Network Analysis

Mykola Chuprynskyy

Contents

Part 1: to be completed at home before the lab

During this practical, we will cover an introduction to network analysis. We cover the following topics:

- metrics to describe network,
- centrality indices,
- community detection,
- network visualization.

We will mainly use the `igraph` package.

You can download the student zip including all needed files for this lab here.

Note: the completed homework has to be **handed in** on BlackBoard and will be **graded** (pass/fail, counting towards your grade for individual assignment). The deadline is two hours before the start of your lab. Hand-in should be a **PDF** file. If you know how to knit pdf files, you can hand in the knitted pdf file. However, if you have not done this before, you are advised to knit to a html file as specified below, and within the html browser, 'print' your file as a pdf file.

For this practical, you will need the following packages:

```r
#install.packages("tidyverse")
library(readr)
library(tidyverse)
library(ggplot2)

#install.packages("igraph")
library(igraph)

#install.packages("RColorBrewer")
library(RColorBrewer)

#install.packages("sbm")
library(sbm)

#install.packages("fossil")
library(fossil)

set.seed(42)
```

We are going to use the `high school temporal contacts` dataset, created in the context of the SocioPatterns project.

The dataset is publicly available in the repository Netzschleuder and it corresponds to the contacts and friendship relations between students in a high school in Marseilles, France, in December 2013. The contacts are measured in four different ways: - with proximity devices (see folder `data/proximity`), - through contact diaries (see folder `data/diaries`), - from reported friendships in a survey (see folder `data/survey`), - from Facebook friendships (see folder `data/facebook`).

Each folder contains the edgelist (`edges.csv`), with the source node, target node, and additional properties (interaction strength or time at which the interaction happened). The `nodes.csv` file contains the node's (students) IDs, the class they belong to, and their gender. In this lab, we will use the `proximity` data, but the same analysis can be repeated for the other three datasets in a similar way.

Reading a network file

1a. **Read the edgelist file (`edges.csv` in the `data/proximity` folder) and store it in the variable `edge_list`. The csv file contains an edge list with two columns: source node (one student), target node (the other student), and the time at which the interaction happened. To load it, use the function `read_csv` from the `readr` package. What is the advantage of storing the network using an edge list, rather than in an adjacency matrix? Is the difference more relevant in the case of sparse or dense networks?**

```
edge_list <- read_csv("data/proximity/edges.csv")
```

```
## Rows: 188508 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (3): source, target, time
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

Answer:

```
# Advantages of Edge List:

# Space Efficiency: Edge lists are more space-efficient, especially for sparse network
# Flexibility: Edge lists can easily include additional information about edges, such
# Relevance to Sparse or Dense Networks:

# The difference is more relevant in the case of sparse networks. For sparse networks,
```

1b. **Read also the node properties file (`nodes.csv` in the `data/proximity` folder) and store it in the variable `node_prop`. What information is in this file?**

```
# Read the node properties
node_prop <- read_csv("data/proximity/nodes.csv")
```

```
## Rows: 329 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): class, gender, _pos
## dbl (2): index, id
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

Answer:

```
# Information in Node Properties File:

# The nodes.csv file contains:

# Node IDs: Unique identifiers for each student.
# Class: The class each student belongs to.
# Gender: The gender of each student.
```

---

2. **Network creation: Create the network using the `edge_list` using the function `graph_from_data_frame` from the `igraph` package. Store it in the variable `g`. If you look for `?graph_from_data_frame` you can find all the arguments of this function. Create an `undirected` graph, and add the node properties by setting the `vertices` argument to the `node_prop` variable**

```
# Create the graph from the edge list and node properties
g <- graph_from_data_frame(d = edge_list, vertices = node_prop, directed = FALSE)
```

Network description I

3a. **Descriptive statistics: count how many students are there per class (using `group_by`), using the `node_prop` variable**

```
# Count the number of students per class
students_per_class <- node_prop %>%
  group_by(class) %>%
  summarise(count = n())
print(students_per_class)
```

```
## # A tibble: 9 x 2
##   class count
##   <chr> <int>
## 1 2BIO1    36
## 2 2BIO2    35
## 3 2BIO3    40
## 4 MP       33
## 5 MP*1     29
## 6 MP*2      38
## 7 PC        44
## 8 PC*       40
## 9 PSI*      34
```

