Prep7S24

Sofia Guttmann

2024-03-29

Reminder: Prep assignments are to be completed individually. Upload a final copy of the .Qmd and renamed .pdf to your private repo, and submit the renamed pdf to Gradescope before the deadline (Sunday night, 3/31/24, by midnight).

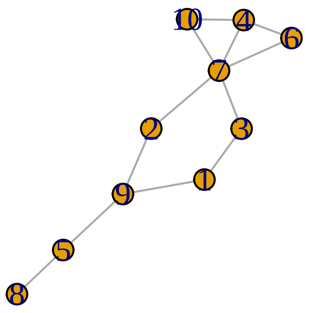
Reading

The associated reading for the week is Chapter 20 on networks and Chapter 17 on spatial data. For Chapter 17, the focus is on working with spatial data, which basically means, making maps and showing appropriate data on them.

1 - Basic Graph Concepts with a Toy Graph

Use the following generated graph to answer the questions that follow. Do NOT change the seed.

```r
set.seed(231)
g <- erdos.renyi.game(n = 10, p = 0.3)
plot(g)
```



part a - How many vertices does the graph, g, have? How many edges?

Solution: 10 vertices and 14 edges.

part b - Compute the diameter of the graph. Then identify a path with that length.

Hint: More than one path may have a length equal to the diameter. Remember that diameter is not the longest path possible. It is the longest of all the shortest paths. To identify a path, list the vertices involved such as 1-2-3 (this is not an actual path in this graph).

Solution:

diameter= 3

path: 1-3-5-10

part c - Compute the degree for vertex 7 without using an R command. Explain what this number represents.

Solution: Vertex 7 is connected to vertex 2. Vertex 7 is connected to vertex 3. Vertex 7 is connected to vertex 9. Thus, the degree of vertex 7 is 3. The degree of a vertex represents the number of edges incident to that vertex. In this context, the degree of vertex 7 tells us that it is directly connected to three other vertices in the graph.

part d - Looking at the plot of the graph, identify a vertex triple - three vertices which are connected by edges. Is your selected vertex triple closed - i.e. is it a triangle?

This is asking you to find a set of vertices that could form a triangle - at least 2 of the 3 potential edges should be there. Determining the fraction of actual triangles out of possible triangles is a measure of what is called the clustering present in the network.

You can list your triple with node numbers. E.G. 1-2-8 (not an actual triple in this graph). The idea would be that 1-2 is an edge and 2-8 is an edge. If 1-8 is also an edge, this would be a triangle.

Solution:

The vertex triple is 1-2-4. Vertex 1 is connected to vertex 4. The selected vertex triple 1-2-4 forms a closed triangle in this graph.

> part e - Imagine you have to walk along this graph. Walking around, what vertices are you most likely to visit often if you move around randomly? (What vertices help connect others a lot?)

No computations necessary here - you don't need to solve this formally - just think through it and give a guess. If you really want, you could look up a formal definition for a random walk on a graph. Basically, at any vertex, you have an equal probability of going to any vertices it has an edge to. For example, in this graph, if you were at vertex 3, you'd have a 50% chance of going to vertex 1 and 50% chance of going to vertex 7.

Solution:

Vertices 1, 4, and 10 appear to have relatively high degrees based on their connections in the plot. Therefore, it's likely that these vertices would be visited more often during a random walk compared to vertices with fewer connections.

> part f - Compute betweenness centrality for all vertices. Identify the top 3 vertices in terms of betweeness. How do these vertices relate to those you listed in part e?

