University College Dublin
Ireland's Global University

The Connected_Politics Laboratory

Collecting and Analyzing Twitter Data

COMPTEXT Conference

Natalia Umansky

University College Dublin

5 May, 2022

Who am I?

Natalia Umansky

nataliaumansky.github.io

PhD Candidate - School of Politics and IR (University College Dublin)

Research Interests

- International Relations
- Security Studies
- Political Communication
- Computational Social Science
- Social Media

The workshop - Overview (1/2)

Theory

1) Motivation
    - Why study Twitter?
2) Considerations
    - Which API do I need?
    - Data sharing
    - Possible bias

The workshop - Overview (2/2)

Practice

3) Data collection
   - Collection strategies
   - Resources
   - Packages
4) Data cleaning
   - Dealing with emojis
   - Noisiness
   - Stop word dictionaries
5) Data analysis
   - Intro to SNA with Twitter data

Part 1

Motivation

Motivation

Why study Twitter?

- Real (online) world dynamics
- Lots of metadata
- Important site of political communication
- Important site of social interaction
- It's very accessible
- Accessible R libraries
- Various types of actors interacting

Part 2

Considerations

Considerations (1/10)

Ideally, we have one or more research questions and hypotheses developed prior to the data collection

- This will help us:
  - Choose the right API
  - Develop the data collection strategy

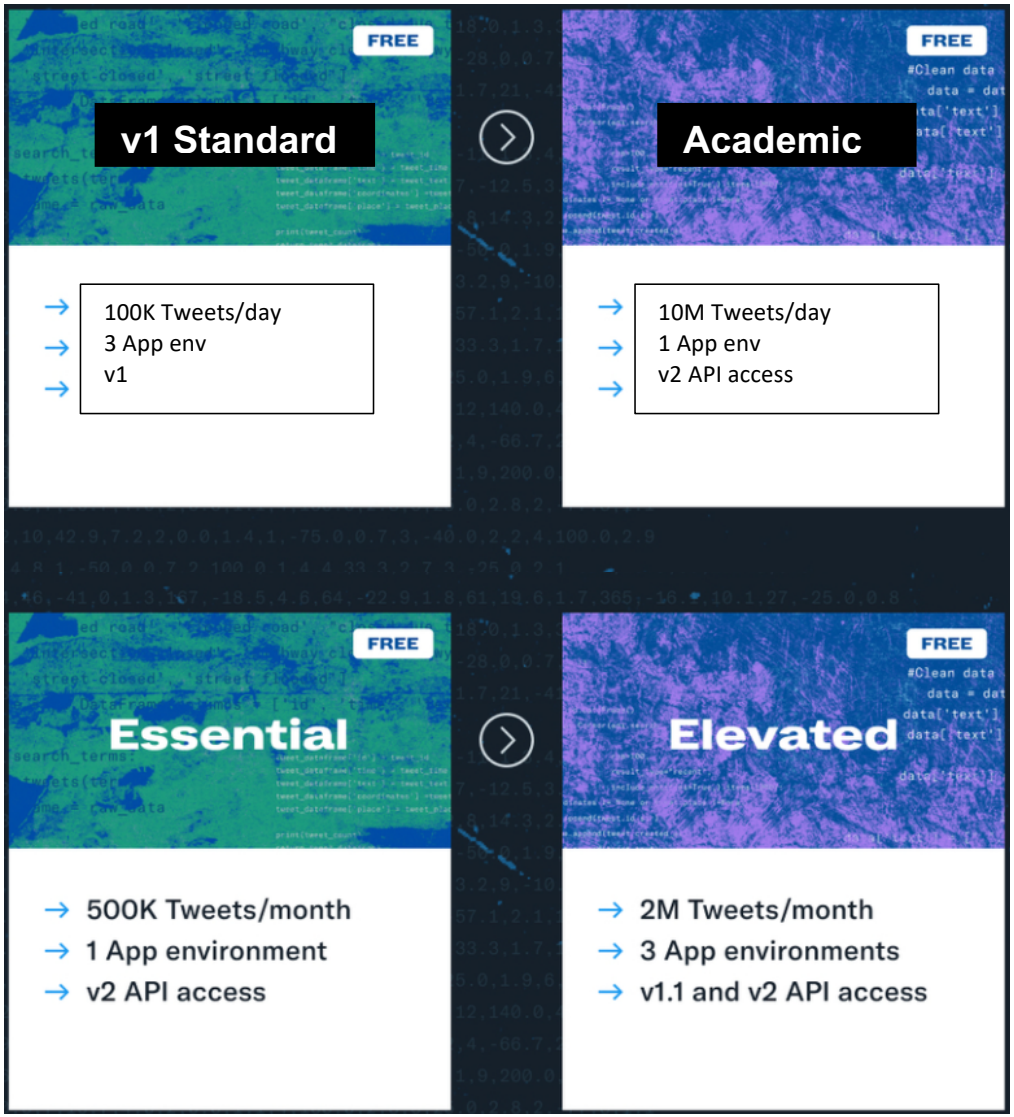
Considerations (2/10)

Many different ways to collect Twitter data

- Streaming
- Hisotrical data
- Users timelines
- Users followers and friends

Considerations (3/10)

Which API should I use?



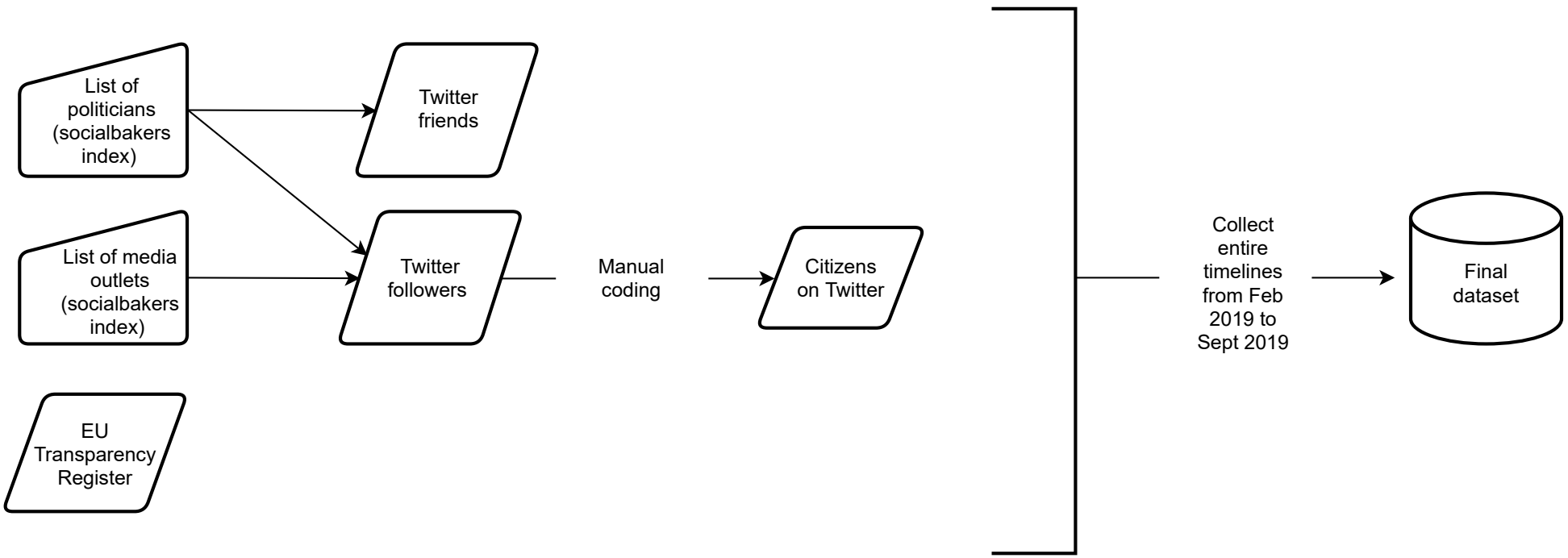
Considerations (4/10)

Final dataset

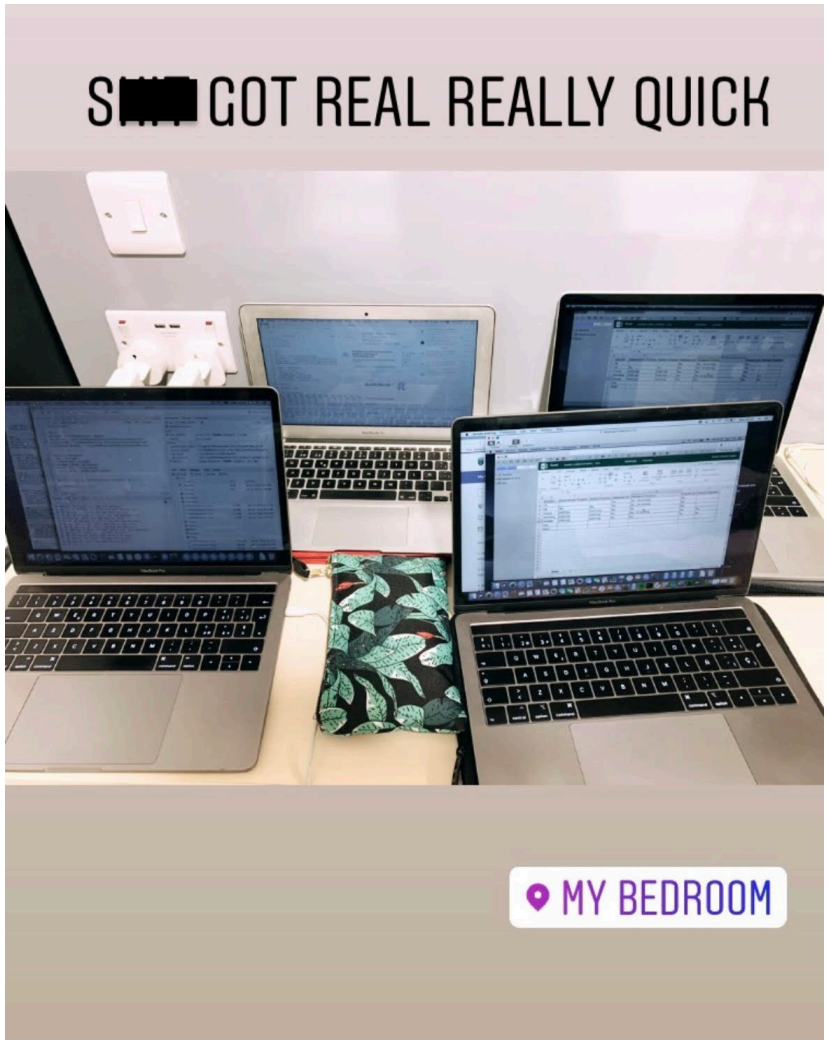
Group	Number of accounts	Number of tweets
Politicians and Governmental Institutions	513	360,028
Advocates	3,266	1,012,508
Politicians' Twitter Friends	15,695	5,062,110
The Media	216	215,026
Citizens on Twitter	23,891	4,312,732
Total	43,581	10,962,404

Considerations (5/10)

My data collection strategy - REST API



Considerations (6/10)



Considerations (7/10)

Academic API

```
library(academictwitteR)

queries <- c("amazon forest", "amazon rainforest", "amazon is burning",
             "amazonwatch", "the planets lungs", "amazon rain forest")

list_tweets <- list()

for (i in queries){
  list_tweets[[i]] <- get_all_tweets(
    query = c(i),
    exact_phrase = T,
    start_tweets = "2019-06-01T00:00:00Z",
    end_tweets = "2019-10-01T00:00:00Z",
    bearer_token = "",
    n=Inf)

  saveRDS(list_tweets, "~/list_tweets.RDS")
}
```

Considerations (8/10)

Sample validity:

- Streaming and historical tweets
  - REST API, users can expect to receive approximately 1% of the public tweet volumes
  - Academic API, full archive of all historic Tweets (within rate limit)

Academic API:

- Incomplete text
- Less metadata
- Long application process

Considerations (9/10)

R packages

- REST API -> rtweet (Kearney, 2019)
- Academic API -> academictwitteR (Barrie & Ho, 2021)

Considerations (10/10)

Data sharing and replicability

The best place to get Twitter Content is directly from Twitter. Consequently, we restrict the redistribution of Twitter Content to third parties. If you provide Twitter Content to third parties, including downloadable datasets or via an API, you may only distribute Tweet IDs, Direct Message IDs, and/or User IDs (except as described below). We also grant special permissions to academic researchers sharing Tweet IDs and User IDs for non-commercial research purposes.

