Health

In [ ]:
```
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

import statsmodels.formula.api as smf
import numpy as np
%matplotlib inline

pd.set_option('display.max_columns', None)
```

In [ ]:
```
data = pd.read_csv("Health_Care_Coverage.csv")
data

sns.scatterplot(x= "Kaiser Coverage", y = "share", data = data)
```

In [ ]:
```
data2 = pd.read_csv("Patient_to_Clinician_Ratios.csv")
data2

g=sns.catplot(x= "Geography",y="Patient to Primary Care Physician Ratio", kind ="bar",data=d
ata2)
g.fig.set_size_inches(18,6)
```

In [ ]:
```
data4= pd.read_csv("Social Needs.csv")
data4
```

In [ ]:
```
deaths = pd.read_csv("NYC_Leading_Causes_of_Death.csv")
deaths

deaths.groupby("Leading Cause").mean()

deaths.groupby("Leading Cause").mean()["Year"].plot(kind= "bar")
```

Wealth

In [ ]:
```
#libraries
import matplotlib.pyplot as plt
import pandas as pd
import statsmodels.formula.api as smf
import seaborn as sns
import numpy as np

%matplotlib inline
```

In [ ]:
```
#import statment
acs = pd.read_csv("acs_data2.csv", skiprows = 1)
pd.set_option('display.max_columns', None)
acs
```

In [ ]:
```
#slice
acs19 = acs[["id",
       "Geographic Area Name",
       "Percent!!EMPLOYMENT STATUS!!Civilian labor force!!Unemployment Rate",
       "Estimate!!INCOME AND BENEFITS (IN 2019 INFLATION-ADJUSTED DOLLARS)!!Total household
s!!Median household income (dollars)"
       ]].copy()
acs19
```

In [ ]:
```
#rename columns
acs19.rename(columns = {"id":"id#",
                          "Geographic Area Name": "area",
    "Percent!!EMPLOYMENT STATUS!!Civilian labor force!!Unemployment Rate" : "unemployment_ra
te(%)",
    "Estimate!!INCOME AND BENEFITS (IN 2019 INFLATION-ADJUSTED DOLLARS)!!Total households!!M
edian household income (dollars)" : "med_income",
                           },
               inplace = True)
```

In [ ]:
```
#view data
acs19
```

In [ ]:
```
#bivariate graph: borough and unemployment rate
g=sns.catplot(x = "area", y = "unemployment_rate(%)", kind = "bar", data = acs19)
g.fig.set_size_inches(18,6)
```

In [ ]:
```
#bivariate graph: borough and median income
g=sns.catplot(x = "area", y = "med_income", kind = "bar", data = acs19)
g.fig.set_size_inches(18,6)
```

In [ ]:
```
#adding race
acs2 = pd.read_csv("acs_data_nyc.csv", skiprows = 1)
pd.set_option('display.max_columns', None)
acs2
#richmond county is extracted after race is added
```

In [ ]:
```
#slice
acs2 = acs2[[ "Geographic Area Name",
       "Population Groups",
       "Estimate!!TOTAL NUMBER OF RACES REPORTED!!Total population",
       "Estimate!!EMPLOYMENT STATUS!!Population 16 years and over!!In labor force!!Civilian
 labor force!!Unemployed!!Unemployment Rate",
       "Estimate!!INCOME IN THE PAST 12 MONTHS (IN 2019 INFLATION-ADJUSTED DOLLARS)!!Househo
lds!!Median household income (dollars)"
           ]].copy()
acs2.head()
```

In [ ]:
```
#rename columns
acs2.rename(columns = {"Geographic Area Name": "Area", "Population Groups":"race",
    "Estimate!!TOTAL NUMBER OF RACES REPORTED!!Total population":"race_pop",
    "Estimate!!EMPLOYMENT STATUS!!Population 16 years and over!!In labor force!!Civilian lab
or force!!Unemployed!!Unemployment Rate":"unemployment_rate(%)",
    "Estimate!!INCOME IN THE PAST 12 MONTHS (IN 2019 INFLATION-ADJUSTED DOLLARS)!!Household
s!!Median household income (dollars)":"med_income"
                           },
   inplace = True)
```

In [ ]:
```
#rename values of race
race_map = {"White alone":"White",
           "Black or African American alone":"Black or African American",
           "American Indian and Alaska Native alone (300, A01-Z99)":"Native American and Nat
ive Alaskan",
           "Asian alone (400-499)":"Asian",
           "Native Hawaiian and Other Pacific Islander alone (500-599)": "Native Hawaiian an
d Pacific Islander",
           "Some other race alone":"Other"}
acs2["race"] = acs2["race"].map(race_map)
acs2
```

In [ ]:
```
#multivariate graph: borough, race, and unemployment rate
g=sns.catplot(x = "Area", y = "unemployment_rate(%)",hue = "race",kind = "bar", data = acs2)
g.fig.set_size_inches(18,6)
plt.xticks(rotation=45)
```

In [ ]:
```
#multivariate graph: borough, race, and median income
g=sns.catplot(x = "Area", y = "med_income",hue = "race", kind = "bar", data = acs2)
g.fig.set_size_inches(18,6)
plt.xticks(rotation=45)
```

In [ ]:
```
#mutlivariate graph: borough, race, and population
g=sns.catplot(x = "race", y = "race_pop", hue = "Area",kind = "bar", data = acs2)
g.fig.set_size_inches(18,6)
plt.xticks(rotation=45)
```

Education

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

In [ ]:
```
df = pd.read_csv("BCEQData/Demographic4.csv")
df = df[['GeographicAreaName','%StudentswithDisabilities','Year','Grade12','%Poverty']]
df = df.dropna()
df
```

In [ ]:
```
TheBronx = df['GeographicAreaName'] == 'BronxCountyNewYork'
Manhattan = df['GeographicAreaName'] == 'NewYorkCountyNewYork'
Queens = df['GeographicAreaName'] == 'QueensCountyNewYork'
Brooklyn = df['GeographicAreaName'] == 'KingsCountyNewYork'

TheBronx_df = df[TheBronx]
```

In [ ]:
```
from statsmodels.regression.linear_model import OLS
import statsmodels.api as sm
```

In [ ]:
```
x_vals = TheBronx_df['Grade12'].values
y_vals = TheBronx_df['%Poverty']


reg_model = OLS(y_vals, sm.add_constant(x_vals)).fit()
display(reg_model.summary())
```

In [ ]:
```
b0 = reg_model.params[0]
b1 = reg_model.params[1]
x_plot = np.linspace(np.min(df['Grade12']), np.max(df['Grade12']), 100)
```

In [ ]:
```
fig, axs = plt.subplots(figsize=(12,8))
axs.scatter(TheBronx_df['Grade12'], TheBronx_df['%Poverty'], c='blue', edgecolors='none', s=
30)
axs.plot(x_plot, x_plot*b1 + b0, color='red')
plt.title("The Number of Grade 12 Students vs. % Poverty", fontsize=20)
axs.set_xlabel("Number of Students", fontsize=18)
axs.set_ylabel("Rate of Poverty", fontsize=18)
axs.tick_params(labelsize=15)
plt.show()
# With a p statistic over 0.5, a statiscally signifcant regression line was not obtained
```

In [ ]:
```
df = pd.read_csv("BCEQData/Demographic4.csv")
df = df[['GeographicAreaName','%StudentswithDisabilities','Year','Grade12','%Poverty']]
df = df.dropna()
df
```

In [ ]:
```
BronxArea = df['GeographicAreaName'] == 'BronxCountyNewYork'
BrooklynArea = df ['GeographicAreaName']== 'KingsCountyNewYork'

area_df = df[BronxArea]
```

In [ ]:
```
from statsmodels.regression.linear_model import OLS
import statsmodels.api as sm
```

In [ ]:
```
x_vals = area_df['%StudentswithDisabilities'].values
y_vals = area_df['%Poverty']

reg_model = OLS(y_vals, sm.add_constant(x_vals)).fit()
display(reg_model.summary())
```

In [ ]:
```
b0 = reg_model.params[0]
b1 = reg_model.params[1]
x_plot = np.linspace(np.min(df['%StudentswithDisabilities']), np.max(df['%StudentswithDisabi
lities']), 100)
```

In [ ]:
```
fig, axs = plt.subplots(figsize=(12,8))
axs.scatter(TheBronx_df['%StudentswithDisabilities'], TheBronx_df['%Poverty'], c='blue', edg
ecolors='none', s=30)
axs.plot(x_plot, x_plot*b1 + b0, color='red')
plt.title("The Number of Students with Disabilities vs. % Poverty", fontsize=20)
axs.set_xlabel("Proportion of Students with Disabilities ", fontsize=18)
axs.set_ylabel("Rate of Poverty", fontsize=18)
axs.tick_params(labelsize=15)
plt.show()
# With a p statistic over less than 0.5, a statiscally signifcant regression line was obtain
ed
```

In [ ]:
```
from scipy import stats
corr = stats.pearsonr(TheBronx_df['%StudentswithDisabilities'], TheBronx_df['%Poverty'])
print('Correlation coefficient:', corr[0])
print('p-value:', corr[1])
```

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


df = pd.read_csv('BCEQData/GradRates.csv')
df = df[['Borough','Cohort Year','# Dropout']]
df = df.dropna()
df
# Are we interested in analysis of the most recent year's data or over a 5/10 year period?
```

In [ ]:
```
Bronx = df['Borough'] == 'Bronx'
Bronx_df = df[Bronx]
Bronx_Dropout = Bronx_df['# Dropout']
```

In [ ]:
```
fig, axs = plt.subplots(figsize=(12,8))
axs.hist(Bronx_Dropout, color="lightsteelblue", edgecolor="cadetblue")
plt.title("Dropout distribution of the Bronx", fontsize=20)
axs.set_xlabel("Number of students", fontsize=18)
axs.set_ylabel("Frequency", fontsize=18)
axs.tick_params(labelsize=15)
plt.show()
```

In [ ]:
```
TheBronx = df['Borough'] == 'Bronx'
Manhattan = df['Borough'] == 'Manhattan'
StatenIsland = df['Borough'] == 'Staten Island'
Queens = df['Borough'] == 'Queens'
Brooklyn = df['Borough'] == 'Brooklyn'


bronx_df = df[TheBronx]
manhattan_df = df[Manhattan]
statenIsland_df = df[StatenIsland]
queens_df = df[Queens]
brooklyn_df = df[Brooklyn]
```

In [ ]:
```
bronx_dropout = bronx_df['# Dropout']
manhattan_dropout = manhattan_df['# Dropout']
statenIsland_dropout = statenIsland_df['# Dropout']
queens_dropout = queens_df['# Dropout']
brooklyn_dropout = brooklyn_df['# Dropout']
```

In [ ]:
```
fig, axs = plt.subplots(figsize=(12,8))
axs.boxplot([bronx_dropout, manhattan_dropout, statenIsland_dropout, queens_dropout, brookly
n_dropout])
plt.title('Dropout distribution of the boroughs (2016)', fontsize=20)
axs.set_xticklabels(['The Bronx','Manhattan','Staten Island', 'Queens', 'Brooklyn'])
axs.set_ylabel('Students', fontsize=18)
axs.tick_params(labelsize=15)
plt.show()
```

In [ ]:
```
dropout = bronx_dropout
mean = np.mean(dropout)
print (mean)
```

In [ ]:
```
dropout = brooklyn_dropout
mean = np.mean(dropout)
print (mean)
```

In [ ]:
```
dropout = queens_dropout
mean = np.mean(dropout)
print(mean)
```

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


df = pd.read_csv('BCEQData/Demographic.csv')
df = df[['Borough','Year','Grade K','Grade 1','Grade 2', 'Grade 3','Grade 4','Grade 12']]
df = df.dropna()
df
```

In [ ]:
```
import matplotlib.pyplot as plt


data = [19579, 18598, 18071, 17680, 16863]

label = ['2016-2017','2017-2018','2018-2019','2019-2020','2020-2021']

fig, axs = plt.subplots(figsize=(10,6))
axs.bar(label, data, color=('blue'), alpha=0.4)
axs.set_title("Bronx Grade 1 Distribution", fontsize=20, fontweight="bold")
axs.set_xlabel("School Year", fontsize=14)
axs.set_ylabel("Number of Students", fontsize=14)
axs.tick_params(labelsize=14)

axs.spines['right'].set_visible(False)
axs.spines['top'].set_visible(False)

for i in range(len(data)):
    plt.text(i-0.3, data[i], str(data[i]), color='blue', size=16)

plt.grid(axis='y')

plt.show()
```

In [ ]:
```
import matplotlib.pyplot as plt


data = [13746, 22212, 17196, 18676, 4527]
label = ['Bronx','Brooklyn','Manhattan','Queens','Staten Island']

fig, axs = plt.subplots(figsize=(10,6))
axs.bar(label, data, color=('blue'), alpha=0.4)
axs.set_title("Borough Grade 12 Distribution 2020-2021", fontsize=20, fontweight="bold")
axs.set_xlabel("Borough", fontsize=14)
axs.set_ylabel("Number of Students", fontsize=14)
axs.tick_params(labelsize=14)

axs.spines['right'].set_visible(False)
axs.spines['top'].set_visible(False)

for i in range(len(data)):
    plt.text(i-0.3, data[i], str(data[i]), color='blue', size=16)

plt.grid(axis='y')

plt.show()
```

In [ ]:
```
import matplotlib.pyplot as plt

data = [14623, 14605, 14730, 13322, 13746]
label = ['2016-2017','2017-2018','2018-2019','2019-2020','2020-2021']

fig, axs = plt.subplots(figsize=(10,6))
axs.bar(label, data, color=('blue'), alpha=0.4)
axs.set_title("Bronx Grade 12 Distribution 2020-2021", fontsize=20, fontweight="bold")
axs.set_xlabel("School Year", fontsize=14)
axs.set_ylabel("Number of Students", fontsize=14)
axs.tick_params(labelsize=14)

axs.spines['right'].set_visible(False)
axs.spines['top'].set_visible(False)

for i in range(len(data)):
    plt.text(i-0.3, data[i], str(data[i]), color='blue', size=16)

plt.grid(axis='y')

plt.show()
```

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


df = pd.read_csv('BCEQData/Attendance.csv')
df = df[['Borough','Grade','# Chronically Absent']]
df = df.dropna()
df
```

In [ ]:
```
TheBronx = df['Borough'] == 'Bronx'
Manhattan = df['Borough'] == 'Manhattan'
StatenIsland = df['Borough'] == 'Staten Island'
Queens = df['Borough'] == 'Queens'
Brooklyn = df['Borough'] == 'Brooklyn'

grade = df['Grade'] == 'All Grades'

bronx_df = df[TheBronx & grade]
manhattan_df = df[Manhattan & grade]
statenIsland_df = df[StatenIsland & grade]
queens_df = df[Queens & grade]
brooklyn_df = df[Brooklyn & grade]
```

In [ ]:
```
bronx_absent = bronx_df['# Chronically Absent']
manhattan_absent = manhattan_df['# Chronically Absent']
statenIsland_absent = statenIsland_df['# Chronically Absent']
queens_absent = queens_df['# Chronically Absent']
brooklyn_absent = brooklyn_df['# Chronically Absent']
bronx_absent
```

In [ ]:
```
fig, axs = plt.subplots(figsize=(12,8))
axs.boxplot([bronx_absent, manhattan_absent, statenIsland_absent, queens_absent, brooklyn_ab
sent])
plt.title('Absenteeism distribution of the boroughs (2016)', fontsize=20)
axs.set_xticklabels(['The Bronx','Manhattan','Staten Island', 'Queens', 'Brooklyn'])
axs.set_ylabel('Students', fontsize=18)
axs.tick_params(labelsize=15)
plt.show()
```

In [ ]:
```
absent = bronx_df['# Chronically Absent']

mean = np.mean(absent)
print (mean)
```

In [ ]:
```
absent1 = brooklyn_absent = brooklyn_df['# Chronically Absent']

mean = np.mean(absent1)
print (mean)
```

In [ ]:
```
# Bronx maximum absenteeism
maximum = np.max(absent)
print(maximum)
```

In [ ]:
```
# Brooklyn maximum absenteeism
maximum = np.max(absent1)
print(maximum)
```

Environmental Equity

// The following code used for the environmental equity factors is written in R:

```
install.packages("ggplot")
install.packages("ggpubr")
install.packages("readxl")
install.packages("tidyverse")

library(readxl)
library(ggplot)
library(ggpubr)
library(tidyverse)
library(ggplot2)


nyc_data <- read_excel("FILEPATHNAME", sheet = "NYC")
bronx_data <- read_excel("FILEPATHNAME", sheet = "Bronx")

ggbarplot(bronx_data, x = "Bronx Community Board District", y = "City
Parks in District Per 1000 Residents (Acres)", fill ="steelblue")
ggbarplot(bronx_data, x = "Bronx Community Board District", y =
"Percent Parkland", fill ="steelblue")
ggbarplot(bronx_data, x = "Bronx Community Board District", y =
"Residents Within a 5 Minute Walk of a Park", fill ="steelblue")
ggbarplot(bronx_data, x = "Bronx Community Board District", y = "Park-
Related 311 Calls Per 1000 Residents", fill ="steelblue")
ggbarplot(bronx_data, x = "Bronx Community Board District", y = "City
Parks in District Per 1000 Children (Acres)", fill ="steelblue")
ggbarplot(bronx_data, x = "Bronx Community Board District", y = "City
Parks in District Per 1000 Seniors (Acres)", fill ="steelblue")
ggbarplot(bronx_data, x = "Bronx Community Board District", y = "Tree
Canopy Cover (Percentage)", fill ="steelblue")
ggbarplot(bronx_data, x = "Bronx Community Board District", y = "Air
Pollution (micrograms per m^3)", fill ="steelblue")
ggbarplot(bronx_data, x = "Bronx Community Board District", y = "Parks
Considered Acceptable for Cleanliness", fill ="steelblue")
ggbarplot(bronx_data, x = "Bronx Community Board District", y = "Parks
Considered Acceptable for Condition", fill ="steelblue")


//Next come the Bronx correlations, here is the format of the code,
and the factors tested can be changed simply by changing the fields x,
y, xlab, and ylab.


//This is an example of a possible correlation
//This line of code can be repurposed for every correlation
ggscatter(bronx_data, x = "Tree Canopy Cover (Percentage)", y = "Life
```

```
Expectancy (years)", add = "reg.line", conf.int = TRUE, cor.coef =
TRUE, cor.method = "pearson", xlab = "Tree Canopy Cover (Percentage)",
ylab = "Life Expectancy (years)")
ggscatter(bronx_data, x = "FIRST FACTOR", y = "SECOND FACTOR", add =
"reg.line", conf.int = TRUE, cor.coef = TRUE, cor.method = "pearson",
xlab = "X AXIS LABEL", ylab = "Y AXIS LABEL") //general form


//The same can be carried out for the citwide correlations


ggscatter(nyc_data, x = "City Parks Considered Acceptable for
Cleanliness", y = "Life Expectancy (years)", add = "reg.line",
conf.int = TRUE, cor.coef = TRUE, cor.method = "pearson", xlab = "City
Parks Considered Acceptable for Cleanliness (Percentage)", ylab =
"Life Expectancy (years)")
ggscatter(nyc_data, x = "FIRST FACTOR", y = "SECOND FACTOR", add =
"reg.line", conf.int = TRUE, cor.coef = TRUE, cor.method = "pearson",
xlab = "X AXIS LABEL", ylab = "Y AXIS LABEL") //general form

//This process can be repeated for each possible combination of
factors
//The resulting graphs may be downloaded directly from RStudio
```