**Project 3 - Report:**

**Course: CSC 3356 Data Engineering and Visualization**

**Project Number: 3**

**Submission Date: 04/15/2023**

**Al Akhawayn University in Ifrane**



**By : Adnane El Amrani & Ayoub Maimmadi**

**Spring 2024**

**Under the supervision of : Dr.Tajjeeddine Rachidi**

**Table of contents:**

1. Network Data Representation: (Network\_Data.ipynb)
2. Dataset
3. Several Network Visualization Methods
4. Definition
5. What, How, Why?
6. Source Code
7. Visualization
8. Manipulation
9. Adjacency Matrix View Visualization
10. Definition
11. What, How, Why?
12. Source Code
13. Visualization
14. Manipulation
15. Geospatial Data Representation: (Geo\_Data.ipynb)
16. Dataset
17. Visualizations
18. Definition
19. What, How, Why?
20. Source Code
21. Visualization
22. Manipulation
23. Network Data Representation:

Dataset

The dataset is an undirected network of social-political retweet interactions, capturing the dynamics within a political discourse over a specified period.

Nodes represent Twitter users, and edges symbolize retweets between these users, suggesting a flow of political information or endorsement.

**Nodes:** Each node is annotated with attributes such as the user's ID and their political alignment, classified as either 'right' (1) or 'left' (2). The political alignment serves as a label and helps in grouping and analyzing the social-political clusters or communities within the network.

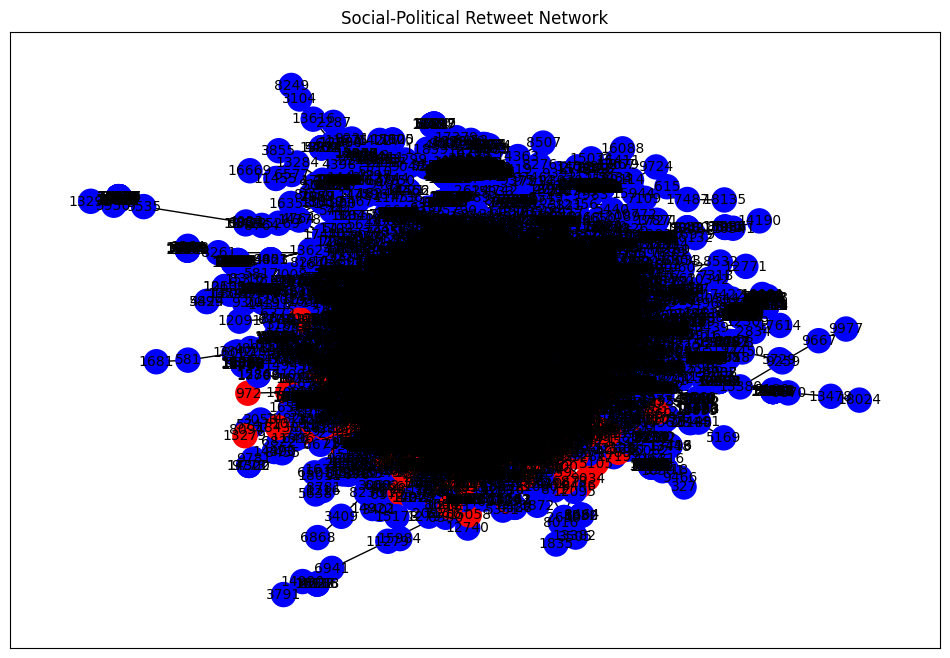
**Edges:** The interactions (edges) do not have explicit weights but are significant in studying the connectivity and influence patterns between users of differing political alignments. While the dataset does not contain additional edge attributes like color or severity, such attributes could be integrated to represent the intensity or frequency of interactions if available.

**Data Format and Handling:** The node and edge data are provided in CSV files, which are loaded and processed using Python libraries such as Pandas and NetworkX.

This choice facilitates effective data manipulation and visualization, despite not being the most efficient format for JavaScript-based w eb visualizations where JSON might be preferred.

**Visualization Intent:** In the ensuing visualizations, nodes will be color-coded based on their political alignment to visually segregate and identify clusters of right-aligned and left-aligned users. This method aims to provide clear visual insights into the polarization or integration within the retweet networks and potentially identify influential nodes within each political group.

|  |  |
| --- | --- |
| Category | Details |
| Idiom and Title of the Visualization Method | Network Visualization of the Social-Political Retweet Network |
| What? |  |
| Data Network | The data represents a social-political retweet network where nodes are Twitter users and edges represent retweets between these users. |
| Network | The network is visualized as an undirected graph, highlighting the connections between users without directional influence in the visualization. |
| How? |  |
| Encode | Visual encoding techniques used to represent the data. |
| Shape Channel | Nodes are displayed as circles, which is a common representation for nodes in network diagrams to facilitate recognition and avoid bias in shape perception. |
| Color Channel | Nodes are color-coded based on political alignment: red for 'right' (1) and blue for 'left' (2), which helps to visually distinguish between different political affiliations in the network. |
| Size Channel | All nodes are the same size in this example, though size could be varied to represent additional data such as the influence or activity level of a user. |
| Why? |  |
| Task | The primary tasks facilitated by this visualization include exploration, analysis, and understanding of the social-political interactions and clusters within the network. |
| Actions | The actions supported are identifying, comparing, and summarizing the interactions and political groupings within the network. |
| Targets | The targets of these actions are the nodes (users) and the links (retweets) between them. |
| Scale |  |
| Nodes | Represents individual Twitter users within the dataset. |
| Links | Represents retweets between users, indicating social interactions based on shared or endorsed content. |
| Node/Link Density | The density of nodes and links can provide insights into the clustering of political groups and the prevalence of interaction within and between these groups. |



Idiom and Title: Clearly defines the type of visualization and its main purpose, which is to display the network of social-political interactions.

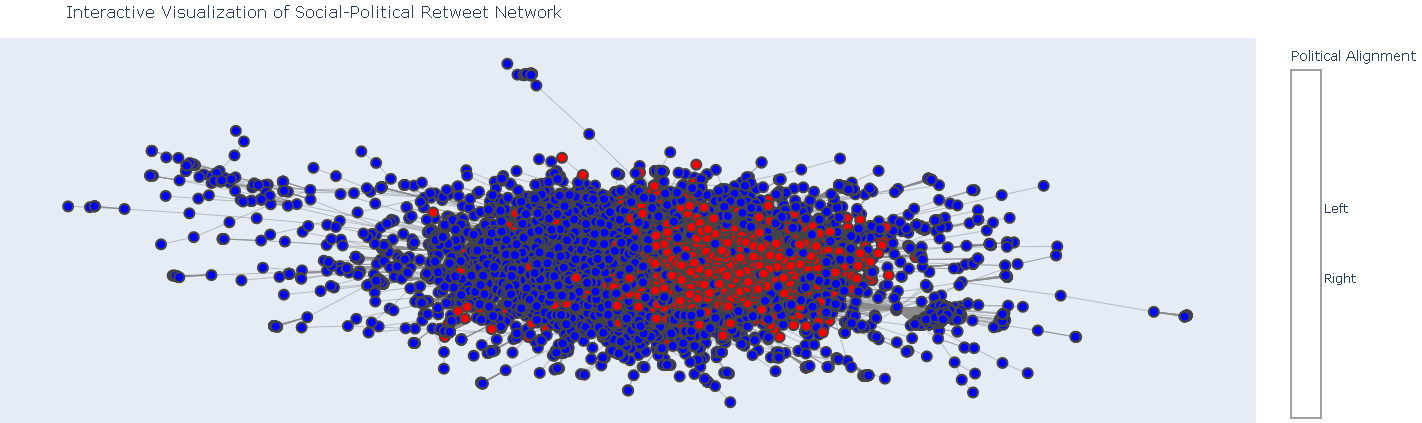
What?: Describes the nature of the data and the overall structure of the network being visualized.

How?: Breaks down the visual encoding methods used to represent the data, explaining how different attributes of the data are mapped to visual properties such as shape, color, and size.

Why?: Explains the analytical tasks that the visualization supports, which helps users understand what they can achieve or learn from it.

Scale: Details about the scale provide context on the scope of the visualization, indicating the breadth and depth of the data represented.

|  |  |
| --- | --- |
| Category | Details |
| Idiom and Title of the Visualization Method | Interactive Network Visualization of the Social-Political Retweet Network |
| What? |  |
| Data Network | This visualization represents a social-political retweet network, where nodes are Twitter users and edges signify retweets between these users, reflecting social interactions and information flow. |
| Network | The network is visualized as a dynamic graph that allows user interactions such as zooming, panning, and hovering to obtain more detailed information about each node. |
| How? |  |
| Encode | Visual encoding techniques used to represent the data dynamically. |
| Shape Channel | Nodes are represented as circular markers, a common visualization choice that avoids bias and is visually straightforward for network diagrams. |
| Color Channel | Nodes are dynamically color-coded based on their political alignment, with a colorbar indicating political affiliations (**red** for right and **blue** for left), enhancing the visual distinction between different groups. |
| Size Channel | The size of each node is uniform in this example; however, varying sizes could be used to indicate additional metrics like user influence or activity level. |
| Why? |  |
| Task | The primary tasks facilitated by this dynamic visualization include interactive exploration, detailed analysis, and enhanced understanding of complex social-political interactions within the network. |
| Actions | Supported actions include identifying connections, comparing nodes based on their attributes, and summarizing the network's structure through direct interaction. |
| Targets | The targets of these actions are the nodes and edges within the network, which represent users and their retweets, respectively. |
| Scale |  |
| Nodes | Represents individual Twitter users, visually encoded to reflect their political alignment and connectivity within the network. |
| Links | Represents retweets between users, visually encoded as lines between nodes. The discontinuity created by **None** ensures clear visualization of individual connections. |
| Node/Link Density | Visual density can provide insights into clustering and the centrality of nodes within the network, indicating regions of high activity or influence. |



Idiom and Title: Clarifies the type of visualization and its purpose, which is an interactive exploration of a network graph representing social-political dynamics on Twitter.

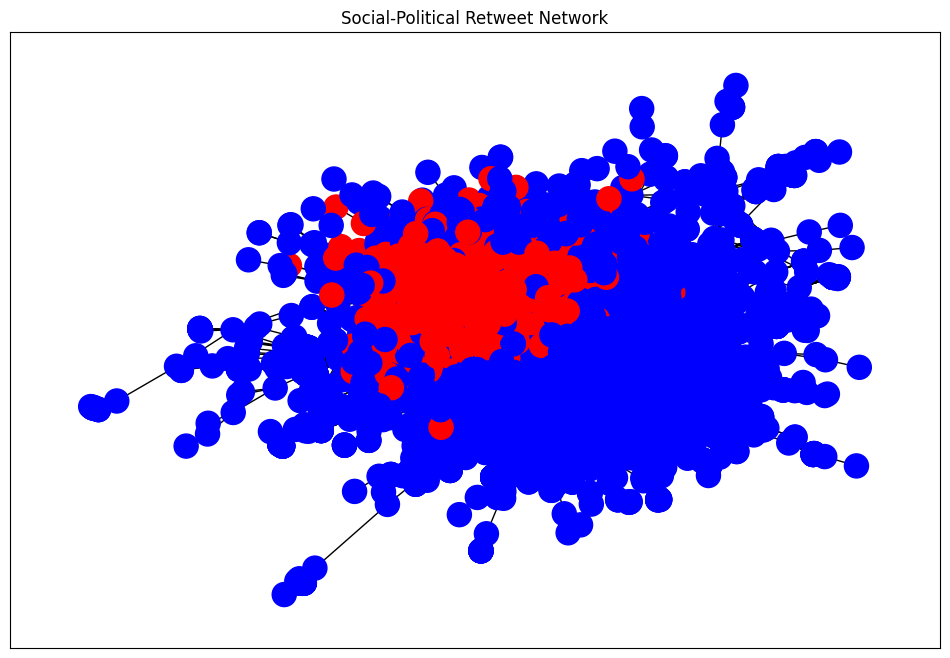
What?: Defines the data being visualized, explaining both the content and the context of the network.

How?: Describes the technical and aesthetic choices for how the data is visually represented, detailing each channel used to encode information visually.

Why?: Explains the intended use of the visualization, outlining what tasks it supports and how it helps users achieve these tasks through interaction.

Scale: Details the scope of the visualization in terms of the number of elements displayed and the level of interaction possible, giving an idea of the visualization's capacity and intended usage scale.

|  |  |
| --- | --- |
| Category | Details |
| Idiom and Title of the Visualization Method | Static Network Visualization of the Social-Political Retweet Network |
| What? |  |
| Data Network | Represents a social-political retweet network where nodes symbolize Twitter users and edges denote retweets between these users, capturing social interactions and the spread of information. |
| Network | The network is visualized as an undirected graph, which focuses on the presence of retweets between users rather than the directionality of those retweets. |
| How? |  |
| Encode | Visual encoding techniques used to represent the data in the network graph. |
| Shape Channel | Nodes are displayed as circles, which are standard for representing entities in network diagrams to facilitate easy recognition and avoid bias in shape interpretation. |
| Color Channel | Nodes are color-coded according to their political alignment: red for 'right' (label 1) and blue for 'left' (label 2). This helps to visually distinguish between different political affiliations within the network. |
| Size Channel | Node size is uniform in this visualization, emphasizing equality in representation; however, node size could vary to depict additional metrics such as influence or activity level. |
| Why? |  |
| Task | The primary tasks supported by this visualization include exploration, analysis, and understanding of social-political dynamics within the network, such as identifying clusters and key influencers. |
| Actions | The visualization enables actions like identifying direct links, comparing nodes based on attributes, and summarizing network-wide interactions. |
| Targets | The primary targets are the nodes (users) and edges (retweets), highlighting interactions and political alignments. |
| Scale |  |
| Nodes | Represents individual Twitter users as part of the dataset, each identified by a unique node. |
| Links | Represents retweets between users, which are key to understanding the flow and reach of information. |
| Node/Link Density | Visual density helps to indicate the clustering of nodes and the intensity of interactions, providing insights into the structure of the network. |



Idiom and Title: Clearly defines the type of visualization and its main purpose, which is to display the network of social-political interactions in a static format.

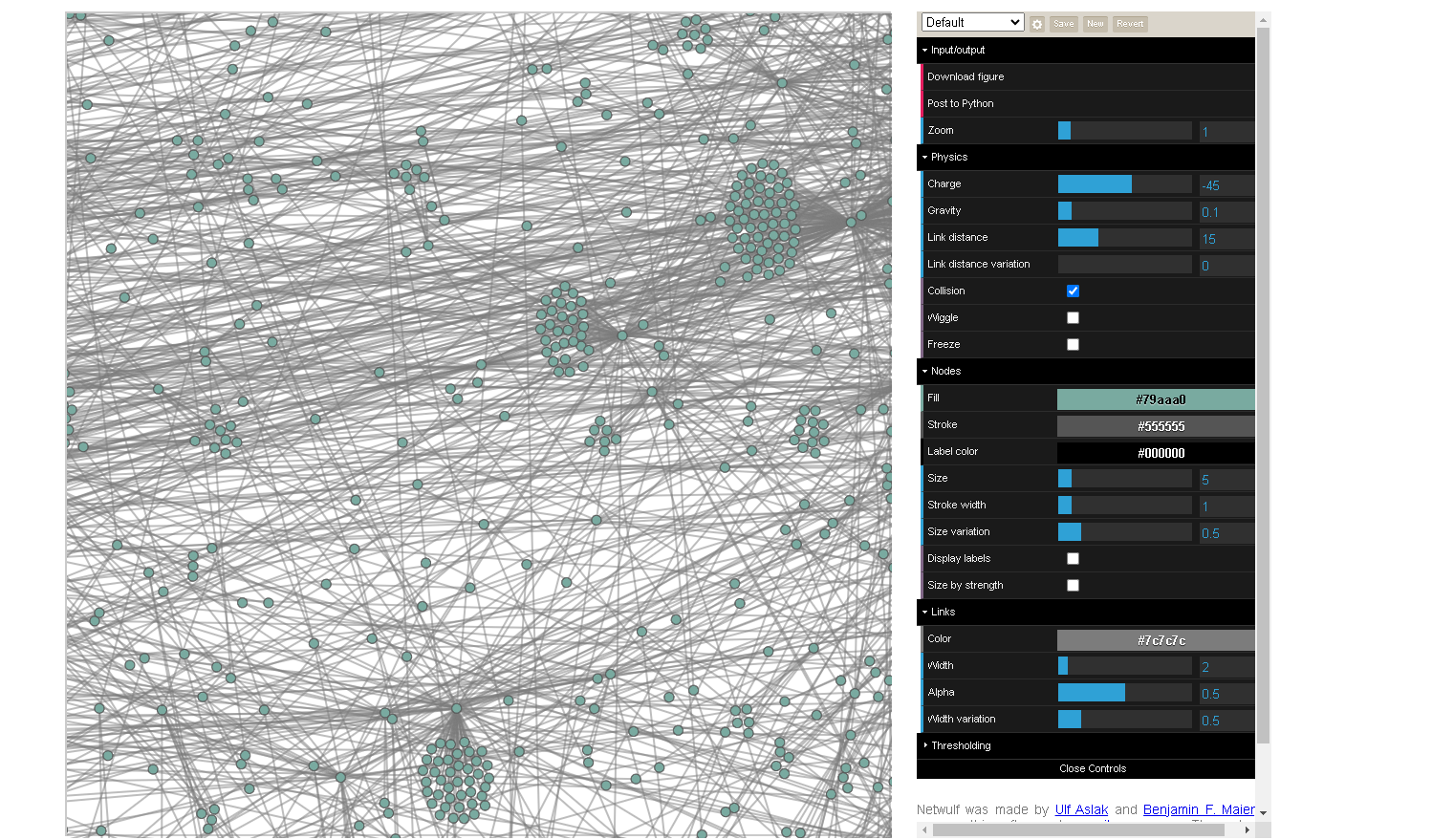
What?: Outlines the data being visualized, explaining both the content (social-political tweets) and the context (retweet interactions) of the network.

How?: Details the visual encoding methods used to represent the data, explaining how different attributes of the data are mapped to visual properties such as shape, color, and size.

Why?: Explains the analytical tasks that the visualization supports, helping users understand what they can achieve or learn from it.

Scale: Provides context on the scale of the visualization in terms of the number of elements displayed and the density of these elements, giving insights into the complexity and depth of the data represented.

|  |  |
| --- | --- |
| Category | Details |
| Idiom and Title of the Visualization Method | Dynamic and Interactive Network Visualization using Netwulf and Plotly |
| What? |  |
| Data Network | This visualization represents a social-political network where nodes symbolize Twitter users and edges denote retweets between these users, capturing the flow of information and interactions. |
| Network | The network is visualized dynamically, allowing users to interact with the data such as panning, zooming, and hovering to gain more detailed insights. |
| How? |  |
| Encode | Visual encoding techniques used to represent the data in an interactive format. |
| Shape Channel | Nodes are displayed as circular markers, a common visualization choice that facilitates easy recognition and avoids bias in shape interpretation. |
| Color Channel | Nodes are color-coded based on their political alignment using a gradient color scale, enhancing the visual distinction between different political affiliations within the network. |
| Size Channel | Node size is uniform in this example; however, sizes can vary to represent additional metrics such as influence or activity level. |
| Why? |  |
| Task | The primary tasks supported by this visualization include interactive exploration, detailed analysis, and enhanced understanding of complex social-political interactions within the network. |
| Actions | Supported actions include identifying connections, comparing nodes based on attributes, and summarizing network-wide interactions through direct interaction. |
| Targets | The primary targets are the nodes (users) and edges (retweets), highlighting interactions and political alignments. |
| Scale |  |
| Nodes | Represents individual Twitter users, visually encoded to reflect their political alignment and connectivity within the network. |
| Links | Represents retweets between users, visually encoded as lines between nodes. The discontinuity created by **None** ensures clear visualization of individual connections. |
| Node/Link Density | Visual density can provide insights into clustering and the centrality of nodes within the network, indicating regions of high activity or influence. |



Idiom and Title: Clearly defines the type of visualization and its main purpose, which is to provide a dynamic and interactive exploration of a network graph representing social-political dynamics on Twitter.

What?: Outlines the data being visualized, explaining both the content (social-political tweets) and the context (retweet interactions) of the network.

How?: Details the technical and aesthetic choices for how the data is visually represented, explaining how different attributes of the data are mapped to visual properties such as shape, color, and size.

Why?: Explains the analytical tasks that the visualization supports, helping users understand what they can achieve or learn from it.

Scale: Provides context on the scale of the visualization in terms of the number of elements displayed and the level of interaction possible, giving insights into the complexity and depth of the data represented.

|  |  |
| --- | --- |
| Category | Details |
| Idiom and Title of the Visualization Method | Sparse Matrix Visualization of Adjacency Matrix |
| What? |  |
| Data Network | The visualization shows the adjacency matrix of a network where nodes could represent entities such as users, and edges represent connections such as retweets or interactions between them. This approach focuses on the presence and absence of direct links between nodes. |
| Network | The network's adjacency matrix is visualized in a sparse format, highlighting non-zero entries that indicate direct connections between nodes. This method is particularly useful for large graphs to visualize the structure efficiently. |
| How? |  |
| Encode | Visual encoding techniques used to represent the adjacency matrix data. |
| Shape Channel | Non-zero entries in the adjacency matrix are represented as square markers. This shape is chosen for its clarity and effective use in grid-based visualizations. |
| Color Channel | The color 'blue' is used for the markers, which denotes the presence of an edge between nodes in the adjacency matrix. This color choice helps the markers stand out against typically white backgrounds of plots. |
| Size Channel | Marker size is set uniformly small (**ms=1**) to handle high-density areas in large graphs, ensuring that the plot remains readable and not overly cluttered. |
| Why? |  |
| Task | The primary task facilitated by this visualization is to analyze the network structure, identify connectivity patterns, and detect clusters or isolated nodes. |
| Actions | The visualization supports the actions of identifying and comparing connection patterns across different nodes in the network. |
| Targets | The primary targets are the nodes and their connections as represented by the adjacency matrix entries. This visualization is particularly adept at showcasing which nodes are directly connected. |
| Scale |  |
| Nodes | Represents all nodes in the network as indices on the axes of the matrix. |
| Links | Represents direct links between nodes as non-zero entries in the matrix. |
| Node/Link Density | Visual density is effectively communicated through the distribution of markers in the plot, revealing areas of high and low connectivity. |

Adjacency Matrix: The adjacency matrix is generated using nx.adjacency\_matrix(G), which returns a sparse matrix representation of the node connectivity in the graph G. The matrix is converted to a dense array format for easy visualization with toarray() method.

Visualization with Seaborn: seaborn.heatmap() is used to plot the adjacency matrix. The heatmap visually represents the matrix with colors indicating the presence of edges between nodes. The parameters:

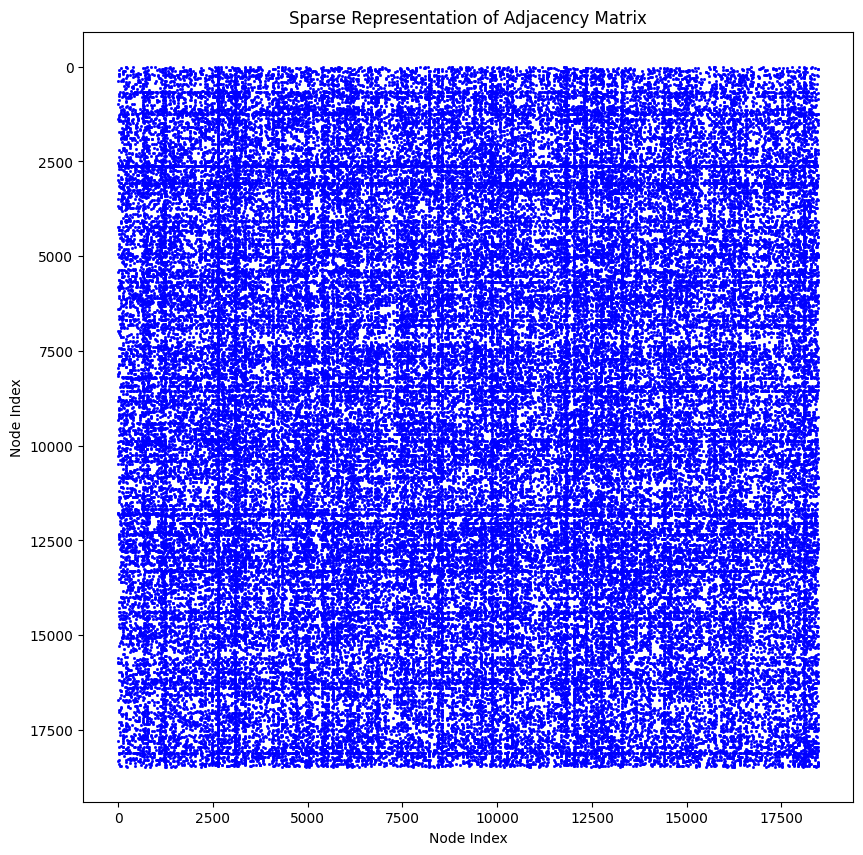
annot=False ensures that the values (0s and 1s) are not printed inside the squares.

cmap="Blues" uses a blue color scale to indicate the presence of an edge.

cbar=False disables the color bar on the side since the data is binary.

square=True makes each cell in the heatmap square-shaped.

Plot Customization: Labels and titles are added to make the plot more informative. The x and y labels represent the node indices, which correspond to the indices of the nodes in the graph as used in the adjacency matrix.



Idiom and Title: Clearly defines the visualization method and its primary purpose, which in this case, is to provide a clear and efficient view of the adjacency matrix of a potentially large network.

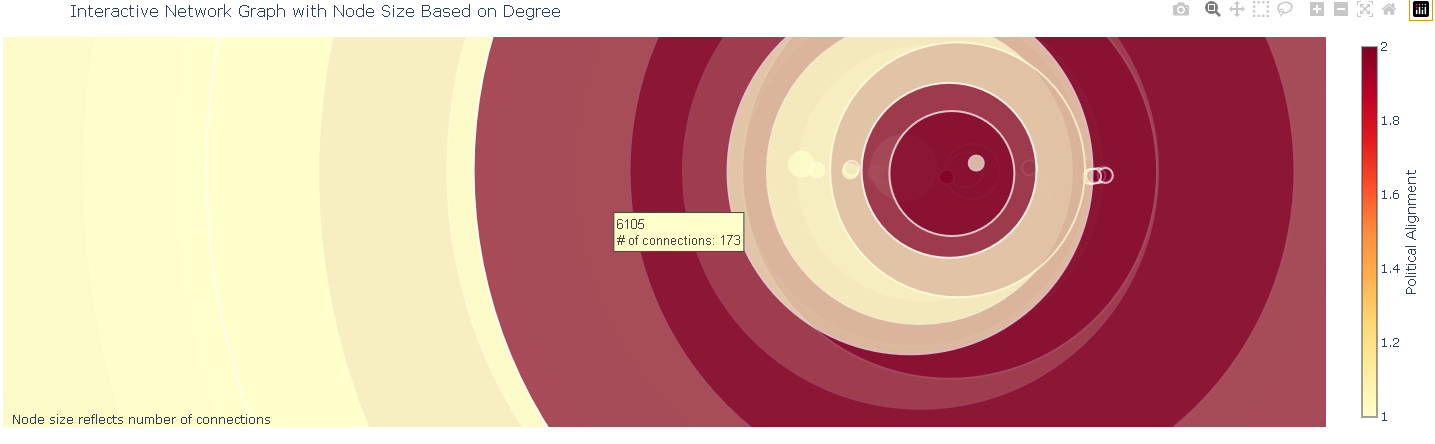
What?: Describes the data being visualized—the network's adjacency matrix—and the context within which it is being examined.

How?: Details the visual encoding methods used, explaining how different attributes of the adjacency matrix data are mapped to visual properties such as shape, color, and size.

Why?: Explains the analytical tasks that the visualization supports, helping users understand what insights can be gained from examining the matrix.

Scale: Provides context on the scale of the visualization in terms of the number of elements displayed, giving insights into the complexity and depth of the network's structure as represented by the adjacency matrix.

|  |  |
| --- | --- |
| Category | Details |
| Idiom and Title of the Visualization Method | Interactive Network Graph with Node Size Based on Degree |
| What? |  |
| Data Network | The visualization represents a network where nodes are entities such as users, locations, or items, and edges signify relationships or interactions such as links or communications between these entities. |
| Network | The network is visualized dynamically, allowing users to interact through zooming, panning, and hovering for more detailed information on nodes and connections. |
| How? |  |
| Encode | Visual encoding techniques used to represent the data dynamically. |
| Shape Channel | Nodes are represented using circular markers, commonly utilized in network diagrams for easy recognition and to avoid bias in shape perception. |
| Color Channel | Nodes are dynamically color-coded based on their political alignment, enhancing the visual distinction between groups or affiliations within the network. |
| Size Channel | Node size varies based on the degree (number of connections) of each node, emphasizing nodes with higher connectivity and influence within the network. |
| Why? |  |
| Task | The primary tasks supported by this visualization include interactive exploration, detailed analysis, and understanding the relative importance of nodes based on their connectivity within the network. |
| Actions | Supported actions include identifying key nodes, comparing nodes based on their degree, and analyzing the overall structure of the network. |
| Targets | The primary targets are the nodes and their connections. The visualization aims to highlight nodes with significant connections and detail their roles within the network structure. |
| Scale |  |
| Nodes | Represents individual entities within the network, each marked with unique identifiers and additional connectivity information accessible via hovering. |
| Links | Represents interactions or relationships between entities, visually encoded as lines connecting nodes. |
| Node/Link Density | The visual density and distribution of nodes can provide insights into clustering and the centralization of connections within the network, crucial for understanding network dynamics. |



**Visualizations all available within the video and the notebook**

**Examples of manipulations:**

* Data Loading: The script starts by loading node and edge data from CSV files.
* Graph Creation: A graph is created using NetworkX from the edge list.
* Filtering Nodes: Nodes are filtered based on their degree to focus the visualization on highly connected nodes.
* Aggregating Edge Data: The edge data is aggregated to count multiple interactions between the same nodes, emphasizing more frequent interactions.
* Visualization: The graph is visualized using Plotly, with edge thickness representing the frequency of interactions (retweets) and node size based on the node degree. This helps to visualize both the structure of the network and the intensity of interactions.

import pandas as pd

import networkx as nx

import plotly.graph\_objects as go

# Load your data (assuming paths are set and files are formatted correctly)

node\_labels\_df = pd.read\_csv('path\_to\_node\_labels.csv', header=None, names=['node', 'label'])

edges\_df = pd.read\_csv('path\_to\_edges.csv', header=None, names=['source', 'target', 'timestamp'])

# Create a graph

G = nx.from\_pandas\_edgelist(edges\_df, 'source', 'target', create\_using=nx.Graph())

# Add node attributes for labels

node\_labels = pd.Series(node\_labels\_df.label.values, index=node\_labels\_df.node).to\_dict()

nx.set\_node\_attributes(G, node\_labels, 'label')

# Manipulation 1: Filter nodes by degree

min\_degree = 10 # Threshold for degree

nodes\_with\_high\_degree = [node for node, degree in G.degree() if degree >= min\_degree]

subgraph = G.subgraph(nodes\_with\_high\_degree)

# Manipulation 2: Aggregate edges to count retweets between nodes

# This step assumes edges\_df has multiple entries for the same source-target pairs

edge\_count\_df = edges\_df.groupby(['source', 'target']).size().reset\_index(name='weight')

# Create a new graph with this aggregated information

weighted\_graph = nx.from\_pandas\_edgelist(edge\_count\_df, 'source', 'target', edge\_attr='weight', create\_using=nx.Graph())

# Calculate positions using a layout algorithm

pos = nx.spring\_layout(weighted\_graph)

# Prepare Plotly traces

edge\_trace = []

for edge in weighted\_graph.edges(data=True):

x0, y0 = pos[edge[0]]

x1, y1 = pos[edge[1]]

weight = edge[2]['weight']

edge\_trace.append(go.Scatter(x=[x0, x1, None], y=[y0, y1, None],

line=dict(width=weight/5, color='grey'),

hoverinfo='none',

mode='lines'))

node\_trace = go.Scatter(

x=[pos[node][0] for node in weighted\_graph],

y=[pos[node][1] for node in weighted\_graph],

text=[f'{node} (connections: {weighted\_graph.degree(node)})' for node in weighted\_graph],

mode='markers',

hoverinfo='text',

marker=dict(

showscale=True,

colorscale='YlGnBu',

size=[5 \* weighted\_graph.degree(node) for node in weighted\_graph],

color=[weighted\_graph.nodes[node]['label'] for node in weighted\_graph],

colorbar=dict(title='Political Alignment'),

line\_width=2))

# Define the layout of the plot

fig = go.Figure(data=edge\_trace + [node\_trace],

layout=go.Layout(

title='Network Graph of Social-Political Retweet Data',

showlegend=False,

hovermode='closest',

margin=dict(b=20, l=5, r=5, t=40),

xaxis=dict(showgrid=False, zeroline=False, showticklabels=False),

yaxis=dict(showgrid=False, zeroline=False, showticklabels=False)

))

# Show the plot

fig.show()

1. Geospatial Data Representation:

**Dataset:**

**Basic Information**

**Name:** ParkingAreas

**Title:** Parking Areas

**Source File Name:** rpkarea

**Keywords:** Parking, Parking Lot, Transportation, Infrastructure

**Metadata and Usage**

Date Last Modified: July 22, 2016

Published Date: July 22, 2016

Access Level: Public

**Summary Description:** The dataset consists of geospatial data representing various types of parking areas within Bloomington, including paved and gravel parking lots, as well as "non-parking island features". These islands are typically spaces within or adjacent to parking areas that are not used for parking, often landscaped or used for other infrastructure elements like light poles or signage.

