted in step 1 as points on the map using *geom_point()*. Figure 3 shows the resulting map of UK overlain with locations of 85 geotagged tweets found in our data.

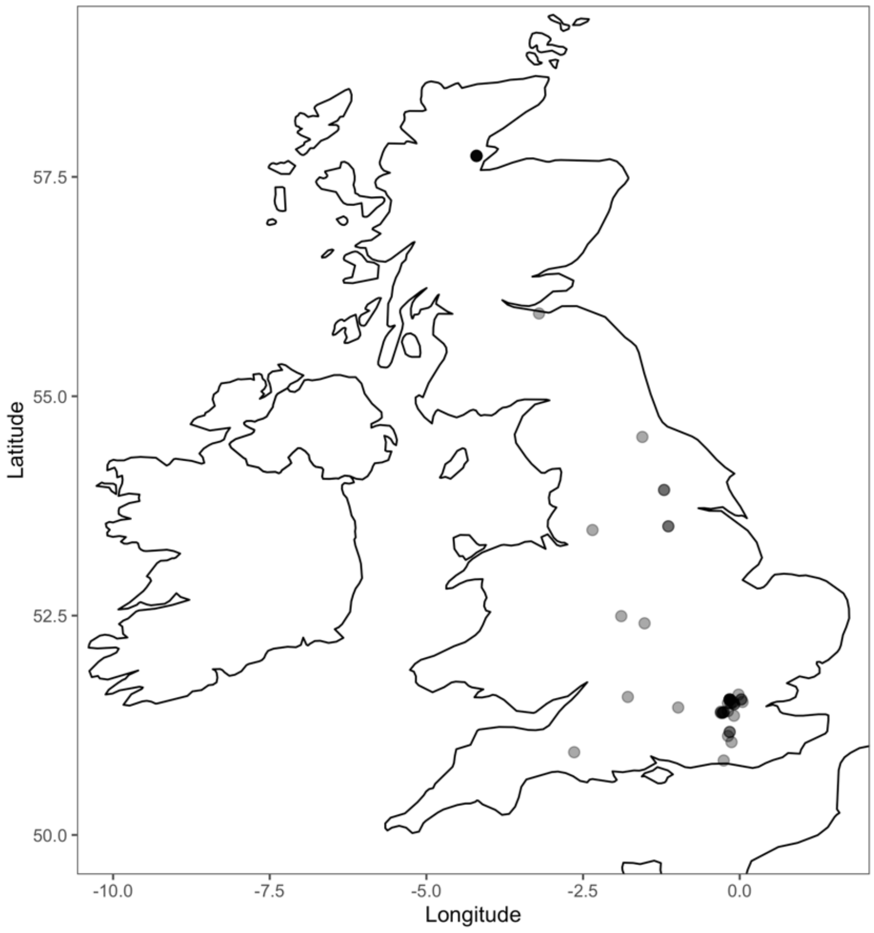

***Figure 3. Mapping Twitter data**. Map of the UK overlain with locations from geotagged tweets found in our data.*

```
library(maps)# v3.3.0 Requires R (>= 3.0.0)

#1# Append parsed Twitter data with geographical coordinates
# and remove rows with missing coordinates
tweets_coords <- lat_lng(tweets)
tweets_coords <- na.omit(tweets_coords[, c("lat", "lng")])

#2# Convert geo data from the maps package into
#a ggplot2 compatible dataframe
worldmap_df = map_data('world')
```

```
#3# Plot the map using ggplot2
ggplot() +
  #Draw a world map with data from worldmap_df
  geom_polygon(data = worldmap_df,
               aes(x = long, y = lat, group = group),
               fill = 'white', color = 'black') +
  #Specify a bounding box to limit the map to the UK
  coord_fixed(ratio = 1.4, xlim = c(-10,1.5), ylim = c(50, 59)) +
  #Overlay locations from twitter data
  geom_point(data = tweets_coords,
             aes(lng, lat), color = "black",
             alpha = 0.4, size = 2,
             ) +
  labs(x = "Longitude", y = "Latitude") +
  theme_few()
```

## Twitter Data as a Time Series-like Object

Earlier we examined the content of the tweets by itself, we will now combine it with the time-stamp variable *created_at* already present in the data in order to examine the distribution of tweets (and the contents therein) over time. Such a sequence of data points taken at successive equally spaced points in time is termed as a time series. Time series analysis is a powerful method used most commonly for event detection and forecasting in a wide array of fields. For instance, Bollen et al. (2011) measured daily variations in public mood from the text content of large-scale Twitter feeds and found a statistically significant correlation to daily changes in the stock market (Dow Jones Industrial Average closing values).

