Lab 6: Networks and Influence

Social Media and Web Analytics @ TiSEM

Last updated: 31 May, 2021

Motivation

All social media interactions occur within a network of users who are connected to one and other. In this tutorial you revisit how extract the network relationships from raw Twitter data into and summarize aspects of the network and individual nodes.

Exercise 1 revisits notions from Lab 1, and constructs a mentions network from Twitter Data. Exercise 2 computes summary statistics about a network to help understand the network's size and level of connectivity. Exercise 3 turns to measuring the importance of individual nodes on a network by looking at their connectivity to other nodes. Exercise 4 explores how to detect sub-communities within an network and identify influential users within these communities.

Learning Goals

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

1. Construct Samples from a larger network
2. Compute summary statistics for a given network
3. Define metrics of node importance
4. Analyse the importance of individual nodes based on various measures of node centrality
5. Explain how the infomap and Louvain community detection models work
6. Implement the infomap and Louvain community models
7. Visualize sub-communities within a network.
8. Assess the marketing importance of finding influential nodes and sub-communities.

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-06_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 both the question and answer files using Git. To clone a copy of this repository to your own PC, use the following command:

```
$ git clone https://github.com/tisem-digital-marketing/smwa-lab-06.git
```

Once you have your copy, open the answer document in RStudio as an RStudio 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 class Slack channel `#lab-06-discussion`.

Getting Started: Data & R Packages

This Lab revisits the data on tweets that use the #rstats hashtag that we used in Lab 01.

To gain access to the data, run the following code to download it and save it in the `data` directory:

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

You might need to use the following `R` libraries throughout this exercise:[1]

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

Exercise 1: From Tweets to Networks

In this exercise we will transform the data from the raw data provided by Twitter into a `tidygraph`. Much of the content will be a revision of Lab 01.

1. Load the data into R.

solution

```
tweets <- read_rds('data/rstats_tweets.rds')
```

2. We will construct the network using mentions. Drop all tweets that do not include a mention, and keep only columns that include the author of the tweet and the name(s) of the users that are mentioned.

solution

```
connections <-
    tweets %>%
    filter(mentions_screen_name != 'NA') %>%
    select(from = screen_name, to = mentions_screen_name)
```

3. Transform your data from (2) to be 'tidy'. In particular if multiple users are mentioned in a tweet, there should be one row per username rather than multiple names nested in a single column. Also, drop any occurrences where a user mentions themselves.

solution

```
connections <-
    connections %>%
    unnest_longer(to) %>%
    filter(from != to)
```

4. To keep computation time manageable, we will use a sample of users from the data in (3). To construct this sample, proceed as follows:

(a) Set R's seed to `1234567890`, so that we all get the same answer.

[1] If you haven't installed one or more of these packages, do so by entering `install.packages("PKG_NAME")` into the R console and pressing ENTER.

(b) Sample 250 users from the network. Weight users based on the number of times their `screen_name` appears in your answer from (3), so that those who tweet and mention relatively more often are more likely to be sampled.
(c) Find all the unique usernames that are mentioned by this sample of users.
(d) Find all unique 'mentioner - mentionee' pairs where the author of the tweet is either one of the 250 seed users OR a user identified in (c).

Here's some code to get you started with each part:

```
# (a)
set.seed(YOUR_CODE)

# (b)
seed_users <-
    YOUR_DATA %>%
    # Count the number of times a user name appear as a mentioner
    # if one tweet mentions 2 people, the authors name appears,
    # and will be counted twice -- this is OK
    YOUR_CODE %>%
    # Sample 250 users
    YOUR_CODE(250, weight = YOUR_CODE)

# (c)
seed_connections <-
    YOUR_DATA %>%
    filter(YOUR_CODE %in% YOUR_CODE)

first_step <- unique(YOUR CODE)

# (d)
all_users <- unique(c(YOUR_CODE, YOUR_CODE))

edgelist <-
    YOUR_DATA %>%
    filter(YOUR_CODE) %>%
    distinct()
```

solution

```r
# a
set.seed(1234567890)

#b
seed_users <-
    connections %>%
    count(from) %>%
    sample_n(250, weight = n)

#c
seed_connections <-
    connections %>%
    filter(from %in% seed_users$from)

first_step <- unique(seed_connections$to)

#d
all_users <- unique(c(seed_users$from, first_step))

edgelist <-
    connections %>%
    filter(from %in% all_users,
           to %in% all_users
           ) %>%
    distinct()
```

5. Convert the edge-list you created in (4) to an undirected network object.

solution

```r
tg <-
    as_tbl_graph(edgelist) %>%
    convert(to_undirected)
```

Exercise 2: Network Statistics

With a network in hand, next we turn to describing the network, in terms of its' size and density.

Throughout these questions, you will be asked to provide some definitions of concepts you will use. We encourage you to look for these definitions yourself, and a useful starting point maybe this resource (Ignore any code they provide and focus on the explanations).

1. How many nodes are in the network? Use the function `gorder()` to find the answer.

solution

```r
gorder(tg)
```

```
## [1] 9442
```

2. How many edges are there in the network? Use the function `gsize()` to find the answer.

solution

```
gsize(tg)
```

```
## [1] 53017
```

3. How many possible connections are there in the data? What fraction of these potential edges do we see in the network? Based on your answers, do you think the network is sparsely connected?

solution

```
potential_edges <- (gorder(tg) * (gorder(tg - 1))) / 2

# manually
density <- gsize(tg) / potential_edges
# or using the command suggested
edge_density(tg)
```

```
## [1] 0.001189496
```

4. Define the term 'clustering coefficient' (also called transitivity). What is the clustering coefficient in the data? Use the functions `transitivity()` to find the answer and interpret the result.

solution

```
transitivity(tg)
```

```
## [1] 0.05247316
```

5. Why might a marketing analyst care about the summary statistics you documented above?

Exercise 5: Finding Influential Users

We now turn to measuring centrality of users/nodes in the network. There are alternative measures we could use, so we will explore some of the common ones in the exercise.

As we progress, we will be adding information about about each node's influence. Use and extend the following code to add this information to each node:

```
tg <-
    tg %>%
    activate(nodes) %>%
    mutate(VAR_NAME = FUNCTION())
```

1. Define what a node's degree is.
2. Compute each node's degree using the `centrality_degree()` function.

solution

```
tg <-
    tg %>%
    activate(nodes) %>%
    mutate(degree = centrality_degree())
```

3. Provide intuitive definitions of betweenness centrality, eigenvector centrality and PageRank centrality.

4. Compute each of the measures in (3), using the functions `centrality_betweenness()`, `centrality_eigen()` and `centrality_pagerank()`.

solution

```
tg <-
    tg %>%
    activate(nodes) %>%
    mutate(betweenness = centrality_betweenness(),
           eigen       = centrality_eigen(),
           page_rank   = centrality_pagerank()
           )
```

Let's move these measures into a dataframe that we can explore. Run the following code:

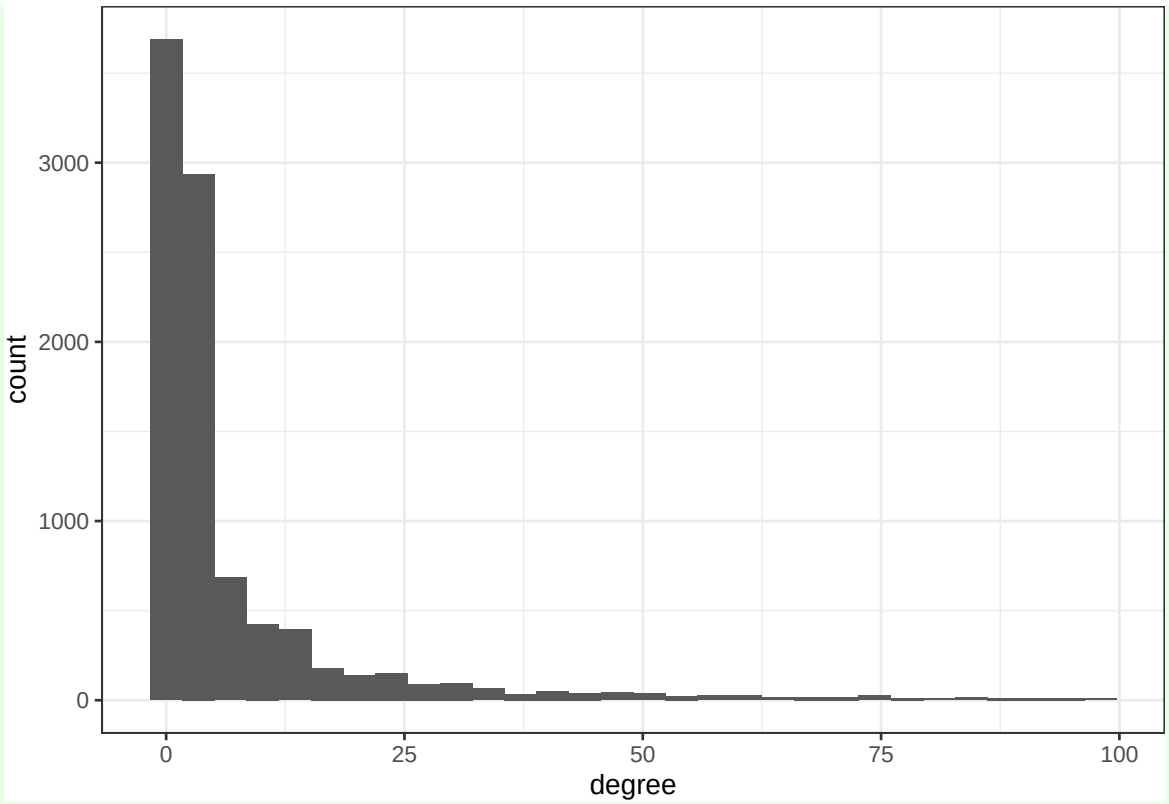
```
centrality_measures <-
    tg %>%
    activate(nodes) %>%
    as_tibble()
```

For each measure of centrality, a higher value means that a node is more influential, although the scale of the metrics are not all the same. Let's explore the measure of influence in our data.

5. Restrict your sample to users that have a degree centrality score score of less than 100. Plot the distribution of degree centrality as a histogram. Describe the pattern you see.

solution

```
centrality_measures %>%
    filter(degree < 100) %>%
    ggplot(aes(x = degree)) +
    geom_histogram(bins = 30) +
    theme_bw()
```



6. Restrict the data to the top 20 nodes based on the PageRank measure of centrality. For each measure of centrality, compute a node's rank compared to others. Use the following code to get started:

```
top_20 <-
    centrality_measures %>%
    arrange(desc(page_rank)) %>%
    head(20) %>%
    mutate(page_rank_rank = dense_rank(page_rank) #,
           # YOUR_CODE
    )
```