**Technical Details**

**Use:** This dataset is a crucial part of the City of Bloomington's Geographic Information System (GIS), particularly as a base map layer. It's used for general GIS applications such as viewing, querying, and producing map outputs. The data also plays a role in calculating impervious surfaces for stormwater management.

**Supplemental Information:** The native data format is GenaMap GIS Vector (4).

**Coordinate Reference System:** North American Datum of 1983 (NAD83), Indiana State Plane West Zone 1302, measured in US Survey Feet.

* EPSG Code: 2966
* Extent: Specifically covers the Bloomington GIS mapped area.
* Bounding Coordinates: 3078000.000000, 1392000.000000, 3162000.000000, 1470000.000000

**Source and Quality**

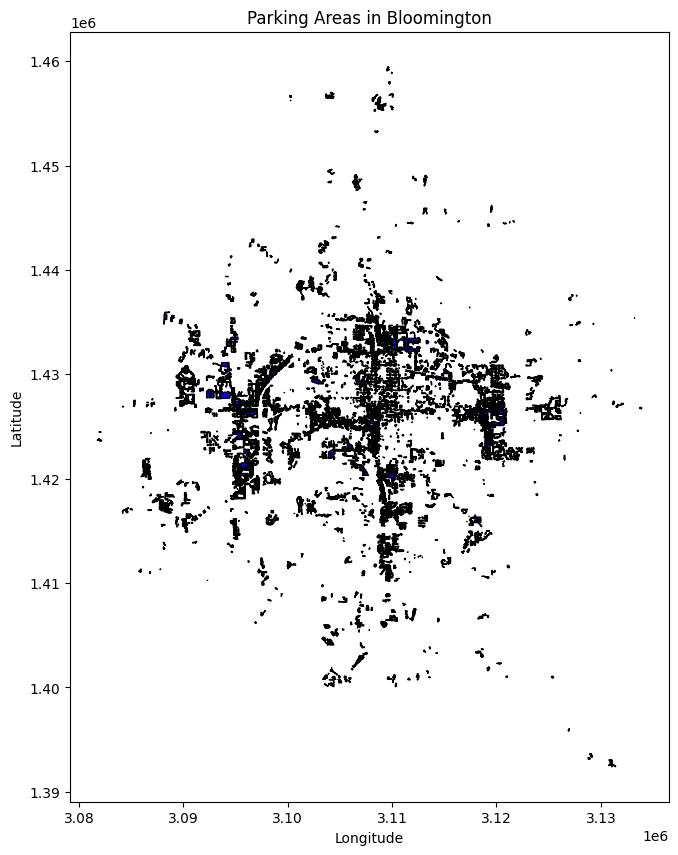
* Source Data: Derived and updated from orthophotography taken during multiple years (1991-92, 1998, 2005, 2006, 2010, 2011, 2014), along with updates from plat or development plans.
* Quality Rating: Good
* Completeness: Complete
* Accuracy: The data from 2006 and earlier are photogrammetrically created and adhere to National Map Accuracy Standards with an accuracy of +/-2.5 US feet. Features added later were digitized based on the best available orthophotos.

**Maintenance and Contact Information**

* Status: Current
* Maintenance: Regularly maintained by Information Technology Services (ITS) and City of Bloomington Utilities (CBU) GIS staff, following any new construction or significant changes in the urban landscape.
* Attributes: Includes a unique ID key for each feature.
* Feature Type: Area (3606 features)
* URL: City of Bloomington Maps
* Contact Information: GIS Manager at Bloomington ITS Department
* Email: gis@bloomington.in.gov
* Phone: 812-349-3454 (helpdesk)
* Address: 401 N Morton Street, PO Box 100, Bloomington, IN 47402-0100

This dataset serves as a foundational element in various city-wide GIS analyses and is indispensable for urban planning, infrastructure management, and environmental assessments in Bloomington.

|  |  |
| --- | --- |
| Category | Description |
| Idiom and Title | **Static Map Visualization: "Parking Areas in Bloomington"** |
| What? | **Geospatial representation of parking areas.** |
| Data Network | **GeoJSON data representing various parking areas.** |
| Network | **Not applicable (N/A) as this is not a network-based visualization.** |
| How? | **Loading data with Geopandas, fixing geometries, simplifying data for performance, and visualizing with Matplotlib.** |
| Encode | **Mapping of geographical data points to visual elements.** |
| Shape Channel | **Geometries of the parking areas (Polygon shapes).** |
| Color Channel | **Blue for parking areas, black for edges.** |
| Size Channel | **N/A - Size is intrinsic to the area each parking occupies.** |
| Why? | **To visualize the spatial distribution and scope of parking infrastructure in Bloomington.** |
| Task | **Allow users to identify locations and extents of parking areas.** |
| Actions | **Viewing and interpreting the static map.** |
| Targets | **Users in urban planning, public works, or local government.** |
| Scale | **Fixed scale based on plot dimensions and GeoJSON data extents.** |

****

Idiom and Title: Specifies the type and purpose of the visualization.

