Network Science Homework 1

Arghya Kannadaguli (ak5357)

2025-01-30

Assignment Description: Find a real-world network and perform an analysis of basic characteristics of the network, including degree distribution, diameters, clustering coefficient, number of connected components, size of the largest component, etc.

```
library(igraph)
library(tidyverse)
library(ggplot2)
```

About the Data:

Physicians *Network Dataset*

"This directed network captures innovation spread among 246 physicians in for towns in Illinois, Peoria, Bloomington, Quincy and Galesburg. The data was collected in 1966. A node represents a physician and an edge between two physicians shows that the left physician told that the right physician is his friend or that he turns to the right physician if he needs advice or is interested in a discussion. There always only exists one edge between two nodes even if more than one of the listed conditions are true."

Data Source: The Konnect Project (http://konect.cc/networks/moreno_innovation/)

Data Import

Find a real-world network.

```
physician_edgelist =
  readr::read_delim("data/moreno_innovation/out.moreno_innovation_innovation", skip = 1) |>
  janitor::remove_empty(which = "cols") |>
  janitor::clean_names() |>
  rename(node1 = percent, node2 = x1098)

physician_graph =
  as.matrix(physician_edgelist) |>
  graph_from_edgelist()
```

Basic Analysis

Perform an analysis of basic characteristics of the network.

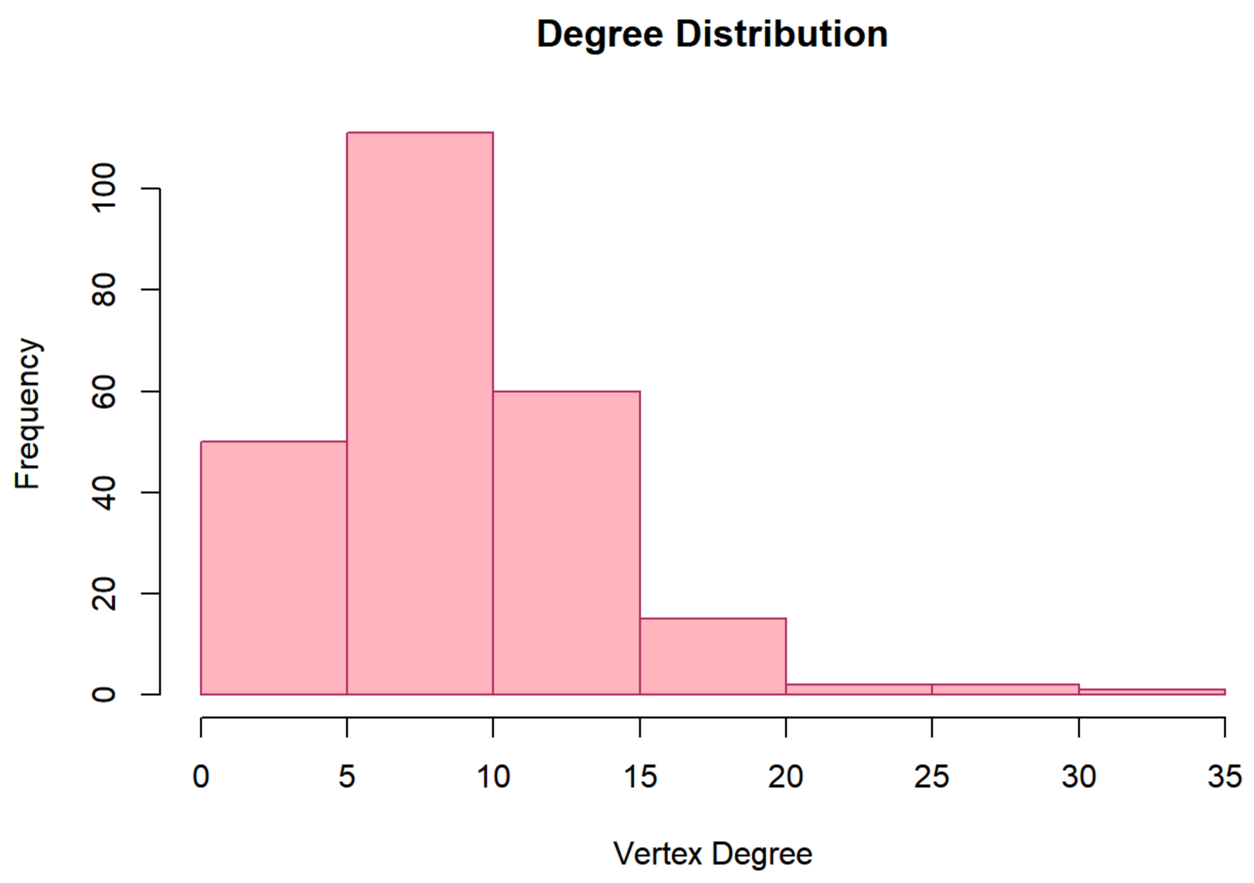
```
n_edges = ecount(physician_graph)
n_nodes = vcount(physician_graph)
```