solution

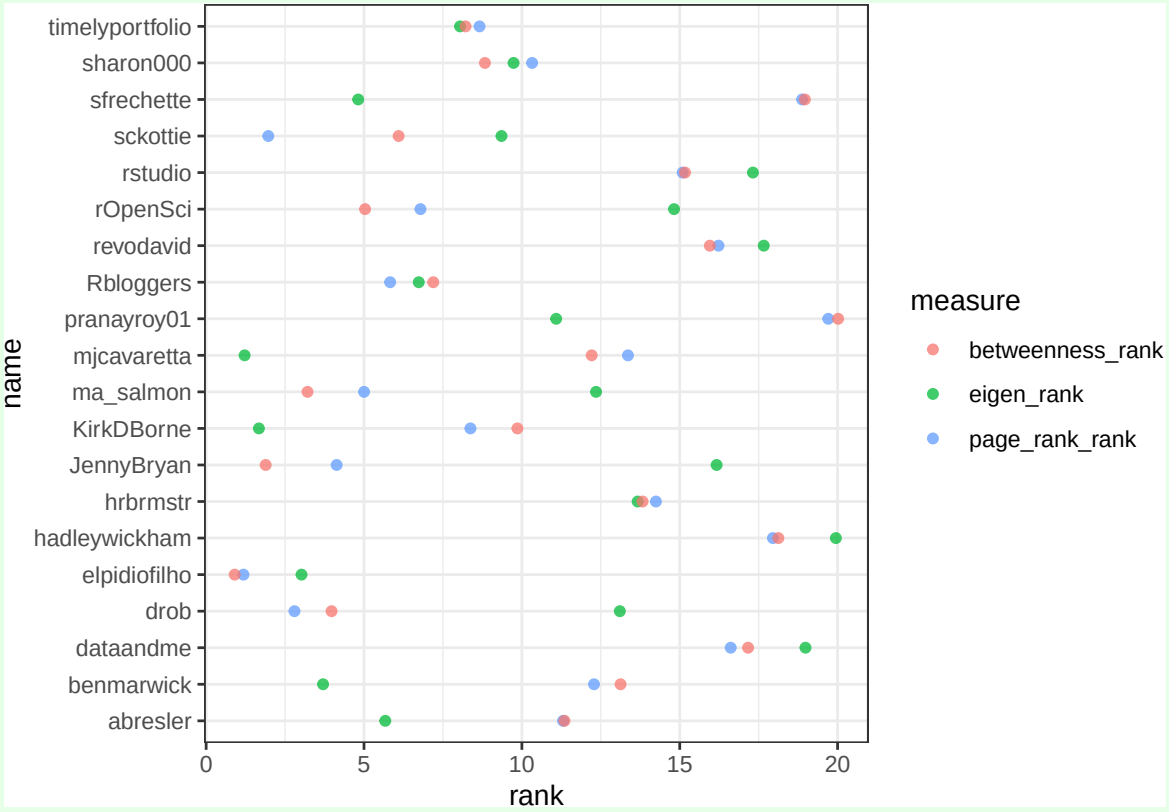
```
top_20 <-
    centrality_measures %>%
    arrange(desc(page_rank)) %>%
    head(20) %>%
    mutate(page_rank_rank = dense_rank(page_rank),
           eigen_rank = dense_rank(eigen),
           betweenness_rank = dense_rank(betweenness)
           )
```

7. Do the rankings yield similar results across alternative measures of centrality? Can you show this in a graph?

solution

```r
# here's the best graph I could quickly put together
top_20_long <-
    top_20 %>%
    select(-c(page_rank, eigen,betweenness)) %>%
    pivot_longer(
        cols = ends_with("rank"),
        names_to = "measure",
        values_to = "rank"
    )

ggplot(data=top_20_long,
        aes(x=name,
            y=rank,
            color = measure)
       )+
    geom_jitter(alpha = 0.75, width = 0) +
    coord_flip() +
    theme_bw()
```



8. As a marketer, what is the value of computing these measures of centrality and ranking users? Could you use these results to kick start some form of a marketing campaign? Describe why you came to your conclusion.

Exercise 4: Grouping Algorithms

Within a network we can group sets of nodes into 'communities' where subsets of nodes have strong inter-relations. In this final exercise we will look at how to implement two common community detection algorithms to find such communities. We can then visualize the communities among the larger network, and look for

influential users within each community

1. The first community detection algorithm we will look at is `infomap`. Do some research online in order to provide an intuitive explanation of how the infomap algorithm works.
2. Implement the infomap community detection algorithm by running the following code:

solution

```
tg <-
    tg %>%
    activate(nodes) %>%
    mutate(grp_info = group_infomap())
```

3. How many communities did the algorithm detect?
4. How large is each community?
5. Visualize the top 5 communities in terms of group size by completing this code:

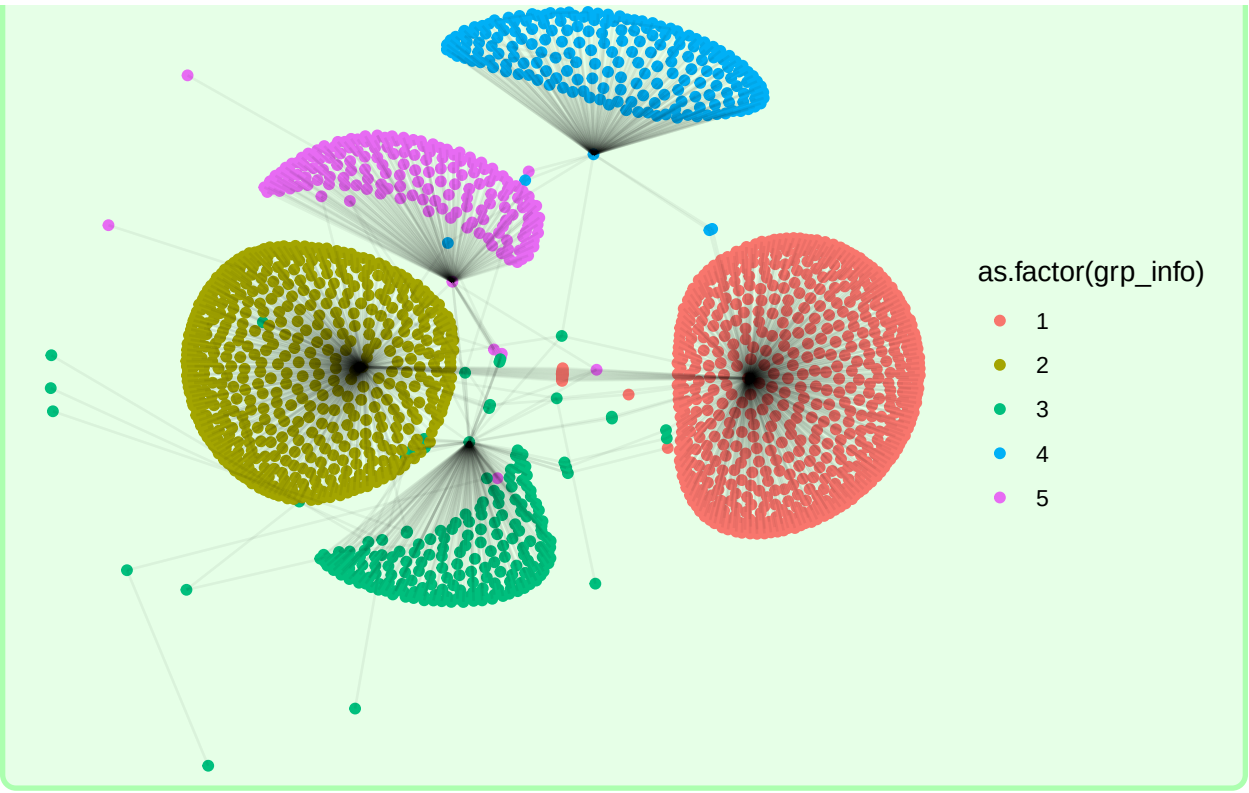
```
tg_5_comm <-
    YOUR_CODE  %>%
    activate(nodes) %>%
    filter(YOUR_CODE)

tg_5_comm %>%
    ggraph(layout = "stress") +
    # Add the nodes to the graph
    YOUR_CODE +
    # Add the edges
    # set alpha = .05 so edges are transparent enough to see the nodes
    YOUR_CODE +
    # this last line of code adds a ggplot2 theme suitable for network graphs
    theme_void()
```

solution

```
tg_5_comm <-
    tg  %>%
    activate(nodes) %>%
    filter(grp_info <= 5)

tg_5_comm %>%
    ggraph(layout = "stress") +
    # this adds the points to the graph
    geom_node_point(aes(color = as.factor(grp_info))) +
    # this adds the links, or the edges;
    # alpha = .2 makes it so that the lines are partially transparent
    geom_edge_link(alpha = .05) +
    # this last line of code adds a ggplot2 theme suitable for network graphs
    theme_void()
```



Let's try a different community detection algorithm, called the Louvain method.

6. Do some research online in order to provide an intuitive explanation of how the Louvain algorithm works.
7. Adapt your code from (2) to identify communities using the Louvain Method. The function you will want to use is `group_louvain()`

solution

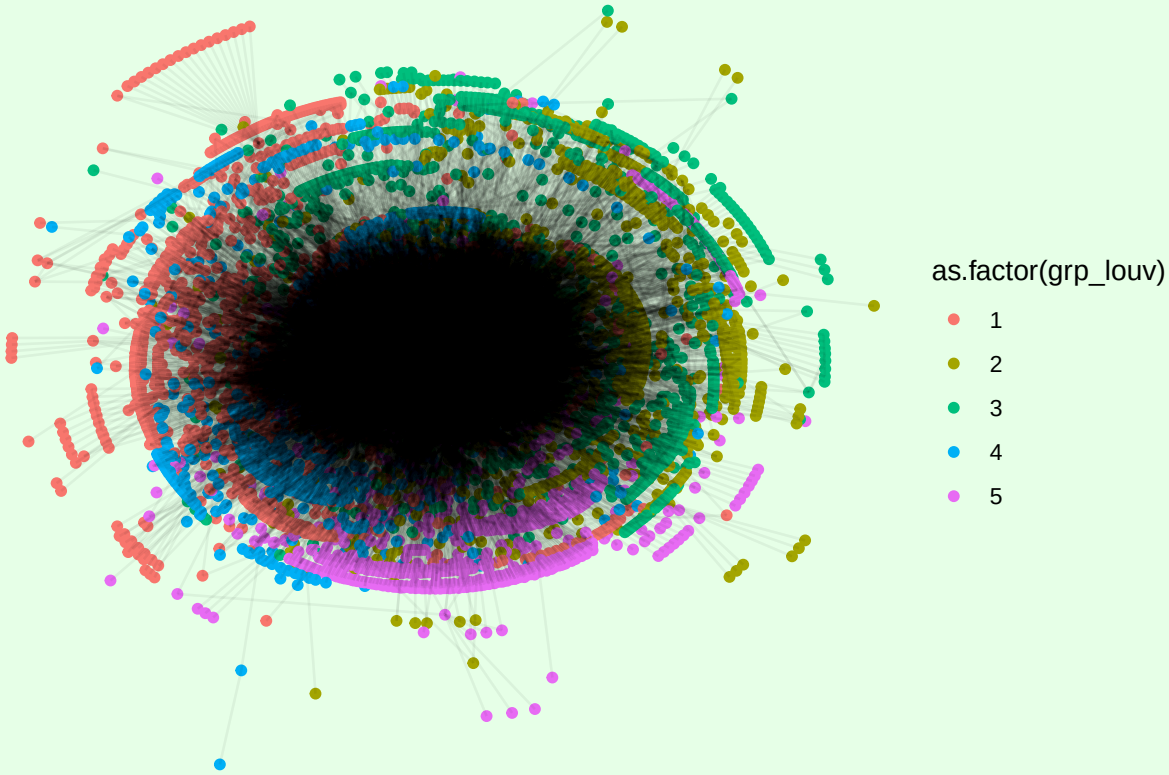
```
tg <-
    tg %>%
    activate(nodes) %>%
    mutate(grp_louv = group_louvain())
```

8. How many communities did the algorithm detect? What are the community sizes of each algorithm?
9. Visualize the top 5 communities identified in (7) by adapting the code you constructed in (5).

solution

```
tg_5_comm <-
    tg  %>%
    activate(nodes) %>%
    filter(grp_louv <= 5)

tg_5_comm %>%
    ggraph(layout = "stress") +
    # this adds the points to the graph
    geom_node_point(aes(color = as.factor(grp_louv))) +
    # this adds the links, or the edges;
    # alpha = .2 makes it so that the lines are partially transparent
    geom_edge_link(alpha = .05) +
    # this last line of code adds a ggplot2 theme suitable for network graphs
    theme_void()
```



10. For each of the top 5 groups we identified using the Louvain model, find the 5 most influential users as measured by PageRank.
11. Explain how identifying communities and influential users within them can be useful in the design of a marketing campaign aimed at using influencers.