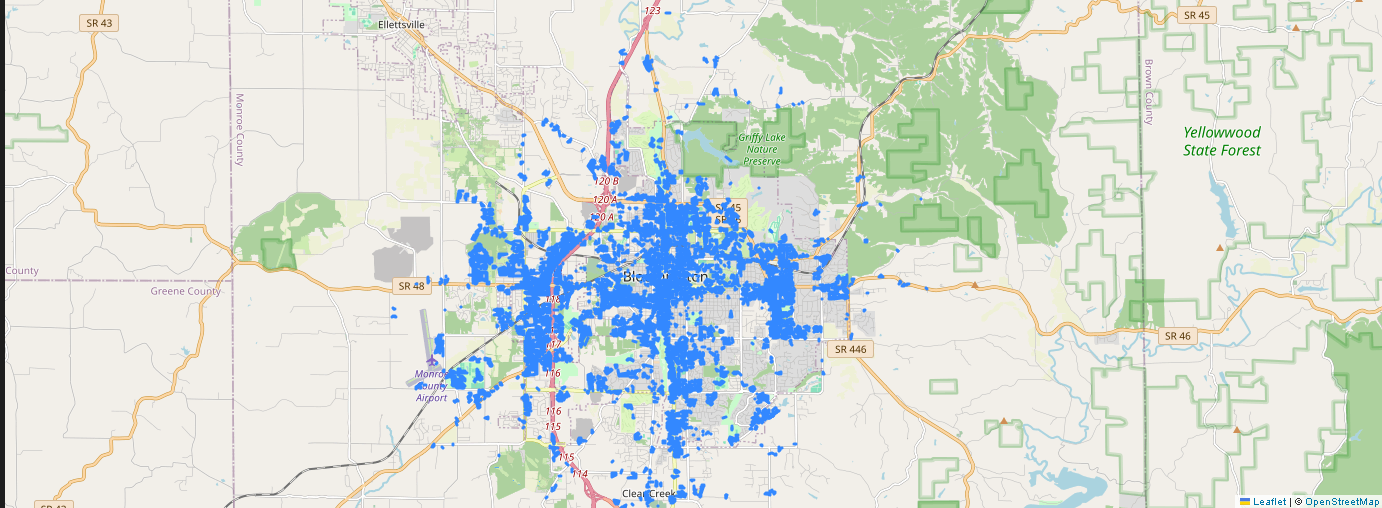
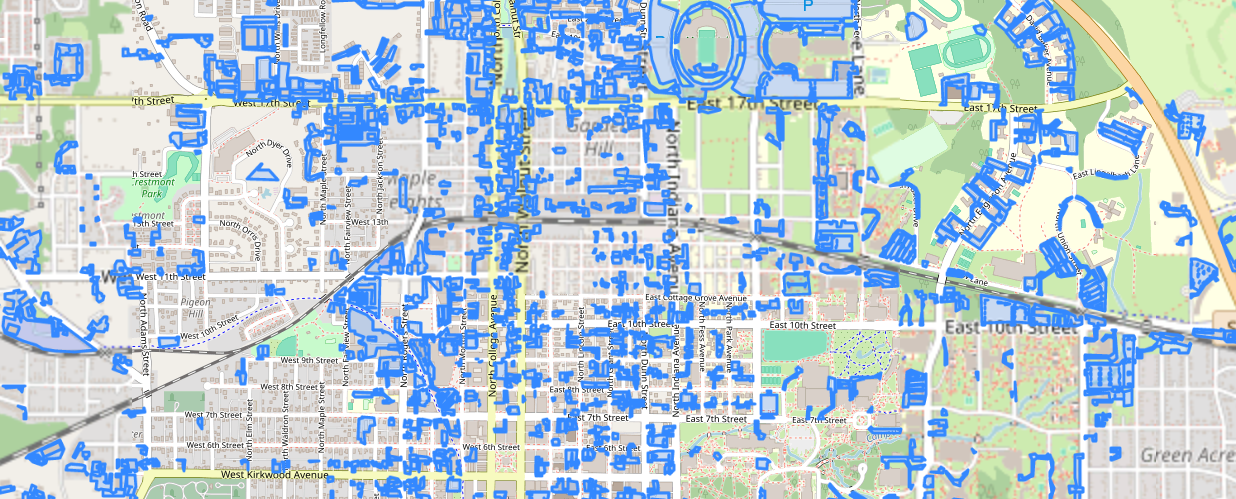
What and How: Describes what the data represents and how the visualization is produced.

Encode and Channels: Details on how data attributes are represented visually (shapes, colors, sizes).

Why, Task, Actions, and Targets: Explains the purpose of the visualization, what it aims to achieve, how users interact with it, and who the intended audience is.

Scale, Nodes, Links, Node/Link Density: Discusses the scale and notes that network-related terms are not applicable as this visualization focuses on spatial data of physical locations rather than connections or relationships.

|  |  |
| --- | --- |
| Category | Description |
| Idiom and Title | **Interactive Web Map Visualization: "Parking Areas in Bloomington"** |
| What? | **Interactive geospatial representation of parking areas.** |
| Data Network | **GeoJSON data containing geographic details of parking areas.** |
| Network | **N/A - This is not a network graph visualization.** |
| How? | **Loading and processing GeoJSON data with Geopandas; using Folium to create an interactive map.** |
| Encode | **Assigning geographical data to visual and interactive map elements.** |
| Shape Channel | **Represented by the polygons or multipolygons of parking areas.** |
| Color Channel | **Default or customized color for parking area polygons.** |
| Size Channel | **The size or scale of shapes is based on real-world proportions translated onto the map.** |
| Why? | **To enable users to interactively explore and visualize the distribution and details of parking areas in Bloomington.** |
| Task | **Users can interact with the map to gain insights about parking logistics and planning.** |
| Actions | **Users can zoom in/out, pan, and view specific details about parking areas on click.** |
| Targets | **Urban planners, residents, visitors, and municipal authorities.** |
| Scale | **Interactive scale that adjusts as the user zooms in and out.** |

**** ****

Idiom and Title: Defines the type of visualization and its main focus.

What and How: Explains the data representation and the technical methods used to create the visualization.

Encode and Channels: Discusses how data attributes are visually encoded using shapes, colors, and sizes.

Why, Task, Actions, and Targets: Describes the purpose of the visualization, the user interactions it supports, the actions it allows, and its intended audience.

Scale, Nodes, Links, Node/Link Density: Provides details on the map's scalability and clarifies that terms related to network analysis do not apply to this visualization.

**Example of manipulations:**

**load\_data: Loads GeoJSON data using Geopandas.**

**filter\_data: Filters the GeoDataFrame based on a given condition, such as selecting only 'Paved' surface types.**

**aggregate\_data: Aggregates the data to count how many parking areas exist for each type of surface.**

**add\_area\_column: Adds a new column to the GeoDataFrame calculating the area of each parking lot.**

**visualize\_data: Creates a bar chart to visualize the aggregated data, helping to interpret patterns like the prevalence of different surface types.**

**main: Orchestrates the loading, manipulating, and visualizing of data.**

import geopandas as gpd

import matplotlib.pyplot as plt

def load\_data(file\_path):

# Load the GeoJSON file

return gpd.read\_file(file\_path)

def filter\_data(gdf, condition):

# Filter the data based on a condition

return gdf[condition]

def aggregate\_data(gdf, column):

# Aggregate data by a specific column

return gdf.groupby(column).size()

def add\_area\_column(gdf):

# Calculate the area of each polygon and add it as a new column

gdf['area'] = gdf.geometry.area

return gdf

def visualize\_data(gdf, title, column\_to\_plot):

# Create a bar chart of the data

plt.figure(figsize=(10, 6))

gdf[column\_to\_plot].plot(kind='bar', color='skyblue')

plt.title(title)

plt.xlabel('Category')

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

def main():

file\_path = 'path\_to\_your\_geojson\_file.geojson'

gdf = load\_data(file\_path)

# Example manipulation: Filter only paved parking areas

paved\_parking = filter\_data(gdf, gdf['surface\_type'] == 'Paved')

print("Filtered paved parking areas:")

print(paved\_parking)

# Aggregate by surface type

parking\_counts = aggregate\_data(gdf, 'surface\_type')

print("Parking counts by surface type:")

print(parking\_counts)

# Add area column

gdf = add\_area\_column(gdf)

print("Data with added area column:")

print(gdf[['surface\_type', 'area']].head())

# Visualize aggregated data

visualize\_data(parking\_counts.reset\_index(name='Count'), 'Parking Counts by Surface Type', 'Count')

if \_\_name\_\_ == '\_\_main\_\_':

main()

* print\_columns function: This new function prints all column names in the dataset, which will help you identify the right columns to use for filtering and aggregating.
* Generalizing filter and aggregation functions: The filter function now accepts any column and condition as arguments, making it more versatile. Ensure you replace 'parking\_type' and 'Public' with the appropriate column and value based on your data.

import geopandas as gpd

import matplotlib.pyplot as plt

def load\_data(file\_path):

# Load the GeoJSON file

return gpd.read\_file(file\_path)

def print\_columns(gdf):

# Print column names

print("Available columns in the dataset:")

print(gdf.columns)

def filter\_data(gdf, column, condition):

# Filter the data based on a condition

return gdf[gdf[column] == condition]

def aggregate\_data(gdf, column):

# Aggregate data by a specific column

return gdf.groupby(column).size()

def add\_area\_column(gdf):

# Calculate the area of each polygon and add it as a new column

gdf['area'] = gdf.geometry.area

return gdf

def visualize\_data(gdf, title, column\_to\_plot):

# Create a bar chart of the data

plt.figure(figsize=(10, 6))

gdf[column\_to\_plot].plot(kind='bar', color='skyblue')

plt.title(title)

plt.xlabel('Category')

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

def main():

file\_path = 'path\_to\_your\_geojson\_file.geojson'

gdf = load\_data(file\_path)

# Print column names to identify the correct column for filtering

print\_columns(gdf)

# Assuming we identify the correct column from the output, use it below

# Example manipulation: Filter a column found e.g., 'parking\_type'

filtered\_parking = filter\_data(gdf, 'parking\_type', 'Public')

print("Filtered data based on parking type 'Public':")

print(filtered\_parking)

# Aggregate by a valid column name you've verified

parking\_counts = aggregate\_data(gdf, 'parking\_type')

print("Parking counts by type:")

print(parking\_counts)

# Add area column

gdf = add\_area\_column(gdf)

print("Data with added area column:")

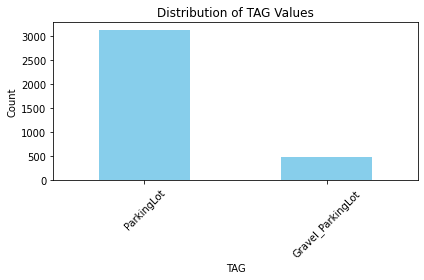
print(gdf[['parking\_type', 'area']].head())

# Visualize aggregated data

visualize\_data(parking\_counts.reset\_index(name='Count'), 'Parking Counts by Type', 'Count')

if \_\_name\_\_ == '\_\_main\_\_':

main()



Another manipulation :

* Calculates the area for each parking area.
* Simplifies the geometries to improve processing performance, especially useful for complex or detailed polygons.
* Classifies parking areas based on size (small, medium, large) after calculating the areas.

Blue: Represents Small parking areas. These are areas less than 0.1 square kilometers.

Green: Represents Medium parking areas. These areas are between 0.1 and 1 square kilometer.

Red: Represents Large parking areas. These areas are greater than 1 square kilometer.

**Color-Coded Visualization:**

The color-coding in the map helps to quickly differentiate between various sizes of parking areas. This visual distinction makes it easier for viewers to identify and understand the spatial distribution of small, medium, and large parking areas within the geographic context of the data.

**Purpose of Color-Coding:**

The choice of colors (blue, green, red) is designed to provide a clear and intuitive understanding of the scale of parking areas:

Blue is often associated with smaller, more contained features.

Green typically suggests an intermediate level, indicating areas that are more significant than the smallest but not the largest.

Red is used to highlight the largest areas, drawing attention to major parking facilities.

This method of visualization enhances the map's readability and enables stakeholders, such as urban planners or municipal authorities, to make informed decisions based on the spatial analysis of parking infrastructure. It also supports quick visual assessments of the data, which is beneficial in presentations or reports where rapid comprehension is necessary.

* Area Calculation: Converts the CRS to a projection suitable for area calculation (EPSG:3857) and calculates the area in square kilometers.
* Geometry Simplification: Simplifies complex polygon geometries to enhance performance, which is helpful for visualization and spatial analysis.
* Classification by Area: Defines parking areas as small, medium, or large based on their size. This classification helps in analyzing and visualizing spatial distribution effectively.
* Visualization: Uses matplotlib to create a color-coded map based on the size category of each parking area.

import geopandas as gpd

import matplotlib.pyplot as plt

import pandas as pd

def load\_data(file\_path):

# Load the GeoJSON file

return gpd.read\_file(file\_path)

def calculate\_area(gdf):

# Calculate the area of each polygon in square meters (adjust the CRS if necessary)

gdf['area'] = gdf.to\_crs(epsg=3857).geometry.area / 10\*\*6 # converting to square kilometers

return gdf

def simplify\_geometries(gdf, tolerance=0.001):

# Simplify the geometries to improve performance

gdf['simplified\_geometry'] = gdf.geometry.simplify(tolerance)

return gdf

def classify\_areas(gdf):

# Classify areas based on their size

bins = [0, 0.1, 1, float('inf')] # thresholds in square kilometers

labels = ['Small', 'Medium', 'Large']

gdf['size\_category'] = pd.cut(gdf['area'], bins=bins, labels=labels)

return gdf

def visualize\_data(gdf):

# Create a color-coded plot based on area size category

fig, ax = plt.subplots(1, 1, figsize=(10, 10))

colors = {'Small': 'blue', 'Medium': 'green', 'Large': 'red'}

for category, color in colors.items():

gdf[gdf['size\_category'] == category].plot(ax=ax, color=color, label=category)

ax.legend(title='Parking Area Size')

ax.set\_title('Classified Parking Areas by Size')

ax.set\_xlabel('Longitude')

ax.set\_ylabel('Latitude')

plt.show()

def main():

file\_path = 'C:\\Users\\LENOVO\\Desktop\\CSC3356\_Project3\\parking-areas-gis-data-geojson-4.geojson'

gdf = load\_data(file\_path)

gdf = calculate\_area(gdf)

gdf = simplify\_geometries(gdf)

gdf = classify\_areas(gdf)

print("Preview of the data with added columns:")

print(gdf[['id', 'TAG', 'area', 'size\_category']].head())

visualize\_data(gdf)

if \_\_name\_\_ == '\_\_main\_\_':

main()

A screenshot of a computer screen

Description automatically generated