# Install necessary packages if not already installed

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

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

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

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

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

# Load required libraries

library(sf)

library(igraph)

library(visNetwork)

library(dplyr) # For data manipulation

library(lwgeom) # For geometry operations

# Read the data file

file\_path <- "/Users/dominickpasutto/Desktop/data.csv" # Replace with your actual file path

data <- read.csv(file\_path, stringsAsFactors = FALSE)

# Ensure that the 'WKT' column exists

if (!"WKT" %in% colnames(data)) {

stop("The 'WKT' column is missing from your data.")

}

# Determine the 'name' column

name\_column <- if ("name" %in% colnames(data)) "name" else "osm\_id"

# Filter rows where 'name' column is not empty and 'WKT' is valid

filtered\_data <- data %>%

filter(

!is.na(.data[[name\_column]]) & .data[[name\_column]] != "" &

!is.na(WKT) & WKT != ""

)

# Check if 'filtered\_data' is not empty

if (nrow(filtered\_data) == 0) {

stop("No data after filtering. Please check that the 'WKT' column contains valid geometries.")

}

# Parse WKT geometries to 'sfc' objects

geometries <- st\_as\_sfc(filtered\_data$WKT, crs = 4326, na.fail = FALSE)

# Remove invalid geometries

valid\_indices <- sapply(geometries, function(g) !is.null(g) && !inherits(g, "try-error"))

if (all(!valid\_indices)) {

stop("All WKT geometries are invalid. Please check your data.")

}

filtered\_data <- filtered\_data[valid\_indices, ]

geometries <- geometries[valid\_indices]

# Create 'sf' object

elements <- st\_sf(filtered\_data, geometry = geometries)

# Fix invalid geometries

elements$geometry <- st\_make\_valid(elements$geometry)

# Remove any geometries that are still invalid

valid\_indices <- st\_is\_valid(elements$geometry)

elements <- elements[valid\_indices, ]

filtered\_data <- filtered\_data[valid\_indices, ]

# Ensure unique node IDs by combining 'name' and 'osm\_id'

filtered\_data$unique\_id <- paste0(filtered\_data[[name\_column]], "\_", filtered\_data$osm\_id)

# If duplicates still exist, append row number

if (any(duplicated(filtered\_data$unique\_id))) {

filtered\_data$unique\_id <- paste0(filtered\_data$unique\_id, "\_", seq\_len(nrow(filtered\_data)))

}

# Build adjacency matrix using st\_intersects

adj\_matrix <- st\_intersects(elements, sparse = FALSE)

# Build edge list

edges\_list <- which(adj\_matrix, arr.ind = TRUE)

edges\_list <- edges\_list[edges\_list[, 1] < edges\_list[, 2], ] # Avoid duplicate edges

edges\_df <- data.frame(

from = filtered\_data$unique\_id[edges\_list[, 1]],

to = filtered\_data$unique\_id[edges\_list[, 2]],

stringsAsFactors = FALSE

)

# Prepare a nodes data frame

centroids <- st\_centroid(elements$geometry)

coords <- st\_coordinates(centroids)

nodes\_df <- data.frame(

id = filtered\_data$unique\_id,

label = filtered\_data[[name\_column]],

x = coords[, 1],

y = coords[, 2],

stringsAsFactors = FALSE

)

# Remove nodes with missing or non-finite coordinates

valid\_coords\_indices <- which(is.finite(nodes\_df$x) & is.finite(nodes\_df$y))

if (length(valid\_coords\_indices) < nrow(nodes\_df)) {

warning("Some nodes have missing or non-finite coordinates and will be removed.")

nodes\_df <- nodes\_df[valid\_coords\_indices, ]

}

# Update the edge list to include only valid nodes

valid\_node\_ids <- nodes\_df$id

edges\_df <- edges\_df %>%

filter(from %in% valid\_node\_ids & to %in% valid\_node\_ids)

# Calculate node degrees

graph <- graph\_from\_data\_frame(edges\_df, directed = FALSE)

node\_degrees <- degree(graph)

nodes\_df$degree <- node\_degrees[match(nodes\_df$id, names(node\_degrees))]

# Prepare nodes and edges for visNetwork

nodes\_vis <- nodes\_df %>%

mutate(

title = paste0("<p><b>", label, "</b><br>Degree: ", degree, "</p>"),

value = degree, # For node sizing

group = ifelse(degree > 5, "High Degree", "Low Degree") # Optional grouping

)

edges\_vis <- edges\_df

# Create the interactive network graph

visNetwork(nodes\_vis, edges\_vis, width = "100%", height = "800px") %>%

visOptions(

highlightNearest = list(enabled = TRUE, degree = 1, hover = TRUE),

nodesIdSelection = TRUE

) %>%

visInteraction(

navigationButtons = TRUE,

tooltipDelay = 0

) %>%

visLayout(randomSeed = 123) %>%

visPhysics(stabilization = TRUE) %>%

visEdges(smooth = FALSE)