SocialMediaLab Tutorial - University of Sydney (2018)

Tim Graham and Rob Ackland (ANU)

5 June 2018

Introduction

This is a tutorial for the *SocialMediaLab* R package. In this tutorial you will learn how to collect social media data from Twitter (using the #auspol hashtag), create networks, and perform basic social network analysis (SNA). Note: this tutorial has been adapted from a workshop given at the 2017 International Communication Association Annual Conference (ICA17).

SocialMediaLab enables users to collect social media data and create different kinds of networks for analysis. It is a 'Swiss army knife' for this kind research, enabling a swift work flow from concept to data to fully-fledged network, ready for SNA and other analysis. Drawing on several key R packages, it can handle large datasets and create very large networks, upwards of a million or more nodes (depending on your computer's resources). The following data sources are currently supported, although in this tutorial we will only be collecting data from Twitter:

1. Facebook
2. YouTube
3. Twitter
4. Instagram (although API access is extremely limited without an approved app)

Installation and setup

First ensure that the *SocialMediaLab* package is installed and loaded.

We also want to install the *magrittr* package, so we can simplify the work flow by using 'verb' functions that pipe together. We will also be using the *igraph* package for network analysis.

The following commands will check if the packages are installed and install them as necessary, then load them.

Note: SocialMediaLab is available as an official package on CRAN, but the latest development version is available on GitHub. We suggest installing the latest version from GitHub, using the code below.

Note: **Recent changes in the `httr` package caused problems for the `twitteR` package. We resolve this using a quick-fix by installing an earlier version of `httr`. However, first we have to install the most recent version of `httr` package, before downgrading it to the earlier version.**

```
## Only run this if you don't have the packages installed!

install.packages("httr")
if (!"devtools" %in% installed.packages()) install.packages("devtools")
require(devtools)
devtools::install_version("httr", version="0.6.0", repos="http://cran.us.r-project.org")

if (!"SocialMediaLab" %in% installed.packages()) {
  devtools::install_github("voson-lab/SocialMediaLab/SocialMediaLab")
}
require(SocialMediaLab)

if (!"magrittr" %in% installed.packages()) install.packages("magrittr")
```

```
require(magrittr)

if (!"igraph" %in% installed.packages()) install.packages("igraph")
require(igraph)
```

```
require(SocialMediaLab)

require(magrittr)

require(igraph)
```

You will also need to get API access for Twitter. You will *not* be able to collect any data until you have acquired API credentials. Step-by-step instructions for obtaining API access are available from the VOSON website.

Twitter data collection and analysis

In this section we will run through how to collect data from Twitter, create networks, and perform different kinds of analysis.

It is currently possible to create 3 different types of networks using Twitter data collected with `SocialMediaLab`. These are (1) *actor* networks; (2) *bimodal* networks; and (3) *semantic* networks. In this session we will create an *actor* and a *semantic* network.

First, define the API credentials. Due to the Twitter API specifications, it is not possible to save authentication token between sessions. The `Authenticate()` function is called only for its side effect, which provides access to the Twitter API for the current session.

```
# REPLACE WITH YOUR API KEY
myapikey <- "xxxx"
# REPLACE WITH YOUR API SECRET
myapisecret <- "xxxx"
# REPLACE WITH YOUR ACCESS TOKEN
myaccesstoken <- "xxxx"
# REPLACE WITH YOUR ACCESS TOKEN SECRET
myaccesstokensecret <- "xxxx"
```

Given that we are going to be creating two different types of Twitter networks (actor and semantic), we will `Collect()` the data, but not pipe it directly through to `Network()` straight away. This means we can reuse the data multiple times to create two different kinds of networks for analysis.

We will collect 50 recent tweets that have used the #auspol hashtag. This is the dominant hashtag for Australian politics.

```
myTwitterData <- Authenticate("twitter",
                              apiKey=myapikey,
                              apiSecret=myapisecret,
                              accessToken=myaccesstoken,
                              accessTokenSecret=myaccesstokensecret) %>%
                 Collect(searchTerm="#auspol",
                              numTweets=50,
                              writeToFile=T,
                              verbose=TRUE)
```

We can have a quick look at the data we just collected:

```
View(myTwitterData)
```

Note the class of the dataframe, which lets `SocialMediaLab` know that this is an object of class `dataSource`, which we can then pass to the `Create()` function to generate different kinds of networks:

```
class(myTwitterData)
```

First, we will create an *actor* network. In this actor network, edges represent interactions between Twitter users. An interaction is defined as a 'mention' or 'reply' or 'retweet' from user i to user j, given tweet m. In a nutshell, a Twitter actor network shows us who is interacting with who in relation to a particular hashtag or search term.

```
g_twitter_actor <- myTwitterData %>% Create("Actor")

# optionally write the graph to file (e.g. to import into Gephi)
write.graph(g_twitter_actor,"demo_session_graph.graphml",format="graphml")
```

We can now examine the description of our network:

```
g_twitter_actor
```

Who are the top 3 important users in our #auspol actor network? There are several ways to do this. We will use the PageRank algorithm implementation in `igraph` to calculate this:

```
pageRank_auspol_actor <- sort(page.rank(g_twitter_actor)$vector,decreasing=TRUE)
head(pageRank_auspol_actor,n=3)
```

What about the 3 least important users (with all due respect...):

```
tail(pageRank_auspol_actor,n=3)
```

Is there any kind of community structure within the user network? As per the previous Facebook analysis we will use the infomap algorithm implementation in `igraph`.

```
imc <- infomap.community(as.undirected(g_twitter_actor), nb.trials = 1) # increase nb.trials for better

# create a vector of users with their assigned community number
communityMembership_auspol <- membership(imc)
# summarise the distribution of users to communities
commDistribution <- summary(as.factor(communityMembership_auspol))
# which community has the max number of users
tail(sort(commDistribution),n=1)

# create a list of communities that includes the users assigned to each community
communities_auspol <- communities(imc)
# look at the members of the most populated community
communities_auspol[names(tail(sort(commDistribution),n=1))]
```

Next, we will create a *semantic* network. In this network nodes represent unique concepts (in this case unique terms/words extracted from a set of 50 tweets), and edges represent the co-occurrence of terms for all observations in the data set. For example, for this Twitter semantic network, nodes represent either hashtags (e.g. "#auspol") or single terms ("happy"). If there are 50 tweets in the data set (i.e. 50 observations), and the hashtag *#auspol* and the term *happy* appear together once in every tweet, then this would be represented by an edge with weight equal to 50.

```
g_twitter_semantic <- myTwitterData %>% Create("Semantic")
```

Let's have a look at the network description:

```
g_twitter_semantic
```

What are the top 10 important terms in our #auspol semantic network? Once again we will calculate this using PageRank:

```
pageRank_auspol_semantic <- sort(page.rank(g_twitter_semantic)$vector,decreasing=TRUE)
pageRank_auspol_semantic[1:10]
```

Next we will import our graphml file into Gephi and make beautiful networks (hopefully).

Conclusion

I hope that you enjoyed this tutorial and that you find *SocialMediaLab* useful in your research. Please feel free to get in contact should you have any questions or comments.

All the best,

Tim Graham