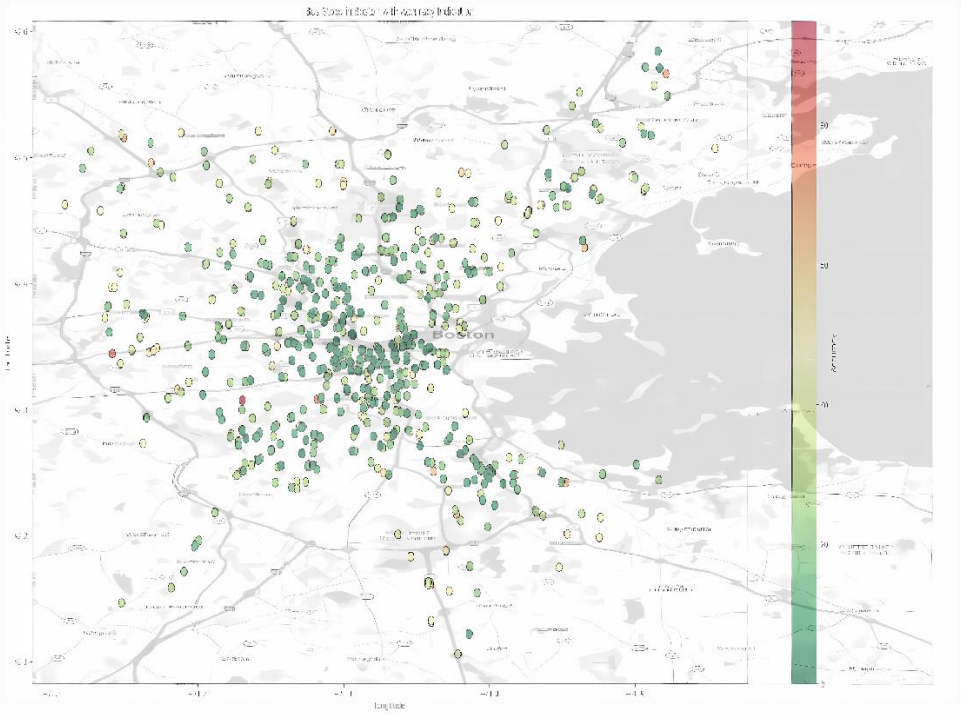
In [ ]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import geopandas as gpd
import os
from google.colab import drive
drive.mount('/content/MyDrive/')

dir_path = '/content/MyDrive/MyDrive/City of Boston: Transit & Performance A/Data Files/'
files = files = os.listdir(dir_path)

combined_df = pd.read_csv(f"{dir_path}/MBTA-Bus-Arrival-Departure-Times_2022-01.csv")
def is_on_time(row):
    schedual = row['scheduled']
    actual = row['actual']
    if isinstance(actual, str):
        return schedual > actual
    return False

# Group by the 'X' column and calculate the percentage of true conditions
result = combined_df.groupby('stop_id').apply(lambda group: (group.apply(is_on_time, axis=1).sum() / len(group)) * 100)
result = result.sort_values(ascending=False)

gdf = gpd.read_file(f"{dir_path}/MBTA_Bus_Routes_and_Stops.geojson")
stop_info = pd.DataFrame(gdf)
result = pd.DataFrame(result)
stop_ids = result.index
accuracies = result.iloc[:, 0].values

# Creating a new DataFrame
new_df = pd.DataFrame({
    'stop_id': stop_ids,
    'accuracy': accuracies
})

# Display the new DataFrame
print(new_df)

new_df['stop_id'] = new_df['stop_id'].astype(stop_info['STOP_ID'].dtype)

merged_df = pd.merge(new_df, stop_info[['STOP_ID', 'geometry']],
                     left_on='stop_id', right_on='STOP_ID', how='left')

# Dropping the duplicate STOP_ID column if you don't need it
merged_df = merged_df.drop(columns=['STOP_ID'])
merged_df.to_pickle("/content/MyDrive/MyDrive/City of Boston: Transit & Performance A/Data Files/stop_info.pkl")
```

In [ ]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import geopandas as gpd
from shapely import wkt
from google.colab import drive
import os
drive.mount('/content/MyDrive/')

dir_path = '/content/MyDrive/MyDrive/City of Boston: Transit & Performance A/Data Files/'
files = files = os.listdir(dir_path)

df = pd.read_pickle(f"{dir_path}/stop_info.pkl")
df = df.dropna()
df['geometry'] = df['geometry'].apply(lambda point: str(point))
df['geometry'] = df['geometry'].apply(wkt.loads)
gdf = gpd.GeoDataFrame(df, geometry='geometry')

world_cities = gpd.read_file(gpd.datasets.get_path('naturalearth_cities'))
boston_map = world_cities[world_cities['name'] == 'Boston']
# Load Boston map


# Plotting
fig, ax = plt.subplots(figsize=(10, 10))


# Normalize the accuracy for color mapping
norm = plt.Normalize(gdf['accuracy'].min(), gdf['accuracy'].max())
cmap = plt.cm.RdYlGn_r

# Plot each point
for idx, row in gdf.iterrows():
    ax.scatter(row.geometry.x, row.geometry.y,
               color=cmap(norm(row.accuracy)),
               edgecolor='black',
               s=100) # arbitrary size

# Add color bar
sm = plt.cm.ScalarMappable(cmap=cmap, norm=norm)
sm._A = []
plt.colorbar(sm, ax=ax, orientation='vertical', label='Accuracy')

# Set labels and title
ax.set_title('Bus Stops in Boston with Accuracy Indication')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')

plt.show()
```


Boston
Longitude
Latitude