# Final Narrative

NET ID：yuqianc3 Name : Yuqian Cao

**Project Abstract:**

The project I choose is the third type: Programming project. The theme of my project is about how to use Python to analyze data. I noticed that by importing some modules to Python, we could analyze the data as well as draw some plots just like we did in R language. To be more specific, I will work on a dataset which includes some information about different county in American. For instance, the average per capita income of each county, the average persons per household in each county and something like that. My analysis will focus on finding some useful information from the county dataset.

**Project Context:**

This project will base on processing an existing dataset. I used to cope with data using R and I thought if I could do the same thing using Python. In this project, I will focus on three Problem.

1. Cleaning the raw data, removing empty elements in the dataset.
2. Adding new information calculated by the original data to the dataset.
3. Confirm the data types and descriptive values such as mean values, middle values, and sd.
4. Finding out the relationship between two variables and draw the corresponding pictures.
5. Analyzing the feather of the distribution of the variable.

**Project Details:**

First of all, by importing necessary packages of *numpy*, *pandas*, *matplotlib*, data can be easily extracted from the specified CSV file and clearly shown in plots. The packages *numpy* and *pandas* will be used to help read data from files, extract features, and handle dataset as data frames or feature arrays, while the package matplotlib will be used to draw plots of data points or lines of regression.

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model
import collections
```

Second, retrieve the specified CSV file name and load data from the specified CSV file. The function *load_file* will check the existence of the file. If the check fails, an exception will be raised to notify the caller that the file is not existed. Otherwise, the program will run normally, and data will be read and stored in data frames object followed by being

returned. The tactics of dropping null values will depends on the decision of analyzers. By using the function *pd.DataFrame.dropna*, the related rows will be deleted while the parameter axis is assigned to 0, and 1 for related columns.
We could also add a new column to this csv file, we noticed that we already have the amount of households and persons per household, we could use these two variables to calculate the whole population in that county by households times persons per household, then create a new variable *total_person*. Finally, I will export the new data frame (without na value) to a new csv file "new_county.csv":