3b. **Descriptive statistics: print (a) the number of nodes (function `vcount`) and (b) the number of edges (function `ecount`) (c) the longest path on the network (`diameter`, setting the variable `unconnected` to false), (d) the average path length (`mean_distance`, setting the variable `unconnected` to false), (e) the global clustering coefficient (`transitivity`).**

```r
# (a) Number of nodes
num_nodes <- vcount(g)
cat("Number of nodes:", num_nodes, "\n")
```

```
## Number of nodes: 329
```

```r
# (b) Number of edges
num_edges <- ecount(g)
cat("Number of edges:", num_edges, "\n")
```

```
## Number of edges: 188508
```

```r
# (c) Longest path (diameter)
diam <- diameter(g, unconnected = FALSE)
cat("Diameter (longest path):", diam, "\n")
```

```
## Diameter (longest path): Inf
```

```r
# (d) Average path length
avg_path_length <- mean_distance(g, unconnected = FALSE)
cat("Average path length:", avg_path_length, "\n")
```

```
## Average path length: Inf
```

```r
# (e) Global clustering coefficient
global_clustering_coef <- transitivity(g, type = "global")
cat("Global clustering coefficient:", global_clustering_coef, "\n")
```

```
## Global clustering coefficient: 0.4444214
```

3c. **How do you interpret the results of the diameter and the average path length? How do you interpret the clustering coefficient?**

```
# Diameter and Average Path Length:

# The diameter indicates the longest shortest path between any two nodes in the networ
# The average path length measures the average number of steps along the shortest path
# Clustering Coefficient:

# The clustering coefficient measures the degree to which nodes in a graph tend to clu
```

Part 2: during the lab

Network manipulaton

4a. **Simplify network: the current network has isolated components (some students do not interact with anybody, maybe because they were sick). Print the number of components (function `components`). Identify the isolated nodes with (`V(g)$name[degree(g) == 0]`) and remove (`delete_vertices`) those nodes from the network. Check again the number of connected components in the new network.**

4b. **Simplify network: the current network has self-loops (i.e. if there is a record of a student's proximity with themselves) and duplicated edges (because students interact several times). Print if there are self-loops (function `any_multiple`), and the number of duplicated edges (function `which_multiple`). We will then remove self-loops (function `simplify`) and collapse all duplicated edges into one weighted edge (`simplify(g, edge.attr.comb = "sum")`. Store the new graph to the variable `g_simple`**

4c. **Check again the number of connected components in the new network. Compute the descriptive statistics for the new network. Did the values change? Why? Is it always a good choice to focus on the largest connected component? Provide an example of research question when this is not the case.**

```
# Number of vertices

# Number of edges

# Diameter (maximum eccentricity)


# Average path length


# Clustering coefficient
```

Answer:

____

____

Network description II

5a. Degree distribution: use function `degree` to store the degree of each node in the network in the variable `degree_dist`. What's the mean degree and its standard deviation? Plot the `degree_dist` as a histogram with ggplot (`geom_histogram`).

```
# Calculate the degree distribution

# Calculate mean and standard deviation

# Create a data frame for the degree distribution

# Plot the degree distribution using ggplot2
```

5b. Degree distribution: compare the degree distribution to the class sizes. How do you interpret these results?

Answer:

____

____

Network visualization

6a. Network Visualization: visualize the simplified network using the function `plot`.

6b. This is a bit ugly, let's assign different colors to school classes

6c. Now, add a layout. We will use a "spring" algorithm for visualization, where nodes that are connected get pushed together, and nodes that are not connected get pushed apart. Store the coordinates of each node (`coords = layout_with_fr(g_simple)`), to plot them in the same position in the next plots (you can experiment with different layout algorithms, see ?layout_).

6d. Make the plot prettier! Play with the `plot` function options (e.g., vertex.size = 5, vertex.label = NA, edge.width = 0.1, edge.arrow.size = 0) until you are happy with the results. Make sure you have a legend

6e. Do students in the same class interact more? Why are there so many connections between different classes? Do you notice any pattern?

