**Pandas**

**Introduction to Data Visualization using Pandas**

Start by loading these modules to your jupyter notebook.

**import** **matplotlib.pyplot** **as** **plt**

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

df **=** pd**.**DataFrame({

'name':['john','mary','peter','jeff','bill','lisa','jose'],

'age':[23,78,22,19,45,33,20],

'gender':['M','F','M','M','M','F','M'],

'state':['california','dc','california','dc','california','texas','texas'],

'num\_children':[2,0,0,3,2,1,4],

'num\_pets':[5,1,0,5,2,2,3]

})

1. **Try the following Python code:**

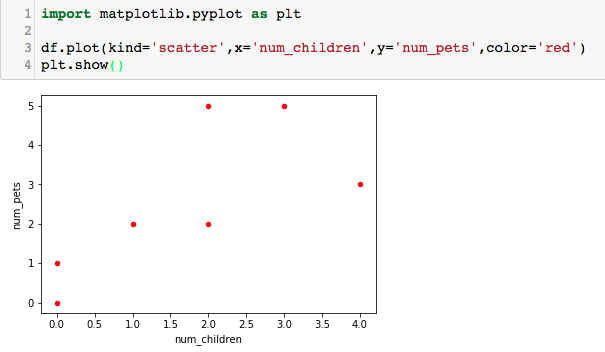


Pandas has **tight integration** with matplotlib.

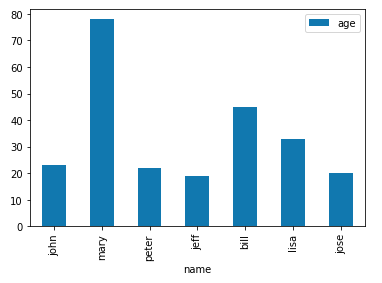
You can plot data directly from your DataFrame using the [plot()](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html) method:

1. **Try the following .plot()**  Note: If you have already imported matplotlib in any of the previous cells, you do not need to import it again.

Now you will see a graph showing a number of children in x-axis, where y-axis is a number of children (for each record).

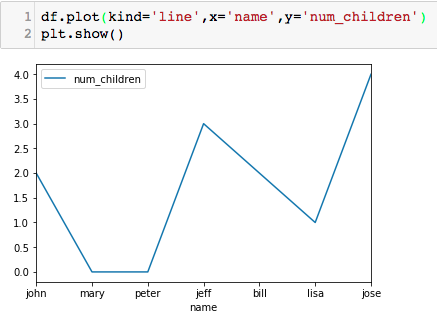


1. **Try the above .plot() with x = ‘age’**
2. **Try .plot() with kind = ‘bar’ and change x-axis to ‘name’ and y-axis to ‘age’ (expected Bar graph is as follows)**

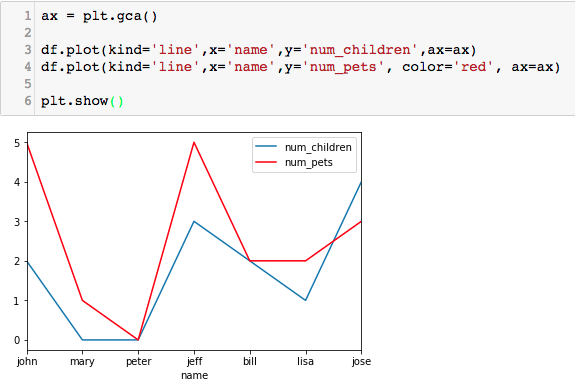
****

1. **With the above .plot(), change kind to ‘barh’ and observe the printed Bar graph.**
2. **Try .plot() with different x-axis and y-axis. For example, how many children does each person have? How many pets does each person own?**

Next you will learn how to do Line plot with multiple columns. Try the following code.



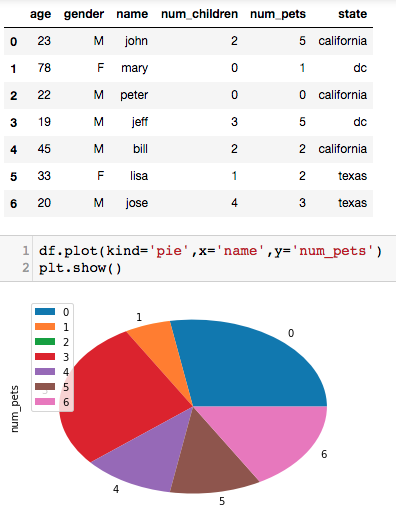
*plot()*takes an optional argument 'ax' which allows you to reuse an Axis to plot multiple lines. We need to put ax = plt.gca(), where gca stands for get current axis. .gcs() tells matplotlib to reuse the existing axis to plot other lines. Try the following code.



