PPL4246 Final Assignment

2024-10-29

```
# Packages needed
library(bootnet)
```

```
## Loading required package: ggplot2
```

```
## This is bootnet 1.6
```

```
## For questions and issues, please see github.com/SachaEpskamp/bootnet.
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
##
## The following objects are masked from 'package:lubridate':
##
##     %--%, union
##
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
##
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
##
## The following object is masked from 'package:tidyr':
##
```

```
##     crossing
##
## The following object is masked from 'package:tibble':
##
##     as_data_frame
##
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
##
## The following object is masked from 'package:base':
##
##     union
```

```
library(qgraph)
library(ggplot2)
```

Participant Data

```
# Read csv file
stars.data <- read.csv('stars-data_99.csv', header = TRUE)

# Relevant columns (items 1 to 51)
stars.data1 <- stars.data |> select(item_1:item_51)

# Calculate the sum of STARS scores for each student (row)
stars.data1$total_score <- rowSums(stars.data1)

# Calculate the average STARS score across all students
average_score <- mean(stars.data1$total_score, na.rm = TRUE)
median_score <- median(stars.data1$total_score, na.rm = TRUE)
# Calculate standard deviation of the total STARS scores
sd_total_score <- sd(stars.data1$total_score, na.rm = TRUE)

average_score
```

```
## [1] 129.8776
```

```
median_score
```

```
## [1] 128
```

```
sd_total_score
```

