Collecting Twitter Data for Research

Francesca Giannetti

09/29/2020

About

This workshop presents a few of the best and most approachable tools for collecting Twitter data for research. We'll also address the kinds of information we can glean from social media data as well as several important factors and limitations to consider when doing social media research.

Prerequisites

For this workshop, you'll need a Google account (Gmail or Scarletmail) and a Twitter account. Freate and/or log into both now.

- Google
- Twitter

The hands-on portion of this workshop uses Martin Hawksey's TAGS, which is a Google Sheets add-on, and Michael Kearney's rtweet, an R package, for interacting with Twitter's APIs. Both are widely used in research, as you can tell by searching for references in Google Scholar.[1] We'll discuss briefly the strengths and limitations of each tool.

Characteristics of Twitter Data

It is often held that social media data provide critical and unique insights into human behavior. And social media platforms offer a relatively economical, quick, and targeted way of collecting data, especially as compared to some other sources, e.g. polls, surveys, demographic and economic data. And yet each social media platform presents with structural biases that are important to consider.[2]

Twitter data have limitations pertaining to representativeness. From the Pew Research Center's most recent Social Media Fact Sheet:

- Just over a fifth of Americans use Twitter.
- Younger Americans are much more likely than older Americans to be on Twitter.
- Almost a third of Twitter users in the U.S. are college educated and middle- or upper-middle class.
- Roughly equivalent percentages of Twitter users are Hispanic (25%), Black (24%), and white (21%). Compare this to Pinterest (mostly white), for example.
- Twitter users are more likely to live in urban (26%) or suburban (22%) areas than rural (13%) ones.

Additionally, Twitter is not as global as is often thought. The top ten countries by user count are as follows (as of July 2020). Twitter is blocked in China, although users there may circumvent the block through the use of VPNs.

[1] You may need to append a `-tweet` to your search for Kearney's rtweet package to eliminate references to an unrelated author named Tweet (!).

[2] Tufekci provides a strong overview of the methodological considerations of using social media data for research in "Big Questions for Social Media Big Data: Representativeness, Validity and Other Methodological Pitfalls".

country	users (in millions)
United States	62.55
Japan	49.1
India	17
Brazil	15.7
United Kingdom	15.25
Turkey	12.7
Saudi Arabia	12
Indonesia	11.2
Mexico	10.4
Philipines	7.75

Source: Statista

Furthermore, the Twitter REST API, which both TAGS and rtweet query, has important limitations to take into account. For starters, one cannot use it to retrieve tweets that are older than 6-9 days, at least not without a premium (paid) account. Consider also the rate limits that Twitter imposes on all of its APIs, which put a cap on how many times you can query the API in a given time window. Both TAGS and rtweet offer ways of managing this issue for you. But if the phenomenon you hope to investigate is large (i.e. relatively high numbers of tweets per minute), the rate limiting will slow down your data collection. Furthermore, none of the Twitter APIs support the creation of complete collections of tweets, and instead use a complex formula to arrive at a representative sample of tweets.

These limitations are important to keep in mind as you frame your research question. For instance, we probably can't make generalizations about the population as a whole based on Twitter data. We can however measure short-term effects, and look for insights relating to the variables we've identified as being implicated in our research question. As a purely practical tip, it may make sense to investigate larger topics that generate lots of activity, rather than smaller topics where the resulting dataset may be a less faithful representation of the subject or phenomenon we want to investigate.

Formulate a Research Question

Let's spend a few minutes elaborating a research idea or question. Try out your search terms at twitter.com/explore to see how viable your topic is. Your question doesn't have to be detailed or complicated. For example, you could ask if the use of the hashtag #shitmystudentswrite is unethical... or if is it just good fun. Adding to that topic, are the tweets with the hashtag #shitacademicssay similar or dissimilar to those with #shitmystudentswrite? (In other words, do academics say things just as silly as their students?) You can use any combination of hashtags, handles, and free text you identify that capture your topic, based on these preliminary findings. The tools we will use introduce a few other search possibilities that are inaccessible via the regular web interface, e.g. simple Boolean logic, to/from specific Twitter accounts, and geolocated tweets, so you might also think about what could be done with these added capabilities.

Some definitions

APIs

Many social media platforms provide access to their platforms in the form of APIs (application programming interfaces). APIs facilitate the work of data collection. APIs have affordances and limitations. They change over time, and sometimes they disappear.

An API is a set of methods for applications to receive requests and send responses. Developers use APIs to build applications using someone else's content. For example, Mark Sample's Shark Girls uses an OCEARCH

API to tweet the locations of two great white sharks. Web APIs send a structured response in JSON or XML format. Most social media APIs send responses as JSON.

Collecting vs Scraping

Collecting/harvesting/gathering data from APIs is not the same thing as web scraping, which would involve going to the website for human users (e.g. https://twitter.com) and scraping the HTML representation of the data. This would be hard and messy and the data would most likely be incomplete.[3]

JavaScript Object Notation

JavaScript Object Notation (JSON) is a common format for representing structured data. It is based on JavaScript object syntax, but can be used independently of JavaScript code. JSON exists as a text string. This is useful when you want to transmit data across a network. JSON can represent complex, branching data structures. For example...

```
// { key: value, key: value... }, e.g.

{
  "cats": [
    "tortoiseshell",
    "tuxedo",
    "tabby"
    ]
  }
```

JSON can include objects within objects. For instance, note the `characters` object embeded within this opera object.

```
{
  "work": "Norma",
  "composer": "Bellini, Vincenzo, 1801-1835",
  "librettist": "Romani, Felice, 1788-1865",
  "premiere": "1831-12-26",
  "characters": [
    {
      "name": "Norma",
      "voice-type": "soprano",
      "arias": [
        "Casta diva"
        ]
      },
      {
        "name": "Adalgisa",
        "voice-type": "soprano",
        "arias": [
          "Sgombra è la sacra selva...Deh! proteggimi, o Dio!"
          ]
        },
        {
          "name": "Pollione",
          "voice-type": "tenor",
          "arias": [
            "Meco all'altar di Venere"
```

[3]Circumventing the limitations of the Twitter APIs is sometimes a reason for scraping. If this is your case, and you have a bit of Python know-how, you might try using twint.

```
            ]
        }
      ],
    "setting": "Gaul, c. 50-100 BC"
  }
```

The last JSON example restructed as a table would be as follows. Notice how the `characters` object has been flattened to fit the single hierarchy of a table. Imagine how it would look if the `arias` object had more than one aria.

work	composer	librettist	premiere	characters___name	characters___type	characters___arias	setting
Norma	Bellini, Vincenzo, 1801-1835	Romani, Felice, 1788-1865	1831-12-26	Norma	soprano	Casta diva	Gaul, c. 50–100 BC
				Adalgisa	soprano	Sgombra è la sacra selva... Deh! proteggimi, o Dio!	
				Pollione	tenor	Meco all'altar di Venere	

Most social media APIs deliver data in JSON format. While both of the tools we'll use today reformat tweet objects as tables for ease of use, it's important to know that tables will flatten the original data structure.

- Examples of tweet JSON: https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json
- Explanations of tweet fields and links to child objects: https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object

There are a number of fundamental attributes in a tweet object, like the time the tweet was posted (`created_at`), the unique identifier of the tweet (`id_str`), and the actual message (`text`) of the tweet.

The Tweet object is considered the 'parent object' to several child objects like the sender of the tweet (`user`), and the additional metadata that comes with the contextual content in tweets like links, hashtags, and attached media (`entities` and `extended_entities`).

The example below shows some of the name-value pairs you will see in the Tweet object.

```
{
"created_at": "Thu May 10 15:24:15 +0000 2018",
 "id_str": "850006245121695744",
 "text": "Here is the Tweet message.",
 "user": {
 },
 "place": {
 },
 "entities": {
 },
 "extended_entities": {
 }
}
```

Let's break down each of the above names and their corresponding values:

`created_at`: Time the tweet was created. All tweets are recorded in UTC.

`id_str`: Every tweet has a unique identifier. "id_str" is the string representation of the unique identifier.

`text`: The actual text of the tweet.

`user`: The user who posted the tweet.

`place`: Place is not always present. It is important to note the distinction between "place" and "coordinates". Place indicates that a tweet is associated with a place, but does not necessarily originate from a place. Coordinates represent the geographic location reported by the user or application.

`entities`: Additional metadata that provides contextual information about a Tweet like links, hashtags, and attached media

`extended_entities`: Tweets with photos, videos, and animated GIFs will include an "extended_entities" JSON object

TAGS setup

Let's move now to TAGS. Open your browser of choice and navigate to https://tags.hawksey.info/. Click on *Get TAGS*. My recommendation is that we use TAGS 6.1.[4]

Clicking on the TAGS v6.1 button will take you to a Google Drive window in which you are asked to make a copy of TAGS. Do so. This creates a copy of the TAGS spreadsheet in your Google Drive. Rename it to something descriptive, i.e. My Test Archive.

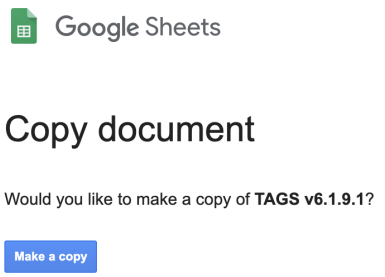


Figure 1: make a copy

In the navigation menu at the top of your screen, you should see a TAGS button. Go to **TAGS > Setup Twitter Access**. You'll be guided through a series of steps that will grant the TAGS tool access to your Drive and to your Twitter account. ***Nota bene***: be sure to enable pop-ups in your browser! In the pop-up window for granting access to Twitter, choose the **Easy Setup** button.

Creating your Twitter archive

Now that those preliminaries are out of the way, we can concentrate on retrieving our Twitter data. Enter your search string in the box next to **Enter term** of the TAGS instructions.

Then go to **TAGS > Run now!** After allowing for the script to run for a moment or two, click over to the Archive tab of your spreadsheet. Do you see your tweets? Have a look at the header row of your Archive tab. It should remind you of the Twitter JSON output we looked at earlier. Hawksey has mostly reused the Twitter field names, but not all of them are included in the default installation of the tool. If you are

[4]What's the difference between "easy setup" and "custom keys"? Hawksey introduced a simplification in version 6.1 that eliminates the necessity of creating a Twitter Developer account and registering an application. If you use TAGS 6.1 (recommended), what that means is that you are using the developer's Twitter API key and secret. So if Hawksey were to delete his account, or exceed the maximum number of users, your TAGS archive would not function. To exercise greater control, you might want to follow the instructions for v6.0 with the custom keys option, but we'll skip this for now.

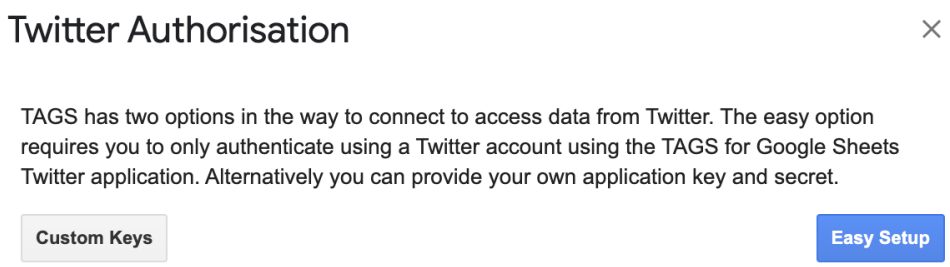


Figure 2: easy setup

Figure 3: enter term

keen on adding some more, like retweet_count and favorite_count, this is what to do. Take a big breath, go to **TAGS > Wipe Archive Sheet**, and delete all those tweets you just collected. Then, go to the far right side of the spreadsheet and add a new column after the field entities_str. In the header row, add the fields you want. This is the list of what is available to add. I might suggest adding a column header with retweet_count, and a second with favorite_count.[5] Then, go to **TAGS > Run now!**, and repopulate your spreadsheet. Scroll over to the right again. Did it capture your added Twitter fields? Excellent. You may also go to **TAGS > Update archive every hour** to keep collecting tweets automatically. The TAGS tool will continue issuing hourly calls to the Twitter REST API without you even needing to be logged into Google. To stop, just go to **TAGS > Stop updating archive every hour**.

Exploring and filtering data

Go to **TAGS > Add Summary Sheet** and **TAGS > Add Dashboard Sheet**. Two new tabs should appear in your Google spreadsheet, with some interesting data about your archive, like Top Tweeters, Tweets with the most RTs (retweets), and time plots of Twitter activity. What can these views tell you about your data? Hawksey has created two interface tools to interact with the Twitter data you collect: TAGSExplorer and TAGS Archive. You can click on them from either your Summary or Dashboard tab. ***Note***: You need to make your spreadsheet public on the web in order for this to work. Go to **File > Publish to the web**.

Each dot (also called a node) represents a Twitter user. The dot is sized by the number of connections (also called edges) it has with the other users in this dataset. So, the larger the node, the more edges it has (i.e. more tweets, mentions, replies). You can click on any node to get a window with the detail of those connections. TAGSExplorer will load only direct replies as the connections at first. But notice in the lower right-hand corner how you can add mentions and retweets as edges as well?

The edges will be differentiated by the type of line, e.g. solid gray for replies, dotted gray for mentions, and

[5]The fields `favorite_count` and `retweet_count` are often considered a measure of a tweet's "interestingness," although user behaviors are admittedly complex in this arena. See for example mock-retweeting.

Public web views	TAGSExplorer	<- conversation explorer
	TAGS Archive	<- searchable archive
Note	Grey = calculated fields	
	Blue = Require File > Published to the web	

Figure 4: TAGSExplorer and TAGS Archive

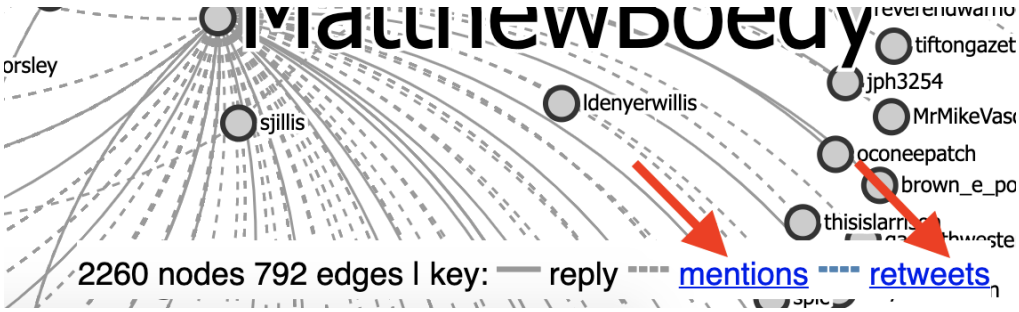


Figure 5: edges

dotted blue for retweets. Either way, if you load the mentions and retweets as edges or not, you will probably encounter what is known in network analysis as the hairball problem. There are many tools and techniques for grooming the hairball, but for right now, you might as well appreciate the awesome unruliness of your data. Can you make any sense of it? Who are the major actors in your Twitter network? Take time also to explore the Top Tweeters, Top Hashtags, and Top Conversationalists in your TAGSExplorer page. What can these basic visualizations tell you about your dataset?

If you want an easy way of drilling down to the level of individual tweets, you might be aided by the TAGS Archive interface, which will allow you to retrieve tweets from your data set by searching for any combination of user names, hashtags or phrases. What phenomenon observed in the TAGSExplorer interface might prompt you to look at individual tweets?

rtweet

The TAGS tool is among the easiest applications for collecting Twitter data, in that it doesn't require familiarity with programming languages or the command line, and it does a pretty good job of automating data collection, API limitations notwithstanding. With that said, it does not allow you to search for geo-coded tweets (tweets with a non-null `coordinates` field), which might be important if your research question has a geographical component.

In addition to supporting search strings in the way that TAGS does, rtweet also allows us to search by latitude and longitude coordinates.[6] And it includes some nice functions that allow us to inspect and visualize tweets, in addition to collecting them. If you are already familiar with programming in the R language, rtweet will help you to keep your workflow all in one place.

[6]Only a small percentage of Tweets contain geocoordinates, but it is better than nothing. If point coordinates are an insufficiently precise representation for your research question, you may need to define a geographic boundary and filter for tweets posted from within that boundary. See https://docs.google.com/document/d/1TIwZVl9hsz2q6ApvCr0pnGCC826M439JUndkRIU9xug/edit for an overview of geofencing using twarc.

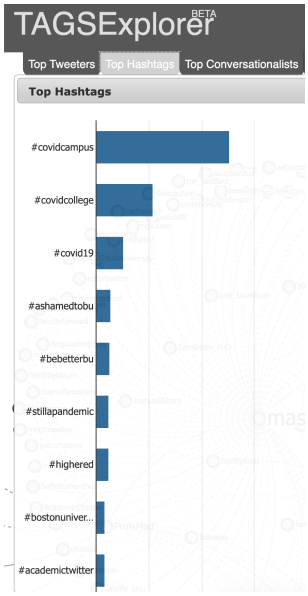


Figure 6: top n

Figure 7: TAGS Archive

To keep things simple, we will used RStudio Cloud to run the next few examples interactively. Go to https://rstudio.cloud and create an account. Then, click the dropdown arrow next to the **New Project** button and select *New Project from Git Repository*. Enter the following URL to create a project based on this handout: https://github.com/giannetti/collecting-twitter-data. Open the file called `collecting-twitter-data.Rmd`. In the script editor pane of RStudio Cloud (upper left), scroll down to this section, then press the little green arrow in the upper right corner of the code block below to run the set-up.

Create a Twitter App

On RStudio Cloud, there is an extra step to get rtweet up and running (you won't need to do this if your are using RStudio desktop). Follow the instructions to create a Twitter Developer App at https://web.archive.org/web/20200510175109/https://rtweet.info/articles/auth.html. Ccopy the name of your App and your key and token into their respective fields below. We will use the `saveRDS()` function to store our app credentials as an R object to reaccess as needed.

```
## Enter the details of your Twitter app and run this line of code. Be sure to remove the space charact

appname <- " " # whatever you called your app
api_key <- " "
api_secret_key <- " "
access_token <- " "
access_token_secret <- " "

token <- create_token(
  app = appname,
  consumer_key = api_key,
  consumer_secret = api_secret_key,
  access_token = access_token,
  access_secret = access_token_secret)

saveRDS(token, "twitter_token.rds")
```

A few examples of data collection using rtweet

To learn about the `search_tweets` function and the arguments it takes, enter `?search_tweets()` at the R console. For help with the full rtweet package, type `help(package="rtweet")` at the console or visit the intro vignette at https://web.archive.org/web/20200515224405/https://rtweet.info/articles/intro.html.

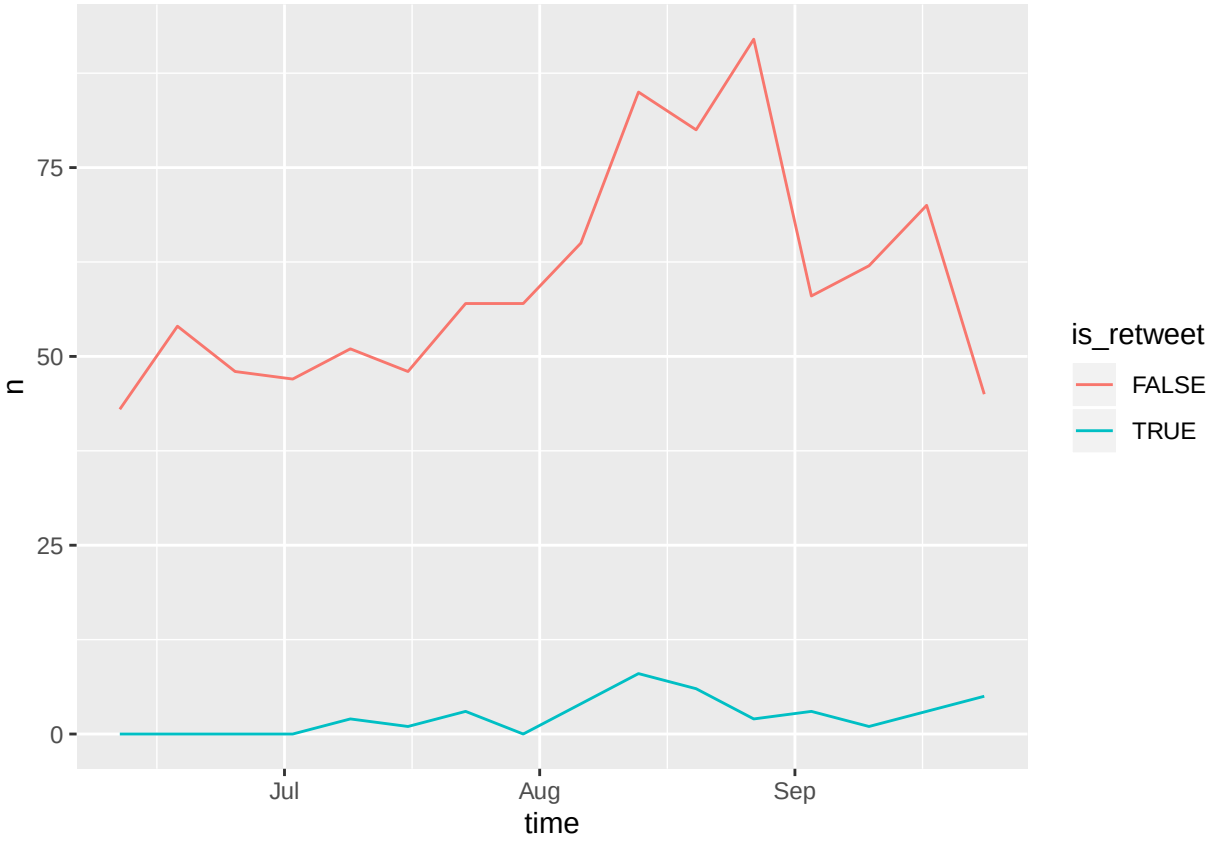
Note: The preceding step is critial on rstudio.cloud, or else the subsequent code blocks won't run. If you happen to be on RStudio desktop, you can dispense with the preceding code block and remove the `token = token` instruction, as the first time you run one of these functions, an authentication process will launch in a broswer window asking for you to connect your Twitter account to Kearney's Twitter Developer keys. This is similar to what we did earlier with TAGS.

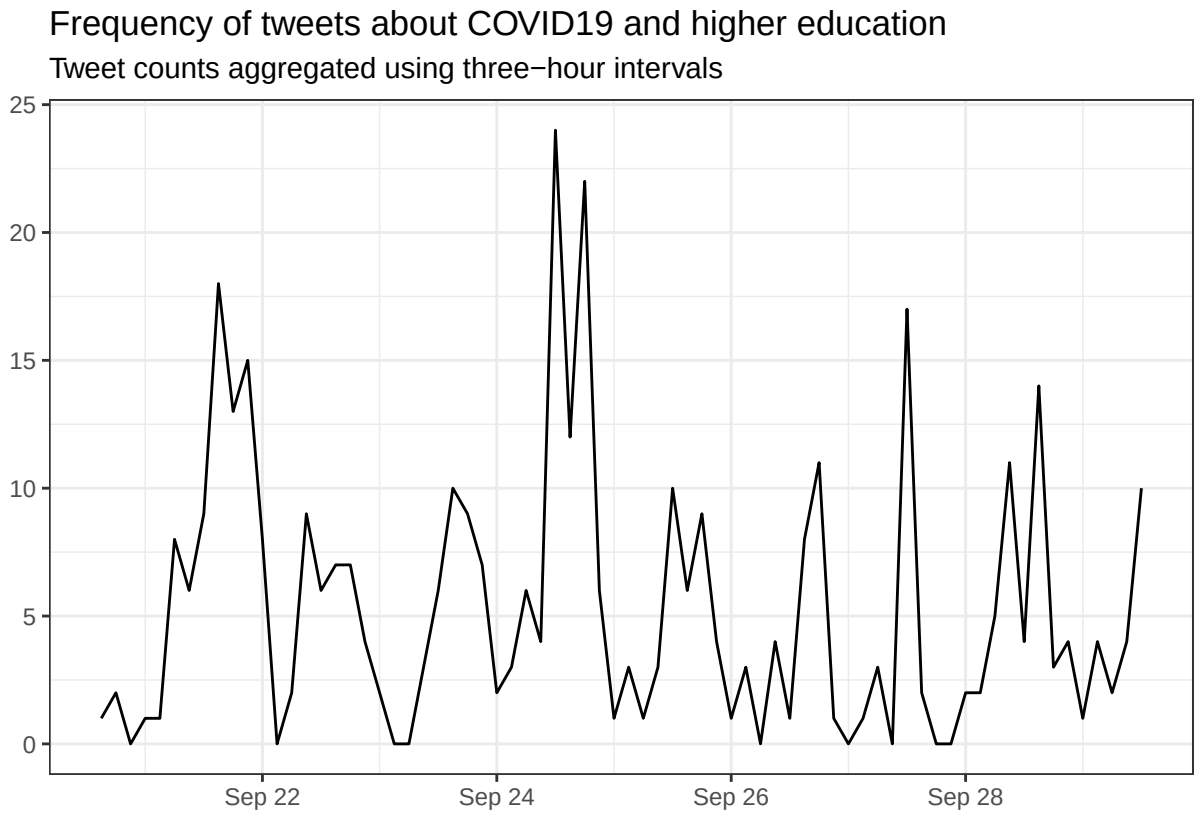
```
# depending on your session, you may need to read your key and token back into memory
token <- readRDS("twitter_token.rds")

# get most recent 1000 tweets posted by Biden's account
joe_tml <- get_timeline("JoeBiden", n = 1000, token = token)

## search by lat-lon pair
## you can look this up at https://www.latlong.net/
oldqueens_tweets <- search_tweets(
  geocode = "40.4987461,-74.4484437,5mi", include_rts = FALSE, token = token
```

```
)

## search using multiple queries
pandemic_tweets <- search_tweets2(
  c("\"pandemic university\"", "#covidcampus OR #covidcollege"),
  n = 500, token = token
)
```

Exploratory visualizations

rtweet allows you to quickly visualize frequency of tweets over time with the `ts_plot()` function. Enter `?ts_plot()` at the console for more examples with this function.





Try your own data collection

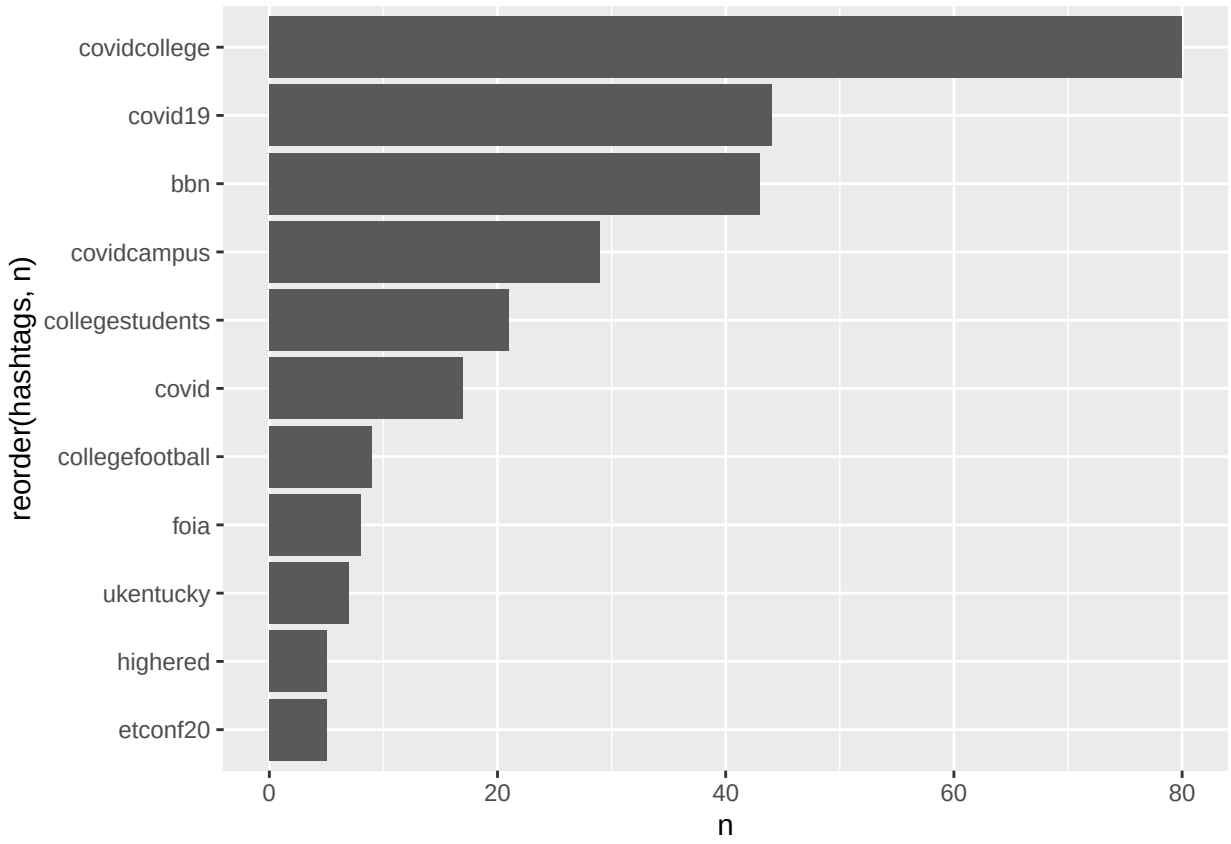
```
my_archive <- search_tweets(
  # enter your search terms and arguments
  # type ?search_tweets() at the console for help
)
```

Counting things

The subject of manipulation and analysis will be addressed in the second part of this workshop. But to leave you with a taste, this code block counts the hashtag entities present in the `library_tweets` object and sorts them in descending order by frequency, and visualizes the top ten as a bar graph.

```
select(pandemic_tweets, hashtags) %>%
  unnest(cols = c(hashtags)) %>%
  mutate(hashtags = tolower(hashtags)) %>%
  count(hashtags, sort=TRUE) %>%
  filter(hashtags != "NA") %>%
  top_n(10) %>%
  ggplot(aes(x=reorder(hashtags, n), y=n)) +
  geom_bar(stat="identity") + coord_flip()
```

```
## Selecting by n
```



Twitter developer policy and terms

Twitter requires that you adhere to its developer policy, which covers a range of technical matters, as well as its terms, which address ethical considerations, in order for you to use its APIs. Examine both documents carefully if you intend to use Twitter data for research purposes. The most salient point from the policy: downloadable datasets of over 50,000 tweets should include Tweet IDs and/or User IDs only.

Some of my favorite R learning resources

Arnold, T. and L. Tilton, "Set-Up" and "A Short Introduction to R," *Humanities Data in R* (Springer, 2015).

Bryan, J. et al. "Stat 545." https://stat545.com/index.html

Torfs, P. and C. Brauer, "A (very) short introduction to R" (2014).

Wickham, H. "Introduction," *R for Data Science* (O'Reilly, 2016).

Archiving Twitter data using twarc (Python)

Twarc, from DocNow, is another recommended tool for collecting twitter data. It has some wonderful functions and utilities that are not at present included in rtweet. Learn about it with these resources.

- *Collect Twitter Data with Twarc!*, https://scholarslab.github.io/learn-twarc/
- *Twitter Data Tutorials Series*, https://docs.google.com/document/d/1PfA9Q6tgMFgXWXMg9TRd2OTzUiz0In9HLbGiM edit