Final Project Report

Qingyuan Feng

qf2093

qf2093@nyu.edu

Abstract

This project conducts a detailed analysis of the New York City subway system's resilience to extreme weather events, with a particular focus on the centrality measures of degree, betweenness, and closeness across subway stations. The research identifies critical nodes and paths within the network that are vital for maintaining functionality during disruptions, thus aiding in effective recovery and future infrastructure planning. By mapping the centrality metrics, the study provides insights into the structural importance of each subway station, facilitating targeted resilience strategies to enhance the overall robustness of the transportation network against future adversities.

1. Introduction

Following the devastating impact of a hurricane, the New York City subway system faced unprecedented disruptions, with numerous stations flooded and key transit lines out of service. This event underscored the subway system's susceptibility to extreme weather and the urgent need for robust infrastructure resilience. In light of these challenges, the analysis of centrality measures—degree, betweenness, and closeness—becomes essential. Degree centrality identifies the most crucial nodes for restoring operational integrity, focusing repair efforts where they can reestablish the most connections. Betweenness centrality assesses stations' critical roles in passenger flow, pinpointing where service restorations would mitigate the most disruption and reveal the extent of system-wide damage. Finally, closeness centrality highlights the accessibility of stations, guiding enhancements to ensure that no area remains isolated during recovery efforts. This analysis not only aids in the immediate recovery but also informs long-term strategies to fortify the subway against similar future catastrophes, providing the MTA and city residents with the knowledge needed to enhance system resilience and reliability.

2. Results and Demonstration

In the original graph done by Matplotlib, the heat map background stands for the population density in each zip code, nodes stand for 494 subway stations in NYC, and lines between nodes stand for the connections between them. Heat map color path by 'OrRd' and alpha = 0.5. There is no interaction applied on the graph, but this graph compares the densities of subway stations and population.

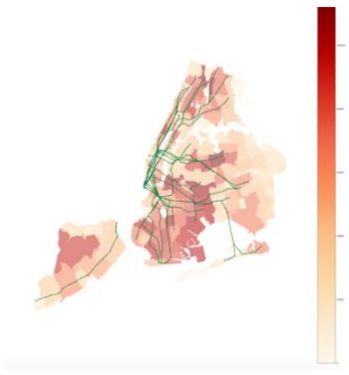
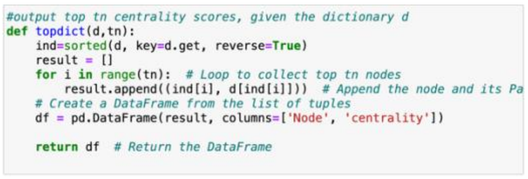


Figure 1. Heat map by population by zip code stacked with subway lines

After plotting out the basic graph, it is naturally curious about the centrality of the station. Centrality finds out the most important station in various scales, and here, I will be using the degrees, betweenness, and closeness as the centrality matrix.

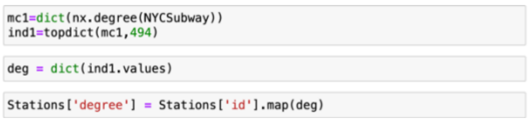


```
#output top tn centrality scores, given the dictionary d
def topdict(d,tn):
    ind=sorted(d, key=d.get, reverse=True)
    result = []
    for i in range(tn):  # Loop to collect top tn nodes
        result.append((ind[i], d[ind[i]]))  # Append the node and its Pa
    # Create a DataFrame from the list of tuples
    df = pd.DataFrame(result, columns=['Node', 'centrality'])

    return df  # Return the DataFrame
```

Figure 2. def for generating the table of centrality score for future node size

Degree of each station: Degree stands for the number of lines passing through each station, and more degree stands for more convenience for travelers' connections between subway lines.



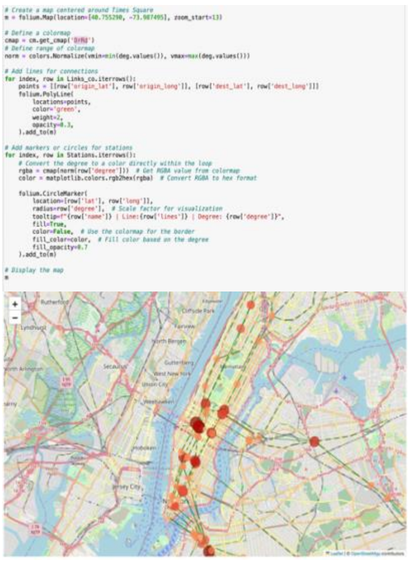
```
mc1=dict(nx.degree(NYCSubway))
ind1=topdict(mc1,494)
```

```
deg = dict(ind1.values)
```

```
Stations['degree'] = Stations['id'].map(deg)
```

Figure 3. Codes for finding degrees and storing the degree score for plotting node size and colors

Plotting for the degrees of each subway station using the folium package. The size and color of nodes are fixed by the table of degree table we generated before, using color path 'OrRd' for the node's colors to illustrate the degree of the nodes. The interactions: Zoom, mouse over each node (subway station name, lines passing through, and degree).



```
# Create a map centered around Times Square
m = folium.Map(location=[40.755290, -73.987495], zoom_start=13)

# Define a colormap
cmap = cm.get_cmap('OrRd')
# Define range of colormap
norm = colors.Normalize(vmin=min(deg.values()), vmax=max(deg.values()))

# Add lines for connections
for index, row in Links_co.iterrows():
    points = [[row['origin_lat'], row['origin_long']], [row['dest_lat'], row['dest_long']]]
    folium.PolyLine(
        locations=points,
        color='green',
        weight=2,
        opacity=0.3,
    ).add_to(m)

# Add markers or circles for stations
for index, row in Stations.iterrows():
    # Convert the degree to a color directly within the loop
    rgba = cmap(norm(row['degree']))  # Get RGBA value from colormap
    color = matplotlib.colors.rgb2hex(rgba)  # Convert RGBA to hex format

    folium.CircleMarker(
        location=[row['lat'], row['long']],
        radius=row['degree'],  # Scale factor for visualization
        tooltip=f"{row['name']} | Line:{row['lines']} | Degree: {row['degree']}",
        fill=True,
        color=False,  # Use the colormap for the border
        fill_color=color,  # Fill color based on the degree
        fill_opacity=0.7
    ).add_to(m)

# Display the map
m
```

Figure 4. Visualize the degrees of subway stations in NYC+ codes.

Betweenness of each station: Betweenness stands for the importance of each station. It tells stakeholders like MTA that if the designated station is out of work, the connection availability of any other pairs of nodes will be affected. This measure greatly helps in extreme weather like storms and hurricanes, which might affect the subway's working. This measure helps stakeholders focus more on protecting the stations with high betweenness scores.

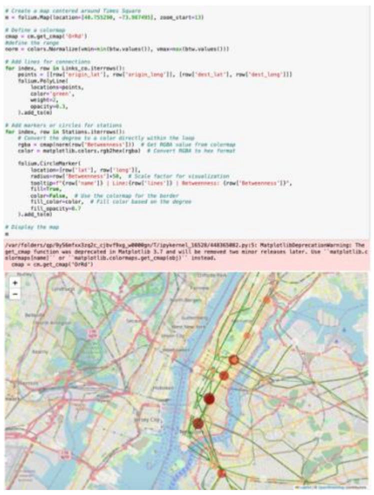
```
mc2=dict(nx.betweenness_centrality(NYCSubway))
ind2=topdict(mc2,494)
```

```
btw = dict(ind2.values)
```

```
Stations['Betweenness'] = Stations['id'].map(btw)
```

Figure 5. Create the table to store the betweenness score of each subway station

The visualization of the betweenness also keeps using the size and color path to represent the score of each node(stations) using folium. The interactions are Zooming and hovering over the station's name, lines, and betweenness score.



```
# Create a map centered around Times Square
m = folium.Map(location=[40.755290, -73.987495], zoom_start=13)

# Define a colormap
cmap = cm.get_cmap('OrRd')
#define the range
norm = colors.Normalize(vmin=min(btw.values()), vmax=max(btw.values()))

# Add lines for connections
for index, row in Links_co.iterrows():
    points = [[row['origin_lat'], row['origin_long']], [row['dest_lat'], row['dest_long']]]
    folium.PolyLine(
        locations=points,
        color='green',
        weight=2,
        opacity=0.3,
    ).add_to(m)

# Add markers or circles for stations
for index, row in Stations.iterrows():
    # Convert the degree to a color directly within the loop
    rgba = cmap(norm(row['Betweenness']))  # Get RGBA value from colormap
    color = matplotlib.colors.rgb2hex(rgba)  # Convert RGBA to hex format

    folium.CircleMarker(
        location=[row['lat'], row['long']],
        radius=row['Betweenness']*50,  # Scale factor for visualization
        tooltip=f"{row['name']} | Line:{row['lines']} | Betweenness: {row['Betweenness']}",
        fill=True,
        color=False,  # Use the colormap for the border
        fill_color=color,  # Fill color based on the degree
        fill_opacity=0.7
    ).add_to(m)

# Display the map
m
```

```
/var/folders/gp/9y56mfxx3zq2c_cjbvf9xg_w0000gn/T/ipykernel_16528/448363082.py:5: MatplotlibDeprecationWarning: The
get_cmap function was deprecated in Matplotlib 3.7 and will be removed two minor releases later. Use ``matplotlib.c
olormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)`` instead.
  cmap = cm.get_cmap('OrRd')
```

Figure 6. Visualization of the betweenness of each subway station in NYC + codes

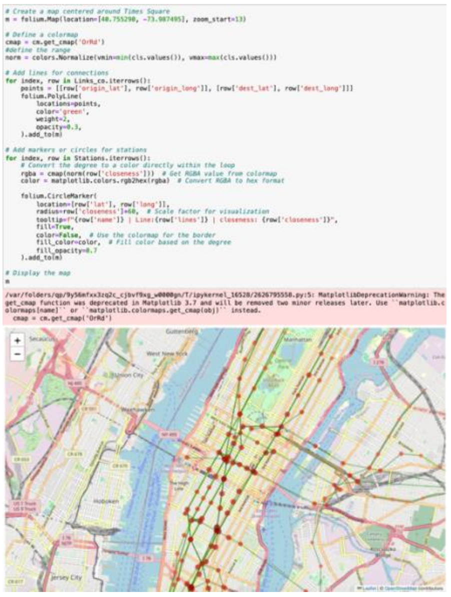
The closeness of each station: Closeness stands for the centralization of the station; the higher the closeness score, the more central the station is in the entire network. This score is used to determine the edges of the entire network; we can also use it to determine the most central point among the subway stations.

```
mc3=dict(nx.closeness_centrality(NYCSubway))
ind3=topdict(mc3,494)
cls = dict(ind3.values)
```

```
Stations['closeness'] = Stations['id'].map(cls)
```

Figure 7. The table that stores the closeness score of each station

The visualization keeps along the setting of the previous 2 visualizations and the interactions: Zoom, mouseover (subway station name, lines passing through, and closeness scores).



```
# Create a map centered around Times Square
m = folium.Map(location=[40.755290, -73.987495], zoom_start=13)

# Define a colormap
cmap = cm.get_cmap('OrRd')
#define the range
norm = colors.Normalize(vmin=min(cls.values()), vmax=max(cls.values()))

# Add lines for connections
for index, row in Links_co.iterrows():
    points = [[row['origin_lat'], row['origin_long']], [row['dest_lat'], row['dest_long']]]
    folium.PolyLine(
        locations=points,
        color='green',
        weight=2,
        opacity=0.3,
    ).add_to(m)

# Add markers or circles for stations
for index, row in Stations.iterrows():
    # Convert the degree to a color directly within the loop
    rgba = cmap(norm(row['closeness']))  # Get RGBA value from colormap
    color = matplotlib.colors.rgb2hex(rgba)  # Convert RGBA to hex format

    folium.CircleMarker(
        location=[row['lat'], row['long']],
        radius=row['closeness']*60,  # Scale factor for visualization
        tooltip=f"{row['name']} | Line:{row['lines']} | closeness: {row['closeness']}",
        fill=True,
        color=False,  # Use the colormap for the border
        fill_color=color,  # Fill color based on the degree
        fill_opacity=0.7
    ).add_to(m)

# Display the map
m
```

```
/var/folders/qp/9y56mfxx3zq2c_cjbvf9xg_w0000gn/T/ipykernel_16528/2626795558.py:5: MatplotlibDeprecationWarning: The
get_cmap function was deprecated in Matplotlib 3.7 and will be removed two minor releases later. Use ``matplotlib.c
olormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)`` instead.
  cmap = cm.get_cmap('OrRd')
```

Figure 8. The closeness of each station in NYC + codes

3. Implementation

The research focuses on the centrality of each subway station in the very first visualization. The color path representing the population density in each zip code has been removed and refilled in gray to illustrate the research focus. There is no interaction in the first visualization, and color path and size have been applied to the nodes to represent the degree of the stations.

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

# get rid of color path of the map, and show the boundary
zipgdp.plot(legend=True, ax=ax,color = 'gray',edgecolor='black', alpha=0.30)
# Normalize the color map for nodes, with the maximum value of degree in deep red and minimum in light red
norm = colors.Normalize(vmin=min(deg.values()), vmax=max(deg.values()))

nx.draw(NYCSubway,pos=GPos,with_labels=False,arrows=False,
        node_size=[deg.get(node, 0) * 10 for node in NYCSubway.nodes()],
        node_color=[plt.cm.OrRd(norm(deg[node])) for node in NYCSubway.nodes()],
        width=2,edge_color='green')

sm = cm.ScalarMappable(cmap=plt.cm.OrRd, norm=norm)
plt.colorbar(sm, ax=ax,orientation='horizontal', location='top')
plt.show()
```

Figure 9. Degree for each station in NYC without interaction + codes

The shortness of the visualization above is that it does not tell the information of each station, it is hard to tell which station is shown on the map, and it does not support use to test if our visualization is correctly shown. So, another visualization package, *Altair*, has been used for the mouseover effect, and it tells the station's name, lines, and degree.

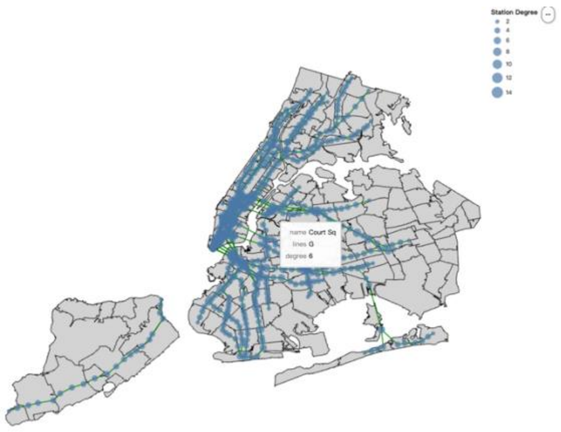
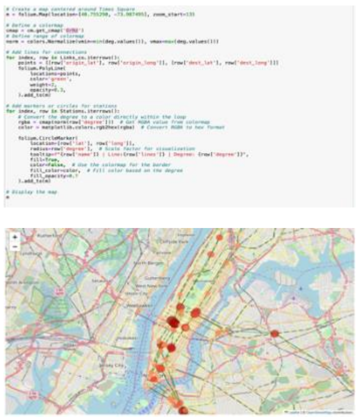


Figure 10. Degree visualization by *altair*

However, the Altair does not support color path visualization, and the nodes stack together in the map; it is hard to select the desired station. In the final version of the visualization, the package *folium* was used to determine the degree and other two centralities. It allows the visualization to happen based on google Maps and allows the Zoom interaction in the visualization. The size of the network layer on the Google map allows fixing to match the zoomed map, so it avoids the node stacking problems.



```
# Create a map centered around Times Square
m = folium.Map(location=[40.755290, -73.987495], zoom_start=13)

# Define a colormap
cmap = cm.get_cmap('OrRd')
# Define range of colormap
norm = colors.Normalize(vmin=min(deg.values()), vmax=max(deg.values()))

# Add lines for connections
for index, row in Links_co.iterrows():
    points = [[row['origin_lat'], row['origin_long']], [row['dest_lat'], row['dest_long']]]
    folium.PolyLine(
        locations=points,
        color='green',
        weight=2,
        opacity=0.3,
    ).add_to(m)

# Add markers or circles for stations
for index, row in Stations.iterrows():
    # Convert the degree to a color directly within the loop
    rgba = cmap(norm(row['degree']))  # Get RGBA value from colormap
    color = matplotlib.colors.rgb2hex(rgba)  # Convert RGBA to hex format

    folium.CircleMarker(
        location=[row['lat'], row['long']],
        radius=row['degree'],  # Scale factor for visualization
        tooltip=f"{row['name']} | Line:{row['lines']} | Degree: {row['degree']}",
        fill=True,
        color=False,  # Use the colormap for the border
        fill_color=color,  # Fill color based on the degree
        fill_opacity=0.7
    ).add_to(m)

# Display the map
m
```

Figure 4. visualize the degrees of subway stations in NYC+ codes

4. Discussions

The analysis provides significant insights into the operational vulnerabilities of the NYC subway system during extreme weather events and suggests focal points for enhancing resilience. The centrality measures employed offer a robust framework for understanding the impact of specific stations on the overall network during crises. Future work could explore the integration of additional data layers, such as economic factors or historical disruption patterns, to further refine resilience strategies. A limitation of the current system is the reliance on static population data, which may not accurately reflect real-time or event-specific population movements.

5. References

"Example Gallery#." Example Gallery - Vega-Altair 5.3.0. Documentation, altair-viz.github.io/gallery/index.html#scatter-plots. Accessed 12 May 2024.

Story, Rob. Folium 0.16.0, pypi.org/project/folium/.