EMS Response in NYC: Data Manipulation & Analysis

Diya Mohan & Sumana Chilakamarri

Purpose & Interest

- Disparity in healthcare accessibility in marginalized communities versus communities of privilege
- EMS Dispatch & Response Times in New York City boroughs
    - NYC has a large-scale model that displays extreme wealth disparities, densely concentrated populations, etc., that serve as a common indicator of wealthy and poorer areas.

Goals

- Clean, analyze & visualize data to gain insights on different factors regarding EMS Response Time
- Compare NYC boroughs based on Dispatch and Response Time
- Predict future demographic data using predictive analytics - use this to compare future EMS Response/Dispatch times

Hypotheses

We hypothesize that we will find statistically significant indicators that low-income boroughs do in fact receive significantly **less emergency medical response resources** than high-income boroughs on the basis of EMS efficiency.

Phase I: Dataset Collection

Data Collection Process

Collecting Data

We looked up emergency response rate records for NYC for datasets, found relevant data in government websites

.csv Data

.csv was found at: https://data.cityofnewyork.us/, large file with ~25M rows containing dispatch and response time, corresponding borough, etc.

Web Scraping

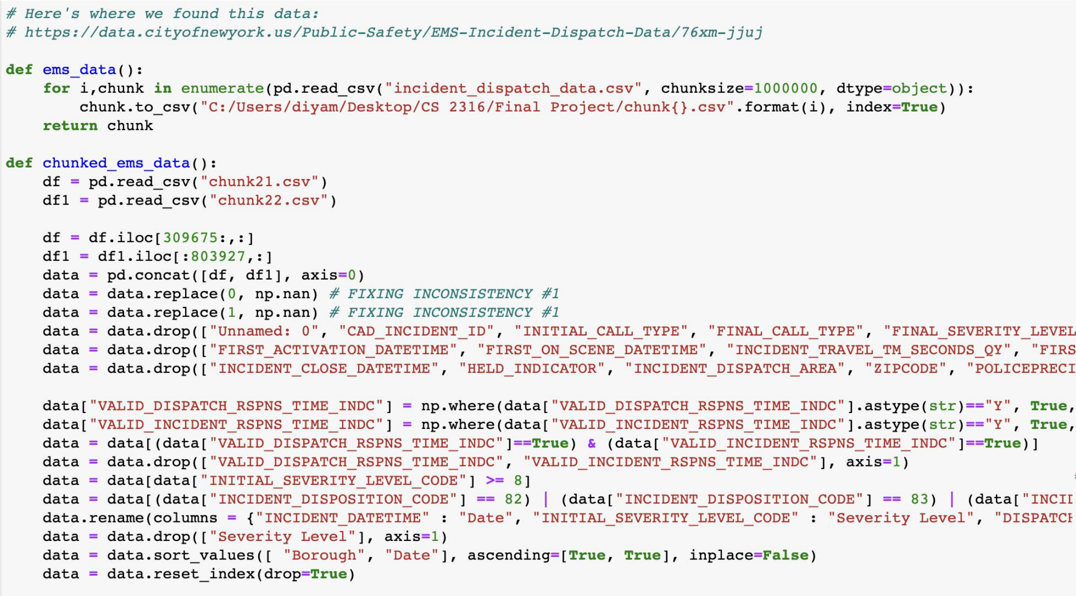
Looked up demographics for NYC boroughs and found relevant data at: furmancenter.org

.json Data

Found data from data.gov, contained data with all types of emergency response times & number of incidents

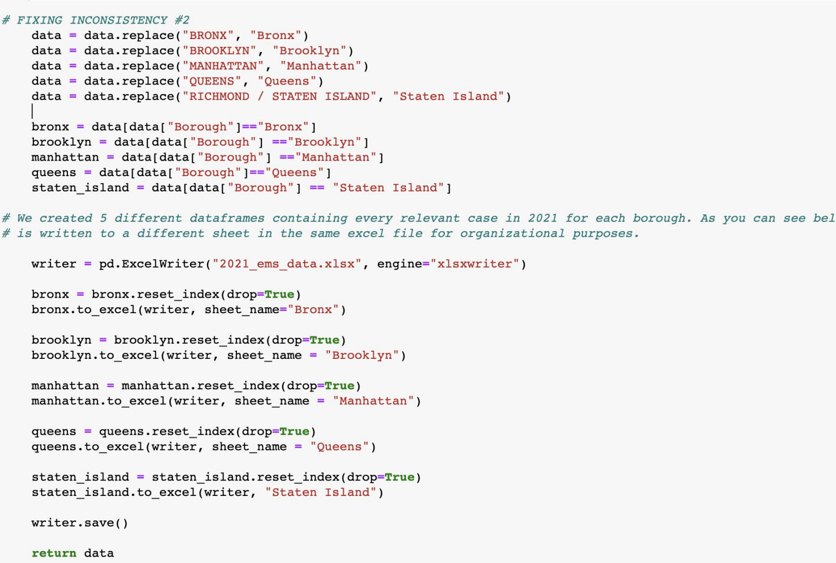
Phase II: Data Manipulation & Cleaning

Cleaning our Big Dataset (CSV)



```
# Here's where we found this data:
# https://data.cityofnewyork.us/Public-Safety/EMS-Incident-Dispatch-Data/76xm-jjuj

def ems_data():
    for i,chunk in enumerate(pd.read_csv("incident_dispatch_data.csv", chunksize=1000000, dtype=object)):
        chunk.to_csv("C:/Users/diyam/Desktop/CS 2316/Final Project/chunk{}.csv".format(i), index=True)
    return chunk

def chunked_ems_data():
    df = pd.read_csv("chunk21.csv")
    df1 = pd.read_csv("chunk22.csv")

    df = df.iloc[309675:,:]
    df1 = df1.iloc[:803927,:]
    data = pd.concat([df, df1], axis=0)
    data = data.replace(0, np.nan) # FIXING INCONSISTENCY #1
    data = data.replace(1, np.nan) # FIXING INCONSISTENCY #1
    data = data.drop(["Unnamed: 0", "CAD_INCIDENT_ID", "INITIAL_CALL_TYPE", "FINAL_CALL_TYPE", "FINAL_SEVERITY_LEVEL_CO
    data = data.drop(["FIRST_ACTIVATION_DATETIME", "FIRST_ON_SCENE_DATETIME", "INCIDENT_TRAVEL_TM_SECONDS_QY", "FIRST_T
    data = data.drop(["INCIDENT_CLOSE_DATETIME", "HELD_INDICATOR", "INCIDENT_DISPATCH_AREA", "ZIPCODE", "POLICEPRECINCT

    data["VALID_DISPATCH_RSPNS_TIME_INDC"] = np.where(data["VALID_DISPATCH_RSPNS_TIME_INDC"].astype(str)=="Y", True, Fa
    data["VALID_INCIDENT_RSPNS_TIME_INDC"] = np.where(data["VALID_INCIDENT_RSPNS_TIME_INDC"].astype(str)=="Y", True, Fa
    data = data[(data["VALID_DISPATCH_RSPNS_TIME_INDC"]==True) & (data["VALID_INCIDENT_RSPNS_TIME_INDC"]==True)]
    data = data.drop(["VALID_DISPATCH_RSPNS_TIME_INDC", "VALID_INCIDENT_RSPNS_TIME_INDC"], axis=1)
    data = data[data["INITIAL_SEVERITY_LEVEL_CODE"] >= 8]
    data = data[(data["INCIDENT_DISPOSITION_CODE"] == 82) | (data["INCIDENT_DISPOSITION_CODE"] == 83) | (data["INCII
    data.rename(columns = {"INCIDENT_DATETIME" : "Date", "INITIAL_SEVERITY_LEVEL_CODE" : "Severity Level", "DISPATCH
    data = data.drop(["Severity Level"], axis=1)
    data = data.sort_values([ "Borough", "Date"], ascending=[True, True], inplace=False)
    data = data.reset_index(drop=True)
```

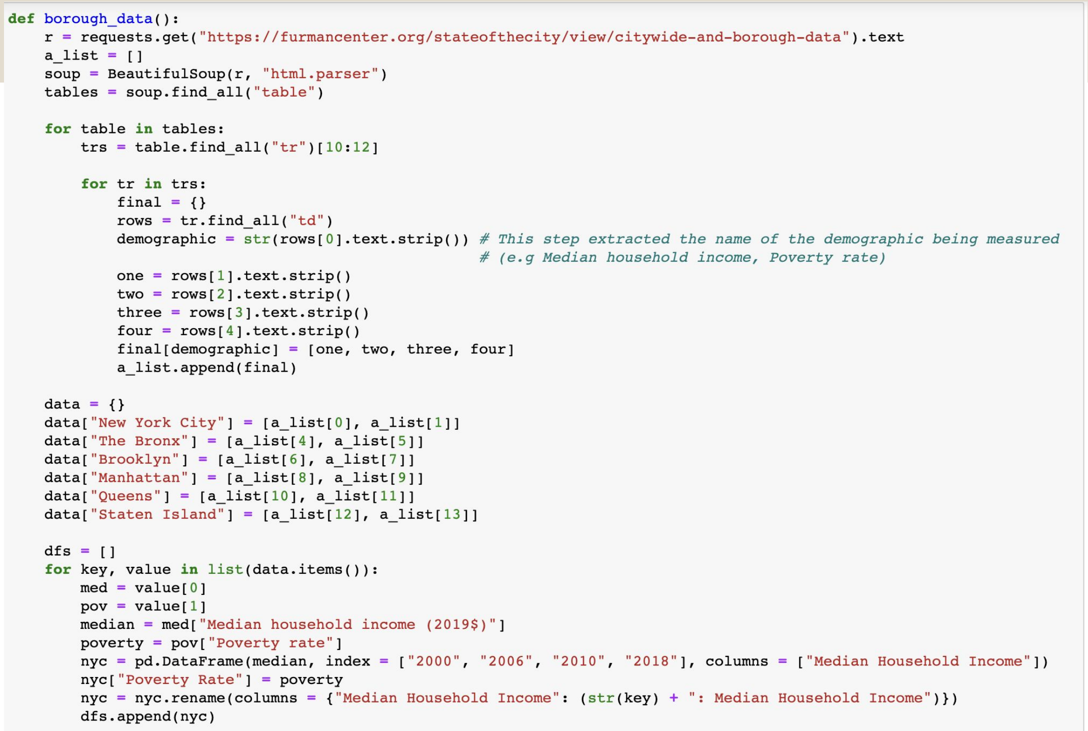
- Since our data was ~25M rows, we split it into multiple chunks and only used the chunks with relevant data (from 2021).
- We dropped columns that were not relevant to the insights we wanted to find to support our hypothesis
- We renamed columns for readability, sorted it, and wrote it to files to cross reference it.



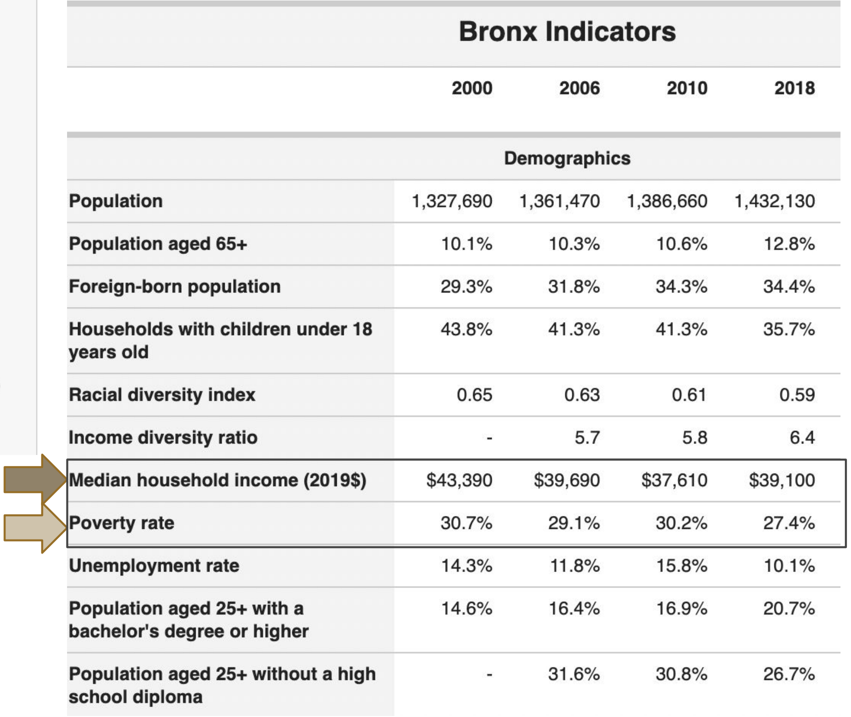
```
# FIXING INCONSISTENCY #2
    data = data.replace("BRONX", "Bronx")
    data = data.replace("BROOKLYN", "Brooklyn")
    data = data.replace("MANHATTAN", "Manhattan")
    data = data.replace("QUEENS", "Queens")
    data = data.replace("RICHMOND / STATEN ISLAND", "Staten Island")

    bronx = data[data["Borough"]=="Bronx"]
    brooklyn = data[data["Borough"] =="Brooklyn"]
    manhattan = data[data["Borough"] =="Manhattan"]
    queens = data[data["Borough"]=="Queens"]
    staten_island = data[data["Borough"] == "Staten Island"]

# We created 5 different dataframes containing every relevant case in 2021 for each borough. As you can see bel
# is written to a different sheet in the same excel file for organizational purposes.

    writer = pd.ExcelWriter("2021_ems_data.xlsx", engine="xlsxwriter")

    bronx = bronx.reset_index(drop=True)
    bronx.to_excel(writer, sheet_name="Bronx")

    brooklyn = brooklyn.reset_index(drop=True)
    brooklyn.to_excel(writer, sheet_name = "Brooklyn")

    manhattan = manhattan.reset_index(drop=True)
    manhattan.to_excel(writer, sheet_name = "Manhattan")

    queens = queens.reset_index(drop=True)
    queens.to_excel(writer, sheet_name = "Queens")

    staten_island = staten_island.reset_index(drop=True)
    staten_island.to_excel(writer, "Staten Island")

    writer.save()

    return data
```