The physicians network has **241 nodes** and **1098 edges**.

Degree distribution

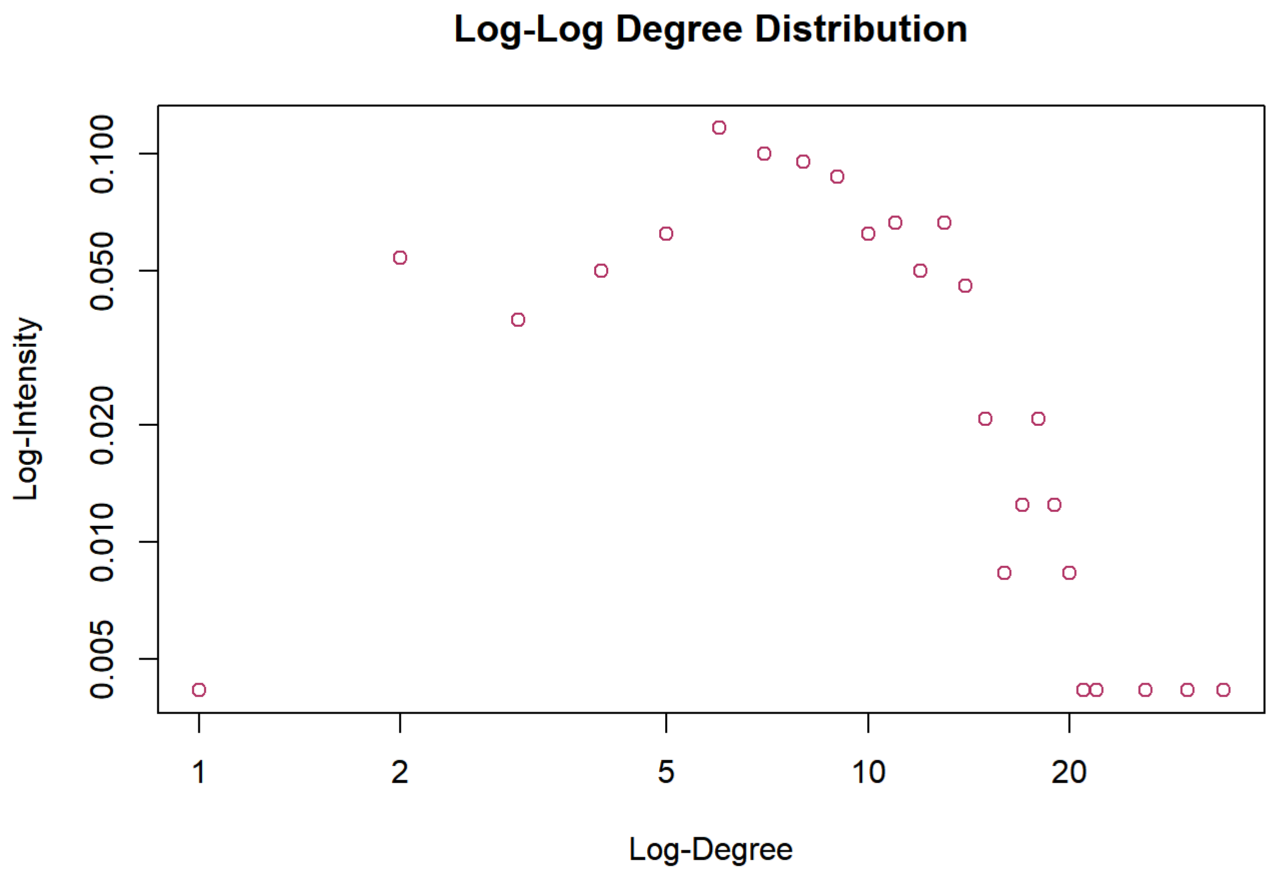
Histogram

```
physician_degrees = degree(physician_graph)
hist(physician_degrees,
     col = "lightpink",
     border = "maroon",
     xlab = "Vertex Degree",
     ylab = "Frequency",
     main = "Degree Distribution"
     )
```

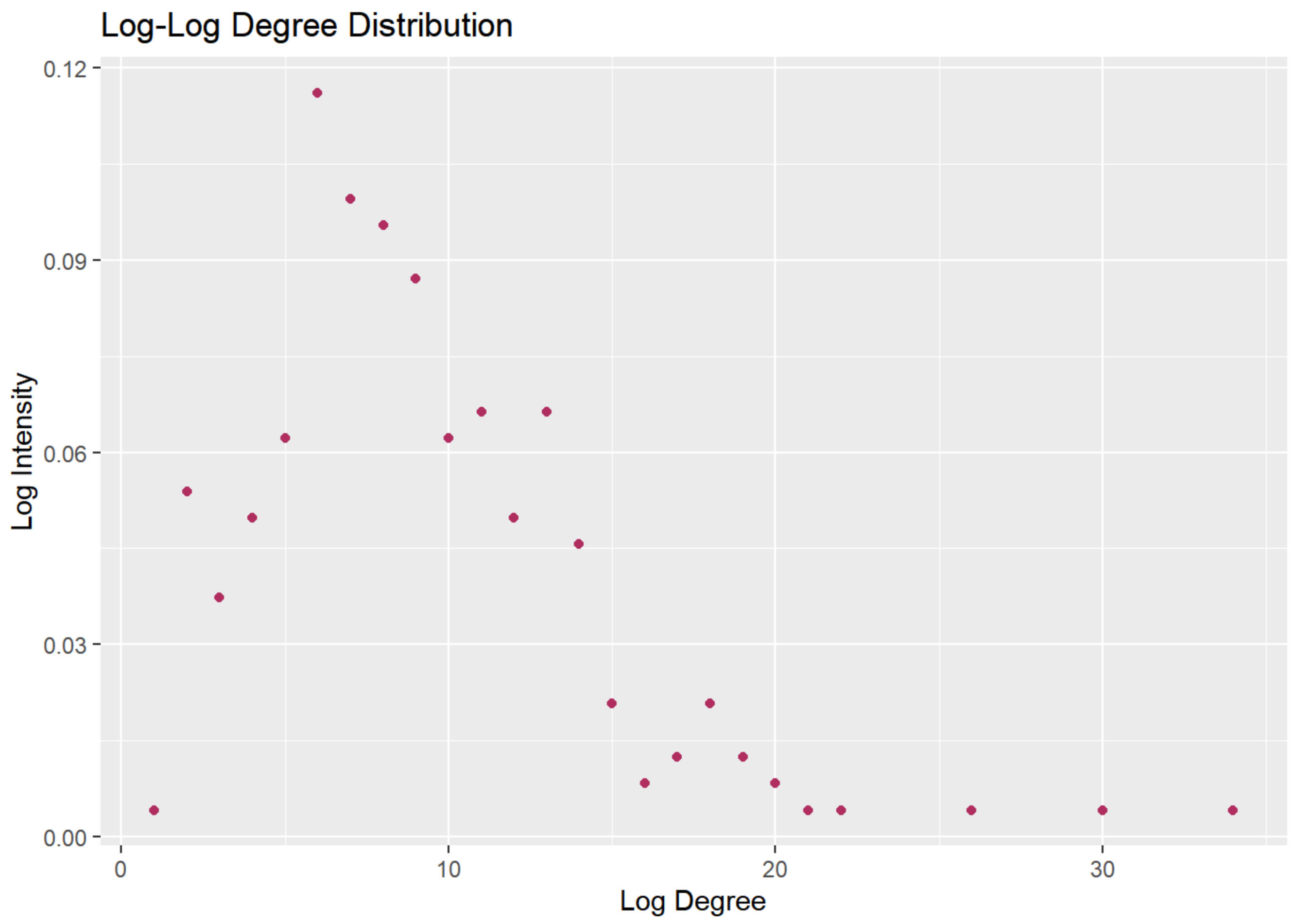


Plotting Degree Distribution in a log-log scale

```r
# Get the degree distribution
physician_dd = degree_distribution(physician_graph)

# Get the range of degrees, starting from 0
degree_range = 0:max(physician_degrees)

# Logical vector of non-zero-frequency degree values
ind = (physician_dd != 0)

# Generate base-R plot
plot(
  degree_range[ind],
  physician_dd[ind],
  log = "xy",
  col = "maroon",
  xlab = "Log-Degree",
  ylab = "Log-Intensity",
  main = "Log-Log Degree Distribution"
)
```



```
# Generate ggplot for fun
tibble(
  `Log Degree` = degree_range[ind],
  `Log Intensity` = physician_dd[ind]
) |>
  ggplot(aes(x = `Log Degree`, y = `Log Intensity`)) +
  geom_point(col = "maroon") +
  labs(
    title = "Log-Log Degree Distribution"
  )
```

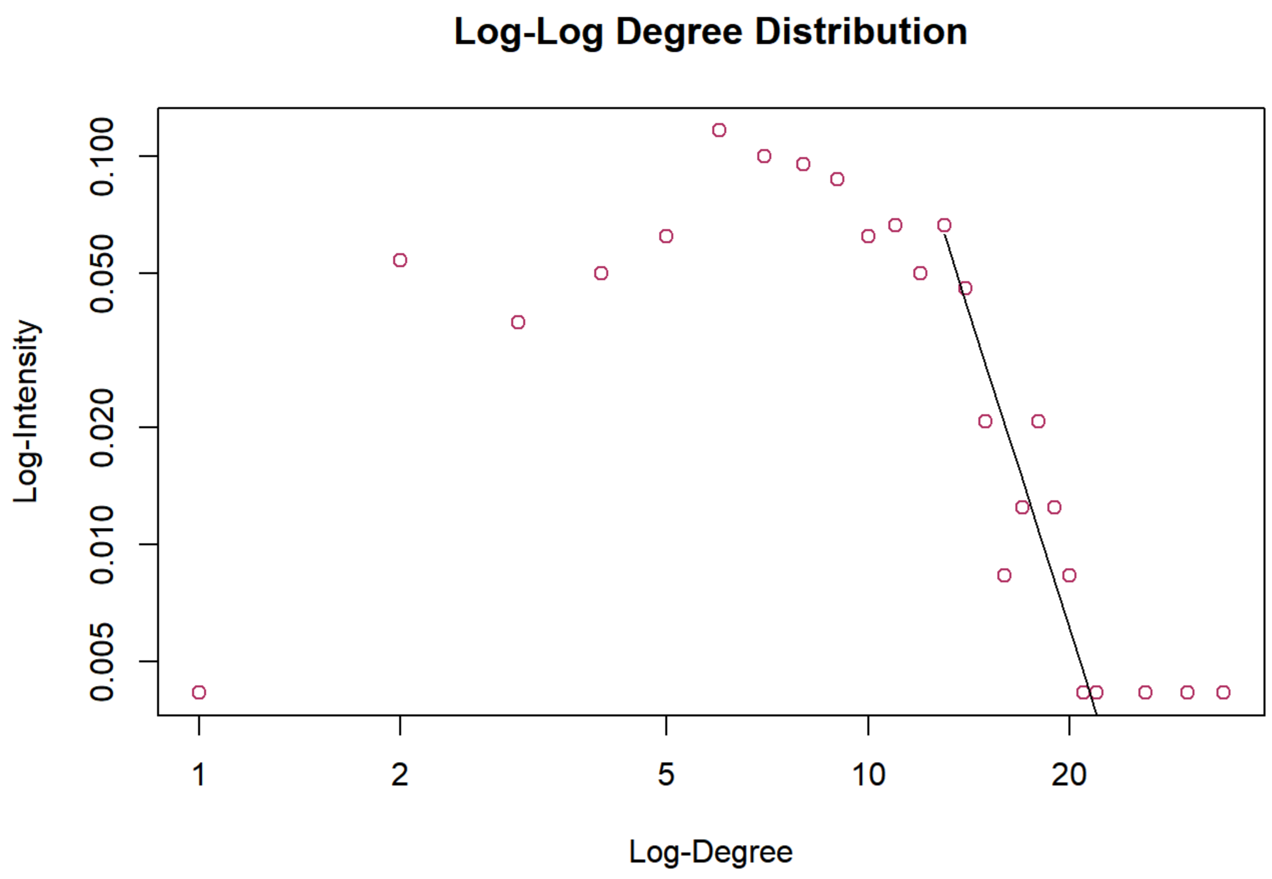


Fit a Power-Law Distribution

```
fit = fit_power_law(physician_degrees)
alpha = fit$alpha
xmin = fit$xmin
xmax = max(physician_degrees)
```

The power-law exponent is **5.415**. The xmin is **13** and xmax is **34**.

```
# Add a line to the Log-Log plot
x = seq(from = xmin, to = xmax)
y = x^(-alpha)

# Normalization
y = y/sum(y) *(sum(physician_degrees>=xmin)/length(physician_degrees))

# Replot with power-fit line
plot(
  degree_range[ind],
  physician_dd[ind],
  log = "xy",
  col = "maroon",
  xlab = "Log-Degree",
  ylab = "Log-Intensity",
  main = "Log-Log Degree Distribution"
)

# Add power-law fit line
lines(x, y)
```



Number of Connected Components

```
# Check if network is connected
is_connected(physician_graph)
```

```
## [1] FALSE
```

The network is not fully connected.

```
# Find all components
comps = decompose(physician_graph)

# How many components
n_comps = length(comps)

# Compute sizes of all components
comps_info_df = comps |>
  sapply(vcount) |>
  tibble() |>
  rename(comp_size = `sapply(comps, vcount)`) |>
  arrange(-comp_size) |>
  mutate(
    id = row_number(),
    comp_frc = paste0(comp_size, "/", n_nodes),
    comp_pct = 100*comp_size/n_nodes) |>
  relocate(id)
```

There are **4 components** in the Physicians network. This makes sense because Konnect's description of the Physicians network mentions that the physicans are located in four separate cities. Perhaps the four cities each form one of the components. The sizes of the components are listed below. The largest component has **117 nodes**.

```
comps_info_df |>
  knitr::kable(
    col.names = c("id", "Component Size", "Fraction of Total Nodes", "Percent (%) of Total Node
s"),
    align = "r",
    digits = 1)
```

id	Component Size	Fraction of Total Nodes	Percent (%) of Total Nodes
1	117	117/241	48.5
2	48	48/241	19.9
3	41	41/241	17.0
4	35	35/241	14.5

Diameters

Find the average shortest path length in each component.

```
comps_info_df = comps_info_df |>
  mutate(
    mean_dist = map(comps, mean_distance),
    diameter = map(comps, diameter)
    ) |>
  unnest(c(mean_dist, diameter))
```

The table below shows the mean distance and diameter of each of the 4 components.

```
comps_info_df |>
  select(-c(comp_frc, comp_pct)) |>
  knitr::kable(
    col.names = c("id", "Component Size", "Mean Distance", "Diameter"),
    align = "r",
    digits = 2)
```

id	Component Size	Mean Distance	Diameter
1	117	3.54	9
2	48	2.69	6
3	41	2.71	6
4	35	2.37	5

Clustering coefficient

The clustering coefficient (or transitivity) is defined as

$$C = \frac{(\text{number of triangles}) \times 3}{(\text{number of connected triples})}.$$

Compute the clustering coefficients of the physicians network.

```
physician_cc = transitivity(physician_graph)
physician_cc
```

```
## [1] 0.2511837
```

The clustering coefficient for this network is **0.251**. This means that **25.1%** of connected triples in the Physicians network are closed.

Assortativity

A network is assortative if a significant fraction of the edges run between nodes of the same type, ie. nodes with similar amounts of edges. A “significant fraction” means that the fraction of edges between same-type nodes is significantly greater than what we would expect if the edges were randomly placed.

```
physician_asrt = assortativity_degree(physician_graph)
physician_asrt
```

```
## [1] -0.0208696
```

The assortativity coefficient for the Physician network is **-0.021**. Since this value is negative, we can say that *dis*similar nodes tend to connect to each other in this network more than they would if the edges were randomly generated.