Answer:

Centrality measures

7a. Centrality measures: during the lecture we discussed different types of centrality measures, that are useful to quantify the importance of nodes in the network. The most widely used ones are: degree, betweenness, closeness, and pagerank centrality. Explain each measure and how it differs from the others. Compute all the centrality measures (with functions `degree`, `betweenness`, `closeness`, and `page_rank`). Find the most central nodes according to these measures (`which(centrality == max(centrality))`). You can use the pre-made function `calculate_and_print_max_centrality` below. Is the same node the most central node by all definitions?

- Degree:
- Betweenness:
- Closeness:
- Pagerank:

```r
calculate_and_print_max_centrality <- function(graph, centrality_type) {
  # Calculate the specified centrality based on the type provided
  centrality <- switch(centrality_type,
                       "degree" = degree(graph),
                       "betweenness" = betweenness(graph),
                       "closeness" = closeness(graph),
                       "pagerank" = page_rank(graph)$vector,
                       stop("Invalid centrality type"))

  # Find the node(s) with the highest centrality
  max_value <- max(centrality)
  highest_nodes <- which(centrality == max_value)

  # Print results
  cat(sprintf("Node id(s) with highest %s centrality: %s\n", centrality_type, toString(
}
```

7b. Let's label those nodes. You can again use the `plot` function to create the plot and set `vertex.label = labels`. Also, set `vertex.label.size=1000` to be able to see the labels

```
# Conditional labeling of nodes remove the comment
# labels <- ifelse(V(g_simple)$name %in% c("39", "318"), V(g_simple)$name, NA)

# Plot the graph with selective labeling

# Adding a legend
```

Community detection

8a. Community detection: we would like to detect communities in the network. Let's start with the `cluster_leiden` function, which creates communities that maximize a metric (either CPM or modularity). Create the communities using modularity (see ?cluster_leiden) and store the results in a variable (e.g. `modularity`).

8b. Now, plot the network with the communities. You can just visualize the network using the function `plot(commdet, g_simple)`, where `commdet` may be substituted with the variable name where you stored the community detection results. Remember to fix `layout = coords` to plot the nodes always in the same position.

8c. How do the communities align with the classrooms (question 6)?

Answer:

9a. Community detection: There are other methods implemented in igraph: `cluster_spinglass`, based on the spinglass model from statistical mechanics; `cluster_walktrap`, based on random walks; and `cluster_label_prop`, based on labeling all nodes and then updating the labels by majority voting. We will also employ the `SBM` (Stochastic Block Model) from the homonym package. Use these methods to find communities (e.g. `cluster_spinglass(g_simple)`). Store the results for each method in different variables. There are more methods available in `igraph`, feel free to try others. For the SBM we provide the code below. It may take a few minutes.

```
## Remove the comments to get the code running
## adj <- as_adjacency_matrix(g_simple, sparse=FALSE)
## simple_sbm <- estimateSimpleSBM(adj, 'bernoulli', estimOptions = list(verbose = 0,
```

9b. **Compare communities visually: let's visualize the network as in the modularity case above. For SBM we provide code below.**

9c. **How do the community assignments produced by the algorithms comparing to the division of students in classrooms (question 6)?**

Answer:

10a. **EXTRA QUESTION: Compare the similarity between the communities detected and the classrooms. We will use the Rand Index, a measure that quantifies the similarity between two data clusterings by considering all pairs of elements and checking whether the pair is either assigned to the same or different clusters in both clusterings. Compute the Rand Index between the classroom assignment (`as.integer(factor(V(g_simple)$class))`), which we consider now as ground truth, and the community assignment (`commdet$membership`) for each method. Use the function `rand.index` from the `fossil` package. Give a formal definition of the Rand Index. Which method(s) is better capturing the "true" subdivision?**

Answer:

10b. **There is a corrected-by-chance version of the Rand Index called Adjusted Rand Index (`adj.rand.index(group1, group2)`). Give the definition and repeat the same done for the RI. Are the results different? Which method is best?**

Answer: