# Lab 1: Collecting Social Media Data and Visualizing Social Media Networks

Social Media and Web Analytics

Last updated: 09 April, 2021

## Learning Goals

By the end of this tutorial you will be able to:

- Use the `rtweet` package to access the Twitter API and download tweets that mention a set of words or hashtags.
- Use the `rtweet` package to retrive the friends and followers of a Twitter user.
- Collect recent tweets from a particular Twitter user.
- Construct an edge list that connects Twitter users based on mentions, replies and/or retweets
- Plot networks of Twitter users using `tidygraph` and `ggraph`

## Instructions to Students

These tutorials are **not graded**, but we encourage you to invest time and effort into working through them from start to finish. Add your solutions to the `lab-01_answer.Rmd` file as you work through the exercises so that you have a record of the work you have done.

Obtain a copy of this document and the answer document using Git. To clone a copy of this repository to your own PC:

```
git clone https://github.com/tisem-digital-marketing/smwa-lab-01.git
```

Once you have cloned the files, open the cloned repository in RStudio as a new project and work through the questions.

The goal of the tutorials is to explore how to "do" the technical side of social media analytics. Use this as an opportunity to push your limits and develop new skills. When you are uncertain or do not know what to do next - ask questions of your peers and the instructors on the classes Slack channel `#lab-01-2021`.

## Exercise 1: Collecting Twitter Data with `rtweet`

You will need to use the following `R` libraries throughout this exercise:

```
library(rtweet)
```

In this exercise you will access the Twitter API programmatically to collect recent tweets related to covid-19 posted from the Netherlands. The goal of this exercise is to get comfortable with the `rtweet` package so you can harvest Twitter data on topics and users you are interested in for the future.

**Watch out!**

Note that because you will likely work on this tutorial at a different time to some of your peers, the data that you collect will likely differ from person to person - this is totally OK.

1. Search for the most recent 500 tweets that use the hashtag `#covid19`. Include retweets in what is returned.

```
covid_tweets <- # YOUR CODE HERE
```

**solution**

```
covid_tweets <- search_tweets(
  "#covid19",
  n = 500,
  include_rts = TRUE
)
```

2. Adjust your search to return 500 tweets that use the hashtag `#covid19` but do not include retweets, or replies.

```
covid_no_rt <- # YOUR CODE HERE
```

**solution**

```
covid_no_rt <- search_tweets(
  "#covid19",
  n = 500,
  include_rts = FALSE,
  `-filter` = "replies",
)
```

For the remainder of the exercise, we will always include retweets and replies.

3. Now, let's geo-restrict the tweets we collect. Collect 500 tweets using the hashtag `#covid19` that lie within a 50 km radius of Rotterdam. HINT: The geo-cordinates of Rotterdam are `'51.9244,4.4777,50km'`.[1]

```
rdam_geocode <- '51.9244,4.4777,50km'
covid_rdam <- # YOUR CODE HERE
```

**solution**

```
covid_rdam <- search_tweets(
  "#covid19",
  n = 500,
  geocode = rdam_geocode,
  include_rts = TRUE
)
```

4. Adjust your search above to only include tweets written in Dutch. Try to extract up to 50,000 tweets (i.e. the *more* than the nominal limit from the API).

```
covid_rdam_nl <- # YOUR CODE HERE
```

**solution**

[1] Hat-tip to Ana Martinovici at Rotterdam School of Management for knowing and sharing these!

```
covid_rdam_nl <- search_tweets(
  "#covid19",
  n = 50000,
  geocode = rdam_geocode,
  lang = "nl",
  include_rts = TRUE,
  retryonratelimit = TRUE
)
```