```
def load_file():
    if not os.path.exists("countyComplete.csv"):
        print("File not exist")
    dfs = pd.read_csv("countyComplete.csv")
    dfs = pd.DataFrame.dropna(dfs, 1)
    total_person = dfs['households'] * dfs['persons_per_household']
    dfs['total_person'] = total_person
    dfs.to_csv("new_county.csv")
    return dfs
```

We could take a look at the new csv file, there is no NA value inside, and the new variable *total_person* has been added:

| T | U | V | W | X | Y | Z | AA | AB | AC | AD | AE | AF | AG | AH | AI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| mean_work_ | housing_un | home_owne | housing_mu | median_val | households | persons_per | per_capita_i | median_hou | poverty | private_non | private_non | building_pe | area | density | total_person |
| 25.1 | 22135 | 77.5 | 7.2 | 133900 | 19718 | 2.7 | 24568 | 53255 | 10.6 | 877 | 10628 | 191 | 594.44 | 91.8 | 53238.6 |
| 25.8 | 104061 | 76.7 | 22.6 | 177200 | 69476 | 2.5 | 26469 | 50147 | 12.2 | 4812 | 52233 | 696 | 1589.78 | 114.6 | 173690 |
| 23.8 | 11829 | 68 | 11.1 | 88200 | 9795 | 2.52 | 15875 | 33219 | 25 | 522 | 7990 | 10 | 884.88 | 3 | 24683.4 |
| 28.3 | 8981 | 82.9 | 6.6 | 81200 | 7441 | 3.02 | 19918 | 41770 | 12.6 | 318 | 2927 | 8 | 622.58 | 36.8 | 22471.82 |
| 33.2 | 23887 | 82 | 3.7 | 113700 | 20605 | 2.73 | 21070 | 45549 | 13.4 | 749 | 6968 | 18 | 644.78 | 88.9 | 56251.65 |
| 28.1 | 4493 | 76.9 | 9.9 | 66300 | 3732 | 2.85 | 20289 | 31602 | 25.3 | 120 | 1919 | 1 | 622.81 | 17.5 | 10636.2 |
| 25.1 | 9964 | 69 | 13.7 | 70200 | 8019 | 2.58 | 16916 | 30659 | 25 | 446 | 5400 | 3 | 776.83 | 27 | 20689.02 |
| 22.1 | 53289 | 70.7 | 14.3 | 98200 | 46421 | 2.46 | 20574 | 38407 | 19.5 | 2444 | 38324 | 107 | 605.87 | 195.7 | 114195.66 |
| 23.6 | 17004 | 71.4 | 8.7 | 82200 | 13681 | 2.51 | 16626 | 31467 | 20.3 | 568 | 6241 | 10 | 596.53 | 57.8 | 34339.31 |
| 26.2 | 16267 | 77.5 | 4.3 | 97100 | 11352 | 2.22 | 21322 | 40690 | 17.6 | 350 | 3600 | 6 | 553.7 | 46.9 | 25201.44 |
| 32 | 19278 | 75.1 | 4.4 | 103700 | 16563 | 2.59 | 20517 | 39486 | 18.4 | 744 | 7048 | 22 | 692.85 | 63 | 42898.17 |
| 34.8 | 7269 | 85.6 | 3.9 | 63400 | 5296 | 2.61 | 17214 | 31076 | 18.7 | 276 | 2969 | 1 | 913.5 | 15.2 | 13822.56 |
| 25.1 | 12638 | 80 | 6.3 | 73300 | 9144 | 2.84 | 17372 | 27439 | 29.2 | 623 | 6944 | 5 | 1238.47 | 20.9 | 25968.96 |
| 27.3 | 6776 | 72.8 | 11.2 | 82900 | 5851 | 2.24 | 18332 | 35595 | 18.8 | 204 | 3253 | 2 | 603.96 | 23.1 | 13106.24 |
| 29.1 | 6718 | 74.9 | 5.3 | 87900 | 5397 | 2.71 | 17490 | 36077 | 17.1 | 180 | 2071 | 8 | 560.1 | 26.7 | 14625.87 |
| 20.7 | 22330 | 69.7 | 13.6 | 116700 | 18928 | 2.51 | 22797 | 42253 | 17.2 | 1016 | 12499 | 239 | 678.97 | 73.6 | 47509.28 |
| 22.7 | 25758 | 73.5 | 12.3 | 94700 | 22165 | 2.41 | 21079 | 39610 | 15.7 | 1290 | 19703 | 88 | 592.62 | 91.8 | 53417.65 |
| 25.2 | 7093 | 81.6 | 6 | 72500 | 4861 | 2.74 | 15755 | 26944 | 30.6 | 216 | 2612 | 1 | 850.16 | 15.6 | 13319.14 |
| 30.2 | 6478 | 83.7 | 1.9 | 73100 | 4561 | 2.43 | 19209 | 35560 | 16 | 106 | 1050 | 0 | 650.93 | 17.6 | 11083.23 |

The next problem focused on is to build a line model and draw its corresponding plot.

```
x = np.array(dfs['hs_grad']).reshape(-1, 1)
y = np.array(dfs['poverty'])
lm_profile = linear_model.LinearRegression()
lm_profile.fit(X=x, y=y)
a, b = lm_profile.coef_, lm_profile.intercept_
print("The slope is: " + str(a) + ", the intercept is " + str(b))
```

We firstly choose two valuables: 'hs_grad', 'poverty', the first as the explanatory variable and the second as the response variable. The variable 'hs_grad' describes the percentage of people already graduated in certain county. The larger this variable is, the more people

educated. The other variable ‘poverty’ describes the poverty of the county. The larger this variable is, the poorer the county is. Next, we would like to do linear regression with these two variables. Trying to use the educational degree to explain the poverty degree. We will extract these two variables to variable x and variable y, (x as the explanatory variable and y as the response variable), then use the function linear_model.LinearRegression to build a linear model. We could also use lm_profile.coef_ to print out the slope and the intercept.

```
The slope is: [-0.59074624], the intercept is 64.59437487273459
```

Then we look into the descriptive values following the preparing data values. To be specific, we transform the *pandas* data frames to *numpy* arrays with the object method *values*. The new array will be a two-dimension array with the state names and county names as the first two columns. By specifying indexes from 2 to the end, we can print all data of attributes for all counties. It is also shown in two-dimension format. In addition, by using the method *info*, we can get a summary of fields of the data frames. With the method *describe*, we can get the means, standard derivation, maximum, and minimum in just one line of code. Then the data analysis can be conducted using packages with analyzing algorithms efficiently.

```
dfs = load_file()
print('Info: ', dfs.info())
print('Describe: ', dfs.describe())
print('Values:', dfs.values[2:])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3143 entries, 0 to 3142
Data columns (total 34 columns):
state                                3143 non-null object
name                                 3143 non-null object
FIPS                                 3143 non-null int64
pop2010                              3143 non-null int64
age_under_5                          3143 non-null float64
age_under_18                         3143 non-null float64
age_over_65                          3143 non-null float64
female                               3143 non-null float64
white                                3143 non-null float64
two_plus_races                       3143 non-null float64
hispanic                             3143 non-null float64
white_not_hispanic                   3143 non-null float64
no_move_in_one_plus_year             3143 non-null float64
foreign_born                         3143 non-null float64
```

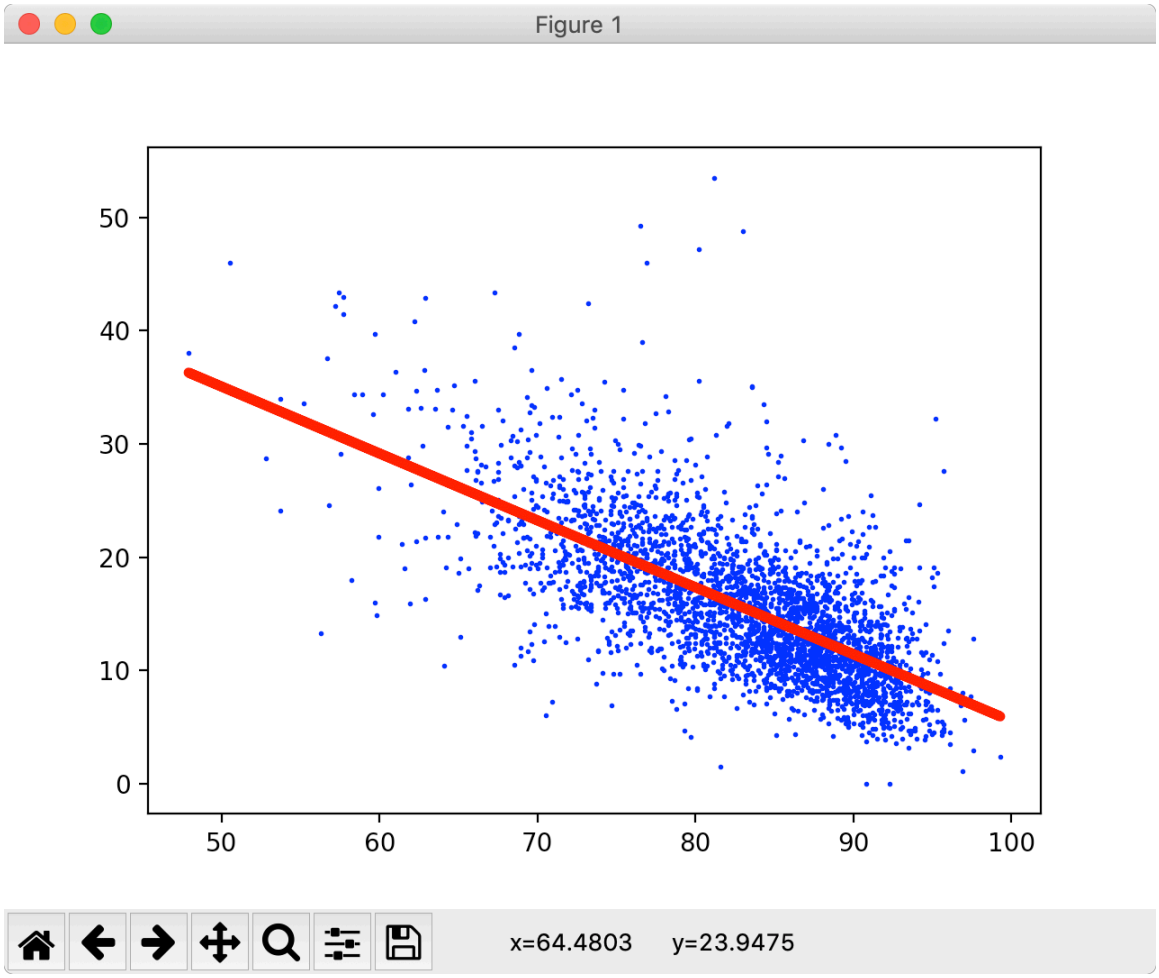
```
memory usage: 654.9+ KB
None
              FIPS       pop2010      ...           density  total_person
count   3143.000000  3.143000e+03      ...       3143.000000  3.143000e+03
mean   30390.411709  9.823275e+04      ...        259.322431  9.416478e+04
std    15164.717720  3.129012e+05      ...       1724.159773  3.025806e+05
min     1001.000000  8.200000e+01      ...          0.000000  4.092000e+01
25%    18178.000000  1.110450e+04      ...         16.900000  1.063670e+04
50%    29177.000000  2.585700e+04      ...         45.200000  2.468836e+04
75%    45082.000000  6.669900e+04      ...        113.850000  6.396964e+04
max    56045.000000  9.818605e+06      ...      69467.500000  9.557130e+06

[8 rows x 32 columns]
[['Alabama' 'Barbour County' 1005      884 88 31 0 24683 4]
```

This table contains statistic conclusions for each variable in dataset. It consists of count of data entries, mean value, sd, and some inter-values. With this table we can have a brief idea about the data like the range of the all data and how it is distributed.

Next, we will draw the corresponding scatter plot and add the regression line to this plot:

```
plt.scatter(x, y, color='blue', s=1)
plt.plot(x, lm_profile.predict(X=x), color='red', linewidth=4)
plt.show()
```




The x-axis means the educational degree and the y-axis means the poverty degree. The red line lies near the center of those blue points. According to this plot we could draw the conclusion that education degree is negative related to poverty degree.

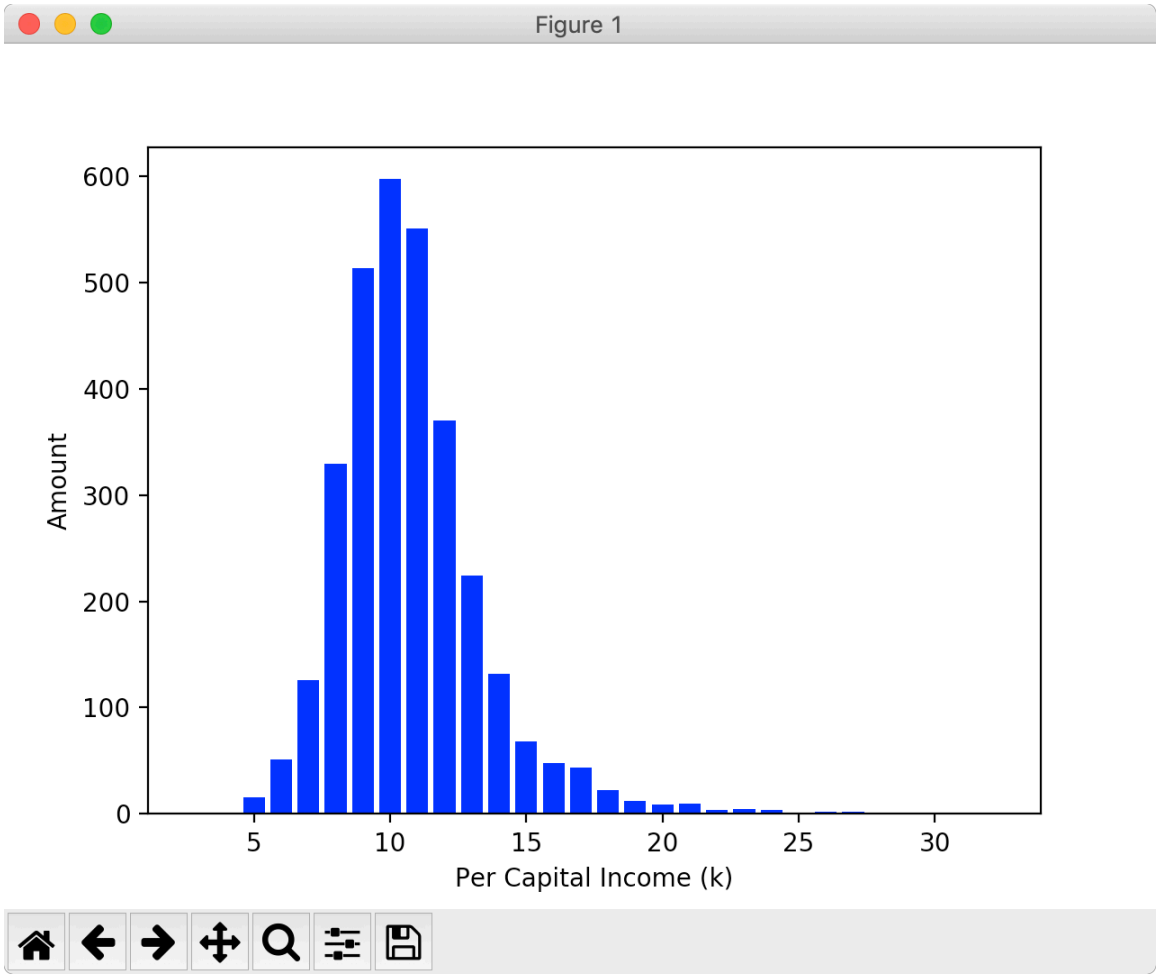
Then we would focus on the distribution of per capita income in different counties. I made an assumption that most of the counties have similar per capita income while a little prosper counties may have extremely high per capita income and some poor counties have extremely low per capita income. We will next draw a distribution plot to justify our assumption. I draw the distribution plot. First of all, we need to set a interval of this histogram, I chose 2000 as the interval. Then I will go through each element in the data frame and count.

```
interval = 2000
income_counter = collections.Counter()
for ele in dfs['per_capita_income']:
    income_counter[ele // interval] += 1

income = [i for i in income_counter.keys()]
income_amounts = [i * interval for i in income_counter.values()]

plt.bar(income, income_amounts, color='blue')
plt.xlabel('Per Capital Income')
plt.ylabel('Amount')
plt.show()
```

The variable ‘interval’ identifies the length of interval. Then traverse every element in the dataset, and divide the element value by length of interval, finally I can get the result of which interval this element should be located in. After knowing where this element lies, add 1 to the count value of related item inside the counter. After all of this process, we can finally retrieve a counter contains counts of items in different intervals. Use the data inside the counter to draw a graph.




According to the plot, axis x is for per capital income, while the axis y is for the number of counties. As we can see, the trendy is similar to a normal distribution, the centric part has the largest number of samples and the head and tail has less samples, which is corresponding with our assumption.

**Github Link:**
https://github.com/YuqianCao/IS452-final-yuqianc3