Statistics from Stock Data

July 10, 2018

1 Statistics from Stock Data

In this lab we will load stock data into a Pandas Dataframe and calculate some statistics on it. We will be working with stock data from Google, Apple, and Amazon. All the stock data was downloaded from yahoo finance in CSV format. In your workspace you should have a file named GOOG.csv containing the Google stock data, a file named AAPL.csv containing the Apple stock data, and a file named AMZN.csv containing the Amazon stock data. (You can see the workspace folder by clicking on the Jupyter logo in the upper left corner of the workspace.) All the files contain 7 columns of data:

Date Open High Low Close Adj_Close Volume

We will start by reading in any of the above CSV files into a DataFrame and see what the data looks like.

In [24]:
```
# We import pandas into Python
import pandas as pd

# We read in a stock data data file into a data frame and see what it looks like
df = pd.read_csv('./GOOG.csv')

# We display the first 5 rows of the DataFrame
df.head()
```

Out[24]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2004-08-19	49.676899	51.693783	47.669952	49.845802	49.845802	44994500
1	2004-08-20	50.178635	54.187561	49.925285	53.805050	53.805050	23005800
2	2004-08-23	55.017166	56.373344	54.172661	54.346527	54.346527	18393200
3	2004-08-24	55.260582	55.439419	51.450363	52.096165	52.096165	15361800
4	2004-08-25	52.140873	53.651051	51.604362	52.657513	52.657513	9257400

We clearly see that the Dataframe is has automatically labeled the row indices using integers and has labeled the columns of the DataFrame using the names of the columns in the CSV files.

2 To Do

You will now load the stock data from Google, Apple, and Amazon into separte DataFrames. However, for each stock data you will only be interested in loading the `Date` and `Adj Close` columns into the Dataframe. In addtion, you want to use the `Date` column as your row index.

Finally, you want the DataFrame to recognize the dates as actual dates (year/month/day) and not as strings. For each stock, you can accomplish all theses things in just one line of code by using the appropiate keywords in the `pd.read_csv()` function. Here are a few hints:

- Use the `index_col` keyword to indicate which column you want to use as an index. For example `index_col = ['Open']`
- Set the `parse_dates` keyword equal to `True` to convert the Dates into real dates of the form year/month/day
- Use the `usecols` keyword to select which columns you want to load into the DataFrame. For example `usecols = ['Open', 'High']`

Fill in the code below:

```
In [25]: # We load the Google stock data into a DataFrame
         google_stock = pd.read_csv('./GOOG.csv')

         # We load the Apple stock data into a DataFrame
         apple_stock = pd.read_csv('./AAPL.csv')

         # We load the Amazon stock data into a DataFrame
         amazon_stock = pd.read_csv('./AMZN.csv')
```

You can check that you have loaded the data correctly by displaying the head of the DataFrames.

```
In [26]: # We display the google_stock DataFrame
         amazon_stock.head()
```

Out[26]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2000-01-03	81.5000	89.5625	79.046799	89.3750	89.3750	16117600
1	2000-01-04	85.3750	91.5000	81.750000	81.9375	81.9375	17487400
2	2000-01-05	70.5000	75.1250	68.000000	69.7500	69.7500	38457400
3	2000-01-06	71.3125	72.6875	64.000000	65.5625	65.5625	18752000
4	2000-01-07	67.0000	70.5000	66.187500	69.5625	69.5625	10505400

You will now join the three DataFrames above to create a single new DataFrame that contains all the `Adj Close` for all the stocks. Let's start by creating an empty DataFrame that has as row indices calendar days between `2000-01-01` and `2016-12-31`. We will use the `pd.date_range()` function to create the calendar dates first and then we will create a DataFrame that uses those dates as row indices:

```
In [39]: # We create calendar dates between '2000-01-01' and  '2016-12-31'
         dates = pd.date_range('2000-01-01', '2016-12-31')

         # We create and empty DataFrame that uses the above dates as indices
         all_stocks = pd.DataFrame(index = dates)
```

3 To Do

You will now join the the individual DataFrames, `google_stock`, `apple_stock`, and `amazon_stock`, to the `all_stocks` DataFrame. However, before you do this, it is necessary that you change the name of the columns in each of the three dataframes. This is because the column labels in the `all_stocks` dataframe must be unique. Since all the columns in the individual dataframes have the same name, `Adj Close`, we must change them to the stock name before joining them. In the space below change the column label `Adj Close` of each individual dataframe to the name of the corresponding stock. You can do this by using the `pd.DataFrame.rename()` function.

```
In [29]: # Change the Adj Close column label to Google
         google_stock =google_stock.rename(columns = {'Adj Close': 'Adj Close1'})


         # Change the Adj Close column label to Apple
         apple_stock = apple_stock.rename(columns = {'Adj Close': 'Adj Close2'})

         # Change the Adj Close column label to Amazon
         amazon_stock = amazon_stock.rename(columns = {'Adj Close': 'Adj Close1'})
```

You can check that the column labels have been changed correctly by displaying the datadrames

```
In [30]: # We display the google_stock DataFrame
         google_stock.head()
```

```
Out[30]:          Date       Open       High        Low      Close  Adj Close1  \
         0  2004-08-19  49.676899  51.693783  47.669952  49.845802   49.845802
         1  2004-08-20  50.178635  54.187561  49.925285  53.805050   53.805050
         2  2004-08-23  55.017166  56.373344  54.172661  54.346527   54.346527
         3  2004-08-24  55.260582  55.439419  51.450363  52.096165   52.096165
         4  2004-08-25  52.140873  53.651051  51.604362  52.657513   52.657513

             Volume
         0  44994500
         1  23005800
         2  18393200
         3  15361800
         4   9257400
```

Now that we have unique column labels, we can join the individual DataFrames to the `all_stocks` DataFrame. For this we will use the `dataframe.join()` function. The function `dataframe1.join(dataframe2)` joins `dataframe1` with `dataframe2`. We will join each dataframe one by one to the `all_stocks` dataframe. Fill in the code below to join the dataframes, the first join has been made for you:

```
In [71]: all_stocks = pd.DataFrame(index = dates)

         # We join the Google stock to all_stocks
```

```
all_stocks = all_stocks.join(apple_stock)

# We join the Apple stock to all_stocks
all_stocks = all_stocks.join(google_stock)

# We join the Amazon stock to all_stocks
all_stocks = all_stocks.join(amazon_stock)
```

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

ValueError                                Traceback (most recent call last)

<ipython-input-71-c8e5bd7c1239> in <module>()
  5
  6 # We join the Apple stock to all_stocks
----> 7 all_stocks = all_stocks.join(google_stock)
  8
  9 # We join the Amazon stock to all_stocks


/opt/conda/lib/python3.6/site-packages/pandas/core/frame.py in join(self, other, on, how
4667         # For SparseDataFrame's benefit
4668         return self._join_compat(other, on=on, how=how, lsuffix=lsuffix,
-> 4669                                  rsuffix=rsuffix, sort=sort)
4670
4671     def _join_compat(self, other, on=None, how='left', lsuffix='', rsuffix='',


/opt/conda/lib/python3.6/site-packages/pandas/core/frame.py in _join_compat(self, other,
4682             return merge(self, other, left_on=on, how=how,
4683                          left_index=on is None, right_index=True,
-> 4684                          suffixes=(lsuffix, rsuffix), sort=sort)
4685         else:
4686             if on is not None:


/opt/conda/lib/python3.6/site-packages/pandas/core/reshape/merge.py in merge(left, right
 52                          right_index=right_index, sort=sort, suffixes=suffixes,
 53                          copy=copy, indicator=indicator)
---> 54     return op.get_result()
 55
 56


/opt/conda/lib/python3.6/site-packages/pandas/core/reshape/merge.py in get_result(self)
573
574         llabels, rlabels = items_overlap_with_suffix(ldata.items, lsuf,
```

```
--> 575                                                    rdata.items, rsuf)
    576
    577         lindexers = {1: left_indexer} if left_indexer is not None else {}


/opt/conda/lib/python3.6/site-packages/pandas/core/internals.py in items_overlap_with_su
   4699         if not lsuffix and not rsuffix:
   4700             raise ValueError('columns overlap but no suffix specified: %s' %
-> 4701                              to_rename)
   4702
   4703         def lrenamer(x):


ValueError: columns overlap but no suffix specified: Index(['Date', 'Open', 'High', 'Low
```

You can check that the dataframes have been joined correctly by displaying the `all_stocks` dataframe

```
In [72]: # We display the google_stock DataFrame
         all_stocks.head()
```

Out[72]:

	Date	Open	High	Low	Close	Adj Close2	Volume
2000-01-01 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2000-01-02 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2000-01-03 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2000-01-04 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2000-01-05 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN

4 To Do

Before we proceed to get some statistics on the stock data, let's first check that we don't have any *NaN* values. In the space below check if there are any *NaN* values in the `all_stocks` dataframe. If there are any, remove any rows that have *NaN* values:

```
In [70]: # Check if there are any NaN values in the all_stocks dataframe

         all_stocks.isnull()
         a =  all_stocks.isnull().sum().sum()
         print('Number of NaN values in our DataFrame:', a)

         # Remove any rows that contain NaN values
         all_stocks.dropna(axis = 0)
```

```
Number of NaN values in our DataFrame: 43470
```

```
Out[70]: Empty DataFrame
         Columns: [Date, Open, High, Low, Close, Adj Close1, Volume]
         Index: []
```

Now that you have eliminated any *NaN* values we can now calculate some basic statistics on the stock prices. Fill in the code below

```
In [81]: # Print the average stock price for each stock
         import numpy as np

         print()
         print('avg(google) = \n', np.mean(google_stock))
         print()
         print('avg(apple) = \n', np.mean(apple_stock))
         print()
         print('avg(amazon) = \n', np.mean(amazon_stock))

         # Print the median stock price for each stock
         print()
         print('Median of all elements in X:', np.median(google_stock))
         print()
         print('avg(apple) = \n', np.median(apple_stock))
         print()
         print('avg(amazon) = \n', np.median(amazon_stock))

         # Print the standard deviation of the stock price for each stock

         print('Standard Deviation of all elements in X:', google_stock.std())
         print('Standard Deviation of all elements in X:', google_stock.std())
         print('Standard Deviation of all elements in X:', google_stock.std())
         # Print the correlation between stocks


avg(google) =
 Open          3.801861e+02
High          3.834937e+02
Low           3.765193e+02
Close         3.800725e+02
Adj Close1    3.800725e+02
Volume        8.038476e+06
dtype: float64

avg(apple) =
 Open          4.256831e+01
High          4.296574e+01
Low           4.213103e+01
Close         4.255951e+01
Adj Close2    4.006996e+01
Volume        1.254019e+08
dtype: float64

avg(amazon) =
```

```
 Open          1.997222e+02
High          2.019472e+02
Low           1.973396e+02
Close         1.997543e+02
Adj Close1    1.997543e+02
Volume        6.786860e+06
dtype: float64
```

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

TypeError                                 Traceback (most recent call last)

<ipython-input-81-355dc83f9d96> in <module>()
     11 # Print the median stock price for each stock
     12 print()
---> 13 print('Median of all elements in X:', np.median(google_stock))
     14 print()
     15 print('avg(apple) = \n', np.median(apple_stock))


/opt/conda/lib/python3.6/site-packages/numpy/lib/function_base.py in median(a, axis, out
   3942     """
   3943     r, k = _ureduce(a, func=_median, axis=axis, out=out,
-> 3944                     overwrite_input=overwrite_input)
   3945     if keepdims:
   3946         return r.reshape(k)


/opt/conda/lib/python3.6/site-packages/numpy/lib/function_base.py in _ureduce(a, func, *
   3856         keepdim = [1] * a.ndim
   3857
-> 3858     r = func(a, **kwargs)
   3859     return r, keepdim
   3860


/opt/conda/lib/python3.6/site-packages/numpy/lib/function_base.py in _median(a, axis, ou
   3975             part = a
   3976     else:
-> 3977         part = partition(a, kth, axis=axis)
   3978
   3979     if part.shape == ():


/opt/conda/lib/python3.6/site-packages/numpy/core/fromnumeric.py in partition(a, kth, ax
```

```
    637     else:
    638         a = asanyarray(a).copy(order="K")
--> 639     a.partition(kth, axis=axis, kind=kind, order=order)
    640     return a
    641

TypeError: '>' not supported between instances of 'float' and 'str'
```