We will, however, restrict ourselves to converting tweets data to a time-series like object and its visualization. We will first extract some predefined words of interest (nhs, vote, coronavirus and brexit) in the tweets text from all five politicians pooled together from the *tidy_tweets* data frame, which is already in tidy format with one word per row. Since it also includes a *created_at* column with a time-stamp for each word, we can bin the words of interest into weekly segments in order to show how the mentions of these words by the five major British politicians (as a proxy for trends in British political communication) varied over time. For this, we will use the *ts_data* function from the *rtweet* package.

```
#1# Predefine words of interest and
#Convert to time-series like object
uktrends <- tidy_tweets %>%
  filter(word %in% c("nhs", "vote", "coronavirus", "brexit")) %>%
  group_by(word) %>%
  ts_data("weeks")

#2# Plot a Spaghetti Chart
spaghetti <- uktrends %>%
  mutate(word2=word)

uktrends %>%
  ggplot( aes(x=time, y=n)) +
  geom_line(data=spaghetti %>% select(-word), aes(group=word2),
            color="grey50", size=0.3, alpha=0.5) +
  geom_line(aes(color=word), color = "black", size = 0.3) +
  facet_wrap(~word, nrow = 4) +
  labs(x="Time", y = "Weekly Mentions by British Politicians")+
  theme_few()
```

Figure 4 shows how the weekly frequency of mentions of these words in British political communication (as represented *in the corpus of tweets pooled across the top 5 British politicians*.) varied over time. Some trends are clearly consistent with real world events. Mentions of ‘Brexit’ peaked before the actual date of Britain officially leaving the EU at the start of 2020 while mentions of ‘coronavirus’ peaked in the spring of 2020. If one disregards the predictable peak in mentions of ‘nhs’ (National Health Service) with the first peak in ‘coronavirus’, there is a hint of a seasonal trend in the frequency of its mentions – seen in the several ‘nhs’ peaks right before the two major elections in the UK in this time period (marked itself by the peaks in mentions of the word ‘vote’).

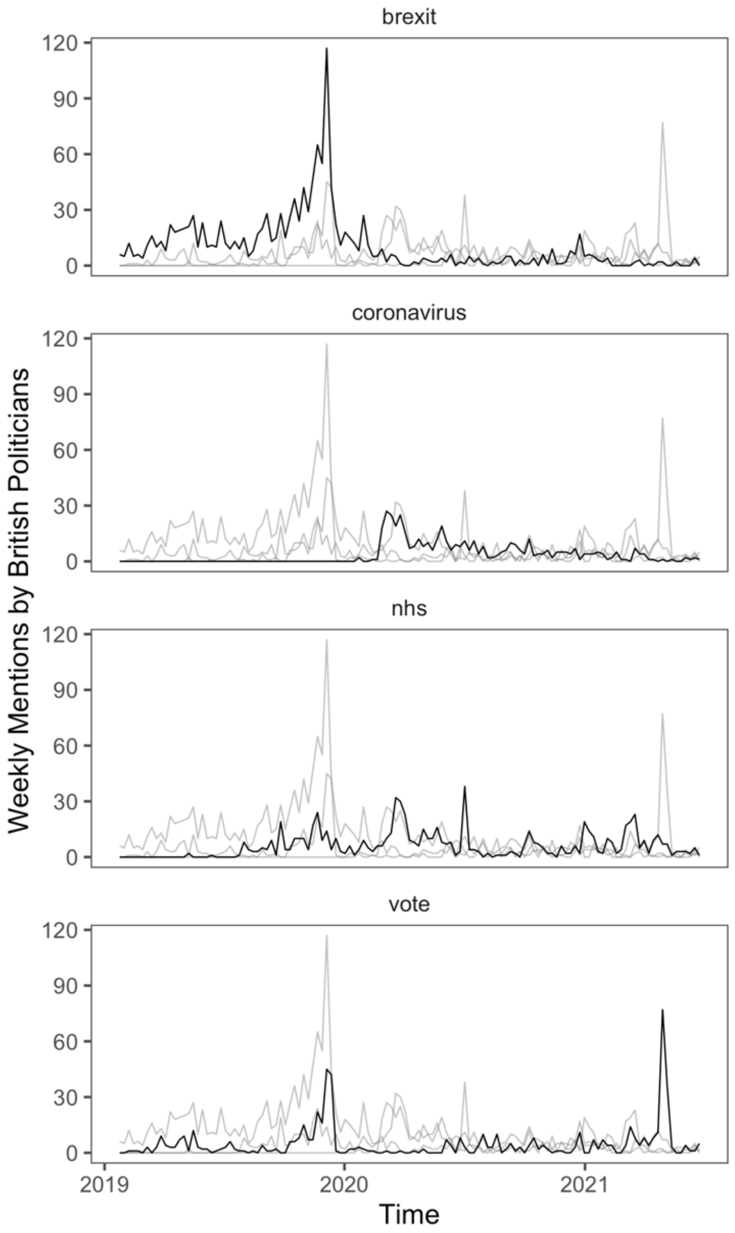

***Figure 4. Trends over time***. *Variation over time in weekly mentions of specific words in the corpus of tweets pooled across the top 5 British politicians. Each highlighted line represents popularity over time of the word, labelled on top of the subplot, in British political communication in general (as represented in tweets by the top leaders pooled together). Please note that all four subplots show the same figure except that a different line, representing a specific word, is highlighted in each to facilitate comparison.*

Dealing with time series data always implies looking for general trends, however one should be mindful that some of the trends might be specific to the mechanics of Twitter use. For example, if the overall number of active users in a sample has increased over time, then it would be no surprise that tweets sent later in time will get more retweets. Or for instance, differences in user activity on Twitter between weekdays and weekends might influence seasonal trends, and this temporal variation might be different for different regions and topics.

## Conclusion and Caveats

A solid grasp of analytical tools is an essential foundation for social media research and R is a particularly good way to build that. In this chapter, we demonstrated how to use R to collect, analyze and visualize data from Twitter. We also introduced the reader to a wide range of R packages and preliminary analysis they facilitate in a variety of domains. We collected tweets from 5 different British politicians, analyzed them for what they were about (quantitative text analysis), the sentiment therein (sentiment analysis), changes in trends over time (time series analysis), and drew a map of their locations (geospatial mapping) – each with a handful of code. The primary purpose of these analyses was to demonstrate the respective techniques to a beginner and not validation and generalization of the findings, which we would like to stress are only offered here as interesting hints. Having said that, the presented methods and code themselves are ready and suitable to be applied as part of more extensive investigations.

We hope the power and flexibility that R provides in analyzing social media data is amply clear to anyone who’s worked through the code for themselves. The well-established community of researchers, developers and users behind R and the associated packages is amply present on a variety of websites and forums such as stackoverflow.com, community.rstudio.com and rbloggers.com and functions as a reliable source of help and guidance, should one run into a problem at any level.

We would be remiss not to remind the reader of a number of caveats that apply to the analyses presented here. Ethical and legal issues, especially when related to consent and privacy of individual users who are not public figures, can often be convoluted and require careful deliberation. Also, Twitter data collected at different points in time is often not reproducible, which could be because of sampling by Twitter, or because tweets could have been deleted or entire accounts deactivated by users. Lastly, it should be noted that data collected from Twitter (or other social media platforms), rarely represents a representative sample from the general population (Barberá & Rivero, 2015) and is most informative either when studying social media itself or when used as a supplementary rather than a principal data source.

## References

Barberá, P., & Rivero, G. (2015). Understanding the Political Representativeness of Twitter Users. *Social Science Computer Review*, *33*(6), 712–729. https://doi.org/10.1177/0894439314558836

Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, *2*(1), 1–8. https://doi.org/10.1016/j.jocs.2010.12.007

*Developer terms - More about restricted uses of the Twitter APIs* (2021). Retrieved September 3, 2021, from https://developer.twitter.com/en/developer-terms/more-on-restricted-use-cases

dplyr documentation—Top_n—Select top (or bottom) n rows (by value) (2021). Retrieved September 3, 2021, from https://dplyr.tidyverse.org/reference/top_n.html