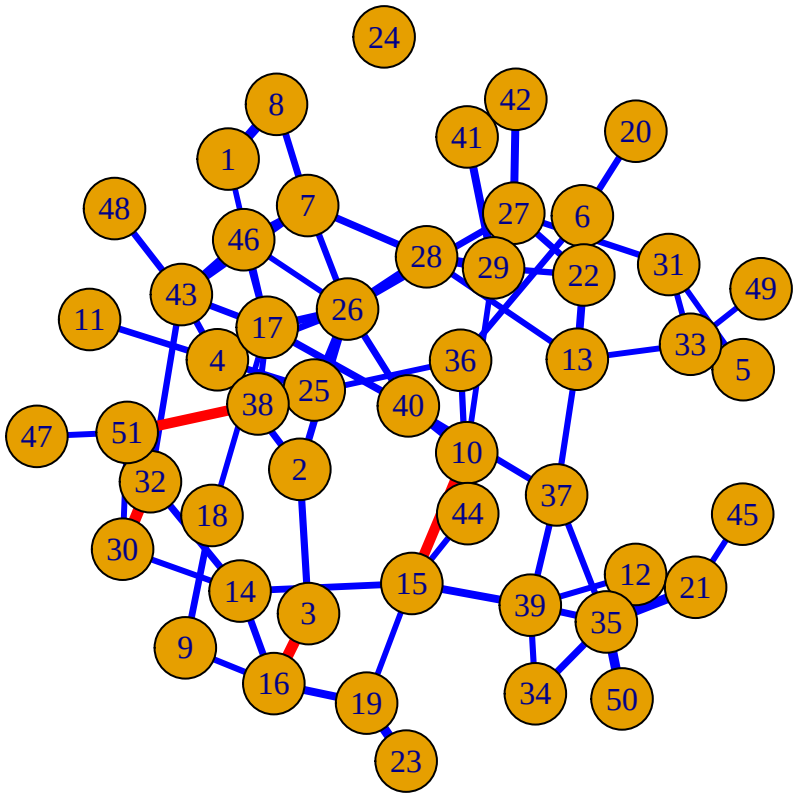
```
## [1] 30.02517
```

Create network

```r
# Relevant columns (items 1 to 51)
stars.data2 <- stars.data |> select(item_1:item_51)

# Estimate network using partial correlations and apply significance threshold
stars.matrix <- estimateNetwork(stars.data2, default = "pcor", threshold = 'sig')
```

```
## Estimating Network. Using package::function:
##   - qgraph::qgraph(..., graph = 'pcor') for network computation
##   - psych::corr.p for significance thresholding
```

```r
# Extract adjacency matrix
stars_adj_mat <- stars.matrix$graph

# Create igraph object from adjacency matrix
stars_adj_mat <- abs(stars.matrix$graph)
stars_network <- graph_from_adjacency_matrix(stars_adj_mat, mode = "undirected", weighted = TRUE, diag =

# Rename nodes to numbers
V(stars_network)$name <- as.character(seq_along(V(stars_network)))

# Adjust edge width by correlation coefficient, scaled for better visibility
E(stars_network)$width <- E(stars_network)$weight * 10

# Color edges based on correlation threshold (red for strong, blue for weaker correlations)
E(stars_network)$color <- ifelse(E(stars_network)$weight > 0.5, "red", "blue")

# Plot network
layout <- layout_with_fr(stars_network)
par(mar = c(0.2, 0.2, 0.2, 0.2)) #set margins
plot(stars_network, edge.width = E(stars_network)$width, edge.color = E(stars_network)$color, vertex.si
```




```
summary(stars_network)
```

```
## IGRAPH 2c992c7 UNW- 51 83 --
## + attr: name (v/c), weight (e/n), width (e/n), color (e/c)
```

Basic Info

```
# Number of nodes
num_nodes <- vcount(stars_network)

# Number of edges
num_edges <- ecount(stars_network)

# Network density
network_density <- edge_density(stars_network)

# Network diameter
network_diameter <- diameter(stars_network)

# Network degree distribution
degree_distribution <- degree(stars_network)

# Print basic metrics
cat("Number of nodes:", num_nodes, "\n")
```

```
## Number of nodes: 51
```

```
cat("Number of edges:", num_edges, "\n")
```

```
## Number of edges: 83
```

```
cat("Network density:", network_density, "\n")
```

```
## Network density: 0.06509804
```

```
cat("Network diameter:", network_diameter, "\n")
```

```
## Network diameter: 2.661297
```

Centrality Measures

```
# Degree Centrality
degree_centrality <- degree(stars_network)

# Sort nodes by degree
node_degree_sorted <- sort(degree_centrality, decreasing = TRUE)
cat("Nodes with highest degree:\n")
```

```
## Nodes with highest degree:
```

```
head(node_degree_sorted)
```

```
## 26 17  4 35 43  7
##  9  8  7  6  6  5
```

```
# Betweenness Centrality
betweenness_centrality <- igraph::betweenness(stars_network, normalized = TRUE)

# Sort nodes by betweenness centrality
betweenness_sorted <- sort(betweenness_centrality, decreasing = TRUE)
cat("Nodes with highest betweenness centrality:\n")
```

```
## Nodes with highest betweenness centrality:
```

```
head(betweenness_sorted)
```

```
##        26        37        15        17        40        39
## 0.2000000 0.1975510 0.1746939 0.1469388 0.1469388 0.1371429
```

```
# Closeness Centrality
closeness_centrality <- closeness(stars_network, normalized = TRUE)

# Sort nodes by closeness centrality
closeness_sorted <- sort(closeness_centrality, decreasing = TRUE)
cat("Nodes with highest closeness centrality:\n")
```

```
## Nodes with highest closeness centrality:
```

```
head(closeness_sorted)
```

```
##        26        17        40        37         4        38
## 1.0838069 1.0719898 1.0614647 1.0055608 0.9928715 0.9665401
```

```
# Strength
strength_centrality <- strength(stars_network)

# Sort nodes by strength
strength_sorted <- sort(strength_centrality, decreasing = TRUE)
cat("Nodes with highest strength:\n")
```

```
## Nodes with highest strength:
```

```
head(strength_sorted)
```

```
##       26       17       35        4       43       29
## 3.163503 2.729945 2.423849 2.383934 2.027628 1.999646
```

Community Detection (Louvain Method)

```
# Louvain Method for Community Detection
louvain_communities <- cluster_louvain(stars_network)
cat("Louvain Community Detection:\n")
```

```
## Louvain Community Detection:
```

```
data.frame(node = V(stars_network)$name, community = louvain_communities$membership) |> head(10)
```

```
##    node community
## 1     1         1
## 2     2         1
## 3     3         2
## 4     4         1
## 5     5         3
## 6     6         4
## 7     7         1
## 8     8         1
## 9     9         2
## 10   10         4
```

```
table(louvain_communities$membership)
```

```
##
##  1  2  3  4  5  6  7
## 14  6 11  6  8  5  1
```

