Challenge

August 29, 2019

1 DATA 420 Assignment 1 Peng Shen (57408055)

2 Challenge

2.1 Q1

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import plotly.plotly as py
        import plotly.tools as tls
        import os
        from collections import OrderedDict
```

2.1.1 Investigating other elements

```
In [2]: # Load the parquet file as pandas dataframe
        df = pd.read_parquet('./element_count_by_day.parquet', engine='pyarrow')
```

```
In [3]: df.head()
```

```
Out[3]:   ELEMENT      DATE  ELEMENT_COUNT
        0    WESD  20090102           1193
        1    WT01  20090105            644
        2    TOBS  20090105           5587
        3    WSF2  20090108            929
        4    WESD  20090109           1230
```

```
In [4]: df.DATE = pd.to_datetime(df.DATE, format='%Y-%m-%d')
        df.head()
```

```
Out[4]:   ELEMENT       DATE  ELEMENT_COUNT
        0    WESD 2009-01-02           1193
        1    WT01 2009-01-05            644
        2    TOBS 2009-01-05           5587
        3    WSF2 2009-01-08            929
        4    WESD 2009-01-09           1230
```

```
In [5]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 241032 entries, 0 to 241031
Data columns (total 3 columns):
ELEMENT          241032 non-null object
DATE             241032 non-null datetime64[ns]
ELEMENT_COUNT    241032 non-null int64
dtypes: datetime64[ns](1), int64(1), object(1)
memory usage: 5.5+ MB
```

```
In [6]: df.DATE.describe()

Out[6]: count                  241032
        unique                  52384
        top       2013-05-21 00:00:00
        freq                       10
        first     1851-05-19 00:00:00
        last      2017-09-10 00:00:00
        Name: DATE, dtype: object
```

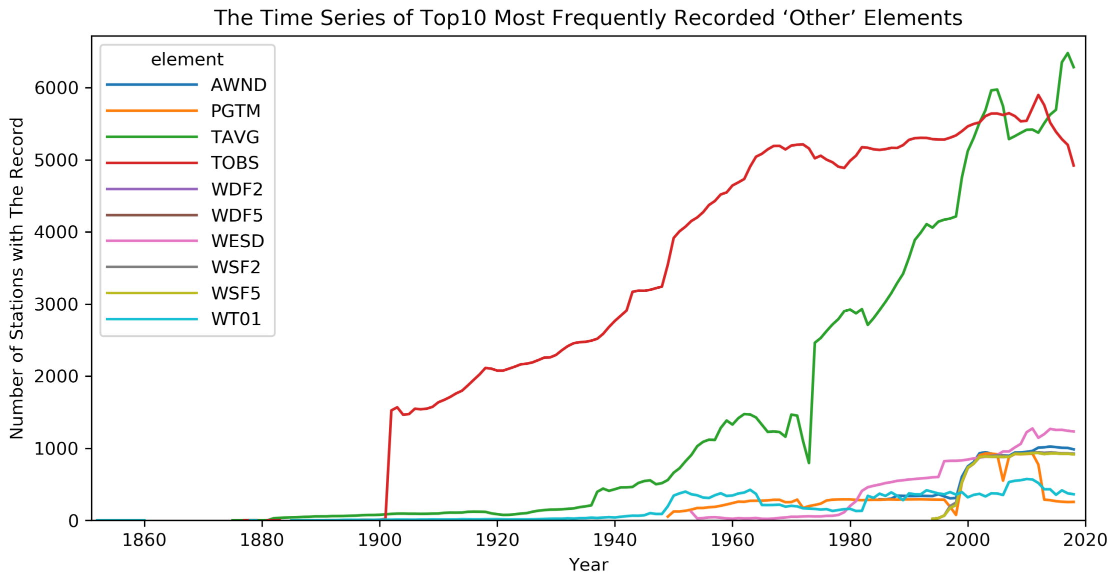
```
In [7]: df.ELEMENT_COUNT.describe()

Out[7]: count    241032.000000
        mean       1313.928752
        std        1740.280533
        min           1.000000
        25%         135.000000
        50%         570.000000
        75%        1442.000000
        max        6719.000000
        Name: ELEMENT_COUNT, dtype: float64
```

```
In [8]: data = df.pivot_table(index='DATE',columns='ELEMENT',values='ELEMENT_COUNT')
```

```
In [9]: # Plot the time series of frequency of each other element collected by stations all ov
        f, a = plt.subplots(dpi=300, figsize=(10, 5))  # affects output resolution (dpi) and f
        data = data.resample('Y').mean()
        a.plot(data, label=data.columns)  # assign label to include in legend
        a.set_ylim([0, 6720])  # exapnd axes slightly beyond [1, 6720]
        a.set_xlim("1851", "2020")

        # Legend
        a.legend(data.columns, title='element',fontsize=10, handlelength=5)

        # Labels
        a.set_title(f"The Time Series of Top10 Most Frequently Recorded Other Elements")
        a.set_xlabel("Year")
        a.set_ylabel("Number of Stations with The Record")
```

```
/Users/dylan/anaconda3/lib/python3.7/site-packages/pandas/plotting/_converter.py:129: FutureWa

Using an implicitly registered datetime converter for a matplotlib plotting method. The conver

To register the converters:
        >>> from pandas.plotting import register_matplotlib_converters
        >>> register_matplotlib_converters()
```

```
Out[9]: Text(0, 0.5, 'Number of Stations with The Record')
```



```
In [10]: # Outputs
         output_path = os.path.expanduser("~/Documents/plots")  # M:/plots on windows
         if not os.path.exists(output_path):
             os.makedirs(output_path)

         # Save
         plt.tight_layout()  # reduce whitespace
         f.savefig(os.path.join(output_path, f"The time series of top10 most frequently record
         plt.close(f)
```

```
<Figure size 432x288 with 0 Axes>
```

2.1.2 Geographical map of average wind speed of each states in 2015

```
In [11]: # Load the parquet file as pandas dataframe
         df = pd.read_parquet('./AWND_2015_by_states.parquet', engine='pyarrow')
```

```
        df.AWND_2015 = df.AWND_2015/10
        df.head()

Out[11]:   STATE  STATE_NAME  AWND_2015
        0    NE    NEBRASKA   4.331857
        1    IA        IOWA   4.368354
        2    WA  WASHINGTON   2.514538
        3    VT     VERMONT   2.345664
        4    NY    NEW YORK   3.411802

In [12]: tls.set_credentials_file(username='dylansp', api_key='I3hOHdVaQKa1gbSLegzU')

In [13]: df.describe()

Out[13]:        AWND_2015
        count  50.000000
        mean    3.308813
        std     0.693650
        min     2.156481
        25%     2.717551
        50%     3.419003
        75%     3.805487
        max     4.659579

In [14]: # Define elements for
        data = [dict(type='choropleth',
                     autocolorscale=True,
                     locations=df.STATE,
                     z=df.AWND_2015,
                     locationmode='USA-states',
                     colorbar=dict(title='Average Wind Speed(m/s)')
                    )
               ]
        data

Out[14]: [{'type': 'choropleth', 'autocolorscale': True, 'locations': 0     NE
          1     IA
          2     WA
          3     VT
          4     NY
          5     IL
          6     AK
          7     RI
          8     ME
          9     TN
          10    FL
          11    KS
          12    HI
          13    CA
```

```
14    ID
15    NH
16    TX
17    MI
18    MS
19    DE
20    WY
21    OK
22    NJ
23    MO
24    VA
25    SC
26    OR
27    NM
28    CO
29    CT
30    AL
31    OH
32    UT
33    MN
34    NC
35    MA
36    WI
37    NV
38    MD
39    PA
40    ND
41    MT
42    AZ
43    WV
44    IN
45    KY
46    AR
47    SD
48    GA
49    LA
Name: STATE, dtype: object, 'z': 0     4.331857
1     4.368354
2     2.514538
3     2.345664
4     3.411802
5     3.884009
6     3.761742
7     3.448950
8     2.777854
9     2.379204
10    3.238691
11    4.256632
```

```
12    3.955446
13    2.715781
14    2.156481
15    3.578222
16    3.877096
17    3.699491
18    2.722862
19    3.448834
20    3.583639
21    4.188444
22    2.782848
23    3.426204
24    3.041210
25    2.667991
26    2.298589
27    4.428667
28    3.806221
29    3.068389
30    2.352806
31    3.507555
32    2.979724
33    3.984323
34    2.630969
35    3.526347
36    3.603933
37    2.614519
38    3.143450
39    2.926130
40    4.590034
41    3.803284
42    4.302700
43    2.576243
44    3.567382
45    2.739963
46    2.645683
47    4.659579
48    2.375226
49    2.745112
Name: AWND_2015, dtype: float64, 'locationmode': 'USA-states', 'colorbar': {'title'
```

```
In [15]: # Define layout
         layout = dict(title='Average Wind Speed for USA States in 2015',
                       geo = dict(scope='usa',
                                  projection=dict(type='albers usa')
                                 )
                      )
         layout
```

```
Out[15]: {'title': 'Average Wind Speed for USA States in 2015',
          'geo': {'scope': 'usa', 'projection': {'type': 'albers usa'}}}
```

```
In [16]: fig = dict(data=data, layout=layout)
         py.iplot(fig, filename='Wind Speed')
```

```
/Users/dylan/anaconda3/lib/python3.7/site-packages/IPython/core/display.py:689: UserWarning:

Consider using IPython.display.IFrame instead
```

```
Out[16]: <chart_studio.tools.PlotlyDisplay object>
```

2.1.3 Time Series of Daily Average Wind Speed for Each States in 2015

```
In [17]: df.count()
```

```
Out[17]: STATE         50
         STATE_NAME    50
         AWND_2015     50
         dtype: int64
```

```
In [18]: # Get 10 representative states
         top_states = list(df.sort_values('AWND_2015', ascending=False).STATE[0:50:10])
         top_states
```

```
Out[18]: ['SD', 'IL', 'MA', 'UT', 'NC']
```

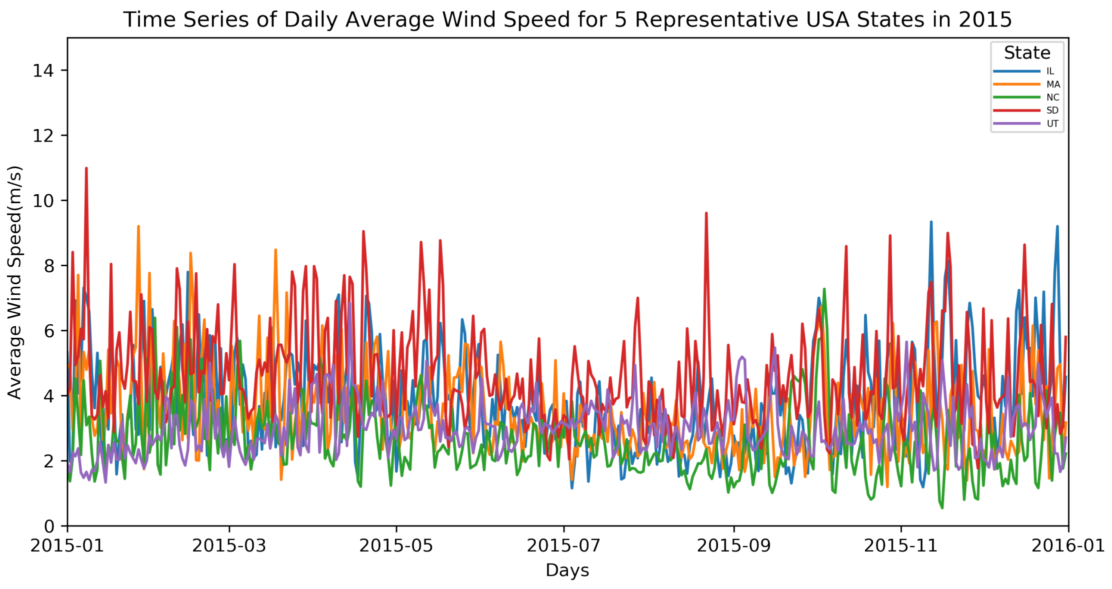
```
In [19]: # Load the parquet file as pandas dataframe
         df = pd.read_parquet('./daily_AWND_by_states.parquet', engine='pyarrow')
         df = df[df.STATE.isin(top_states)]
         df.AVG_STATE_WIND = df.AVG_STATE_WIND/10
         df.head()
```

```
Out[19]:         DATE STATE  AVG_STATE_WIND
         3   20150103    UT        2.253846
         15  20150110    NC        2.208333
         20  20150111    IL        3.616667
         26  20150113    UT        2.566667
         27  20150113    NC        5.066667
```

```
In [20]: df.DATE = pd.to_datetime(df.DATE, format='%Y-%m-%d')
         df.head()
```

```
Out[20]:          DATE STATE  AVG_STATE_WIND
         3  2015-01-03    UT        2.253846
         15 2015-01-10    NC        2.208333
         20 2015-01-11    IL        3.616667
         26 2015-01-13    UT        2.566667
         27 2015-01-13    NC        5.066667
```

```
In [21]: df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1825 entries, 3 to 18226
Data columns (total 3 columns):
DATE              1825 non-null datetime64[ns]
STATE             1825 non-null object
AVG_STATE_WIND    1825 non-null float64
dtypes: datetime64[ns](1), float64(1), object(1)
memory usage: 57.0+ KB


In [22]: df.AVG_STATE_WIND.describe()

Out[22]: count    1825.000000
         mean        3.534853
         std         1.464547
         min         0.545833
         25%         2.479167
         50%         3.247059
         75%         4.311111
         max        10.986667
         Name: AVG_STATE_WIND, dtype: float64

In [23]: data = df.pivot_table(index='DATE',columns='STATE',values='AVG_STATE_WIND')

In [24]: # Plot the time series of frequency of each other element collected by stations all o
         f, a = plt.subplots(dpi=300, figsize=(10, 5))  # affects output resolution (dpi) and 
         a.plot(data, label=data.columns)  # assign label to include in legend
         a.set_ylim([0, 15])  # exapnd axes slightly beyond [1, 6720]
         a.set_xlim("2015-01-01", "2016-01-01")

         # Legend
         a.legend(data.columns, title='State',fontsize=5, handlelength=5)

         # Labels
         a.set_title(f"Time Series of Daily Average Wind Speed for 5 Representative USA States
         a.set_xlabel("Days")
         a.set_ylabel("Average Wind Speed(m/s)")

Out[24]: Text(0, 0.5, 'Average Wind Speed(m/s)')
```



```
In [25]: # Outputs
         output_path = os.path.expanduser("~/Documents/plots")  # M:/plots on windows
         if not os.path.exists(output_path):
             os.makedirs(output_path)

         # Save
         plt.tight_layout()  # reduce whitespace
         f.savefig(os.path.join(output_path, f"Time series of daily average wind speed for 5 s
         plt.close(f)
```

```
<Figure size 432x288 with 0 Axes>
```

```
In [ ]:
```