Project 2

Logan Bolton

2025-02-08

Setup

```
print("Data structure:")
```

```
## [1] "Data structure:"
```

```
str(D)
```

```
## 'data.frame':    13288 obs. of  1 variable:
##  $ X0.4...weight...0.002105263157894737.: chr  "0 12 {'weight': 0.002105263157894737}" "0 18 {'weigh
```

```
print("First few rows:")
```

```
## [1] "First few rows:"
```

```
head(D)
```

```
##   X0.4...weight...0.002105263157894737.
## 1 0 12 {'weight': 0.002105263157894737}
## 2 0 18 {'weight': 0.002105263157894737}
## 3 0 25 {'weight': 0.004210526315789474}
## 4 0 30 {'weight': 0.002105263157894737}
## 5  0 46 {'weight': 0.00631578947368421}
## 6 0 55 {'weight': 0.002105263157894737}
```

```
# Load the library
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
library(stringr)
```

```
# Format the data
edges_df <- data.frame(
  from = as.numeric(sub("^(\\d+).*", "\\1", D$X0.4...weight...0.002105263157894737.)),
  to = as.numeric(sub("^\\d+\\s+(\\d+).*", "\\1", D$X0.4...weight...0.002105263157894737.)),
  weight = as.numeric(sub(".*'weight':\\s*([0-9.]+).*", "\\1", D$X0.4...weight...0.002105263157894737.))
)
```

```r
# Create the graph
g <- graph_from_data_frame(edges_df, directed = TRUE)
```

Graph Characteristics

Network Understanding

```r
print("Network Order (number of vertices):")
```

```
## [1] "Network Order (number of vertices):"
```

```r
vcount(g)
```

```
## [1] 475
```

```r
# Network size (number of edges)
print("Network Size (number of edges):")
```

```
## [1] "Network Size (number of edges):"
```

```r
ecount(g)
```

```
## [1] 13288
```

```r
# Network density
print("Network Density:")
```

```
## [1] "Network Density:"
```

```r
edge_density(g)
```

```
## [1] 0.05901843
```

```r
# Check strong connectivity
components <- components(g, mode="strong")
cat("\nNumber of strongly connected components:", components$no, "\n")
```

```
##
## Number of strongly connected components: 7
```

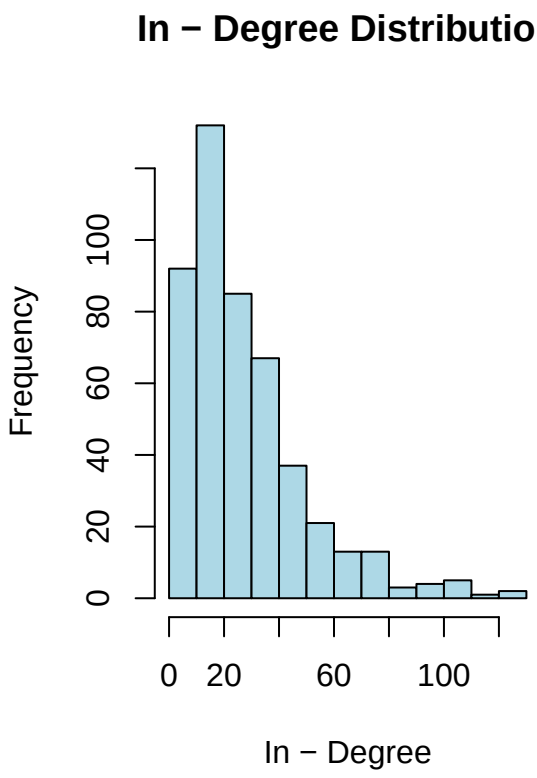
```r
cat("Size of largest strongly connected component:", max(components$csize), "\n")
```

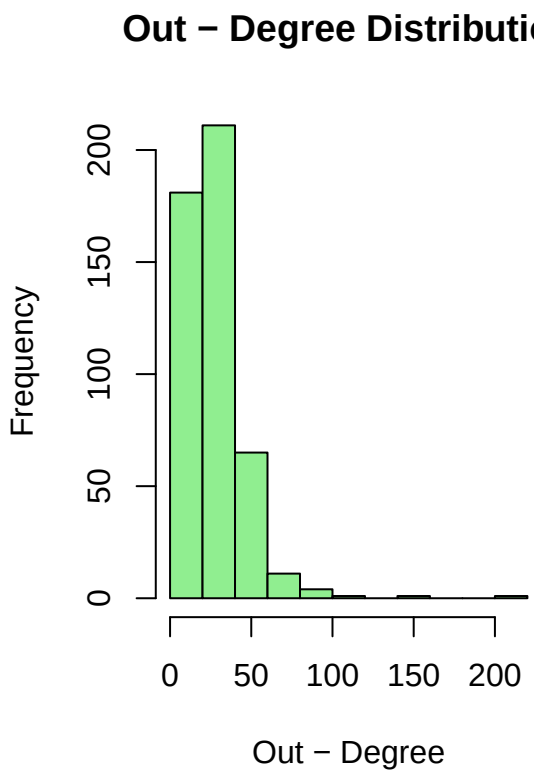
```
## Size of largest strongly connected component: 469
```