Grimmer, J., & Stewart, B. M. (2013). Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts. *Political Analysis*, *21*(3), 267–297. https://doi.org/10.1093/pan/mps028

Grinberg, N., Joseph, K., Friedland, L., Swire-Thompson, B., & Lazer, D. (2019). Fake news on Twitter during the 2016 U.S. presidential election. *Science*, *363*(6425), 374–378. https://doi.org/10.1126/science.aau2706

*Guide to the future of the Twitter API*. (2021). Retrieved June 30, 2021, from https://developer.twitter.com/en/products/twitter-api/early-access/guide

Hegelich, S. (2020). Facebook needs to share more with researchers. *Nature*, *579*(7800), 473–473.

Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews. *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 168–177. https://doi.org/10.1145/1014052.1014073

Kearney, M. W. (2019). rtweet: Collecting and analyzing Twitter data. *Journal of Open Source Software*, *4*(42), 1829. https://doi.org/10.21105/joss.01829

Lazer, D., Hargittai, E., Freelon, D., Gonzalez-Bailon, S., Munger, K., Ognyanova, K., & Radford, J. (2021). Meaningful measures of human society in the twenty-first century. *Nature*, *595*(7866), 189–196. https://doi.org/10.1038/s41586-021-03660-7

Mai, E., & Hranac, R. (2013). *Twitter Interactions as a Data Source for Transportation Incidents* (No. 13–1636). Article 13–1636. Transportation Research Board 92nd Annual MeetingTransportation Research Board. https://trid.trb.org/view/1241097

Muenchen, R. A. (2019). *The Popularity of Data Science Software | r4stats.com*. Retrieved June 30, 2021, from http://r4stats.com/articles/popularity/

Papakyriakopoulos, O., Serrano, J. C. M., & Hegelich, S. (2020). Political communication on social media: A tale of hyperactive users and bias in recommender systems. *Online Social Networks and Media*, *15*, 100058. https://doi.org/10.1016/j.osnem.2019.100058

Seymour, R. (2020). *The twittering machine*. Verso.

Shahrezaye, M., Papakyriakopoulos, O., Serrano, J. M., & Hegelich, S. (2019). Estimating the Political Orientation of Twitter Users in Homophilic Networks. *AAAI Spring Symposium: Interpretable AI for Well-Being*.

Silge, J., & Robinson, D. (2017). *Text Mining with R: A Tidy Approach* (1st edition). O'Reilly Media.

Singh, J. P., Dwivedi, Y. K., Rana, N. P., Kumar, A., & Kapoor, K. K. (2019). Event classification and location prediction from tweets during disasters. *Annals of Operations Research*, *283*(1), 737–757. https://doi.org/10.1007/s10479-017-2522-3

Sloan, L., Morgan, J., Housley, W., Williams, M., Edwards, A., Burnap, P., & Rana, O. (2013). Knowing the Tweeters: Deriving Sociologically Relevant Demographics from Twitter. *Sociological Research Online*, *18*(3), 74–84. https://doi.org/10.5153/sro.3001

Sloan, L., & Morgan, J. (2015). Who Tweets with Their Location? Understanding the Relationship between Demographic Characteristics and the Use of Geoservices and Geotagging on Twitter. *PLoS ONE*, *10*(11), e0142209. https://doi.org/10.1371/journal.pone.0142209

Tornes, A., & Trujillo, L. (2021). *Enabling the future of academic research with the Twitter API*. Retrieved September 3, 2021, from https://blog.twitter.com/developer/en_us/topics/tools/2021/enabling-the-future-of-academic-research-with-the-twitter-api

Tumasjan, A., Sprenger, T. O., Sandner, P. G., & Welpe, I. M. (2011). Election Forecasts With Twitter: How 140 Characters Reflect the Political Landscape. *Social Science Computer Review*, *29*(4), 402–418. https://doi.org/10.1177/0894439310386557

Twitter Inc. (2021). *Twitter Q1 2021 Letter to Shareholders*. Retrieved June 30, 2021, from https://investor.twitterinc.com/home/default.aspx

Welbers, K., Van Atteveldt, W., & Benoit, K. (2017). Text Analysis in R. *Communication Methods and Measures*, *11*(4), 245–265. https://doi.org/10.1080/19312458.2017.1387238

Wickham, H., & Grolemund, G. (2016). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media, Inc.