5. The RIVM seem to be the most active Government Department tweeting about health matters. Collect the 3200 most recent tweets from them (that's the upper limit Twitter will let you get for free).

```
rivm_tweets <- # YOUR CODE HERE
```

solution

```
rivm_tweets <- get_timeline("rivm", n = 3200)
```

6. Collect the `user_ids` for everyone that RIVM follows.

```
rivm_friends <- # YOUR CODE HERE
```

solution

```
rivm_friends <- get_friends("rivm")
```

7. Collect the `user_ids` for 500 users who follow the RIVM.

```
rivm_follow <- # YOUR CODE HERE
```

solution

```
rivm_follow <- get_followers("rivm", n = 500)
```

Take a random sample of 5 people that follow the RIVM from the list above as follows:

```
smple <-
  rivm_friends %>%
  sample_n(5) %>%
  select(user_id) %>%
  tibble::deframe() # this converts a dataframe to a vector
```

8. Collect the user information for these 5 accounts.

```
rivm_users <- # YOUR CODE HERE
```

solution

```
rivm_users <- lookup_users(rivm_users)
```

9. Get the most recent tweets from these 5 accounts.

```
user_tweets <- # YOUR CODE HERE
```

**solution**

```
user_tweets <- tweets_data(rivm_users)
```

10. (Unrelated to above) Collect a sample for 30 seconds of all tweets within 10km of Rotterdam.

```
rdam_geocode <- '51.9244,4.4777,10km'
rdam_stream <- # YOUR CODE HERE
```

solution

```
rdam_stream <-stream_tweets(
  "",
  geocode = rdam_geocode
  )
```

## Exercise 2: Plotting Social Media Networks

You will need to use the following `R` libraries throughout this exercise:

```
library(readr)
library(tidygraph)
library(ggraph)
library(dplyr)
library(tidyr)
library(tibble)
```

In this exercise you will work with some existing Twitter data so that you can check your results and understanding on a static dataset. The data you will use is a collection of tweets that all have the hashtag `#rstats` in their text.

The data are collected from 2018, and can be downloaded from the internet:

```
url <- "https://bit.ly/3r8Gu4M"
# where to save data
out_file = "data/rstats_tweets.rds"
# download it!
download.file(url, destfile = out_file)
```

The data that you downloaded are an `.rds` file, so you can load them with the `read_rds` function from the `readr` library:

```
tweets <- read_rds(out_file)
```

Your goal will be to construct a network graph that visualizes the connections between Twitter users. For this exercise you are interested in connections between Twitter users who reply to each others tweets. That is, two users are connected if user A has replied to user B's tweet or vice versa.

Before you work through the guided exercise, we recommend that you take some time to look at the data and understand it's basic structure. There are a lot of column names, and you will want to understant what is in them.

Now, let's begin the analysis:

1. Create a new dataset that only includes tweets that contain replies:

```
tweets_replies <- # YOUR CODE HERE
```

solution

```
tweets_replies <-
  tweets %>%
  drop_na(reply_to_screen_name)
```

2. Further reduce the size of the data by dropping the columns you will not need. Your new dataset should only include the columns named `screen_name` and `reply_to_screen_name`. Rename these columns to `from` and `to`.

```
tweets_replies <- # YOUR CODE HERE
```

**solution**

```
tweets_replies <-
  tweets_replies %>%
  select(from = screen_name, to = reply_to_screen_name)
```

3. When a user on Twitter writes a long series of tweets about the same topic, they often connect multiple tweets together by replying to their own previous tweet to chain their posts together. Remove these replies from the data.

```
edgelist <- # YOUR CODE HERE
```

**solution**

```
edgelist <-
  tweet_replies %>%
  filter(from != to)
```

4. Now, you are going to trim down the size of the edgelist. You will do this mainly so that your computer won't freeze when it comes time to plot the network. Proceed in two steps:

(a) Create a dataset that counts the number of times a user replies to anyone in the data. Keep only users who have replied less than 50 times.

```
interactions_sent <- # YOUR CODE HERE
```

**solution**

```
interactions_sent <- edgelist %>%
  count(from) %>%
  arrange(desc(n)) %>%  # this step is unnecessary
  filter(n > 50)
```

(b) Update your the edgelist so that only users who have engaged in at least 50 replies are included.

```
edgelist <- # YOUR CODE HERE
```

**solution**

```
edgelist <- edgelist %>%
  # the first of the two lines below filters to include only repliers
  # in the interactions_sent data frame
  # the second line does the same, for reply recievers
  filter(from %in% interactions_sent$from,
         to %in% interactions_sent$from)
```

5. Convert your `data.frame` containin all the edges to a tidygraph object.

```
tg <- # YOUR CODE HERE
```

**solution**

```
tg <- as_tbl_graph(edgelist)
```

6. Plot the network. Use the layout `kk` in your solution.

```
# YOUR CODE HERE
```

**solution**

```
tg %>%
  ggraph(layout = "kk") +
  # this adds the points to the graph
  geom_node_point() +
  # this adds the links, or the edges;
  # alpha = .2 makes it so that the lines are partially transparent
  geom_edge_link(alpha = .2) +
  # this last line of code adds a ggplot2 theme suitable for network graphs
  theme_graph()
```

7. Explore different layouts and find the one you think works best visually. You could explore the choice "stress" or any of the following: "dh", "drl", "fr", "gem", "graphopt", "lgl", "mds", "sugiyama", "bipartite", "star" or "tree".

8. The plot you produced weighted the edges by the frequency in which two nodes had replied to each other (it does this implicitly because the same edge occured many times in the edgelist). Prevent this from happening by . You will have to re-run parts of you code to produce the new graph.

```
# YOUR CODE HERE
```

**solution**

```
edgelist <- edgelist %>%
  dplyr::distinct() %>%
  # the first of the two lines below filters to include only senders
  # in the interactions_sent data frame
  # the second line does the same, for receivers
  filter(from %in% interactions_sent$from,
         to %in% interactions_sent$from)

tg <- as_tbl_graph(edgelist)

tg %>%
  ggraph(layout = "kk") +
  # this adds the points to the graph
  geom_node_point() +
  # this adds the links, or the edges;
  # alpha = .2 makes it so that the lines are partially transparent
  geom_edge_link(alpha = .2) +
  # this last line of code adds a ggplot2 theme suitable for network graphs
  theme_graph()
```

9. You can add color to the plot that you just created. Color (and re-size) the nodes based on their influence as measured by `centrality_authority`.

```
# YOUR CODE HERE
```

solution

```
tg %>%
  # this calculates the centrality of each individual using the built-in
  # centrality_authority() function
  mutate(centrality = centrality_authority()) %>%
  ggraph(layout = "kk") +
  geom_node_point(aes(size = centrality, color = centrality)) +
  # this line colors the points based upon their centrality
  scale_color_continuous(guide = 'legend') +
  geom_edge_link(alpha = 0.05) +
  theme_graph()
```

10. Save the last plot you created as 'rstats-replies.pdf'.

```
# YOUR CODE HERE
```

solution

```
ggsave("rstats-replies.pdf")
```

## Exercise 3: Putting It All Together (Optional, Unguided)

Now that you have explored collecting social media data from Twitter and plotting social networks constructed from Twitter data, you can combine these two steps to build a network graph from your own data.

We recommend doing the following:

1. Use the `rtweet` package to collect data from Twitter for a keyword or hashtag that is of interest to you. Include retweets, but feel free to play around with location constraints.
2. Plot the retweet and mentions networks from your data. Can you find a way to plot them side by side? Are there interesting dynamics in how the network grows over time (and can you visualize it)?
3. Upload the final images to the Slack channel `#lab-01-student-viz`. We are keen to see the network patterns that you uncover.[2]

## License



## Suggested Citation

Deer, Lachlan and de With, Hendrik. 2021. Social Media and Web Analytics: Lab 1 - Collecting Social Media Data and Visualizing Social Media Networks. Tilburg University. url = "https://github.com/tisem-digital-marketing/smwa-lab-01"

[2]We are also interested in the underlying data. We will prompt you to upload the data sets and underlying scripts you have created to a online repository that all students can look at and explore.