Web Scraping

We wanted to extract only **median household income** & **poverty rates** across several years for each borough, and the website we used had many demographics we had to parse through.



```python
def borough_data():
    r = requests.get("https://furmancenter.org/stateofthecity/view/citywide-and-borough-data").text
    a_list = []
    soup = BeautifulSoup(r, "html.parser")
    tables = soup.find_all("table")

    for table in tables:
        trs = table.find_all("tr")[10:12]

        for tr in trs:
            final = {}
            rows = tr.find_all("td")
            demographic = str(rows[0].text.strip()) # This step extracted the name of the demographic being measured
                                                    # (e.g Median household income, Poverty rate)
            one = rows[1].text.strip()
            two = rows[2].text.strip()
            three = rows[3].text.strip()
            four = rows[4].text.strip()
            final[demographic] = [one, two, three, four]
            a_list.append(final)

    data = {}
    data["New York City"] = [a_list[0], a_list[1]]
    data["The Bronx"] = [a_list[4], a_list[5]]
    data["Brooklyn"] = [a_list[6], a_list[7]]
    data["Manhattan"] = [a_list[8], a_list[9]]
    data["Queens"] = [a_list[10], a_list[11]]
    data["Staten Island"] = [a_list[12], a_list[13]]

    dfs = []
    for key, value in list(data.items()):
        med = value[0]
        pov = value[1]
        median = med["Median household income (2019$)"]
        poverty = pov["Poverty rate"]
        nyc = pd.DataFrame(median, index = ["2000", "2006", "2010", "2018"], columns = ["Median Household Income"])
        nyc["Poverty Rate"] = poverty
        nyc = nyc.rename(columns = {"Median Household Income": (str(key) + ": Median Household Income")})
        dfs.append(nyc)
```



Bronx Indicators

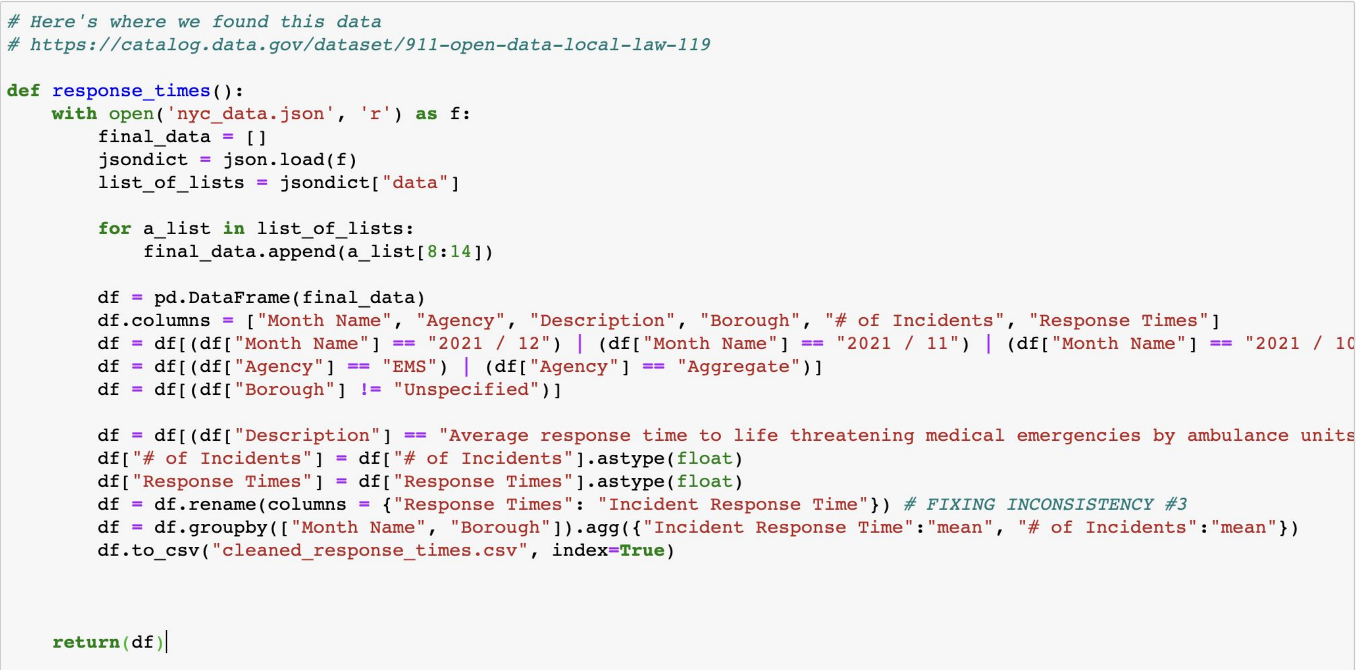
	2000	2006	2010	2018
Demographics				
Population	1,327,690	1,361,470	1,386,660	1,432,130
Population aged 65+	10.1%	10.3%	10.6%	12.8%
Foreign-born population	29.3%	31.8%	34.3%	34.4%
Households with children under 18 years old	43.8%	41.3%	41.3%	35.7%
Racial diversity index	0.65	0.63	0.61	0.59
Income diversity ratio	-	5.7	5.8	6.4
Median household income (2019$)	$43,390	$39,690	$37,610	$39,100
Poverty rate	30.7%	29.1%	30.2%	27.4%
Unemployment rate	14.3%	11.8%	15.8%	10.1%
Population aged 25+ with a bachelor's degree or higher	14.6%	16.4%	16.9%	20.7%
Population aged 25+ without a high school diploma	-	31.6%	30.8%	26.7%

- Used BeautifulSoup to parse through data
- Only appended relevant values to our list; added labelled values to DataFrame for easier comparison & visualization

JSON data

We appended relevant values from the .json to a Pandas Dataframe and only kept columns in the year matching to that of the big .csv for comparison purposes.

We used json.load() to convert to a dictionary, converted dictionary to Pandas DataFrame and only included # of Incidents and Response Times



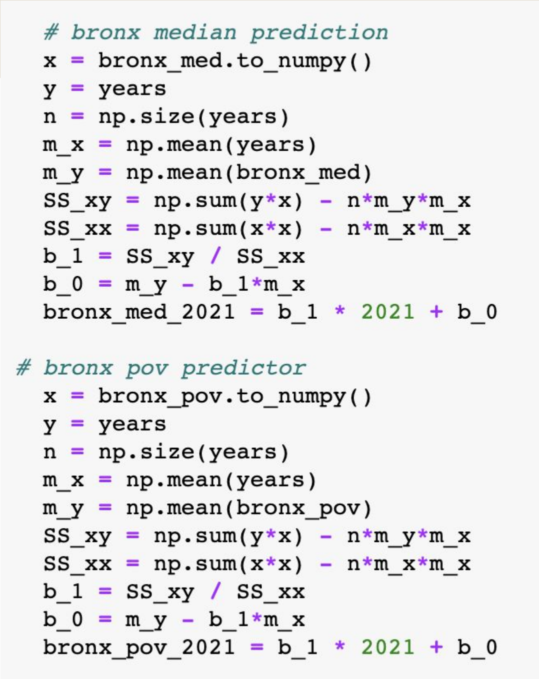
```
# Here's where we found this data
# https://catalog.data.gov/dataset/911-open-data-local-law-119

def response_times():
    with open('nyc_data.json', 'r') as f:
        final_data = []
        jsondict = json.load(f)
        list_of_lists = jsondict["data"]

        for a_list in list_of_lists:
            final_data.append(a_list[8:14])

        df = pd.DataFrame(final_data)
        df.columns = ["Month Name", "Agency", "Description", "Borough", "# of Incidents", "Response Times"]
        df = df[(df["Month Name"] == "2021 / 12") | (df["Month Name"] == "2021 / 11") | (df["Month Name"] == "2021 / 10
        df = df[(df["Agency"] == "EMS") | (df["Agency"] == "Aggregate")]
        df = df[(df["Borough"] != "Unspecified")]

        df = df[(df["Description"] == "Average response time to life threatening medical emergencies by ambulance units
        df["# of Incidents"] = df["# of Incidents"].astype(float)
        df["Response Times"] = df["Response Times"].astype(float)
        df = df.rename(columns = {"Response Times": "Incident Response Time"}) # FIXING INCONSISTENCY #3
        df = df.groupby(["Month Name", "Borough"]).agg({"Incident Response Time":"mean", "# of Incidents":"mean"})
        df.to_csv("cleaned_response_times.csv", index=True)



    return(df)
```

We used masking to only include severe, life threatening accidents for better data visualization

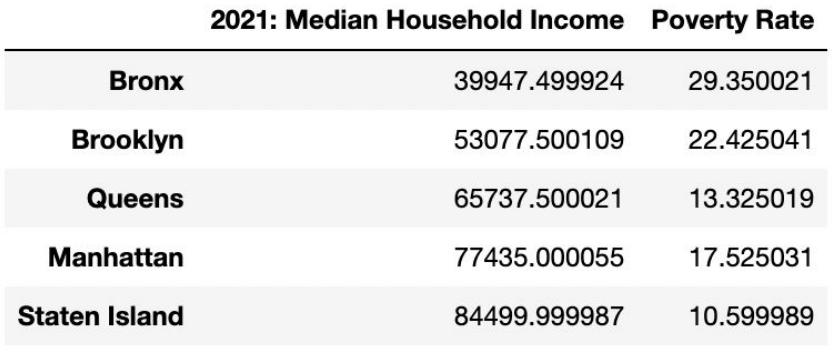
Phase III: Data Analysis

Insight 1

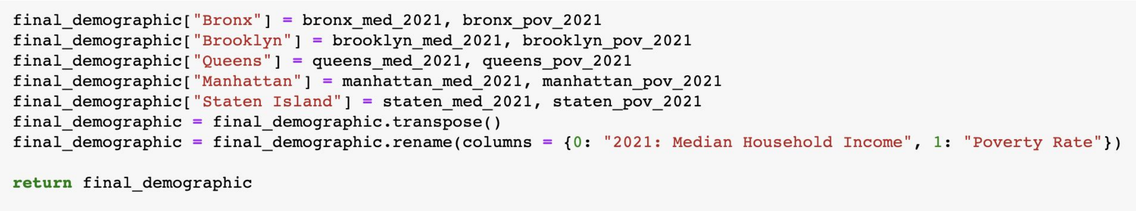
Boroughs such as the Bronx and Brooklyn had the lowest median household incomes and highest poverty rates while boroughs such as Manhattan and Staten Island had the highest median incomes and lowest poverty rates in 2021.



```
 # bronx median prediction
 x = bronx_med.to_numpy()
 y = years
 n = np.size(years)
 m_x = np.mean(years)
 m_y = np.mean(bronx_med)
 SS_xy = np.sum(y*x) - n*m_y*m_x
 SS_xx = np.sum(x*x) - n*m_x*m_x
 b_1 = SS_xy / SS_xx
 b_0 = m_y - b_1*m_x
 bronx_med_2021 = b_1 * 2021 + b_0

# bronx pov predictor
 x = bronx_pov.to_numpy()
 y = years
 n = np.size(years)
 m_x = np.mean(years)
 m_y = np.mean(bronx_pov)
 SS_xy = np.sum(y*x) - n*m_y*m_x
 SS_xx = np.sum(x*x) - n*m_x*m_x
 b_1 = SS_xy / SS_xx
 b_0 = m_y - b_1*m_x
 bronx_pov_2021 = b_1 * 2021 + b_0
```



	2021: Median Household Income	Poverty Rate
Bronx	39947.499924	29.350021
Brooklyn	53077.500109	22.425041
Queens	65737.500021	13.325019
Manhattan	77435.000055	17.525031
Staten Island	84499.999987	10.599989

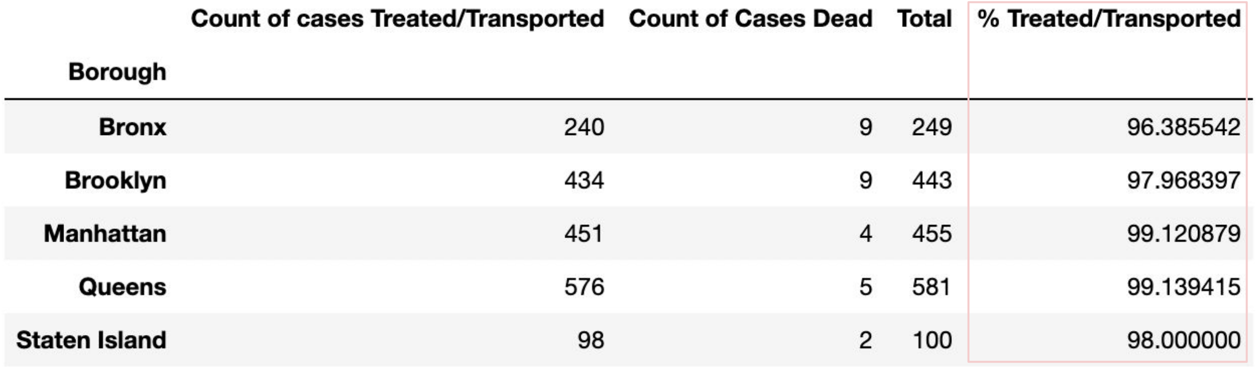


```
final_demographic["Bronx"] = bronx_med_2021, bronx_pov_2021
final_demographic["Brooklyn"] = brooklyn_med_2021, brooklyn_pov_2021
final_demographic["Queens"] = queens_med_2021, queens_pov_2021
final_demographic["Manhattan"] = manhattan_med_2021, manhattan_pov_2021
final_demographic["Staten Island"] = staten_med_2021, staten_pov_2021
final_demographic = final_demographic.transpose()
final_demographic = final_demographic.rename(columns = {0: "2021: Median Household Income", 1: "Poverty Rate"})

return final_demographic
```

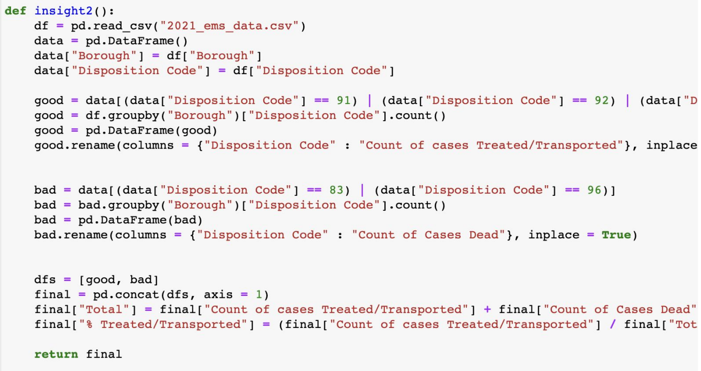
This data helped us to **rank the boroughs** and recognize which ones fall under low-income and high-income in 2021.

Insight 2

The lowest percent of EMS incidents treated and transported fall under lower-income boroughs such as Brooklyn and Bronx.



	Count of cases Treated/Transported	Count of Cases Dead	Total	% Treated/Transported
Borough				
Bronx	240	9	249	96.385542
Brooklyn	434	9	443	97.968397
Manhattan	451	4	455	99.120879
Queens	576	5	581	99.139415
Staten Island	98	2	100	98.000000

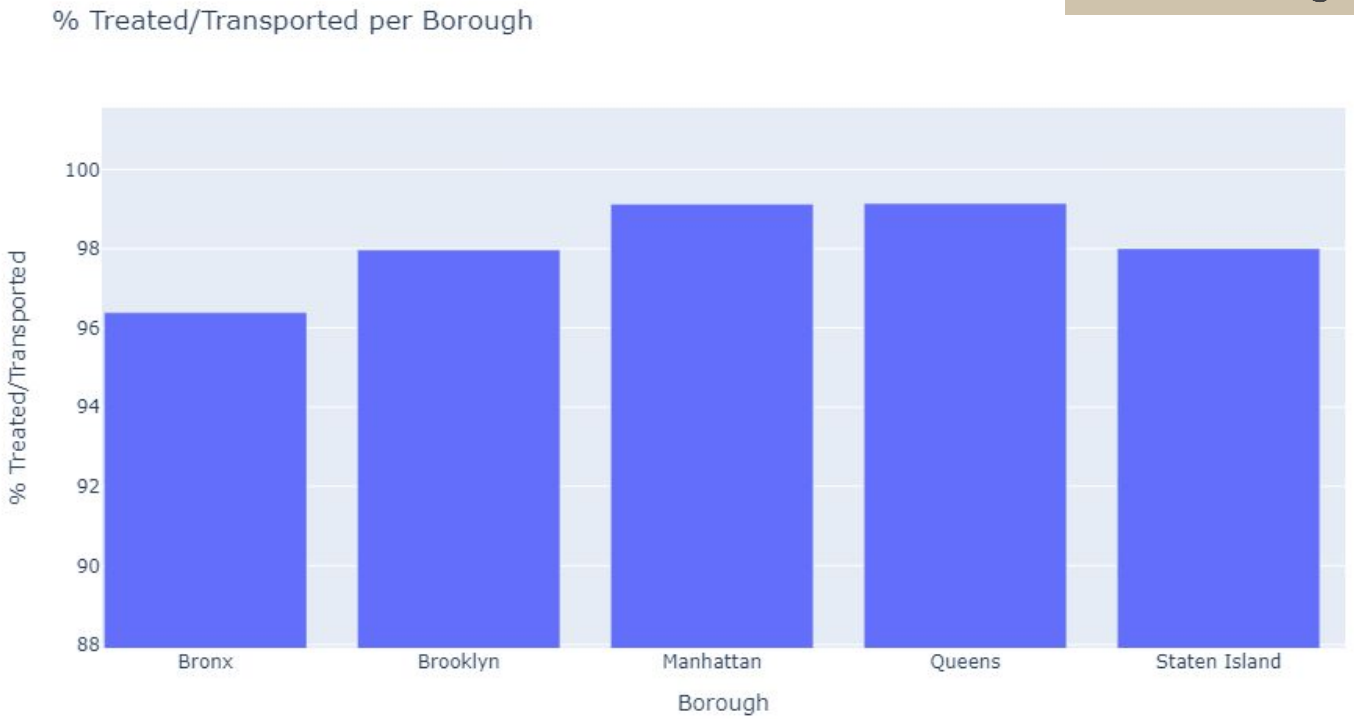


```
def insight2():
    df = pd.read_csv("2021_ems_data.csv")
    data = pd.DataFrame()
    data["Borough"] = df["Borough"]
    data["Disposition Code"] = df["Disposition Code"]

    good = data[(data["Disposition Code"] == 91) | (data["Disposition Code"] == 92) | (data["D
    good = df.groupby("Borough")["Disposition Code"].count()
    good = pd.DataFrame(good)
    good.rename(columns = {"Disposition Code" : "Count of cases Treated/Transported"}, inplace


    bad = data[(data["Disposition Code"] == 83) | (data["Disposition Code"] == 96)]
    bad = bad.groupby("Borough")["Disposition Code"].count()
    bad = pd.DataFrame(bad)
    bad.rename(columns = {"Disposition Code" : "Count of Cases Dead"}, inplace = True)


    dfs = [good, bad]
    final = pd.concat(dfs, axis = 1)
    final["Total"] = final["Count of cases Treated/Transported"] + final["Count of Cases Dead"
    final["% Treated/Transported"] = (final["Count of cases Treated/Transported"] / final["Tot

    return final
```

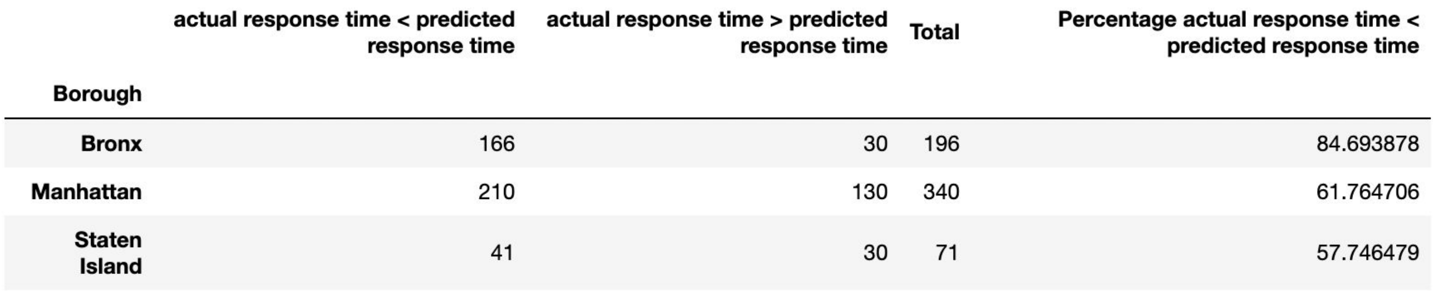
We used masking to group by disposition code to see the patient outcomes per borough.

Data Visualization #1

This bar graph supports our second insight: the % treated for lower income boroughs is lower than that of higher income boroughs



Insight #3

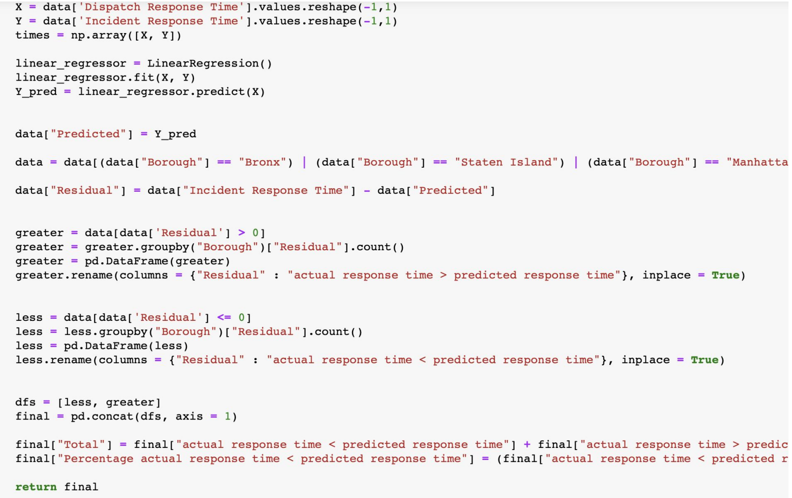


	actual response time < predicted response time	actual response time > predicted response time	Total	Percentage actual response time < predicted response time
Borough				
Bronx	166	30	196	84.693878
Manhattan	210	130	340	61.764706
Staten Island	41	30	71	57.746479

Lower income boroughs (Bronx) had the greatest percentage of their cases where actual response time was less than predicted as opposed to higher income boroughs such as Manhattan and Staten Island.

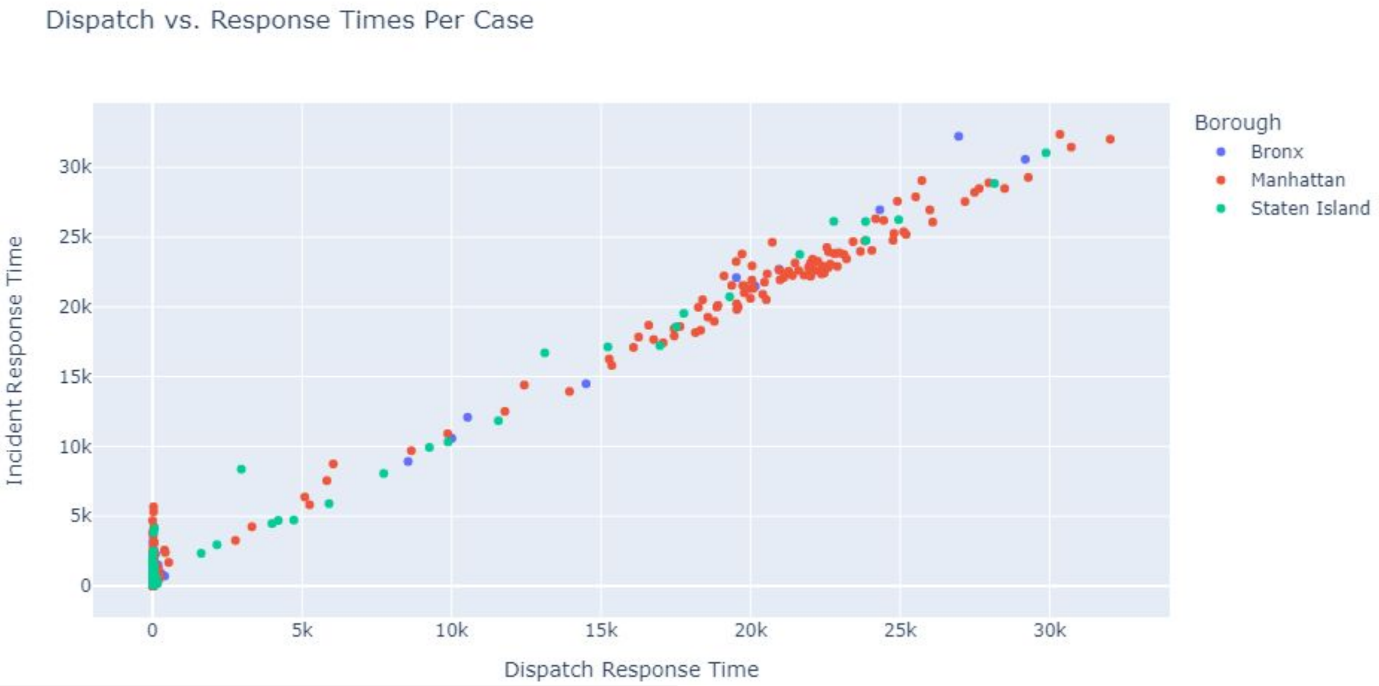
We ran an ordinary linear regression between the two variables and found residuals for incident response times:

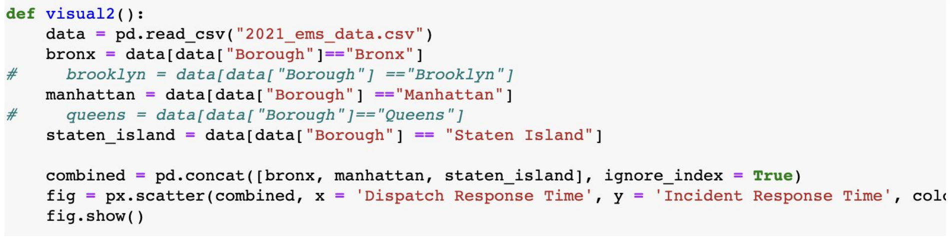
- defined as the actual incident response time minus the predicted incident response time based on our linear regression
- for each borough, we counted the positive residuals and the negative residuals



```
X = data['Dispatch Response Time'].values.reshape(-1,1)
Y = data['Incident Response Time'].values.reshape(-1,1)
times = np.array([X, Y])

linear_regressor = LinearRegression()
linear_regressor.fit(X, Y)
Y_pred = linear_regressor.predict(X)


data["Predicted"] = Y_pred

data = data[(data["Borough"] == "Bronx") | (data["Borough"] == "Staten Island") | (data["Borough"] == "Manhatta

data["Residual"] = data["Incident Response Time"] - data["Predicted"]


greater = data[data['Residual'] > 0]
greater = greater.groupby("Borough")["Residual"].count()
greater = pd.DataFrame(greater)
greater.rename(columns = {"Residual" : "actual response time > predicted response time"}, inplace = True)


less = data[data['Residual'] <= 0]
less = less.groupby("Borough")["Residual"].count()
less = pd.DataFrame(less)
less.rename(columns = {"Residual" : "actual response time < predicted response time"}, inplace = True)


dfs = [less, greater]
final = pd.concat(dfs, axis = 1)

final["Total"] = final["actual response time < predicted response time"] + final["actual response time > predic
final["Percentage actual response time < predicted response time"] = (final["actual response time < predicted r

return final
```

Data Visualization #2



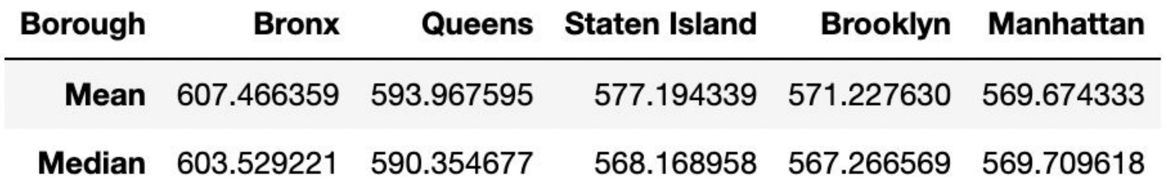


```
def visual2():
    data = pd.read_csv("2021_ems_data.csv")
    bronx = data[data["Borough"]=="Bronx"]
#     brooklyn = data[data["Borough"] =="Brooklyn"]
    manhattan = data[data["Borough"] =="Manhattan"]
#     queens = data[data["Borough"]=="Queens"]
    staten_island = data[data["Borough"] == "Staten Island"]

    combined = pd.concat([bronx, manhattan, staten_island], ignore_index = True)
    fig = px.scatter(combined, x = 'Dispatch Response Time', y = 'Incident Response Time', col
    fig.show()
```

- This visual supports the regression made in insight 3: it displays the strong, positive correlation in Dispatch response time vs Incident response time
- Shows the different boroughs and where their data points lie, clusters in data, etc.

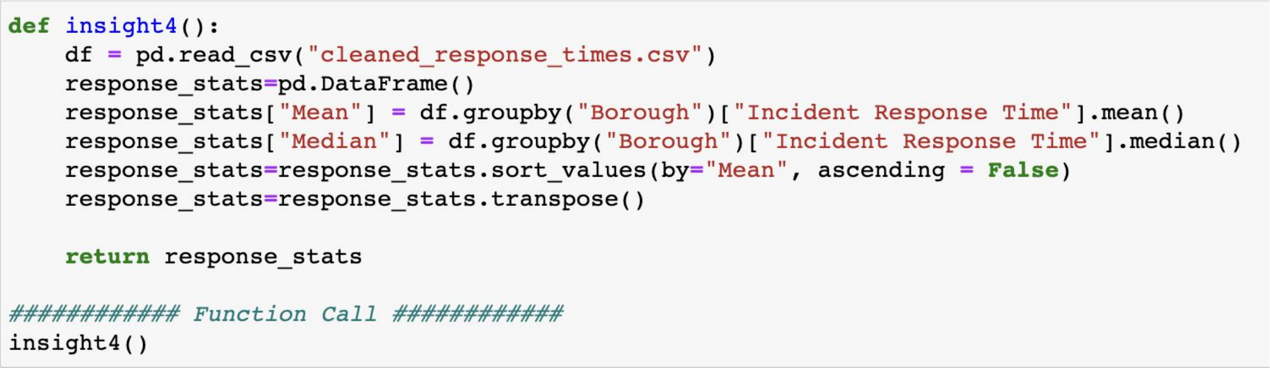
Insight 4

Lower income boroughs have a higher mean and median incident response time as compared to higher income boroughs.



Borough	Bronx	Queens	Staten Island	Brooklyn	Manhattan
Mean	607.466359	593.967595	577.194339	571.227630	569.674333
Median	603.529221	590.354677	568.168958	567.266569	569.709618

We used aggregate methods to find the average and median, both measures of center, of the incident response time in each borough and sorted them from highest response time to lowest response time.



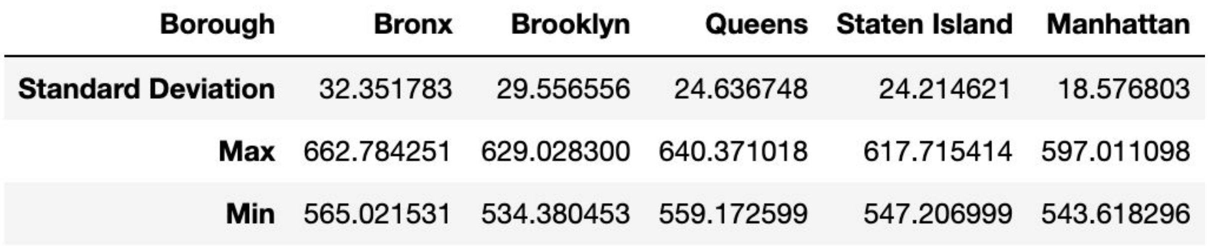
```
def insight4():
    df = pd.read_csv("cleaned_response_times.csv")
    response_stats=pd.DataFrame()
    response_stats["Mean"] = df.groupby("Borough")["Incident Response Time"].mean()
    response_stats["Median"] = df.groupby("Borough")["Incident Response Time"].median()
    response_stats=response_stats.sort_values(by="Mean", ascending = False)
    response_stats=response_stats.transpose()

    return response_stats

############ Function Call ############
insight4()
```

Insight 5

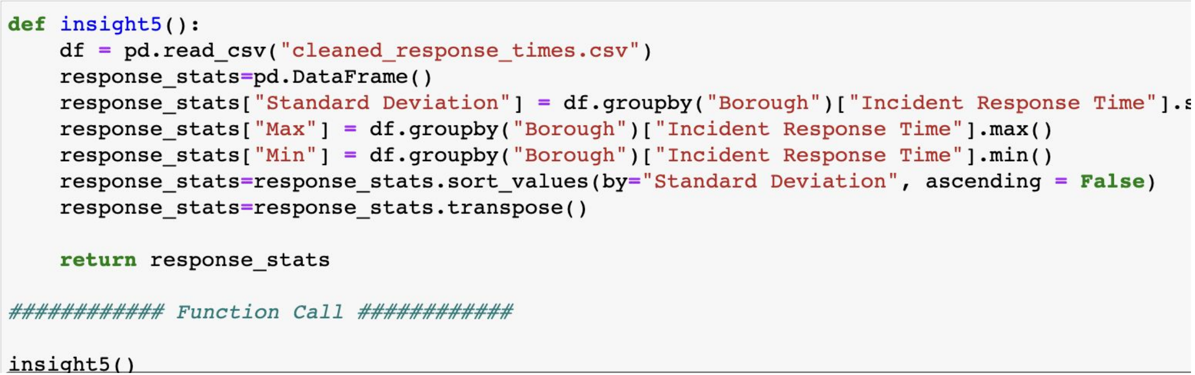
Lower income boroughs, including the Bronx and Brooklyn, have a greater spread of incident response times than higher income boroughs, including Manhattan.

The spread of data varies significantly from borough to borough:

- Standard Deviation in the Bronx is much higher than that of Manhattan
- There is more spread in the Bronx data, meaning that values are extremely high and extremely low

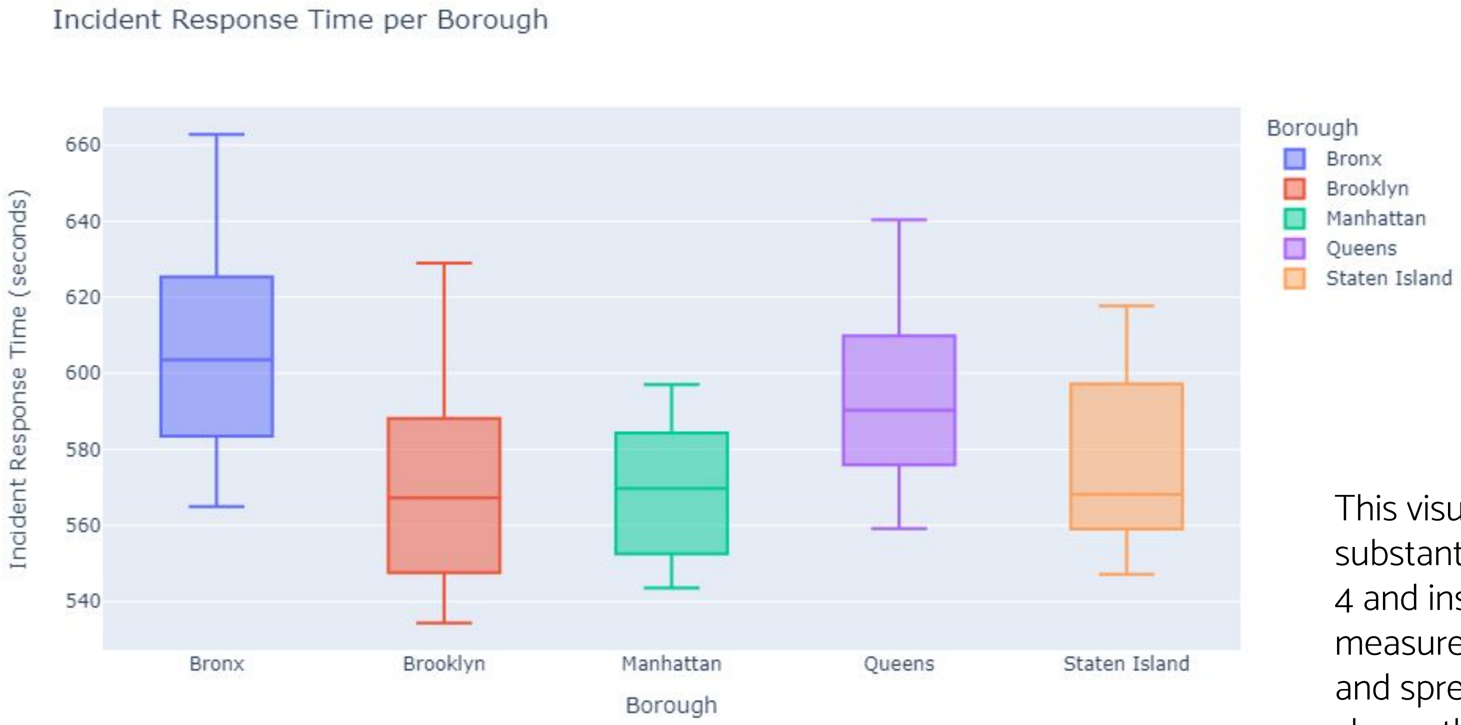


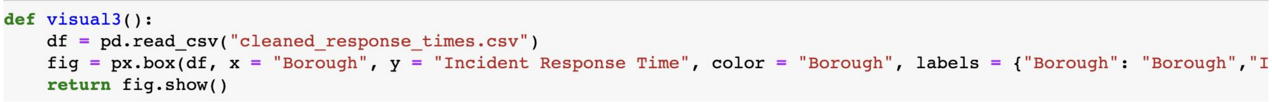
Borough	Bronx	Brooklyn	Queens	Staten Island	Manhattan
Standard Deviation	32.351783	29.556556	24.636748	24.214621	18.576803
Max	662.784251	629.028300	640.371018	617.715414	597.011098
Min	565.021531	534.380453	559.172599	547.206999	543.618296



```
def insight5():
    df = pd.read_csv("cleaned_response_times.csv")
    response_stats=pd.DataFrame()
    response_stats["Standard Deviation"] = df.groupby("Borough")["Incident Response Time"].s
    response_stats["Max"] = df.groupby("Borough")["Incident Response Time"].max()
    response_stats["Min"] = df.groupby("Borough")["Incident Response Time"].min()
    response_stats=response_stats.sort_values(by="Standard Deviation", ascending = False)
    response_stats=response_stats.transpose()

    return response_stats

############ Function Call ############

insight5()
```

Data Visualization 3





```
def visual3():
    df = pd.read_csv("cleaned_response_times.csv")
    fig = px.box(df, x = "Borough", y = "Incident Response Time", color = "Borough", labels = {"Borough": "Borough","I
    return fig.show()
```

This visualization substantiates insight 4 and insight 5: The measures of center and spread are shown through the median and overall range of the data for each borough.

Overall Results

- There are indicators in our cleaned data that show that there is a difference between EMS response times in high-income boroughs versus low-income boroughs.

Conclusions & Takeaways

- Manipulating data into similar Data Frames using Pandas, numpy, etc., cleans data so that it can be cross referenced and compared with other datasets.

Challenges

- Parsing through our dataset with ~25M rows as Excel didn’t show all the rows of data
  - We had to break the dataset up into chunks of data and only include the data relevant to our hypothesis
- Code regarding regression for predictive analytics within the HTML dataset
  - Using different forms of linear regression in our code