Degree Distribution

```r
# Calculate different degree measures
in_deg <- degree(g, mode="in")
out_deg <- degree(g, mode="out")
total_deg <- degree(g, mode="total")

par ( mfrow = c (1 ,2))
hist ( in_deg , main = " In - Degree Distribution " ,
xlab = " In - Degree " , ylab = " Frequency " , col = " lightblue " )
hist ( out_deg , main = " Out - Degree Distribution " ,
xlab = " Out - Degree " , ylab = " Frequency " , col = " lightgreen " )
```





PageRank

```
page_rank <- page_rank (g , weights = E ( g ) $weight , directed = TRUE ) $vector
```

Hub and Authority Scores

Note: The depreciation warning about 'hub_score' and 'authority_score' appears to not actually be true.

```
hub_scores <- hub_score(g , scale = TRUE ) $vector # Hub scores
```

```
## Warning: `hub_score()` was deprecated in igraph 2.0.3.
## i Please use `hits_scores()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
auth_scores <- authority_score(g , scale = TRUE ) $vector # Authority scores
```

```
## Warning: `authority_score()` was deprecated in igraph 2.1.0.
## i Please use `hits_scores()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

Closeness Centrality

```
harmonic_close <- harmonic_centrality (g , weights = E ( g ) $weight , mode = "out" )
```

Betweenness Centrality

```
betweenness <- betweenness (g , weights = E ( g ) $weight , directed = TRUE , normalized = TRUE )
```

Nodes

```
get_top_10 <- function (metric, metric_name) {
  top_indices <- order ( metric , decreasing = TRUE )[1:10]
  data.frame (
  Metric = rep(metric_name, 10) ,
  Node = top_indices ,
  Value = round ( metric [ top_indices ] , 4)
  )
}

top_nodes <- rbind (
  get_top_10 ( in_deg , " In - Degree " ) ,
  get_top_10 ( out_deg , " Out - Degree " ) ,
  get_top_10 ( page_rank , " PageRank " ) ,
  get_top_10 ( harmonic_close , " Harmonic Closeness " ) ,
  get_top_10 ( betweenness , " Betweenness " ) ,
  get_top_10 ( hub_scores, " Hub Scores ") ,
  get_top_10 ( auth_scores, " Authority Scores ")
)
# Print results
print ( " Top 10 nodes by different centrality measures : " )
```

```
## [1] " Top 10 nodes by different centrality measures : "
```

```
print ( top_nodes )
```

```
##                    Metric Node       Value
## 322          In - Degree   212    127.0000
## 208          In - Degree   397    121.0000
## 190          In - Degree    36    120.0000
## 111          In - Degree    71    109.0000
## 385          In - Degree    82    108.0000
## 254          In - Degree   172    108.0000
## 269          In - Degree   236    106.0000
## 192          In - Degree    37    105.0000
## 303          In - Degree   151     97.0000
## 147          In - Degree   205     97.0000
## 367         Out - Degree   163    210.0000
## 3221        Out - Degree   212    157.0000
## 393         Out - Degree   121    111.0000
## 71          Out - Degree   157     97.0000
## 399         Out - Degree   383     89.0000
## 436         Out - Degree   355     85.0000
## 179         Out - Degree    94     84.0000
## 2541        Out - Degree   172     79.0000
## 105         Out - Degree   135     75.0000
## 87          Out - Degree    16     71.0000
## 3222            PageRank   212      0.0167
## 1471            PageRank   205      0.0128
```

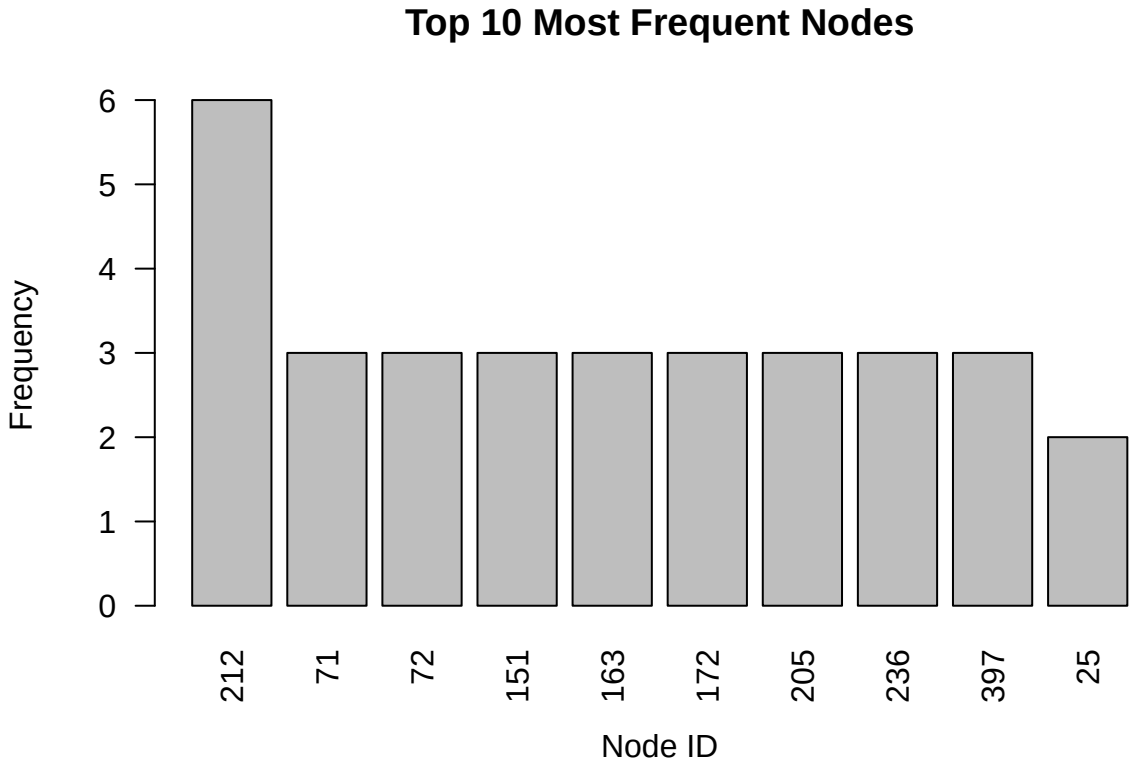
```
## 389             PageRank   140      0.0111
## 2691            PageRank   236      0.0107
## 215             PageRank   148      0.0106
## 2081            PageRank   397      0.0089
## 92              PageRank    70      0.0082
## 246             PageRank    78      0.0082
## 113             PageRank    72      0.0080
## 3031            PageRank   151      0.0080
## 17    Harmonic Closeness    25 258396.3888
## 149   Harmonic Closeness    93 197943.6471
## 1111  Harmonic Closeness    71 189102.1252
## 3     Harmonic Closeness    60 188666.1494
## 88    Harmonic Closeness    92 185805.9653
## 428   Harmonic Closeness    39 184838.3983
## 32    Harmonic Closeness    63 180308.9408
## 263   Harmonic Closeness   160 176995.4495
## 3671  Harmonic Closeness   163 171500.8813
## 22    Harmonic Closeness    62 171473.9764
## 1112         Betweenness    71      0.0845
## 3223         Betweenness   212      0.0707
## 171          Betweenness    25      0.0677
## 4281         Betweenness    39      0.0619
## 1472         Betweenness   205      0.0500
## 3672         Betweenness   163      0.0450
## 3032         Betweenness   151      0.0438
## 2542         Betweenness   172      0.0412
## 1131         Betweenness    72      0.0399
## 2151         Betweenness   148      0.0376
## 226           Hub Scores    76      1.0000
## 159           Hub Scores   232      0.8593
## 3991          Hub Scores   383      0.8423
## 164           Hub Scores   442      0.6993
## 129           Hub Scores   412      0.6380
## 220           Hub Scores   233      0.6114
## 118           Hub Scores   294      0.5920
## 440           Hub Scores   433      0.5685
## 354           Hub Scores   239      0.5546
## 3224          Hub Scores   212      0.5460
## 3225    Authority Scores   212      1.0000
## 3891    Authority Scores   140      0.8251
## 2692    Authority Scores   236      0.6272
## 1132    Authority Scores    72      0.5033
## 2461    Authority Scores    78      0.4468
## 2082    Authority Scores   397      0.4260
## 188     Authority Scores   286      0.4085
## 318     Authority Scores   138      0.3433
## 1921    Authority Scores    37      0.3400
## 335     Authority Scores    81      0.3315
```

```r
node_freq <- table(top_nodes$Node)

# Convert to dataframe and sort
node_freq_df <- data.frame(
  Node = names(node_freq),
```

```
  Frequency = as.numeric(node_freq)
)
node_freq_df <- node_freq_df[order(-node_freq_df$Frequency), ]

# Create the plot using base R
barplot(node_freq_df$Frequency[1:10],
        names.arg = node_freq_df$Node[1:10],
        main = "Top 10 Most Frequent Nodes",
        xlab = "Node ID",
        ylab = "Frequency",
        las = 2)  # Rotate x-axis labels
```



Justification