week2

October 8, 2017

1 Introduction to Pandas

- It is a python 3rd party library
- Used for data analysis and visualization
- Part of Anaconda python distribution
- Best used with Jupyter notebook, can be used with regular python programs
- Main feature is the Data Frame

```
In [1]: # Load the pandas library to let python know you will use it
        import pandas as pd
```

2 What is a Data Frame?

- Its a data structure, like lists and dictionaries
- Consists of rows and columns, similar to SQL tables and excel spreadsheets
- Columns are attributes or variables
- Rows are records or single observations
- Operations are typically performed on columns
- Has both numeric and named indexing

3 Loading data into a data frame

- Data is usually loaded from an external source, like a csv or excel file.
- Download the weather data set from vega-dataset (**right click and save as**)
- Place it in the same directory as the jupyter notbook you are working on

```
In [5]: # load the data using pandas library
        # do you remember what was pd?
        pd.read_csv("weather.csv")

        # Jupter notebook tip:
        # type: pd.
        # then hit tab, see what happens
        # try also: pd.read_ (then hit tab)
```

```
Out[5]:      location              date  precipitation  temp_max  temp_min  wind
        0     Seattle  2012-01-01 00:00            0.0      12.8       5.0   4.7
```

```
1       Seattle  2012-01-02 00:00            10.9        10.6          2.8   4.5
2       Seattle  2012-01-03 00:00             0.8        11.7          7.2   2.3
3       Seattle  2012-01-04 00:00            20.3        12.2          5.6   4.7
4       Seattle  2012-01-05 00:00             1.3         8.9          2.8   6.1
5       Seattle  2012-01-06 00:00             2.5         4.4          2.2   2.2
6       Seattle  2012-01-07 00:00             0.0         7.2          2.8   2.3
7       Seattle  2012-01-08 00:00             0.0        10.0          2.8   2.0
8       Seattle  2012-01-09 00:00             4.3         9.4          5.0   3.4
9       Seattle  2012-01-10 00:00             1.0         6.1          0.6   3.4
10      Seattle  2012-01-11 00:00             0.0         6.1         -1.1   5.1
11      Seattle  2012-01-12 00:00             0.0         6.1         -1.7   1.9
12      Seattle  2012-01-13 00:00             0.0         5.0         -2.8   1.3
13      Seattle  2012-01-14 00:00             4.1         4.4          0.6   5.3
14      Seattle  2012-01-15 00:00             5.3         1.1         -3.3   3.2
15      Seattle  2012-01-16 00:00             2.5         1.7         -2.8   5.0
16      Seattle  2012-01-17 00:00             8.1         3.3          0.0   5.6
17      Seattle  2012-01-18 00:00            19.8         0.0         -2.8   5.0
18      Seattle  2012-01-19 00:00            15.2        -1.1         -2.8   1.6
19      Seattle  2012-01-20 00:00            13.5         7.2         -1.1   2.3
20      Seattle  2012-01-21 00:00             3.0         8.3          3.3   8.2
21      Seattle  2012-01-22 00:00             6.1         6.7          2.2   4.8
22      Seattle  2012-01-23 00:00             0.0         8.3          1.1   3.6
23      Seattle  2012-01-24 00:00             8.6        10.0          2.2   5.1
24      Seattle  2012-01-25 00:00             8.1         8.9          4.4   5.4
25      Seattle  2012-01-26 00:00             4.8         8.9          1.1   4.8
26      Seattle  2012-01-27 00:00             0.0         6.7         -2.2   1.4
27      Seattle  2012-01-28 00:00             0.0         6.7          0.6   2.2
28      Seattle  2012-01-29 00:00            27.7         9.4          3.9   4.5
29      Seattle  2012-01-30 00:00             3.6         8.3          6.1   5.1
...         ...               ...             ...         ...          ...   ...
2892  New York  2015-12-02 00:00             3.0        13.9          8.3   2.0
2893  New York  2015-12-03 00:00             0.0        13.3          7.2   7.2
2894  New York  2015-12-04 00:00             0.0        11.7          5.0   4.7
2895  New York  2015-12-05 00:00             0.0        11.7          1.7   2.4
2896  New York  2015-12-06 00:00             0.0        10.6          3.3   2.9
2897  New York  2015-12-07 00:00             0.0        12.8          4.4   3.4
2898  New York  2015-12-08 00:00             0.0        10.6          4.4   3.5
2899  New York  2015-12-09 00:00             0.0        12.8          1.1   3.4
2900  New York  2015-12-10 00:00             0.0        15.0          8.9   3.0
2901  New York  2015-12-11 00:00             0.0        14.4          7.8   2.7
2902  New York  2015-12-12 00:00             0.0        17.8          9.4   1.9
2903  New York  2015-12-13 00:00             0.0        21.1         11.7   3.1
2904  New York  2015-12-14 00:00             9.1        16.1         11.7   4.8
2905  New York  2015-12-15 00:00             2.3        17.8         11.7   8.2
2906  New York  2015-12-16 00:00             1.3        11.7          7.2   4.1
2907  New York  2015-12-17 00:00            29.7        15.0         10.0   4.1
2908  New York  2015-12-18 00:00             0.3        14.4          3.9   6.1
2909  New York  2015-12-19 00:00             0.0         5.0          2.2   9.0
```

```
2910  New York  2015-12-20 00:00            0.0       6.7       1.7   5.1
2911  New York  2015-12-21 00:00            0.0      12.8       3.3   5.3
2912  New York  2015-12-22 00:00            4.8      15.6      11.1   3.8
2913  New York  2015-12-23 00:00           29.5      17.2       8.9   4.5
2914  New York  2015-12-24 00:00            0.5      20.6      13.9   4.9
2915  New York  2015-12-25 00:00            2.5      17.8      11.1   0.9
2916  New York  2015-12-26 00:00            0.3      15.6       9.4   4.8
2917  New York  2015-12-27 00:00            2.0      17.2       8.9   5.5
2918  New York  2015-12-28 00:00            1.3       8.9       1.7   6.3
2919  New York  2015-12-29 00:00           16.8       9.4       1.1   5.3
2920  New York  2015-12-30 00:00            9.4      10.6       5.0   3.0
2921  New York  2015-12-31 00:00            1.5      11.1       6.1   5.5

      weather
0     drizzle
1        rain
2        rain
3        rain
4        rain
5        rain
6        rain
7         sun
8        rain
9        rain
10        sun
11        sun
12        sun
13       snow
14       snow
15       snow
16       snow
17       snow
18       snow
19       snow
20       rain
21       rain
22       rain
23       rain
24       rain
25       rain
26    drizzle
27       rain
28       rain
29       rain
...       ...
2892      fog
2893      sun
2894      sun
```

```
2895       sun
2896       sun
2897   drizzle
2898       sun
2899       sun
2900   drizzle
2901   drizzle
2902       fog
2903   drizzle
2904       fog
2905       fog
2906       fog
2907       fog
2908       sun
2909       sun
2910       sun
2911       sun
2912       fog
2913       fog
2914       fog
2915       fog
2916   drizzle
2917       fog
2918      snow
2919       fog
2920       fog
2921       fog

[2922 rows x 7 columns]
```

4 Now it is your turn

Download airport.csv then load it into the notebook

Remember: Right click on the link and select **save target as**

```
In [ ]:
```

5 How to work with the data?

- You must place it in a variable so you can refer to it
- The current data was displayed and not assigned to a variable, so you cannot use it
- Assign it to a variable named **my_df**

```
In [3]: my_df = pd.read_csv("weather.csv")
```

```
In [ ]: # Your turn: Load airports.csv into airports_df
```

6 Let us discover how the data looks like

```
In [9]: my_df.head()
```

```
Out[9]:   location                 date  precipitation  temp_max  temp_min  wind  weat
        0  Seattle  2012-01-01 00:00            0.0      12.8       5.0   4.7  driz
        1  Seattle  2012-01-02 00:00           10.9      10.6       2.8   4.5     r
        2  Seattle  2012-01-03 00:00            0.8      11.7       7.2   2.3     r
        3  Seattle  2012-01-04 00:00           20.3      12.2       5.6   4.7     r
        4  Seattle  2012-01-05 00:00            1.3       8.9       2.8   6.1     r
```

```
In [ ]: # You can pass a number in the head() method to show more data
        # show 10 items (try it)

        # do the same for airports_df
```

```
In [13]: # To know which columns are available use the columns attribute
         my_df.columns
```

```
Out[13]: Index(['location', 'date', 'precipitation', 'temp_max', 'temp_min', 'wind'
                'weather'],
               dtype='object')
```

```
In [ ]: # Your turn: explore the columns for airports_df
```

7 Data types

- Each **column** will have its own data type
- Remember, variables will be in columns
- Observations in rows
- Use dtypes attribute of to discover columns and datatypes
- **OOP**: What is the difference between a *function*, a *method*, an *attribute*, and a *variable*?

```
In [18]: my_df.dtypes
```

```
Out[18]: location          object
         date              object
         precipitation    float64
         temp_max         float64
         temp_min         float64
         wind             float64
         weather           object
         dtype: object
```

```
In [ ]: # Your turn: Find out the data types for the airports_df column
```

```
In [18]: # Pandas uses data types provided by numpy
         # load numpy
         import numpy as np
```

```
# convert the column to datetime
my_df.date.astype(np.datetime64)
```

```
Out[18]: 0      2012-01-01
         1      2012-01-02
         2      2012-01-03
         3      2012-01-04
         4      2012-01-05
         5      2012-01-06
         6      2012-01-07
         7      2012-01-08
         8      2012-01-09
         9      2012-01-10
         10     2012-01-11
         11     2012-01-12
         12     2012-01-13
         13     2012-01-14
         14     2012-01-15
         15     2012-01-16
         16     2012-01-17
         17     2012-01-18
         18     2012-01-19
         19     2012-01-20
         20     2012-01-21
         21     2012-01-22
         22     2012-01-23
         23     2012-01-24
         24     2012-01-25
         25     2012-01-26
         26     2012-01-27
         27     2012-01-28
         28     2012-01-29
         29     2012-01-30
                   ...
         2892   2015-12-02
         2893   2015-12-03
         2894   2015-12-04
         2895   2015-12-05
         2896   2015-12-06
         2897   2015-12-07
         2898   2015-12-08
         2899   2015-12-09
         2900   2015-12-10
         2901   2015-12-11
         2902   2015-12-12
         2903   2015-12-13
         2904   2015-12-14
```

```
          2905   2015-12-15
          2906   2015-12-16
          2907   2015-12-17
          2908   2015-12-18
          2909   2015-12-19
          2910   2015-12-20
          2911   2015-12-21
          2912   2015-12-22
          2913   2015-12-23
          2914   2015-12-24
          2915   2015-12-25
          2916   2015-12-26
          2917   2015-12-27
          2918   2015-12-28
          2919   2015-12-29
          2920   2015-12-30
          2921   2015-12-31
          Name: date, dtype: datetime64[ns]
```

```
In [5]: # an alternative way to do it is using
        pd.to_datetime(my_df.date).head() # do you remember head method?
```

```
Out[5]: 0   2012-01-01
        1   2012-01-02
        2   2012-01-03
        3   2012-01-04
        4   2012-01-05
        Name: date, dtype: datetime64[ns]
```

```
In [6]: # now let us examine the date column
        my_df.date.head()

        # why is it still of type object?
        # How to fix it?
```

```
Out[6]: 0    2012-01-01 00:00
        1    2012-01-02 00:00
        2    2012-01-03 00:00
        3    2012-01-04 00:00
        4    2012-01-05 00:00
        Name: date, dtype: object
```

```
In [ ]: # just like the dataframe, the command creates a copy
        # but does not store it
        # We need to replace the old date column with the new one
        my_df.date = my_df.date.astype(np.datetime64)
```

```
In [23]: # check the types
         my_df.dtypes
```

```
Out[23]: location                       object
         date                   datetime64[ns]
         precipitation                 float64
         temp_max                      float64
         temp_min                      float64
         wind                          float64
         weather                        object
         dtype: object
```

```
In [ ]: # Your turn: examine the airports_df dataframe
        # are there any date columns that you can convert?
        # Check then umeric columns, what should their data type be?
```

8 Why convert an object column into a date column?

- As you will find out later, pandas can do more fancy things if it knows the column is a date
- For example:
- Sort
- Filter based on date range
- Date arethmatic
- Always make sure date/time columns have the correct data type

9 Indexing Columns

- Using square brackets [ ]
- Using dot notation .

```
In [7]: # a single column is known as a series
        my_df['location']
```

```
Out[7]: 0         Seattle
        1         Seattle
        2         Seattle
        3         Seattle
        4         Seattle
        5         Seattle
        6         Seattle
        7         Seattle
        8         Seattle
        9         Seattle
        10        Seattle
        11        Seattle
        12        Seattle
        13        Seattle
        14        Seattle
        15        Seattle
        16        Seattle
```

```
17         Seattle
18         Seattle
19         Seattle
20         Seattle
21         Seattle
22         Seattle
23         Seattle
24         Seattle
25         Seattle
26         Seattle
27         Seattle
28         Seattle
29         Seattle
              ...
2892    New York
2893    New York
2894    New York
2895    New York
2896    New York
2897    New York
2898    New York
2899    New York
2900    New York
2901    New York
2902    New York
2903    New York
2904    New York
2905    New York
2906    New York
2907    New York
2908    New York
2909    New York
2910    New York
2911    New York
2912    New York
2913    New York
2914    New York
2915    New York
2916    New York
2917    New York
2918    New York
2919    New York
2920    New York
2921    New York
Name: location, dtype: object
```

```
In [8]: # Some methods that work on Dataframes also work on Series
        my_df['location'].head()
```

```
Out[8]: 0    Seattle
        1    Seattle
        2    Seattle
        3    Seattle
        4    Seattle
        Name: location, dtype: object
```

```
In [13]: # Dot notation to access series
         my_df.location.head()
```

```
Out[13]: 0    Seattle
         1    Seattle
         2    Seattle
         3    Seattle
         4    Seattle
         Name: location, dtype: object
```

```
In [ ]: # Your turn: Try to index the columns for airports_df using square brackets
        # Use head() to get an idea of what the data is
```

```
In [14]: # Descriptive statistics
         my_df['location'].describe()
```

```
Out[14]: count         2922
         unique           2
         top       New York
         freq          1461
         Name: location, dtype: object
```

```
In [15]: # works also on dataframe
         my_df.describe()
```

```
Out[15]:        precipitation     temp_max     temp_min         wind
         count    2922.000000  2922.000000  2922.000000  2922.000000
         mean        2.944764    16.769131     8.612320     4.101129
         std         7.695286     8.644596     7.511776     1.880791
         min         0.000000    -7.700000   -16.000000     0.400000
         25%         0.000000    10.000000     3.300000     2.700000
         50%         0.000000    16.100000     8.900000     3.800000
         75%         1.800000    23.900000    13.900000     5.100000
         max       118.900000    37.800000    26.700000    16.200000
```

```
In [26]: # Different data types will have different descriptives
         my_df['date'].describe()
```

```
Out[26]: count                    2922
         unique                   1461
         top       2013-06-05 00:00:00
         freq                        2
         first     2012-01-01 00:00:00
         last      2015-12-31 00:00:00
         Name: date, dtype: object
```

```
In [21]: my_df.precipitation.describe()
```

```
Out[21]: count    2922.000000
         mean        2.944764
         std         7.695286
         min         0.000000
         25%         0.000000
         50%         0.000000
         75%         1.800000
         max       118.900000
         Name: precipitation, dtype: float64
```

```
In [ ]: # Your turn: Use describe() on airports_df

        # Which columns are included in describe?

        # Try it on the columns that were excluded:

        # Why were these columns excluded?
```

10 You can also plot a dataframe

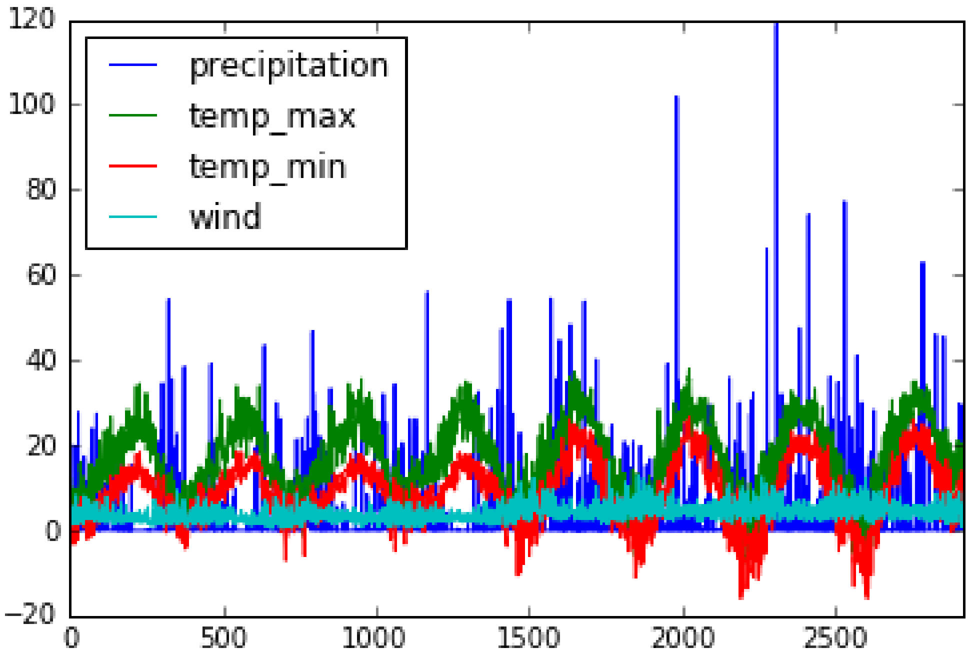
Pandas will try to show it in the best way possible

```
In [ ]: my_df.plot()
```

```
In [29]: # You need to tell pandas that you want to display plots in the notebook
         %matplotlib inline
```

```
In [31]: # now try it
         my_df.plot()
```

```
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x1147020b8>
```



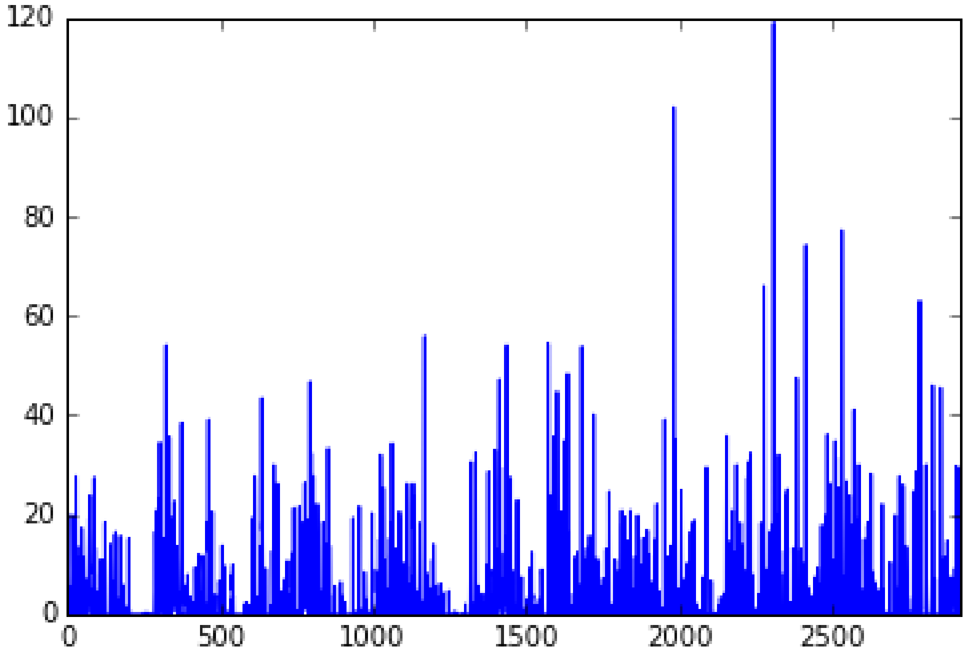
11 Don’t forget!

Always include in your notebook:

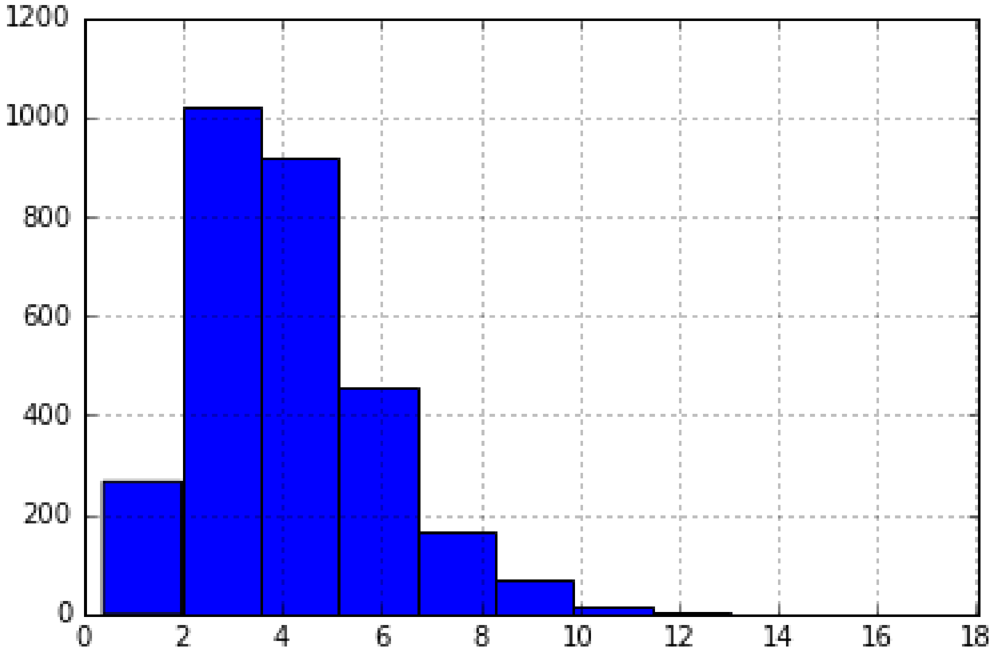
```
# this is the first cell in your notebook
import pandas as pd
%matplotlib inline   # dont forget this
```

```
In [32]: # It's more meaningful to plot Series
         my_df['precipitation'].plot()
```

```
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x1147250f0>
```



```
In [9]: # Remember plots show change from one observation to the next
        my_df['wind'].plot()

        # in some cases it might not be useful

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x10b33f710>

In [40]: # you can try a histogram
         my_df['wind'].hist()

         # Which is useful to know distributions

Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x116d89240>
```



12 How can you find out if percipitation is usually high in the year or low?

```
In [ ]: # Your turn:
```

```
In [45]: # Sometime pandas cannot plot it
         my_df['location'].plot()
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-45-e083b00ff51a> in <module>()
          1 # Sometime pandas cannot plot it
    ----> 2 my_df['location'].plot()


        /Users/koutbo6/anaconda/lib/python3.5/site-packages/pandas/tools/plotting.p
       3564                              colormap=colormap, table=table, yerr=yerr,
       3565                              xerr=xerr, label=label, secondary_y=secondar
    -> 3566                              **kwds)
       3567     __call__.__doc__ = plot_series.__doc__
       3568
        /Users/koutbo6/anaconda/lib/python3.5/site-packages/pandas/tools/plotting.p
       2643                       yerr=yerr, xerr=xerr,
       2644                       label=label, secondary_y=secondary_y,
    -> 2645                       **kwds)
       2646
       2647


        /Users/koutbo6/anaconda/lib/python3.5/site-packages/pandas/tools/plotting.p
       2439          plot_obj = klass(data, subplots=subplots, ax=ax, kind=kind, **k
       2440
    -> 2441      plot_obj.generate()
       2442      plot_obj.draw()
       2443      return plot_obj.result


        /Users/koutbo6/anaconda/lib/python3.5/site-packages/pandas/tools/plotting.p
       1024      def generate(self):
       1025          self._args_adjust()
    -> 1026          self._compute_plot_data()
       1027          self._setup_subplots()
       1028          self._make_plot()


        /Users/koutbo6/anaconda/lib/python3.5/site-packages/pandas/tools/plotting.p
       1133          if is_empty:
       1134              raise TypeError('Empty {0!r}: no numeric data to '
    -> 1135                              'plot'.format(numeric_data.__class__.__name


       1137          self.data = numeric_data


        TypeError: Empty 'DataFrame': no numeric data to plot
```

```
In [ ]: # Your turn: try to plot the columns in airport_df using either plot() or h


        # What can you find out about the data?
```

```
In [49]: # Such variables are usually categorical and you can get frequencies like
         my_df['location'].value_counts()
```

```
Out[49]: New York    1461
         Seattle     1461
         Name: location, dtype: int64
```

```
In [ ]: # Your turn: Examine the columns for airports_df
        # what would be the best columns to check frequencies for?
        # try it:


        # The best columns are:

        # The reason frequencies is best calculated on them is because:

        # What did you find out about your data?
```

13 Are data frames immutable?

- Yes, however, all operations that change values will produce a copy and not change the original
- You have to use assignment to change columns or dataframes
- **So be careful!**