We can save graphs by calling plt.savefig('outputfile.png'), instead of calling plt.show() to show the graph.

1. **Save one of the previous graph in your working directory and check your directory for the saved file.**

You can also plot a Pie chart by changing kind to ‘pie’. With the following datagframe, try the given Python code.



As you may observe from the graph, its Legend is index values. It might cause confusion for person to understand it. It becomes necessary to define a label, which can be done as follows:



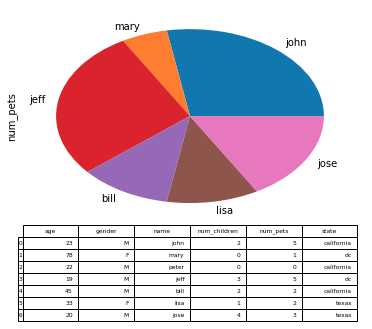
1. **Try the above code.**

The legend shown in the square box may be displayed on top of Pie chart, if you want to remove that it can be done by adding legend = False in the .plot()

1. **Put -- legend = False – as one of the argument in .plot()**

You can add DataFrame as a table together with a graph by adding table = df (df is your dataframe variable; if your dataframe variable is zoo, then it must be table = zoo).

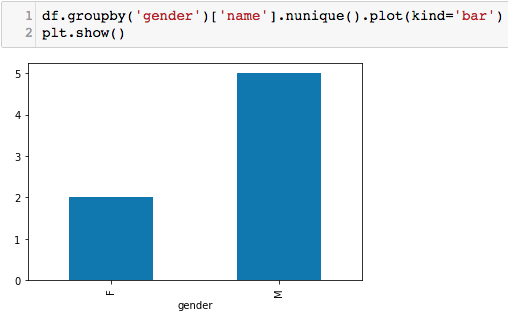
1. **Put table = df in .plot() (Expected output is as follows.)**

****

If you want to adjust a size, it can be done using figsize = (10,10) [Note that values (10,10) in the blanket are the width and height values]

1. **Make a graph bigger and smaller.**

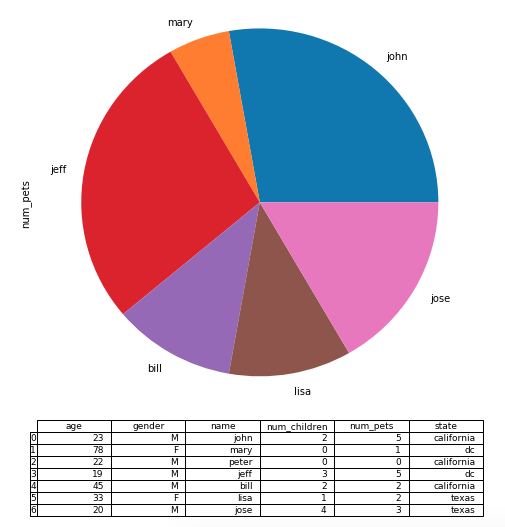
We can plot a graph based on a group of data using .groupby(). For example,



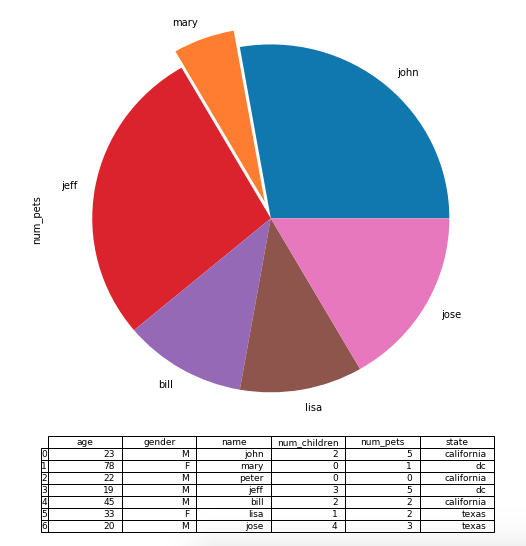
The above Bar graph shows data based on groups of gender, and ‘name’ is defined as a key index which is a unique value [ .nunique() ]. Note that if values in ‘name’ column are not unique, another column must be used.

1. **With reference to the above example, change to Pie graph.**
2. **Group data by state and plot Pie and Bar graphs**

From exercise 10,

****

We can highlight certain pies by adding explode = (0,0.1,0,0,0,0) in .plot(). The order is anti-clockwise and the first pie starts at 180degree. In this case, the first pie is John and we want to explode the second pie, which is mary.

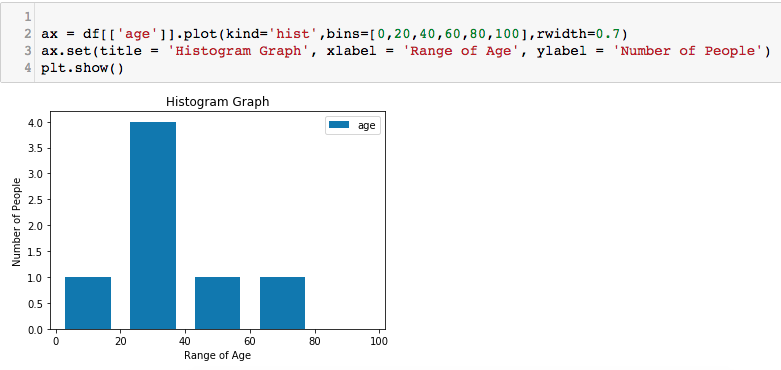


1. **Try changing the values in, e.g., explode = (0,0,0.5,0.2,0,0) to observe different explosions in Pie graph.**
2. **Highlight jeff and jose pies.**

The following example illustrates how to plot Histogram based on ‘age’. By default, x-axis and y-axis do not show any labels.



We can define a label for each axis and a title of the graph by using .set() as shown below.



1. **From the Pie graph in exercises 12-13, set a title of the graph to ‘Pets Owned by Each Person’ and set a label in y-axis to ‘Percentage of Pets’.**

The following exercises deal with Scottish’s hill data. Scottish\_hills.csv can be downloaded from the portal. Main data in this .csv file contains latitude, longitude and height of scottish’s hills.

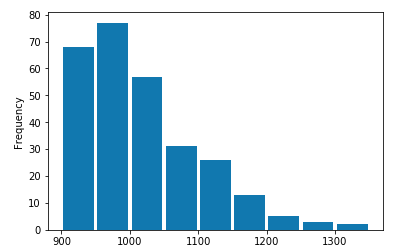
****

1. **Load the data from scottish\_hills.csv to the dataframe called *sh*. For example,**

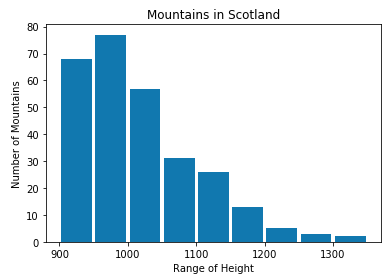
**sh = pd.read\_csv…..**

1. **Plot a Histogram for ‘height’ and define bin = 900,950,1000,1050,1100,1150,1200,1250,1300,1350, with rwidth = 0.9**

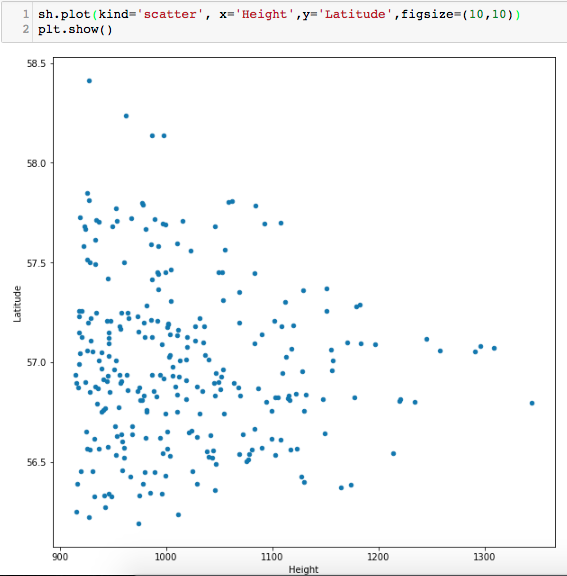
**The expected graph is shown below.**

****

1. **Set a title and labels for each axis as follows.**

****

Next, we are going to do a scatter plot to show the relationship between ‘height’ and ‘latitude’. Try the following code to see the graph.



1. **Set a title to ‘Mountains in Scotland’ and label for y-axis to ‘Mountain Height’.**

We can set a filter to show plots for specific values. For example, if you want to see only hills with a height of 1,200 meters or higher. We can use the following code.

ax = sh[sh.Height > 1200].plot(kind='scatter', x='Height',y='Latitude',figsize=(10,10))

1. **Plot Scatter graph with hills’ height lower than 1,100 meters.**
2. **Plot Scatter graph with latitude lower than 57 degree.**

We can also specify a range of data to be plotted. For example, if we want to plot only hills with a height between 900 and 1000 meters. We need to do the following things. First, we specify a height (lower bound) and store our results in another datagrame, such as sh1.

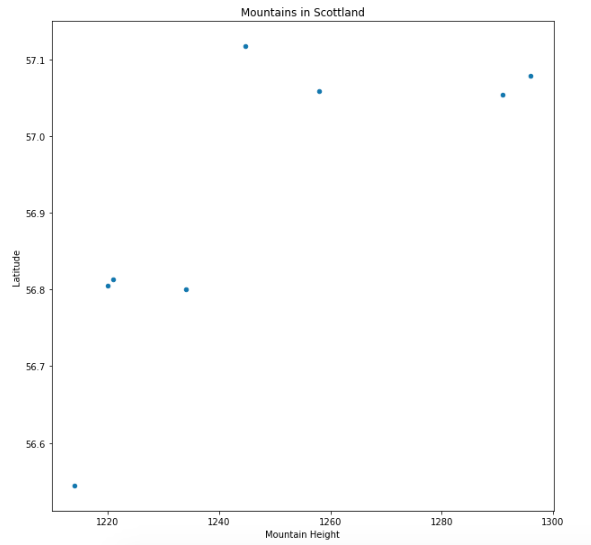
sh1 = sh[sh.Height > 900]…..

Then we do it again with an upperbound value and store the results in another dataframe such as sh2.

sh2 = sh1[sh1.Height < 1100]

Then run plt.show()

1. **Do Scatter plot with a height between 1,200 and 1,300 meters. The expected graph is shown below.**

****

1. **Scatter plot with latitude between 57 and 58 degrees.**

Since we do have both latitude and longitude values in scottish\_hills.csv, we can show the exact location of hills on Scotland map. To do so, we need to import cartopy.

First thing first, since your environment may not have this package, you need to install the package prior to using it. Go to Anaconda’s command prompt and

conda install -c conda-forge cartopy

Then use the given python code below to plot locations of hills on the map.

import cartopy.crs **as** ccrs

from cartopy.mpl.ticker import LongitudeFormatter, LatitudeFormatter

import cartopy.feature **as** cfeature

plt**.**figure(figsize**=**(20,10))

ax **=** plt**.**axes(projection**=**ccrs**.**Mercator())

ax**.**coastlines('10m')

ax**.**xaxis**.**set\_visible(True)

ax**.**yaxis**.**set\_visible(True)

ax**.**set\_yticks([56,57,58,59], crs**=**ccrs**.**PlateCarree()) #Ranges of Latitude

ax**.**set\_xticks([**-**8, **-**6, **-**4, **-**2], crs**=**ccrs**.**PlateCarree()#Ranges of Longitude

lon\_formatter **=** LongitudeFormatter(zero\_direction\_label**=**True)

lat\_formatter **=** LatitudeFormatter()

ax**.**xaxis**.**set\_major\_formatter(lon\_formatter)

ax**.**yaxis**.**set\_major\_formatter(lat\_formatter)

ax**.**set\_extent([**-**8, **-**1.5, 55.3, 59]) #boundaries of graph based on Lat/Long

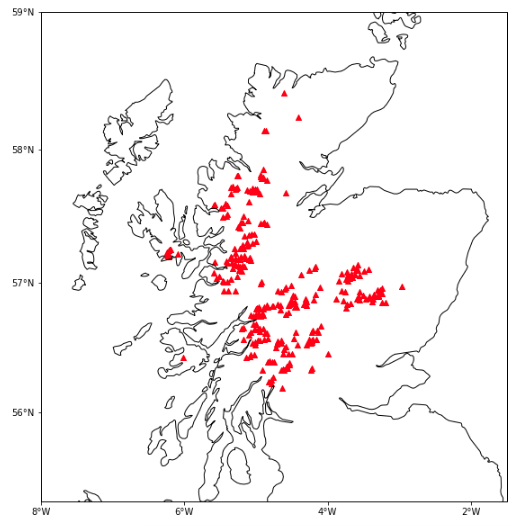
plt**.**scatter(sh['Longitude'],sh['Latitude'],

color**=**'red', marker**=**'^', transform**=**ccrs**.**PlateCarree())

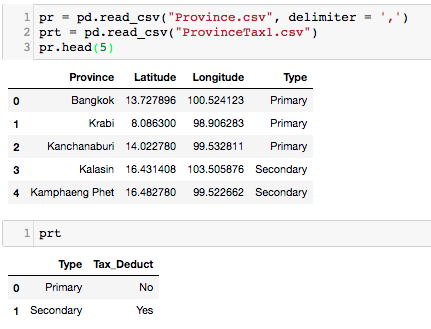
plt.show()

#plt**.**savefig("munros.png")

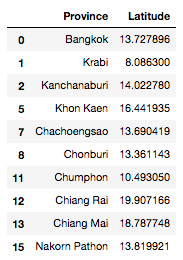
With the above codes, the expected graph is as follows.



**Load the data from Province.csv and ProvinceTax.csv**



1. **Show the first 10 records of primary cities, and show only ‘Province’ and ‘Latitude’ columns. The expected output is**



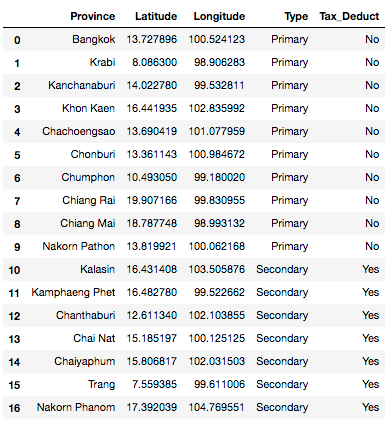
1. **Show a number of primary and secondary cities. The expected output is**

****

1. **Merge prt to pr and store the results in a dataframe called *pr\_merge*.**

***pr\_merge = pr.merge…..***

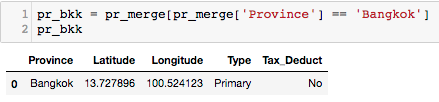
**The expected output is**

****

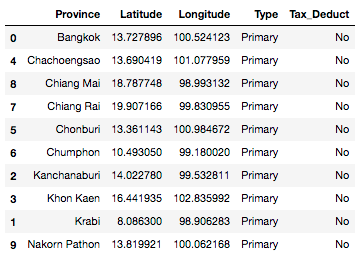
1. **Sort the values by Province and store the results in a dataframe called *pr\_merge\_sort*, the expected output is**

****

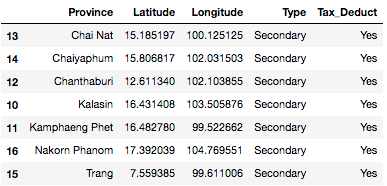
In a case that we want to select one particular record, for example Bangkok, we can use the following code.



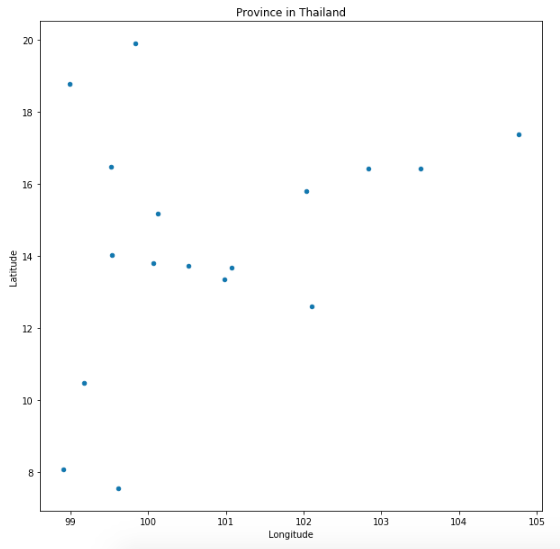
1. **Select all records there Type is primary and store the results in a dataframe called *pr\_primary*. The expected output is**

****

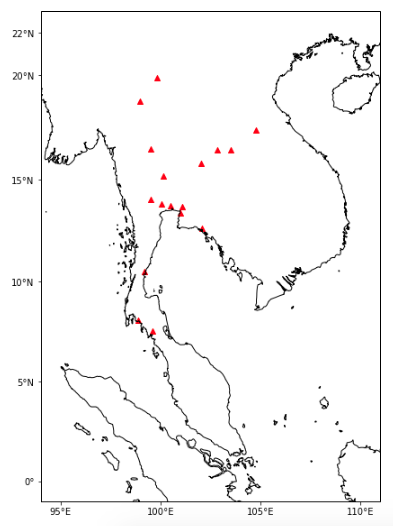
1. **Select all records there Type is secondary and store the results in a dataframe called *pr\_secondary.* The expected output is**

****

1. **Show Scatter plot with x-axis is Longitude and y-axis is Latitude. Also set a title of the graph to Province in Thailand. The expected graph is**

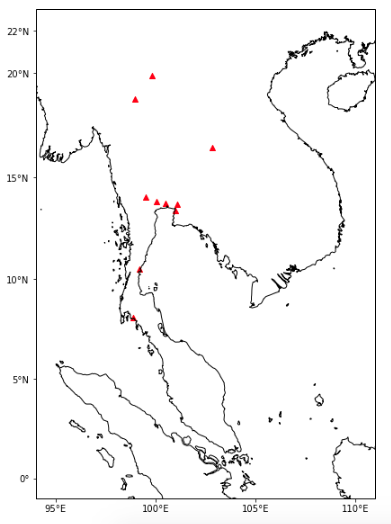
****

1. **Show Scatter plot on Thailand map. The expected graph is**

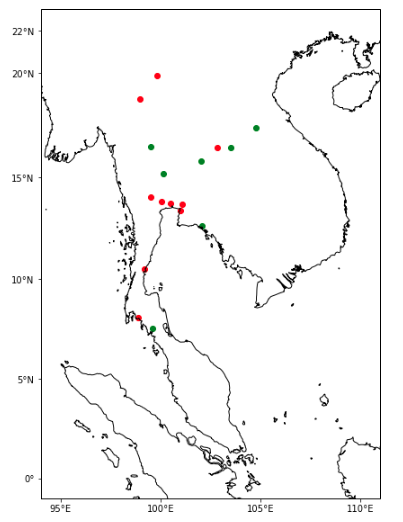
****

Based on the two dataframes, namely pr\_primary and pr\_secondary,

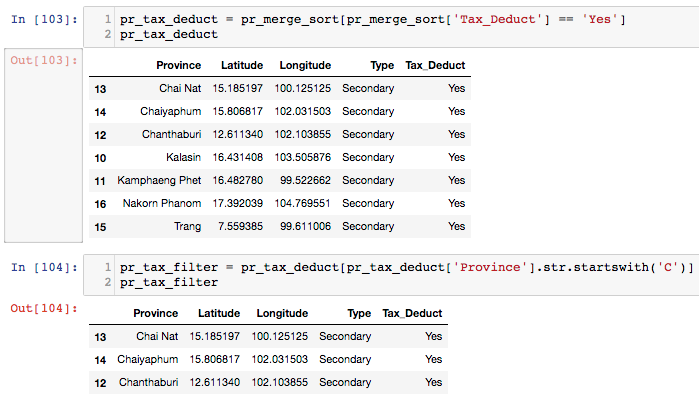
1. **Show Scatter plot of primary cities only.**

****

1. **Show Scatter plot of both primary and secondary cites, where a marker is set to 'o', and a color of marker for primary and secondary cities is set to red and green, respectively. The expected graph is**

****

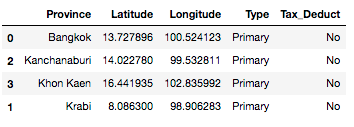
1. **Show a Scatter plot of Bangkok only**

From the dataframe *pr\_merge\_*sort, we can show only cities that expenses (spend in that city) can be tax deductible. We can also show only cities that a first character is ‘C’.

1. **Show primary cities where expenses are not tax deductible and a city name is started with ‘K’. The expected output is**

****

1. **Show primary cities where expenses are not tax deductible and a city name is started with ‘B’ and ‘K’. The expected output is**

****

**Last but not least,**

1. **Load zoo.csv and zoo\_eats.csv. Merge two dataframes and show different data visualizations (try to plot different graphs with different x-axis and y-axis). Try as much as you can for your own practices.**