We will now look at how we can compute some rolling statistics, also known as moving statistics. We can calculate for example the rolling mean (moving average) of the Google stock price by using the Pandas `dataframe.rolling().mean()` method. The `dataframe.rolling(N).mean()` calculates the rolling mean over an `N`-day window. In other words, we can take a look at the average stock price every `N` days using the above method. Fill in the code below to calculate the average stock price every 150 days for Google stock

```
In [87]: # We compute the rolling mean using a 150-Day window for Google stock
         rollingMean = all_stocks['google'].rolling(150).mean()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)

/opt/conda/lib/python3.6/site-packages/pandas/core/indexes/base.py in get_loc(self, key,
   2441             try:
-> 2442                 return self._engine.get_loc(key)
   2443             except KeyError:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc (pandas/_libs/index.c:5

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc (pandas/_libs/index.c:5

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_

KeyError: 'google'

During handling of the above exception, another exception occurred:
```

```
KeyError                                  Traceback (most recent call last)

<ipython-input-87-344d194aabe9> in <module>()
  1 # We compute the rolling mean using a 150-Day window for Google stock
----> 2 rollingMean = all_stocks['google'].rolling(150).mean()
  3


/opt/conda/lib/python3.6/site-packages/pandas/core/frame.py in __getitem__(self, key)
1962             return self._getitem_multilevel(key)
1963         else:
-> 1964             return self._getitem_column(key)
1965
1966     def _getitem_column(self, key):


/opt/conda/lib/python3.6/site-packages/pandas/core/frame.py in _getitem_column(self, key
1969         # get column
1970         if self.columns.is_unique:
-> 1971             return self._get_item_cache(key)
1972
1973         # duplicate columns & possible reduce dimensionality


/opt/conda/lib/python3.6/site-packages/pandas/core/generic.py in _get_item_cache(self, i
1643         res = cache.get(item)
1644         if res is None:
-> 1645             values = self._data.get(item)
1646             res = self._box_item_values(item, values)
1647             cache[item] = res


/opt/conda/lib/python3.6/site-packages/pandas/core/internals.py in get(self, item, fastp
3588
3589             if not isnull(item):
-> 3590                 loc = self.items.get_loc(item)
3591             else:
3592                 indexer = np.arange(len(self.items))[isnull(self.items)]


/opt/conda/lib/python3.6/site-packages/pandas/core/indexes/base.py in get_loc(self, key,
2442                 return self._engine.get_loc(key)
2443             except KeyError:
-> 2444                 return self._engine.get_loc(self._maybe_cast_indexer(key))
2445
2446         indexer = self.get_indexer([key], method=method, tolerance=tolerance)
```

```
pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc (pandas/_libs/index.c:5

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc (pandas/_libs/index.c:5

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_

KeyError: 'google'
```

We can also visualize the rolling mean by plotting the data in our dataframe. In the following lessons you will learn how to use **Matplotlib** to visualize data. For now I will just import matplotlib and plot the Google stock data on top of the rolling mean. You can play around by changing the rolling mean window and see how the plot changes.

```
In [88]: # this allows plots to be rendered in the notebook
         %matplotlib inline

         # We import matplotlib into Python
         import matplotlib.pyplot as plt


         # We plot the Google stock data
         plt.plot(all_stocks['google'])

         # We plot the rolling mean ontop of our Google stock data
         plt.plot(rollingMean)
         plt.legend(['Google Stock Price', 'Rolling Mean'])
         plt.show()


         ---------------------------------------------------------------------------

         KeyError                                  Traceback (most recent call last)

         /opt/conda/lib/python3.6/site-packages/pandas/core/indexes/base.py in get_loc(self, key,
        2441             try:
     -> 2442                 return self._engine.get_loc(key)
        2443             except KeyError:


         pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc (pandas/_libs/index.c:5
```

```
    pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc (pandas/_libs/index.c:5

    pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_

    pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_

    KeyError: 'google'


During handling of the above exception, another exception occurred:


    KeyError                                  Traceback (most recent call last)

    <ipython-input-88-a64b97e05cfa> in <module>()
      7
      8 # We plot the Google stock data
----> 9 plt.plot(all_stocks['google'])
     10
     11 # We plot the rolling mean ontop of our Google stock data


    /opt/conda/lib/python3.6/site-packages/pandas/core/frame.py in __getitem__(self, key)
   1962             return self._getitem_multilevel(key)
   1963         else:
-> 1964             return self._getitem_column(key)
   1965
   1966     def _getitem_column(self, key):


    /opt/conda/lib/python3.6/site-packages/pandas/core/frame.py in _getitem_column(self, key
   1969         # get column
   1970         if self.columns.is_unique:
-> 1971             return self._get_item_cache(key)
   1972
   1973         # duplicate columns & possible reduce dimensionality


    /opt/conda/lib/python3.6/site-packages/pandas/core/generic.py in _get_item_cache(self, i
   1643         res = cache.get(item)
   1644         if res is None:
-> 1645             values = self._data.get(item)
   1646             res = self._box_item_values(item, values)
```

```
   1647             cache[item] = res


    /opt/conda/lib/python3.6/site-packages/pandas/core/internals.py in get(self, item, fastp
   3588
   3589             if not isnull(item):
-> 3590                 loc = self.items.get_loc(item)
   3591             else:
   3592                 indexer = np.arange(len(self.items))[isnull(self.items)]


    /opt/conda/lib/python3.6/site-packages/pandas/core/indexes/base.py in get_loc(self, key,
   2442                 return self._engine.get_loc(key)
   2443             except KeyError:
-> 2444                 return self._engine.get_loc(self._maybe_cast_indexer(key))
   2445
   2446         indexer = self.get_indexer([key], method=method, tolerance=tolerance)


    pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc (pandas/_libs/index.c:5


    pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc (pandas/_libs/index.c:5


    pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_


    pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_


    KeyError: 'google'
```