Assignment

February 23, 2018

1 Assignment

The assignment requires you to use some libraries other than what we have learnt today such as SciPy and scikit-image. Three main tasks that you are going to do are **Loading data**, **Processing or Analysis**, and **Visualization**.

1.1 Loading data

- There are several data set available in `data` folder. The data is in `.csv` format.
- You can list filename directly in this jupyter notebook cell by command `ls ../data`.
- You can also use example images from scikit-image data module.

```
In [ ]: %ls "../data"
```

1.2 Processing or Analysis

- You can do anything on the data start from sum or finding average of the values, the more the better.
- If you can apply any other functions not in an example, that will be good for you.
- Some methods as an idea to do are: sum, mean, subtract, groupby.
- Main modules to be used are loaded in the next cell.

```
In [ ]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from os import pardir, path
        %matplotlib inline

        # to be able to render thai font
        # on linux system, uses 'Loma'
        # on windows system, uses 'Angsana New' or 'TH SarabunPSK'
        plt.rcParams["font.family"] = ["Loma", 'Angsana New', 'TH SarabunPSK']
```

```
In [ ]: # define function to return full path of data from data filename
        # There are 3 naming styles used here. However, it is recommented to use meaningful na
        # Shorthand name should only be used when prototyping or personal code.

        # These functions do the same thing, return the same result.
```

```
# shorthand naming
def dp(dataFileName):
    '''
    This function is used to get full data path (dp) from data file name.
    '''
    return path.join(pardir, 'data', dataFileName)

# camelCase naming
def getDataPath(dataFileName):
    '''
    This function is used to get full data path from data file name.
    '''
    return path.join(pardir, 'data', dataFileName)

# delimiter separated naming
def get_data_path(dataFileName):
    '''
    This function is used to get full data path from data file name.
    '''
    return path.join(pardir, 'data', dataFileName)
```

```
In [ ]: # here we get data from https://data.go.th/
        # some of them are encoded with 'utf-8' some of them are 'cp874'
        # you will know when you first load data, if it cannot be read change encoding

        df = pd.read_csv(dp('traffic_amount_2558.csv'), encoding='cp874')
        # df = pd.read_csv(dp('travel_amount_2558.csv'))
```

1.3 Visualization

- Matplotlib is important here. However if you don't need to customize the plot just used `plot` from `pandas` should be ok.
- We will give 2 examples for visualization here, the first one will be just plot one column of data, the second will be selecting desired columns to plot.
- First thing to do is to look at big picture of the dataframe. Just type dataframe variable in the cell and run.

```
In [ ]: df
```

1.3.1 Plotting single column

After observing dataframe roughly, we may think that we could use **DISTRICT** column to plot some data. First example will be plotting the average number of **2 /3 VEH1_T** in each **DISTRICT**.

- First thing to do is to group by **DISTRICT**.
- Then find average values of that grouped data.
- Finally we will plot it.

```
In [ ]: # groupby and find average
        # .groupby() is a function of class DataFrame, so we can directly call it after functi
        # .mean() is a function of groupby object, so we can also call it after getting groupb

        to_plot_df = df.groupby([' DISTRICT']).mean()

        to_plot_df
```

```
In [ ]: # then we plot desired column
        pt = to_plot_df['2 /3 VEH1_T'].plot(kind='bar', figsize=(18,8), fontsize=12)
        plt.xlabel(pt.get_xlabel(), fontsize=22)
        plt.ylabel('Average of 2 /3 VEH1_T', fontsize=22)
        plt.title('Average of 2 /3 VEH1_T by  DISTRICT', fontsize=30)
```

1.3.2 Plotting selected columns

To plot multiple columns, we first need to choose which columns to plot. Read through the example below.

```
In [ ]: # look at list of columns name below and select what to plot
        for i, item in enumerate(df.columns.unique().tolist()):
            print(i, item)
```

```
In [ ]: # suppose we need columns from '2 /3 VEH1_T' to ' VEH13_T' (15 to 27)

        df_to_plot = df.iloc[:, 15:27]

        # iloc means index location
        # : means selecting all rows
        # 15:27 means selecting columns from 15 to 27
```

```
In [ ]: # try plotting

        pt = df_to_plot.sum().plot(kind='bar', figsize=(18, 5), rot=0, fontsize=16)
        plt.xlabel('Vehicle Types', fontsize=22)
        plt.ylabel('Number of Vehicles', fontsize=22)
        plt.title('Number of vehicles by vehicle types', fontsize=30)

        # default kind is 'line'
        # we sum to see big picture of amount of vehicle by types
        # figsize controls size of the plot, width is max at ~18 and height can be calculated
        # rot is rotation of x axis label, default at 90
        # fontsize controls overall font size of the plot
```

```
In [ ]: # you can see the x axis labels doesn't look good, so we will change column name befor

        df_to_plot.columns.tolist()
```

```
In [ ]: # here we loop through list of old column names and chop out only the keyword of vehic
        # - split without argument ignore all spaces
```

```
        # - [x for x in LIST] returns list of item in LIST, we can do anything to first x in t
        # - [-1] is an index of last element in list or array in Python language

        new_col = [item.split()[-1] for item in df_to_plot.columns.tolist()]

        # you can uncomment below line to see the result
        # print(new_col)

        # we the assign new column names to the old df

        df_to_plot.columns = new_col
```

In [ ]:
```
        # plot again
        df_to_plot.sum().plot(kind='bar', figsize=(18, 5), rot=0, fontsize=16)
        plt.xlabel('Vehicle Types', fontsize=22)
        plt.ylabel('Number of Vehicles', fontsize=22)
        plt.title('Number of vehicles by vehicle types', fontsize=30)
```

In [ ]:
```
        # or plot as a line
        df_to_plot.sum().plot(figsize=(18, 5), fontsize=16)
        plt.xlabel('Vehicle Types', fontsize=22)
        plt.ylabel('Number of Vehicles', fontsize=22)
        plt.title('Number of vehicles by vehicle types', fontsize=30)
```

2 Student part

- Visualize any data from `data/` or example data from `scikit-learn` or `scikit-image`.
- Do your best and DO NOT copy from your friends.
- 3 Things to do: Load, Process and Visualize.
- Processing step doesn't have to be complex, just simple function like sum and average are ok. But feel free to show your advance skill as much as you want would be best.
- Your advance knowledge would be a plus in this class.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: