Workshop: Tracing Twitter @ IRS Spring Academy 2018

Maier, D. (FU Berlin) & Stoltenberg, D. (WWU Münster)

Introduction

In our work, we use Twitter data to investigate digital public spheres. However, there are many other contexts where Twitter data might be of interest, ranging from disciplinary contexts of the social sciences to geography or computer science.

Public spheres are discursive spaces, where individuals - and sometimes collective actors - discuss matters of common interest and try to reach a common judgment about them (Hauser, 1999). While this reflects Habermas's (1981) concept of an "ideal-speech situation", modern communication theory characterized public spheres as mass mediated communication spaces where institutionalized speakers and audiences have clearly delineated roles (Gerhards & Neidhardt, 1991).

Digital public spheres, particularly on the social web, differ remarkably from this assessment, however, and have therefore renewed interest in public sphere research. They show a greater fluidity in speaker and audiene roles as well as a growing visibility and importance of the relations and networks between actors (Benkler, 2006; Castells, 2008). Furthermore, there is greater topical diversity and so-called issue publics emerge (Berman & Mulligan, 2003), which may be read as both a sign of greater diversity or of fragmentation. Lastly, it is less clear how to differentiate between the public and the private communication (Brosius, 2016; Strippel et al., 2018). All of these features can be observed well on Twitter.

The spatial dimension of digital public spheres has not been explicitly addressed in public sphere theory thus far. Rather implicit, the research focused on national public spheres. To lesser extent reseach investigated transnational, particularly European public spheres and - less often still - urban public spheres. The patterns of digital communication in general cast doubt on whether this is (still) appropriate. Empirical observations point towards the digital linkage of very specific places, which can in part be explained by frequent travel, political and economic relations and (sub)cultural similarities or shared experiences. We refer to this as a potential translocalization of public spheres.

To determine the role space plays in digital public spheres, we need relatively vast amounts of (social media) data. We also need to gain insights into geographical information embedded in such data. In this workshop, we share some of the ways we tackle these problems.

Before we can start…

… we need to get our program ready. To use this script you must include some functions that makes it much more convenient for you to go through all the steps we apply here.

This document is full of `R` code. Each chunk of code is highlighted can be executed by marking it and pressing `ctrl` + `Enter`. By executing the following chunk you tell `R` to visit the URL and to read and save some convenience functions that we wrote for this class.

```r
# Please execute this line of code
source("https://raw.githubusercontent.com/danielmaier-fub/irs_2018/master/utils.R")
```

Some very brief notes on R

`R` is essentially a glorified calculator, in that it can do anything ranging from very easy to very hard math. *R Studio* is a development interface that makes `R` a little easier to use. In it, you have four main parts. At the top left is your script. Here you write your code. At the bottom left is your console. Here, it prints any commands you executed and the results. At the top right is your environment where you see all your dataframes, variables, and any other objects you've created. At the bottom right, you have space for plots you've created, as well as a help page, an overview of packages you've installed and so on.

If you enter a basic math problem, then run the code, the answer will pop up in the console.

```r
#yay basic math!
10*2
```

Adding a number sign `#` before some text marks the rest of the line as a comment for R, so it will not attempt to execute whatever is written there.

If you want to create a more permanent object, you need to assign a value to an object, using a `<-` operator. For instance:

```r
a <- 10*2
```

After you run this line of code, an object named `a` with the value `20` should be visible in your environment at the top right. You can now keep working with this later on in the script, for instance:

```r
a^2
```

will give you 400, because `R` remembers that `a` is equal to 20. This works not just for individual numbers, but for more complex variables, as well.

```r
#gives you a vector containing the numbers 1 through 5.
b <- 1:5

#gives you a vector containing five names
c <- c("James", "John", "David", "Patricia", "Maria")

#gives you a dataframe consisting of the two variables we created just before.
df <-data.frame(b,c)

# take a look...
View(df)
```

If we move beyond mere calculator functionality, R works with functions which are basically pre-made chunks of code that allow you to do quite advanced things with very little code. For instance, we use the print function a lot, if we want to see the content of some object.

```r
print("hello world!")
```

A lot of general functions are implemented in R already if you download it. For more specific use cases (like querying the Twitter API) there are so-called packages or libraries you can download that contain functions relevant for a specific context. Packages need to be downloaded only once, but loaded every time you start a new session. The latter is done using the `library` function.

```r
# package used for downloading and analyzing tweets
install.packages("rtweet")

# load the package
library(rtweet)
```

Of course, a lot more can be learned about `R`, but what has been said should get you through this script without problems. If you want to learn more, here is a Youtube tutorial on basic R. Furthermore, the R for Dummies webpage has some fine tutorials covering most basic issues. Almost any question you can imagine will have been asked and answered at some point on the Stack Overflow forum.

Accessing Twitter's Search and Streaming APIs

Twitter allows any person to access data via their Application Programming Interfaces (APIs). This is a convenient way to access data because the retrieved content comes in the form of highly standardized datasets and because as long as you comply with the API's basic rules, you do not have to worry about any legal issues regarding data retrieval. Twitter provides an easy-to-understand (albeit sometimes frustratingly vague) documentation to its APIs here.

Two APIs may be relevant when collecting Twitter content: the Search API and the Streaming API. The Streaming API allows you to access all tweets regarding a specified query (as long as the volume doesn't exceed 1 percent of all content being tweeted at any given time) in real time.

The Search API enables you to access tweets from the past (roughly) seven to ten days. You can also access user profile infos, followers and friends, as well as many other infos through the Search API. Unlike the Streaming API, the Search API has rate limits regarding how much info you can access per 15 minute interval. For instance, you may access up to 18,000 tweets or 90,000 user profiles per 15 minutes, after which the API denies you further queries. However, you can easily pick it up again where you left after the 15 minutes are over. Do not worry too much about coincidentally hitting the rate limits. As long as it doesn't happen too frequently, Twitter won't care.

For all queries to the Twitter APIs we can use the `rtweet` package, which has functions for standard requests readily embedded. If you want to do anything more ambitious than what we discussed during the workshop, the package's documentation may be good place to start searching.

Registering an app

Before you can start querying the APIs, though, you need to confirm you are, in fact, allowed to do so. For this, you need to register an app with Twitter. Don't worry, it only takes five minutes. All you need is a regular Twitter account that has been verified with a mobile phone number (see this page if you're not sure about how to do that). Next, in your browser go here and click *Create New App*. Here, you need to enter a name and brief description for your app (don't bother with too much detail) and a website (pick a fake one if you don't have one). Make sure to set the *Callback URL* to http://127.0.0.1:1410, otherwise the authentification process will not work later on. Once you click on the register button at the bottom of the page, you're good to go.

On the app page, go to the *Keys and Access Tokens* tab. From there, copy your **app name**, **Consumer Key (API Key)** and **Consumer Secret (API Secret)**. Go to your R script and replace the XXX with the info there. Then run the `create_token` function.

```r
# Your App Information
appname <- "XXX"
key <- "XXX"
secret <- "XXX"

# Create Access Token
twitter_token <- create_token(app = appname, consumer_key = key, consumer_secret = secret)
```

If you get a `401 error` (unauthorized), try the following: * Make sure you copied all the info correctly and without any additional spaces, * Make sure your callback URL is set correctly, * Go to the *Settings* tab on the app page and uncheck the "Enable Callback Locking" option.

Querying the Twitter APIs for tweets from - or about - a place

As we have discussed, there are several places geographical info may be embedded within a tweet, ranging from the extremely standardized (geotags attached to tweets) and the somewhat easy to identify (geo location field in the user info) to the extremely messy (places mentioned in the tweet or elsewhere in the user info). Depending on which of these we are interested in retrieving, different ways to query the APIs make sense.

1. Using the `search_tweets2` function and the location name as a query casts a wide net, picking up on any of the above references.
2. Using the `search_tweets2` function and a geocode (consisting of a "latitude,longitude,radius" template) only gets you geotagged tweets and tweets by users who mention the relevant location in their geo location field. These two approaches can be combined to get tweets by users in a location who tweet about a specific topic. Keep in mind that the language of the query (English here) pretty much pre-determines the languages of tweets you will find.
3. Using the Streaming API function `stream_tweets` and a bounding box (consisting of four latitude/longitude points), we can access exclusively geotagged tweets. However, this function does not allow for the possibility of combining a location and a topical search.

```
# Option 1
query_Berlin <- search_tweets2(q = "Berlin", n = 200,
                               type = "recent", include_rts = FALSE,
                               retryonratelimit = TRUE)

# Option 2
geocode_Berlin <- search_tweets2(q = "", n = 200,
                                 type = "recent", include_rts = FALSE,
                                 retryonratelimit = TRUE,
                                 geocode = "52.52001,13.40495,20mi")

#Combining a text query and geocode
query_traffic_geocode_Berlin <- search_tweets2(q = "traffic", n = 200,
                                               type = "recent",
                                               include_rts = FALSE,
                                               retryonratelimit = TRUE,
                                               geocode = "52.52001,13.40495,20mi")

#Option 3
geotag_Berlin <- stream_tweets(q = c(13.0882097323,52.3418234221,
                                     13.7606105539,52.6697240587),
                               timeout = 60, verbose = TRUE)
```

Some additional pointers on the arguments of the functions above: In the `search_tweets2` function `q` specifies the search query. The argument must be present, but can be left empty, if we are only interested in a specific geocode. `n` specifies the number of tweets we want to download. `type` specifies whether we want to see the most "recent", vs. "popular" or "mixed" tweets. `include_rts` specifies whether retweets should be included in our sample vs. only original content. `retryonratelimit` only becomes relevant once we want more content than is allowed by the API per 15 minute interval. This leads the system to take a break after the allowed number of queries and start again once 15 minutes have passed. `geocode` will lead to a sample of only tweets from users from the specified area (point radius).

In the `stream_tweets` function, `q` may be a bounding box for geotags (as above) or a topical query, but not both (unfortunately…). `timeout` gives the duration for which the stream shall be open (in seconds, so timeout = 60 gives you one minute worth of tweets). `verbose`, if TRUE, allows you to see info on the data collection process in the Console.

Most of the time, if you do not specify an argument, it will default to some option and still deliver results. But I like to be quite specific in my code so I can easily see what sort of results I'll get.

If you don't know the latitude/longitude of a location, use lookup_coords, which gives you both a point and bounding box of coordinates. Careful, though, this might lead you astray for ambiguous place names.

```r
# look for geo-coordinates of a city
Berlin_coords <- lookup_coords("Berlin")
```

The data you get from both the `search_tweets2` and the `stream_tweets` functions doesn't include info on the users who tweeted them (beyond their ids and screennames). However, if we want to access, for instance the location field, or even the profile descriptions, we need additional user info. This can be acquired using the `lookup_users` function. Our dataset already contains a list of the users we need in the `user_id` column, which we can specify by giving the function the name of the dataset, followed by a `$`-sign and the name of the column we want to query.

In principle, the APIs can deal equally well with queries for user-ids and screennames. I'd usually recommend going for the ids, however, as these are fixed for each account while users do have the ability to change their screennames.

The `merge` function is a convenient way to join the two matching datasets of tweets and user info into one. (Note: This is only possible because both datasets share common identifiers, such as the user-ids).

```r
# get user information
query_traffic_geocode_Berlin_users <- lookup_users(query_traffic_geocode_Berlin$user_id)

# merge user information with tweet dataset
query_traffic_geocode_Berlin_merged <- merge(query_traffic_geocode_Berlin,
                                             query_traffic_geocode_Berlin_users)
```

Training Data

Because of time restrictions, we already have a data set that we want to explore together. Please visit this page and save the file in some folder. I saved the file on my desktop, and you can do that as well.

Now, we need to load that dataset into `R` to analyze it. Therefore we have to enter the path where we saved it. Please consider that `R` cannot handle "\", because it is interpreted as a so called escape character. That is why we must use "/" instead of "\" for each path we enter. Additionally we have to consider the file extension, which is ".RData" in this case. There are numerous options to read files. Whether you have a ".csv"-File an excel ".xls"/".xlsx", an SPSS/SAS-file … whatever file format you have you can almost certainly read it with `R`, which is a big advantage. However, we've got the very easy to read and load ".RData"-format here for practice.

```r
# This is where I saved the file
pathToFile <- "C:/Users/maier/Desktop/data_irs.RData"

# Dear R, please load the file associated with the path...
tweets <- get(load(pathToFile))

# class
class(tweets)
```

Now that we have the file loaded, we can take a look at what objects it brought us. In our file is one object, and that is our data set called "tweets". Data sets that you retrieve via rtweet (whatever data that is) is in the 'data.table' format, which is also of class 'data.frame'.

The data class should not bother you too much. However, in R, certain operations can only be done with certain data classes. Think about a data.table/data.frame as stuctured like an Excel spreadsheet with columns and rows, where each tweet is a row and each column has some information about the tweet such as the text and URLs within the text, or very important, geotags.

The first task we want to accomplish is to take a look at the variables that we have in the data set and how many tweets we got. We will use two very generic functions: one for looking at the names of the columns (indicating what variables we have in the data) and second count how many tweets we have.

```r
# take a look at the data
View(tweets)

# what are the names of the variables we have in the data set?
names(tweets)

# how many tweets have we got?
length(tweets[,1])
```

As you can see the data-set is quite extensive. Now, what we want to know is how many tweets are written in what language?

```r
# get a table counting the number of occurrences of a language
tlang <- table(tweets$lang)

# print out the object saved
tlang

# sort
tlang <- sort(tlang, decreasing = TRUE)

# create a barplot
barplot(tlang, main = "Languages in Twitter Data Set")
```

Bot or not?

On Twitter we frequently meet so-called bots. Bots are computer programs that tweet automatically or respond to messages of other users tweeting about certain topics. Although bots are a really interesting subject to study, we are either more interested in the communication of real human beings or we simply need to know what tweets in our data set corresonds to bot content and what content does not. We can use computer programs to investigate what user is likely a bot. One of these program tools is `borrnot`. Let's check it out:

```r
# names of two bots and two "human" users
names <- c("VegivonRou", "zufallshorst", "realDonaldTrump", "85maier1")

# check bot probability
botprob <- botornot(names)

# look at the bot-probability
botprob
```

As you can see the two true bots that we handed over to botrnot are highly likely to be idetified as robots whereas realDonaldTrump is unfortunately not very likely to be a robot. My twitter account is also likely a robot although I am not . . . :)

Bot-identifier programs are usually based on some kind of machine learning algorithm that is trained with pre-evaluated data. Such programs learn that bots are usually associated with a specific tweeting profile, e.g., continually tweeting, they are likely to tweet hundereds of messages within a very short time period and they often tweet similar or the same messages. However, some bots are also not very active. Those bots are similar to my tweeting profile because I do not tweet a lot, but am more an observer.

Finding networks on Twitter

Twitter is an interactive communication platform not only because you and everyone signing in has the ability to publically communicate, but also because users are able to interact with one another. I can respond to someone else's messages by using the @reply-function or by @mention-ing someone. I might also share the message of a user I am following, i.e., retweeting a message. And this type of communicative relation is something we can make use of to reconstruct networks. Networks are made out of two components, nodes and egdes (Wasserman & Faust, 1994). Actors such as twitter users can be considered the nodes of a network and edges can be considered the relationships between them. Network analysis builds on mathematical graph theory. That is why networks are frequently referred to as graphs. The `graphTweets`-packages offers a neat way to build networks from data coming straigt out of `rtweet`. The resulting network can then be analyzed and one of the first steps in network analysis is usually vizualization. For network visualization we use the `networkD3`-package.

```r
frac <- 1:100

# Find communication relations
edges <- gt_edges(tweets[frac,], tweets = text, screen_name, status_id)

# Convert relations to Networks
graph <- gt_graph(edges)

# change format
nwD <- igraph_to_networkD3(graph)
nw.df <- data.frame(nwD$links$source, nwD$links$target)

# Visual analysis
networkD3::simpleNetwork(nw.df)
```

Unfortunately we cannot dive into network analysis and different visualization techniques. These types of analysis are research fields of their own. But for whoever is interested we offer futher readings. Please see the list of recommended literature at the end of this document.

Twitter activity on a map

In order to map out twitter activity spatially, we need to figure out where a tweet was sent from. Luckily, Twitter offers users the option to uncover where they are sending their tweets from. However, only a small fraction of users truly use this option. Most users hide their location from other twitter users for privacy reasons. Although there are possibilities to estimate the location of users with relatively high accuracy, we will not focus on these techniques. Instead, we want to map those who share their local information publically.

```r
# which Tweets have a geotag?
geolocated <- which(lapply(lapply(tweets$geo_coords, is.na), all) == FALSE)

# take only the subset of the data with geotags
# and those columns we need ...
names(tweets)
columnsWeNeed <- c(2,5,66)
dataGeo <- data[geolocated, columnsWeNeed]

# Use function from pre-loaded data in step 1 (before we start...)
df <- getGeoCoords(dataGeo)

# create a map
```

```r
map <- leaflet() %>% addTiles() %>%
                     addMarkers(lng=df$lon, lat=df$lat,
                                popup=c(paste0(df$screen_name, ": ", df$text)))

# Print the map
map

mapGeoNet(data)
```

Conclusion

This was only a very small fraction and far from a comprehensive overview. However, we hope that this short workshop here at the IRS Spring School gave you a taste of what lies ahead if you decide to dive deeper into `R` and the big pool of publically available data.

References and Recommended Readings

References

- Benkler, Y. (2006). *The wealth of networks: How social production transforms markets and freedom.* Yale University Press.
- Berman, J. & Mulligan, D. K. (2003). Issue Advocacy in the Age of the Internet. In D. M. Anderson & M. Cornfield (Eds.), *The Civic Web: Online Politics and Democratic Values* (pp. 77-83). Landham: Rowman & Littlefield.
- Brosius, H.-B. (2016).Warum Kommunikation im Internet öffentlich ist. Zu Andreas Hepps Beitrag "Kommunikations- und Medienwissenschaft in datengetriebenen Zeiten". *Publizistik, 61*, 363-372.
- Castells, M. (2008). The New Public Sphere: Global Civil Society, Communication Networks, and Global Governance. *The Annals of the American Academy of Political and Social Science, 616*(1), 78-93. doi:10.1177/0002716207311877
- Gerhards, J. & Neidhardt, F. (1991): Strukturen und Funktionen moderner Öffentlichkeit. Fragestellungen und Ansätze. In Müller-Doohm, S. & Neumann-Braun, K. (Eds.), *Öffentlichkeit, Kultur, Massenkommunikation. Beiträge zur Medien- und Kommunikationssoziologie* (pp. 31-59). Oldenburg.
- Habermas, J. (1984). *The theory of communicative action (Vol. 2).* Beacon press.
- Hauser, G. A. (1999). *Vernacular voices: The rhetoric of publics and public spheres.* Univ of South Carolina Press.
- Strippel, C., Bock, A., Katzenbach, C., Mahrt, M., Merten, L., Nuernbergk, C., Pentzold, C., Puschmann, C., & Waldherr, A. (2018). Die Zukunft der Kommunikationswissenschaft ist schon da, sie ist nur ungleich verteilt. Eine Kollektivreplik. *Publizistik, 63*, 11-27.
- Wasserman, S. & Faust, K. (1994). *Social Network Analysis.* Cambridge, MA: Cambridge University Press.

Recommended Readings (Network Analysis)

- Borgatti, S. P., Everett, M. G., & Johnson, J. C. (2018). *Analyzing social networks.* London: Sage.
- Luke, D. (2015). *A User's Guide to Network Analysis* in R. Cham, Heidelberg, New York,. . . : Springer.

Recommended Readings (Locating Users)

- Graham, M., Hale, S. A., & Gaffney, D. (2014). Where in the world are you? Geolocation and language identification in Twitter. *The Professional Geographer, 66*(4), 568-578.
- Hecht, B., Hong, L., Suh, B., & Chi, E. H. (2011). Tweets from Justin Bieber's Heart: The Dynamics of the "Location" Field in User Profiles. In *Proceedings of CHI 2011, Session: Twitter Systems* (pp. 237-246). Vancouver.
- Kinsella, S., Murdock, V., & O'Hare, N. (2011). "I'm Eating a Sandwich in Glasgow": Modeling Locations with Tweets. In *Proceedings of SMUC '11* (pp. 61-68). Glasgow.
- Poorthuis, A., Zook, M., Taylor Shelton, M. G., & Stephens, M. (2016). Using Geotagged Digital Social Data in Geographic Research. In Clifford, N., Cope, M., Gillespie, T. & French, S. (Eds.) *Key Methods in Geography* (pp. 248-269). London: Sage.