Solution: The vertices with the highest betweenness centrality in this graph are 1 and 10, which were also among those predicted to be frequently visited in part e. Vertex 2, which has the second-highest betweenness centrality, is also a high-degree vertex, consistent with our prediction. This suggests that vertices with high betweenness centrality are indeed important for connecting other vertices in the graph and would likely be visited frequently during a random walk.

```r
# Compute betweenness centrality for all vertices
betweenness_values <- betweenness(g)

# Identify the top 3 vertices in terms of betweenness
top_betweenness <- sort(betweenness_values, decreasing = TRUE)[1:3]

# Print the top 3 vertices with their betweenness centrality values
```

```r
print("Top 3 vertices in terms of betweenness:")
```

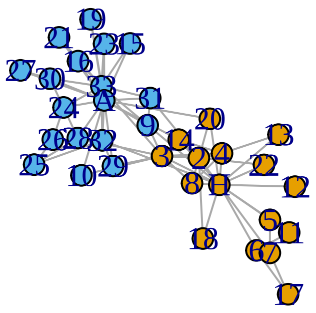
```
[1] "Top 3 vertices in terms of betweenness:"
```

```r
for (vertex in names(top_betweenness)) {
  print(paste("Vertex:", vertex, "- Betweenness centrality:", top_betweenness[vertex]))
}
```

2 - Exploring a Social Network

For this question, we'll explore a famous social network example known as "Zachary's Karate Club". This is information about a Karate club from 1970-1972 that split into 2 factions/groups based on a rift between members denoted "A" for "John A" and "H" for "Mr Hi" in the data set (these are not real names). Let's investigate a little bit!

```
# Visualize the network
# Igraph default plot, may not be "pretty"
data(karate)
karate <- upgrade_graph(karate) # used due to changes in igraph
plot(karate)
```



part a - Plot and describe the degree distribution of the karate network.

Hint: ggplot2 requires data to be in a data.frame to plot, so the code below will get you the degree distribution in data set to use, along with faction information for later use.

```
faction <- get.vertex.attribute(karate)$Faction
kdegree <- degree(karate)
karatedata <- data.frame(kdegree, faction)
```

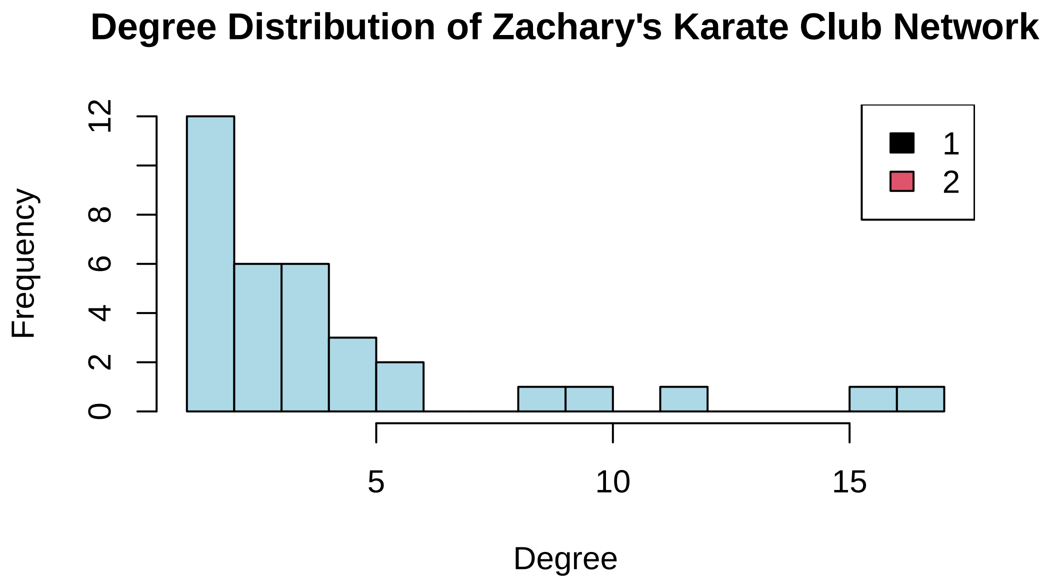
Solution:

```
# Extract faction attribute from vertices
faction <- get.vertex.attribute(karate)$Faction

# Compute degree of vertices
kdegree <- degree(karate)

# Create a data frame containing degree and faction information
karatedata <- data.frame(kdegree, faction)
```

```r
# Plot degree distribution by faction
histogram <- hist(karatedata$kdegree, breaks = 20, main = "Degree Distribution of Zachary'
                  xlab = "Degree", ylab = "Frequency", col = "lightblue", border = "black"
legend("topright", legend = unique(karatedata$faction), fill = unique(karatedata$faction))
```



```r
# Describe degree distribution
mean_degree <- mean(karatedata$kdegree)
median_degree <- median(karatedata$kdegree)
max_degree <- max(karatedata$kdegree)
min_degree <- min(karatedata$kdegree)

cat("Mean Degree:", mean_degree, "\n")
```

```
Mean Degree: 4.588235
```

```r
cat("Median Degree:", median_degree, "\n")
```

```
Median Degree: 3
```

```r
cat("Maximum Degree:", max_degree, "\n")
```

```
Maximum Degree: 17
```

```
cat("Minimum Degree:", min_degree, "\n")
```

```
Minimum Degree: 1
```

> part b - Identify the individuals with the top 5 highest degree values. Do they include "John A" and "Mr Hi"?

Solution: Yes, they do

```
# Sort the data frame by degree in descending order
sorted_data <- karatedata[order(-karatedata$kdegree), ]

# Select the top 5 individuals with the highest degree values
top_5_degrees <- sorted_data[1:5, ]

# Print the top 5 individuals with their degree values
print(top_5_degrees)
```

```
         kdegree faction
John A        17       2
Mr Hi         16       1
Actor 33      12       2
Actor 3       10       1
Actor 2        9       1
```

> part c - Determine the eigenvector centrality of all vertices. Identify the individuals with the top 5 eigenvector centrality values. Do they include "John A" and "Mr Hi"?

Hint: The igraph command is eigen_centrality.

Solution:

```
# Compute eigenvector centrality
eigenvector_centrality <- eigen_centrality(karate_network)$vector

# Combine eigenvector centrality values with node names
eigenvector_data <- data.frame(node = V(karate_network)$name, eigenvector_centrality)

# Sort the data frame by eigenvector centrality in descending order
```

```r
sorted_eigenvector_data <- eigenvector_data[order(-eigenvector_data$eigenvector_centrality

# Select the top 5 individuals with the highest eigenvector centrality values
top_5_eigenvector <- sorted_eigenvector_data[1:5, ]

# Print the top 5 individuals with their eigenvector centrality values
print(top_5_eigenvector)
```

part d - We investigated k-means as a method of clustering observations to find natural groups. Clustering can be performed on networks, though very different methods are needed to do so. Run the code below to perform a clustering analysis on this network. Then use the provided output to assess whether the clusters found match the provided factions assigned in the network by the researcher who studied the karate club. What did you find? Describe the findings in a few sentences.

Note: This clustering analysis may not find just 2 clusters.

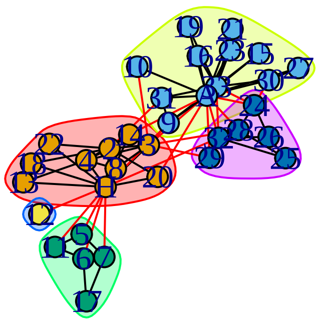
```r
# run clustering on karate network
# this is just one example clustering algorithm for a network
kclusters <- leading.eigenvector.community(karate)
# number of clusters
length(kclusters)
```

```
[1] 5
```

```r
# size of each cluster
sizes(kclusters)
```

```
Community sizes
 1  2  3  4  5
10 12  5  1  6
```

```r
#how to plot the solution
plot(kclusters, karate)
```



```
# Get cluster memberships to compare to faction
karatedata <- karatedata %>%
  mutate(clusters = as.numeric(membership(kclusters)))
mosaic::tally(clusters ~ faction, data = karatedata)
```

```
        faction
clusters  1  2
       1 10  0
       2  0 12
       3  5  0
       4  1  0
       5  0  6
```

Solution:

3 - Spatial data basics

Chapter 17 introduces a number of spatial data specific terms.

> part a - What are two examples of spatial data structure formats?

Solution:

> part b - What does `EPSG` stand for in `EPSG codes`? Why is it important to know the `EPSG` number, if applicable for your map?

Solution:

> part c - What is the primary package used in the textbook to work with spatial data?

Solution:

> part d - What term/concept did you find hardest to understand from the reading?

Solution:

4 - Reproducing a map

Section 17.1 introduces *shapefiles*, and includes an example of working with a shapefile to re-create Snow's cholera map. This exercise mostly follows along with the text code.

Setup

Load the **sf** package in the `setup` code chunk. Verify your working directory is the folder *this* .Rmd file is in. Then, create a *data* subfolder in this directory.

part a - Run the code below line-by-line to understand what each part is doing. You can use *command + enter* or *ctrl + enter* to run one selected or highlighted set of code at a time. Confirm that you get a figure similar to that of Figure 17.3 in the textbook.

```
# Download SnowGIS_SHP zip file
download.file("http://rtwilson.com/downloads/SnowGIS_SHP.zip",
              destfile = "data/SnowGIS_SHP.zip")

# Unzip file in same folder
unzip(zipfile = "data/SnowGIS_SHP.zip",
      exdir = "data")
```

```
# Create filepath to unzipped files so we don't need to re-type
data_path <- "data/SnowGIS_SHP"

# List files in SnowGIS_SHP
list.files(data_path)

# List layers
st_layers(data_path)

# Load second layer
cholera_deaths <- st_read(data_path, layer = "Cholera_Deaths")

class(cholera_deaths)
head(cholera_deaths)

# Context-less plot
ggplot(cholera_deaths) +
  geom_sf()
```

Solution:

part b - Now use the **ggspatial** package to overlay the London street map. Make sure you (install then) load the **ggspatial** package in the `setup` code chunk before running the code. What is wrong with this map?

```r
# if you get an error that a package is missing, you may
# need to install some dependencies
ggplot(cholera_deaths) +
  annotation_map_tile(type = "osm", zoomin = 0) +
  geom_sf(aes(size = Count), alpha = 0.7)
```

Solution:

part c - Set the coordinates from the cholera data as the `espg:27700` coordinate system using `st_set_crs()`, then transform them to the `espg:4326` system using `st_transform()`, and finally plot the new, correctly projected data. What does "crs" stand for in this code?

```r
cholera_latlong <- cholera_deaths %>%
  st_set_crs(27700) %>%
  st_transform(4326)

ggplot(cholera_latlong) +
  annotation_map_tile(type = "osm", zoomin = 0) +
  geom_sf(aes(size = Count), alpha = 0.7)
```

Solution:

part d - Repeat the layer loading and coordinate transformation procedure to add the water pumps to the plot. Looking at your plot, do you agree that the water pump on Broad Street seems to have been the problem for this outbreak?

```r
pumps_latlong <- st_read(data_path, layer = "Pumps") %>%
  st_set_crs(27700) %>%
  st_transform(4326)

ggplot(cholera_latlong) +
  annotation_map_tile(type = "osm", zoomin = 0) +
  geom_sf(aes(size = Count), alpha = 0.7) +
  geom_sf(data = pumps_latlong, size = 3, color = "red")
```

Solution:

part e - Finally, try out the code below to create a dynamic map using the **leaflet** package (install and load as before!). Zoom in and out of the map to confirm that (1) there is a death in the middle of Hopkins Street, and (2) there is a pump near the intersection of Broadwick Street and Lexington Street. What seem to be the main types of businesses along the modern day Kingly street (look left of the pump)?

```
# create dynamic map
leaflet() %>%
  addTiles() %>%
  addCircleMarkers(data = cholera_latlong,
                   radius =  ~ Count,
                   color = "navy",
                   stroke = FALSE,
                   fillOpacity = 0.7) %>%
  addCircleMarkers(data = pumps_latlong,
                   radius = 6,
                   color = "red",
                   stroke = FALSE,
                   fillOpacity = 0.7)
```

Solution: