**CSV:**

# Load required library

if (!requireNamespace("igraph", quietly = TRUE)) install.packages("igraph")

library(igraph)

residue\_data <- data.frame(Aa, Ab, Ac, Ca, Cb, Cc, Da, Db, Dc, Ea, Eb, Ec, Fa, Fb, Fc, Ga, Gb, Gc, Ha, Hb, Hc, Ia, Ib, Ic, Ka, Kb, Kc, La, Lb, Lc, Ma, Mb, Mc, Na, Nb, Nc, Pa, Pb, Pc, Qa, Qb, Qc, Ra, Rb, Rc, Sa, Sb, Sc, Ta, Tb, Tc, Va, Vb, Vc, Wa, Wb, Wc, Ya, Yb, Yc)

# Compute correlation matrix for conservation scores

cor\_matrix <- cor(residue\_data, method = "pearson", use = "pairwise.complete.obs") # Pearson correlation

# Lower correlation threshold to increase network connectivity

cor\_threshold <- 0.1 # Allow more weak but meaningful connections

cor\_matrix[abs(cor\_matrix) < cor\_threshold] <- 0 # Remove weak connections

# Convert correlation matrix into an adjacency matrix

adjacency\_matrix <- abs(cor\_matrix) # Keep only strong correlations

# Convert the adjacency matrix into a graph object

network\_graph <- graph\_from\_adjacency\_matrix(adjacency\_matrix, mode = "undirected", weighted = TRUE, diag = FALSE)

# Compute Betweenness Centrality for all residues

betweenness\_scores <- betweenness(network\_graph, directed = FALSE, weights = E(network\_graph)$weight, normalized = TRUE)

# Compute Degree Centrality

degree\_scores <- degree(network\_graph, mode = "all", normalized = TRUE)

# Compute Weighted Degree Centrality (Strength)

strength\_scores <- strength(network\_graph, mode = "all")

# Check for network components (disconnected clusters)

components\_info <- components(network\_graph)

print(paste("Number of Disconnected Components:", components\_info$no))

# Compute a score based on Betweenness Centrality (normalized to 0-100)

if (max(betweenness\_scores) - min(betweenness\_scores) > 0) {

betweenness\_score <- (betweenness\_scores - min(betweenness\_scores)) / (max(betweenness\_scores) - min(betweenness\_scores)) \* 100

} else {

betweenness\_score <- betweenness\_scores # Avoid NaN if all values are the same

}

# Create a data frame of results

centrality\_df <- data.frame(Residue = colnames(residue\_data),

Betweenness\_Centrality = betweenness\_scores,

Degree\_Centrality = degree\_scores,

Weighted\_Degree\_Centrality = strength\_scores,

Score = betweenness\_score)

# Sort by highest Weighted Degree Centrality (instead of betweenness)

centrality\_df <- centrality\_df[order(-centrality\_df$Weighted\_Degree\_Centrality), ]

# Print the ranked results

print(centrality\_df)

# Save results to a CSV file

write.csv(centrality\_df, "/Users/tanerkaragol/Desktop/centrality\_scores.csv", row.names = FALSE)

# Plot network to visualize connectivity

plot(network\_graph, vertex.size = 8, vertex.label.cex = 0.8, main = "Residue Interaction Network")

# Plot bar chart of residue importance based on Weighted Degree Centrality

barplot(centrality\_df$Weighted\_Degree\_Centrality, names.arg = centrality\_df$Residue, las = 2, col = "steelblue", main = "Residue Importance by Weighted Degree Centrality")

**Graphs:**

# Load necessary libraries

if (!requireNamespace("igraph", quietly = TRUE)) install.packages("igraph")

if (!requireNamespace("ggraph", quietly = TRUE)) install.packages("ggraph")

if (!requireNamespace("tidygraph", quietly = TRUE)) install.packages("tidygraph")

library(igraph)

library(ggraph)

library(tidygraph)

library(ggplot2)

# Compute the absolute adjacency matrix from the correlation matrix

adjacency\_matrix <- abs(cor\_matrix) # Use your existing cor\_matrix

# Convert to graph

network\_graph <- graph\_from\_adjacency\_matrix(adjacency\_matrix, mode = "undirected", weighted = TRUE, diag = FALSE)

E(network\_graph)$weight <- E(network\_graph)$weight \* 5

tidy\_graph <- as\_tbl\_graph(network\_graph)

svg("/content/residue\_correlation\_network.svg", width = 4000, height = 4000)

ggraph(tidy\_graph, layout = "fr") +

geom\_edge\_link(aes(edge\_alpha = weight), color = "black", width = 1, show.legend = FALSE) +

geom\_node\_point(color = "#1f78b4", size = 7) +

geom\_node\_text(aes(label = name), repel = TRUE, size = 5.5, fontface = "bold") +

theme\_minimal() +

theme(

panel.border = element\_rect(colour = "black", fill = NA, size = 1.2), # Border around plot panel

axis.text = element\_text(face = "bold", size = 18),

axis.title = element\_text(face = "bold", size = 18),

axis.title.x = element\_blank(),

axis.title.y = element\_blank(),

axis.ticks = element\_line(color = "black"),

axis.ticks.length = unit(0.15, "cm"),

panel.grid.major = element\_line(color = "grey70", linetype = "dotted", size = 0.6),

panel.grid.minor = element\_line(color = "grey70", linetype = "dotted", size = 0.6),

plot.title = element\_text(face = "bold", size = 24, hjust = 0.5)

) +

ggtitle("Residue Correlation Network")

dev.off()

# Load required libraries

if (!requireNamespace("corrplot", quietly = TRUE)) install.packages("corrplot")

library(corrplot) # For visualizing correlation matrix

# Assuming residue conservation scores are already stored in variables like Aa, Ab, Ac, etc.

# Create a data frame from residue conservation data

residue\_data <- data.frame(Aa, Ab, Ac, Ca, Cb, Cc, Da, Db, Dc, Ea, Eb, Ec,

Fa, Fb, Fc, Ga, Gb, Gc, Ha, Hb, Hc, Ia, Ib, Ic,

Ka, Kb, Kc, La, Lb, Lc, Ma, Mb, Mc, Na, Nb, Nc,

Pa, Pb, Pc, Qa, Qb, Qc, Ra, Rb, Rc, Sa, Sb, Sc,

Ta, Tb, Tc, Va, Vb, Vc, Wa, Wb, Wc, Ya, Yb, Yc)

# Compute Pearson correlation matrix

cor\_matrix <- cor(residue\_data, method = "pearson", use = "pairwise.complete.obs")

# Apply correlation threshold (removes weak connections)

cor\_threshold <- 0.1

cor\_matrix[abs(cor\_matrix) < cor\_threshold] <- 0

# Print the correlation matrix in the console

print(cor\_matrix)

# Save the heatmap as a TIFF file

svg("/content/correlation\_matrix.svg", width = 14, height = 14)

corrplot(cor\_matrix,

method = "color",

col = colorRampPalette(c("blue", "white", "red"))(200),

tl.cex = 1.3, # Increased text label size

tl.col = "black",

tl.font = 2, # Set font to bold (2 for bold)

tl.srt = 90,

title = "",

mar = c(1, 1, 3, 1),

frame.plot = TRUE,

addgrid.col = "gray",

addshade = "all",

shade.col = "gray90",

cl.text.font = 2, # Set font to bold for color legend text

cl.cex = 1.3 # Increase size of color legend text

)

title(main = "Pearson Correlation Matrix", font.main = 2, cex.main = 2)

dev.off()