In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
```

```
---------------------------------------------------------------------------
--
KeyboardInterrupt                         Traceback (most recent call las
t)
<ipython-input-1-3de8ca4200a6> in <module>()
      1 import numpy as np
----> 2 import pandas as pd
      3 import matplotlib.pyplot as plt
      4 import math

/anaconda3/lib/python3.6/site-packages/pandas/__init__.py in <module>()
     56
     57 from pandas.util._print_versions import show_versions
---> 58 from pandas.io.api import *
     59 from pandas.util._tester import test
     60 import pandas.testing

/anaconda3/lib/python3.6/site-packages/pandas/io/api.py in <module>()
      5 # flake8: noqa
      6
----> 7 from pandas.io.parsers import read_csv, read_table, read_fwf
      8 from pandas.io.clipboards import read_clipboard
      9 from pandas.io.excel import ExcelFile, ExcelWriter, read_excel

/anaconda3/lib/python3.6/site-packages/pandas/io/parsers.py in <module>()
     43
     44 import pandas._libs.lib as lib
---> 45 import pandas._libs.parsers as parsers
     46 from pandas._libs.tslibs import parsing
     47

pandas/_libs/parsers.pyx in init pandas._libs.parsers()

/anaconda3/lib/python3.6/importlib/_bootstrap.py in _find_and_load(name,
 import_)

KeyboardInterrupt:
```

Preparing Air Quality Data

First, I will read in the dataset on air quality in Oakland that was obtained from Google and modified with the help of Fengyang Lin (she added a grid id variable that signals what grid each data point was collected in and latitude and longitude bounds for each grid).

In [ ]:
```
air_quality_with_grid_data = pd.read_csv('grid_oak201604_05.csv')
```

In [ ]:
```
air_quality_with_grid_data.head()
```

Now, I will get descriptive statistics such as number of observations, mean, standard deviation, minimum and maximum values, and percentiles for different variables in the dataset.

In [ ]:
```
air_quality_with_grid_data.describe()
```

I group all the data points by which grid they belong to, and aggregate across all readings over time to get average statistics on NO2, NO, and BC levels.

In [ ]:
```
air_quality_data_grouped_by_grid = air_quality_with_grid_data.groupby('id')
avg_no2_per_grid = list(air_quality_data_grouped_by_grid["NO2"].agg(np.mean)
avg_no_per_grid = list(air_quality_data_grouped_by_grid["NO"].agg(np.mean))
avg_bc_per_grid = list(air_quality_data_grouped_by_grid["BC"].agg(np.mean))
lat_for_each_grid = []
lon_for_each_grid = []
grid_lon_lower_bound = list(air_quality_data_grouped_by_grid["xmin"].agg(np.
grid_lon_upper_bound = list(air_quality_data_grouped_by_grid["xmax"].agg(np.
grid_lat_lower_bound = list(air_quality_data_grouped_by_grid["ymin"].agg(np.
grid_lat_upper_bound = list(air_quality_data_grouped_by_grid["ymax"].agg(np.

grid_ids = list(air_quality_with_grid_data.groupby('id').groups.keys())

for grid_id in grid_ids:

    data_for_grid_id = air_quality_with_grid_data[air_quality_with_grid_data

    data_for_grid_id_xmin = data_for_grid_id.iloc[0, 13]
    data_for_grid_id_xmax = data_for_grid_id.iloc[0, 14]
    center_lon = (data_for_grid_id_xmin + data_for_grid_id_xmax) / 2
    lon_for_each_grid.append(center_lon)

    data_for_grid_id_ymin = data_for_grid_id.iloc[0, 15]
    data_for_grid_id_ymax = data_for_grid_id.iloc[0, 16]
    center_lat = (data_for_grid_id_ymin + data_for_grid_id_ymax) / 2
    lat_for_each_grid.append(center_lat)

lat_lon = pd.DataFrame({"Grid Latitude Center": lat_for_each_grid,
                        "Grid Longitude Center": lon_for_each_grid
                       })

grid_bounds = pd.DataFrame({"Grid Latitude Lower Bound": grid_lat_lower_boun
                            "Grid Latitude Upper Bound": grid_lat_upper_boun
                            "Grid Longitude Lower Bound": grid_lon_lower_bou
                            "Grid Longitude Upper Bound": grid_lon_upper_bou
                           })

emissions_levels = pd.DataFrame({"NO2": avg_no2_per_grid,
                                 "NO": avg_no_per_grid,
                                 "BC": avg_bc_per_grid
                                })

air_quality_aggregated = lat_lon.join(grid_bounds).join(emissions_levels)
```

In [ ]:
```
air_quality_aggregated.head(14)
```

Now, I will get descriptive statistics such as number of observations, mean, standard deviation, minimum and maximum values, and percentiles for different variables in the aggregated dataset.

In [ ]:
```
air_quality_aggregated.describe()
```

For this part, I'll normalize the NO2, NO, and BC values to compute an aggregated air pollution index.

In [ ]:
```
NO2_mean = air_quality_aggregated["NO2"].mean()
NO2_std = air_quality_aggregated["NO2"].std()
NO_mean = air_quality_aggregated["NO"].mean()
NO_std = air_quality_aggregated["NO"].std()
BC_mean = air_quality_aggregated["BC"].mean()
BC_std = air_quality_aggregated["BC"].std()

pollution_index = (air_quality_aggregated["NO2"] - NO2_mean) / (NO2_std) + (
air_quality_aggregated["Pollution Index"] = pollution_index
```

In [ ]:
```
air_quality_aggregated = air_quality_aggregated.sort_values(by="Pollution In
```

Let's sort the grids by the amount of air pollution, largest to smallest.

In [ ]:
```
air_quality_aggregated.head(14)
```

In [ ]:
```
air_quality_aggregated.to_csv(path_or_buf = "/Users/ryanlim/Desktop/Air_Qual
```

Let's consider the 5 blocks with the highest pollution indexes our air air pollution hotspots.

In [ ]:
```
air_pollution_hotspots = air_quality_aggregated.iloc[range(5), :]
```

In [ ]:
```
air_pollution_hotspots.to_csv(path_or_buf = "/Users/ryanlim/Desktop/Air_Poll
```

Preparing Public Transportation Data

For this part, I will get data about AC Transit bus stop locations.

In [ ]:
```
ac_transit_stops = pd.read_csv('AC_Transit_Stops.txt')
```

In [ ]:
```
ac_transit_stops.head()
```

I'll get the important variables and reformat the dataset.

In [ ]:
```
ac_transit_stops = ac_transit_stops.filter(items=['stop_name', 'stop_lat', '
ac_transit_stops.columns = ["Stop Name", "Latitude", "Longitude"]
ac_transit_stops.head()
```

In [ ]:
```
ac_transit_stops.shape
```

For this part, I will get data about BART train stop locations.

In [ ]:
```
bart_stops = pd.read_csv('BART_Stops.txt')
```

In [ ]:
```
bart_stops.head()
```

I'll get the important variables and reformat the dataset.

In [ ]:
```
bart_stops = bart_stops.filter(items=["stop_name", "stop_lat", "stop_lon"])
bart_stops.columns = ["Stop Name", "Latitude", "Longitude"]
bart_stops.head(10)
```

In [ ]:
```
bart_stops.shape
```

Preparing Block Level Data on Median Household Income

For this part, I'll get data on median household incomes for blocks in Alameda county from the 2010 U.S. Census.

In [ ]:
```
block_level_group_data = pd.read_csv('ACS_11_5yr_bg_alameda.csv')
```

In [ ]:
```
block_level_group_data.head()
```

In [ ]:
```
block_level_group_data.shape
```

I'll get the important variables and reformat the dataset.

In [ ]:
```
block_level_group_lat_lon_income = block_level_group_data.filter(items=['INT
block_level_group_lat_lon_income.columns = ["Latitude", "Longitude", "Median
block_level_group_lat_lon_income.head()
```

In [ ]:
```
block_level_group_lat_lon_income.shape
```

In [ ]:
```
block_level_group_lat_lon_income.to_csv(path_or_buf = "/Users/ryanlim/Deskto
```

Calculating Statistics

For this part, I'll calculate statistics for areas with different levels of pollution.

In [ ]:
```
air_quality_aggregated_ascending = air_quality_aggregated.sort_values(by="Po
num_grids = 14
median_household_income_comparison_lat = block_level_group_lat_lon_income["I
median_household_income_comparison_lon = block_level_group_lat_lon_income["I
```

In [ ]:
```
air_quality_aggregated_ascending.head(14)
```

For each grid, I use a distance metric to determine the median household income of the area surrounding the center of the grid, as well as the number of AC transit bus stops and BART train stops within the boundaries of the grid.

In [ ]:
```
_household_income = []
_transit_stops_nearby = []
rt_stops_nearby = []

.n range(14):

.d_lat = air_quality_aggregated_ascending.iloc[i,0]
.d_lon = air_quality_aggregated_ascending.iloc[i,1]

.d_lat_lower_bound = air_quality_aggregated_ascending.iloc[i,2]
.d_lat_upper_bound = air_quality_aggregated_ascending.iloc[i,3]
.d_lon_lower_bound = air_quality_aggregated_ascending.iloc[i,4]
.d_lon_upper_bound = air_quality_aggregated_ascending.iloc[i,5]

stance_metric = np.sqrt(((68.99 / 53.06) * (median_household_income_compariso
_distance_index = distance_metric.idxmin()

dian = block_level_group_lat_lon_income.iloc[min_distance_index,2]
dian_household_income.append(median)

_transit_stops_closeby = ac_transit_stops[(ac_transit_stops["Latitude"] >= gr
_ac_transit_stops_closeby = len(ac_transit_stops_closeby.index)
_ac_transit_stops_nearby.append(num_ac_transit_stops_closeby)

rt_stops_closeby = bart_stops[(bart_stops["Latitude"] >= grid_lat_lower_bound
_bart_stops_closeby = len(bart_stops_closeby.index)
_bart_stops_nearby.append(num_bart_stops_closeby)

tics = [("Grid Latitude Center", air_quality_aggregated_ascending["Grid Latit
        ("Grid Longitude Center", air_quality_aggregated_ascending["Grid Long
        ("Grid Latitude Lower Bound", air_quality_aggregated_ascending["Grid
        ("Grid Latitude Upper Bound", air_quality_aggregated_ascending["Grid
        ("Grid Longitude Lower Bound", air_quality_aggregated_ascending["Grid
        ("Grid Latitude Upper Bound", air_quality_aggregated_ascending["Grid
        ("Pollution Index", air_quality_aggregated_ascending["Pollution Index
        ('Median Household Income', median_household_income),
        ('Number of AC Transit Stops Within Grid', num_ac_transit_stops_nearb
        ('Number of BART Stops Within Grid', num_bart_stops_nearby),
      ]

tics_dataframe = pd.DataFrame.from_items(statistics)
```

In [ ]:
```
air_pollution_hotspots_statistics = statistics_dataframe.iloc[-5:,:]
air_pollution_hotspots_statistics = air_pollution_hotspots_statistics.sort_v
air_pollution_hotspots_statistics.head()
```

In [ ]:
```
air_pollution_hotspots_statistics_values = air_pollution_hotspots_statistics
```

In [ ]:
```
air_pollution_hotspots_statistics.describe()
```

In [ ]:
```
statistics_dataframe.head(14)
```

In [ ]:
```
statistics_dataframe.describe()
```