Appendix 4: R-code

Carl Emil

2025-05-22

```
    inscriptions <- read_excel(target_file, sheet = "Sheet1")
    cat("✓ SUCCESS: Data loaded with", nrow(inscriptions), "rows and", ncol(inscri
ptions), "columns\n")

    # Show first few column names
    cat("First 5 columns:", paste(head(names(inscriptions), 5), collapse = ", "),
"\n")

  }, error = function(e) {
    cat("✗ LOADING ERROR:", e$message, "\n")

    # Try with the first available sheet instead
    if (length(sheets) > 0) {
      cat("Trying with first available sheet:", sheets[1], "\n")
      tryCatch({
        inscriptions <- read_excel(target_file, sheet = sheets[1])
        cat("✓ SUCCESS with sheet", sheets[1], ":", nrow(inscriptions), "rows\n")
      }, error = function(e2) {
        cat("✗ Still failed:", e2$message, "\n")
      })
    }
  })

} else {
  cat("✗ Target file NOT found\n")
  cat("Please check:\n")
  cat("  1. File name spelling (including .xlsx extension)\n")
  cat("  2. File location (is it in the working directory?)\n")
  cat("  3. File permissions\n")
}

# Load required libraries
library(readxl)      # For reading Excel files
library(dplyr)       # For data manipulation

##
## Vedhæfter pakke: 'dplyr'
```

```
## De følgende objekter er maskerede fra 'package:stats':
##
##     filter, lag

## De følgende objekter er maskerede fra 'package:base':
##
##     intersect, setdiff, setequal, union

library(igraph)      # For network analysis and basic plotting

##
## Vedhæfter pakke: 'igraph'

## De følgende objekter er maskerede fra 'package:dplyr':
##
##     as_data_frame, groups, union

## De følgende objekter er maskerede fra 'package:stats':
##
##     decompose, spectrum

## Det følgende objekt er maskeret fra 'package:base':
##
##     union

library(ggraph)      # For advanced network visualization with ggplot2 syntax

## Indlæser krævet pakke: ggplot2

library(tidygraph)   # For tidy network data manipulation

##
## Vedhæfter pakke: 'tidygraph'

## Det følgende objekt er maskeret fra 'package:igraph':
##
##     groups

## Det følgende objekt er maskeret fra 'package:stats':
##
##     filter

library(ggplot2)     # For advanced plotting
library(stringr)     # For string manipulation
library(ggrepel)     # For better label positioning
library(RColorBrewer) # For better color palettes
library(scales)      # For better axis formatting
library(gridExtra)   # For combining plots

##
## Vedhæfter pakke: 'gridExtra'
```

```
## Det følgende objekt er maskeret fra 'package:dplyr':
##
##     combine

library(viridis)     # For modern color schemes

## Indlæser krævet pakke: viridisLite

##
## Vedhæfter pakke: 'viridis'

## Det følgende objekt er maskeret fra 'package:scales':
##
##     viridis_pal

# Step 1: Read and examine the updated dataset
inscriptions <- read_excel("Inscriptions_combined_with_all_Ephesus.xlsx", sheet =
"Sheet1")

cat("=== COMPREHENSIVE DATASET CHARACTERISTICS ===\n")

## === COMPREHENSIVE DATASET CHARACTERISTICS ===

# Calculate comprehensive dataset statistics
dataset_characteristics <- function(data) {

  # Basic counts
  total_inscriptions <- nrow(data)

  # Cities
  cities_represented <- data %>%
    filter(!is.na(Findspot_city), Findspot_city != "None", Findspot_city != "#NA")
%>%
    pull(Findspot_city) %>%
    n_distinct()

  # Emperors mentioned (parse all emperor mentions)
  all_emperors <- data %>%
    filter(!is.na(Emperor_mentioned), Emperor_mentioned != "None", Emperor_mention
ed != "#NA") %>%
    pull(Emperor_mentioned) %>%
    str_split("[;,]") %>%
    unlist() %>%
    str_trim() %>%
    unique()
  all_emperors <- all_emperors[all_emperors != "" & all_emperors != "None"]
  emperors_mentioned <- length(all_emperors)

  # Local deities
```

```r
  all_deities <- data %>%
    filter(!is.na(Local_deities), Local_deities != "None", Local_deities != "#NA")
%>%
    pull(Local_deities) %>%
    str_split("[;,]") %>%
    unlist() %>%
    str_trim() %>%
    unique()
  all_deities <- all_deities[all_deities != "" & all_deities != "None"]
  local_deities_mentioned <- length(all_deities)

  # Date coverage
  inscriptions_with_dates <- data %>%
    filter(!is.na(Date_start)) %>%
    nrow()

  date_range <- data %>%
    filter(!is.na(Date_start)) %>%
    summarise(
      min_date = min(Date_start, na.rm = TRUE),
      max_date = max(Date_start, na.rm = TRUE)
    )

  # Structural contents (findspot structures)
  structural_contents <- data %>%
    filter(!is.na(Findspot_structure), Findspot_structure != "None", Findspot_stru
cture != "#NA") %>%
    pull(Findspot_structure) %>%
    str_split("[;,]") %>%
    unlist() %>%
    str_trim() %>%
    unique()
  structural_contents <- structural_contents[structural_contents != "" & structura
l_contents != "None"]
  structural_contents_count <- length(structural_contents)

  # Inscription types
  inscription_types <- data %>%
    filter(!is.na(Inscription_type), Inscription_type != "None", Inscription_type
!= "#NA") %>%
    pull(Inscription_type) %>%
    str_split("[;,]") %>%
    unlist() %>%
    str_trim() %>%
    unique()
  inscription_types <- inscription_types[inscription_types != "" & inscription_typ
es != "None"]
```

```r
  inscription_types_count <- length(inscription_types)

  # Inscriptions with both imperial and local elements
  both_imperial_local <- data %>%
    filter(
      !is.na(Emperor_mentioned), Emperor_mentioned != "None", Emperor_mentioned !=
"#NA",
      !is.na(Local_deities), Local_deities != "None", Local_deities != "#NA"
    ) %>%
    nrow()

  # Create summary table
  characteristics_table <- data.frame(
    Characteristic = c(
      "Total Inscriptions",
      "Cities represented",
      "Emperors mentioned",
      "Local Deities mentioned",
      "Inscriptions with dates",
      "Structural contents",
      "Inscription types",
      "Date range",
      "Inscriptions with Both Imperial and Local Elements"
    ),
    Count_Description = c(
      as.character(total_inscriptions),
      as.character(cities_represented),
      as.character(emperors_mentioned),
      as.character(local_deities_mentioned),
      as.character(inscriptions_with_dates),
      as.character(structural_contents_count),
      as.character(inscription_types_count),
      paste(date_range$min_date, "to", date_range$max_date, "CE"),
      as.character(both_imperial_local)
    ),
    stringsAsFactors = FALSE
  )

  return(list(
    table = characteristics_table,
    cities = unique(data$Findspot_city[!is.na(data$Findspot_city) & data$Findspot_
city != "None"]),
    emperors = all_emperors,
    deities = all_deities,
    structures = structural_contents,
    types = inscription_types
  ))
```

```r
}

# Generate the characteristics
dataset_summary <- dataset_characteristics(inscriptions)

# Display the characteristics table
cat("Dataset Characteristics Summary:\n")

## Dataset Characteristics Summary:

cat(paste(rep("=", 60), collapse = ""), "\n")

## ============================================================

print(format(dataset_summary$table, width = 20, justify = "left"), row.names = FAL
SE)

##                                      Characteristic    Count_Description
##  Total Inscriptions                                  70
##  Cities represented                                  4
##  Emperors mentioned                                  29
##  Local Deities mentioned                             18
##  Inscriptions with dates                             70
##  Structural contents                                 28
##  Inscription types                                   37
##  Date range                                          -50 to 340 CE
##  Inscriptions with Both Imperial and Local Elements 25

cat(paste(rep("=", 60), collapse = ""), "\n\n")

## ============================================================

# Additional details
cat("CITIES REPRESENTED:\n")

## CITIES REPRESENTED:

cat(paste(dataset_summary$cities, collapse = ", "), "\n\n")

## Aphrodisias, Boubon, Ephesus, Pergamon

cat("TOP 10 EMPERORS MENTIONED:\n")

## TOP 10 EMPERORS MENTIONED:

emperor_counts <- inscriptions %>%
  filter(!is.na(Emperor_mentioned), Emperor_mentioned != "None") %>%
  pull(Emperor_mentioned) %>%
  str_split("[;,]") %>%
  unlist() %>%
  str_trim() %>%
```

```
  table() %>%
  sort(decreasing = TRUE) %>%
  head(10)
print(emperor_counts)
```

```
## .
##   Hadrian  Augustus  Tiberius    Trajan  Domitian Gallienus      Nero  Commodus
##        10         9         7         6         5         4         4         3
## Vespasian Caracalla
##         3         2
```

```
cat("\nTOP 8 LOCAL DEITIES MENTIONED:\n")
```

```
##
## TOP 8 LOCAL DEITIES MENTIONED:
```

```
deity_counts <- inscriptions %>%
  filter(!is.na(Local_deities), Local_deities != "None") %>%
  pull(Local_deities) %>%
  str_split("[;,]") %>%
  unlist() %>%
  str_trim() %>%
  table() %>%
  sort(decreasing = TRUE) %>%
  head(8)
print(deity_counts)
```

```
## .
##   Artemis Aphrodite      Roma  Augustus      Hera    Aeneas      Ares     Demos
##         7         6         4         2         2         1         1         1
```

```
cat("\nTOP STRUCTURAL CONTEXTS:\n")
```

```
##
## TOP STRUCTURAL CONTEXTS:
```

```
structure_counts <- inscriptions %>%
  filter(!is.na(Findspot_structure), Findspot_structure != "None") %>%
  pull(Findspot_structure) %>%
  str_split("[;,]") %>%
  unlist() %>%
  str_trim() %>%
  table() %>%
  sort(decreasing = TRUE) %>%
  head(8)
print(structure_counts)
```

```
## .
##                  Sebasteion          Theatre_Aphrodisias
##                          20                            8
```

```
##  Boubon_Sebasteion_East_Wall Boubon_Sebasteion_North_Wall
##                            7                            7
##             Temple_of_Trajan                       Agora
##                            3                            2
##                    Augusteum                   Hadrianeion
##                            2                            2
```

```
cat("\n")
```

```
# Step 2: Enhanced data preprocessing with better data quality checks
inscriptions_clean <- inscriptions %>%
  filter(
    !is.na(Findspot_city) &
    !is.na(Emperor_mentioned) &
    Findspot_city != "None" &
    Emperor_mentioned != "None" &
    Findspot_city != "#NA" &
    Emperor_mentioned != "#NA"
  ) %>%
  # Add enhanced chronological categories
  mutate(
    period = case_when(
      Date_start <= 50 ~ "Early Empire (to 50 CE)",
      Date_start <= 150 ~ "High Empire (50-150 CE)",
      Date_start <= 250 ~ "Later Empire (150-250 CE)",
      Date_start > 250 ~ "Late Empire (after 250 CE)",
      TRUE ~ "Unknown Period"
    ),
    century = case_when(
      Date_start <= 100 ~ "1st Century",
      Date_start <= 200 ~ "2nd Century",
      Date_start <= 300 ~ "3rd Century",
      Date_start > 300 ~ "4th+ Century",
      TRUE ~ "Unknown"
    )
  )

cat("Clean dataset for network analysis:", nrow(inscriptions_clean), "inscriptions
\n")
```

```
## Clean dataset for network analysis: 53 inscriptions
```

```
cat("Cities represented:", length(unique(inscriptions_clean$Findspot_city)), "\n")
```

```
## Cities represented: 4
```

```
cat("Percentage of total used:", round(nrow(inscriptions_clean)/nrow(inscriptions)
*100, 1), "%\n\n")
```

```
## Percentage of total used: 75.7 %

# Step 3: Enhanced network edge creation with comprehensive attributes
create_enhanced_edges <- function(data) {
  edges <- data.frame()

  for (i in 1:nrow(data)) {
    city <- data$Findspot_city[i]
    emperors_string <- data$Emperor_mentioned[i]
    date_start <- data$Date_start[i]
    period <- data$period[i]
    century <- data$century[i]
    inscription_type <- data$Inscription_type[i]
    inscription_id <- data$inscription_id[i]

    # Parse multiple emperors more robustly
    emperors_list <- str_split(emperors_string, "[;,]")[[1]]  # Allow both ; and ,
separators
    emperors_list <- str_trim(emperors_list)
    emperors_list <- emperors_list[emperors_list != "" & emperors_list != "None"]

    # Create edges with comprehensive attributes
    for (emperor in emperors_list) {
      new_edge <- data.frame(
        from = city,
        to = emperor,
        type = "city_emperor",
        date_start = date_start,
        period = period,
        century = century,
        inscription_type = inscription_type,
        inscription_id = inscription_id,
        stringsAsFactors = FALSE
      )
      edges <- rbind(edges, new_edge)
    }
  }

  # Calculate enhanced connection weights and statistics
  edges_weighted <- edges %>%
    group_by(from, to) %>%
    summarise(
      weight = n(),
      earliest_date = min(date_start, na.rm = TRUE),
      latest_date = max(date_start, na.rm = TRUE),
      date_span = ifelse(is.infinite(latest_date - earliest_date), 0, latest_date
- earliest_date),
```

```
      periods = paste(unique(period[!is.na(period)]), collapse = "; "),
      centuries = paste(unique(century[!is.na(century)]), collapse = "; "),
      inscription_types = paste(unique(inscription_type[!is.na(inscription_type)])
, collapse = "; "),
      inscription_ids = paste(unique(inscription_id[!is.na(inscription_id)]), coll
apse = "; "),
      .groups = 'drop'
    )

  return(edges_weighted)
}

edge_list <- create_enhanced_edges(inscriptions_clean)

cat("=== NETWORK EDGE ANALYSIS ===\n")

## === NETWORK EDGE ANALYSIS ===

cat("Total city-emperor connections:", nrow(edge_list), "\n")

## Total city-emperor connections: 41

cat("Connections with multiple mentions:", sum(edge_list$weight > 1), "\n")

## Connections with multiple mentions: 19

cat("Maximum connection weight:", max(edge_list$weight), "\n")

## Maximum connection weight: 6

cat("Average connection weight:", round(mean(edge_list$weight), 2), "\n\n")

## Average connection weight: 1.88

# Step 4: Comprehensive node attribute calculation
cities <- unique(edge_list$from)
emperors <- unique(edge_list$to)

# Enhanced city statistics
city_stats <- edge_list %>%
  group_by(from) %>%
  summarise(
    connections = n(),
    unique_emperors = n_distinct(to),
    total_weight = sum(weight),
    avg_weight = round(mean(weight), 2),
    earliest_connection = min(earliest_date, na.rm = TRUE),
    latest_connection = max(latest_date, na.rm = TRUE),
    time_span = ifelse(is.infinite(latest_connection - earliest_connection), 0, la
```

```
test_connection - earliest_connection),
    periods_active = n_distinct(periods[periods != ""]),
    .groups = 'drop'
  ) %>%
  mutate(
    connection_intensity = total_weight / connections,  # Average mentions per uni
que emperor
    temporal_consistency = ifelse(time_span > 0, connections / (time_span/100 + 1)
, connections)  # Connections per century
  )

# Enhanced emperor statistics
emperor_stats <- edge_list %>%
  group_by(to) %>%
  summarise(
    connections = n(),
    unique_cities = n_distinct(from),
    total_weight = sum(weight),
    avg_weight = round(mean(weight), 2),
    earliest_mention = min(earliest_date, na.rm = TRUE),
    latest_mention = max(latest_date, na.rm = TRUE),
    geographic_spread = n_distinct(from) / length(cities),  # Proportion of cities
mentioning this emperor
    .groups = 'drop'
  )

# Create comprehensive nodes data frame with enhanced attributes
nodes <- data.frame(
  name = c(cities, emperors),
  type = c(rep("City", length(cities)), rep("Emperor", length(emperors))),
  stringsAsFactors = FALSE
)

# Add enhanced statistics to nodes
nodes$connections <- c(
  city_stats$connections[match(cities, city_stats$from)],
  emperor_stats$connections[match(emperors, emperor_stats$to)]
)

nodes$total_weight <- c(
  city_stats$total_weight[match(cities, city_stats$from)],
  emperor_stats$total_weight[match(emperors, emperor_stats$to)]
)

nodes$importance_score <- c(
  city_stats$connection_intensity[match(cities, city_stats$from)],
  emperor_stats$geographic_spread[match(emperors, emperor_stats$to)]
```

```r
)

# Step 5: Network creation and advanced analysis
network <- graph_from_data_frame(d = edge_list, vertices = nodes, directed = FALSE
)

# Calculate advanced network metrics
centrality_metrics <- data.frame(
  name = V(network)$name,
  degree = degree(network),
  betweenness = betweenness(network, normalized = TRUE),
  closeness = closeness(network, normalized = TRUE),
  eigenvector = eigen_centrality(network)$vector
)

# Add centrality scores to nodes
nodes <- nodes %>%
  left_join(centrality_metrics, by = "name")

cat("=== ENHANCED NETWORK STATISTICS ===\n")
```

```
## === ENHANCED NETWORK STATISTICS ===
```

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

```
## Total nodes: 33
```

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

```
## Total edges: 41
```

```r
cat("Network density:", round(edge_density(network), 4), "\n")
```

```
## Network density: 0.0777
```

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

```
## Average degree: 2.48
```

```r
cat("Network diameter:", diameter(network), "\n")
```

```
## Network diameter: 12
```

```r
cat("Average path length:", round(mean_distance(network), 2), "\n")
```

```
## Average path length: 4.85
```

```r
cat("Global clustering coefficient:", round(transitivity(network), 3), "\n")
```

```
## Global clustering coefficient: 0
```

```r
cat("Network centralization (degree):", round(centr_degree(network)$centralization
, 3), "\n\n")
```

```
## Network centralization (degree): 0.329
```

```r
# Enhanced city and emperor rankings
cat("=== TOP CITIES BY DIFFERENT METRICS ===\n")
```

```
## === TOP CITIES BY DIFFERENT METRICS ===
```

```r
cat("By total connections:\n")
```

```
## By total connections:
```

```r
print(city_stats %>% arrange(desc(connections)) %>% select(from, connections, uniq
ue_emperors) %>% head(4))
```

```
## # A tibble: 4 × 3
##   from        connections unique_emperors
##   <chr>             <int>           <int>
## 1 Boubon               13              13
## 2 Ephesus              11              11
## 3 Aphrodisias          10              10
## 4 Pergamon              7               7
```

```r
cat("\nBy connection intensity (avg mentions per emperor):\n")
```

```
##
## By connection intensity (avg mentions per emperor):
```

```r
print(city_stats %>% arrange(desc(connection_intensity)) %>% select(from, connecti
on_intensity, total_weight) %>% head(4))
```

```
## # A tibble: 4 × 3
##   from        connection_intensity total_weight
##   <chr>                      <dbl>        <int>
## 1 Pergamon                    2.57           18
## 2 Aphrodisias                 1.9            19
## 3 Ephesus                     1.82           20
## 4 Boubon                      1.54           20
```

```r
cat("\nBy temporal span:\n")
```

```
##
## By temporal span:
```

```r
print(city_stats %>% arrange(desc(time_span)) %>% select(from, time_span, earliest
_connection, latest_connection) %>% head(4))
```

```
## # A tibble: 4 × 4
##   from        time_span earliest_connection latest_connection
```

```
##   <chr>           <dbl>               <dbl>             <dbl>
## 1 Ephesus           388                 -48               340
## 2 Aphrodisias       237                  20               257
## 3 Boubon            200                  54               254
## 4 Pergamon          123                  14               137
```

```
cat("\n=== TOP EMPERORS BY DIFFERENT METRICS ===\n")
```

```
##
## === TOP EMPERORS BY DIFFERENT METRICS ===
```

```
cat("By total mentions:\n")
```

```
## By total mentions:
```

```
print(emperor_stats %>% arrange(desc(total_weight)) %>% select(to, total_weight, u
nique_cities) %>% head(8))
```

```
## # A tibble: 8 × 3
##   to        total_weight unique_cities
##   <chr>            <int>         <int>
## 1 Hadrian             10             4
## 2 Augustus             9             3
## 3 Tiberius             7             2
## 4 Trajan               6             3
## 5 Domitian             5             1
## 6 Gallienus            4             2
## 7 Nero                 4             2
## 8 Commodus             3             2
```

```
cat("\nBy geographic spread:\n")
```

```
##
## By geographic spread:
```

```
print(emperor_stats %>% arrange(desc(geographic_spread)) %>% select(to, geographic
_spread, unique_cities) %>% head(8))
```

```
## # A tibble: 8 × 3
##   to        geographic_spread unique_cities
##   <chr>                 <dbl>         <int>
## 1 Hadrian                1                4
## 2 Augustus               0.75             3
## 3 Trajan                 0.75             3
## 4 Commodus               0.5              2
## 5 Gallienus              0.5              2
## 6 Nero                   0.5              2
## 7 Tiberius               0.5              2
## 8 Valerian               0.5              2
```

```r
# Step 6: Create focused city-specific visualizations

# First, create deity network edges
create_deity_edges <- function(data) {
  edges <- data.frame()

  for (i in 1:nrow(data)) {
    city <- data$Findspot_city[i]
    deities_string <- data$Local_deities[i]

    if (!is.na(deities_string) && deities_string != "None" && deities_string != "#
NA") {
      deities_list <- str_split(deities_string, "[;,]")[[1]]
      deities_list <- str_trim(deities_list)
      deities_list <- deities_list[deities_list != "" & deities_list != "None"]

      for (deity in deities_list) {
        new_edge <- data.frame(
          from = city,
          to = deity,
          type = "city_deity",
          stringsAsFactors = FALSE
        )
        edges <- rbind(edges, new_edge)
      }
    }
  }

  # Calculate connection weights
  edges_weighted <- edges %>%
    group_by(from, to) %>%
    summarise(weight = n(), .groups = 'drop')

  return(edges_weighted)
}

deity_edges <- create_deity_edges(inscriptions_clean)

# Function to create individual city network plots
create_city_network_plot <- function(city_name, connections, connection_type, colo
r_scheme) {
  if (nrow(connections) == 0) {
    return(NULL)
  }

  # Create nodes for this city's network
  city_nodes <- data.frame(
```

```r
    name = c(city_name, unique(connections$to)),
    type = c("City", rep(connection_type, length(unique(connections$to)))),
    stringsAsFactors = FALSE
  )

  # Add weights to nodes
  connection_weights <- connections %>%
    group_by(to) %>%
    summarise(total_weight = sum(weight), .groups = 'drop')

  city_nodes$weight <- c(
    sum(connections$weight), # City gets total of all connections
    connection_weights$total_weight[match(city_nodes$name[-1], connection_weights$
to)]
  )

  # Create the network
  city_network <- graph_from_data_frame(d = connections, vertices = city_nodes, di
rected = FALSE)

  # Create the plot
  set.seed(42)
  p <- city_network %>%
    as_tbl_graph() %>%
    ggraph(layout = "star") + # Star layout puts city in center

    geom_edge_link(
      aes(width = weight),
      color = "gray50",
      alpha = 0.7
    ) +

    geom_node_point(
      aes(color = type, size = weight),
      alpha = 0.9
    ) +

    geom_node_text(
      aes(label = name),
      size = 3,
      repel = TRUE,
      point.padding = unit(0.3, "lines"),
      box.padding = unit(0.3, "lines"),
      force = 2,
      fontface = "bold"
    ) +
```

```
    scale_color_manual(values = color_scheme, name = "Node Type") +
    scale_size_continuous(range = c(4, 12), guide = "none") +
    scale_edge_width_continuous(range = c(0.5, 2.5), guide = "none") +

    labs(
      title = paste(city_name, "-", str_to_title(connection_type), "Network"),
      subtitle = paste("Connections:", nrow(connections), "|", "Unique", paste0(st
r_to_lower(connection_type), "s:"), n_distinct(connections$to))
    ) +

    theme_graph() +
    theme(
      plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
      plot.subtitle = element_text(hjust = 0.5, size = 10),
      legend.position = "none",
      plot.margin = margin(15, 15, 15, 15)
    )

  return(p)
}

# Color schemes
emperor_colors <- c("City" = "#E53E3E", "Emperor" = "#3182CE")
deity_colors <- c("City" = "#E53E3E", "Deity" = "#8E4EC6")

cat("\n=== CREATING INDIVIDUAL CITY NETWORKS ===\n")

##
## === CREATING INDIVIDUAL CITY NETWORKS ===

# Create emperor network plots for each city
emperor_plots <- list()
deity_plots <- list()

cities <- unique(edge_list$from)

for (city in cities) {
  # Emperor connections
  city_emperor_connections <- edge_list %>% filter(from == city)
  if (nrow(city_emperor_connections) > 0) {
    emperor_plots[[city]] <- create_city_network_plot(city, city_emperor_connectio
ns, "Emperor", emperor_colors)
    cat("Created emperor network for", city, ":", nrow(city_emperor_connections),
"connections\n")
  }

  # Deity connections
```

```r
  city_deity_connections <- deity_edges %>% filter(from == city)
  if (nrow(city_deity_connections) > 0) {
    deity_plots[[city]] <- create_city_network_plot(city, city_deity_connections,
"Deity", deity_colors)
    cat("Created deity network for", city, ":", nrow(city_deity_connections), "con
nections\n")
  }
}
```

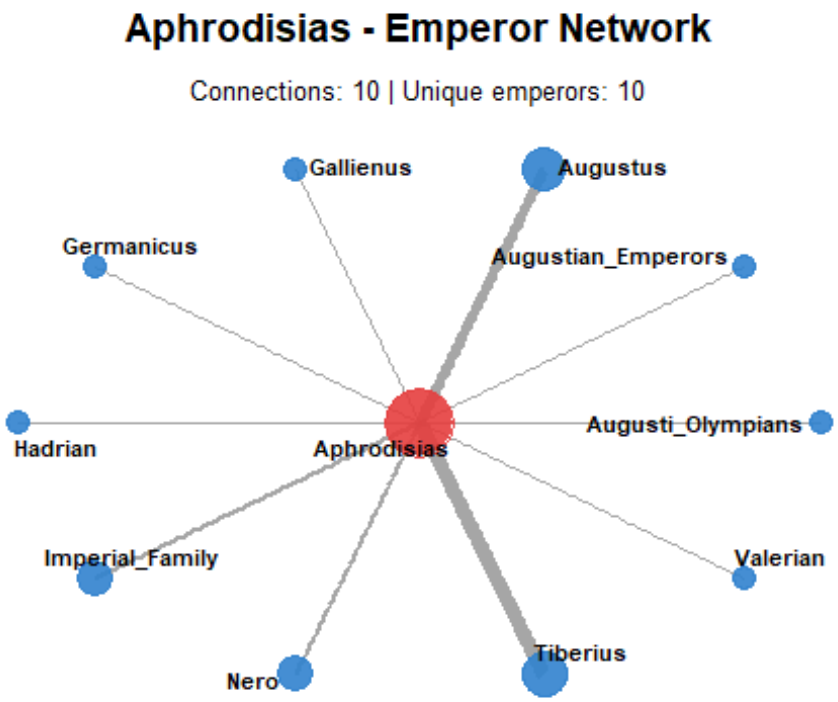
```
## Created emperor network for Aphrodisias : 10 connections
## Created deity network for Aphrodisias : 6 connections
## Created emperor network for Boubon : 13 connections
## Created emperor network for Ephesus : 11 connections
## Created deity network for Ephesus : 5 connections
## Created emperor network for Pergamon : 7 connections
## Created deity network for Pergamon : 5 connections
```

```r
# Display all emperor network plots
cat("\n=== CITY-EMPEROR NETWORKS ===\n")
```

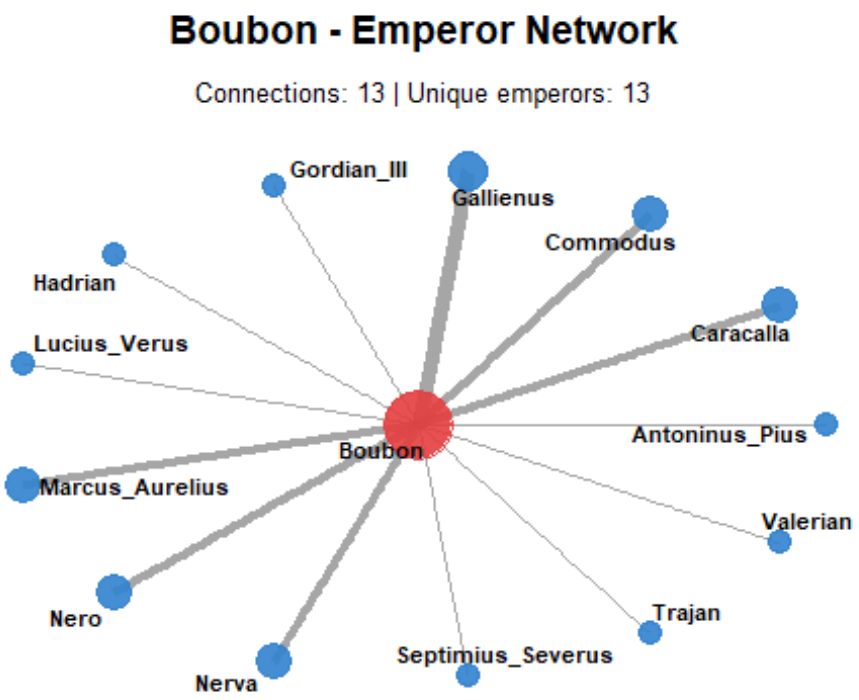
```
##
## === CITY-EMPEROR NETWORKS ===
```

```r
for (city in names(emperor_plots)) {
  if (!is.null(emperor_plots[[city]])) {
    print(emperor_plots[[city]])
  }
}
```

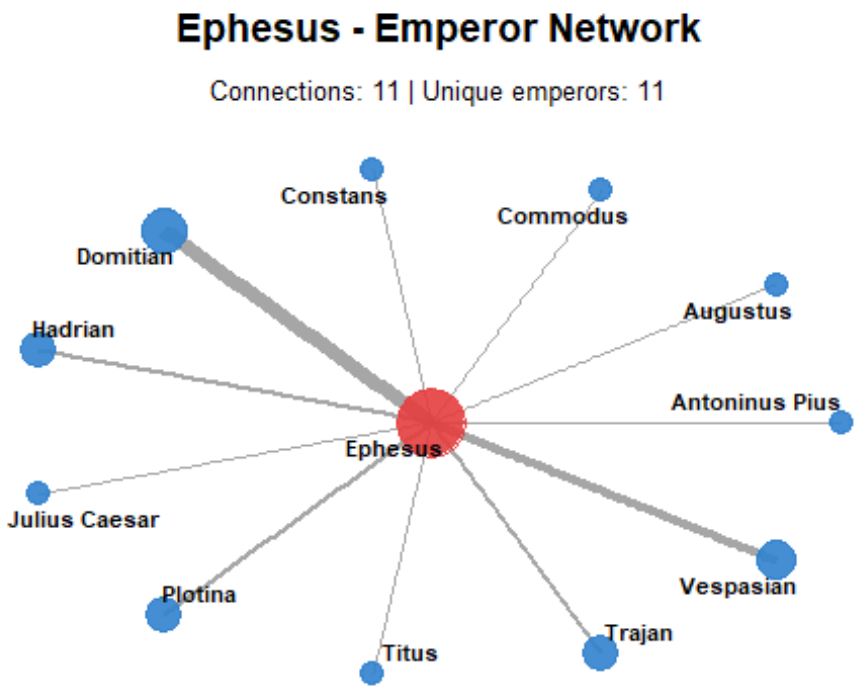
```
## Warning in grid.Call(C_stringMetric, as.graphicsAnnot(x$label)): font family
## not found in Windows font database
## Warning in grid.Call(C_stringMetric, as.graphicsAnnot(x$label)): font family
## not found in Windows font database

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font family not found in Windows font database
## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font family not found in Windows font database
```



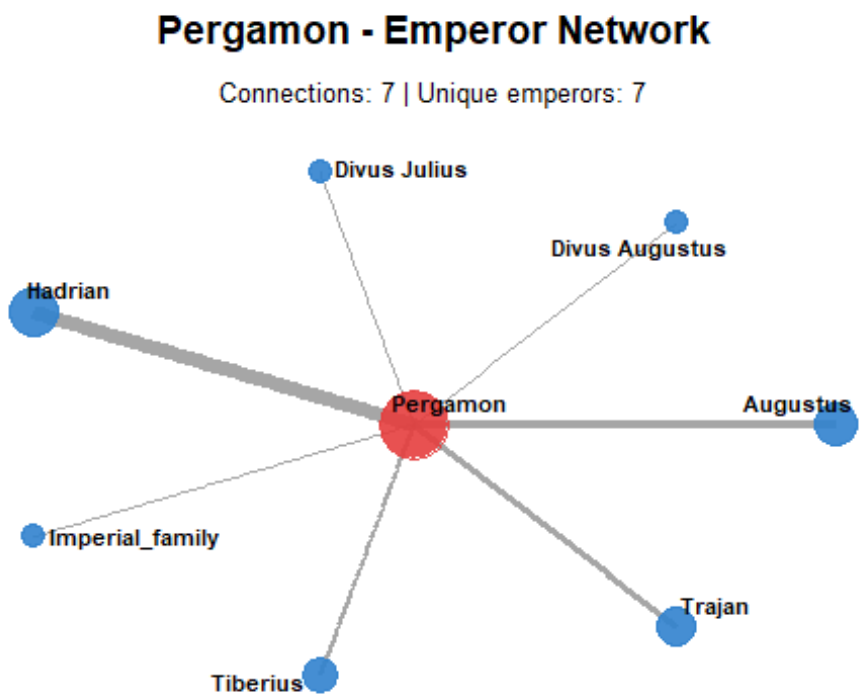

```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
```



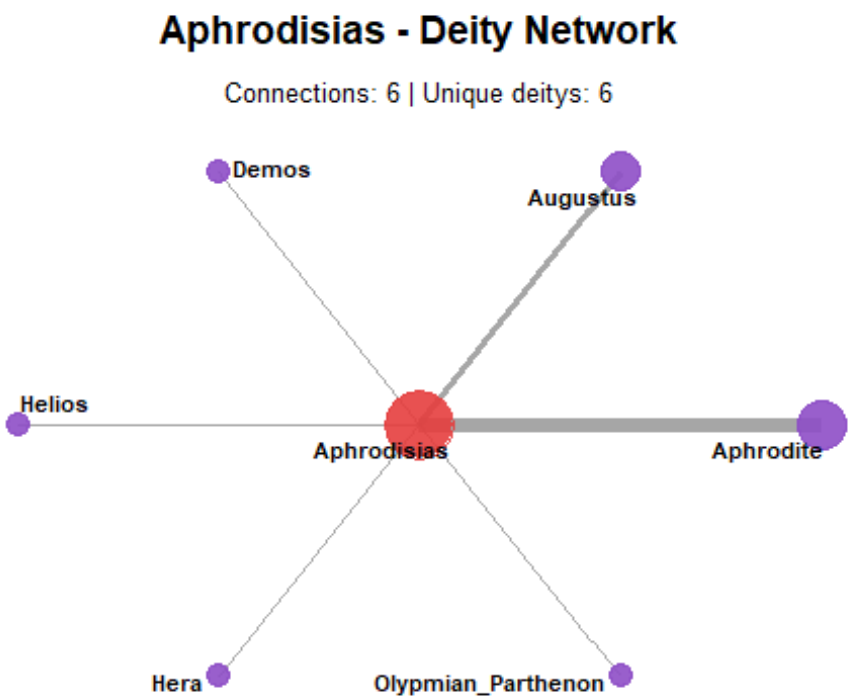

```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
```



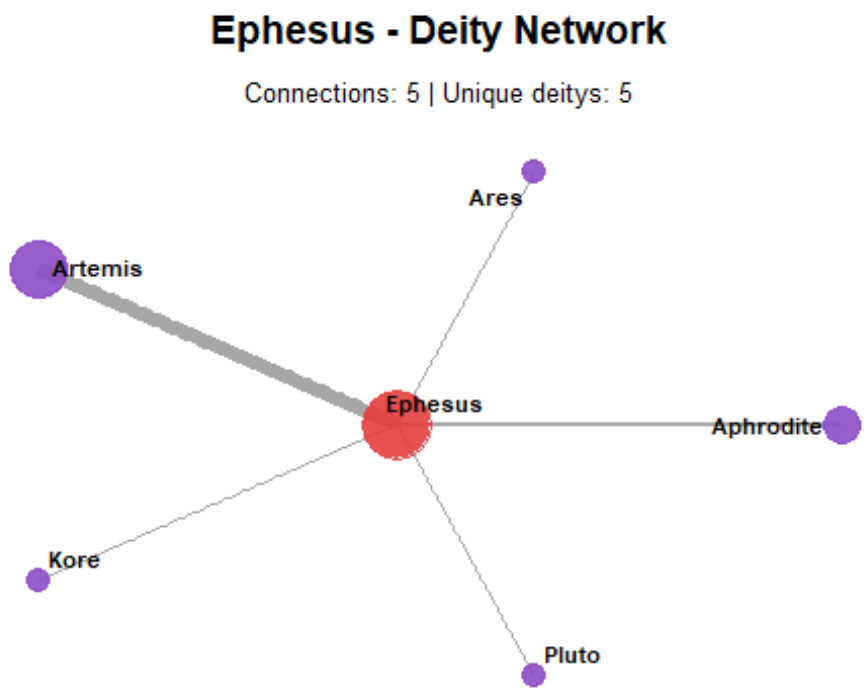

```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
```



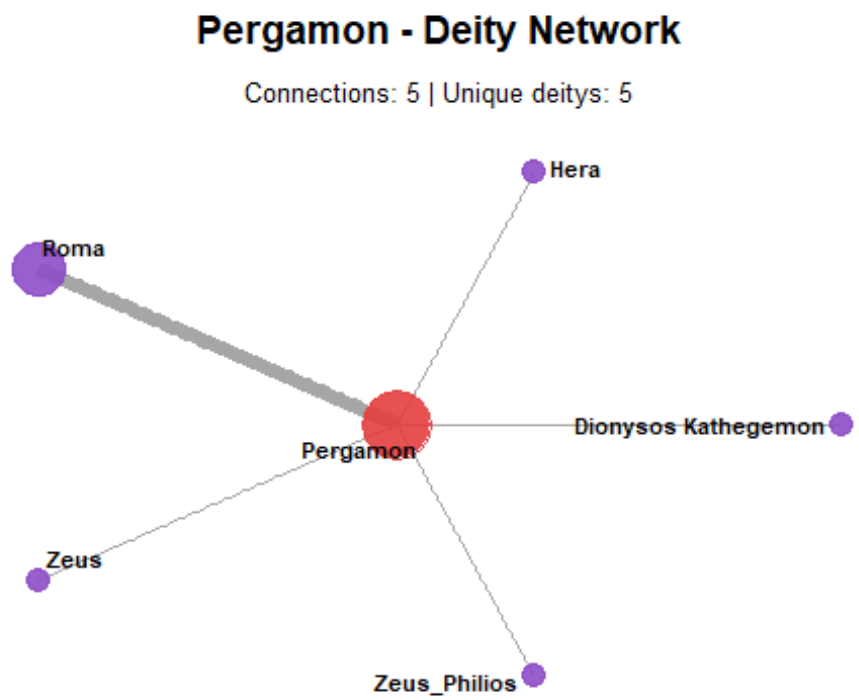

```
# Display all deity network plots
cat("\n=== CITY-DEITY NETWORKS ===\n")

##
## === CITY-DEITY NETWORKS ===

for (city in names(deity_plots)) {
  if (!is.null(deity_plots[[city]])) {
    print(deity_plots[[city]])
  }
}

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
```




```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
```




```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
```




```
# Step 7: Create comparative map showing shared emperor and deity connections
cat("\n=== ANALYZING SHARED CONNECTIONS ===\n")
```

```
##
## === ANALYZING SHARED CONNECTIONS ===
```

```
# Find shared emperors (emperors mentioned by multiple cities)
shared_emperors <- edge_list %>%
  group_by(to) %>%
  summarise(cities = n_distinct(from), city_list = paste(unique(from), collapse =
", "), .groups = 'drop') %>%
  filter(cities > 1) %>%
  arrange(desc(cities))

cat("Shared Emperors (mentioned by multiple cities):\n")
```

```
## Shared Emperors (mentioned by multiple cities):
```

```
print(shared_emperors)
```

```
## # A tibble: 8 × 3
##   to        cities city_list
##   <chr>      <int> <chr>
## 1 Hadrian        4 Aphrodisias, Boubon, Ephesus, Pergamon
## 2 Augustus       3 Aphrodisias, Ephesus, Pergamon
```

```
## 3 Trajan          3 Boubon, Ephesus, Pergamon
## 4 Commodus        2 Boubon, Ephesus
## 5 Gallienus       2 Aphrodisias, Boubon
## 6 Nero            2 Aphrodisias, Boubon
## 7 Tiberius        2 Aphrodisias, Pergamon
## 8 Valerian        2 Aphrodisias, Boubon
```

```
# Find shared deities (deities mentioned by multiple cities)
shared_deities <- deity_edges %>%
  group_by(to) %>%
  summarise(cities = n_distinct(from), city_list = paste(unique(from), collapse =
", "), .groups = 'drop') %>%
  filter(cities > 1) %>%
  arrange(desc(cities))

cat("\nShared Deities (mentioned by multiple cities):\n")
```

```
##
## Shared Deities (mentioned by multiple cities):
```

```
print(shared_deities)
```

```
## # A tibble: 2 × 3
##   to        cities city_list
##   <chr>      <int> <chr>
## 1 Aphrodite      2 Aphrodisias, Ephesus
## 2 Hera           2 Aphrodisias, Pergamon
```

```
# Find cities that have BOTH shared emperor and shared deity connections
cities_with_both <- intersect(
  unique(edge_list$from[edge_list$to %in% shared_emperors$to]),
  unique(deity_edges$from[deity_edges$to %in% shared_deities$to])
)

cat("\nCities with both shared emperor AND shared deity connections:\n")
```

```
##
## Cities with both shared emperor AND shared deity connections:
```

```
cat(paste(cities_with_both, collapse = ", "), "\n")
```

```
## Aphrodisias, Ephesus, Pergamon
```

```
# Create the final comparative network
if (length(cities_with_both) > 1 && nrow(shared_emperors) > 0 && nrow(shared_deiti
es) > 0) {

  # Get shared connections for these cities
  shared_emperor_edges <- edge_list %>%
```

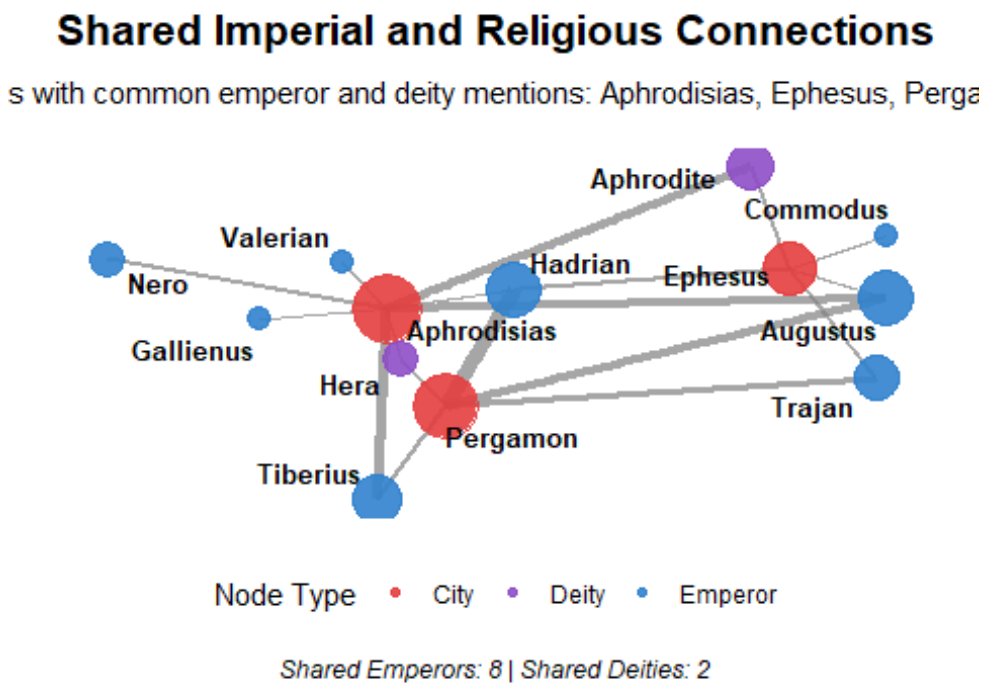
```r
    filter(from %in% cities_with_both, to %in% shared_emperors$to)

  shared_deity_edges <- deity_edges %>%
    filter(from %in% cities_with_both, to %in% shared_deities$to) %>%
    mutate(type = "city_deity")

  # Combine edges
  comparative_edges <- bind_rows(
    shared_emperor_edges %>% mutate(connection_type = "Emperor"),
    shared_deity_edges %>% mutate(connection_type = "Deity")
  )

  # Create nodes for comparative network
  comparative_nodes <- data.frame(
    name = c(
      cities_with_both,
      shared_emperors$to,
      shared_deities$to
    ),
    type = c(
      rep("City", length(cities_with_both)),
      rep("Emperor", length(shared_emperors$to)),
      rep("Deity", length(shared_deities$to))
    ),
    stringsAsFactors = FALSE
  )

  # Remove duplicates if any
  comparative_nodes <- comparative_nodes[!duplicated(comparative_nodes$name), ]

  # Add weights
  connection_weights <- comparative_edges %>%
    group_by(to) %>%
    summarise(total_weight = sum(weight), .groups = 'drop')

  city_weights <- comparative_edges %>%
    group_by(from) %>%
    summarise(total_weight = sum(weight), .groups = 'drop')

  comparative_nodes$weight <- ifelse(
    comparative_nodes$type == "City",
    city_weights$total_weight[match(comparative_nodes$name, city_weights$from)],
    connection_weights$total_weight[match(comparative_nodes$name, connection_weigh
ts$to)]
  )

  # Handle any NA weights
```

```r
comparative_nodes$weight[is.na(comparative_nodes$weight)] <- 1

# Create the comparative network
comparative_network <- graph_from_data_frame(
  d = comparative_edges %>% select(from, to, weight),
  vertices = comparative_nodes,
  directed = FALSE
)

# Create the comparative visualization
set.seed(42)
p_comparative <- comparative_network %>%
  as_tbl_graph() %>%
  ggraph(layout = "stress") +

  geom_edge_link(
    aes(width = weight),
    color = "gray50",
    alpha = 0.7
  ) +

  geom_node_point(
    aes(color = type, size = weight),
    alpha = 0.9
  ) +

  geom_node_text(
    aes(label = name),
    size = 3.5,
    repel = TRUE,
    point.padding = unit(0.4, "lines"),
    box.padding = unit(0.4, "lines"),
    force = 3,
    fontface = "bold"
  ) +

  scale_color_manual(
    values = c("City" = "#E53E3E", "Emperor" = "#3182CE", "Deity" = "#8E4EC6"),
    name = "Node Type"
  ) +

  scale_size_continuous(range = c(4, 12), guide = "none") +
  scale_edge_width_continuous(range = c(0.5, 2.5), guide = "none") +

  labs(
    title = "Shared Imperial and Religious Connections",
    subtitle = paste("Cities with common emperor and deity mentions:",
```

```
                      paste(cities_with_both, collapse = ", ")),
      caption = paste("Shared Emperors:", nrow(shared_emperors),
                      "| Shared Deities:", nrow(shared_deities))
    ) +

    theme_graph() +
    theme(
      plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
      plot.subtitle = element_text(hjust = 0.5, size = 11),
      plot.caption = element_text(hjust = 0.5, size = 9),
      legend.position = "bottom",
      plot.margin = margin(20, 20, 20, 20)
    )

  cat("\n=== SHARED CONNECTIONS NETWORK ===\n")
  print(p_comparative)

} else {
  cat("\nInsufficient shared connections for comparative network visualization.\n"
)
  cat("Need cities with both shared emperors and shared deities.\n")
}
```

```
##
## === SHARED CONNECTIONS NETWORK ===

## Warning in grid.Call(C_stringMetric, as.graphicsAnnot(x$label)): font family
## not found in Windows font database

## Warning in grid.Call(C_stringMetric, as.graphicsAnnot(x$label)): font family
## not found in Windows font database

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database

## Warning in grid.Call(C_stringMetric, as.graphicsAnnot(x$label)): font family
## not found in Windows font database
## Warning in grid.Call(C_stringMetric, as.graphicsAnnot(x$label)): font family
## not found in Windows font database

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : font
## family not found in Windows font database

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font family not found in Windows font database
```

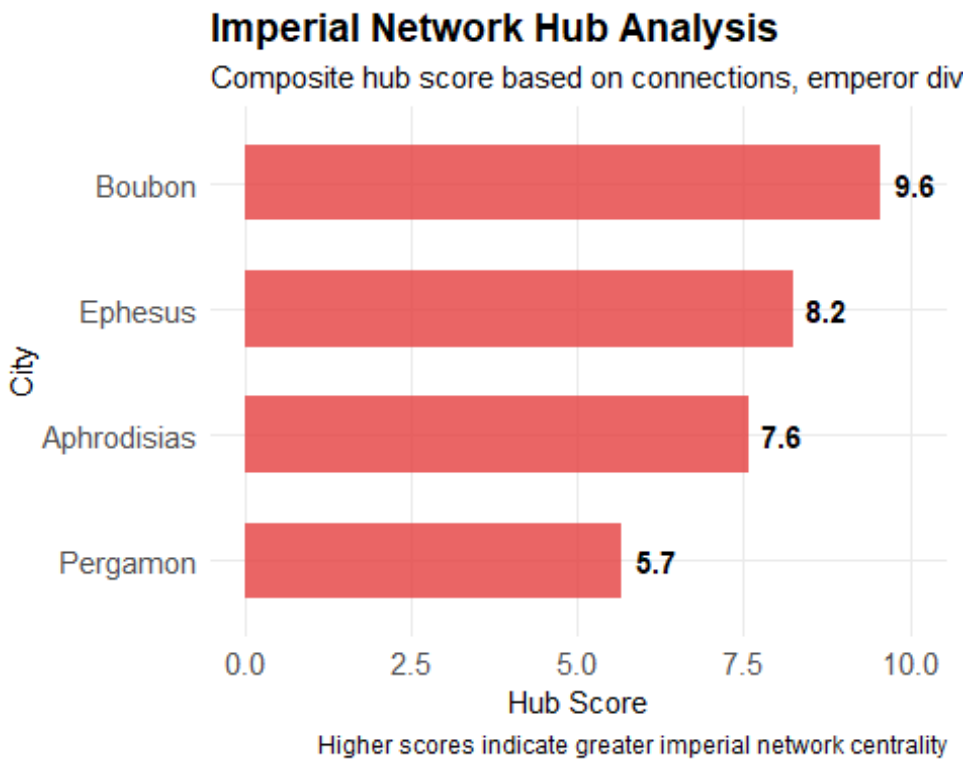
```
## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font family not found in Windows font database
## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font family not found in Windows font database
## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font family not found in Windows font database
## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x, x$y, :
## font family not found in Windows font database
```




```
# Additional analysis of shared patterns
cat("\n=== DETAILED SHARED CONNECTION ANALYSIS ===\n")

##
## === DETAILED SHARED CONNECTION ANALYSIS ===

cat("Most connected shared emperor:", shared_emperors$to[1],
    "(appears in", shared_emperors$cities[1], "cities)\n")

## Most connected shared emperor: Hadrian (appears in 4 cities)

if (nrow(shared_deities) > 0) {
  cat("Most connected shared deity:", shared_deities$to[1],
      "(appears in", shared_deities$cities[1], "cities)\n")
} else {
  cat("No deities shared between multiple cities.\n")
}
```

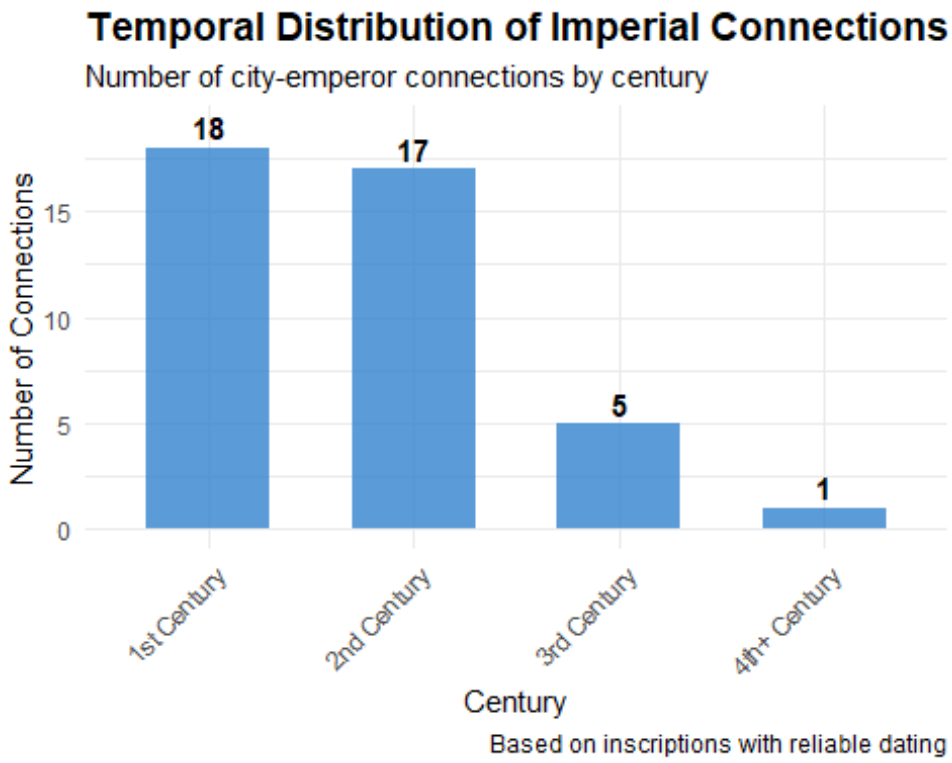
```
## Most connected shared deity: Aphrodite (appears in 2 cities)

# Step 7: City-focused hub analysis optimized for 4-city network
cat("\n=== HUB CITY ANALYSIS (ALL 4 CITIES) ===\n")

##
## === HUB CITY ANALYSIS (ALL 4 CITIES) ===

# Since we only have 4 cities, analyze all of them as potential hubs
hub_analysis <- city_stats %>%
  mutate(
    hub_score = (connections * 0.4) + (unique_emperors * 0.3) + (connection_intens
ity * 0.3),
    hub_rank = rank(-hub_score)
  ) %>%
  arrange(desc(hub_score))

print(hub_analysis %>% select(from, connections, unique_emperors, connection_inten
sity, hub_score, hub_rank))

## # A tibble: 4 × 6
##   from       connections unique_emperors connection_intensity hub_score hub_ran
k
##   <chr>            <int>           <int>                <dbl>     <dbl>    <dbl
>
## 1 Boubon              13              13                 1.54      9.56
1
## 2 Ephesus             11              11                 1.82      8.25
2
## 3 Aphrodisi…          10              10                 1.9       7.57
3
## 4 Pergamon             7               7                 2.57      5.67
4

# Create hub comparison visualization
p_hub_comparison <- hub_analysis %>%
  ggplot(aes(x = reorder(from, hub_score))) +
  geom_col(aes(y = hub_score), fill = "#E53E3E", alpha = 0.8, width = 0.6) +
  geom_text(aes(y = hub_score + 0.5, label = round(hub_score, 1)),
            color = "black", fontface = "bold", size = 4) +
  coord_flip() +
  labs(
    title = "Imperial Network Hub Analysis",
    subtitle = "Composite hub score based on connections, emperor diversity, and i
ntensity",
    x = "City",
    y = "Hub Score",
    caption = "Higher scores indicate greater imperial network centrality"
  ) +
```

```
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14),
    plot.subtitle = element_text(size = 11),
    axis.text = element_text(size = 11),
    panel.grid.minor = element_blank()
  )

print(p_hub_comparison)
```




```
# Step 8: Enhanced temporal analysis
temporal_network_analysis <- edge_list %>%
  filter(!is.na(earliest_date)) %>%
  mutate(
    century = case_when(
      earliest_date <= 100 ~ "1st Century",
      earliest_date <= 200 ~ "2nd Century",
      earliest_date <= 300 ~ "3rd Century",
      earliest_date > 300 ~ "4th+ Century",
      TRUE ~ "Unknown"
    )
  ) %>%
  group_by(century) %>%
  summarise(
    connections = n(),
```

```
    unique_cities = n_distinct(from),
    unique_emperors = n_distinct(to),
    total_weight = sum(weight),
    avg_weight = round(mean(weight), 2),
    .groups = 'drop'
  )

cat("\n=== TEMPORAL EVOLUTION OF IMPERIAL CONNECTIONS ===\n")

##
## === TEMPORAL EVOLUTION OF IMPERIAL CONNECTIONS ===

print(temporal_network_analysis)

## # A tibble: 4 × 6
##   century      connections unique_cities unique_emperors total_weight avg_weigh
t
##   <chr>              <int>         <int>           <int>        <int>      <dbl
>
## 1 1st Century           18             4              14           39       2.1
7
## 2 2nd Century           17             4              11           30       1.7
6
## 3 3rd Century            5             2               3            7       1.4
## 4 4th+ Century           1             1               1            1       1

# Temporal visualization
p_temporal <- temporal_network_analysis %>%
  filter(century != "Unknown") %>%
  ggplot(aes(x = century)) +
  geom_col(aes(y = connections), fill = "#3182CE", alpha = 0.8, width = 0.6) +
  geom_text(aes(y = connections + 1, label = connections),
            color = "black", fontface = "bold", size = 4) +
  labs(
    title = "Temporal Distribution of Imperial Connections",
    subtitle = "Number of city-emperor connections by century",
    x = "Century",
    y = "Number of Connections",
    caption = "Based on inscriptions with reliable dating"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", size = 14),
    plot.subtitle = element_text(size = 11),
    axis.text.x = element_text(angle = 45, hjust = 1)
  )

print(p_temporal)
```




```
# Step 9: Network resilience and structure analysis
cat("\n=== NETWORK STRUCTURE ANALYSIS ===\n")

##
## === NETWORK STRUCTURE ANALYSIS ===

# Analyze what happens if we remove the most connected city
most_connected_city <- city_stats$from[which.max(city_stats$connections)]
network_without_hub <- delete_vertices(network, most_connected_city)

cat("Original network components:", components(network)$no, "\n")

## Original network components: 1

cat("Network without", most_connected_city, "components:", components(network_with
out_hub)$no, "\n")

## Network without Boubon components: 8

cat("Original network efficiency:", round(global_efficiency(network), 3), "\n")

## Original network efficiency: 0.275

cat("Network efficiency without", most_connected_city, ":", round(global_efficienc
y(network_without_hub), 3), "\n")
```

```
## Network efficiency without Boubon : 0.163

# Analyze emperor removal impact
most_mentioned_emperor <- emperor_stats$to[which.max(emperor_stats$total_weight)]
network_without_top_emperor <- delete_vertices(network, most_mentioned_emperor)

cat("Network efficiency without", most_mentioned_emperor, ":", round(global_effici
ency(network_without_top_emperor), 3), "\n")

## Network efficiency without Hadrian : 0.262

# Step 10: Save all visualizations
cat("\n=== SAVING VISUALIZATIONS ===\n")

##
## === SAVING VISUALIZATIONS ===

# Save individual city-emperor networks
for (city in names(emperor_plots)) {
  if (!is.null(emperor_plots[[city]])) {
    filename <- paste0("emperor_network_", tolower(gsub(" ", "_", city)), "_2024.p
ng")
    ggsave(filename, plot = emperor_plots[[city]],
           width = 10, height = 8, dpi = 300, bg = "white")
    cat("Saved:", filename, "\n")
  }
}

## Saved: emperor_network_aphrodisias_2024.png

## Saved: emperor_network_boubon_2024.png

## Saved: emperor_network_ephesus_2024.png

## Saved: emperor_network_pergamon_2024.png

# Save individual city-deity networks
for (city in names(deity_plots)) {
  if (!is.null(deity_plots[[city]])) {
    filename <- paste0("deity_network_", tolower(gsub(" ", "_", city)), "_2024.png
")
    ggsave(filename, plot = deity_plots[[city]],
           width = 10, height = 8, dpi = 300, bg = "white")
    cat("Saved:", filename, "\n")
  }
}

## Saved: deity_network_aphrodisias_2024.png

## Saved: deity_network_ephesus_2024.png
```

```
## Saved: deity_network_pergamon_2024.png

# Save comparative network if it exists
if (exists("p_comparative")) {
  ggsave("shared_connections_network_2024.png", plot = p_comparative,
         width = 12, height = 10, dpi = 300, bg = "white")
  cat("Saved: shared_connections_network_2024.png\n")
}

## Saved: shared_connections_network_2024.png

# Save hub comparison chart
ggsave("imperial_hub_analysis_2024.png", plot = p_hub_comparison,
       width = 10, height = 6, dpi = 300, bg = "white")

# Save temporal analysis
ggsave("imperial_temporal_analysis_2024.png", plot = p_temporal,
       width = 10, height = 6, dpi = 300, bg = "white")

cat("All visualizations saved successfully!\n")

## All visualizations saved successfully!

# Step 11: Comprehensive city profiles with enhanced metrics
cat("\n=== COMPREHENSIVE CITY PROFILES ===\n")

##
## === COMPREHENSIVE CITY PROFILES ===

for (i in 1:nrow(city_stats)) {
  city <- city_stats$from[i]
  stats <- city_stats[i, ]
  city_connections <- edge_list %>% filter(from == city)
  city_centrality <- centrality_metrics %>% filter(name == city)

  cat("\n", toupper(city), " - Imperial Network Profile:\n")
  cat("Rank by hub score:", hub_analysis$hub_rank[hub_analysis$from == city], "of
4\n")
  cat("Total connections:", stats$connections, "\n")
  cat("Unique emperors:", stats$unique_emperors, "\n")
  cat("Total connection weight:", stats$total_weight, "\n")
  cat("Connection intensity:", round(stats$connection_intensity, 2), "\n")
  cat("Temporal span:", stats$time_span, "years (", stats$earliest_connection, "-"
, stats$latest_connection, "CE)\n")
  cat("Degree centrality:", round(city_centrality$degree, 2), "\n")
  cat("Betweenness centrality:", round(city_centrality$betweenness, 3), "\n")

  # Top emperors for this city
  top_emperors <- city_connections %>%
```

```
    arrange(desc(weight)) %>%
    head(5) %>%
    mutate(emperor_weight = paste0(to, " (", weight, ")"))

  cat("Top emperors:", paste(top_emperors$emperor_weight, collapse = ", "), "\n")
  cat(paste(rep("-", 80), collapse = ""), "\n")
}

##
##  APHRODISIAS  - Imperial Network Profile:
## Rank by hub score: 3 of 4
## Total connections: 10
## Unique emperors: 10
## Total connection weight: 19
## Connection intensity: 1.9
## Temporal span: 237 years ( 20 - 257 CE)
## Degree centrality: 10
## Betweenness centrality: 0.302
## Top emperors: Tiberius (5), Augustus (4), Imperial_Family (2), Nero (2), August
i_Olympians (1)
## ------------------------------------------------------------------------------
-
##
##  BOUBON  - Imperial Network Profile:
## Rank by hub score: 1 of 4
## Total connections: 13
## Unique emperors: 13
## Total connection weight: 20
## Connection intensity: 1.54
## Temporal span: 200 years ( 54 - 254 CE)
## Degree centrality: 13
## Betweenness centrality: 0.533
## Top emperors: Gallienus (3), Caracalla (2), Commodus (2), Marcus_Aurelius (2),
Nero (2)
## ------------------------------------------------------------------------------
-
##
##  EPHESUS  - Imperial Network Profile:
## Rank by hub score: 2 of 4
## Total connections: 11
## Unique emperors: 11
## Total connection weight: 20
## Connection intensity: 1.82
## Temporal span: 388 years ( -48 - 340 CE)
## Degree centrality: 11
## Betweenness centrality: 0.439
## Top emperors: Domitian (5), Vespasian (3), Hadrian (2), Plotina (2), Trajan (2)
## ------------------------------------------------------------------------------
```

```
-
##
##  PERGAMON  - Imperial Network Profile:
## Rank by hub score: 4 of 4
## Total connections: 7
## Unique emperors: 7
## Total connection weight: 18
## Connection intensity: 2.57
## Temporal span: 123 years ( 14 - 137 CE)
## Degree centrality: 7
## Betweenness centrality: 0.22
## Top emperors: Hadrian (6), Augustus (4), Trajan (3), Tiberius (2), Divus August
us (1)
## ------------------------------------------------------------------------
-

cat("\n=== ENHANCED ANALYSIS COMPLETE ===\n")

##
## === ENHANCED ANALYSIS COMPLETE ===

cat("Individual City Network Analysis:\n")

## Individual City Network Analysis:

cat("- Created separate emperor networks for", length(emperor_plots), "cities\n")

## - Created separate emperor networks for 4 cities

cat("- Created separate deity networks for", length(deity_plots), "cities\n")

## - Created separate deity networks for 3 cities

cat("- Most central city:", most_connected_city, "(", max(city_stats$connections),
"emperor connections)\n")

## - Most central city: Boubon ( 13 emperor connections)

if (exists("shared_emperors") && nrow(shared_emperors) > 0) {
  cat("- Shared emperors across cities:", nrow(shared_emperors), "\n")
  cat("- Most widespread emperor:", shared_emperors$to[1],
      "(", shared_emperors$cities[1], "cities)\n")
}

## - Shared emperors across cities: 8
## - Most widespread emperor: Hadrian ( 4 cities)

if (exists("shared_deities") && nrow(shared_deities) > 0) {
  cat("- Shared deities across cities:", nrow(shared_deities), "\n")
  cat("- Most widespread deity:", shared_deities$to[1],
      "(", shared_deities$cities[1], "cities)\n")
```

```
} else {
  cat("- No deities shared between multiple cities\n")
}

## - Shared deities across cities: 2
## - Most widespread deity: Aphrodite ( 2 cities)

if (exists("cities_with_both") && length(cities_with_both) > 0) {
  cat("- Cities with both shared emperors and deities:", length(cities_with_both),
"\n")
  cat("- These cities:", paste(cities_with_both, collapse = ", "), "\n")
}

## - Cities with both shared emperors and deities: 3
## - These cities: Aphrodisias, Ephesus, Pergamon

cat("\nKey Network Findings:\n")

##
## Key Network Findings:

cat("- Network density:", round(edge_density(network), 4), "\n")

## - Network density: 0.0777

cat("- Total emperor connections:", nrow(edge_list), "\n")

## - Total emperor connections: 41

cat("- Total deity connections:", nrow(deity_edges), "\n")

## - Total deity connections: 16

cat("- Temporal range:", min(edge_list$earliest_date, na.rm = TRUE), "-",
    max(edge_list$latest_date, na.rm = TRUE), "CE\n")

## - Temporal range: -48 - 340 CE

cat("\nVisualization Approach:\n")

##
## Visualization Approach:

cat("- Individual city-emperor networks provide focused analysis\n")

## - Individual city-emperor networks provide focused analysis

cat("- Individual city-deity networks reveal religious patterns\n")

## - Individual city-deity networks reveal religious patterns

cat("- Shared connections network identifies common cultural elements\n")
```

```
## - Shared connections network identifies common cultural elements

cat("- This multi-map approach offers clearer insights than single complex network
\n")

## - This multi-map approach offers clearer insights than single complex network

cat("\nAll individual city networks and comparative analysis saved.\n")

##
## All individual city networks and comparative analysis saved.

cat("This approach reveals both city-specific patterns and shared cultural connect
ions\n")

## This approach reveals both city-specific patterns and shared cultural connectio
ns

cat("across the", length(cities), "cities in the imperial commemorative network.\n
")

## across the 4 cities in the imperial commemorative network.

# Simplified Imperial Network Visualizations
# Creates clean, focused network plots for each city and comparative analysis

# Load required libraries
library(readxl)      # For reading Excel files
library(dplyr)       # For data manipulation
library(igraph)      # For network analysis
library(ggraph)      # For network visualization
library(tidygraph)   # For tidy network data
library(ggplot2)     # For plotting
library(stringr)     # For string manipulation

# Step 1: Load and prepare data
inscriptions <- read_excel("Inscriptions_combined_with_all_Ephesus.xlsx", sheet =
"Sheet1")

# Clean the data
inscriptions_clean <- inscriptions %>%
  filter(
    !is.na(Findspot_city) &
    !is.na(Emperor_mentioned) &
    Findspot_city != "None" &
    Emperor_mentioned != "None" &
    Findspot_city != "#NA" &
    Emperor_mentioned != "#NA"
  )
```

```r
cat("=== SIMPLIFIED NETWORK ANALYSIS ===\n")
```

```
## === SIMPLIFIED NETWORK ANALYSIS ===
```

```r
cat("Total inscriptions for network analysis:", nrow(inscriptions_clean), "\n")
```

```
## Total inscriptions for network analysis: 53
```

```r
cat("Cities:", length(unique(inscriptions_clean$Findspot_city)), "\n")
```

```
## Cities: 4
```

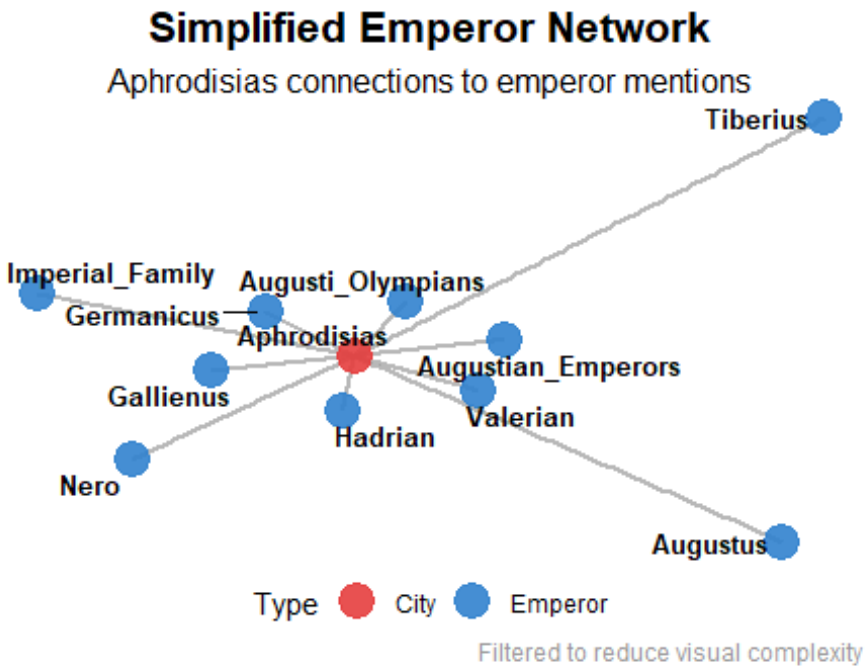
```r
# Step 2: Create emperor network edges
create_emperor_edges <- function(data) {
  edges <- data.frame()

  for (i in 1:nrow(data)) {
    city <- data$Findspot_city[i]
    emperors_string <- data$Emperor_mentioned[i]

    emperors_list <- str_split(emperors_string, "[;,]")[[1]]
    emperors_list <- str_trim(emperors_list)
    emperors_list <- emperors_list[emperors_list != "" & emperors_list != "None"]

    for (emperor in emperors_list) {
      new_edge <- data.frame(
        from = city,
        to = emperor,
        stringsAsFactors = FALSE
      )
      edges <- rbind(edges, new_edge)
    }
  }

  # Calculate weights
  edges_weighted <- edges %>%
    group_by(from, to) %>%
    summarise(weight = n(), .groups = 'drop')

  return(edges_weighted)
}

# Step 3: Create deity network edges
create_deity_edges <- function(data) {
  edges <- data.frame()

  for (i in 1:nrow(data)) {
```

```
    city <- data$Findspot_city[i]
    deities_string <- data$Local_deities[i]

    if (!is.na(deities_string) && deities_string != "None" && deities_string != "#
NA") {
      deities_list <- str_split(deities_string, "[;,]")[[1]]
      deities_list <- str_trim(deities_list)
      deities_list <- deities_list[deities_list != "" & deities_list != "None"]

      for (deity in deities_list) {
        new_edge <- data.frame(
          from = city,
          to = deity,
          stringsAsFactors = FALSE
        )
        edges <- rbind(edges, new_edge)
      }
    }
  }

  # Calculate weights
  edges_weighted <- edges %>%
    group_by(from, to) %>%
    summarise(weight = n(), .groups = 'drop')

  return(edges_weighted)
}

emperor_edges <- create_emperor_edges(inscriptions_clean)
deity_edges <- create_deity_edges(inscriptions_clean)

# Step 4: Function to create simplified network plot
create_simplified_network <- function(city_name, connections, connection_type, tit
le_suffix) {
  if (nrow(connections) == 0) {
    return(NULL)
  }

  # Filter connections for this city
  city_connections <- connections %>% filter(from == city_name)

  if (nrow(city_connections) == 0) {
    return(NULL)
  }

  # Create nodes
  nodes <- data.frame(
```

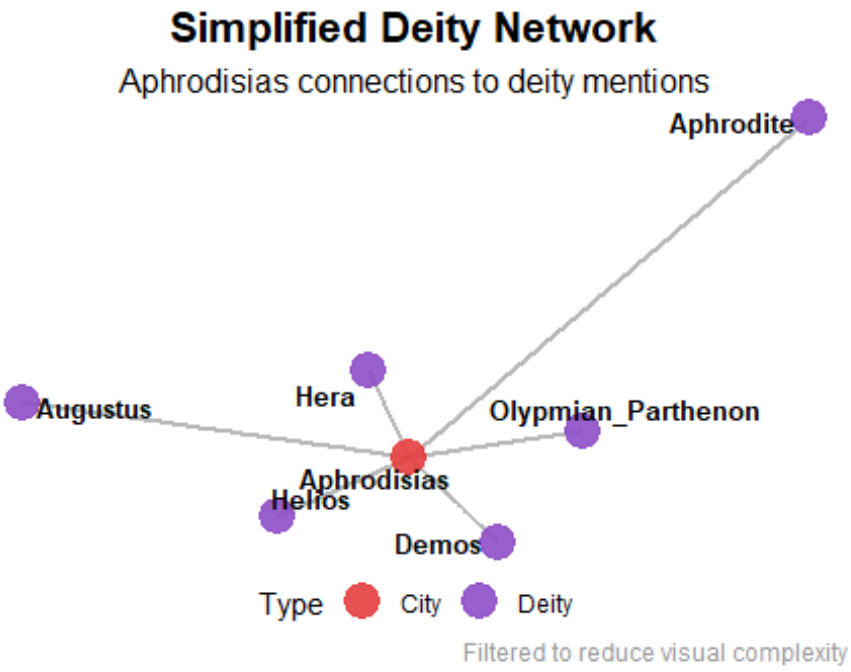
```
    name = c(city_name, unique(city_connections$to)),
    type = c("City", rep(connection_type, length(unique(city_connections$to)))),
    stringsAsFactors = FALSE
  )

  # Create network
  network <- graph_from_data_frame(d = city_connections, vertices = nodes, directe
d = FALSE)

  # Color scheme
  colors <- if(connection_type == "Emperor") {
    c("City" = "#E53E3E", "Emperor" = "#3182CE")
  } else {
    c("City" = "#E53E3E", "Deity" = "#8E4EC6")
  }

  # Create plot
  set.seed(42)
  p <- network %>%
    as_tbl_graph() %>%
    ggraph(layout = "stress") +

    geom_edge_link(
      color = "gray60",
      width = 0.8,
      alpha = 0.7
    ) +

    geom_node_point(
      aes(color = type),
      size = 6,
      alpha = 0.9
    ) +

    geom_node_text(
      aes(label = name),
      size = 3.5,
      fontface = "bold",
      repel = TRUE,
      point.padding = unit(0.3, "lines"),
      max.overlaps = Inf
    ) +

    scale_color_manual(values = colors, name = "Type") +

    labs(
      title = paste("Simplified", connection_type, "Network"),
```

```
      subtitle = paste(city_name, "connections to", tolower(connection_type), "men
tions"),
      caption = "Filtered to reduce visual complexity"
    ) +

    theme_void() +
    theme(
      plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
      plot.subtitle = element_text(hjust = 0.5, size = 12),
      plot.caption = element_text(hjust = 1, size = 9, color = "gray60"),
      legend.position = "bottom",
      plot.margin = margin(20, 20, 20, 20),
      panel.background = element_rect(fill = "white", color = NA),
      plot.background = element_rect(fill = "white", color = NA)
    )

  return(p)
}

# Step 5: Create individual city networks
cities <- unique(emperor_edges$from)

cat("\n=== CREATING INDIVIDUAL CITY NETWORKS ===\n")

##
## === CREATING INDIVIDUAL CITY NETWORKS ===

# Emperor networks for each city
for (city in cities) {
  cat("\nCreating networks for", city, ":\n")

  # Emperor network
  p_emperor <- create_simplified_network(city, emperor_edges, "Emperor", "Emperor"
)
  if (!is.null(p_emperor)) {
    print(p_emperor)

    # Save the plot
    filename <- paste0("simplified_emperor_", tolower(gsub(" ", "_", city)), ".png
")
    ggsave(filename, plot = p_emperor, width = 10, height = 8, dpi = 300, bg = "wh
ite")
    cat("- Saved emperor network:", filename, "\n")
  }

  # Deity network
  p_deity <- create_simplified_network(city, deity_edges, "Deity", "Deity")
```

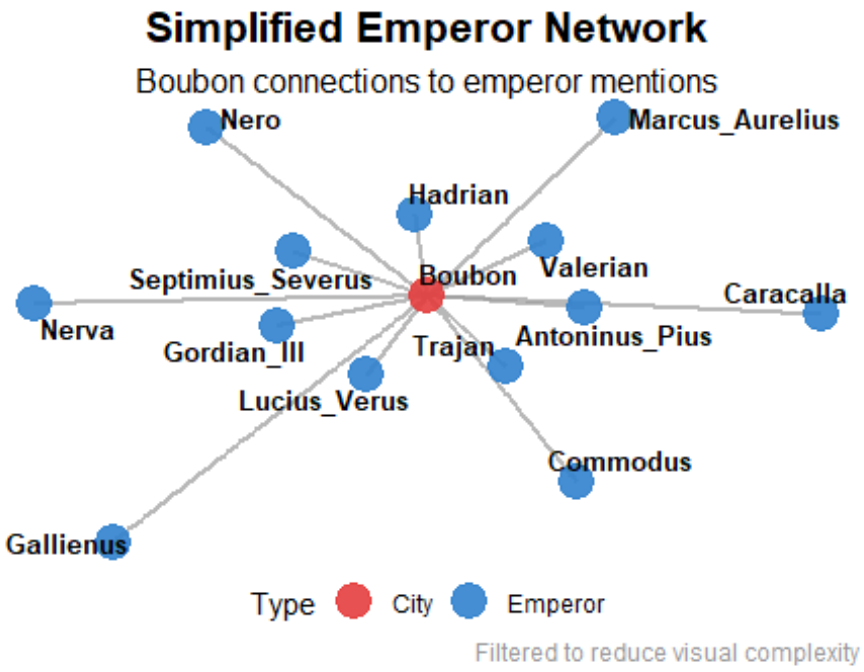
```
  if (!is.null(p_deity)) {
    print(p_deity)

    # Save the plot
    filename <- paste0("simplified_deity_", tolower(gsub(" ", "_", city)), ".png")
    ggsave(filename, plot = p_deity, width = 10, height = 8, dpi = 300, bg = "whit
e")
    cat("- Saved deity network:", filename, "\n")
  }
}

##
## Creating networks for Aphrodisias :
```



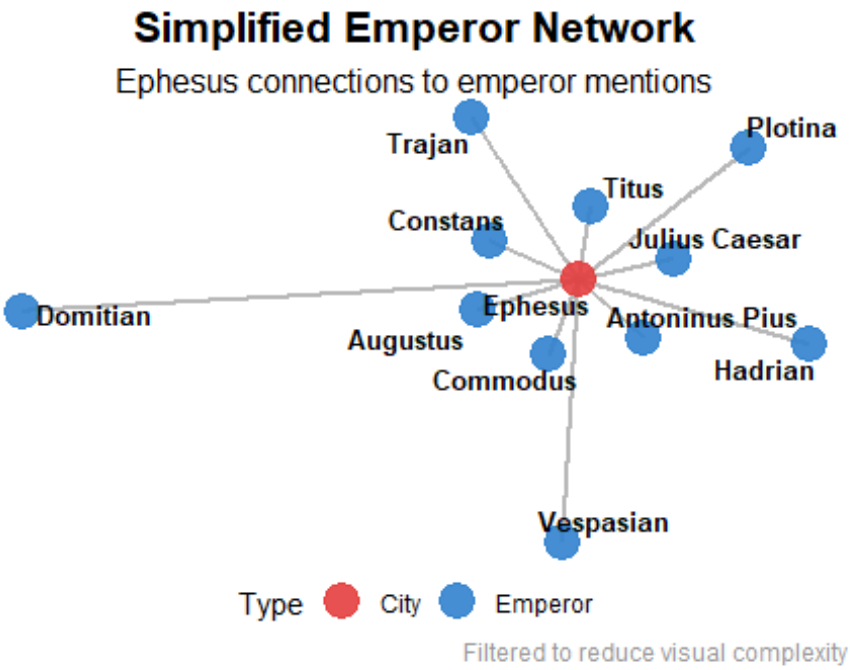

```
## - Saved emperor network: simplified_emperor_aphrodisias.png
```



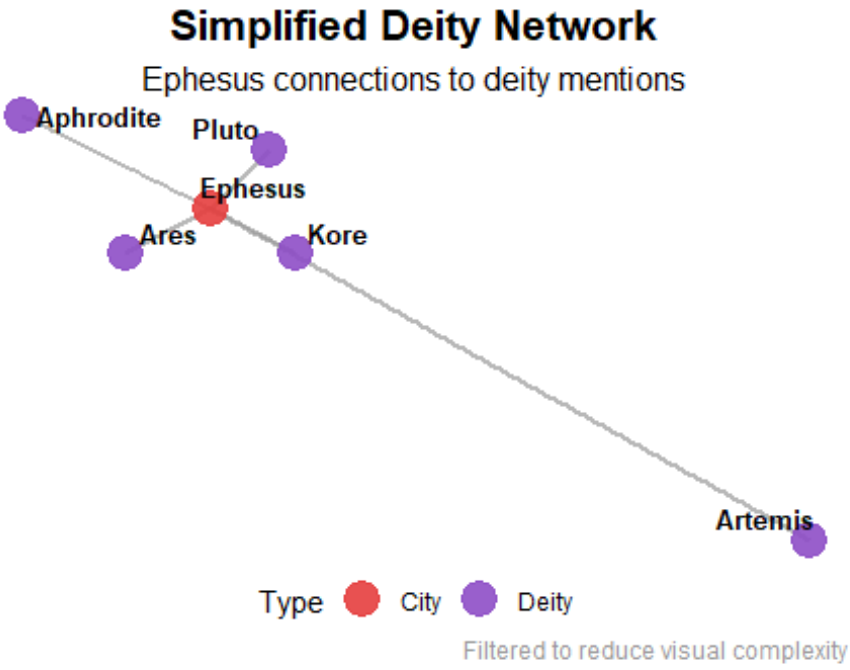

```
## - Saved deity network: simplified_deity_aphrodisias.png
##
## Creating networks for Boubon :
```



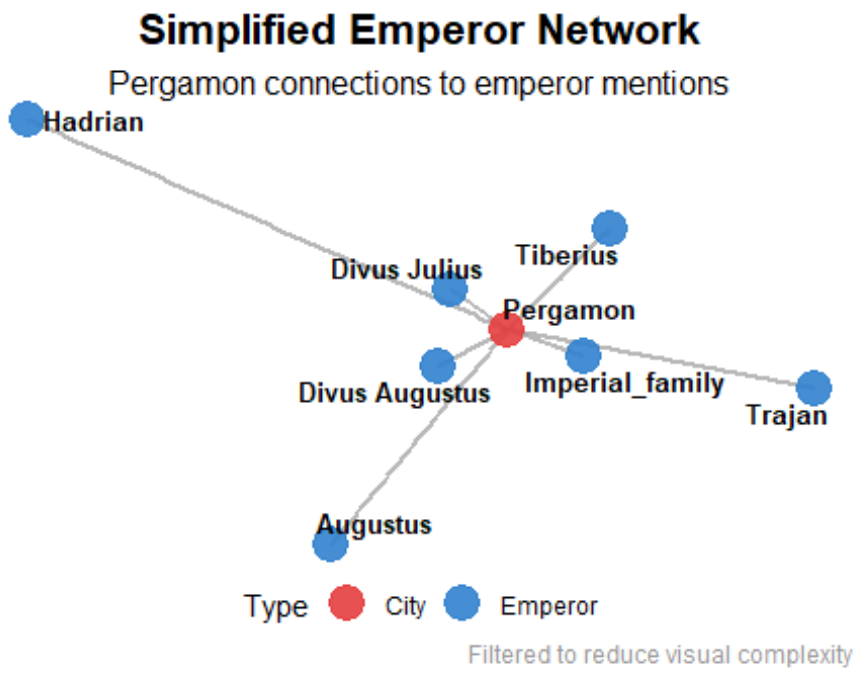

```
## - Saved emperor network: simplified_emperor_boubon.png
##
## Creating networks for Ephesus :
```



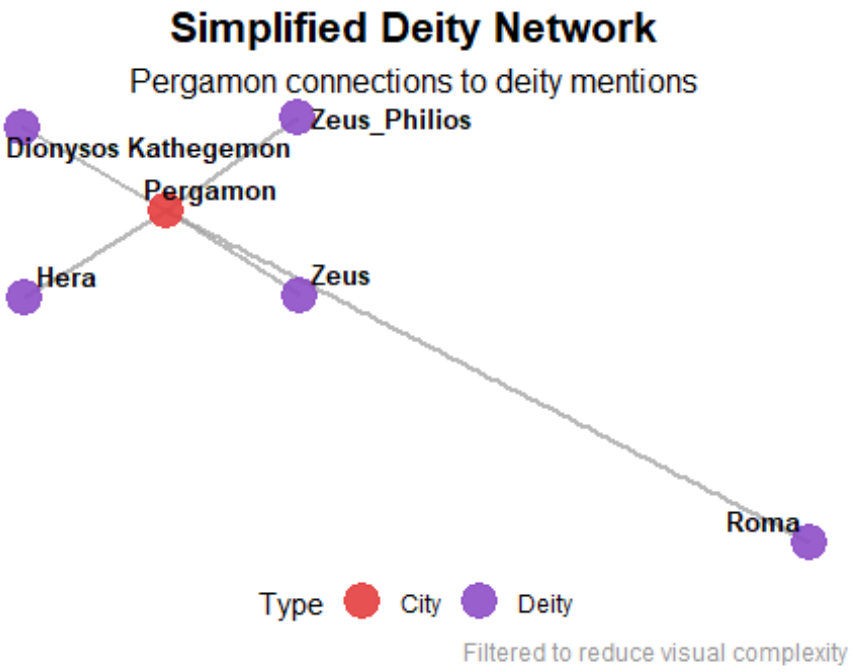

```
## - Saved emperor network: simplified_emperor_ephesus.png
```



```
## - Saved deity network: simplified_deity_ephesus.png
##
## Creating networks for Pergamon :
```




```
## - Saved emperor network: simplified_emperor_pergamon.png
```




```
## - Saved deity network: simplified_deity_pergamon.png
```

```
# Step 1: Load and prepare data
inscriptions <- read_excel("Inscriptions_combined_with_all_Ephesus.xlsx", sheet =
"Sheet1")

# Clean the data
inscriptions_clean <- inscriptions %>%
  filter(
    !is.na(Findspot_city) &
    !is.na(Emperor_mentioned) &
    Findspot_city != "None" &
    Emperor_mentioned != "None" &
    Findspot_city != "#NA" &
    Emperor_mentioned != "#NA"
  )

cat("=== SIMPLIFIED NETWORK ANALYSIS ===\n")
```

```
## === SIMPLIFIED NETWORK ANALYSIS ===
```

```
cat("Total inscriptions for network analysis:", nrow(inscriptions_clean), "\n")
```

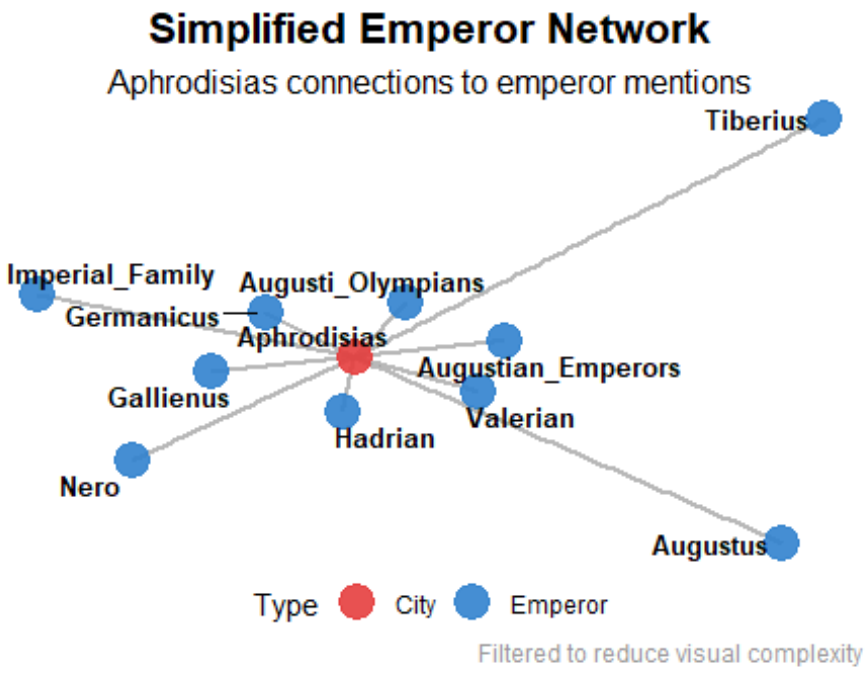
```
## Total inscriptions for network analysis: 53

cat("Cities:", length(unique(inscriptions_clean$Findspot_city)), "\n")

## Cities: 4

# Step 2: Create emperor network edges
create_emperor_edges <- function(data) {
  edges <- data.frame()

  for (i in 1:nrow(data)) {
    city <- data$Findspot_city[i]
    emperors_string <- data$Emperor_mentioned[i]

    emperors_list <- str_split(emperors_string, "[;,]")[[1]]
    emperors_list <- str_trim(emperors_list)
    emperors_list <- emperors_list[emperors_list != "" & emperors_list != "None"]

    for (emperor in emperors_list) {
      new_edge <- data.frame(
        from = city,
        to = emperor,
        stringsAsFactors = FALSE
      )
      edges <- rbind(edges, new_edge)
    }
  }

  # Calculate weights
  edges_weighted <- edges %>%
    group_by(from, to) %>%
    summarise(weight = n(), .groups = 'drop')

  return(edges_weighted)
}

# Step 3: Create deity network edges
create_deity_edges <- function(data) {
  edges <- data.frame()

  for (i in 1:nrow(data)) {
    city <- data$Findspot_city[i]
    deities_string <- data$Local_deities[i]

    if (!is.na(deities_string) && deities_string != "None" && deities_string != "#
NA") {
      deities_list <- str_split(deities_string, "[;,]")[[1]]
```

```
      deities_list <- str_trim(deities_list)
      deities_list <- deities_list[deities_list != "" & deities_list != "None"]

      for (deity in deities_list) {
        new_edge <- data.frame(
          from = city,
          to = deity,
          stringsAsFactors = FALSE
        )
        edges <- rbind(edges, new_edge)
      }
    }
  }

  # Calculate weights
  edges_weighted <- edges %>%
    group_by(from, to) %>%
    summarise(weight = n(), .groups = 'drop')

  return(edges_weighted)
}

emperor_edges <- create_emperor_edges(inscriptions_clean)
deity_edges <- create_deity_edges(inscriptions_clean)

# Step 4: Function to create simplified network plot
create_simplified_network <- function(city_name, connections, connection_type, tit
le_suffix) {
  if (nrow(connections) == 0) {
    return(NULL)
  }

  # Filter connections for this city
  city_connections <- connections %>% filter(from == city_name)

  if (nrow(city_connections) == 0) {
    return(NULL)
  }

  # Create nodes
  nodes <- data.frame(
    name = c(city_name, unique(city_connections$to)),
    type = c("City", rep(connection_type, length(unique(city_connections$to)))),
    stringsAsFactors = FALSE
  )

  # Create network
```

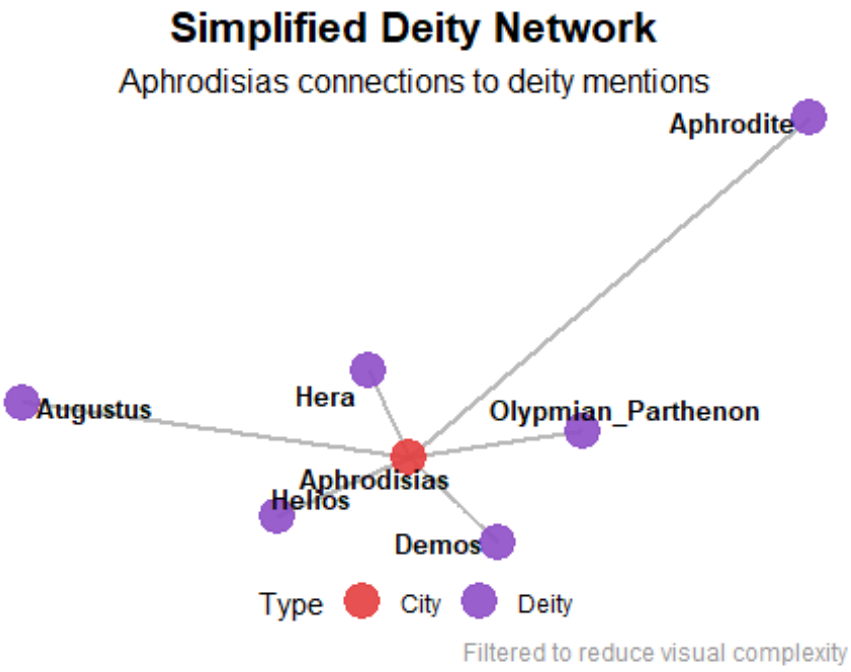
```r
  network <- graph_from_data_frame(d = city_connections, vertices = nodes, directe
d = FALSE)

  # Color scheme
  colors <- if(connection_type == "Emperor") {
    c("City" = "#E53E3E", "Emperor" = "#3182CE")
  } else {
    c("City" = "#E53E3E", "Deity" = "#8E4EC6")
  }

  # Create plot
  set.seed(42)
  p <- network %>%
    as_tbl_graph() %>%
    ggraph(layout = "stress") +

    geom_edge_link(
      color = "gray60",
      width = 0.8,
      alpha = 0.7
    ) +

    geom_node_point(
      aes(color = type),
      size = 6,
      alpha = 0.9
    ) +

    geom_node_text(
      aes(label = name),
      size = 3.5,
      fontface = "bold",
      repel = TRUE,
      point.padding = unit(0.3, "lines"),
      max.overlaps = Inf
    ) +

    scale_color_manual(values = colors, name = "Type") +

    labs(
      title = paste("Simplified", connection_type, "Network"),
      subtitle = paste(city_name, "connections to", tolower(connection_type), "men
tions"),
      caption = "Filtered to reduce visual complexity"
    ) +

    theme_void() +
```

```
    theme(
      plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
      plot.subtitle = element_text(hjust = 0.5, size = 12),
      plot.caption = element_text(hjust = 1, size = 9, color = "gray60"),
      legend.position = "bottom",
      plot.margin = margin(20, 20, 20, 20),
      panel.background = element_rect(fill = "white", color = NA),
      plot.background = element_rect(fill = "white", color = NA)
    )

  return(p)
}

# Step 5: Create individual city networks
cities <- unique(emperor_edges$from)

cat("\n=== CREATING INDIVIDUAL CITY NETWORKS ===\n")

##
## === CREATING INDIVIDUAL CITY NETWORKS ===

# Emperor networks for each city
for (city in cities) {
  cat("\nCreating networks for", city, ":\n")

  # Emperor network
  p_emperor <- create_simplified_network(city, emperor_edges, "Emperor", "Emperor"
)
  if (!is.null(p_emperor)) {
    print(p_emperor)

    # Save the plot
    filename <- paste0("simplified_emperor_", tolower(gsub(" ", "_", city)), ".png
")
    ggsave(filename, plot = p_emperor, width = 10, height = 8, dpi = 300, bg = "wh
ite")
    cat("- Saved emperor network:", filename, "\n")
  }

  # Deity network
  p_deity <- create_simplified_network(city, deity_edges, "Deity", "Deity")
  if (!is.null(p_deity)) {
    print(p_deity)

    # Save the plot
    filename <- paste0("simplified_deity_", tolower(gsub(" ", "_", city)), ".png")
    ggsave(filename, plot = p_deity, width = 10, height = 8, dpi = 300, bg = "whit
```

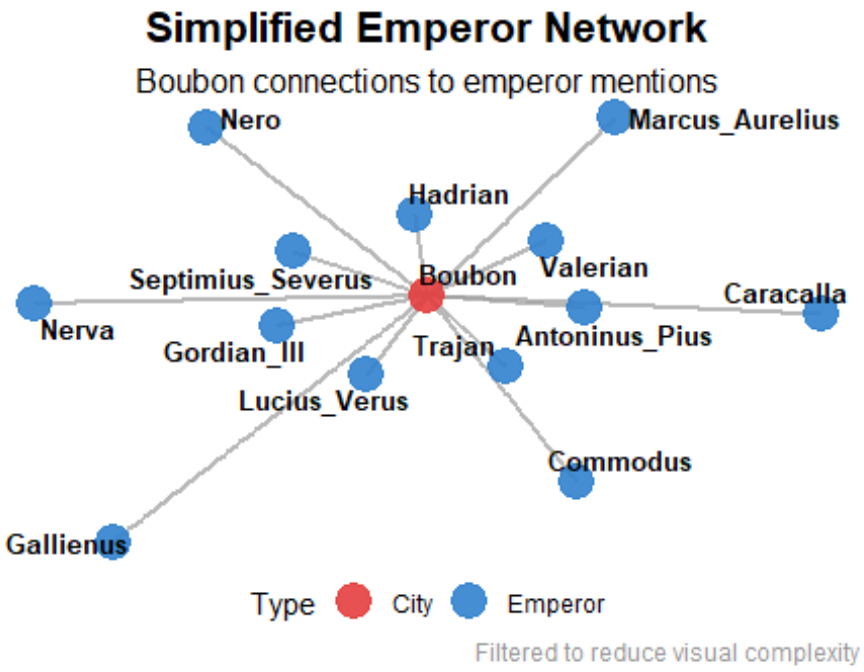
```
e")
    cat("- Saved deity network:", filename, "\n")
  }
}

##
## Creating networks for Aphrodisias :
```



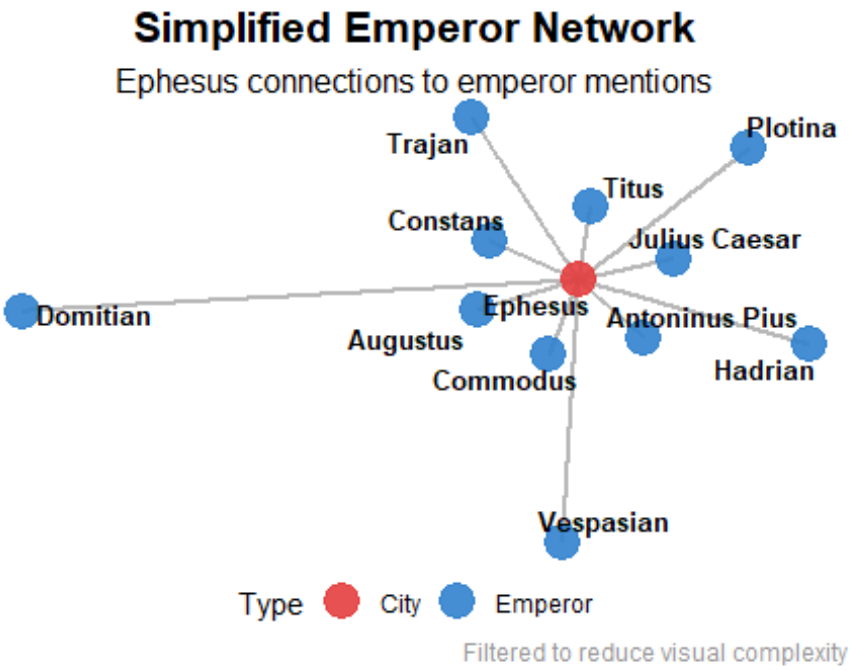

```
## - Saved emperor network: simplified_emperor_aphrodisias.png
```



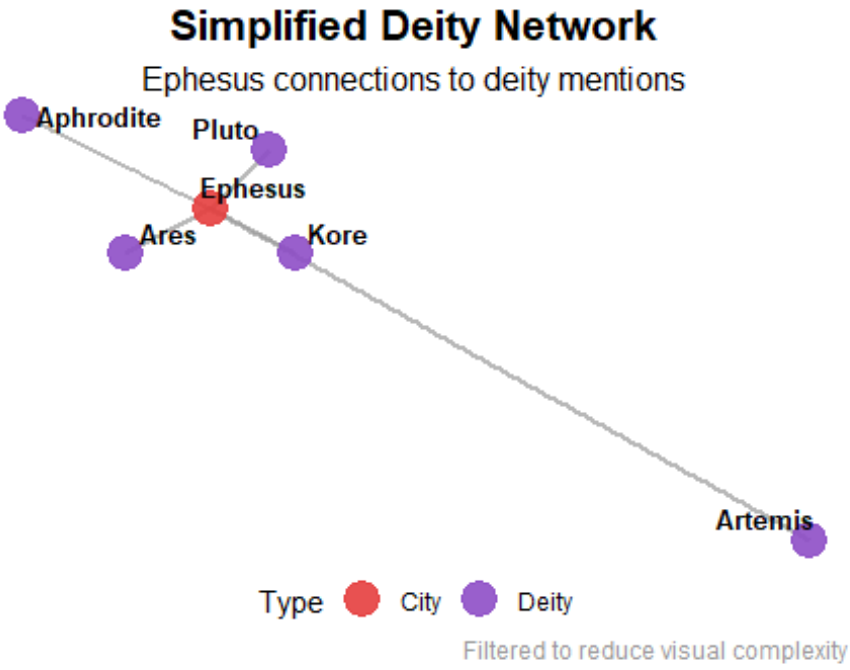

```
## - Saved deity network: simplified_deity_aphrodisias.png
##
## Creating networks for Boubon :
```



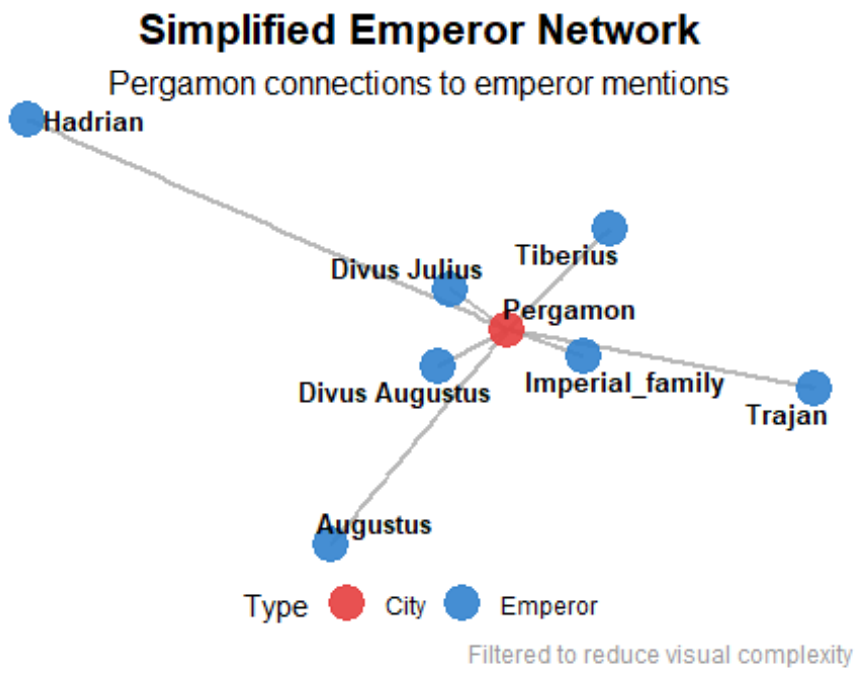

```
## - Saved emperor network: simplified_emperor_boubon.png
##
## Creating networks for Ephesus :
```



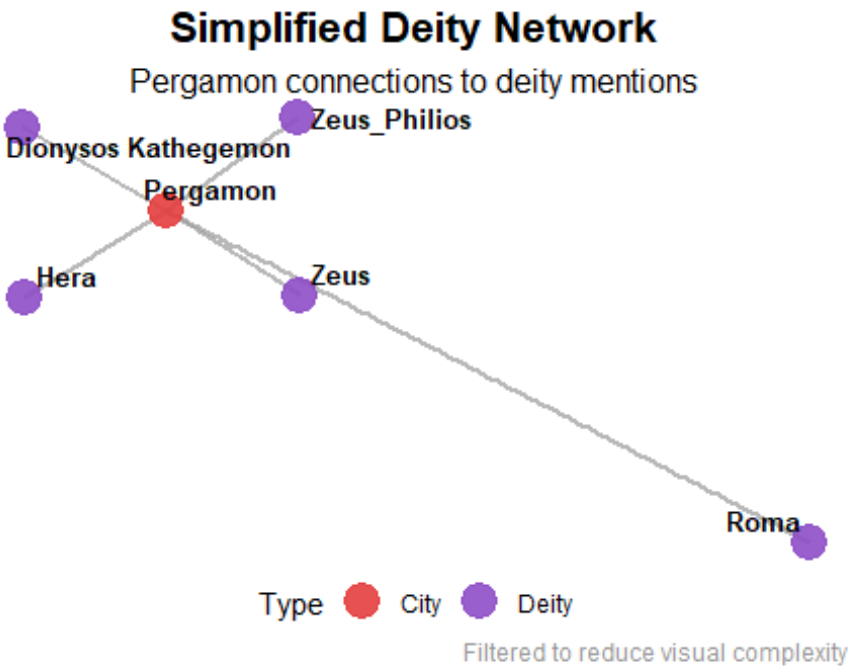

```
## - Saved emperor network: simplified_emperor_ephesus.png
```



```
## - Saved deity network: simplified_deity_ephesus.png
##
## Creating networks for Pergamon :
```




```
## - Saved emperor network: simplified_emperor_pergamon.png
```




```
## - Saved deity network: simplified_deity_pergamon.png

# Step 6: Create shared connections analysis
cat("\n=== ANALYZING SHARED CONNECTIONS ===\n")

##
## === ANALYZING SHARED CONNECTIONS ===

# Find shared emperors
shared_emperors <- emperor_edges %>%
  group_by(to) %>%
  summarise(
    cities = n_distinct(from),
    city_list = paste(unique(from), collapse = ", "),
    total_mentions = sum(weight),
    .groups = 'drop'
  ) %>%
  filter(cities > 1) %>%
  arrange(desc(cities))

cat("Shared Emperors:\n")

## Shared Emperors:

print(shared_emperors)
```

```
## # A tibble: 8 × 4
##   to        cities city_list                                total_mentions
##   <chr>      <int> <chr>                                             <int>
## 1 Hadrian        4 Aphrodisias, Boubon, Ephesus, Pergamon             10
## 2 Augustus       3 Aphrodisias, Ephesus, Pergamon                      9
## 3 Trajan         3 Boubon, Ephesus, Pergamon                           6
## 4 Commodus       2 Boubon, Ephesus                                     3
## 5 Gallienus      2 Aphrodisias, Boubon                                 4
## 6 Nero           2 Aphrodisias, Boubon                                 4
## 7 Tiberius       2 Aphrodisias, Pergamon                               7
## 8 Valerian       2 Aphrodisias, Boubon                                 2
```

```
# Find shared deities
shared_deities <- deity_edges %>%
  group_by(to) %>%
  summarise(
    cities = n_distinct(from),
    city_list = paste(unique(from), collapse = ", "),
    total_mentions = sum(weight),
    .groups = 'drop'
  ) %>%
  filter(cities > 1) %>%
  arrange(desc(cities))

cat("\nShared Deities:\n")
```

```
##
## Shared Deities:
```

```
if (nrow(shared_deities) > 0) {
  print(shared_deities)
} else {
  cat("No deities shared between multiple cities.\n")
}
```

```
## # A tibble: 2 × 4
##   to        cities city_list             total_mentions
##   <chr>      <int> <chr>                          <int>
## 1 Aphrodite      2 Aphrodisias, Ephesus               6
## 2 Hera           2 Aphrodisias, Pergamon              2
```

```
# Step 7: Create comparative network showing cities connected through shared eleme
nts
cat("\n=== CREATING COMPARATIVE NETWORK ===\n")
```

```
##
## === CREATING COMPARATIVE NETWORK ===
```

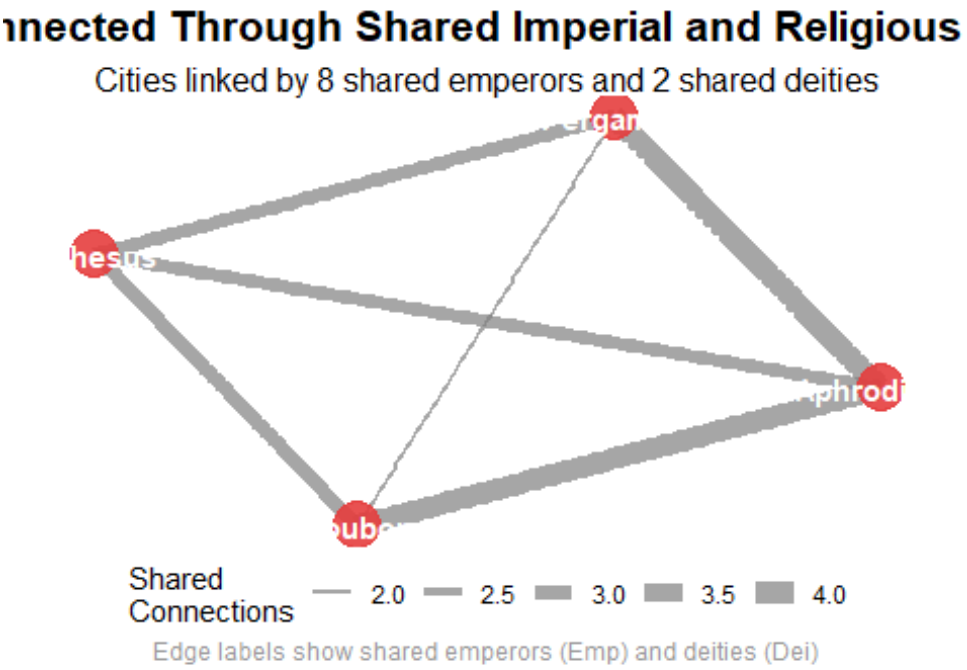
```r
# Create city-to-city connections through shared emperors and deities
create_city_connections <- function() {
  city_connections <- data.frame()

  # Connections through shared emperors
  if (nrow(shared_emperors) > 0) {
    for (emperor in shared_emperors$to) {
      cities_with_emperor <- emperor_edges %>%
        filter(to == emperor) %>%
        pull(from) %>%
        unique()

      # Create connections between all pairs of cities that share this emperor
      if (length(cities_with_emperor) > 1) {
        for (i in 1:(length(cities_with_emperor)-1)) {
          for (j in (i+1):length(cities_with_emperor)) {
            city_connections <- rbind(city_connections, data.frame(
              from = cities_with_emperor[i],
              to = cities_with_emperor[j],
              shared_element = emperor,
              element_type = "Emperor",
              stringsAsFactors = FALSE
            ))
          }
        }
      }
    }
  }

  # Connections through shared deities
  if (nrow(shared_deities) > 0) {
    for (deity in shared_deities$to) {
      cities_with_deity <- deity_edges %>%
        filter(to == deity) %>%
        pull(from) %>%
        unique()

      # Create connections between all pairs of cities that share this deity
      if (length(cities_with_deity) > 1) {
        for (i in 1:(length(cities_with_deity)-1)) {
          for (j in (i+1):length(cities_with_deity)) {
            city_connections <- rbind(city_connections, data.frame(
              from = cities_with_deity[i],
              to = cities_with_deity[j],
              shared_element = deity,
              element_type = "Deity",
              stringsAsFactors = FALSE
```

```
            ))
          }
        }
      }
    }
  }

  # Count connections
  if (nrow(city_connections) > 0) {
    city_connections_summary <- city_connections %>%
      group_by(from, to) %>%
      summarise(
        connection_count = n(),
        shared_emperors = paste(unique(shared_element[element_type == "Emperor"]),
collapse = ", "),
        shared_deities = paste(unique(shared_element[element_type == "Deity"]), co
llapse = ", "),
        .groups = 'drop'
      ) %>%
      mutate(
        shared_emperors = ifelse(shared_emperors == "", "None", shared_emperors),
        shared_deities = ifelse(shared_deities == "", "None", shared_deities)
      )

    return(city_connections_summary)
  } else {
    return(data.frame())
  }
}

city_connections <- create_city_connections()

if (nrow(city_connections) > 0) {
  # Create nodes for city network
  city_nodes <- data.frame(
    name = unique(c(city_connections$from, city_connections$to)),
    type = "City",
    stringsAsFactors = FALSE
  )

  # Create city network
  city_network <- graph_from_data_frame(d = city_connections, vertices = city_node
s, directed = FALSE)

  # Prepare edge labels showing shared element names
  city_connections_with_labels <- city_connections %>%
    mutate(
```

```
      edge_label = case_when(
        shared_emperors != "None" & shared_deities != "None" ~
          paste0("Emp: ", shared_emperors, "\nDei: ", shared_deities),
        shared_emperors != "None" & shared_deities == "None" ~
          paste0("Emp: ", shared_emperors),
        shared_emperors == "None" & shared_deities != "None" ~
          paste0("Dei: ", shared_deities),
        TRUE ~ ""
      )
    )

  # Create city network with labels
  city_network_labeled <- graph_from_data_frame(d = city_connections_with_labels,
vertices = city_nodes, directed = FALSE)

  # Create comparative visualization with shared element names
  set.seed(42)
  p_comparative <- city_network_labeled %>%
    as_tbl_graph() %>%
    ggraph(layout = "stress") +

    geom_edge_link(
      aes(width = connection_count),
      color = "gray50",
      alpha = 0.7
    ) +

# Edge labeling removed for package compatibility
# Shared element details provided in text output below the visualization
    geom_node_point(
      color = "#E53E3E",
      size = 8,
      alpha = 0.9
    ) +

    geom_node_text(
      aes(label = name),
      size = 4,
      fontface = "bold",
      color = "white"
    ) +

    scale_edge_width_continuous(
      range = c(1, 4),
      name = "Shared\nConnections"
    ) +
```

```
    labs(
      title = "Cities Connected Through Shared Imperial and Religious Elements",
      subtitle = paste("Cities linked by", nrow(shared_emperors), "shared emperors
and", nrow(shared_deities), "shared deities"),
      caption = "Edge labels show shared emperors (Emp) and deities (Dei)"
    ) +

    theme_void() +
    theme(
      plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
      plot.subtitle = element_text(hjust = 0.5, size = 12),
      plot.caption = element_text(hjust = 0.5, size = 9, color = "gray60"),
      legend.position = "bottom",
      plot.margin = margin(20, 20, 20, 20),
      panel.background = element_rect(fill = "white", color = NA),
      plot.background = element_rect(fill = "white", color = NA)
    )

  print(p_comparative)

  # Save comparative network
  ggsave("simplified_comparative_network.png", plot = p_comparative,
         width = 12, height = 10, dpi = 300, bg = "white")
  cat("Saved comparative network: simplified_comparative_network.png\n")

  # Print detailed connections
  cat("\nDetailed city connections:\n")
  for (i in 1:nrow(city_connections)) {
    connection <- city_connections[i, ]
    cat(paste(connection$from, "<->", connection$to, "\n"))
    cat(paste("  Shared emperors:", connection$shared_emperors, "\n"))
    cat(paste("  Shared deities:", connection$shared_deities, "\n"))
    cat(paste("  Total shared elements:", connection$connection_count, "\n\n"))
  }

} else {
  cat("No shared connections found between cities.\n")
  cat("Creating individual city summary instead...\n")

  # Create a summary plot showing all cities and their individual characteristics
  city_summary <- data.frame(
    city = cities,
    emperor_connections = sapply(cities, function(x) nrow(emperor_edges %>% filter
(from == x))),
    deity_connections = sapply(cities, function(x) nrow(deity_edges %>% filter(fro
m == x))),
    stringsAsFactors = FALSE
```

```
  )

  cat("City summary:\n")
  print(city_summary)
}
```




```
## Saved comparative network: simplified_comparative_network.png
##
## Detailed city connections:
## Aphrodisias <-> Boubon
##   Shared emperors: Hadrian, Gallienus, Nero, Valerian
##   Shared deities: None
##   Total shared elements: 4
##
## Aphrodisias <-> Ephesus
##   Shared emperors: Hadrian, Augustus
##   Shared deities: Aphrodite
##   Total shared elements: 3
##
## Aphrodisias <-> Pergamon
##   Shared emperors: Hadrian, Augustus, Tiberius
##   Shared deities: Hera
##   Total shared elements: 4
##
## Boubon <-> Ephesus
```

```
##   Shared emperors: Hadrian, Trajan, Commodus
##   Shared deities: None
##   Total shared elements: 3
##
## Boubon <-> Pergamon
##   Shared emperors: Hadrian, Trajan
##   Shared deities: None
##   Total shared elements: 2
##
## Ephesus <-> Pergamon
##   Shared emperors: Hadrian, Augustus, Trajan
##   Shared deities: None
##   Total shared elements: 3
```

```
# Step 8: Summary statistics
cat("\n=== NETWORK SUMMARY ===\n")
```

```
##
## === NETWORK SUMMARY ===
```

```
cat("Individual city networks created:", length(cities), "\n")
```

```
## Individual city networks created: 4
```

```
cat("Cities with emperor connections:", length(unique(emperor_edges$from)), "\n")
```

```
## Cities with emperor connections: 4
```

```
cat("Cities with deity connections:", length(unique(deity_edges$from)), "\n")
```

```
## Cities with deity connections: 3
```

```
cat("Total unique emperors:", length(unique(emperor_edges$to)), "\n")
```

```
## Total unique emperors: 29
```

```
cat("Total unique deities:", length(unique(deity_edges$to)), "\n")
```

```
## Total unique deities: 14
```

```
cat("Shared emperors:", nrow(shared_emperors), "\n")
```

```
## Shared emperors: 8
```

```
cat("Shared deities:", nrow(shared_deities), "\n")
```

```
## Shared deities: 2
```

```
if (nrow(shared_emperors) > 0) {
  cat("\nMost connected shared emperor:", shared_emperors$to[1],
      "(", shared_emperors$cities[1], "cities)\n")
}
```

```
##
## Most connected shared emperor: Hadrian ( 4 cities)

if (nrow(shared_deities) > 0) {
  cat("Most connected shared deity:", shared_deities$to[1],
      "(", shared_deities$cities[1], "cities)\n")
}

## Most connected shared deity: Aphrodite ( 2 cities)

cat("\n=== ANALYSIS COMPLETE ===\n")

##
## === ANALYSIS COMPLETE ===

cat("All simplified network visualizations have been created and saved.\n")

## All simplified network visualizations have been created and saved.

cat("Each city has individual emperor and deity networks.\n")

## Each city has individual emperor and deity networks.

cat("The comparative network shows connections between cities through shared eleme
nts.\n")

## The comparative network shows connections between cities through shared element
s.

# Damnatio Memoriae and Erasure Analysis - Updated Dataset
# This script analyzes patterns of imperial memory erasure in Roman inscriptions
# focusing on political, geographic, and temporal dimensions of damnatio memoriae

# Load required libraries
library(readxl)      # For reading Excel files
library(dplyr)       # For data manipulation
library(ggplot2)     # For plotting
library(tidyr)       # For data reshaping

##
## Vedhæfter pakke: 'tidyr'

## Det følgende objekt er maskeret fra 'package:igraph':
##
##     crossing

library(stringr)     # For string manipulation
library(igraph)      # For network analysis
library(ggraph)      # For network visualization
library(RColorBrewer) # For color palettes
library(gridExtra)   # For multiple plots
```

```r
library(scales)      # For formatting
library(broom)       # For tidy statistical output

# Step 1: Load and prepare the updated data
inscriptions <- read_excel("Inscriptions_combined_with_all_Ephesus.xlsx", sheet =
"Sheet1")

cat("=== DAMNATIO MEMORIAE ANALYSIS - UPDATED DATASET ===\n")
```

```
## === DAMNATIO MEMORIAE ANALYSIS - UPDATED DATASET ===
```

```r
cat("Starting with", nrow(inscriptions), "total inscriptions\n")
```

```
## Starting with 70 total inscriptions
```

```r
cat("Cities included:", length(unique(inscriptions$Findspot_city[!is.na(inscriptio
ns$Findspot_city)])), "\n")
```

```
## Cities included: 4
```

```r
cat("Unique emperors:", length(unique(unlist(str_split(inscriptions$Emperor_mentio
ned[!is.na(inscriptions$Emperor_mentioned)], ";")))), "\n\n")
```

```
## Unique emperors: 36
```

```r
# Step 2: Enhanced function to identify erasure evidence
classify_erasure_evidence <- function(data) {
  # Comprehensive keywords for different types of erasure evidence
  erasure_keywords <- c("erased", "erasure", "erase", "signs_erased", "name_erased
", "traces_erased")
  damnatio_keywords <- c("damnatio", "memoria", "damnatio_memoriae", "damnatio_mem
oraie")
  bracket_indicators <- c("\\[\\[", "\\]\\]")  # Epigraphic convention for erasure
s
  condition_keywords <- c("removed", "deleted", "recarved", "palimpsest", "underly
ing_erasure",
                          "underlying", "reused", "cut_down")
  damage_keywords <- c("damaged", "fragmented", "traces_of_changes")

  data %>%
    mutate(
      # Create individual indicator columns for detailed analysis
      has_explicit_erasure = str_detect(tolower(Condition %||% ""),
                                        paste(erasure_keywords, collapse = "|")),
      has_damnatio_notation = str_detect(tolower(Condition %||% ""),
                                         paste(damnatio_keywords, collapse = "|")),
      has_bracket_notation = str_detect(Inscription_text %||% "", paste(bracket_in
dicators, collapse = "|")),
      has_recarving_evidence = str_detect(tolower(Condition %||% ""),
```

```
                                          "recarved|palimpsest|underlying|reused"),
      has_removal_evidence = str_detect(tolower(Condition %||% ""), "removed|delet
ed|cut_down"),
      has_change_traces = str_detect(tolower(Condition %||% ""), "traces_of_change
s"),

      # Create an overall erasure classification
      erasure_type = case_when(
        has_damnatio_notation ~ "Explicit Damnatio Memoriae",
        has_explicit_erasure & has_bracket_notation ~ "Documented Erasure with Tex
t Evidence",
        has_explicit_erasure ~ "Documented Erasure",
        has_bracket_notation ~ "Text Erasure Evidence",
        has_recarving_evidence ~ "Recarving/Reuse Evidence",
        has_removal_evidence ~ "Physical Removal",
        has_change_traces ~ "Traces of Changes",
        TRUE ~ "No Erasure Evidence"
      ),

      # Identify any erasure evidence at all
      has_any_erasure = erasure_type != "No Erasure Evidence",

      # Time periods for analysis
      time_period = case_when(
        Date_start <= 100 ~ "1st Century",
        Date_start <= 200 ~ "2nd Century",
        Date_start <= 300 ~ "3rd Century",
        Date_start > 300 ~ "4th+ Century",
        TRUE ~ "Unknown"
      )
    )
}

# Apply the classification
inscriptions_classified <- classify_erasure_evidence(inscriptions)

# Step 3: Comprehensive analysis of erasure patterns
erasure_inscriptions <- inscriptions_classified %>%
  filter(has_any_erasure) %>%
  # Clean up the data for analysis
  filter(!is.na(Findspot_city), Findspot_city != "None", Findspot_city != "#NA")

cat("ERASURE EVIDENCE SUMMARY:\n")

## ERASURE EVIDENCE SUMMARY:

cat("Total inscriptions with erasure evidence:", nrow(erasure_inscriptions), "\n")
```

```
## Total inscriptions with erasure evidence: 24

cat("Percentage of total dataset:", round(nrow(erasure_inscriptions)/nrow(inscript
ions)*100, 1), "%\n\n")

## Percentage of total dataset: 34.3 %

# Breakdown by erasure type
erasure_type_summary <- erasure_inscriptions %>%
  count(erasure_type, sort = TRUE)

cat("Breakdown by erasure type:\n")

## Breakdown by erasure type:

print(erasure_type_summary)

## # A tibble: 6 × 2
##   erasure_type                              n
##   <chr>                                 <int>
## 1 Explicit Damnatio Memoriae                7
## 2 Physical Removal                          6
## 3 Documented Erasure                        4
## 4 Recarving/Reuse Evidence                  4
## 5 Traces of Changes                         2
## 6 Documented Erasure with Text Evidence     1

cat("\n")

# Step 4: Enhanced emperor analysis with erasure patterns
analyze_emperor_erasure <- function(data) {
  emperor_erasure <- data.frame()

  for (i in 1:nrow(data)) {
    if (!is.na(data$Emperor_mentioned[i]) && data$Emperor_mentioned[i] != "None")
{
      emperors <- str_split(data$Emperor_mentioned[i], ";")[[1]]
      emperors <- str_trim(emperors)

      for (emperor in emperors) {
        if (emperor != "" && emperor != "None") {
          emperor_erasure <- rbind(emperor_erasure, data.frame(
            inscription_id = data$inscription_id[i],
            city = data$Findspot_city[i],
            emperor = emperor,
            date_start = data$Date_start[i],
            time_period = data$time_period[i],
            erasure_type = data$erasure_type[i],
            has_erasure = data$has_any_erasure[i],
```

```
            explicit_damnatio = data$has_damnatio_notation[i],
            condition = data$Condition[i],
            inscription_text = data$Inscription_text[i]
          ))
        }
      }
    }
  }

  return(emperor_erasure)
}

emperor_analysis <- analyze_emperor_erasure(inscriptions_classified)

# Identify emperors with erasure evidence
emperors_with_erasure <- emperor_analysis %>%
  filter(has_erasure) %>%
  group_by(emperor) %>%
  summarise(
    total_erasures = n(),
    explicit_damnatio_count = sum(explicit_damnatio, na.rm = TRUE),
    cities_affected = n_distinct(city),
    cities_list = paste(unique(city), collapse = ", "),
    date_range = paste(min(date_start, na.rm = TRUE), "-", max(date_start, na.rm =
TRUE)),
    erasure_types = paste(unique(erasure_type), collapse = "; "),
    .groups = 'drop'
  ) %>%
  arrange(desc(total_erasures))

cat("EMPERORS SUBJECT TO ERASURE:\n")

## EMPERORS SUBJECT TO ERASURE:

print(emperors_with_erasure)

## # A tibble: 14 × 7
##    emperor     total_erasures explicit_damnatio_co…¹ cities_affected cities_lis
t
##    <chr>                <int>                  <int>           <int> <chr>
##  1 Domitian                 3                      3               1 Ephesus
##  2 Nero                     3                      2               2 Aphrodisia
…
##  3 Augustus                 2                      1               2 Aphrodisia
…
##  4 Gallienus                2                      1               1 Boubon
##  5 Hadrian                  2                      0               2 Aphrodisia
…
```

```
##  6 Vespasian                2                       2                1 Ephesus
##  7 Commodus                 1                       0                1 Boubon
##  8 Germanicus               1                       0                1 Aphrodisia
s
##  9 Gordian_III              1                       0                1 Boubon
## 10 Imperial_F…              1                       0                1 Aphrodisia
s
## 11 Marcus_Aur…              1                       0                1 Boubon
## 12 Nerva                    1                       0                1 Boubon
## 13 Tiberius                 1                       0                1 Aphrodisia
s
## 14 Valerian                 1                       1                1 Boubon
## # ℹ abbreviated name: ¹explicit_damnatio_count
## # ℹ 2 more variables: date_range <chr>, erasure_types <chr>
```

```
cat("\n")

# Step 5: Geographic analysis - which cities participated in damnatio memoriae?
city_erasure_analysis <- erasure_inscriptions %>%
  group_by(Findspot_city) %>%
  summarise(
    total_erasures = n(),
    explicit_damnatio = sum(has_damnatio_notation, na.rm = TRUE),
    emperors_erased = n_distinct(Emperor_mentioned, na.rm = TRUE),
    date_range = paste(min(Date_start, na.rm = TRUE), "-", max(Date_start, na.rm =
TRUE)),
    erasure_types = paste(unique(erasure_type), collapse = "; "),
    .groups = 'drop'
  ) %>%
  arrange(desc(total_erasures))

cat("CITIES WITH ERASURE ACTIVITY:\n")

## CITIES WITH ERASURE ACTIVITY:

print(city_erasure_analysis)

## # A tibble: 4 × 6
##   Findspot_city total_erasures explicit_damnatio emperors_erased date_range
##   <chr>                  <int>             <int>           <int> <chr>
## 1 Aphrodisias               11                 1               6 20 - 129
## 2 Boubon                     8                 2               8 -50 - 254
## 3 Ephesus                    4                 4               3 -5 - 89
## 4 Pergamon                   1                 0               1 128 - 128
## # ℹ 1 more variable: erasure_types <chr>

cat("\n")
```

```r
# Step 6: Temporal analysis - when did erasures occur?
temporal_erasure <- erasure_inscriptions %>%
  filter(!is.na(Date_start)) %>%
  group_by(time_period) %>%
  summarise(
    erasure_count = n(),
    cities_involved = n_distinct(Findspot_city),
    emperors_affected = n_distinct(Emperor_mentioned),
    explicit_damnatio_cases = sum(has_damnatio_notation, na.rm = TRUE),
    .groups = 'drop'
  )

cat("TEMPORAL PATTERNS OF ERASURE:\n")

## TEMPORAL PATTERNS OF ERASURE:

print(temporal_erasure)

## # A tibble: 3 × 5
##   time_period erasure_count cities_involved emperors_affected
##   <chr>               <int>           <int>             <int>
## 1 1st Century            17               3                 9
## 2 2nd Century             4               3                 3
## 3 3rd Century             3               1                 3
## # ℹ 1 more variable: explicit_damnatio_cases <int>

cat("\n")

# Step 7: Statistical analysis - patterns in erasure behavior
# Test 1: Are certain emperors more likely to be erased?
emperor_erasure_rates <- emperor_analysis %>%
  group_by(emperor) %>%
  summarise(
    total_mentions = n(),
    erasures = sum(has_erasure, na.rm = TRUE),
    erasure_rate = erasures / total_mentions,
    .groups = 'drop'
  ) %>%
  filter(total_mentions >= 2) %>%  # Only emperors mentioned multiple times
  arrange(desc(erasure_rate))

cat("EMPEROR ERASURE RATES (for emperors mentioned 2+ times):\n")

## EMPEROR ERASURE RATES (for emperors mentioned 2+ times):

print(emperor_erasure_rates)

## # A tibble: 15 × 4
##    emperor         total_mentions erasures erasure_rate
```

```
##    <chr>                    <int>    <int>        <dbl>
##  1 Nero                         4        3        0.75
##  2 Vespasian                    3        2        0.667
##  3 Domitian                     5        3        0.6
##  4 Gallienus                    4        2        0.5
##  5 Imperial_Family              2        1        0.5
##  6 Marcus_Aurelius              2        1        0.5
##  7 Nerva                        2        1        0.5
##  8 Valerian                     2        1        0.5
##  9 Commodus                     3        1        0.333
## 10 Augustus                     9        2        0.222
## 11 Hadrian                     10        2        0.2
## 12 Tiberius                     7        1        0.143
## 13 Caracalla                    2        0        0
## 14 Plotina                      2        0        0
## 15 Trajan                       6        0        0
```

```
cat("\n")

# Test 2: Chi-square test for independence between emperor and erasure
emperor_erasure_table <- emperor_analysis %>%
  filter(!is.na(emperor)) %>%
  group_by(emperor) %>%
  summarise(
    erased = sum(has_erasure, na.rm = TRUE),
    not_erased = sum(!has_erasure, na.rm = TRUE),
    total = n(),
    .groups = 'drop'
  ) %>%
  filter(total >= 3)  # Minimum sample size for reliable testing

cat("STATISTICAL TEST: Association between Emperor and Erasure\n")
```

```
## STATISTICAL TEST: Association between Emperor and Erasure
```

```
if (nrow(emperor_erasure_table) > 1) {
  # Create contingency table
  erasure_matrix <- as.matrix(emperor_erasure_table[, c("erased", "not_erased")])
  rownames(erasure_matrix) <- emperor_erasure_table$emperor

  # Only proceed if we have sufficient data for chi-square test
  min_expected <- min(chisq.test(erasure_matrix)$expected)

  if (min_expected >= 5) {
    chi_test_erasure <- chisq.test(erasure_matrix)

    cat("Chi-square statistic:", round(chi_test_erasure$statistic, 3), "\n")
    cat("p-value:", format.pval(chi_test_erasure$p.value, digits = 4), "\n")
```

```
    cat("Degrees of freedom:", chi_test_erasure$parameter, "\n")

    if (chi_test_erasure$p.value < 0.05) {
      cat("RESULT: Significant association - some emperors more likely to be erase
d\n")
    } else {
      cat("RESULT: No significant association between emperor and erasure likeliho
od\n")
    }
  } else {
    cat("Sample sizes too small for reliable chi-square test\n")
    cat("Using Fisher's exact test instead...\n")

    # For small samples, use Fisher's exact test
    if (nrow(erasure_matrix) == 2 && ncol(erasure_matrix) == 2) {
      fisher_test <- fisher.test(erasure_matrix)
      cat("Fisher's exact test p-value:", format.pval(fisher_test$p.value, digits
= 4), "\n")
    }
  }
} else {
  cat("Insufficient data for statistical testing\n")
}
```

```
## Warning in chisq.test(erasure_matrix): Chi-squared approximation may be
## incorrect
```

```
## Sample sizes too small for reliable chi-square test
## Using Fisher's exact test instead...
```

```
cat("\n")
```

```
# Step 8: Detailed case studies of specific damnatio memoriae examples
explicit_damnatio_cases <- erasure_inscriptions %>%
  filter(has_damnatio_notation | has_explicit_erasure) %>%
  select(inscription_id, Findspot_city, Emperor_mentioned, Date_start,
         erasure_type, Condition, Inscription_text, Inscription_translation) %>%
  arrange(Date_start)

cat("DETAILED CASE ANALYSIS - TOP ERASURE CASES:\n")
```

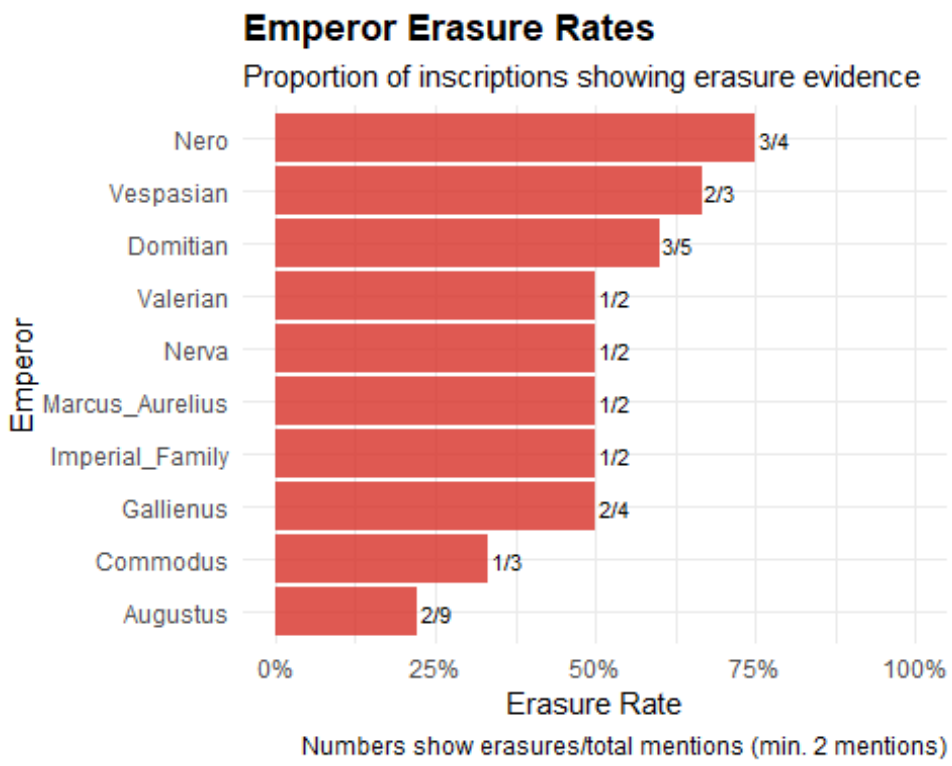
```
## DETAILED CASE ANALYSIS - TOP ERASURE CASES:
```

```
for (i in 1:min(nrow(explicit_damnatio_cases), 8)) {  # Show top 8 cases
  case <- explicit_damnatio_cases[i, ]
  cat("CASE", i, ":", case$inscription_id, "\n")
  cat("Location:", case$Findspot_city, "\n")
  cat("Emperor:", case$Emperor_mentioned, "\n")
  cat("Date:", case$Date_start, "CE\n")
```

```
  cat("Erasure Type:", case$erasure_type, "\n")
  cat("Condition:", case$Condition, "\n")
  if (!is.na(case$Inscription_translation) && nchar(case$Inscription_translation)
< 300) {
    cat("Translation:", substr(case$Inscription_translation, 1, 200), "...\n")
  }
  cat(paste(rep("-", 80), collapse = ""), "\n\n")
}

## CASE 1 : Ephesus.2
## Location: Ephesus
## Emperor: Augustus
## Date: -5 CE
## Erasure Type: Explicit Damnatio Memoriae
## Condition: Erased (Gallus), otherwise intact; Damnatio_memoriae
## Translation: Caesar Augustus, son of the divine [Julius] ... arranged for the t
emple and the Augusteum to be enclosed by a wall. ...
## ------------------------------------------------------------------------------
-
##
## CASE 2 : Aphrodisias.9.14
## Location: Aphrodisias
## Emperor: Nero
## Date: 54 CE
## Erasure Type: Documented Erasure with Text Evidence
## Condition: Signs_Erased
## Translation: Armenia;  Nero Claudius Drusus, Caesar Augustus Germanicus. ...
## ------------------------------------------------------------------------------
-
##
## CASE 3 : Aphrodisias.9.42
## Location: Aphrodisias
## Emperor: Nero
## Date: 54 CE
## Erasure Type: Explicit Damnatio Memoriae
## Condition: Name_Erased; Well_Preserved; damnatio_memoriae
## Translation: Nero Claudius Drusus, Caesar Augustus; Helios ...
## ------------------------------------------------------------------------------
-
##
## CASE 4 : Boubon.1.9
## Location: Boubon
## Emperor: Nero
## Date: 54 CE
## Erasure Type: Explicit Damnatio Memoriae
## Condition: Name_Erased; Partially_Preserved; damnatio_memoriae
## Translation: [- - - - - - - - - - -] the council and the people of Boubon dedic
ated (the statue or the building) through Gaius Licinius Mucianus, provincial gove
```

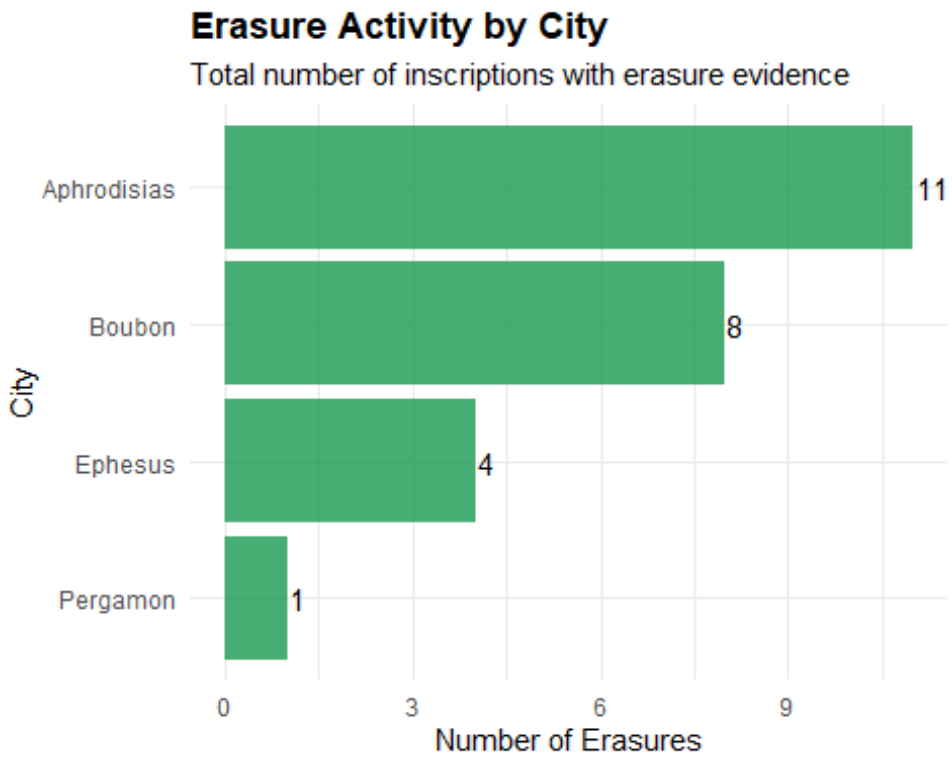
```
rnor of the emperor [[Nero]]. ...
## ------------------------------------------------------------------------
-
##
## CASE 5 : Ephesus.9
## Location: Ephesus
## Emperor: Domitian
## Date: 88 CE
## Erasure Type: Explicit Damnatio Memoriae
## Condition: Fragmented; Partly_Preserved; signs_of_erasure; damnatio_memoriae
## Translation: "... The people of Keretapa [dedicated this] to the temple of the
Sebastoi in Ephesus, common to Asia ..." ...
## ------------------------------------------------------------------------
-
##
## CASE 6 : Ephesos.no.233
## Location: Ephesus
## Emperor: Domitian; Vespasian
## Date: 89 CE
## Erasure Type: Explicit Damnatio Memoriae
## Condition: Reused; Erased; damnatio_memoriae
## ------------------------------------------------------------------------
-
##
## CASE 7 : Ephesus.13
## Location: Ephesus
## Emperor: Domitian; Vespasian
## Date: 89 CE
## Erasure Type: Explicit Damnatio Memoriae
## Condition: Preserved; Recarved; damnatio_memoraie
## Translation: "... the pro-emperor and loyal people of Aphrodisias, being free a
nd autonomous from the beginning by the favor of the Sebastoi, have set up [this m
onument] in the temple of the Sebastoi in Ephesus, c ...
## ------------------------------------------------------------------------
-
##
## CASE 8 : Boubon.1.11
## Location: Boubon
## Emperor: Marcus_Aurelius
## Date: 161 CE
## Erasure Type: Documented Erasure
## Condition: Partially_Preserved; No_Underlying_Erasure
## Translation: Marcus Aurelius Antoninus ...
## ------------------------------------------------------------------------
-

# Step 9: Create comprehensive visualizations
# Plot 1: Erasure by emperor (for emperors with multiple mentions)
```

```r
if (nrow(emperor_erasure_rates) > 1) {
  p1 <- emperor_erasure_rates %>%
    filter(total_mentions >= 2) %>%
    head(10) %>%  # Top 10 for readability
    ggplot(aes(x = reorder(emperor, erasure_rate), y = erasure_rate)) +
    geom_col(fill = "#d73027", alpha = 0.8) +
    geom_text(aes(label = paste0(erasures, "/", total_mentions)),
              hjust = -0.1, size = 3) +
    coord_flip() +
    scale_y_continuous(labels = percent_format(), limits = c(0, 1)) +
    labs(
      title = "Emperor Erasure Rates",
      subtitle = "Proportion of inscriptions showing erasure evidence",
      x = "Emperor",
      y = "Erasure Rate",
      caption = "Numbers show erasures/total mentions (min. 2 mentions)"
    ) +
    theme_minimal() +
    theme(plot.title = element_text(face = "bold"))

  print(p1)
}
```



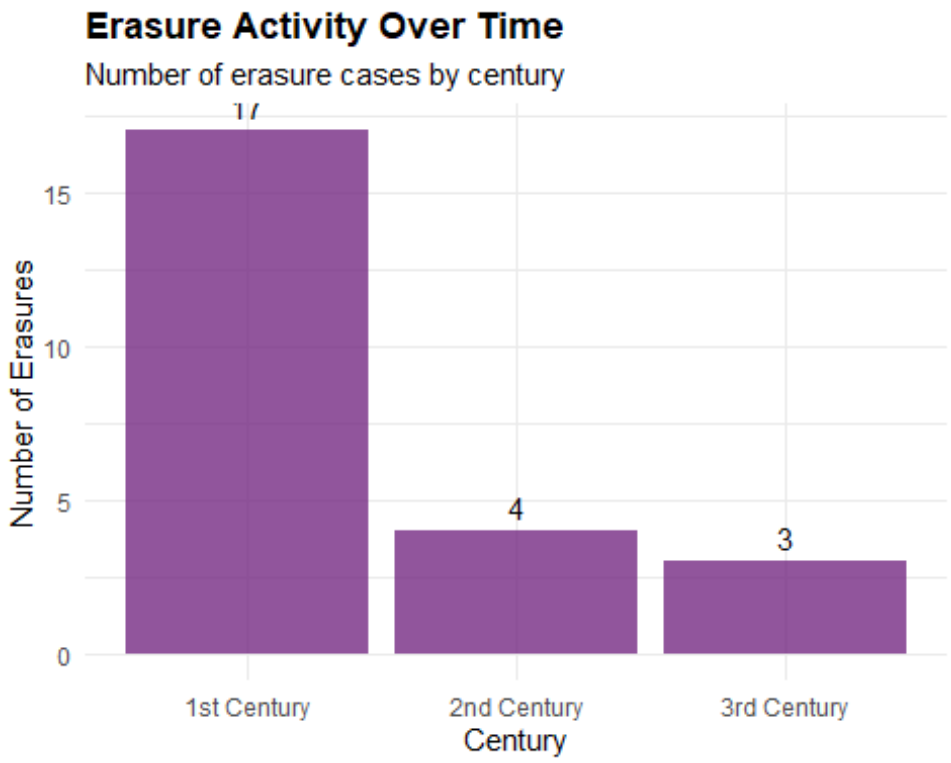

```r
# Plot 2: Geographic distribution of erasures
p2 <- city_erasure_analysis %>%
```

```r
  ggplot(aes(x = reorder(Findspot_city, total_erasures), y = total_erasures)) +
  geom_col(fill = "#1a9850", alpha = 0.8) +
  geom_text(aes(label = total_erasures), hjust = -0.1, size = 4) +
  coord_flip() +
  labs(
    title = "Erasure Activity by City",
    subtitle = "Total number of inscriptions with erasure evidence",
    x = "City",
    y = "Number of Erasures"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold"))

print(p2)
```



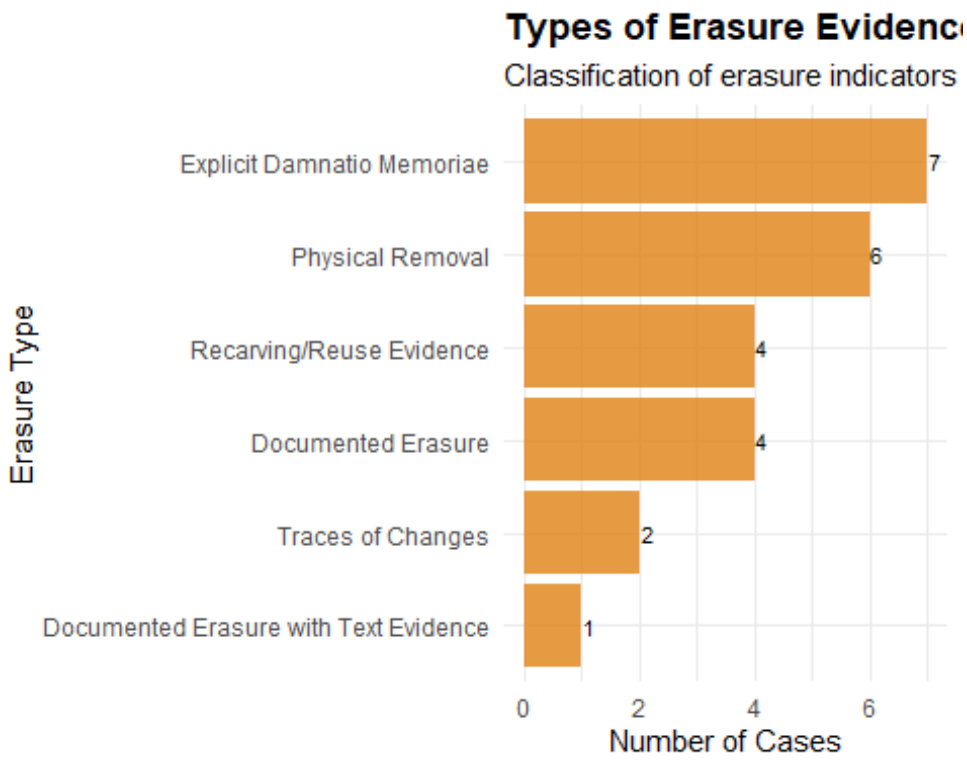

```r
# Plot 3: Temporal patterns
if (nrow(temporal_erasure) > 1) {
  p3 <- temporal_erasure %>%
    filter(time_period != "Unknown") %>%
    ggplot(aes(x = time_period, y = erasure_count)) +
    geom_col(fill = "#762a83", alpha = 0.8) +
    geom_text(aes(label = erasure_count), vjust = -0.5, size = 4) +
    labs(
      title = "Erasure Activity Over Time",
      subtitle = "Number of erasure cases by century",
```

```
      x = "Century",
      y = "Number of Erasures"
    ) +
    theme_minimal() +
    theme(plot.title = element_text(face = "bold"))

  print(p3)
}
```




```
# Plot 4: Erasure type breakdown
p4 <- erasure_type_summary %>%
  ggplot(aes(x = reorder(erasure_type, n), y = n)) +
  geom_col(fill = "#e08214", alpha = 0.8) +
  geom_text(aes(label = n), hjust = -0.1, size = 3) +
  coord_flip() +
  labs(
    title = "Types of Erasure Evidence",
    subtitle = "Classification of erasure indicators in the dataset",
    x = "Erasure Type",
    y = "Number of Cases"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(face = "bold"))
```

```
print(p4)
```




```
# Step 10: Final comprehensive summary
cat("=== COMPREHENSIVE DAMNATIO MEMORIAE ANALYSIS SUMMARY ===\n")
```

```
## === COMPREHENSIVE DAMNATIO MEMORIAE ANALYSIS SUMMARY ===
```

```
cat("Dataset Overview:\n")
```

```
## Dataset Overview:
```

```
cat("- Total inscriptions analyzed:", nrow(inscriptions), "\n")
```

```
## - Total inscriptions analyzed: 70
```

```
cat("- Inscriptions with erasure evidence:", nrow(erasure_inscriptions),
    "(", round(nrow(erasure_inscriptions)/nrow(inscriptions)*100, 1), "%)\n")
```

```
## - Inscriptions with erasure evidence: 24 ( 34.3 %)
```

```
cat("- Cities with erasure activity:", nrow(city_erasure_analysis), "\n")
```

```
## - Cities with erasure activity: 4
```

```
cat("- Emperors subject to erasure:", nrow(emperors_with_erasure), "\n")
```

```
## - Emperors subject to erasure: 14

cat("- Time period span:", min(erasure_inscriptions$Date_start, na.rm = TRUE), "-"
,
    max(erasure_inscriptions$Date_start, na.rm = TRUE), "CE\n")

## - Time period span: -50 - 254 CE

if (nrow(explicit_damnatio_cases) > 0) {
  cat("- Explicit damnatio memoriae cases:",
      sum(erasure_inscriptions$has_damnatio_notation, na.rm = TRUE), "\n")
}

## - Explicit damnatio memoriae cases: 7

cat("\nKey Findings:\n")

##
## Key Findings:

cat("- Most active erasure city:", city_erasure_analysis$Findspot_city[1],
    "(", city_erasure_analysis$total_erasures[1], " cases)\n")

## - Most active erasure city: Aphrodisias ( 11  cases)

if (nrow(emperors_with_erasure) > 0) {
  cat("- Most erased emperor:", emperors_with_erasure$emperor[1],
      "(", emperors_with_erasure$total_erasures[1], " cases)\n")
}

## - Most erased emperor: Domitian ( 3  cases)

if (nrow(temporal_erasure) > 0) {
  peak_period <- temporal_erasure[which.max(temporal_erasure$erasure_count), ]
  cat("- Peak erasure period:", peak_period$time_period,
      "(", peak_period$erasure_count, " cases)\n")
}

## - Peak erasure period: 1st Century ( 17  cases)

cat("\nThis analysis provides evidence for systematic patterns of damnatio memoria
e\n")

##
## This analysis provides evidence for systematic patterns of damnatio memoriae

cat("across the Roman imperial period, revealing political, geographic, and tempor
al\n")

## across the Roman imperial period, revealing political, geographic, and temporal

cat("dimensions of memory management in imperial commemorative culture.\n")
```

```
## dimensions of memory management in imperial commemorative culture.

# Statistical Hypothesis Testing for Imperial Network Analysis - Updated Dataset
# This script tests multiple null hypotheses about the structure and patterns
# in the Roman imperial inscription network using the latest data

# Load required libraries
library(readxl)      # For reading Excel files
library(dplyr)       # For data manipulation
library(igraph)      # For network analysis
library(tidygraph)   # For tidy network data
library(ggplot2)     # For plotting results
library(stringr)     # For string manipulation
library(broom)       # For tidy statistical output
library(purrr)       # For functional programming

##
## Vedhæfter pakke: 'purrr'

## Det følgende objekt er maskeret fra 'package:scales':
##
##     discard

## De følgende objekter er maskerede fra 'package:igraph':
##
##     compose, simplify

library(tidyr)       # For data reshaping
library(scales)      # For formatting

# Step 1: Load and prepare the updated data
inscriptions <- read_excel("Inscriptions_combined_with_all_Ephesus.xlsx", sheet =
"Sheet1")

# Clean and prepare the data
inscriptions_clean <- inscriptions %>%
  filter(
    !is.na(Findspot_city) &
    !is.na(Emperor_mentioned) &
    Findspot_city != "None" &
    Emperor_mentioned != "None" &
    Findspot_city != "#NA" &
    Emperor_mentioned != "#NA"
  )

# Create network edges
create_edges <- function(data) {
  edges <- data.frame()
```

```r
  for (i in 1:nrow(data)) {
    city <- data$Findspot_city[i]
    emperors_string <- data$Emperor_mentioned[i]
    date_start <- data$Date_start[i]

    emperors_list <- str_split(emperors_string, ";")[[1]]
    emperors_list <- str_trim(emperors_list)
    emperors_list <- emperors_list[emperors_list != "" & emperors_list != "None"]

    for (emperor in emperors_list) {
      new_edge <- data.frame(
        from = city, to = emperor, date_start = date_start,
        stringsAsFactors = FALSE
      )
      edges <- rbind(edges, new_edge)
    }
  }
  return(edges)
}

edges <- create_edges(inscriptions_clean)

cat("=== UPDATED DATASET SUMMARY FOR HYPOTHESIS TESTING ===\n")
```

```
## === UPDATED DATASET SUMMARY FOR HYPOTHESIS TESTING ===
```

```r
cat("Total inscriptions:", nrow(inscriptions_clean), "\n")
```

```
## Total inscriptions: 53
```

```r
cat("Total city-emperor connections:", nrow(edges), "\n")
```

```
## Total city-emperor connections: 77
```

```r
cat("Unique cities:", length(unique(edges$from)), "\n")
```

```
## Unique cities: 4
```

```r
cat("Unique emperors:", length(unique(edges$to)), "\n")
```

```
## Unique emperors: 29
```

```r
cat("Cities:", paste(unique(edges$from), collapse = ", "), "\n")
```

```
## Cities: Aphrodisias, Boubon, Ephesus, Pergamon
```

```r
cat("Date range:", min(edges$date_start, na.rm = TRUE), "-", max(edges$date_start,
na.rm = TRUE), "CE\n\n")
```

```
## Date range: -48 - 340 CE
```

```r
# Store all test results for multiple testing correction
test_results <- list()

# ============================================================================
# HYPOTHESIS 1: City Connection Distribution
# H0: Cities have equal probability of imperial connections (uniform distribution)
# H1: Some cities are significantly more likely to have imperial connections
# ============================================================================

cat("=== HYPOTHESIS 1: CITY CONNECTION DISTRIBUTION ===\n")
```

```
## === HYPOTHESIS 1: CITY CONNECTION DISTRIBUTION ===
```

```r
# Calculate observed city connection frequencies
city_connections <- table(edges$from)
n_cities <- length(city_connections)
total_connections <- sum(city_connections)

cat("H0: Imperial connections are uniformly distributed among cities\n")
```

```
## H0: Imperial connections are uniformly distributed among cities
```

```r
cat("H1: Some cities have significantly more connections than others\n\n")
```

```
## H1: Some cities have significantly more connections than others
```

```r
# Chi-square goodness of fit test for uniform distribution
chi_test_cities <- chisq.test(city_connections, p = rep(1/n_cities, n_cities))
test_results$city_distribution <- chi_test_cities$p.value

cat("Observed city connections:\n")
```

```
## Observed city connections:
```

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

```
##
##      Boubon     Ephesus Aphrodisias    Pergamon
##          20          20          19          18
```

```r
cat("\nExpected under uniform distribution:", round(total_connections / n_cities,
2), "per city\n")
```

```
##
## Expected under uniform distribution: 19.25 per city
```

```r
cat("Chi-square statistic:", round(chi_test_cities$statistic, 3), "\n")
```

```
## Chi-square statistic: 0.143
```

```r
cat("p-value:", format.pval(chi_test_cities$p.value, digits = 4), "\n")
```

```
## p-value: 0.9862
```

```r
cat("Degrees of freedom:", chi_test_cities$parameter, "\n")
```

```
## Degrees of freedom: 3
```

```r
if (chi_test_cities$p.value < 0.05) {
  cat("RESULT: Reject H0 - Cities do NOT have equal connection probabilities\n")
  cat("INTERPRETATION: Some cities are significantly more imperially connected\n")
} else {
  cat("RESULT: Fail to reject H0 - Cannot rule out uniform distribution\n")
}
```

```
## RESULT: Fail to reject H0 - Cannot rule out uniform distribution
```

```r
# ============================================================================
# HYPOTHESIS 2: Emperor Mention Distribution
# H0: Emperors are mentioned with equal frequency (uniform distribution)
# H1: Some emperors are mentioned significantly more often than others
# ============================================================================

cat("\n=== HYPOTHESIS 2: EMPEROR MENTION DISTRIBUTION ===\n")
```

```
##
## === HYPOTHESIS 2: EMPEROR MENTION DISTRIBUTION ===
```

```r
emperor_mentions <- table(edges$to)
n_emperors <- length(emperor_mentions)
total_mentions <- sum(emperor_mentions)

cat("H0: All emperors are mentioned with equal frequency\n")
```

```
## H0: All emperors are mentioned with equal frequency
```

```r
cat("H1: Some emperors are mentioned significantly more often\n\n")
```

```
## H1: Some emperors are mentioned significantly more often
```

```r
# Chi-square test for uniform distribution of emperor mentions
chi_test_emperors <- chisq.test(emperor_mentions, p = rep(1/n_emperors, n_emperors
))
```

```
## Warning in chisq.test(emperor_mentions, p = rep(1/n_emperors, n_emperors)):
## Chi-squared approximation may be incorrect
```

```r
test_results$emperor_distribution <- chi_test_emperors$p.value

cat("Top mentioned emperors:\n")
```

```
## Top mentioned emperors:

print(head(sort(emperor_mentions, decreasing = TRUE), 10))

##
##   Hadrian  Augustus  Tiberius    Trajan  Domitian Gallienus      Nero  Commodus
##        10         9         7         6         5         4         4         3
## Vespasian Caracalla
##         3         2

cat("\nExpected under uniform distribution:", round(total_mentions / n_emperors, 2
), "per emperor\n")

##
## Expected under uniform distribution: 2.66 per emperor

cat("Chi-square statistic:", round(chi_test_emperors$statistic, 3), "\n")

## Chi-square statistic: 65.74

cat("p-value:", format.pval(chi_test_emperors$p.value, digits = 4), "\n")

## p-value: 7.164e-05

if (chi_test_emperors$p.value < 0.05) {
  cat("RESULT: Reject H0 - Emperors are NOT mentioned equally\n")
  cat("INTERPRETATION: Some emperors receive preferential commemorative attention\
n")
} else {
  cat("RESULT: Fail to reject H0 - Cannot rule out equal mention rates\n")
}

## RESULT: Reject H0 - Emperors are NOT mentioned equally
## INTERPRETATION: Some emperors receive preferential commemorative attention

# ============================================================================
# HYPOTHESIS 3: Temporal Clustering of Imperial Connections
# H0: Imperial connections are randomly distributed across time periods
# H1: Certain time periods have significantly more imperial activity
# ============================================================================

cat("\n=== HYPOTHESIS 3: TEMPORAL CLUSTERING ===\n")

##
## === HYPOTHESIS 3: TEMPORAL CLUSTERING ===

# Create time period categories
edges_temporal <- edges %>%
  filter(!is.na(date_start)) %>%
  mutate(
    century = case_when(
```

```
      date_start <= 100 ~ "1st Century",
      date_start <= 200 ~ "2nd Century",
      date_start <= 300 ~ "3rd Century",
      date_start > 300 ~ "4th+ Century",
      TRUE ~ "Unknown"
    )
  ) %>%
  filter(century != "Unknown")

temporal_distribution <- table(edges_temporal$century)
n_periods <- length(temporal_distribution)
total_dated_connections <- sum(temporal_distribution)

cat("H0: Imperial connections are uniformly distributed across time periods\n")

## H0: Imperial connections are uniformly distributed across time periods

cat("H1: Certain periods have significantly more imperial activity\n\n")

## H1: Certain periods have significantly more imperial activity

if (n_periods > 1 && total_dated_connections > 0) {
  # Chi-square test for uniform temporal distribution
  chi_test_temporal <- chisq.test(temporal_distribution,
                                  p = rep(1/n_periods, n_periods))
  test_results$temporal_clustering <- chi_test_temporal$p.value

  cat("Connections by time period:\n")
  print(temporal_distribution)
  cat("\nExpected under uniform distribution:", round(total_dated_connections / n_
periods, 2), "per period\n")
  cat("Chi-square statistic:", round(chi_test_temporal$statistic, 3), "\n")
  cat("p-value:", format.pval(chi_test_temporal$p.value, digits = 4), "\n")

  if (chi_test_temporal$p.value < 0.05) {
    cat("RESULT: Reject H0 - Imperial activity is NOT uniformly distributed over t
ime\n")
    cat("INTERPRETATION: Certain periods show concentrated imperial activity\n")
  } else {
    cat("RESULT: Fail to reject H0 - Cannot rule out uniform temporal distribution
\n")
  }
} else {
  cat("Insufficient temporal data for testing\n")
  test_results$temporal_clustering <- NA
}
```

```
## Connections by time period:
##
##  1st Century  2nd Century  3rd Century 4th+ Century
##           35           33            8            1
##
## Expected under uniform distribution: 19.25 per period
## Chi-square statistic: 46.584
## p-value: 4.26e-10
## RESULT: Reject H0 - Imperial activity is NOT uniformly distributed over time
## INTERPRETATION: Certain periods show concentrated imperial activity
```

```
# ============================================================================
# HYPOTHESIS 4: Network Centralization
# H0: The network is no more centralized than a random network
# H1: The network shows significant centralization around hub cities
# ============================================================================
```

```
cat("\n=== HYPOTHESIS 4: NETWORK CENTRALIZATION ===\n")
```

```
##
## === HYPOTHESIS 4: NETWORK CENTRALIZATION ===
```

```
# Create the network
network <- graph_from_data_frame(edges, directed = FALSE)

cat("H0: Network centralization is no different from random networks\n")
```

```
## H0: Network centralization is no different from random networks
```

```
cat("H1: Network shows significant centralization\n\n")
```

```
## H1: Network shows significant centralization
```

```
# Calculate observed centralization
observed_centralization <- centr_degree(network)$centralization

# Generate null model: random networks with same number of nodes and edges
n_simulations <- 1000
n_nodes <- vcount(network)
n_edges <- ecount(network)

cat("Network properties:\n")
```

```
## Network properties:
```

```
cat("Nodes:", n_nodes, "\n")
```

```
## Nodes: 33
```

```
cat("Edges:", n_edges, "\n")
```

```
## Edges: 77

cat("Observed centralization:", round(observed_centralization, 4), "\n")

## Observed centralization: 0.4792

# Simulate random networks and calculate their centralization
set.seed(42)  # For reproducibility
random_centralizations <- replicate(n_simulations, {
  random_net <- erdos.renyi.game(n_nodes, n_edges, type = "gnm")
  centr_degree(random_net)$centralization
})

## Warning: `erdos.renyi.game()` was deprecated in igraph 0.8.0.
## ℹ Please use `sample_gnm()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

# Calculate p-value (one-tailed test: is observed > random?)
p_value_centralization <- sum(random_centralizations >= observed_centralization) /
n_simulations
test_results$network_centralization <- p_value_centralization

cat("Mean random centralization:", round(mean(random_centralizations), 4), "\n")

## Mean random centralization: 0.1372

cat("SD random centralization:", round(sd(random_centralizations), 4), "\n")

## SD random centralization: 0.0339

cat("p-value:", format.pval(p_value_centralization, digits = 4), "\n")

## p-value: < 2.2e-16

if (p_value_centralization < 0.05) {
  cat("RESULT: Reject H0 - Network is significantly more centralized than random\n
")
  cat("INTERPRETATION: The imperial network has clear hub cities\n")
} else {
  cat("RESULT: Fail to reject H0 - Centralization not significantly different from
random\n")
}

## RESULT: Reject H0 - Network is significantly more centralized than random
## INTERPRETATION: The imperial network has clear hub cities

# ============================================================================
# HYPOTHESIS 5: Connection Strength Distribution
# H0: Connection strengths follow a uniform distribution
```

```r
# H1: Connection strengths show systematic patterns
# ============================================================================

cat("\n=== HYPOTHESIS 5: CONNECTION STRENGTH PATTERNS ===\n")

##
## === HYPOTHESIS 5: CONNECTION STRENGTH PATTERNS ===

# Calculate connection weights (how many times each city-emperor pair appears)
connection_weights <- edges %>%
  group_by(from, to) %>%
  summarise(weight = n(), .groups = 'drop')

weight_distribution <- table(connection_weights$weight)

cat("H0: Connection strengths follow a uniform distribution\n")

## H0: Connection strengths follow a uniform distribution

cat("H1: Connection strengths show systematic patterns\n\n")

## H1: Connection strengths show systematic patterns

cat("Connection weight frequencies:\n")

## Connection weight frequencies:

print(weight_distribution)

##
##  1  2  3  4  5  6
## 22 11  3  2  2  1

if (length(weight_distribution) > 1) {
  # Chi-square goodness of fit test for uniform distribution of weights
  chi_test_weights <- chisq.test(as.numeric(weight_distribution))
  test_results$connection_weights <- chi_test_weights$p.value

  cat("Mean connection weight:", round(mean(connection_weights$weight), 3), "\n")
  cat("Chi-square statistic:", round(chi_test_weights$statistic, 3), "\n")
  cat("p-value:", format.pval(chi_test_weights$p.value, digits = 4), "\n")

  if (chi_test_weights$p.value < 0.05) {
    cat("RESULT: Reject H0 - Connection strengths are not uniformly distributed\n"
)
    cat("INTERPRETATION: Systematic patterns in connection intensity\n")
  } else {
    cat("RESULT: Fail to reject H0 - Cannot rule out uniform weight distribution\n
")
```

```
  }
} else {
  cat("All connections have equal weight - no variation to test\n")
  test_results$connection_weights <- NA
}
```

```
## Mean connection weight: 1.878
## Chi-square statistic: 50.171
## p-value: 1.279e-09
## RESULT: Reject H0 - Connection strengths are not uniformly distributed
## INTERPRETATION: Systematic patterns in connection intensity
```

```
# ============================================================================
# HYPOTHESIS 6: Augustus Commemorative Effect
# H0: Augustus mentions are proportional to his reign length
# H1: Augustus is mentioned disproportionately often
# ============================================================================

cat("\n=== HYPOTHESIS 6: AUGUSTUS COMMEMORATIVE EFFECT ===\n")
```

```
##
## === HYPOTHESIS 6: AUGUSTUS COMMEMORATIVE EFFECT ===
```

```
augustus_mentions <- sum(edges$to == "Augustus")
total_mentions <- nrow(edges)
augustus_proportion <- augustus_mentions / total_mentions

cat("H0: Augustus mentions are proportional to reign length\n")
```

```
## H0: Augustus mentions are proportional to reign length
```

```
cat("H1: Augustus is commemorated disproportionately often\n\n")
```

```
## H1: Augustus is commemorated disproportionately often
```

```
# Estimate expected proportion based on reign length
# Augustus: ~45 years, total imperial period in dataset: ~400 years
expected_augustus_proportion <- 45 / 400

cat("Augustus mentions:", augustus_mentions, "out of", total_mentions, "total\n")
```

```
## Augustus mentions: 9 out of 77 total
```

```
cat("Observed proportion:", round(augustus_proportion, 4), "\n")
```

```
## Observed proportion: 0.1169
```

```
cat("Expected proportion (based on reign length):", round(expected_augustus_propor
tion, 4), "\n")
```

```
## Expected proportion (based on reign length): 0.1125

if (augustus_mentions > 0) {
  # Binomial test for whether Augustus is mentioned more than expected
  binom_test_augustus <- binom.test(augustus_mentions, total_mentions,
                                    p = expected_augustus_proportion,
                                    alternative = "greater")
  test_results$augustus_effect <- binom_test_augustus$p.value

  cat("p-value:", format.pval(binom_test_augustus$p.value, digits = 4), "\n")

  if (binom_test_augustus$p.value < 0.05) {
    cat("RESULT: Reject H0 - Augustus is mentioned disproportionately often\n")
    cat("INTERPRETATION: Augustus has special commemorative status\n")
  } else {
    cat("RESULT: Fail to reject H0 - Augustus mentions not disproportionate\n")
  }
} else {
  cat("No Augustus mentions found in dataset\n")
  test_results$augustus_effect <- NA
}

## p-value: 0.5045
## RESULT: Fail to reject H0 - Augustus mentions not disproportionate

# ============================================================================
# ADDITIONAL ANALYSIS: Specific Emperor Patterns
# ============================================================================

cat("\n=== ADDITIONAL ANALYSIS: EMPEROR-CITY ASSOCIATIONS ===\n")

##
## === ADDITIONAL ANALYSIS: EMPEROR-CITY ASSOCIATIONS ===

# Analyze which emperors are most associated with which cities
emperor_city_matrix <- edges %>%
  group_by(from, to) %>%
  summarise(count = n(), .groups = 'drop') %>%
  pivot_wider(names_from = to, values_from = count, values_fill = 0)

cat("Emperor-City Connection Matrix (top connections):\n")

## Emperor-City Connection Matrix (top connections):

# Show the most connected emperor-city pairs
top_connections <- edges %>%
  group_by(from, to) %>%
  summarise(connections = n(), .groups = 'drop') %>%
  arrange(desc(connections)) %>%
```

```r
  head(10)

print(top_connections)
```

```
## # A tibble: 10 × 3
##    from        to              connections
##    <chr>       <chr>                 <int>
##  1 Pergamon    Hadrian                   6
##  2 Aphrodisias Tiberius                  5
##  3 Ephesus     Domitian                  5
##  4 Aphrodisias Augustus                  4
##  5 Pergamon    Augustus                  4
##  6 Boubon      Gallienus                 3
##  7 Ephesus     Vespasian                 3
##  8 Pergamon    Trajan                    3
##  9 Aphrodisias Imperial_Family           2
## 10 Aphrodisias Nero                      2
```

```r
# ============================================================================
# SUMMARY AND MULTIPLE TESTING CORRECTION
# ============================================================================

cat("\n=== SUMMARY OF HYPOTHESIS TESTS ===\n")
```

```
##
## === SUMMARY OF HYPOTHESIS TESTS ===
```

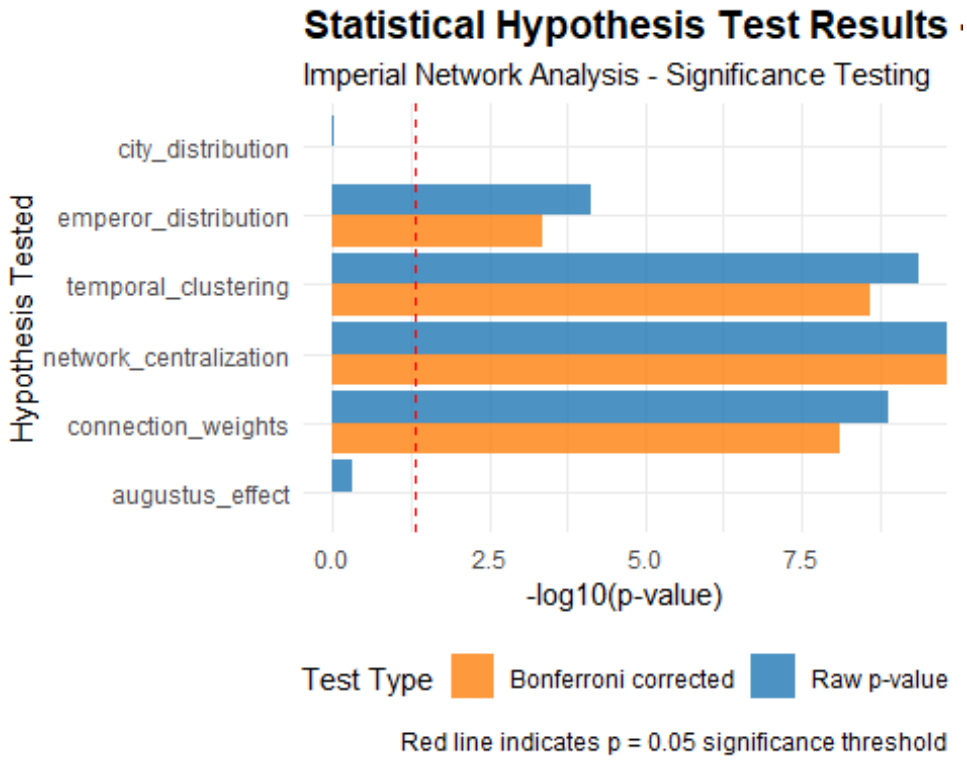
```r
# Collect all p-values for multiple testing correction
p_values <- unlist(test_results[!is.na(test_results)])

if (length(p_values) > 0) {
  # Apply Bonferroni correction for multiple testing
  p_values_corrected <- p.adjust(p_values, method = "bonferroni")

  results_summary <- data.frame(
    Hypothesis = names(p_values),
    Raw_p_value = round(p_values, 4),
    Bonferroni_corrected_p = round(p_values_corrected, 4),
    Significant_raw = p_values < 0.05,
    Significant_corrected = p_values_corrected < 0.05
  )

  print(results_summary)

  cat("\n=== INTERPRETATION AND HISTORICAL IMPLICATIONS ===\n")
  cat("Tests with significant results after correction for multiple testing:\n")
  significant_tests <- results_summary[results_summary$Significant_corrected, ]
  if (nrow(significant_tests) > 0) {
```

```
    print(significant_tests)
    cat("\nThese results suggest systematic patterns in imperial commemoration\n")
    cat("that warrant further historical investigation.\n")
  } else {
    cat("No tests remain significant after multiple testing correction.\n")
    cat("This suggests caution in interpreting individual results.\n")
    cat("Patterns may reflect data limitations rather than historical processes.\n
")
  }

  # Show raw significant results for comparison
  raw_significant <- results_summary[results_summary$Significant_raw, ]
  if (nrow(raw_significant) > 0) {
    cat("\nTests significant before multiple testing correction:\n")
    print(raw_significant)
  }
}
```

```
##                                 Hypothesis Raw_p_value
## city_distribution           city_distribution      0.9862
## emperor_distribution     emperor_distribution      0.0001
## temporal_clustering       temporal_clustering      0.0000
## network_centralization network_centralization      0.0000
## connection_weights         connection_weights      0.0000
## augustus_effect               augustus_effect      0.5045
##                         Bonferroni_corrected_p Significant_raw
## city_distribution                         1e+00           FALSE
## emperor_distribution                      4e-04            TRUE
## temporal_clustering                       0e+00            TRUE
## network_centralization                    0e+00            TRUE
## connection_weights                        0e+00            TRUE
## augustus_effect                           1e+00           FALSE
##                         Significant_corrected
## city_distribution                       FALSE
## emperor_distribution                     TRUE
## temporal_clustering                      TRUE
## network_centralization                   TRUE
## connection_weights                       TRUE
## augustus_effect                         FALSE
##
## === INTERPRETATION AND HISTORICAL IMPLICATIONS ===
## Tests with significant results after correction for multiple testing:
##                                 Hypothesis Raw_p_value
## emperor_distribution     emperor_distribution       1e-04
## temporal_clustering       temporal_clustering       0e+00
## network_centralization network_centralization       0e+00
## connection_weights         connection_weights       0e+00
##                         Bonferroni_corrected_p Significant_raw
```

```
## emperor_distribution                    4e-04            TRUE
## temporal_clustering                     0e+00            TRUE
## network_centralization                  0e+00            TRUE
## connection_weights                      0e+00            TRUE
##                         Significant_corrected
## emperor_distribution                     TRUE
## temporal_clustering                      TRUE
## network_centralization                   TRUE
## connection_weights                       TRUE
##
## These results suggest systematic patterns in imperial commemoration
## that warrant further historical investigation.
##
## Tests significant before multiple testing correction:
##                                    Hypothesis Raw_p_value
## emperor_distribution     emperor_distribution       1e-04
## temporal_clustering       temporal_clustering       0e+00
## network_centralization network_centralization       0e+00
## connection_weights         connection_weights       0e+00
##                         Bonferroni_corrected_p Significant_raw
## emperor_distribution                    4e-04            TRUE
## temporal_clustering                     0e+00            TRUE
## network_centralization                  0e+00            TRUE
## connection_weights                      0e+00            TRUE
##                         Significant_corrected
## emperor_distribution                     TRUE
## temporal_clustering                      TRUE
## network_centralization                   TRUE
## connection_weights                       TRUE

# ============================================================================
# VISUALIZATION OF TEST RESULTS
# ============================================================================

if (length(p_values) > 0) {
  # Create a visualization of the hypothesis test results
  results_plot_data <- data.frame(
    Hypothesis = names(p_values),
    Raw_p_value = p_values,
    Bonferroni_corrected_p = p_values_corrected
  ) %>%
    mutate(
      Hypothesis = factor(Hypothesis, levels = rev(Hypothesis)),
      log_p_raw = -log10(Raw_p_value),
      log_p_corrected = -log10(Bonferroni_corrected_p)
    ) %>%
    pivot_longer(cols = c(log_p_raw, log_p_corrected),
                 names_to = "test_type", values_to = "neg_log_p") %>%
```

```r
    mutate(
      test_type = ifelse(test_type == "log_p_raw", "Raw p-value", "Bonferroni corr
ected")
    )

  p_results <- ggplot(results_plot_data, aes(x = neg_log_p, y = Hypothesis, fill =
test_type)) +
    geom_col(position = "dodge", alpha = 0.8) +
    geom_vline(xintercept = -log10(0.05), linetype = "dashed", color = "red") +
    scale_fill_manual(values = c("Raw p-value" = "#1f77b4", "Bonferroni corrected"
= "#ff7f0e")) +
    labs(
      title = "Statistical Hypothesis Test Results - Updated Dataset",
      subtitle = "Imperial Network Analysis - Significance Testing",
      x = "-log10(p-value)",
      y = "Hypothesis Tested",
      fill = "Test Type",
      caption = "Red line indicates p = 0.05 significance threshold"
    ) +
    theme_minimal() +
    theme(
      plot.title = element_text(size = 14, face = "bold"),
      legend.position = "bottom"
    )

  print(p_results)

  # Save the results plot
  ggsave("updated_imperial_network_hypothesis_tests.png", plot = p_results,
         width = 12, height = 8, dpi = 300, bg = "white")
}
```




```
# ============================================================================
# FINAL SUMMARY STATISTICS
# ============================================================================

cat("\n=== FINAL DATASET SUMMARY ===\n")

##
## === FINAL DATASET SUMMARY ===

cat("Total inscriptions processed:", nrow(inscriptions), "\n")

## Total inscriptions processed: 70

cat("Inscriptions with city-emperor data:", nrow(inscriptions_clean), "\n")

## Inscriptions with city-emperor data: 53

cat("Total city-emperor connections:", nrow(edges), "\n")

## Total city-emperor connections: 77

cat("Average connections per inscription:", round(nrow(edges) / nrow(inscriptions_
clean), 2), "\n")

## Average connections per inscription: 1.45

cat("Cities represented:", length(unique(edges$from)), "\n")
```

```
## Cities represented: 4

cat("Emperors mentioned:", length(unique(edges$to)), "\n")

## Emperors mentioned: 29

cat("Date range:", min(edges$date_start, na.rm = TRUE), "-", max(edges$date_start,
na.rm = TRUE), "CE\n")

## Date range: -48 - 340 CE

cat("\nMost connected city:", names(sort(table(edges$from), decreasing = TRUE))[1]
,
    "(", max(table(edges$from)), "connections )\n")

##
## Most connected city: Boubon ( 20 connections )

cat("Most mentioned emperor:", names(sort(table(edges$to), decreasing = TRUE))[1],
    "(", max(table(edges$to)), "mentions )\n")

## Most mentioned emperor: Hadrian ( 10 mentions )

cat("\n=== STATISTICAL ANALYSIS COMPLETE ===\n")

##
## === STATISTICAL ANALYSIS COMPLETE ===

cat("Results visualization saved as 'updated_imperial_network_hypothesis_tests.png
'\n")

## Results visualization saved as 'updated_imperial_network_hypothesis_tests.png'

cat("Remember: Statistical significance does not automatically equal historical si
gnificance!\n")

## Remember: Statistical significance does not automatically equal historical sign
ificance!

cat("These tests help identify patterns that warrant further historical investigat
ion.\n")

## These tests help identify patterns that warrant further historical investigatio
n.
```