In total, you may not distribute more than 1,500,000 Tweet IDs to any entity (inclusive of multiple individuals associated with a single entity) within any 30 day period unless you have received written permission from Twitter. In addition, all developers may provide up to 50,000 public Tweets Objects and/or User Objects to each person who uses your service on a daily basis if this is done via non-automated means (e.g., download of spreadsheets or PDFs).

Part 3

Data Collection

Twitter — Authorisation

Authorization to use Twitter APIs requires at least three steps*

1) open a user account on Twitter
   - a personal or an institutional (perhaps, for a research project) one
   - done once, takes minutes

2) with that user account, apply for a developer account
   - so that you are recognised as a developer, have access to the developer portal
   - done once per account, takes days to get approved manually

3) with that developer account, register a Twitter app
   - so that you have the keys and tokens for authorisation
   - repeated for every project, takes minutes

- There may be additional steps, such as registering for the Academic Research access.

Data collection

We will collect data through Twitter's Standard v1.1 APIs

- free of charge
- thanks to rtweet's rstats2twitter app, can be used immediately

Data collection

Start your script

```
library(rtweet)
library(tidyverse)
library(tidytext)
library(ggplot2)
```

Rtweet package

There are three main groups of functions to collect historical data, starting with

- search_
  - such as search_tweets or search_users
- lookup_
  - such as lookup_tweets or lookup_users
- get_
  - such as get_followers or get_friends

```
search_tweets(q,
              n = 100,
              type = "recent",
              include_rts = TRUE,
              geocode = NULL,
              max_id = NULL,
              parse = TRUE,
              token = NULL,
              retryonratelimit = FALSE,
              verbose = TRUE,
              ...)
```

Data collection - Exercise

`search_tweets`

1) Collect the latest 30 tweets that
   - include the hashtag "AcademicTwitter"
   - and assign the resulting data frame to df_tweets

2) Take some time to explore the data frame
   - see which variables are in there, and how they are called
   - think about how you could use these variables for research
   - hint: use functions like `View`, `str`, `names`, `tibble::glimpse`

Solution

```
# exercise 1 ----------------------------------------------------------

df_tweets <- search_tweets(q = "#AcademicTwitter",
                           n = 30)
```

Notes

- Twitter usernames under variable screen_name
  - can be misleading
  - Twitter allows user to change their usernames and display names
- User IDs do not change
  - user_id is a better variable for reproducible research
- The date and time data are matched to Greenwhich Mean Time
- You may wish to exclude retweets
  - setting `include_rts = FALSE`

Queries

- Include the word “publish” and “perish”, not necessarily in that order

```
search_tweets(q = "publish perish",
```

- Include the word “publish” or “perish”

```
search_tweets(q = "publish OR perish",
```

- Include the exact phrase “publish or perish”

```
search_tweets(q = "\"publish or perish\"",
```

- Include “publish” but not “perish”

```
search_tweets(q = "publish -perish",
```

Queries - language

- Include “publish” and are written in English

```
search_tweets(q = "publish lang:en",
```

- Include “publish” and are not in German

```
search_tweets(q = "publish -lang:de",
```

Data collection - Exercise

`search_tweets`

3) Collect the latest 50 tweets that include

- the phrase "publish or perish"
- and the word "academia" but not the word "PhD"
- excluding retweets

`search_users`

4) Collect information on 30 users that

- are associated with the word "PhD", but not with the word "rstats"
- read one of these users' bio on their homepage via a browser

Solution

search_tweets

```
# exercise 3 ------------------------------------------------------

df_tweets <- search_tweets(q = "\"publish or perish\" academia -phd",
                           n = 50)
```

search_users

```
# exercise 4 ------------------------------------------------------

df_users <- search_users(q = "PhD -rstats",
                         n = 30)

View(df_users)
```

rate_limit

Check your remaining rate limits for specifically the search_tweets function

```
rate_limit(query = "search/tweets")
```

get_timeline

- Collect the latest posts from one or more users
- Specified by username or user IDs, with the user argument
- Limited to 3,200 tweets per user-timeline
- There is no retryonratelimit argument

```
get_timeline(user,
             n = 100,
             max_id = NULL,
             home = FALSE,
             parse = TRUE,
             check = TRUE,
             token = NULL,
             ...)
```

get_followers

- Collect a list of followers, following one user
- returns a single column of user IDs, not usernames
- Limited with 75,000 followers per 15 minutes
- Use retryonratelimit = TRUE to surpass the limit
- Necessary to iterate to use it for multiple users

```
get_followers(user,
              n = 5000,
              page = "-1",
              retryonratelimit = FALSE,
              parse = TRUE,
              verbose = TRUE,
              token = NULL)
```

Data collection - Exercise

`get_timeline`

5) Collect the most recent tweets posted by the 30 users identified in excercise 3

`get_followers`

6) Collect a list of 20 accounts following The Connected_Politics Lab (@Connected_Pol)

`rate_limit`

7) Check your rate limits

Solution

```
# exercise 5 ----------------------------------------------------------
df_timelines <- get_timeline(user = df_users$user_id)
```

```
# exercise 6 ----------------------------------------------------------
df_followers <- search_users(q = "Connected_Politics")$screen_name
```

```
# exercise 7 ----------------------------------------------------------
df_limits <- rate_limit() %>%
  mutate(difference = limit - remaining) %>%
  arrange(-difference)
```

Part 4

Data Cleaning

Data cleaning

- The rtweet package does a very good job with data preperation to start with
  - returns data frames, with mostly tidy data
- Further data preparation depends on your research project
  - most importantly, on whether you will work with texts or not
  - we will cover some common preparation steps

Data cleaning

- There are many components of tweets as texts
  - e.g., mentions, hashtags, emojis, links etc.
  - but also punctuation, white spaces, upper case letters etc.
  - some of these may need to be taken out before analysis

- I use the stringr (Wickham, 2019) and quanteda (Benoit et al., 2018) for string operations

Data cleaning

- Remove mentions

```
str_remove_all(string = tweet, pattern = "[@][\\w_-]+")
```

- Remove hashtags

```
str_remove_all(string = tweet, pattern = "[#][\\w_-]+")
```

- Remove links

```
str_remove_all(string = tweet, pattern = "http\\S+\\s*")
```

Data cleaning

- Remove emojis

```
iconv(x = tweet, from = "latin1", to = "ASCII", sub = "")
```

- Remove Punctuations

```
str_remove_all(string = tweet, pattern = "[[:punct:]]")
```

- Change case

```
str_to_lower(string = tweet)
```

Data cleaning - Exercise

`tidyverse`

8) Using the df_tweets dataset, create a new variable without mentions, hashtags, links, and emojis

Solution

```
# exercise 8 ----------------------------------------------------------
df_tweets %>%
  mutate(no_mentions =
           str_remove_all(string = text, pattern = "[@][\\w_-]+"),
         no_mentions_hashtags =
           str_remove_all(string = no_mentions, pattern = "[#][\\w_-]+"),
         no_mentions_hashtags_links =
           str_remove_all(string = no_mentions, pattern = "http\\S+\\s*"),
         all_clean =
           iconv(x = no_mentions_hashtags_links,
                 from = "latin1", to = "ASCII", sub = "")) %>%
  select(text, all_clean)  %>%
  View()
```

Remove stop words

This operation requires a tokenised-to-word variable

```
corp <- corpus(tweets)

  tokens<- corp %>%
    tokens(remove_punct = TRUE) %>%
    tokens_tolower())


  tweets.dfm <- dfm(tokens,
                    tolower=T,
                    stem = F,
                    remove = stops,
                    remove_punct=T,
                    remove_numbers =T,
                    remove_symbols = T)
```

Remove stop words

- I use the quanteda dictionary
- Important to check it

```
stopwords("english")
```

Part 5

Data analysis

Data analysis - SNA

Twitter data is suitable for network analysis

- There are at least five networks

- `followers`
- `retweeters`
- `quoters`
- `repliers`
- `likers`

Data analysis - SNA

- Networks are composed of nodes and edges
- The nodes and edges are often kept separate for analysis
- We will use two packages for network analysis
  - `tidyverse` for data manipulation
  - `ggnet2` for visualisation

Data analysis - SNA — tidyverse

```
tweets <- readRDS(url("https://github.com/NataliaUmansky/Twitter-workshop/blob/

tweets %>%
  filter(is_retweet == TRUE) %>%
  group_by(screen_name, retweet_screen_name) %>%
  summarise(rts = n()) %>%
  head()
```

Data analysis - Exercise

9) Load the data

10) Create an edge list

11) Create a retweet network (answer available in 'Exercises.R' file)