CAP5768_Assignment1_Corbin_Adam

September 13, 2019

1 CAP 5768 - Data Science - Adam Corbin- Fall 2019

2 Assignment 1: The Python data science stack

2.0.1 Goals

- To get acquainted with Python and Jupyter notebooks.
- To acquire a basic understanding of the Python "data science stack" (NumPy, Pandas, Matplotlib).
- To have an early experience of manipulating, summarizing, and visualizing small datasets.
- To demonstrate the ability to write Python code to answer questions and test hypotheses based on the contents of those datasets.

2.0.2 Instructions

- This assignment is structured as a game, with three levels of increasing complexity.
- For each level, there will be some Python code to be written and questions to be answered.
- At the end, you should export your notebook to PDF format; it will "automagically" become your report.
- Submit the report (PDF), notebook (.ipynb file), and (optionally) link to the "live" version of your solution on Google Colaboratory via Canvas.
- The total number of points is 320 (plus up to 44 bonus points), distributed as follows: Level 1 (82 pts), Level 2 (70+ pts), Level 3 (132+ pts) and Conclusions (36 pts).

2.0.3 Important

- It is OK to attempt the bonus points, but please **do not overdo it!**
- Remember: this is an early exercise in exploring datasets; learning the syntax and "tricks" of Python, Jupyter notebooks, Numpy, Pandas, and Matplotlib; and writing code to test simple hypotheses and produce answers to simple questions that **you know you should be able to answer** because the answer can be pulled from the data.
- This is not (yet) the time to do sophisticated statistical analysis, train ML models, etc.

2.1 Level 1: Presidents of the USA

The Python code below will load a dataset containing the names of the first 44 presidents of the USA and their heights, available in the file *president_heights.csv*, which is a simple comma-separated list of labels and values.

[23]:
```
# Imports
import numpy as np
import pandas as pd
from pandas import DataFrame, Series

%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

from scipy.stats import pearsonr
```

[24]:
```
file = 'data/president_heights.csv'
presidents = pd.read_csv(file)
presidents
```

[24]:
```
    order                    name  height(cm)
0       1       George Washington         189
1       2              John Adams         170
2       3        Thomas Jefferson         189
3       4           James Madison         163
4       5            James Monroe         183
5       6       John Quincy Adams         171
6       7          Andrew Jackson         185
7       8        Martin Van Buren         168
8       9  William Henry Harrison         173
9      10              John Tyler         183
10     11           James K. Polk         173
11     12          Zachary Taylor         173
12     13        Millard Fillmore         175
13     14         Franklin Pierce         178
14     15          James Buchanan         183
15     16         Abraham Lincoln         193
16     17          Andrew Johnson         178
17     18        Ulysses S. Grant         173
18     19     Rutherford B. Hayes         174
19     20       James A. Garfield         183
20     21       Chester A. Arthur         183
21     23       Benjamin Harrison         168
22     25        William McKinley         170
23     26      Theodore Roosevelt         178
24     27     William Howard Taft         182
25     28          Woodrow Wilson         180
26     29       Warren G. Harding         183
```

```
27      30         Calvin Coolidge          178
28      31          Herbert Hoover          182
29      32   Franklin D. Roosevelt          188
30      33         Harry S. Truman          175
31      34    Dwight D. Eisenhower          179
32      35         John F. Kennedy          183
33      36       Lyndon B. Johnson          193
34      37           Richard Nixon          182
35      38             Gerald Ford          183
36      39            Jimmy Carter          177
37      40           Ronald Reagan          185
38      41       George H. W. Bush          188
39      42            Bill Clinton          188
40      43          George W. Bush          182
41      44            Barack Obama          185
```

2.2 Your turn! (8 points)

Write code to display the histogram (4 pts) of president's heights and compute summary statistics (4 pts):

- Mean height
- Standard deviation
- Minimum height, and
- Maximum height.

2.3 Solution

[25]:
```python
heigth_col = presidents["height(cm)"]
print()

mean_heigth = round(heigth_col.mean(),3)
std_heigth = round(heigth_col.std(),3)
print("Mean\t\t\t" + str(mean_heigth))
print("Standard Deviation \t" + str(std_heigth))
print("Max\t\t\t" + str(heigth_col.min()))
print("Max\t\t\t" + str(heigth_col.max()))
```

```

Mean                    179.738
Standard Deviation      7.016
Max                     163
Max                     193
```

2.4 Your turn! (18 points)

Write Python code to answer the following questions (make sure the messages displayed by your code are complete and descriptive enough):

1. Who was(were) the tallest president(s)?
2. Who was(were) the shortest president(s)?
3. How many presidents were 6′ tall or taller?

2.5 Solution

[26]:
```
tallest_presidents = presidents[presidents["height(cm)"] ==␣
 ↪presidents["height(cm)"].max()]
smallest_presidents = presidents[presidents["height(cm)"] ==␣
 ↪presidents["height(cm)"].min()]
#1 foot = 30.48 cm
heigth_threshold = 6 * 30.48
six_feet_or_taller = presidents[presidents["height(cm)"] >= heigth_threshold]
print("Tallest Presidents")
print(tallest_presidents)
print("-------------------")
print("Shortest Presidents")
print(smallest_presidents)
print("-------------------")
print("Presidents >= 6ft(" + str(heigth_threshold) + "cm)")
print(six_feet_or_taller)
```

```
Tallest Presidents
    order              name  height(cm)
15     16   Abraham Lincoln         193
33     36  Lyndon B. Johnson        193
-------------------
Shortest Presidents
   order           name  height(cm)
3      4  James Madison         163
-------------------
Presidents >= 6ft(182.88cm)
    order                  name  height(cm)
0       1     George Washington         189
2       3      Thomas Jefferson         189
4       5          James Monroe         183
6       7        Andrew Jackson         185
9      10            John Tyler         183
14     15        James Buchanan         183
15     16       Abraham Lincoln         193
19     20      James A. Garfield        183
20     21     Chester A. Arthur         183
26     29      Warren G. Harding        183
29     32  Franklin D. Roosevelt        188
32     35        John F. Kennedy        183
33     36      Lyndon B. Johnson        193
35     38           Gerald Ford         183
```

```
37      40         Ronald Reagan         185
38      41     George H. W. Bush         188
39      42          Bill Clinton         188
41      44          Barack Obama         185
```

This is an extremely small, simple and manageable dataset.
Let's use it to prove a silly hypotheses, for example:
"H1: Even-numbered presidents are, in average, taller than odd-numbered ones."

2.6 Your turn! (24 points)

Write Python code to test hypothesis H1.

2.7 Solution

[27]:
```
even_average_heigth = round(presidents[presidents["order"] % 2 ==␣
↪0]["height(cm)"].mean(),3)
odd_average_heigth = round(presidents[presidents["order"] % 2 ==␣
↪1]["height(cm)"].mean(),3)
if(even_average_heigth > odd_average_heigth):
    print("H1 hypothesis is corret, Even presidents("+str(even_average_heigth)␣
↪+ ") are taller than Odd presidents("+str(odd_average_heigth)+ ")")
elif(even_average_heigth < odd_average_heigth):
    print("H1 hypothesis is false, Odd presidents("+str(odd_average_heigth)+ ")␣
↪are taller than Even presidents("+str(even_average_heigth) + ")")
else:
    print("H1 hypothesis is false, Even and Odd presidents are about the same␣
↪heigth at("+str(even_average_heigth) + ")")
```

```
H1 hypothesis is false, Odd presidents(179.864) are taller than Even
presidents(179.6)
```

2.8 Question 1 (4 points)

Was hypothesis H1 confirmed or refuted?

2.9 Solution

The H1 is refuted since the Odd presidents heigths are on average 179.864cm and the Even presidents heigths are on average 179.6cm

---

Now it's time for you to come up with a different (but also silly) hypotheses, which we will call H2. **Be creative!**

2.10 Your turn! (28 points)

Write Python code to test hypothesis H2 (and some text to explain whether it was confirmed or not).

“H2: Presidents with more vowels are taller than ones with less vowels”

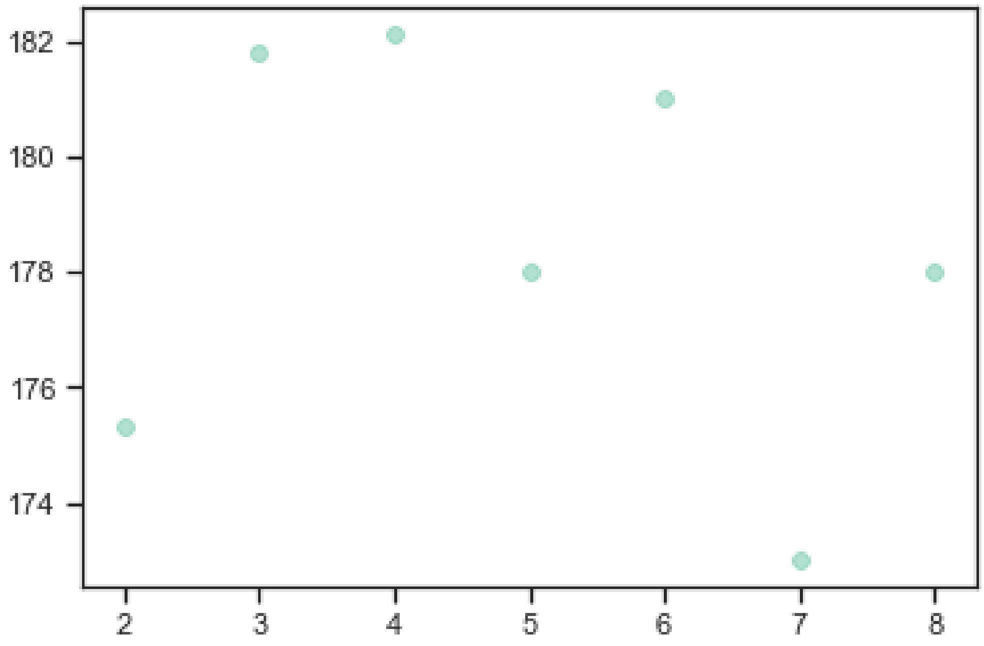
For consistency a the vouel set will be [a, e, i, o, u]

2.11 Solution

[28]:
```
# Get vouel count for each president, group them by vouel count, find if there␣
↪is a coorolation between the groups
president_vouel_count = []
president_vouel_count_list = []
vouels = ['a', 'e', 'i', 'o', 'u']

# Collecting the vouel counts & then adding them to the DataTable
for p in presidents["name"]:
    count = 0
    for v in vouels:
        count += p.count(v)
    president_vouel_count.append(count)
presidents["vouel count"] = president_vouel_count

# Grouping presidents by vouel count
# Index will be considered vouel count
average_heights_per_vouel_count = [0]*10
print("Vouel count|Average Heigth")
vouel_x = []
for vouel_count in range(0, 10):
    group_pres = presidents[presidents["vouel count"] == vouel_count]
    vouel_x.append(vouel_count)
    avg = ""
    if len(group_pres) > 0:
        avg = group_pres["height(cm)"].mean()
        average_heights_per_vouel_count[vouel_count] = avg
        avg = str(round(avg, 2))
#         print(group_pres) ## Uncomment this line to see the group of␣
↪presidents
    else:
        avg = "None"
    print(str(vouel_count) + "\t   |" + avg)

# Trimming off index 0,1,10 since they dont have any values
plt.scatter(vouel_x[2:9], average_heights_per_vouel_count[2:9], alpha=0.5)
plt.show()
```

```
Vouel count|Average Heigth
0          |None
```

```
1           |None
2           |175.33
3           |181.8
4           |182.12
5           |178.0
6           |181.0
7           |173.0
8           |178.0
9           |None
```



2.11.1 Evaluation of H2

In general more vowels doesnt not mean a taller president. The tallest average had 5 vowels where the shortest average was 8 vowels. There was a tendency that the lower number of vowels equated to a taller president but not in all cases. In the end H2 was refuted

---

2.12 Level 2: HR payroll

The Python code below will load a dataset containing the salaries and demographic data of more than 1000 employees of a hypothetical company, available in the file *salaries.csv*, which is a simple comma-separated list of labels and values.

[29]:
```python
salaries = pd.read_csv('data/salaries.csv')
print(salaries.shape)
print(salaries.count())
```

```
(1192, 6)
earn      1192
height    1192
sex       1192
ed        1192
age       1192
race      1192
dtype: int64
```

2.13 Question 2 (6 points)

What is the meaning of each of the six variables/features stored along the columns of the dataset?

2.14 Solution

2.15 Definitions below

earn - Income per year for an individual but not specified if its gross or net
height - The height of the person that I presume is measured inches
sex - Identification if the person is a male or female
ed - Education level assuming number of years where 12 is highschool, 16 is bachelor's degree and anything more is graduate
age - How old the individual is in years
race - identification of race if the individual gave it

Let's explore the dataset by plotting some graphs and displaying summary statistics.

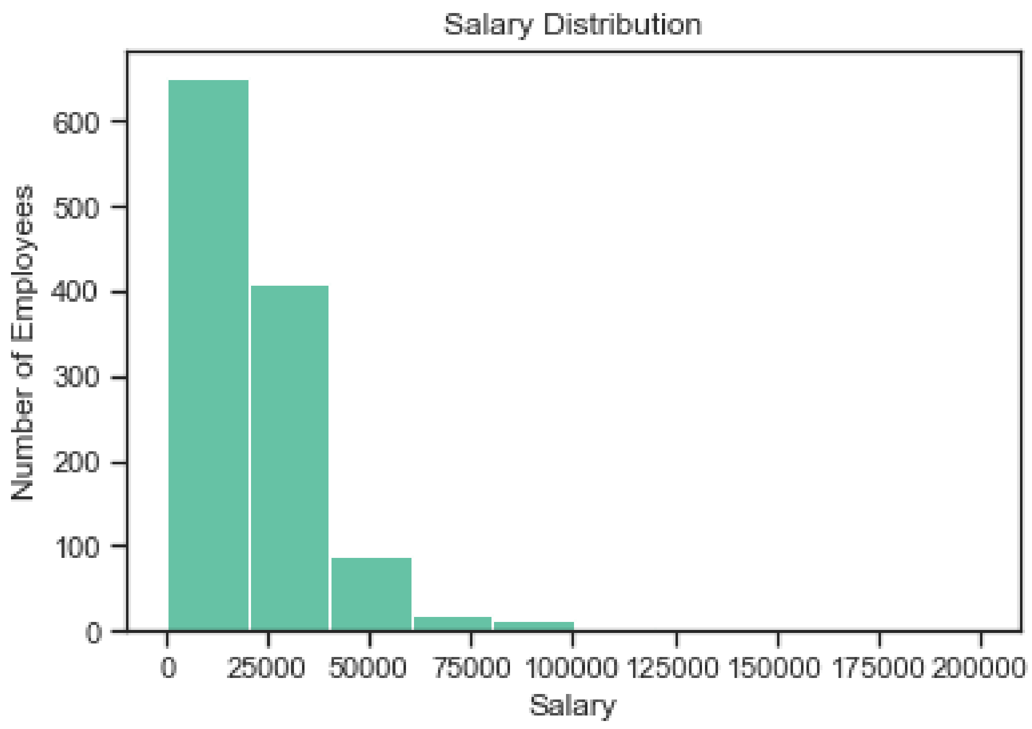
The code below should display: - Min, max, average, and median salary (global) - A histogram of salaries - A scatterplot correlating salaries and years of education - The (Pearson) correlation coefficient between the two variables.

This should help us get started.

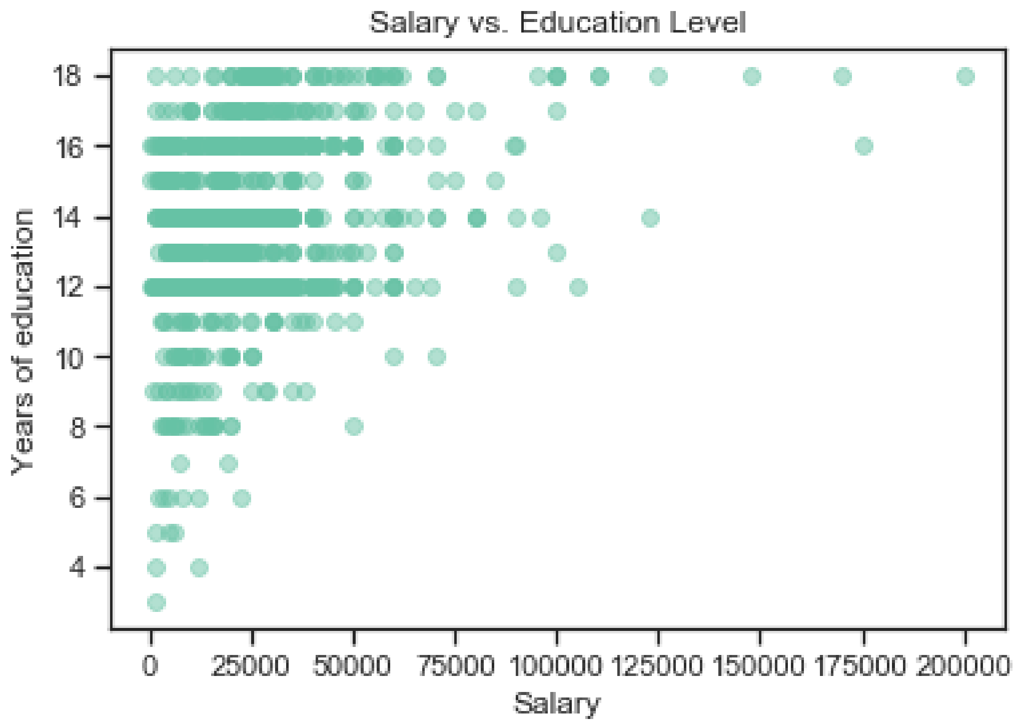
[30]:
```python
salary = np.array(salaries['earn'])
print("Salary statistics")
print("Minimum salary (global):", np.min(salary))
print("Maximum salary (global):", np.max(salary))
print("Average salary (global):", np.mean(salary))
print("Median  salary (global):", np.median(salary))
```

```
Salary statistics
Minimum salary (global): 200.0
Maximum salary (global): 200000.0
Average salary (global): 23154.773489932886
Median  salary (global): 20000.0
```

[31]:
```python
plt.hist(salary)
plt.title('Salary Distribution')
plt.xlabel('Salary')
plt.ylabel('Number of Employees');
```



[32]:
```
years = np.array(salaries['ed'])
plt.title('Salary vs. Education Level')
plt.xlabel('Salary')
plt.ylabel('Years of education');
plt.scatter(salary, years, alpha=0.5)
plt.show()
```



[33]:
```python
# Compute Pearson coefficient
from scipy.stats import pearsonr
corr, _ = pearsonr(salary,years)
print('Correlation coefficient: ',corr)
```

```
Correlation coefficient:  0.3399765246894846
```

The Pearson correlation coefficient (a value between -1 and 1) can be used to summarize the strength of the linear relationship between two data samples.

A simplified way to interpret the result is: - A value of 0 means no correlation - Values below -0.5 or above 0.5 indicates a notable (negative/positive) correlation

2.16 Your turn! (16+ points)

Write code to:

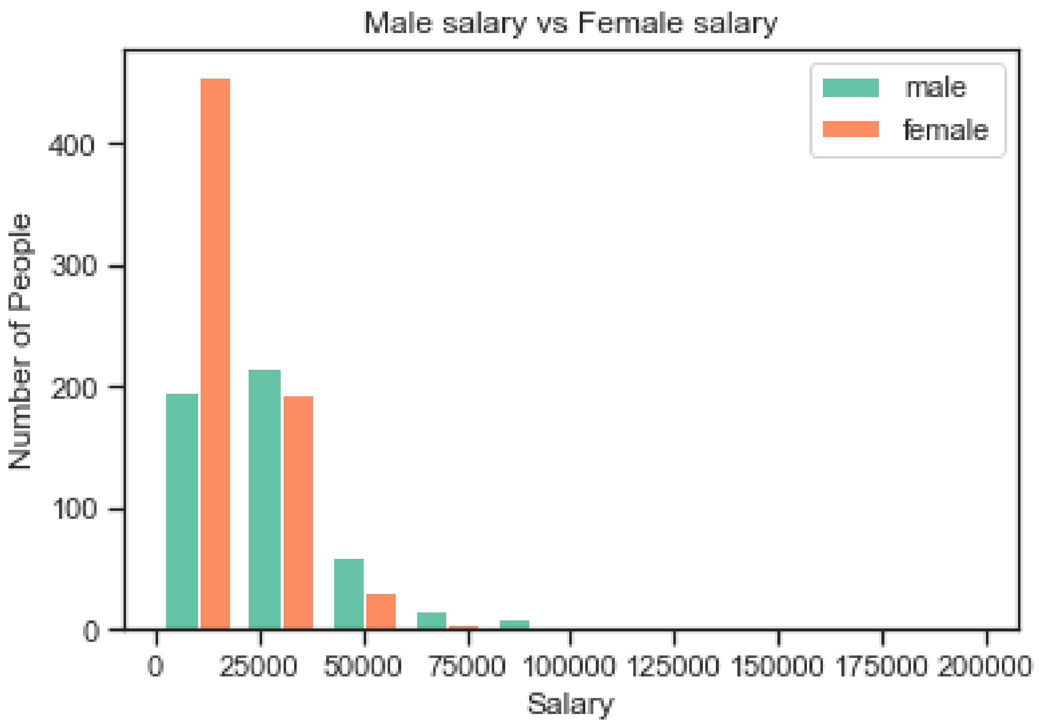
1. Display the total headcount and the number (and %) of male and female employees. (4 pts)
2. Compute and display the min, max, average, and median salary *per gender*. (12 pts)
3. (OPTIONAL) Plot meaningful graphs that could provide insight into the gender inequality (*if any is present*) associated with the salaries in the company. (<= 16 bonus points)

2.17 Solution

[34]:
```python
# ENTER YOUR CODE HERE
total_count = len(salaries)
males = salaries[salaries['sex'] == "male"]
male_count = len(males)
females = salaries[salaries['sex'] == "female"]
female_count = len(females)
print("--Head counts--")
print("|\tSex\t|\tCount\t|\t%\t|")
print("|\tmale\t|\t" + str(male_count) + "\t|\t" +
      str(round(male_count/total_count*100, 2))+"\t|")
print("|\tfemale\t|\t" + str(female_count) + "\t|\t" +
      str(round(female_count/total_count*100, 2))+"\t|")
print("----------")
print("Male statistics")
print("Minimum salary:", males["earn"].min())
print("Maximum salary:", males["earn"].max())
print("Average salary:", round(males["earn"].mean(), 2))
print("Median  salary:", males["earn"].median())
print("----------")
print("Female statistics")
print("Minimum salary:", females["earn"].min())
print("Maximum salary:", females["earn"].max())
print("Average salary:", round(females["earn"].mean(), 2))
print("Median  salary:", females["earn"].median())
print()
```

```
--Head counts--
|       Sex     |       Count   |       %       |
|       male    |       505     |       42.37   |
|       female  |       687     |       57.63   |
----------
Male statistics
Minimum salary: 1000.0
Maximum salary: 200000.0
Average salary: 29786.13
Median  salary: 25000.0
----------
Female statistics
Minimum salary: 200.0
Maximum salary: 123000.0
Average salary: 18280.2
Median  salary: 15000.0
```

[35]:
```
## Add in pretty graph for bonus points
# 2 histogram plots, but joined together
male_salary_array = np.array(males['earn'])
female_salary_array = np.array(females['earn'])

combined_2d_array = [male_salary_array,female_salary_array]
plt.title('Male salary vs Female salary')
plt.xlabel('Salary')
plt.ylabel('Number of People');
label = ["male", "female"]
plt.hist(combined_2d_array, histtype='bar', label=label)
plt.legend(loc='upper right');
plt.show()
```



As you can possibly tell by now, this dataset may help us test hypotheses and answer questions related to possible sources of inequality associated with the salary distribution: gender, age, race, height.

Let's assume, for the sake of argument, that the number of years of education should correlate well with a person's salary (this is clearly a weak argument and the plot and Pearson correlation coefficient computation above suggests that this is *not* the case) and that other suspiciously high (positive or negative) correlations could be interpreted as a sign of inequality.

At this point, formulate 3 different hypotheses that might suggest that the salary distribution is biased by factors such as ageism.

Call these hypotheses H3, H4, and H5.

Be creative, but realistic! These should be reasonable hypotheses, not silly ones!

2.18 Your turn! (48+ points)

Write Python code to test hypotheses H3, H4, and H5 (and some text to explain whether they were confirmed or not). Feel free to (also) use plots, but make your code independent of a human being interpreting those plots. (16 pts per hypothesis with additional <= 8 bonus points per hypotheses).

3 Hypotheses

3.1 H3 Age Vs salary

People who are older will earn more than younger people

3.2 H4 Race Vs salary

People who are white will earn more than other races

3.3 H5 Heigth Vs salary

People who are taller will earn more for both male and female

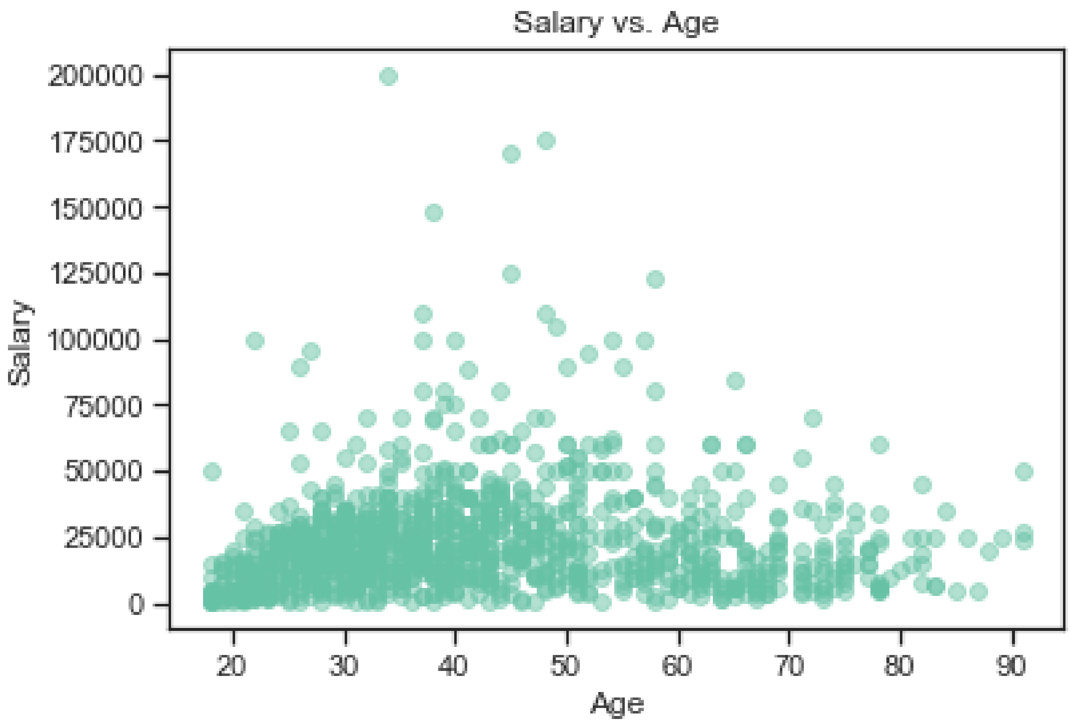
3.4 H6 Education vs Sex

Males have more education background than Females

3.5 Solution

[36]:
```python
# ENTER YOUR CODE HERE
print("H3")
ages = np.array(salaries['age'])
plt.title('Salary vs. Age')
plt.ylabel('Salary')
plt.xlabel('Age');
plt.scatter(ages, salary , alpha=0.5)
plt.show()
corr, _ = pearsonr(salary,ages)
print('Correlation coefficient: ',corr)
```

```
H3
```



```
Correlation coefficient:  0.08100297364190612
```

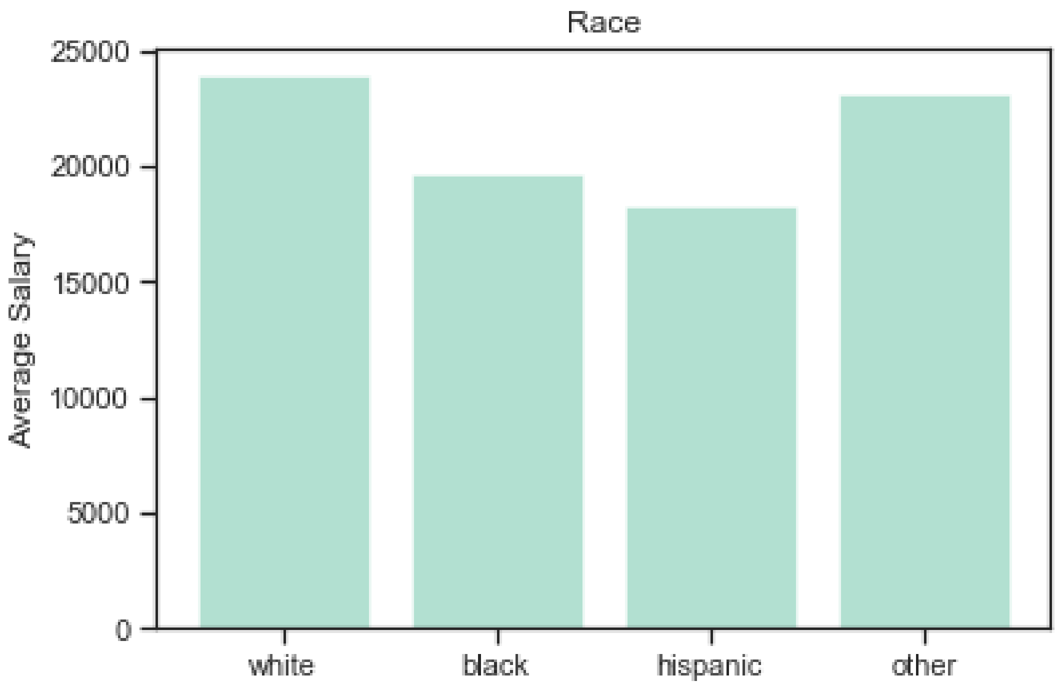
3.5.1 H3 evaluation

Based on the graph there seems to be a tendency the lower incomes cluster around the lower age and higher incomes for older people, but its pretty busy to definitely state that. The pearson correlation coefficient resulted in ~ 0.081 and since that is greater than 0.5 then its showing strengths of a positive correlation. Based on these results I will say we can confirm H3

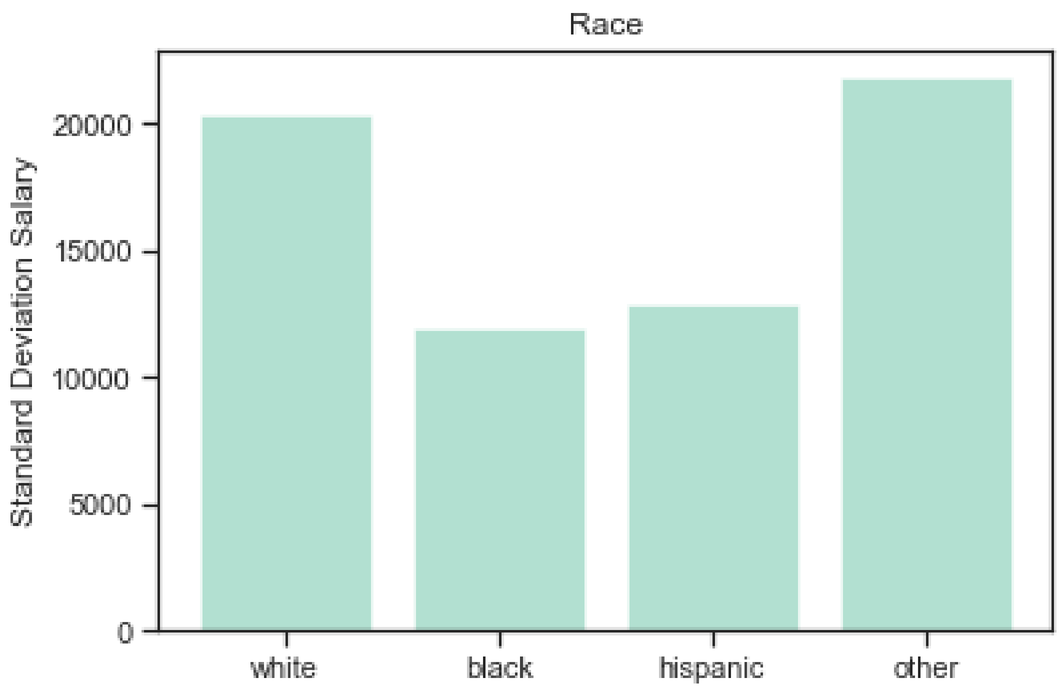
[37]:
```
print("H4")
races = np.array(salaries['race'])
total_count = len(races)
unique_races = list(set(races))
labels = []
# Loop over all the races, collect their salaries so we can have a bargraph
collected_race_earn_data = []
average_data = []
std_data = []
for race in unique_races:
    specific_race_earn_data = salaries[salaries['race'] == race]["earn"]
    print(race.capitalize() + " Statistics")
    size = len(specific_race_earn_data)
    print("Race count(percentage):", size,
          "(" + str(round(size/total_count * 100, 2)) + ")")
    print("Minimum salary:", specific_race_earn_data.min())
    print("Maximum salary:", specific_race_earn_data.max())
    average = round(specific_race_earn_data.mean(), 2)
    average_data.append(average)
    print("Average salary:", average)
    print("Median  salary:", specific_race_earn_data.median())
    std = round(specific_race_earn_data.std(), 2)
    std_data.append(std)
    print("Standard Deviation:", std)
    print()
    collected_race_earn_data.append(specific_race_earn_data)
```

```
H4
White Statistics
Race count(percentage): 989 (82.97)
Minimum salary: 200.0
Maximum salary: 200000.0
Average salary: 23882.47
Median  salary: 20000.0
Standard Deviation: 20374.73

Black Statistics
Race count(percentage): 112 (9.4)
Minimum salary: 600.0
Maximum salary: 60000.0
Average salary: 19624.21
Median  salary: 20000.0
Standard Deviation: 11947.74

Hispanic Statistics
Race count(percentage): 66 (5.54)
Minimum salary: 1400.0
Maximum salary: 55000.0
Average salary: 18263.64
Median  salary: 16000.0
Standard Deviation: 12832.53

Other Statistics
Race count(percentage): 25 (2.1)
Minimum salary: 3000.0
Maximum salary: 110000.0
Average salary: 23096.64
Median  salary: 20000.0
Standard Deviation: 21790.8
```

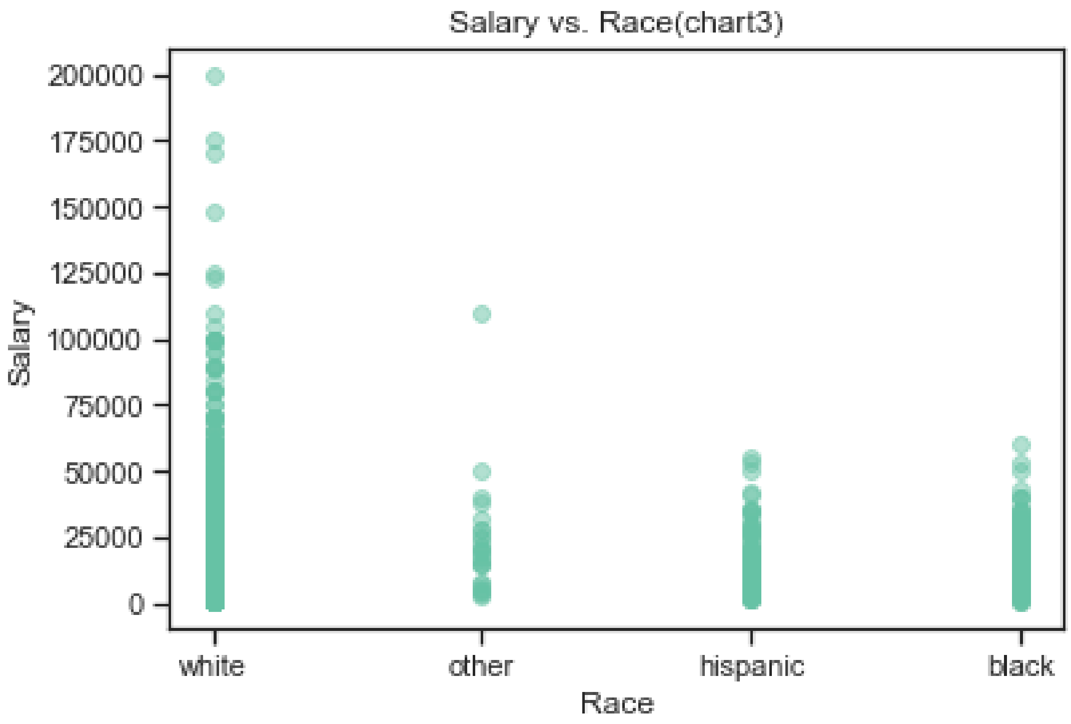
[38]:
```
plt.title('Average Salary vs. Race (chart1)')
y_pos = np.arange(len(unique_races))
plt.bar(y_pos, average_data, align='center', alpha=0.5)
plt.xticks(y_pos, unique_races)
plt.ylabel('Average Salary')
plt.title('Race')
plt.show()
```



[39]:
```
plt.title('Standard Deviation Salary vs. Race(chart2)')
y_pos = np.arange(len(unique_races))
plt.bar(y_pos, std_data, align='center', alpha=0.5)
plt.xticks(y_pos, unique_races)
plt.ylabel('Standard Deviation Salary')
plt.title('Race')
plt.show()
```



[40]:
```
plt.title('Salary vs. Race(chart3)')
plt.ylabel('Salary')
plt.xlabel('Race');
plt.scatter(races, salary , alpha=0.5)
plt.show()
```



3.5.2 H4 evaluation

First off taking a look at the percent spread between different races there was an 82% of the survey was white. We might state that this might be too small of a sample size with not a good amount of ratio between the different races

Moving on to the evaluation, we can see the average salary at chart1 by race goes from high to low as follows [white: 23882.47, other: 23096.64, black: 19624.21, Hispanic: 18263.64]. Looking at the averages this starts to lean towards H3 being plausible.

Taking a look at the Standard Deviation chart2 it shows a little different order [other: 21790.8, white:20374.73, Hispanic: 12832.53, black: 11947.74]. We see other is taking over the lead for this comparison.

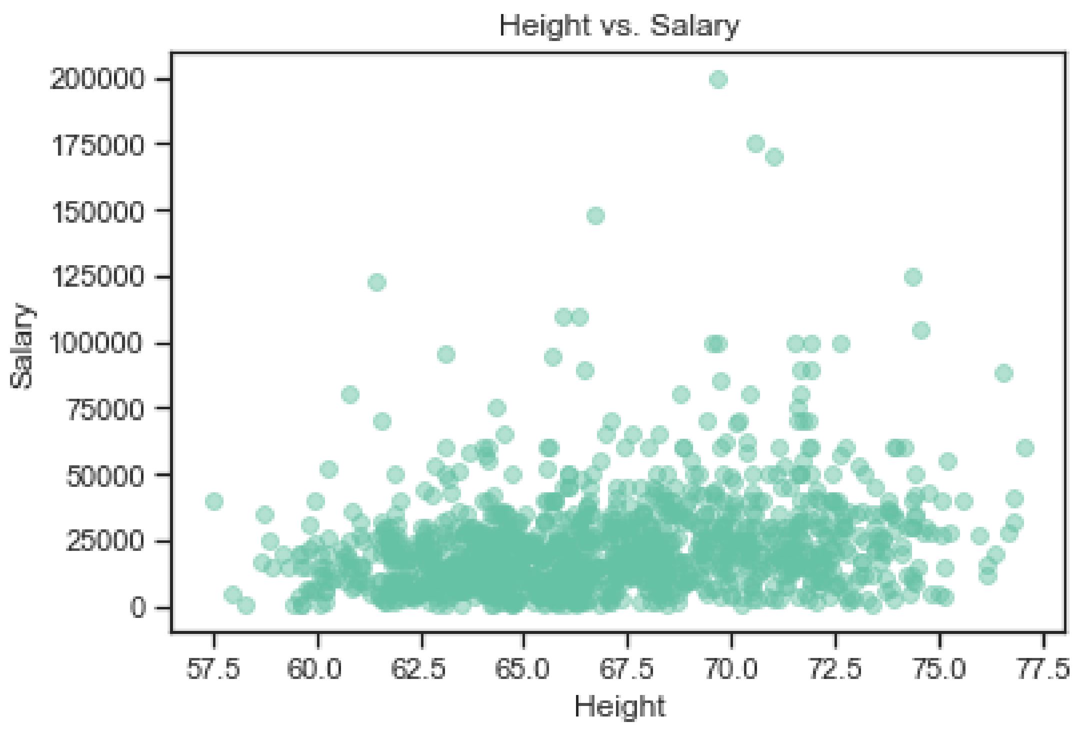
Observing the scatter plot chart3 of all the salaries split by race you can see the clustering of white to be higher than all of the other races.

Based on the data observed and with the caveat that the sample distribution was not good, I would say H4 is plausible. I dont think we can fully confirm or reject H3 based on the data. I would recommend further investigation to get a better dataset.

[41]:
```
print("H5")
ages = np.array(salaries['height'])
plt.title('Height vs. Salary')
plt.xlabel('Height')
plt.ylabel('Salary');
plt.scatter(ages, salary , alpha=0.5)
plt.show()
```

```
corr, _ = pearsonr(salary,ages)
print('Correlation coefficient: ',corr)
```

H5



```
Correlation coefficient:  0.24184814953626707
```

3.5.3 H5 evaluation

For H5 we are looking to see if there is any correlation between someones heigth and their salary. From the scatter plot there is clustering between a height of 60 to 72 and salary ~4,000 to 45,000, but this is just one large cluster. From the human eye I dont see any separation between the groups. The Pearson correlation coefficient is showing a result of 0.2418 which is in between -0.5 and 0.5 which shows an extremely weak correlation between the two attributes based on these results I can refute H5

[42]:
```
print("H6 - For Fun!")

male_edu = salaries[salaries["sex"] == "male"]["ed"]
print("Male count:", male_edu.count(), "percent:",
      round(male_edu.count() / len(salaries)*100, 2))
female_edu = salaries[salaries["sex"] == "female"]["ed"]
print("Female count:", female_edu.count(), "percent:",
      round(female_edu.count() / len(salaries)*100, 2))
```

```
def compute_basic_stats(data):
    print("Min", data.min())
    print("Max", data.max())
    print("Average:", round(data.mean(), 2))
    print("Median:", round(data.median(), 2))
    print("Standard Deviation:", round(data.std(), 2))


print("Male Stats:")
compute_basic_stats(male_edu)
print("-----------")
print("Female Stats:")
compute_basic_stats(female_edu)
```

```
H6 - For Fun!
Male count: 505 percent: 42.37
Female count: 687 percent: 57.63
Male Stats:
Min 4
Max 18
Average: 13.6
Median: 13.0
Standard Deviation: 2.5
-----------
Female Stats:
Min 3
Max 18
Average: 13.44
Median: 13.0
Standard Deviation: 2.36
```

3.5.4 H6 evaluation

Based on purely looking at the simple statistics results it appears that on average Males has .16 more years of education experience. I don't think I have enough analysis to truly confirm that males have more education and with the current analysis its very close. One thing to note is we do have a good spread with 57% female and 42% male, where as previous comparisons in race it was unbalanced.

3.6 Level 3: Fuel consumption

The Python code below will load a dataset containing fuel consumption data for ~400 vehicles produced in the 1970s and the 1980s along with some characteristic information associated with each model.

Here, *displacement* refers to a vehicle's engine size and the fuel efficiency is measured in miles per gallon (mpg).

See: https://archive.ics.uci.edu/ml/datasets/Auto+MPG for additional information.

[43]:
```
sns.set(style='ticks', palette='Set2')
%matplotlib inline
file = 'data/auto-mpg.data-original'
data = pd.read_csv(file,
                   delim_whitespace=True, header=None,
                   names=['mpg', 'cylinders', 'displacement', 'horsepower',␣
 ↪'weight', 'acceleration',
                          'model', 'origin', 'car_name'])
print(data.shape)
```

```
(406, 9)
```

[44]:
```
data.dropna(inplace=True)
data.head()
```

[44]:
```
    mpg  cylinders  displacement  horsepower  weight  acceleration  model  \
0  18.0        8.0         307.0       130.0  3504.0          12.0   70.0
1  15.0        8.0         350.0       165.0  3693.0          11.5   70.0
2  18.0        8.0         318.0       150.0  3436.0          11.0   70.0
3  16.0        8.0         304.0       150.0  3433.0          12.0   70.0
4  17.0        8.0         302.0       140.0  3449.0          10.5   70.0

   origin                   car_name
0     1.0  chevrolet chevelle malibu
1     1.0          buick skylark 320
2     1.0         plymouth satellite
3     1.0              amc rebel sst
4     1.0                ford torino
```

3.7 Question 3 (6 points)

What was the net effect of the line of code
   `data.dropna(inplace=True)`?

3.8 Solution

Based on the Pandas documentation the dropna method will remove rows that have missing values that are in the dataset. Passing the argument in place to True will update the data object. Source: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html

3.9 Your turn! (18+ points)

Write code to:

1. Count the number of 3- and 5-cylinder vehicles in the dataset, display the count, and discard those entries (rows). (12 pts)
2. Compute and display the min, max, and average fuel consumption (in mpg) for 4-, 6-, and 8-cylinder vehicles. (6 pts)
3. (OPTIONAL) Display the name of the most and least fuel efficient vehicles in the dataset (<= 4 points)

3.10 Solution

[45]:
```
print("#1")
print("Number of 3 cylinder Vehicles",len(data[data["cylinders"] == 3]))
print("Number of 5 cylinder Vehicles", len(data[data["cylinders"] == 5]))
print("Total number of cars before removal:", len(data))
data = data[data["cylinders"] != 3] # Removing 3 cylinder cars
data = data[data["cylinders"] != 5] # Removing 5 cylinder cars
print("Total number of cars after removal:", len(data))
```

```
#1
Number of 3 cylinder Vehicles 4
Number of 5 cylinder Vehicles 3
Total number of cars before removal: 392
Total number of cars after removal: 385
```

[46]:
```
print("\n#2")
def compute_stats(data, cylinder_count):
    loc_data = data[data["cylinders"] == cylinder_count]["mpg"]
    print("Fuel consumption(mpg) statistics data for cars␣
 ↪with",cylinder_count,"cylinders")
    print("Min", loc_data.min())
    print("Max", loc_data.max())
    print("Average:", round(loc_data.mean(),2))
    print("-------")
compute_stats(data, 4)
compute_stats(data, 6)
compute_stats(data, 8)
```

```
#2
Fuel consumption(mpg) statistics data for cars with 4 cylinders
Min 18.0
Max 46.6
Average: 29.28
-------
Fuel consumption(mpg) statistics data for cars with 6 cylinders
Min 15.0
Max 38.0
Average: 19.97
```

```
-------
Fuel consumption(mpg) statistics data for cars with 8 cylinders
Min 9.0
Max 26.6
Average: 14.96
-------
```

[47]:
```python
print("\n#3")
min_mpg = data["mpg"].min()
bad_cars = data[data["mpg"] == min_mpg]
# I already know that there is only 1 car, so just indexing right into it
bad_car_name = np.array(bad_cars["car_name"])[0]
bad_car_mpg = np.array(bad_cars["mpg"])[0]
print("Least fuel efficent vehicles:")
print(bad_car_name, "mpg:", bad_car_mpg)
max_mpg = data["mpg"].max()
good_cars = data[data["mpg"] == max_mpg]
# I already know that there is only 1 car, so just indexing right into it
good_car_name = np.array(good_cars["car_name"])[0]
good_car_mpg = np.array(good_cars["mpg"])[0]
print("Most fuel efficent vehicles:")
print(good_car_name, "mpg:", good_car_mpg)
```

```
#3
Least fuel efficent vehicles:
hi 1200d mpg: 9.0
Most fuel efficent vehicles:
mazda glc mpg: 46.6
```

This dataset may help us test hypotheses and answer questions related to fuel consumption. To get started: Which features of a vehicle correlate best with its mpg – *displacement*, *weight*, or *horsepower*?

3.11 Your turn! (48 points)

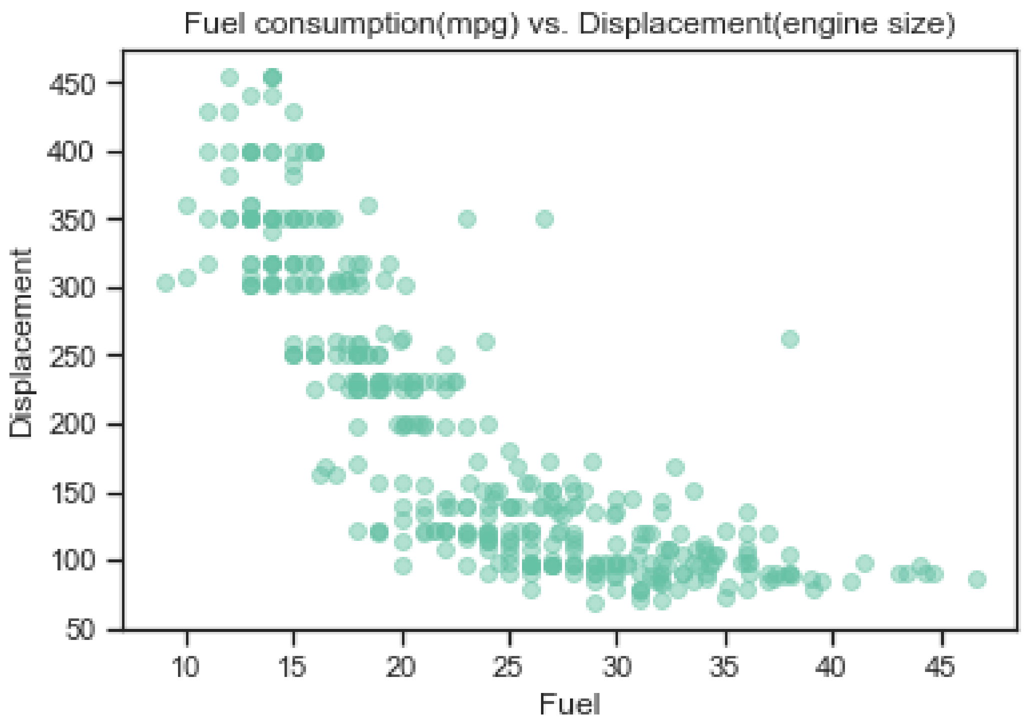
Write Python code to plot the relationship between:

1. Fuel consumption and displacement (engine size)
2. Fuel consumption and weight
3. Fuel consumption and horsepower (HP)

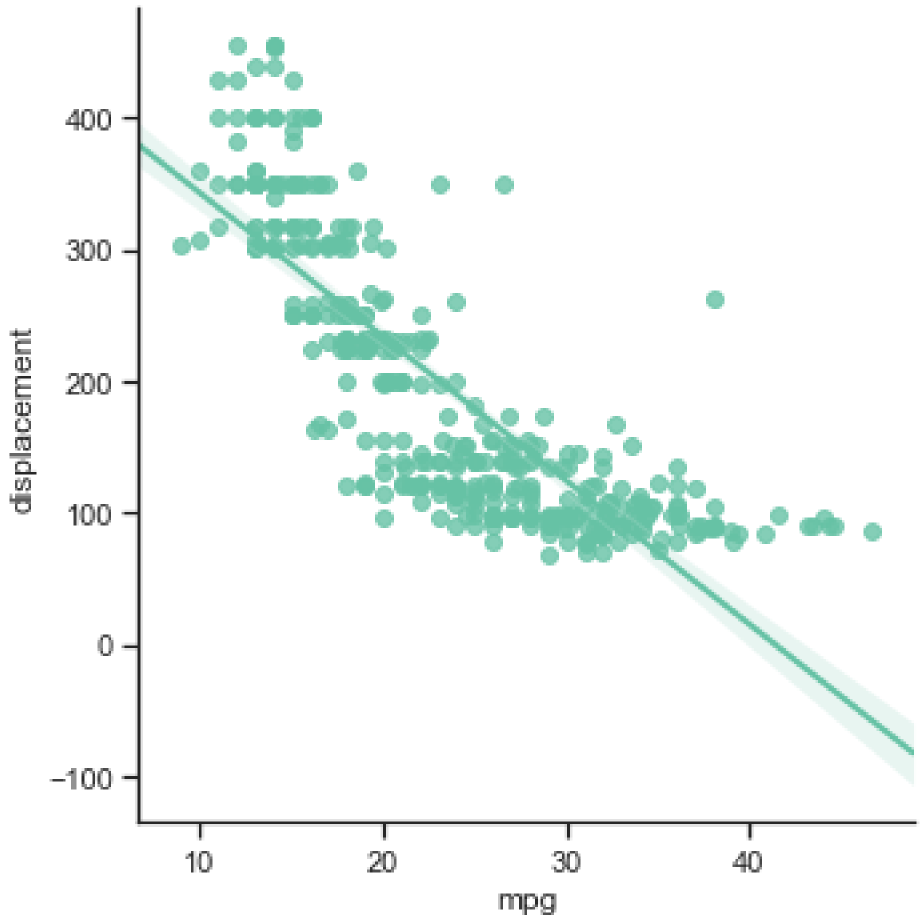
3.12 Solution

[48]:
```
fuel_data = data["mpg"]
print("#1")
displacement_data = data["displacement"]
plt.title('Fuel consumption(mpg) vs. Displacement(engine size)')
plt.xlabel('Fuel')
plt.ylabel('Displacement')
plt.scatter(fuel_data, displacement_data, alpha=0.5)
plt.show()
sns.lmplot("mpg","displacement",data)
corr, _ = pearsonr(fuel_data, displacement_data)
print('Correlation coefficient: ', corr)
```

```
#1
```

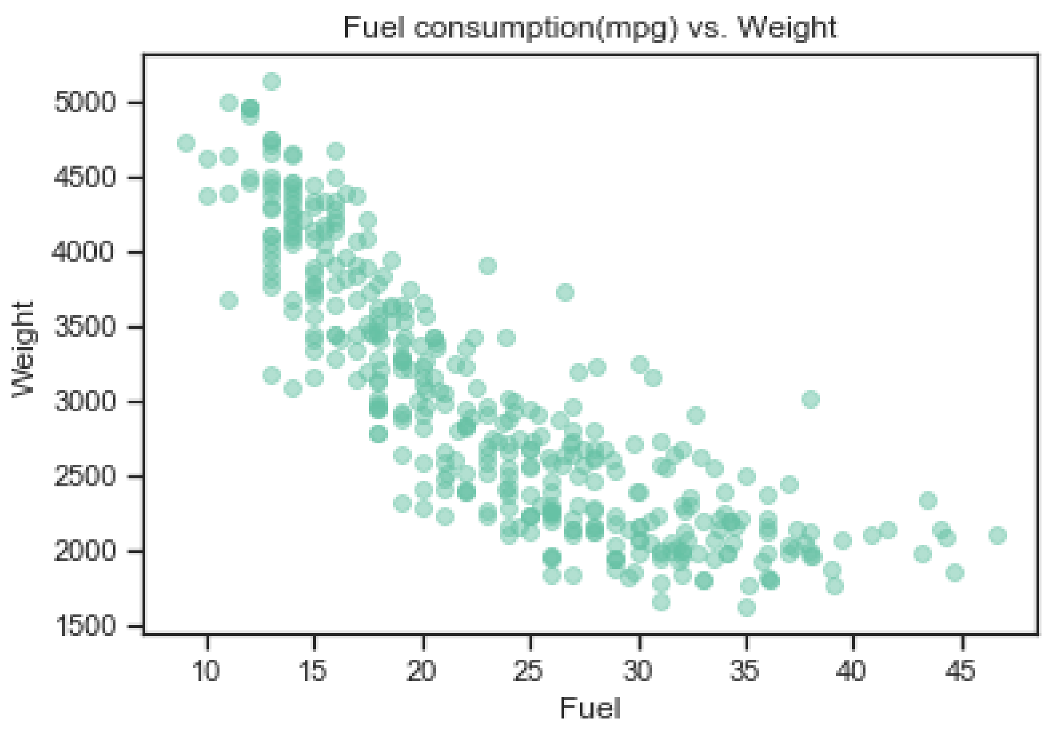


```
Correlation coefficient:  -0.817887362940059
```

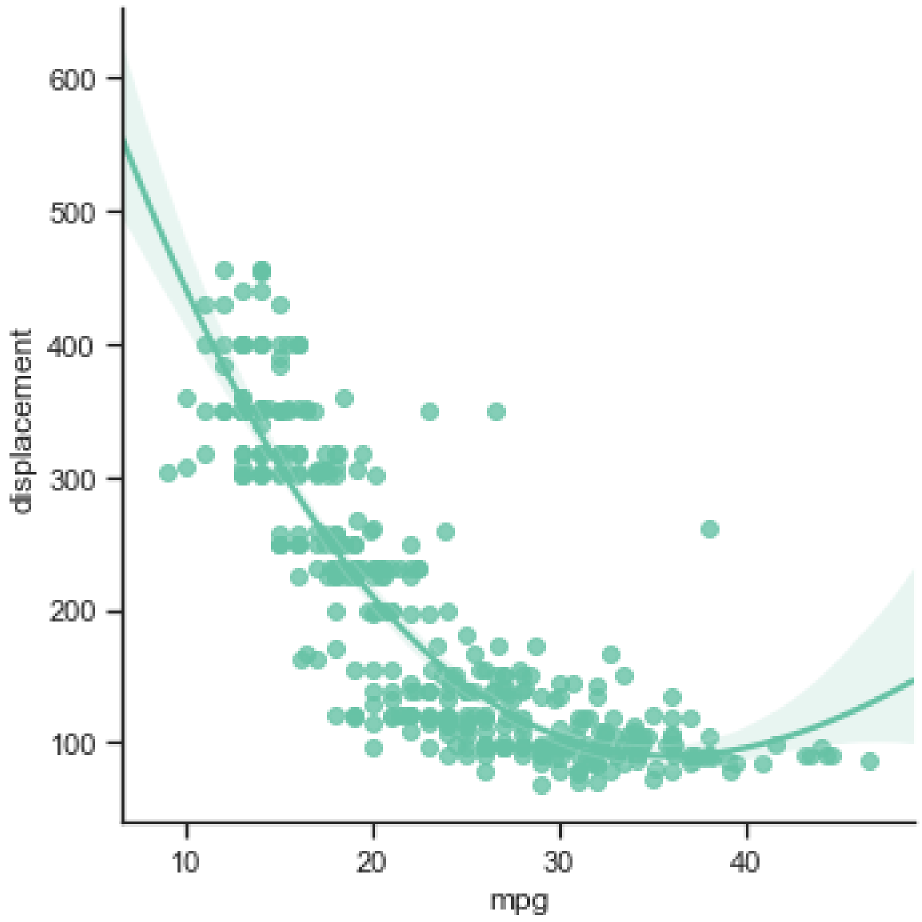


[49]:
```
print("#2")
weight_data = data["weight"]
plt.title('Fuel consumption(mpg) vs. Weight')
plt.xlabel('Fuel')
plt.ylabel('Weight')
plt.scatter(fuel_data, weight_data, alpha=0.5)
plt.show()
sns.lmplot("mpg","displacement", data, order = 3)
corr, _ = pearsonr(fuel_data, weight_data)
print('Correlation coefficient: ', corr)
```

```
#2
```

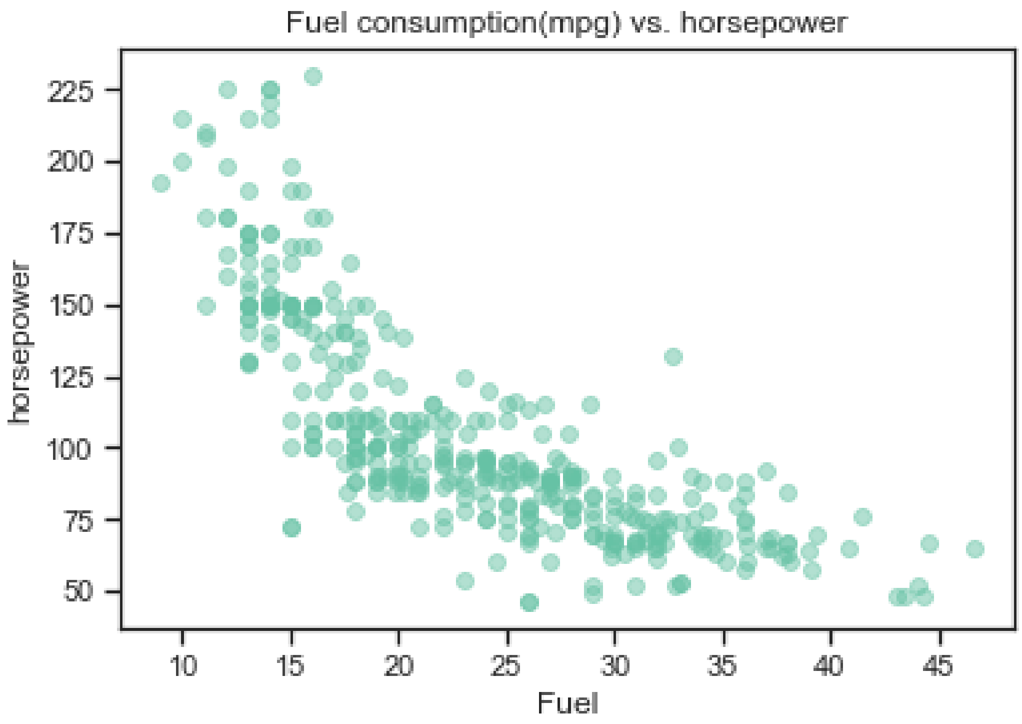


```
Correlation coefficient:  -0.8426809031318667
```

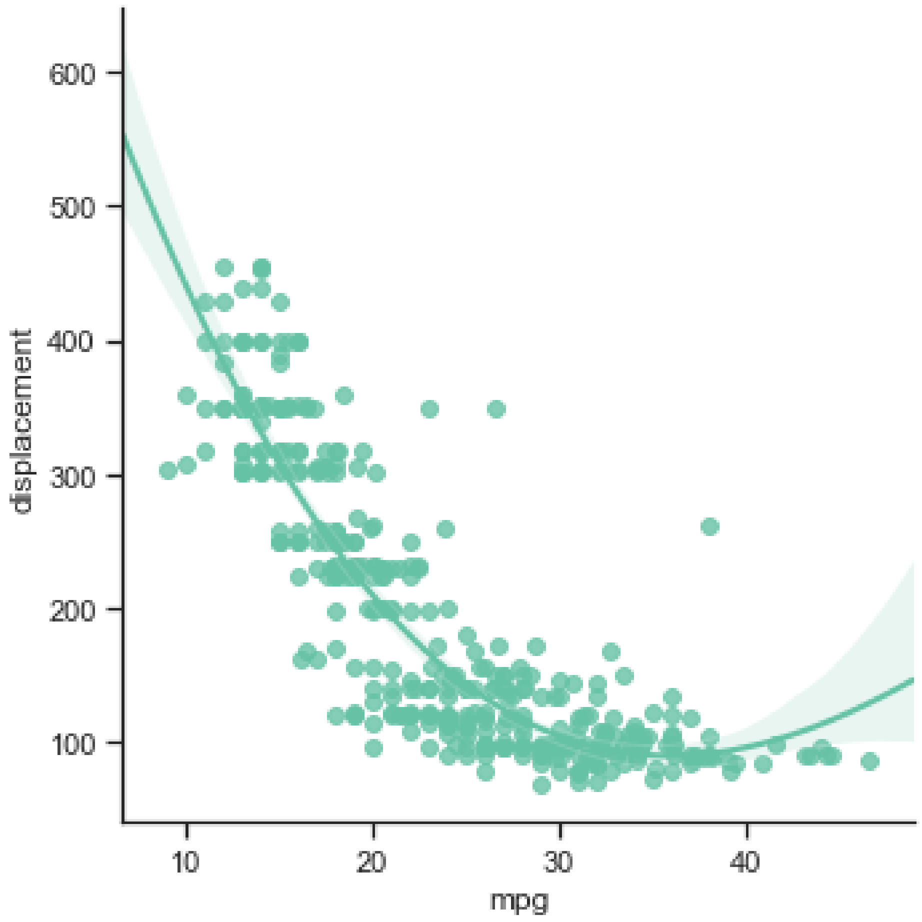


[50]:
```
print("#3")
horsepower_data = data["horsepower"]
plt.title('Fuel consumption(mpg) vs. horsepower')
plt.xlabel('Fuel')
plt.ylabel('horsepower');
plt.scatter(fuel_data, horsepower_data, alpha=0.5)
plt.show()
sns.lmplot("mpg","displacement",data, order=3)
corr, _ = pearsonr(fuel_data,horsepower_data)
print('Correlation coefficient: ',corr)
```

```
#3
```



```
Correlation coefficient:  -0.7802588157322387
```



3.13 Questions 4-6 (30 points, i.e. 10 points each)

For each of the three cases above, use the plots to answer the questions below:

4. Is there a correlation between mpg and displacement? If so: is it positive or negative? linear or non-linear?
5. Is there a correlation between mpg and weight? If so: is it positive or negative? linear or non-linear?
6. Is there a correlation between mpg and horsepower? If so: is it positive or negative? linear or non-linear?

3.14 Solution

4. Yes there is a negative correlation between mpg and displacement since the Pearson coefficient shows -0.81788. Since this falls past -0.5, that means is relatively strong correlation. The negative correlation means with the increase of displacement the MPG will decrease. The majority of the data tends to be linear and when using the Seaborn lmplot with order 0 the data follows the line pretty nicely. I would say mostly linear but has some non-linear characteristics.

5. Yes there is a negative correlation between mpg and weight since the Pearson coefficient shows -0.84268. Since this falls past -0.5, that means is relatively strong correlation. The negative correlation means with the increase of horsepower the MPG will decrease. At of all three comparisons, this shows the strongest negative correlation relative to MPG. Somewhat linear between 12-25mpg. Past 25mpg it doesnt go align with the rest of the data. Using the Seaborn lmplot with order 3 the data follows the curved line pretty well. I would say trending linear but has some strong non-linear characteristics.

6. Yes there is a negative correlation between mpg and horsepower since the Pearson coefficient shows -0.78025. Since this falls past -0.5, that means is relatively strong correlation. The negative correlation means with the increase of horsepower the MPG will decrease. Using the Seaborn lmplot with order 3 the data follows the curved line pretty well. Some portions show strong non-linear characteristics especially past 35 MPG but in general I would tend to lean that this graph shows mostly linear.

3.15 Questions 7-8 (30 points, i.e. 15 points each)

Write Python code to produce (box)plots that should provide good answers the questions below:

7. Did vehicles get more efficient over the years (represented in this dataset, i.e., 1970 through 1982)?
8. Are Japanese cars more fuel efficient than American or European ones?

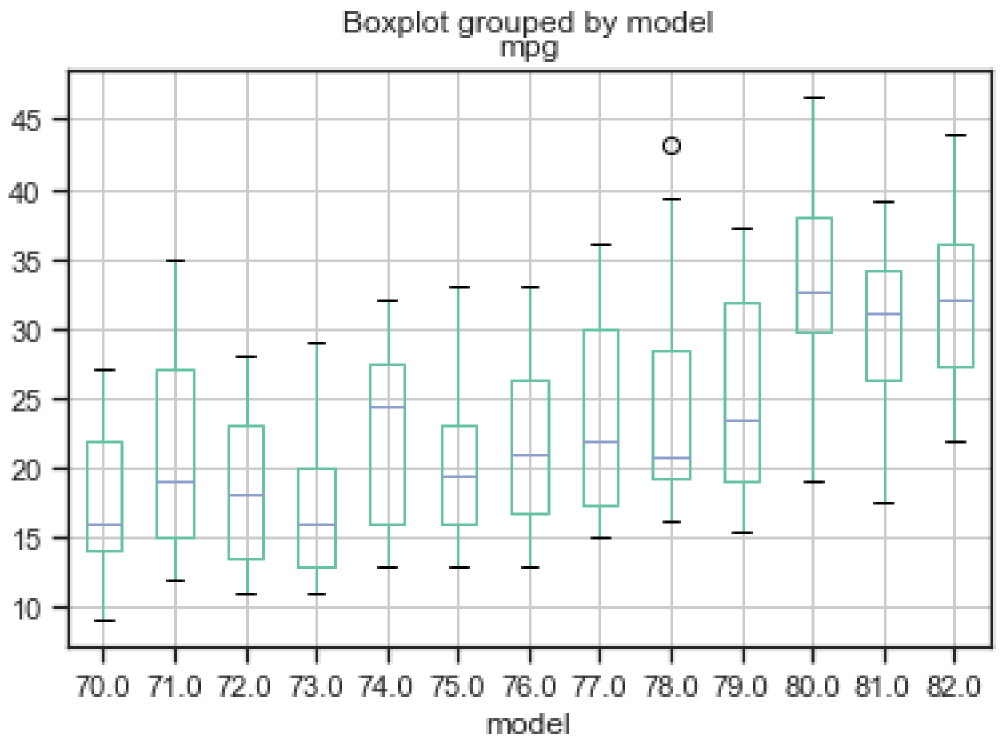
Hint:
`data['Country_code'] = data.origin.replace([1,2,3],['USA','Europe','Japan'])`

3.16 Solution

[51]:
```
print("#7")
data.boxplot(by="model", column = "mpg")
```

```
#7
```

[51]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x192d62a0bc8>
```

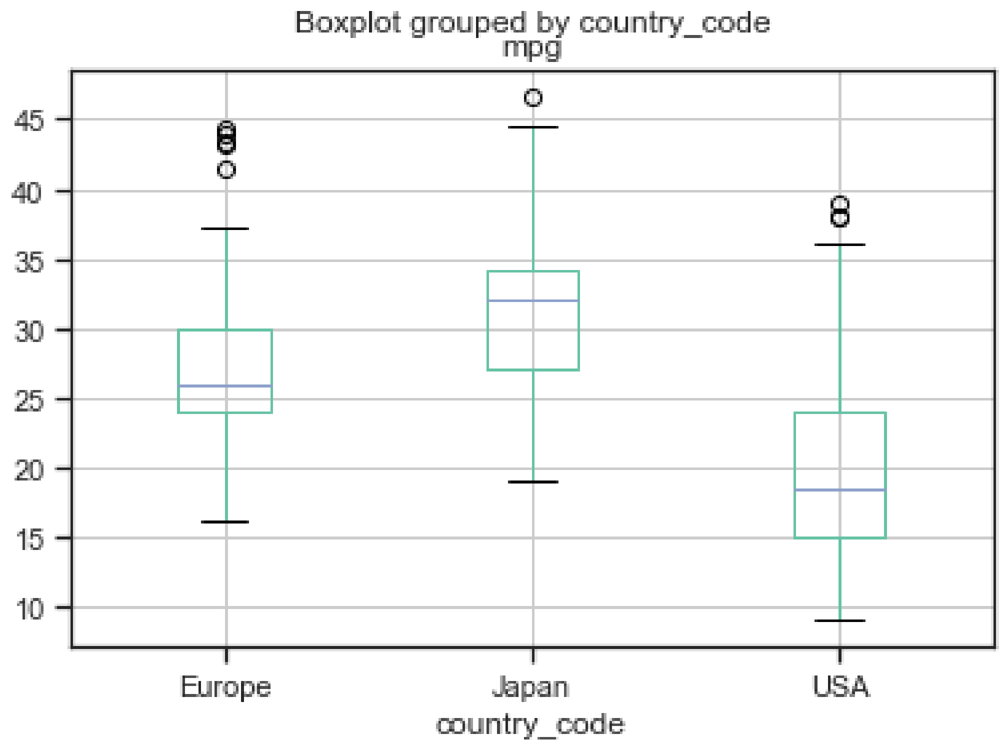


Examination of the MPG over the years(model) it does show tendencies that the MPG improves.

[52]:
```
print("#8")
data['country_code'] = data.origin.replace([1,2,3],['USA','Europe','Japan'])
data.boxplot(by="country_code", column = "mpg")
```

```
#8
```

[52]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x192c11ec248>
```



Examining the box that shows the relationship between cars grouped by country vs the MPG they preform it shows that japanese car have better MPG relative to Europe and USA. Europe follows right behind Japan and USA has the worst overall MPG in the dataset

3.17 Conclusions (36 points)

Write your conclusions and make sure to address the issues below: 1. What have you learned from this assignment? 2. Which parts were the most fun, time-consuming, enlightening, tedious? 3. What would you do if you had an additional week to work on this?

3.18 Solution

(WRITE YOUR CONCLUSIONS HERE)

1. I have learned much about how to use panda DataFrames. How to search, add, remove, extract data from the DataFrames. Learned how to use the basic stat functions that pandas offers. Learned how to plot different datasets and how to try to evaluate trends. Learned about correlation between attributes using Pearson coefficient. I learned much about Jupnyer Labs!
2. I absolutely loved trying to figure out how to graph the data in a good way! Also really nice job laying out this notebook! It really helps organize thoughts and tells the story nicely :). The most time consuming part was just getting use to Pandas and how to search and iterate over the data since its a little different than Python dicts.

3. Probably do some more research on any other functions that I could use to tell if things are linear or non-linear. Also maybe research more about the graphs in how to represent the data. I am sure there are opportunities for improvement such as adding linear lines. I also would like to know how to figure out correlation between attributes that are not numeric