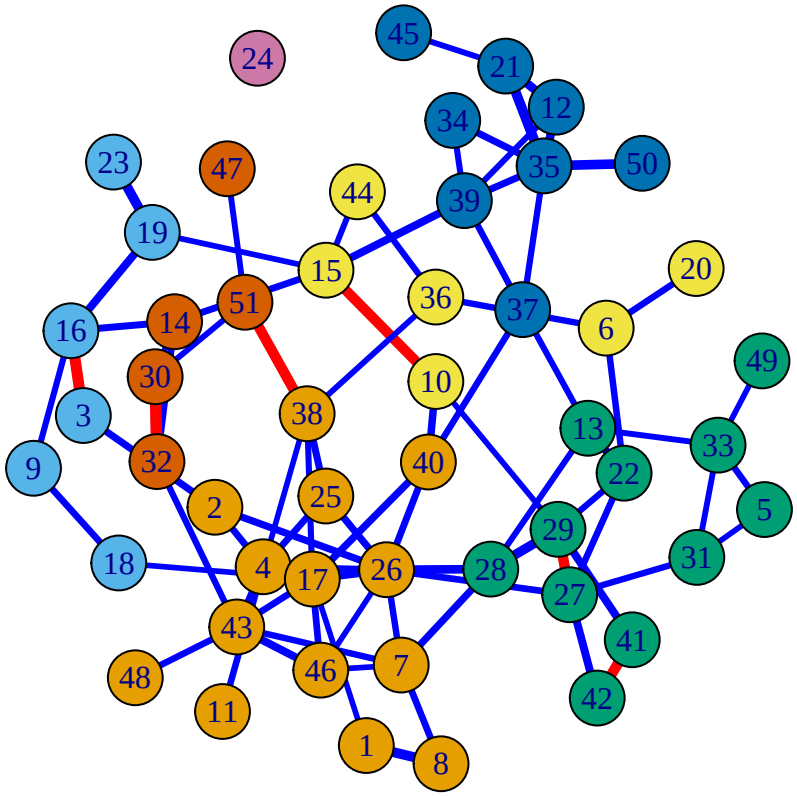
```
membership(louvain_communities)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
##  1  1  2  1  3  4  1  1  2  4  1  5  3  6  4  2  1  2  2  4  5  3  2  7  1  1
## 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
##  3  3  3  6  3  6  3  5  5  4  5  1  5  1  3  3  1  4  5  1  6  1  3  5  6
```

```
modularity(louvain_communities)
```

```
## [1] 0.5822646
```

```
# Plot with community colors
par(mar = c(0.2, 0.2, 0.2, 0.2)) #set margins
plot(stars_network, vertex.color = membership(louvain_communities))
```



Key Player Problem

```
library(keyplayer)
```

```
##
## Attaching package: 'keyplayer'
```

```
## The following object is masked from 'package:igraph':
##
##     contract
```

```
# KPP-Pos
set.seed(1)

kpp_pos <- kpset(stars_adj_mat,
                 size = 3, # Number of key players
                 type = "diffusion",
                 method = "union",
                 T = 1, # Diffusion steps
                 binary = TRUE) # Treat edges as unweighted
# Cohesion score
kpp_pos$centrality / gorder(stars_network) # Normalized
```

```
## [1] 0.1427553
```

```
# Positive Key Player Set
V(stars_network)$name[kpp_pos$keyplayers]
```

```
## [1] "15" "26" "35"
```

```
# KPP-Neg
set.seed(1)

kpp_neg <- kpset(stars_adj_mat,
                 size = 3, # Number of key players
                 type = "fragment",
                 method = "min",
                 binary = F)

# Fragmentation score
kpp_neg$centrality
```

```
## [1] 0.7812052
```

```
# Negative Key Player Set
V(stars_network)$name[kpp_neg$keyplayers]
```

```
## [1] "7"  "17" "26"
```