Deeper analysis and Network statistics

Jinxi_Hu-48528608, Samarth_Grover-38220463

2025-11-09

set up

```
library(readr)
library(igraph)
library(RColorBrewer)
library(ggplot2)
library(reshape2)
library(scales)
library(gridExtra)
set.seed(48528608)
```

Load and prepare connected nodes data (Jinxi Hu)

```
# Load the connected nodes and edges data
nodes_connected <- read.csv("data/nodes_connected.csv")
edges_connected <- read.csv("data/edges_connected.csv")

cat("=== DATA LOADING ===\n")
```

```
## === DATA LOADING ===
```

```
cat("Connected nodes loaded:", nrow(nodes_connected), "\n")
```

```
## Connected nodes loaded: 1582
```

```
cat("Connected edges loaded:", nrow(edges_connected), "\n\n")
```

```
## Connected edges loaded: 3757
```

```
# Create the graph from connected nodes only
graph_connected <- graph_from_data_frame(edges_connected,
                                         vertices = nodes_connected,
                                         directed = TRUE)
# Remove multiple edges and self-loops for cleaner analysis
graph_connected <- simplify(graph_connected,
                            remove.multiple = TRUE,
```

```r
                              remove.loops = TRUE)

cat("Graph created successfully\n")
```

```
## Graph created successfully
```

```r
cat("Final nodes in graph:", vcount(graph_connected), "\n")
```

```
## Final nodes in graph: 1582
```

```r
cat("Final edges in graph:", ecount(graph_connected), "\n\n")
```

```
## Final edges in graph: 3730
```

Basic Graph Analysis (Jinxi Hu)

```r
cat("=== BASIC GRAPH STATISTICS ===\n")
```

```
## === BASIC GRAPH STATISTICS ===
```

```r
cat("Total nodes:", vcount(graph_connected), "\n")
```

```
## Total nodes: 1582
```

```r
cat("Total edges (citations):", ecount(graph_connected), "\n")
```

```
## Total edges (citations): 3730
```

```r
cat("Network density:", round(edge_density(graph_connected), 6), "\n")
```

```
## Network density: 0.001491
```

```r
cat("Is directed:", is_directed(graph_connected), "\n")
```

```
## Is directed: TRUE
```

```r
cat("Is weighted:", is_weighted(graph_connected), "\n\n")
```

```
## Is weighted: FALSE
```

```r
# Calculate degree statistics
all_degrees <- degree(graph_connected, mode = "all")
in_degrees <- degree(graph_connected, mode = "in")
out_degrees <- degree(graph_connected, mode = "out")

cat("=== DEGREE STATISTICS ===\n")
```

```
## === DEGREE STATISTICS ===
```

```r
cat("Average total degree:", round(mean(all_degrees), 2), "\n")
```

```
## Average total degree: 4.72
```

```r
cat("Average in-degree (citations received):", round(mean(in_degrees), 2), "\n")
```

```
## Average in-degree (citations received): 2.36
```

```r
cat("Average out-degree (citations made):", round(mean(out_degrees), 2), "\n\n")
```

```
## Average out-degree (citations made): 2.36
```

```r
cat("Degree range (total):", min(all_degrees), "-", max(all_degrees), "\n")
```

```
## Degree range (total): 0 - 110
```

```r
cat("Degree range (in):", min(in_degrees), "-", max(in_degrees), "\n")
```

```
## Degree range (in): 0 - 104
```

```r
cat("Degree range (out):", min(out_degrees), "-", max(out_degrees), "\n\n")
```

```
## Degree range (out): 0 - 29
```

```r
cat("Standard deviation (total):", round(sd(all_degrees), 2), "\n")
```

```
## Standard deviation (total): 6.74
```

```r
cat("Standard deviation (in):", round(sd(in_degrees), 2), "\n")
```

```
## Standard deviation (in): 6.2
```

```r
cat("Standard deviation (out):", round(sd(out_degrees), 2), "\n\n")
```

```
## Standard deviation (out): 2.72
```

```r
cat("=== COMPONENT ANALYSIS ===\n")
```

```
## === COMPONENT ANALYSIS ===
```

```r
# Analyze weakly connected components
weak_components <- components(graph_connected, mode = "weak")
cat("Number of weakly connected components:", weak_components$no, "\n")
```

```
## Number of weakly connected components: 58
```

```r
cat("Size of largest weak component:", max(weak_components$csize), "\n")
```

```
## Size of largest weak component: 1433
```

```r
cat("Proportion of nodes in largest weak component:",
    round(max(weak_components$csize) / vcount(graph_connected) * 100, 2), "%\n\n")
```

```
## Proportion of nodes in largest weak component: 90.58 %
```

```r
# Analyze strongly connected components
strong_components <- components(graph_connected, mode = "strong")
cat("Number of strongly connected components:", strong_components$no, "\n")
```

```
## Number of strongly connected components: 1569
```

```r
cat("Size of largest strong component:", max(strong_components$csize), "\n")
```

```
## Size of largest strong component: 6
```

```r
cat("Proportion of nodes in largest strong component:",
    round(max(strong_components$csize) / vcount(graph_connected) * 100, 2), "%\n\n")
```

```
## Proportion of nodes in largest strong component: 0.38 %
```

```r
# Component size distribution
cat("Weak component sizes (top 10):\n")
```

```
## Weak component sizes (top 10):
```

```r
weak_sizes <- sort(weak_components$csize, decreasing = TRUE)
print(head(weak_sizes, 10))
```

```
##  [1] 1433   27    9    5    5    4    3    3    3    3
```

```r
cat("\nStrong component sizes (top 10):\n")
```

```
##
## Strong component sizes (top 10):
```

```r
strong_sizes <- sort(strong_components$csize, decreasing = TRUE)
print(head(strong_sizes, 10))
```

```
##  [1] 6 2 2 2 2 2 2 2 2 1
```

```r
# Create comprehensive degree distribution analysis

# Prepare data for plotting
degree_data <- data.frame(
  node_id = V(graph_connected)$name,
  total_degree = all_degrees,
  in_degree = in_degrees,
  out_degree = out_degrees
)

# Add node attributes for additional analysis
degree_data$institution <- V(graph_connected)$institution
degree_data$subtopic <- V(graph_connected)$subtopic
degree_data$year <- V(graph_connected)$year
degree_data$citations <- V(graph_connected)$citations

cat("=== DEGREE DISTRIBUTION SUMMARY ===\n")
```

```
## === DEGREE DISTRIBUTION SUMMARY ===
```

```r
cat("Total degree quartiles:\n")
```

```
## Total degree quartiles:
```

```r
print(quantile(all_degrees))
```

```
##   0%  25%  50%  75% 100%
##    0    1    3    5  110
```

```r
cat("\nIn-degree quartiles:\n")
```

```
##
## In-degree quartiles:
```

```r
print(quantile(in_degrees))
```

```
##   0%  25%  50%  75% 100%
##    0    0    1    2  104
```

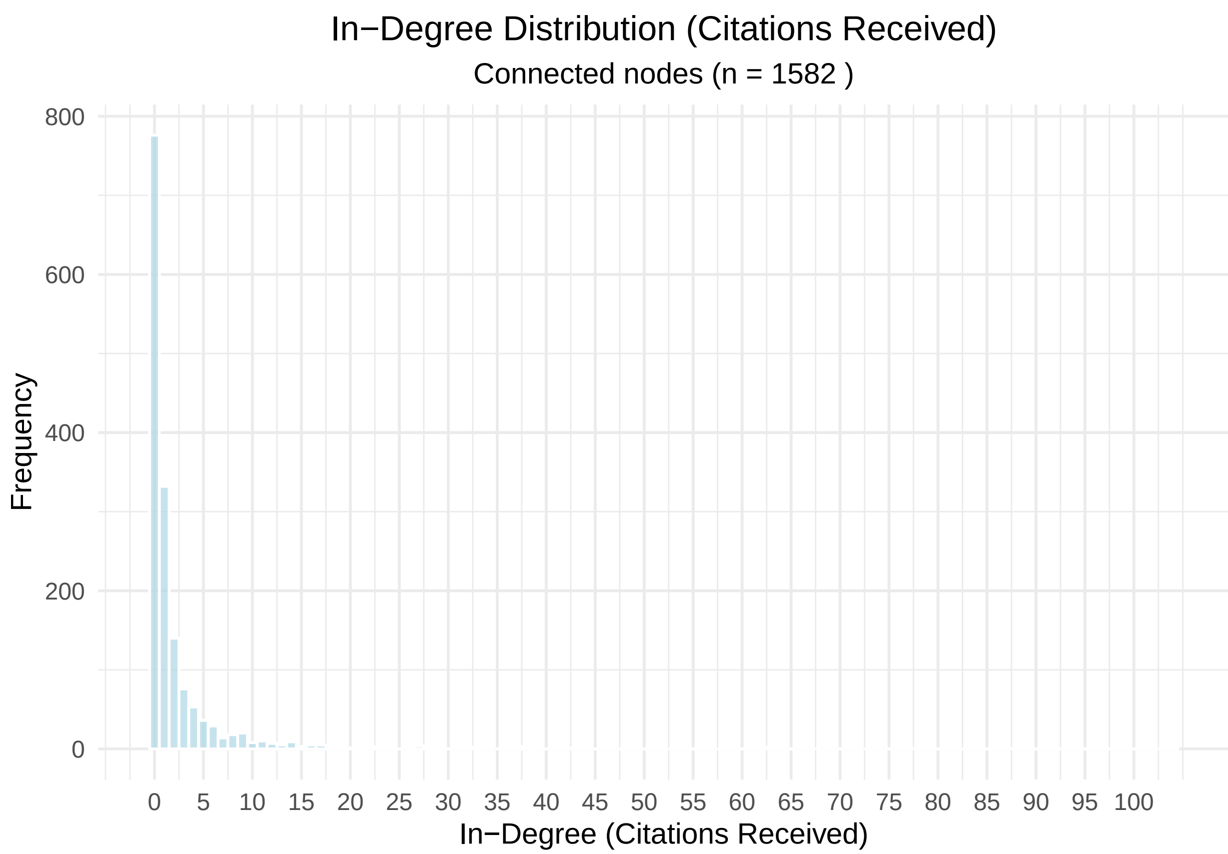
```r
cat("\nOut-degree quartiles:\n")
```

```
##
## Out-degree quartiles:
```

```r
print(quantile(out_degrees))
```

```
##   0%  25%  50%  75% 100%
##    0    1    2    3   29
```

```r
# In-degree distribution histogram
ggplot(degree_data, aes(x = in_degree)) +
  geom_histogram(binwidth = 1, fill = "lightblue", alpha = 0.7, color = "white") +
  labs(title = "In-Degree Distribution (Citations Received)",
       subtitle = paste("Connected nodes (n =", vcount(graph_connected), ")"),
       x = "In-Degree (Citations Received)",
       y = "Frequency") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5)) +
  scale_x_continuous(breaks = seq(0, max(in_degrees), by = 5))
```




```r
# Print summary statistics
cat("\nIN-DEGREE DISTRIBUTION STATISTICS:\n")
```

```
##
## IN-DEGREE DISTRIBUTION STATISTICS:
```

```r
cat("Mean:", round(mean(in_degrees), 2), "\n")
```

```
## Mean: 2.36
```

```r
cat("Median:", median(in_degrees), "\n")
```

```
## Median: 1
```

```r
cat("Mode:", names(sort(table(in_degrees), decreasing = TRUE))[1], "\n")
```

```
## Mode: 0
```

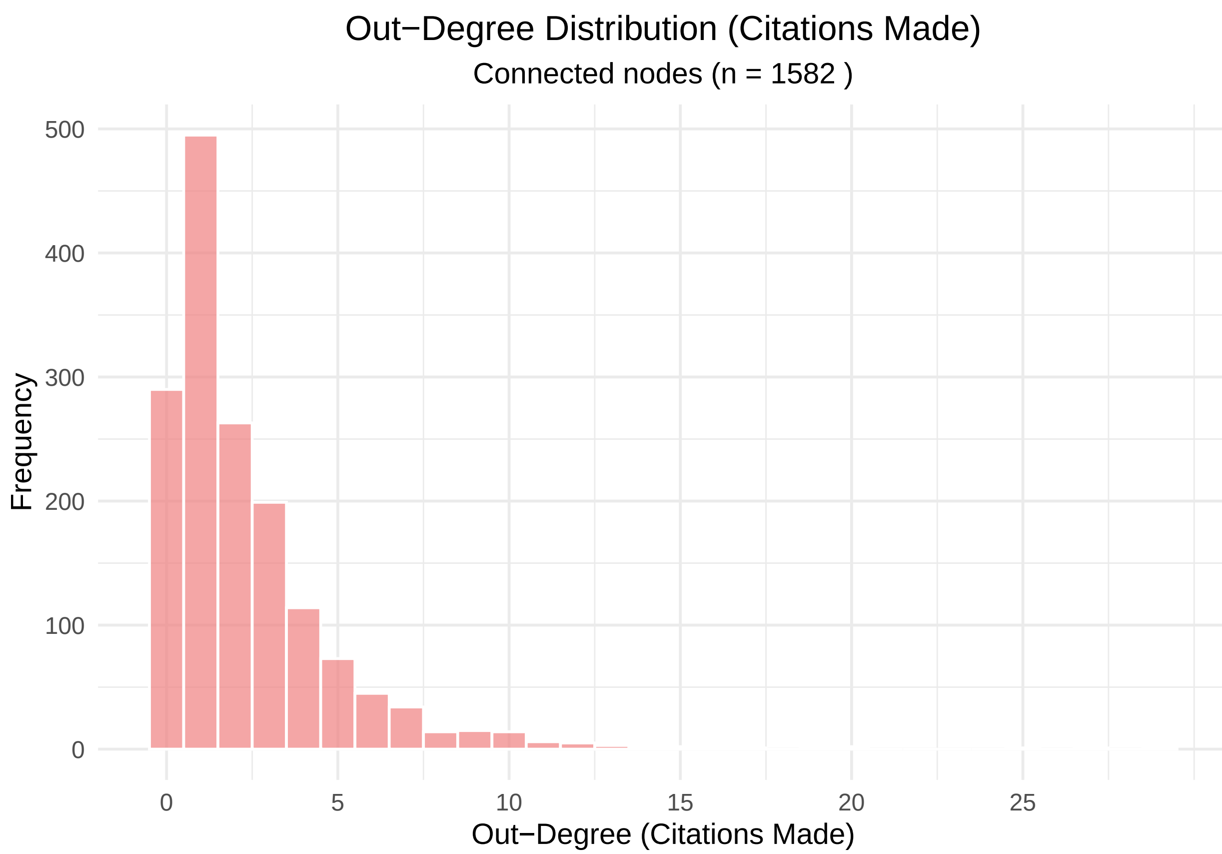
```r
cat("Nodes with 0 in-degree:", sum(in_degrees == 0), "\n")
```

```
## Nodes with 0 in-degree: 776
```

```r
cat("Nodes with >10 in-degree:", sum(in_degrees > 10), "\n")
```

```
## Nodes with >10 in-degree: 80
```

```r
# Out-degree distribution histogram
ggplot(degree_data, aes(x = out_degree)) +
  geom_histogram(binwidth = 1, fill = "lightcoral", alpha = 0.7, color = "white") +
  labs(title = "Out-Degree Distribution (Citations Made)",
       subtitle = paste("Connected nodes (n =", vcount(graph_connected), ")"),
       x = "Out-Degree (Citations Made)",
       y = "Frequency") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5)) +
  scale_x_continuous(breaks = seq(0, max(out_degrees), by = 5))
```




```r
# Print summary statistics
cat("\nOUT-DEGREE DISTRIBUTION STATISTICS:\n")
```

```
##
## OUT-DEGREE DISTRIBUTION STATISTICS:
```

```r
cat("Mean:", round(mean(out_degrees), 2), "\n")
```

```
## Mean: 2.36
```

```r
cat("Median:", median(out_degrees), "\n")
```

```
## Median: 2
```

```r
cat("Mode:", names(sort(table(out_degrees), decreasing = TRUE))[1], "\n")
```

```
## Mode: 1
```

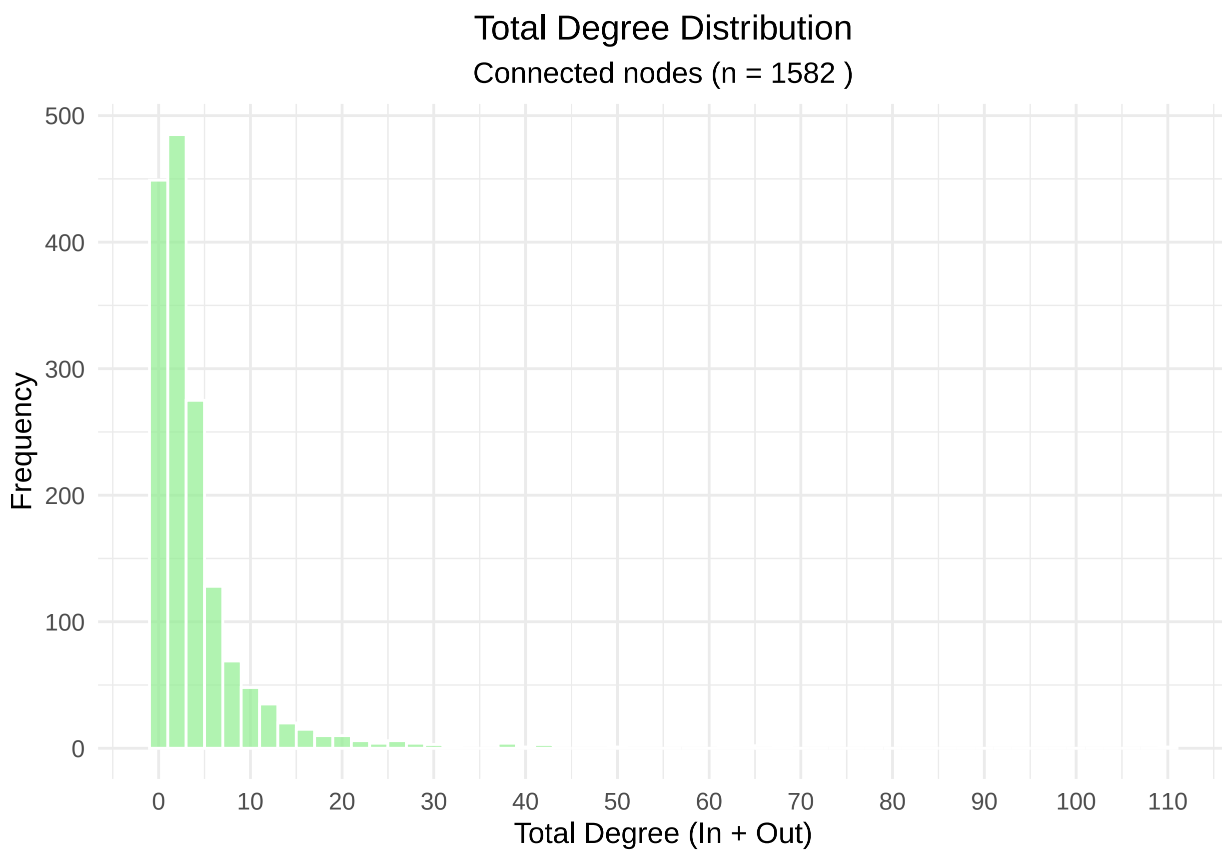
```r
cat("Nodes with 0 out-degree:", sum(out_degrees == 0), "\n")
```

```
## Nodes with 0 out-degree: 290
```

```r
cat("Nodes with >10 out-degree:", sum(out_degrees > 10), "\n")
```

```
## Nodes with >10 out-degree: 26
```

```r
# Total degree distribution histogram
ggplot(degree_data, aes(x = total_degree)) +
  geom_histogram(binwidth = 2, fill = "lightgreen", alpha = 0.7, color = "white") +
  labs(title = "Total Degree Distribution",
       subtitle = paste("Connected nodes (n =", vcount(graph_connected), ")"),
       x = "Total Degree (In + Out)",
       y = "Frequency") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5)) +
  scale_x_continuous(breaks = seq(0, max(all_degrees), by = 10))
```




```r
# Print summary statistics
cat("\nTOTAL DEGREE DISTRIBUTION STATISTICS:\n")
```

```
##
## TOTAL DEGREE DISTRIBUTION STATISTICS:
```

```r
cat("Mean:", round(mean(all_degrees), 2), "\n")
```

```
## Mean: 4.72
```

```r
cat("Median:", median(all_degrees), "\n")
```

```
## Median: 3
```

```r
cat("Mode:", names(sort(table(all_degrees), decreasing = TRUE))[1], "\n")
```

```
## Mode: 1
```

```r
cat("Nodes with degree 1:", sum(all_degrees == 1), "\n")
```

```
## Nodes with degree 1: 434
```

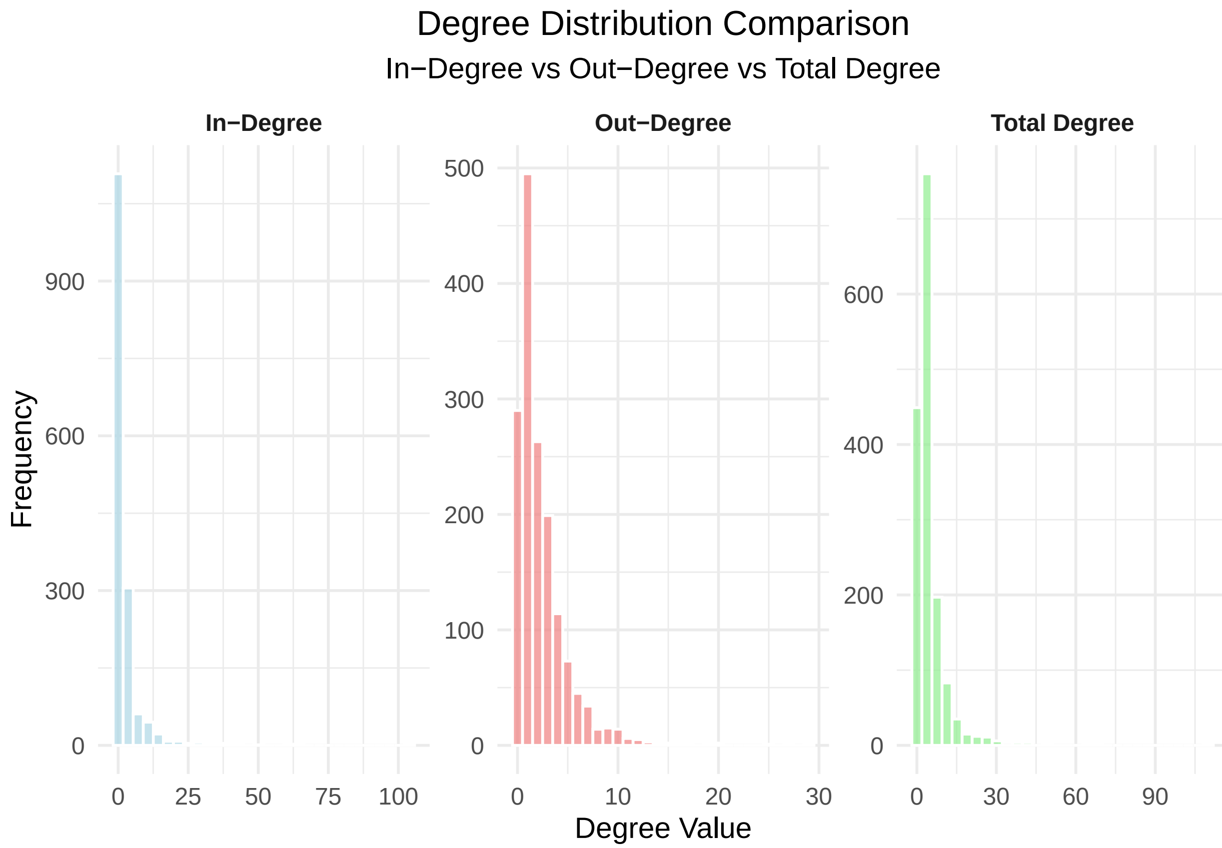
```r
cat("Nodes with degree >20:", sum(all_degrees > 20), "\n")
```

```
## Nodes with degree >20: 43
```

```r
# Create combined degree distribution plot
degree_long <- reshape2::melt(degree_data[, c("in_degree", "out_degree", "total_degree")],
                              variable.name = "degree_type",
                              value.name = "degree_value")
```

```
## No id variables; using all as measure variables
```

```r
# Rename for better labels
degree_long$degree_type <- factor(degree_long$degree_type,
                                  levels = c("in_degree", "out_degree", "total_degree"),
                                  labels = c("In-Degree", "Out-Degree", "Total Degree"))

# Create faceted histogram
ggplot(degree_long, aes(x = degree_value, fill = degree_type)) +
  geom_histogram(alpha = 0.7, color = "white", bins = 30) +
  facet_wrap(~degree_type, scales = "free") +
  scale_fill_manual(values = c("In-Degree" = "lightblue",
                               "Out-Degree" = "lightcoral",
                               "Total Degree" = "lightgreen")) +
  labs(title = "Degree Distribution Comparison",
       subtitle = "In-Degree vs Out-Degree vs Total Degree",
       x = "Degree Value",
       y = "Frequency") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        legend.position = "none",
        strip.text = element_text(face = "bold"))
```




```r
# Analyze the largest component in detail
cat("=== LARGEST COMPONENT DETAILED ANALYSIS ===\n")
```

```
## === LARGEST COMPONENT DETAILED ANALYSIS ===
```

```r
# Extract the largest weakly connected component
largest_comp_nodes <- which(weak_components$membership == which.max(weak_components$csize))
largest_component <- induced_subgraph(graph_connected, largest_comp_nodes)

cat("Largest component statistics:\n")
```

```
## Largest component statistics:
```

```r
cat("Nodes:", vcount(largest_component), "\n")
```

```
## Nodes: 1433
```

```r
cat("Edges:", ecount(largest_component), "\n")
```

```
## Edges: 3634
```

```r
cat("Density:", round(edge_density(largest_component), 6), "\n")
```

```
## Density: 0.001771
```

```r
cat("Average in degree:", round(mean(degree(largest_component, mode = "in")), 2), "\n")
```

```
## Average in degree: 2.54
```

```r
cat("Average out degree:", round(mean(degree(largest_component, mode = "out")), 2), "\n")
```

```
## Average out degree: 2.54
```

```r
cat("Average degree:", round(mean(degree(largest_component, mode = "all")), 2), "\n")
```

```
## Average degree: 5.07
```

```r
cat("Diameter:", diameter(largest_component, directed = FALSE), "\n")
```

```
## Diameter: 13
```

```r
cat("Average path length:", round(mean_distance(largest_component, directed = FALSE), 2), "\n\n")
```

```
## Average path length: 4.5
```

```r
# Check if there are smaller components worth analyzing
if(length(unique(weak_components$csize)) > 1) {
  second_largest_size <- sort(weak_components$csize, decreasing = TRUE)[2]
  cat("Second largest component size:", second_largest_size, "\n")
  cat("Ratio (largest/second largest):", round(max(weak_components$csize) /
  second_largest_size, 2), "\n")
}
```

```
## Second largest component size: 27
## Ratio (largest/second largest): 53.07
```

Advanced Analysis of Largest Component (Jinxi Hu)

```r
cat("=== CENTRALITY ANALYSIS OF LARGEST COMPONENT ===\n")
```

```
## === CENTRALITY ANALYSIS OF LARGEST COMPONENT ===
```

```r
# Calculate various centrality measures
cat("Calculating centrality measures...\n")
```

```
## Calculating centrality measures...
```

```r
# Degree centrality (already calculated above)
degree_cent_in <- degree(largest_component, mode = "in", normalized = TRUE)
degree_cent_out <- degree(largest_component, mode = "out", normalized = TRUE)
degree_cent_all <- degree(largest_component, mode = "all", normalized = TRUE)

# Betweenness centrality
betweenness_cent <- betweenness(largest_component, directed = TRUE, normalized = TRUE)

# Closeness centrality
closeness_cent_in <- closeness(largest_component, mode = "in", normalized = TRUE)
closeness_cent_out <- closeness(largest_component, mode = "out", normalized = TRUE)

# Eigenvector centrality
eigenvector_cent <- eigen_centrality(largest_component, directed = TRUE, scale = TRUE)$vector
```

```
## Warning: The `scale` argument of `eigen_centrality()` is deprecated as of igraph 2.1.1.
## i eigen_centrality() will always behave as if scale=TRUE were used.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
# PageRank centrality
pagerank_cent <- page_rank(largest_component, directed = TRUE)$vector

# Authority and Hub scores (HITS algorithm)
hits_scores <- hub_score(largest_component)
```

```
## 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.
```

```r
hub_cent <- hits_scores$vector
authority_cent <- authority_score(largest_component)$vector
```

```
## 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.
```

```r
cat("All centrality measures calculated successfully!\n\n")
```

```
## All centrality measures calculated successfully!
```

```r
# Create centrality summary
centrality_summary <- data.frame(
  node_id = V(largest_component)$name,
  degree_in = degree_cent_in,
```

```r
  degree_out = degree_cent_out,
  degree_all = degree_cent_all,
  betweenness = betweenness_cent,
  closeness_in = closeness_cent_in,
  closeness_out = closeness_cent_out,
  eigenvector = eigenvector_cent,
  pagerank = pagerank_cent,
  hub = hub_cent,
  authority = authority_cent
)
# Print top nodes by different centrality measures
cat("TOP 5 NODES BY DIFFERENT CENTRALITY MEASURES:\n\n")
```

```
## TOP 5 NODES BY DIFFERENT CENTRALITY MEASURES:
```

```r
cat("Highest In-Degree Centrality (most cited):\n")
```

```
## Highest In-Degree Centrality (most cited):
```

```r
top_in_degree <- centrality_summary[order(centrality_summary$degree_in, decreasing = TRUE)[1:5],
                                   c("node_id", "degree_in")]
print(top_in_degree)
```

```
##       node_id  degree_in
## P0060   P0060 0.07262570
## P0001   P0001 0.04399441
## P0009   P0009 0.04399441
## P0002   P0002 0.04259777
## P0199   P0199 0.04259777
```

```r
cat("\nHighest PageRank (most influential):\n")
```

```
##
## Highest PageRank (most influential):
```

```r
top_pagerank <- centrality_summary[order(centrality_summary$pagerank, decreasing = TRUE)[1:5],
                                  c("node_id", "pagerank")]
print(top_pagerank)
```

```
##       node_id   pagerank
## P0199   P0199 0.04046559
## P0031   P0031 0.03232105
## P0741   P0741 0.01974340
## P0116   P0116 0.01746391
## P0535   P0535 0.01572970
```

```r
cat("\nHighest Betweenness Centrality (most important bridges):\n")
```

```
##
## Highest Betweenness Centrality (most important bridges):
```

```r
top_betweenness <- centrality_summary[order(centrality_summary$betweenness, decreasing = TRUE)[1:5],
                                     c("node_id", "betweenness")]
print(top_betweenness)
```

```
##       node_id betweenness
## P0011   P0011 0.002580361
## P0060   P0060 0.002094395
## P0003   P0003 0.001932865
## P0002   P0002 0.001732924
## P0001   P0001 0.001339179
```

```r
cat("\nHighest Authority Score (most authoritative):\n")
```

```
##
## Highest Authority Score (most authoritative):
```

```r
top_authority <- centrality_summary[order(centrality_summary$authority, decreasing = TRUE)[1:5],
                                   c("node_id", "authority")]
print(top_authority)
```

```
##       node_id authority
## P0009   P0009 1.0000000
## P0060   P0060 0.9173024
## P0017   P0017 0.6726492
## P0014   P0014 0.6544079
## P0006   P0006 0.6283538
```

```r
cat("=== NODE BETWEENNESS DETAILED ANALYSIS ===\n")
```

```
## === NODE BETWEENNESS DETAILED ANALYSIS ===
```

```r
# Detailed betweenness analysis
betweenness_stats <- summary(betweenness_cent)
cat("Betweenness centrality statistics:\n")
```

```
## Betweenness centrality statistics:
```

```r
print(betweenness_stats)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.00e+00 0.00e+00 0.00e+00 2.12e-05 3.66e-06 2.58e-03
```

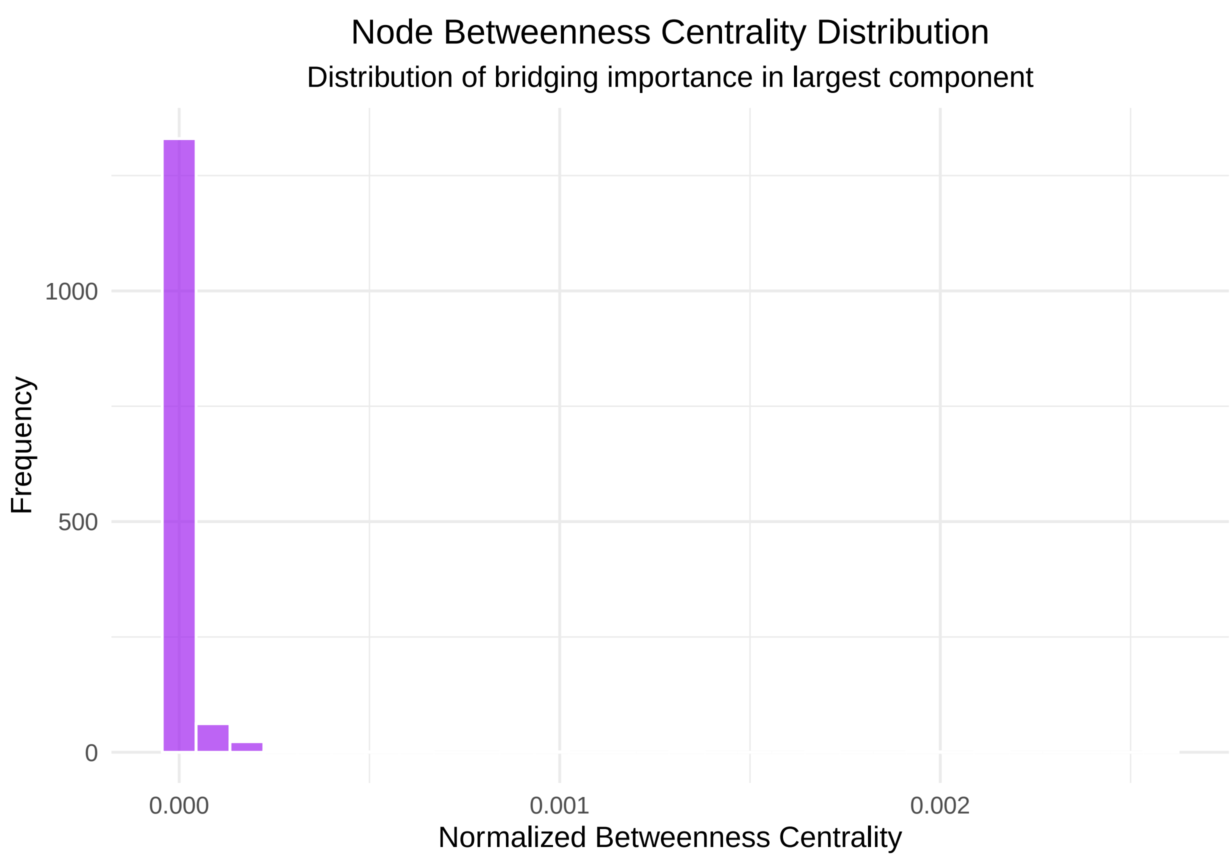
```r
cat("\nNodes with highest betweenness (potential bridges):\n")
```

```
##
## Nodes with highest betweenness (potential bridges):
```

```r
high_betweenness_threshold <- quantile(betweenness_cent, 0.95)
high_betweenness_nodes <- which(betweenness_cent >= high_betweenness_threshold)
cat("Number of high betweenness nodes (top 5%):", length(high_betweenness_nodes), "\n")
```

```
## Number of high betweenness nodes (top 5%): 72
```

```r
# Create betweenness distribution plot
betweenness_df <- data.frame(
  node_id = V(largest_component)$name,
  betweenness = betweenness_cent,
  institution = V(largest_component)$institution,
  subtopic = V(largest_component)$subtopic
)

# Betweenness distribution histogram
ggplot(betweenness_df, aes(x = betweenness)) +
  geom_histogram(bins = 30, fill = "purple", alpha = 0.7, color = "white") +
  labs(title = "Node Betweenness Centrality Distribution",
       subtitle = "Distribution of bridging importance in largest component",
       x = "Normalized Betweenness Centrality",
       y = "Frequency") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```



```r
# Identify top bridging papers
cat("\nTop 10 bridging papers (highest betweenness):\n")
```

```
##
## Top 10 bridging papers (highest betweenness):
```

```r
top_bridges <- betweenness_df[order(betweenness_df$betweenness, decreasing = TRUE)[1:10], ]
print(top_bridges[, c("node_id", "betweenness", "subtopic", "institution")])
```

```
##       node_id  betweenness                                     subtopic
## P0011   P0011 0.0025803608                   Machine Learning in Healthcare
## P0060   P0060 0.0020943953 Artificial Intelligence in Healthcare and Education
## P0003   P0003 0.0019328651 Artificial Intelligence in Healthcare and Education
## P0002   P0002 0.0017329242 Artificial Intelligence in Healthcare and Education
## P0001   P0001 0.0013391788                   Machine Learning in Healthcare
## P0192   P0192 0.0009925863 Artificial Intelligence in Healthcare and Education
## P0484   P0484 0.0008632584 Artificial Intelligence in Healthcare and Education
## P0054   P0054 0.0008526921 Artificial Intelligence in Healthcare and Education
## P0136   P0136 0.0008494446 Artificial Intelligence in Healthcare and Education
## P0021   P0021 0.0006611386 Artificial Intelligence in Healthcare and Education
##                   institution
## P0011     Stanford University
## P0060 University Of Cambridge
## P0003         Yale University
## P0002      Harvard University
## P0001     Stanford University
## P0192  Mayo Clinic In Arizona
## P0484     Stanford Health Care
## P0054     Stanford University
## P0136     Stanford University
## P0021        Emory University
```

```r
cat("\n=== EDGE BETWEENNESS ANALYSIS ===\n")
```

```
##
## === EDGE BETWEENNESS ANALYSIS ===
```

```r
# Calculate edge betweenness
cat("Calculating edge betweenness (this may take a moment)...\n")
```

```
## Calculating edge betweenness (this may take a moment)...
```

```r
edge_betweenness <- edge_betweenness(largest_component, directed = TRUE)

cat("Edge betweenness calculation completed!\n")
```

```
## Edge betweenness calculation completed!
```

```r
cat("Number of edges analyzed:", length(edge_betweenness), "\n")
```

```
## Number of edges analyzed: 3634
```

```r
# Edge betweenness statistics
edge_bet_stats <- summary(edge_betweenness)
cat("\nEdge betweenness statistics:\n")
```

```
##
## Edge betweenness statistics:
```

```r
print(edge_bet_stats)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##    1.000    2.000    5.667   25.071   17.462 2536.235
```

```r
# Find edges with highest betweenness
top_edge_indices <- order(edge_betweenness, decreasing = TRUE)[1:10]
top_edges <- get.edges(largest_component, top_edge_indices)

cat("\nTop 10 edges by betweenness (most critical connections):\n")
```

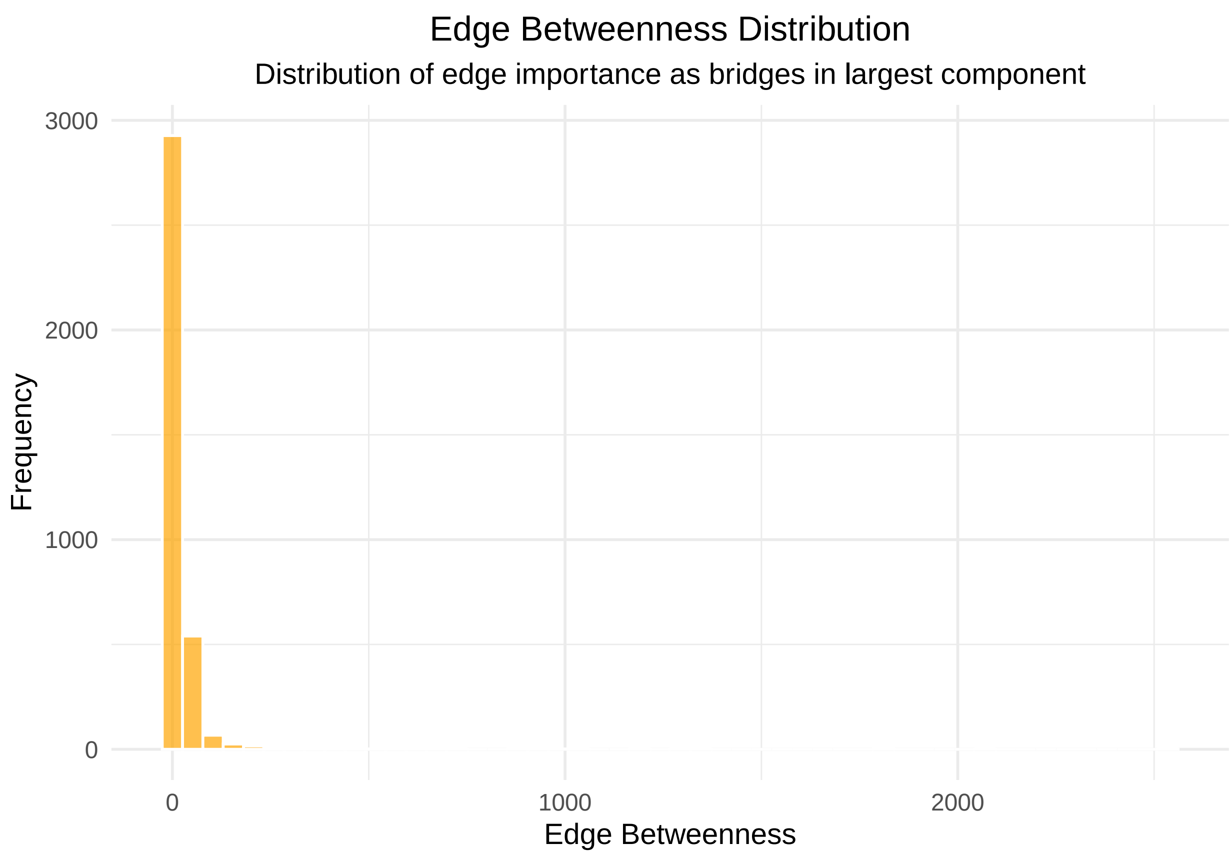
```
##
## Top 10 edges by betweenness (most critical connections):
```

```r
for(i in 1:10) {
  edge_idx <- top_edge_indices[i]
  from_node <- V(largest_component)$name[top_edges[i, 1]]
  to_node <- V(largest_component)$name[top_edges[i, 2]]
  bet_value <- round(edge_betweenness[edge_idx], 4)
  cat(sprintf("%d. %s -> %s (betweenness: %.4f)\n", i, from_node, to_node, bet_value))
}
```

```
## 1. P0001 -> P0003 (betweenness: 2536.2345)
## 2. P0003 -> P0054 (betweenness: 2051.3298)
## 3. P0484 -> P0011 (betweenness: 1825.9821)
## 4. P0192 -> P0136 (betweenness: 1811.0000)
## 5. P0011 -> P0192 (betweenness: 1789.1667)
## 6. P0003 -> P0021 (betweenness: 1346.8333)
## 7. P0002 -> P0107 (betweenness: 1293.0250)
## 8. P0060 -> P0001 (betweenness: 1211.1667)
## 9. P0060 -> P0002 (betweenness: 1039.8810)
## 10. P0011 -> P0003 (betweenness: 1002.4452)
```

```r
# Edge betweenness distribution plot
edge_bet_df <- data.frame(edge_betweenness = edge_betweenness)
ggplot(edge_bet_df, aes(x = edge_betweenness)) +
  geom_histogram(bins = 50, fill = "orange", alpha = 0.7, color = "white") +
  labs(title = "Edge Betweenness Distribution",
       subtitle = "Distribution of edge importance as bridges in largest component",
```

```
      x = "Edge Betweenness",
      y = "Frequency") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```




```
cat("\n=== COMMUNITY DETECTION - METHOD 1: LOUVAIN ALGORITHM ===\n")
```

```
##
## === COMMUNITY DETECTION - METHOD 1: LOUVAIN ALGORITHM ===
```

```
# Method 1: Louvain algorithm (modularity optimization)
cat("Running Louvain algorithm for community detection...\n")
```

```
## Running Louvain algorithm for community detection...
```

```
# Convert to undirected for community detection
largest_component_undirected <- as.undirected(largest_component, mode = "collapse")
```

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

```r
# Louvain algorithm
louvain_communities <- cluster_louvain(largest_component_undirected)

cat("Louvain algorithm completed!\n")
```

```
## Louvain algorithm completed!
```

```r
cat("Number of communities found:", length(louvain_communities), "\n")
```

```
## Number of communities found: 21
```

```r
cat("Modularity score:", round(modularity(louvain_communities), 4), "\n")
```

```
## Modularity score: 0.576
```

```r
# Community size distribution
louvain_sizes <- sizes(louvain_communities)
cat("\nCommunity sizes:\n")
```

```
##
## Community sizes:
```

```r
print(sort(louvain_sizes, decreasing = TRUE))
```

```
## Community sizes
##   1   5   9   7   8   2   4  11  10  12   3   6  14  18  20  15  19  13  17  21
## 219 137 137 129 121  99  93  88  66  48  46  44  41  38  38  34  19  15   9   8
##  16
##   4
```

```r
cat("\nLargest 5 communities:\n")
```

```
##
## Largest 5 communities:
```

```r
large_communities <- sort(louvain_sizes, decreasing = TRUE)[1:5]
print(large_communities)
```

```
## Community sizes
##   1   5   9   7   8
## 219 137 137 129 121
```

```r
# Create community membership dataframe
louvain_membership <- data.frame(
  node_id = V(largest_component_undirected)$name,
  community = membership(louvain_communities),
  institution = V(largest_component_undirected)$institution,
  subtopic = V(largest_component_undirected)$subtopic
)

# Analyze community composition by subtopic
cat("\nCommunity composition analysis:\n")
```

```
##
## Community composition analysis:
```

```r
for(i in 1:min(5, length(louvain_communities))) {
  cat(sprintf("\nCommunity %d (size: %d):\n", i, louvain_sizes[i]))
  community_nodes <- louvain_membership[louvain_membership$community == i, ]
  subtopic_dist <- table(community_nodes$subtopic)
  cat("Main subtopics:\n")
  print(sort(subtopic_dist, decreasing = TRUE)[1:min(3, length(subtopic_dist))])
}
```

```
##
## Community 1 (size: 219):
## Main subtopics:
##
## Artificial Intelligence in Healthcare and Education
##                                                  187
##                      Machine Learning in Healthcare
##                                                   25
##               Artificial Intelligence in Healthcare
##                                                    2
##
## Community 2 (size: 99):
## Main subtopics:
##
## Artificial Intelligence in Healthcare and Education
##                                                   82
##                      Machine Learning in Healthcare
##                                                   11
##               Artificial Intelligence in Healthcare
##                                                    3
##
## Community 3 (size: 46):
## Main subtopics:
## Machine Learning in Materials Science
##                                    46
##
## Community 4 (size: 93):
## Main subtopics:
##
## Artificial Intelligence in Healthcare and Education
##                                                   82
##                      Machine Learning in Healthcare
##                                                    6
##               Artificial Intelligence in Healthcare
##                                                    4
##
## Community 5 (size: 137):
## Main subtopics:
##
## Artificial Intelligence in Healthcare and Education
##                                                  124
##                      Machine Learning in Healthcare
##                                                   12
```

```
##            Machine Learning and Data Classification
##                                                   1
```

```r
cat("\n=== COMMUNITY DETECTION - METHOD 2: EDGE BETWEENNESS ===\n")
```

```
##
## === COMMUNITY DETECTION - METHOD 2: EDGE BETWEENNESS ===
```

```r
# Method 2: Edge betweenness-based community detection
cat("Running edge betweenness community detection...\n")
```

```
## Running edge betweenness community detection...
```

```r
# This method removes edges with highest betweenness iteratively
edge_betweenness_communities <- cluster_edge_betweenness(largest_component_undirected,
                                                         directed = FALSE)

cat("Edge betweenness algorithm completed!\n")
```

```
## Edge betweenness algorithm completed!
```

```r
cat("Number of communities found:", length(edge_betweenness_communities), "\n")
```

```
## Number of communities found: 65
```

```r
cat("Modularity score:", round(modularity(edge_betweenness_communities), 4), "\n")
```

```
## Modularity score: 0.5455
```

```r
# Community size distribution
eb_sizes <- sizes(edge_betweenness_communities)
cat("\nCommunity sizes:\n")
```

```
##
## Community sizes:
```

```r
print(sort(eb_sizes, decreasing = TRUE))
```

```
## Community sizes
##   6   1   5  11   4  12   2  34   9   7  31  26  42  46  19  43   3  25  41   8
## 282 238 150  50  46  35  32  31  27  26  24  22  21  21  20  19  15  15  15  14
##  20  48  13  21  27  10  49  15  60  44  22  28  32  51  52  59  14  23  37  40
##  14  14  13  13  13  12  12  11  11  10   9   9   9   9   9   9   8   8   8   8
##  50  24  38  53  29  47  16  17  33  36  45  57  58  61  18  30  35  39  54  56
##   8   7   7   7   6   6   5   5   5   5   5   5   5   5   4   4   4   4   4   4
##  64  55  62  63  65
##   4   3   3   3   3
```

```r
# Compare the two methods
cat("\n=== COMPARISON OF COMMUNITY DETECTION METHODS ===\n")
```

```
##
## === COMPARISON OF COMMUNITY DETECTION METHODS ===
```

```r
cat("Louvain - Communities:", length(louvain_communities),
    "| Modularity:", round(modularity(louvain_communities), 4), "\n")
```

```
## Louvain - Communities: 21 | Modularity: 0.576
```

```r
cat("Edge Betweenness - Communities:", length(edge_betweenness_communities),
    "| Modularity:", round(modularity(edge_betweenness_communities), 4), "\n")
```

```
## Edge Betweenness - Communities: 65 | Modularity: 0.5455
```

```r
# Create comparison plot
community_comparison <- data.frame(
  Method = c("Louvain", "Edge Betweenness"),
  Communities = c(length(louvain_communities), length(edge_betweenness_communities)),
  Modularity = c(modularity(louvain_communities), modularity(edge_betweenness_communities))
)

# Plot comparison
p1 <- ggplot(community_comparison, aes(x = Method, y = Communities, fill = Method)) +
  geom_bar(stat = "identity", alpha = 0.7) +
  scale_fill_manual(values = c("Louvain" = "lightblue",
                               "Edge Betweenness" = "lightcoral")) +
  labs(title = "Number of Communities",
       y = "Number of Communities") +
  theme_minimal() +
  theme(legend.position = "none")

p2 <- ggplot(community_comparison, aes(x = Method, y = Modularity, fill = Method)) +
  geom_bar(stat = "identity", alpha = 0.7) +
  scale_fill_manual(values = c("Louvain" = "lightblue",
                               "Edge Betweenness" = "lightcoral")) +
  labs(title = "Modularity Score",
       y = "Modularity") +
  theme_minimal() +
  theme(legend.position = "none")

# Display plots side by side
grid.arrange(p1, p2, ncol = 2,
             top = "Community Detection Methods Comparison")
```


```r
cat("\n=== CENTRALITY MEASURES CORRELATION ANALYSIS ===\n")
```

```
##
## === CENTRALITY MEASURES CORRELATION ANALYSIS ===
```

```r
# Analyze correlations between different centrality measures
centrality_cor_data <- centrality_summary[,
                              c("degree_in", "degree_out", "betweenness",
                                "closeness_in", "closeness_out",
                                "eigenvector", "pagerank", "authority", "hub")]

# Calculate correlation matrix
cor_matrix <- cor(centrality_cor_data, use = "complete.obs")
```

```
## Warning in cor(centrality_cor_data, use = "complete.obs"):
```

```r
cat("Correlation matrix calculated\n")
```

```
## Correlation matrix calculated
```

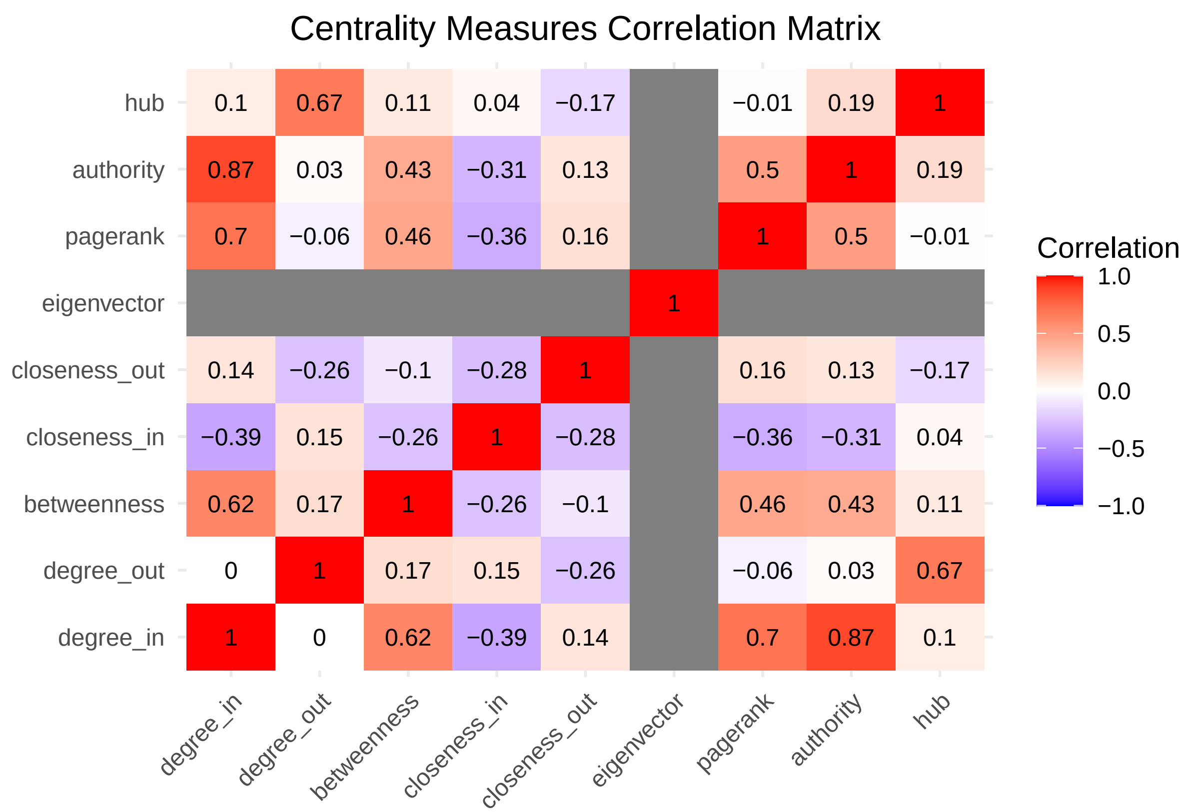
```r
print(round(cor_matrix, 3))
```

```
##                degree_in degree_out betweenness closeness_in closeness_out
```

```
## degree_in          1.000      0.002        0.616        -0.390          0.138
## degree_out         0.002      1.000        0.173         0.152         -0.262
## betweenness        0.616      0.173        1.000        -0.260         -0.101
## closeness_in      -0.390      0.152       -0.260         1.000         -0.277
## closeness_out      0.138     -0.262       -0.101        -0.277          1.000
## eigenvector           NA         NA           NA            NA             NA
## pagerank           0.696     -0.057        0.460        -0.356          0.162
## authority          0.869      0.029        0.435        -0.311          0.125
## hub                0.096      0.670        0.110         0.036         -0.170
##                eigenvector pagerank authority    hub
## degree_in               NA    0.696     0.869  0.096
## degree_out              NA   -0.057     0.029  0.670
## betweenness             NA    0.460     0.435  0.110
## closeness_in            NA   -0.356    -0.311  0.036
## closeness_out           NA    0.162     0.125 -0.170
## eigenvector              1       NA        NA     NA
## pagerank                NA    1.000     0.498 -0.012
## authority               NA    0.498     1.000  0.187
## hub                     NA   -0.012     0.187  1.000
```

```r
# Create correlation heatmap
cor_melted <- melt(cor_matrix)
ggplot(cor_melted, aes(Var1, Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1),
        plot.title = element_text(hjust = 0.5)) +
  labs(title = "Centrality Measures Correlation Matrix",
       x = "", y = "") +
  geom_text(aes(label = round(value, 2)), size = 3)
```

```
## Warning: Removed 16 rows containing missing values or values outside the scale range
## (`geom_text()`).
```



Largest Component Visualization (Jinxi Hu)

```
cat("=== LARGEST COMPONENT VISUALIZATION ===\n")
```

```
## === LARGEST COMPONENT VISUALIZATION ===
```

```
# Basic layout for consistent visualization across all plots
cat("Calculating network layout (this may take a moment)...\n")
```

```
## Calculating network layout (this may take a moment)...
```

```
layout_large <- layout_with_fr(largest_component,
                               weights = NULL,
                               niter = 1000,
                               start.temp = sqrt(vcount(largest_component)))

cat("Layout calculation completed!\n")
```

```
## Layout calculation completed!
```

```r
cat("Plotting largest component with different color schemes...\n\n")
```

```
## Plotting largest component with different color schemes...
```

```r
# Basic plot with default colors
plot(largest_component,
     layout = layout_large,
     vertex.size = 4,
     vertex.label = NA,
     edge.arrow.size = 0.3,
     edge.width = 0.5,
     edge.color = "gray70",
     vertex.color = "lightblue",
     vertex.frame.color = "white",
     main = "Largest Component - Basic View",
     sub = paste("Nodes:", vcount(largest_component),
                 "| Edges:", ecount(largest_component)))
```

Largest Component − Basic View

Nodes: 1433 | Edges: 3634

```r
# Plot colored by research subtopic
cat("1. VISUALIZATION BY RESEARCH SUBTOPIC\n")
```

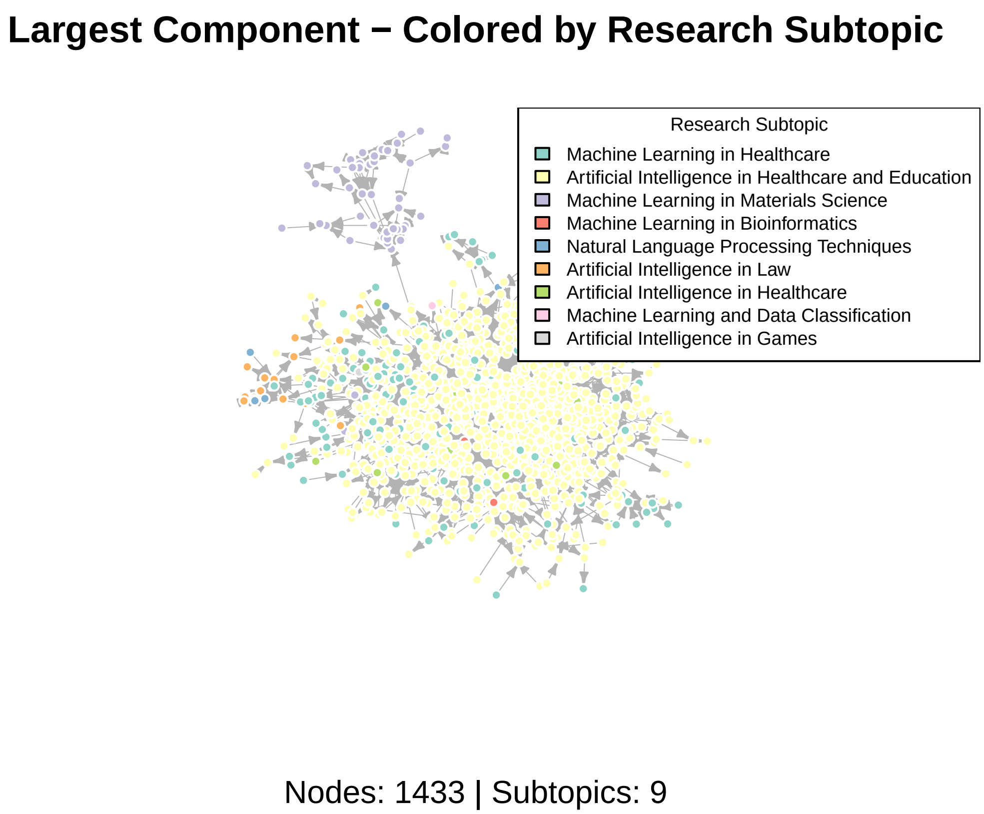
```
## 1. VISUALIZATION BY RESEARCH SUBTOPIC
```

```r
# Get unique subtopics and create color palette
subtopics_large <- V(largest_component)$subtopic
unique_subtopics <- unique(subtopics_large)
n_subtopics <- length(unique_subtopics)

cat("Number of unique subtopics in largest component:", n_subtopics, "\n")
```

```
## Number of unique subtopics in largest component: 9
```

```r
# Create color palette for subtopics
if(n_subtopics <= 12) {
  colors_subtopic <- RColorBrewer::brewer.pal(max(3, n_subtopics), "Set3")
} else {
  # For more than 12 subtopics, use rainbow colors
  colors_subtopic <- rainbow(n_subtopics)
}

# Create named color vector to ensure consistent mapping
names(colors_subtopic) <- unique_subtopics

# Assign colors to nodes using the named vector
vertex_colors_subtopic <- colors_subtopic[subtopics_large]

# Plot by subtopic
plot(largest_component,
     layout = layout_large,
     vertex.size = 4,
     vertex.label = NA,
     edge.arrow.size = 0.3,
     edge.width = 0.5,
     edge.color = "gray70",
     vertex.color = vertex_colors_subtopic,
     vertex.frame.color = "white",
     main = "Largest Component - Colored by Research Subtopic",
     sub = paste("Nodes:", vcount(largest_component),
                 "| Subtopics:", n_subtopics))

# Add legend for subtopics (show top 10 only if too many)
if(n_subtopics > 10) {
  # Show only the most frequent subtopics
  top_subtopics <- names(sort(table(subtopics_large), decreasing = TRUE))[1:10]
  legend_subtopics <- c(top_subtopics, "Others...")
  legend_colors <- c(colors_subtopic[top_subtopics], "gray")
} else {
  legend_subtopics <- unique_subtopics
  legend_colors <- colors_subtopic[unique_subtopics]
}

legend("topright",
       legend = legend_subtopics,
       fill = legend_colors,
       cex = 0.6,
       title = "Research Subtopic",
```

```r
       bg = "white")
```




```r
# Print subtopic distribution
cat("\nSubtopic distribution in largest component:\n")
```

```
##
## Subtopic distribution in largest component:
```

```r
subtopic_dist <- sort(table(subtopics_large), decreasing = TRUE)
print(head(subtopic_dist, 10))
```

```
## subtopics_large
## Artificial Intelligence in Healthcare and Education
##                                                1171
##                       Machine Learning in Healthcare
##                                                 153
##                Machine Learning in Materials Science
##                                                  49
##               Natural Language Processing Techniques
##                                                  21
##                Artificial Intelligence in Healthcare
##                                                  17
##                       Artificial Intelligence in Law
##                                                  16
```

```
##                  Machine Learning in Bioinformatics
##                                                   3
##            Machine Learning and Data Classification
##                                                   2
##                    Artificial Intelligence in Games
##                                                   1
```

```r
# Plot colored by institution
cat("\n2. VISUALIZATION BY INSTITUTION\n")
```

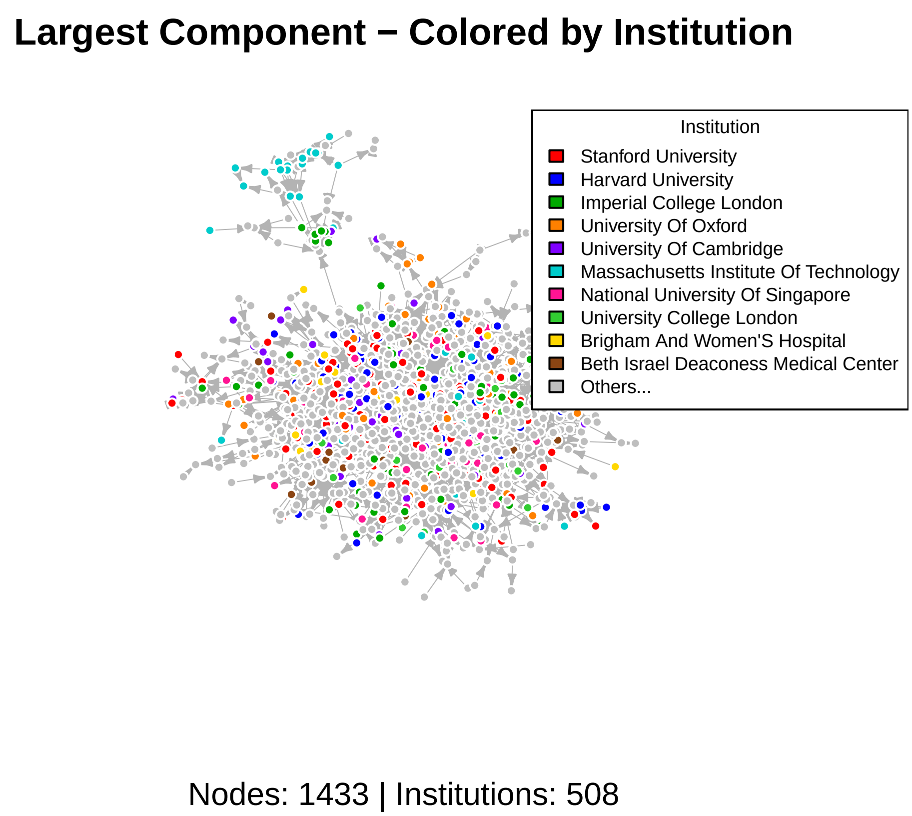
```
##
## 2. VISUALIZATION BY INSTITUTION
```

```r
# Get unique institutions and create color palette
institutions_large <- V(largest_component)$institution
unique_institutions <- unique(institutions_large)
n_institutions <- length(unique_institutions)

cat("Number of unique institutions in largest component:", n_institutions, "\n")
```

```
## Number of unique institutions in largest component: 508
```

```r
# Get top 10 institutions by frequency first
institution_counts <- sort(table(institutions_large), decreasing = TRUE)
if(n_institutions > 10) {
  top_institutions <- names(institution_counts)[1:10]
} else {
  top_institutions <- names(institution_counts)
}

# Create color palette for institutions with better differentiation
if(length(top_institutions) <= 12) {
  # Use highly contrasting colors with maximum visual separation
  colors_for_top <- c("#FF0000", "#0000FF", "#00AA00", "#FF8000", "#8000FF",
                     "#00CCCC", "#FF1493", "#32CD32", "#FFD700", "#8B4513",
                     "#FF69B4", "#4169E1")
  colors_top_institutions <- colors_for_top[1:length(top_institutions)]
  names(colors_top_institutions) <- top_institutions
} else {
  # For more than 12 institutions, use diverse colors with better spacing
  colors_top_institutions <- rainbow(length(top_institutions), s = 1, v = 0.8)
  names(colors_top_institutions) <- top_institutions
}

# Assign colors to nodes
vertex_colors_display <- rep("gray", length(institutions_large))
for(i in 1:length(institutions_large)) {
  if(institutions_large[i] %in% top_institutions) {
    vertex_colors_display[i] <- colors_top_institutions[institutions_large[i]]
  }
}
```

```r
# Plot by institution
plot(largest_component,
     layout = layout_large,
     vertex.size = 4,
     vertex.label = NA,
     edge.arrow.size = 0.3,
     edge.width = 0.5,
     edge.color = "gray70",
     vertex.color = vertex_colors_display,
     vertex.frame.color = "white",
     main = "Largest Component - Colored by Institution",
     sub = paste("Nodes:", vcount(largest_component),
                "| Institutions:", n_institutions))

# Add legend for institutions
if(n_institutions > 10) {
  legend_institutions <- c(top_institutions, "Others...")
  legend_colors_inst <- c(colors_top_institutions[top_institutions], "gray")
} else {
  legend_institutions <- top_institutions
  legend_colors_inst <- colors_top_institutions[top_institutions]
}

legend("topright",
       legend = legend_institutions,
       fill = legend_colors_inst,
       cex = 0.6,
       title = "Institution",
       bg = "white")
```




```r
# Print institution distribution
cat("\nInstitution distribution in largest component:\n")
```

```
##
## Institution distribution in largest component:
```

```r
institution_dist <- sort(table(institutions_large), decreasing = TRUE)
print(head(institution_dist, 10))
```

```
## institutions_large
##                 Stanford University                    Harvard University
##                                 115                                    88
##             Imperial College London                  University Of Oxford
##                                  63                                    59
##             University Of Cambridge Massachusetts Institute Of Technology
##                                  48                                    38
##      National University Of Singapore               University College London
##                                  33                                    25
##         Brigham And Women'S Hospital   Beth Israel Deaconess Medical Center
##                                  24                                    19
```

```r
# Plot colored by Louvain communities
cat("\n3. VISUALIZATION BY LOUVAIN COMMUNITIES\n")
```

```
##
## 3. VISUALIZATION BY LOUVAIN COMMUNITIES
```

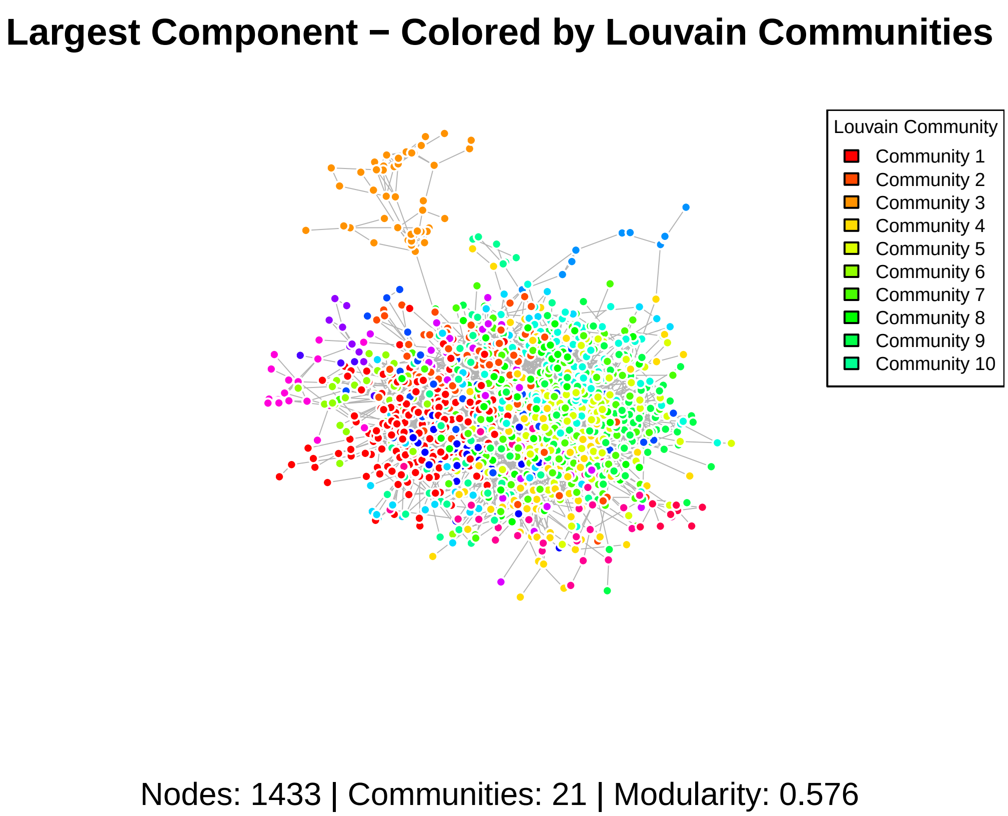
```r
# Get community membership (already calculated in previous chunk)
community_membership <- membership(louvain_communities)
n_communities <- length(louvain_communities)

cat("Number of Louvain communities:", n_communities, "\n")
```

```
## Number of Louvain communities: 21
```

```r
cat("Modularity score:", round(modularity(louvain_communities), 4), "\n")
```

```
## Modularity score: 0.576
```

```r
# Create color palette for communities
if(n_communities <= 12) {
  colors_community <- RColorBrewer::brewer.pal(max(3, n_communities), "Set1")
} else {
  colors_community <- rainbow(n_communities)
}

# Assign colors to nodes based on community membership
vertex_colors_community <- colors_community[community_membership]

# Plot by community
plot(largest_component_undirected,  # Use undirected version for community plot
     layout = layout_large,
     vertex.size = 4,
     vertex.label = NA,
     edge.width = 0.5,
     edge.color = "gray70",
     vertex.color = vertex_colors_community,
     vertex.frame.color = "white",
     main = "Largest Component - Colored by Louvain Communities",
     sub = paste("Nodes:", vcount(largest_component),
                 "| Communities:", n_communities,
                 "| Modularity:", round(modularity(louvain_communities), 3)))

# Add legend for communities
legend("topright",
       legend = paste("Community", 1:min(n_communities, 10)),
       fill = colors_community[1:min(n_communities, 10)],
       cex = 0.6,
       title = "Louvain Community",
       bg = "white")
```




```r
# Print community size distribution
cat("\nCommunity size distribution:\n")
```

```
##
## Community size distribution:
```

```r
community_sizes <- sort(sizes(louvain_communities), decreasing = TRUE)
print(community_sizes)
```

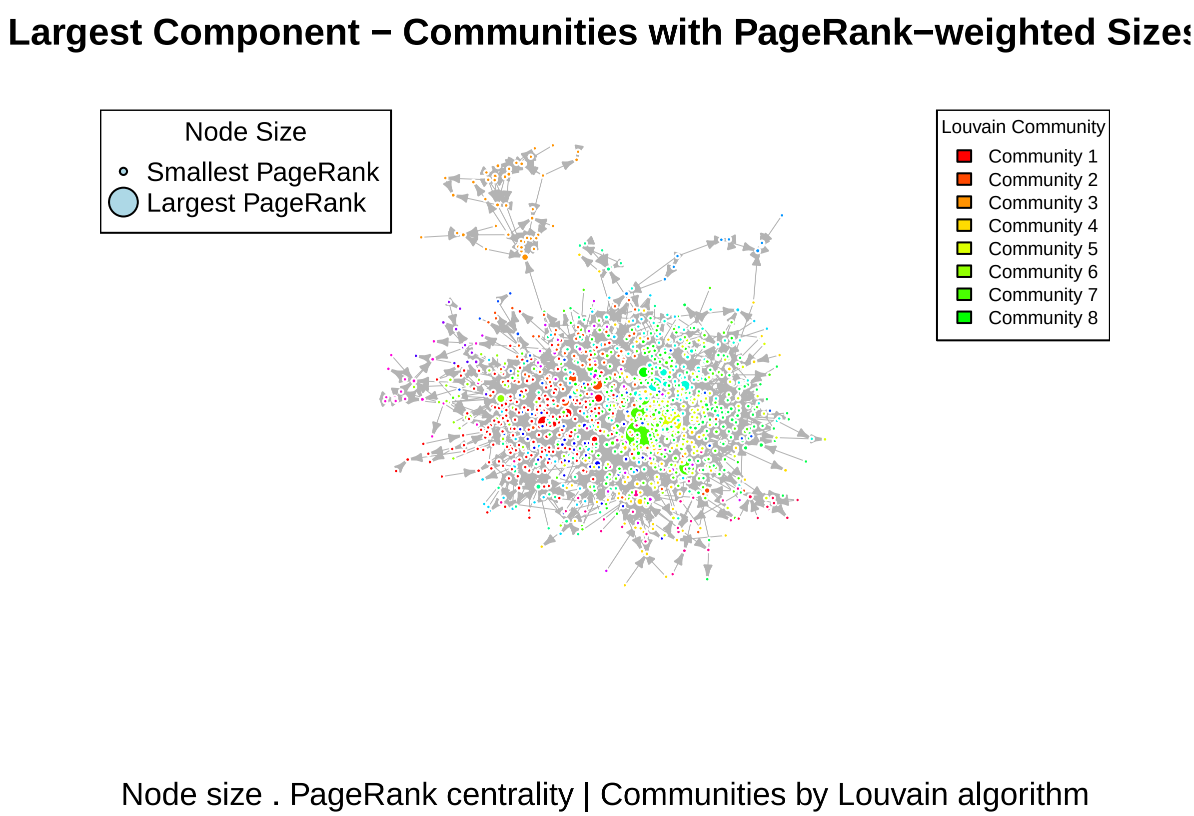
```
## Community sizes
##   1   5   9   7   8   2   4  11  10  12   3   6  14  18  20  15  19  13  17  21
## 219 137 137 129 121  99  93  88  66  48  46  44  41  38  38  34  19  15   9   8
##  16
##   4
```

```r
# Plot with node sizes representing centrality measures
cat("\n4. VISUALIZATION WITH CENTRALITY-WEIGHTED NODE SIZES\n")
```

```
##
## 4. VISUALIZATION WITH CENTRALITY-WEIGHTED NODE SIZES
```

```r
# Use PageRank centrality for node sizes
pagerank_values <- page_rank(largest_component, directed = TRUE)$vector
```

```r
# Scale node sizes appropriately (between 2 and 15)
node_sizes <- 2 + 13 * (pagerank_values - min(pagerank_values)) /
              (max(pagerank_values) - min(pagerank_values))

# Plot with PageRank-weighted sizes and community colors
plot(largest_component,
     layout = layout_large,
     vertex.size = node_sizes,
     vertex.label = NA,
     edge.arrow.size = 0.3,
     edge.width = 0.5,
     edge.color = "gray70",
     vertex.color = vertex_colors_community,
     vertex.frame.color = "white",
     main = "Largest Component - Communities with PageRank-weighted Sizes",
     sub = paste("Node size   PageRank centrality | Communities by Louvain algorithm"))

# Add legends
legend("topleft",
       legend = c("Smallest PageRank", "Largest PageRank"),
       pch = 21,
       pt.cex = c(0.5, 2),
       pt.bg = "lightblue",
       title = "Node Size",
       bg = "white",
       cex = 0.8)

legend("topright",
       legend = paste("Community", 1:min(n_communities, 8)),
       fill = colors_community[1:min(n_communities, 8)],
       cex = 0.6,
       title = "Louvain Community",
       bg = "white")
```




```r
# Print top PageRank nodes
cat("\nTop 10 nodes by PageRank centrality:\n")
```

```
##
## Top 10 nodes by PageRank centrality:
```

```r
top_pagerank_nodes <- order(pagerank_values, decreasing = TRUE)[1:10]
pagerank_summary <- data.frame(
  node_id = V(largest_component)$name[top_pagerank_nodes],
  pagerank = round(pagerank_values[top_pagerank_nodes], 4),
  community = community_membership[top_pagerank_nodes],
  subtopic = V(largest_component)$subtopic[top_pagerank_nodes],
  institution = V(largest_component)$institution[top_pagerank_nodes]
)
print(pagerank_summary)
```

```
##       node_id pagerank community
## P0199   P0199   0.0405         7
## P0031   P0031   0.0323         5
## P0741   P0741   0.0197         7
## P0116   P0116   0.0175         7
## P0535   P0535   0.0157        11
## P0012   P0012   0.0138         7
## P0060   P0060   0.0125         1
## P0002   P0002   0.0114         2
```

```
## P0015   P0015   0.0110         4
## P1179   P1179   0.0104        11
##                                                  subtopic
## P0199 Artificial Intelligence in Healthcare and Education
## P0031 Artificial Intelligence in Healthcare and Education
## P0741 Artificial Intelligence in Healthcare and Education
## P0116 Artificial Intelligence in Healthcare and Education
## P0535 Artificial Intelligence in Healthcare and Education
## P0012                      Machine Learning in Healthcare
## P0060 Artificial Intelligence in Healthcare and Education
## P0002 Artificial Intelligence in Healthcare and Education
## P0015 Artificial Intelligence in Healthcare and Education
## P1179               Artificial Intelligence in Healthcare
##                             institution
## P0199               Harvard University
## P0031          Imperial College London
## P0741               Harvard University
## P0116              Stanford University
## P0535           University Of Michigan
## P0012               Harvard University
## P0060          University Of Cambridge
## P0002               Harvard University
## P0015 National University Of Singapore
## P1179 National University Of Singapore
```

```r
cat("\n=== LARGEST COMPONENT ANALYSIS SUMMARY ===\n")
```

```
##
## === LARGEST COMPONENT ANALYSIS SUMMARY ===
```

```r
# Summary statistics
cat("STRUCTURAL PROPERTIES:\n")
```

```
## STRUCTURAL PROPERTIES:
```

```r
cat("- Nodes:", vcount(largest_component), "\n")
```

```
## - Nodes: 1433
```

```r
cat("- Edges:", ecount(largest_component), "\n")
```

```
## - Edges: 3634
```

```r
cat("- Density:", round(edge_density(largest_component), 6), "\n")
```

```
## - Density: 0.001771
```

```r
cat("- Average clustering coefficient:", round(transitivity(largest_component, type = "average"), 4), "
```

```
## - Average clustering coefficient: 0.1433
```

```r
cat("- Diameter:", diameter(largest_component, directed = FALSE), "\n")
```

```
## - Diameter: 13
```

```r
cat("- Average path length:", round(mean_distance(largest_component, directed = FALSE), 2), "\n\n")
```

```
## - Average path length: 4.5
```

```r
cat("DIVERSITY MEASURES:\n")
```

```
## DIVERSITY MEASURES:
```

```r
cat("- Research subtopics:", length(unique(subtopics_large)), "\n")
```

```
## - Research subtopics: 9
```

```r
cat("- Institutions:", length(unique(institutions_large)), "\n")
```

```
## - Institutions: 508
```

```r
cat("- Louvain communities:", n_communities, "\n")
```

```
## - Louvain communities: 21
```

```r
cat("- Community modularity:", round(modularity(louvain_communities), 4), "\n\n")
```

```
## - Community modularity: 0.576
```

```r
cat("CENTRALITY HIGHLIGHTS:\n")
```

```
## CENTRALITY HIGHLIGHTS:
```

```r
cat("- Highest PageRank node:", V(largest_component)$name[which.max(pagerank_values)],
    "(", round(max(pagerank_values), 4), ")\n")
```

```
## - Highest PageRank node: P0199 ( 0.0405 )
```

```r
cat("- Highest betweenness node:", V(largest_component)$name[which.max(betweenness_cent)],
    "(", round(max(betweenness_cent), 4), ")\n")
```

```
## - Highest betweenness node: P0011 ( 0.0026 )
```

```r
cat("- Most cited node (in-degree):", V(largest_component)$name[which.max(degree_cent_in)],
    "(", max(degree(largest_component, mode = "in")), " citations)\n")
```

```
## - Most cited node (in-degree): P0060 ( 104  citations)
```