AI in Finance
Big Data Analytics
Quantitative Investing with Pandas in Python

1081AIFBDA06
TLVXM2A (M2449) (8497) (Fall 2019)
(MBA, DBETKU) (3 Credits, Required) [Full English Course]
(Master’s Program in Digital Business and Economics)
Tue, 2, 3, 4, (9:10-12:00) (B1012)

Min-Yuh Day, Ph.D.
Associate Professor
Department of Information Management
Tamkang University

http://mail.tku.edu.tw/myday

2019-11-05
<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject/Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2019/09/10</td>
<td>Course Orientation on AI in Finance Big Data Analytics</td>
</tr>
<tr>
<td>2</td>
<td>2019/09/17</td>
<td>AI in FinTech: Financial Services Innovation and Application</td>
</tr>
<tr>
<td>3</td>
<td>2019/09/24</td>
<td>ABC: AI, Big Data, Cloud Computing</td>
</tr>
<tr>
<td>4</td>
<td>2019/10/01</td>
<td>Business Models of Fintech</td>
</tr>
<tr>
<td>5</td>
<td>2019/10/08</td>
<td>Event Studies in Finance</td>
</tr>
<tr>
<td>6</td>
<td>2019/10/15</td>
<td>Case Study on AI in Finance Big Data Analytics I</td>
</tr>
<tr>
<td>7</td>
<td>2019/10/22</td>
<td>Foundations of AI in Finance Big Data Analytics with Python</td>
</tr>
<tr>
<td>8</td>
<td>2019/10/29</td>
<td>Case Study on Financial Industry Practice I</td>
</tr>
<tr>
<td>9</td>
<td>2019/11/05</td>
<td>Quantitative Investing with Pandas in Python</td>
</tr>
<tr>
<td>Week</td>
<td>Date</td>
<td>Subject/Topics</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>10</td>
<td>2019/11/12</td>
<td>Midterm Project Report</td>
</tr>
<tr>
<td>11</td>
<td>2019/11/19</td>
<td>Machine Learning in Finance Application with Scikit-Learn In Python</td>
</tr>
<tr>
<td>13</td>
<td>2019/12/03</td>
<td>Case Study on AI in Finance Big Data Analytics II</td>
</tr>
<tr>
<td>14</td>
<td>2019/12/10</td>
<td>Deep Learning for Financial Time Series Forecasting with TensorFlow II</td>
</tr>
<tr>
<td>15</td>
<td>2019/12/17</td>
<td>Case Study on Financial Industry Practice II</td>
</tr>
<tr>
<td>17</td>
<td>2019/12/31</td>
<td>Final Project Presentation I</td>
</tr>
<tr>
<td>18</td>
<td>2020/01/07</td>
<td>Final Project Presentation II</td>
</tr>
</tbody>
</table>
Quantitative Investing with Pandas in Python
Outline

• Quantitative Investing with Pandas in Python
  – Numpy
  – Pandas
Python in Google Colab

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

```python
# !pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt

# Read Stock Data from Yahoo Finance
end = dt.datetime.now()
start = dt.datetime(2016, 1, 1)
df = web.DataReader('AAPL', 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.head()

df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12, 9))
top = plt.subplot2grid((1, 2), (0, 0), rowspan=10, colspan=9)
bottom = plt.subplot2grid((1, 2), (0, 1), rowspan=8, colspan=9)
top.plot(df.index, df['Adj Close'], color='blue') # df.index gives the dates
bottom.bar(df.index, df['Volume'])

# set the labels
plt.xticks(rotation=90)
top.set_title('AAPL')
top.set_xlabel('Adj Close')
top.set_ylabel('AAPL')
bottom.set_ylabel('Volume')

df['Adj Close'].plot()
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')

# simple moving averages
df['MA5'] = df['Adj Close'].rolling(5).mean() # 5 days
df['MA20'] = df['Adj Close'].rolling(20).mean() # 20 days
df['MA60'] = df['Adj Close'].rolling(60).mean() # 60 days

df = pd.DataFrame({'Adj Close': df['MA5'], 'MA20': df['MA20'], 'MA60': df['MA60']})
df.plot(figsize=(12, 9), legend=True, title='AAPL')

df.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
```
Big Data Analytics with Numpy in Python
Numpy

NumPy
Base
N-dimensional array package
NumPy is the fundamental package for scientific computing with Python.

Source: http://www.numpy.org/
• **NumPy** provides a **multidimensional array** object to store homogenous or heterogeneous data; it also provides **optimized functions/methods** to operate on this array object.

Source: Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly
NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

NumPy is licensed under the BSD license, enabling reuse with few restrictions.

Getting Started

- Getting NumPy
- Installing the SciPy Stack
- NumPy and SciPy documentation page
- NumPy Tutorial
- NumPy for MATLAB® Users
- NumPy functions by category
- NumPy Mailing List

For more information on the SciPy Stack (for which NumPy provides the fundamental array data structure), see scipy.org.

http://www.numpy.org/
NumPy ndarray

### One-dimensional Array (1-D Array)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

### Two-dimensional Array (2-D Array)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m-1</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
v = list(range(1, 6))
v
2 * v

import numpy as np
v = np.arange(1, 6)
v
2 * v
```python
1 v = list(range(1, 6))
2 v

[1, 2, 3, 4, 5]

1 2 * v

[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]

1 import numpy as np
2 v = np.arange(1, 6)
3 v

array([[1, 2, 3, 4, 5]])

1 2 * v

array([[2, 4, 6, 8, 10]])
```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
c = a * b

c

Source: Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly
```python
import numpy as np

a = np.zeros((2,2))  # Create an array of all zeros
print(a)  # Prints "[[ 0.  0.]
            #    [ 0.  0.]]"

b = np.ones((1,2))  # Create an array of all ones
print(b)  # Prints "[[ 1.  1.]]"

c = np.full((2,2), 7)  # Create a constant array
print(c)  # Prints "[[ 7.  7.]
            #    [ 7.  7.]]"

d = np.eye(2)  # Create a 2x2 identity matrix
print(d)  # Prints "[[ 1.  0.]
            #    [ 0.  1.]]"

e = np.random.random((2,2))  # Create an array filled with random values
print(e)  # Might print "[[ 0.91940167 0.08143941]
            #    [ 0.68744134 0.87236687]]"
```

```
[[ 0.  0.]
 [0.  0.]]
[[ 1.  1.]]
[[ 7  7]
 [7  7]]
[[ 1.  0.]
 [0.  1.]]
[[ 0.66258211 0.65552598]
 [0.00429934 0.21695824]]
```
Quickstart tutorial

Prerequisites

Before reading this tutorial you should know a bit of Python. If you would like to refresh your memory, take a look at the Python tutorial.

If you wish to work the examples in this tutorial, you must also have some software installed on your computer. Please see http://scipy.org/install.html for instructions.

The Basics

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called axes. The number of axes is rank.

For example, the coordinates of a point in 3D space [1, 2, 1] is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2-dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

```python
[[ 1., 0., 0.],
 [ 0., 1., 2.]]
```

NumPy's array class is called ndarray. It is also known by the alias array. Note that numpy.array is not the same as the Standard Python Library class array.array, which only handles one-dimensional arrays and offers less functionality. The more important attributes of an ndarray object are:

```python
ndarray.ndim
   the number of axes (dimensions) of the array. In the Python world, the number of dimensions is referred to as rank.

ndarray.shape
```

https://docs.scipy.org/doc/numpy-dev/user/quickstart.html
```python
import numpy as np
a = np.arange(15).reshape(3, 5)
a.shape
a.ndim
a.dtype.name
```

```
import numpy as np
a = np.arange(15).reshape(3, 5)
array([[ 0,  1,  2,  3,  4],
      [ 5,  6,  7,  8,  9],
      [10, 11, 12, 13, 14]])
```

```
print(a.shape)
(3, 5)
a.ndim
2
a.dtype.name
'int64'
```
Matrix

\[ \begin{pmatrix}
  a_{1,1} & a_{1,2} & a_{1,3} & \cdots \\
  a_{2,1} & a_{2,2} & a_{2,3} & \cdots \\
  a_{3,1} & a_{3,2} & a_{3,3} & \cdots \\
  \vdots & \vdots & \vdots & \ddots
\end{pmatrix} \]

**m**-by-**n** matrix

\( m \) rows

\( n \) columns

\( i \) changes

\( j \) changes

Source: https://simple.wikipedia.org/wiki/Matrix_(mathematics)
NumPy ndarray: Multidimensional Array Object
NumPy ndarray

One-dimensional Array (1-D Array)

Two-dimensional Array (2-D Array)
import numpy as np
a = np.array([1,2,3,4,5])

One-dimensional Array
(1-D Array)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

a = np.array([[1,2,3,4,5]])
a
array([1, 2, 3, 4, 5])
Two-dimensional Array
(2-D Array)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>m-1</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

```python
a = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10], [11, 12, 13, 14, 15], [16, 17, 18, 19, 20]])
a
```

```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20]])
```
```python
import numpy as np

a = np.array([[0, 1, 2, 3],
              [10, 11, 12, 13],
              [20, 21, 22, 23]])

a
```

```
+---+---+---+---+
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
</tbody>
</table>
+---+---+---+---+
\[ a = \text{np.array}([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]]) \]
NumPy Basics: Arrays and Vectorized Computation
NumPy Array

axis 0

0

axis 1

0 1 2

0 0,0 0,1 0,2

1 1,0 1,1 1,2

2 2,0 2,1 2,2

Numpy Array

<table>
<thead>
<tr>
<th>Expression</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>arr[:2, 1:]</td>
<td>(2, 2)</td>
</tr>
<tr>
<td>arr[2]</td>
<td>(3,)</td>
</tr>
<tr>
<td>arr[2, :]</td>
<td>(3,)</td>
</tr>
<tr>
<td>arr[2:, :]</td>
<td>(1, 3)</td>
</tr>
<tr>
<td>arr[ :, 2]</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>arr[1, :2]</td>
<td>(2,)</td>
</tr>
<tr>
<td>arr[1:2, :2]</td>
<td>(1, 2)</td>
</tr>
</tbody>
</table>


Materials and IPython notebooks for "Python for Data Analysis" by Wes McKinney, published by O'Reilly Media

<table>
<thead>
<tr>
<th>52 commits</th>
<th>2 branches</th>
<th>0 releases</th>
<th>6 contributors</th>
</tr>
</thead>
</table>

Branch: 2nd-edition

<table>
<thead>
<tr>
<th>File Type</th>
<th>Description</th>
<th>Last Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasets</td>
<td>Add Kaggle titanic dataset</td>
<td>5 months ago</td>
</tr>
<tr>
<td>examples</td>
<td>Remove sex column from tips dataset</td>
<td>4 months ago</td>
</tr>
<tr>
<td>.gitignore</td>
<td>Add .gitignore</td>
<td>2 years ago</td>
</tr>
<tr>
<td>COPYING</td>
<td>Use MIT license for code examples</td>
<td>a month ago</td>
</tr>
<tr>
<td>README.md</td>
<td>Add launch in Azure Notebooks button (#70)</td>
<td>19 days ago</td>
</tr>
<tr>
<td>appa.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch02.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch03.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch04.ipynb</td>
<td>Convert all notebooks to v4 format</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch05.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch06.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch07.ipynb</td>
<td>Convert all notebooks to v4 format</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch08.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch09.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch10.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
</tbody>
</table>

https://github.com/wesm/pydata-book
NumPy Basics: Arrays and

```python
In [ ]:
import numpy as np
np.random.seed(12345)
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(10, 5))
np.set_printoptions(precision=4, suppress=True)

In [ ]:
import numpy as np
my_arr = np.arange(1000000)
my_list = list(range(1000000))

In [ ]:
%time for _ in range(10): my_arr2 = my_arr * 2
%time for _ in range(10): my_list2 = [x * 2 for x in my_list]
```

The NumPy `ndarray`: A Multidimensional Array Object

```python
In [ ]:
import numpy as np
# Generate some random data
data = np.random.randn(2, 3)
data
```
Python Pandas
Python Data Analysis Library

*pandas* is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the *Python* programming language.

*pandas* is a NUMFocus sponsored project. This will help ensure the success of development of *pandas* as a world-class open-source project.

**A Fiscally Sponsored Project of**

NUMFOCUS

**OPEN CODE = BETTER SCIENCE**

0.19.2 Final (December 24, 2016)

This is a minor bug-fix release in the 0.19.x series and includes some small regression fixes, bug fixes and performance improvements.

Highlights include:

- Compatibility with Python 3.6

http://pandas.pydata.org/
pandas

Python Data Analysis Library

providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Source: http://pandas.pydata.org/
pandas Ecosystem

- **Statistics and Machine Learning**
  - Statsmodels
  - sklearn-pandas

- **Visualization**
  - Bokeh
  - yhat/ggplot
  - Seaborn
  - Vincent
  - IPython Vega
  - Plotly
  - Pandas-Qt

- **IDE**
  - IPython
  - quantopian/qgrid
  - Spyder

- **API**
  - pandas-datareader
  - quandl/Python
  - pydatastream
  - pandaSDMX
  - fredapi

- **Domain Specific**
  - Geopandas
  - xarray

- **Out-of-core**
  - Dask
  - Blaze
  - Odo

pandas-datareader

pandas-datareader

Up to date remote data access for pandas, works for multiple versions of pandas.

⚠️ Warning

As of v0.6.0 Yahoo!, Google Options, Google Quotes and EDGAR have been immediately deprecated due to large changes in their API and no stable replacement.

ℹ️ Note

As of v0.6.0 Google finance is still functioning for historical price data, although there are frequent reports of failures. Failure is frequently encountered when bulk downloading historical price data.

Usage

Starting in 0.19.0, pandas no longer supports `pandas.io.data` or `pandas.io.wb`, so you must replace your imports from `pandas.io` with those from `pandas_datareader`:

```python
from pandas.io import data, wb
# becomes
from pandas_datareader import data, wb
```

Many functions from the data module have been included in the top level API.

Get Financial Data Directly into Python

Get millions of financial and economic datasets from hundreds of publishers directly into Python.

Load Quandl Data Directly Into Python

All the Data You Want
Quandl unifies financial and economic datasets from hundreds of publishers on a single user-friendly platform.

Directly Into Python

https://www.quandl.com/tools/python
Project description

**PyDatastream** is a Python interface to the Thomson Dataworks Enterprise (DWE) SOAP API (non free), with some convenience functions for retrieving Datastream data specifically. This package requires valid credentials for this API.

For the documentation please refer to README.md inside the package or on the GitHub (https://github.com/vfillimonov/pydatastream/blob/master/README.md).

https://pypi.org/project/PyDatastream/
pandaSDMX: Statistical Data and Metadata eXchange in Python

pandaSDMX is an Apache 2.0-licensed Python client to retrieve and acquire statistical data and metadata disseminated in SDMX 2.1, an ISO-standard widely used by institutions such as statistics offices, central banks, and international organisations. pandaSDMX exposes datasets and related structural metadata including dataflows, codelists, and datastructure definitions as pandas Series or multi-indexed DataFrames. Many other output formats and storage backends are available thanks to Odo.

Supported data providers

pandaSDMX ships with built-in support for the following agencies (others may be configured by the user):

- Australian Bureau of Statistics (ABS)
- European Central Bank (ECB)
- Eurostat
- French National Institute for Statistics (INSEE)
- Instituto Nacional de la Estadística y Geografía - INEGI (Mexico)
- International Monetary Fund (IMF) - SDMX Central only
- International Labour Organization (ILO)
- Italian statistics Office (ISTAT)
- Norges Bank (Norway)
- Organisation for Economic Cooperation and Development (OECD)
- United Nations Statistics Division (UNSD)
- UNESCO (free registration required)
- World Bank - World Integrated Trade Solution (WITS)
FRED® API

General Documentation | API | Toolkits

The FRED® API is a web service that allows developers to write programs and build applications that retrieve economic data from the FRED® and ALFRED® websites hosted by the Economic Research Division of the Federal Reserve Bank of St. Louis. Requests can be customized according to data source, release, category, series, and other preferences.

General Documentation
- Overview
- What is FRED®?
- What is ALFRED®?
- FRED® versus ALFRED®
- Real-Time Periods
- Errors

API

Categories
- fred/category – Get a category.
- fred/category/children – Get the child categories for a specified parent category.
- fred/category/related – Get the related categories for a category.
- fred/category/series – Get the series in a category.
- fred/category/tags – Get the tags for a category.
- fred/category/related_tags – Get the related tags for a category.

https://research.stlouisfed.org/docs/api/fred/
pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.

pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

The two primary data structures of pandas, `Series` (1-dimensional) and `DataFrame` (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, `DataFrame` provides everything that R's `data.frame` provides and much more. pandas is built on top of NumPy and is

http://pandas.pydata.org/pandas-docs/stable/
pandas:
powerful Python data analysis toolkit

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

Source: http://pandas.pydata.org/pandas-docs/stable/
Series
DataFrame

• Primary data structures of pandas
  – Series (1-dimensional)
  – DataFrame (2-dimensional)

• Handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.

Source: http://pandas.pydata.org/pandas-docs/stable/
pandas DataFrame

• **DataFrame** provides everything that R’s `data.frame` provides and much more.

• pandas is built on top of **NumPy** and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.
# Comparison with SAS

<table>
<thead>
<tr>
<th>pandas</th>
<th>SAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataFrame</td>
<td>data set</td>
</tr>
<tr>
<td>column</td>
<td>variable</td>
</tr>
<tr>
<td>row</td>
<td>observation</td>
</tr>
<tr>
<td>groupby</td>
<td>BY-group</td>
</tr>
<tr>
<td>NaN</td>
<td>.</td>
</tr>
</tbody>
</table>

Creating pd.DataFrame

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
</tr>
</tbody>
</table>

```python
import pandas as pd
df = pd.DataFrame(
    {
        "a": [4, 5, 6],
        "b": [7, 8, 9],
        "c": [10, 11, 12],
    },
    index=[1, 2, 3]
)
```

Pandas DataFrame

type(df)

type(df)
pandas.core.frame.DataFrame
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
print('pandas imported')

s = pd.Series([1, 3, 5, np.nan, 6, 8])
s

dates = pd.date_range('20181001', periods=6)
dates

Source: http://pandas.pydata.org/pandas-docs/stable/10min.html
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

print('pandas imported')

s = pd.Series([1, 3, 5, np.nan, 6, 8])
s

0    1.0
1    3.0
2    5.0
3  NaN
4    6.0
5    8.0
dtype: float64

dates = pd.date_range('2018-10-01', periods=6)
dates
```

DatetimeIndex([ '2018-10-01' , '2018-10-02' , '2018-10-03' , '2018-10-04',
   '2018-10-05' , '2018-10-06' ],
   dtype='datetime64[ns]' , freq='D')
```python
df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))
df
```

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-10-01</td>
<td>-0.336188</td>
<td>0.584621</td>
<td>-1.061433</td>
<td>-0.036278</td>
</tr>
<tr>
<td>2018-10-02</td>
<td>0.903683</td>
<td>-0.839723</td>
<td>-0.270219</td>
<td>-1.099606</td>
</tr>
<tr>
<td>2018-10-03</td>
<td>0.920208</td>
<td>-0.240353</td>
<td>-0.818598</td>
<td>-1.105489</td>
</tr>
<tr>
<td>2018-10-04</td>
<td>0.221045</td>
<td>-0.314589</td>
<td>0.042071</td>
<td>-1.447280</td>
</tr>
<tr>
<td>2018-10-05</td>
<td>0.946862</td>
<td>-1.570305</td>
<td>-1.009180</td>
<td>-0.375659</td>
</tr>
<tr>
<td>2018-10-06</td>
<td>-0.225148</td>
<td>0.510691</td>
<td>2.002372</td>
<td>-0.335005</td>
</tr>
</tbody>
</table>
df = pd.DataFrame(np.random.randn(3,5),
    index=['student1','student2','student3'],
    columns=list('ABCDE'))

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>student1</td>
<td>-0.346884</td>
<td>-1.232934</td>
<td>-0.302072</td>
<td>-1.345084</td>
<td>-0.723880</td>
</tr>
<tr>
<td>student2</td>
<td>1.090955</td>
<td>-0.010483</td>
<td>1.280072</td>
<td>-0.253958</td>
<td>-0.030604</td>
</tr>
<tr>
<td>student3</td>
<td>0.325660</td>
<td>0.808956</td>
<td>-0.395820</td>
<td>-1.498926</td>
<td>1.603471</td>
</tr>
</tbody>
</table>
df2 = pd.DataFrame({'A': 1.,
'B': pd.Timestamp('20181001'),
'C': pd.Series(2.5, index=list(range(4)), dtype='float32'),
'D': np.array([3] * 4, dtype='int32'),
'E': pd.Categorical(['test', 'train', 'test', 'train']),
'F': 'foo'})

df2

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0</td>
<td>2018-10-01</td>
<td>2.5</td>
<td>3</td>
<td>test</td>
<td>foo</td>
</tr>
<tr>
<td>1</td>
<td>1.0</td>
<td>2018-10-01</td>
<td>2.5</td>
<td>3</td>
<td>train</td>
<td>foo</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>2018-10-01</td>
<td>2.5</td>
<td>3</td>
<td>test</td>
<td>foo</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>2018-10-01</td>
<td>2.5</td>
<td>3</td>
<td>train</td>
<td>foo</td>
</tr>
</tbody>
</table>
df2.dtypes

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>float64</td>
</tr>
<tr>
<td>B</td>
<td>datetime64[ns]</td>
</tr>
<tr>
<td>C</td>
<td>float32</td>
</tr>
<tr>
<td>D</td>
<td>int32</td>
</tr>
<tr>
<td>E</td>
<td>category</td>
</tr>
<tr>
<td>F</td>
<td>object</td>
</tr>
</tbody>
</table>

dtype: object
Python Pandas for Finance

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/
Yves Hilpisch (2018), Python for Finance: Mastering Data-Driven Finance, O'Reilly

https://github.com/yhilpisch/py4fi2nd

Source: https://www.amazon.com/Python-Finance-Mastering-Data-Driven/dp/1492024333
! pip install pandas_datareader
conda install pandas-datareader

[iMyday-MacBook-Pro:~ imyday$ conda install pandas-datareader
Fetching package metadata ...........
Solving package specifications: .

Package plan for installation in environment /Users/imyday/anaconda:

The following NEW packages will be INSTALLED:

    pandas-datareader: 0.2.1-py36_0
    requests-file: 1.4.1-py36_0

Proceed ([y]/n)? y

requests-file- 100% |##################################| Time: 0:00:00 1.55 MB/s
pandas-datareader 100% |##################################| Time: 0:00:00 409.66 kB/s
[iMyday-MacBook-Pro:~ imyday$ conda list
# packages in environment at /Users/imyday/anaconda:
#
_license  1.1   py36_1
alabaster  0.7.9 py36_0
anaconda  4.3.1  np111py36_0
anaconda-client  1.6.0 py36_0
anaconda-navigator  1.5.0 py36_0
anaconda-project  0.4.1 py36_0
# !pip install pandas_datareader
import pandas_datareader.data as web
import datetime as dt
#Read Stock Data from Yahoo Finance
end = dt.datetime(2017, 12, 31)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
# !pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline

#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0), rowspan=10, colspan=9)
bottom = plt.subplot2grid((12,9), (10,0), rowspan=2, colspan=9)
top.plot(df.index, df['Adj Close'], color='blue')  # df.index gives the dates
bottom.bar(df.index, df['Volume'])
# set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')

plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')
# simple moving averages

df['MA05'] = df['Adj Close'].rolling(5).mean()  
#5 days

df['MA20'] = df['Adj Close'].rolling(20).mean()  
#20 days

df['MA60'] = df['Adj Close'].rolling(60).mean()  
#60 days

df2 = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA05': df['MA05'], 'MA20': df['MA20'], 'MA60': df['MA60']})

df2.plot(figsize=(12, 9), legend=True, title='AAPL')
df2.to_csv('AAPL_MA.csv')

fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plf.show()
# !pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline

# Read Stock Data from Yahoo Finance
end = dt.datetime.now()
# start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2016, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0), colspan=9)
bottom = plt.subplot2grid((12,9), (10,0), colspan=9)
top.plot(df.index, df['Adj Close'], color='blue')  # df.index gives the dates
bottom.bar(df.index, df['Volume'])

# set the labels
top.axes.get_xaxis().set_visible(False)
top.set_title('AAPL')
top.set_ylabel('Adj Close')
bottom.set_ylabel('Volume')

plt.figure(figsize=(12,9))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')

# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean()  # 5 days
df['MA20'] = df['Adj Close'].rolling(20).mean()  # 20 days
df['MA60'] = df['Adj Close'].rolling(60).mean()  # 60 days
df2 = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA05': df['MA05'], 'MA20': df['MA20'], 'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True, title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()
```python
# pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt

# Read Stock Data from Yahoo Finance
end = dt.datetime.now()
start = dt.datetime(end.year-2, end.month, end.day)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()

df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
plt.figure(figsize=(12,9))
top = plt.subplot2grid((12,9), (0, 0), colspan=9, rowspan=10)
bottom = plt.subplot2grid((12,9), (10,0), colspan=9, rowspan=2)
top.plot(df.index, df['Adj Close'], color='blue')  # df.index gives the dates
bottom.bar(df.index, df['Volume'])

# set the labels
top.set_title('AAPL')
top.set_xlabel('Adj Close')
top.set_ylabel('Volume')

sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')

# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean()  # 5 days
df['MA20'] = df['Adj Close'].rolling(20).mean()  # 20 days
df['MA60'] = df['Adj Close'].rolling(60).mean()  # 60 days
df2 = pd.DataFrame({'Adj Close': df['Adj Close'], 'MA05': df['MA05'], 'MA20': df['MA20'], 'MA60': df['MA60']})
df2.plot(figsize=(12, 9), legend=True, title='AAPL')
df2.to_csv('AAPL_MA.csv')
fig = plt.gcf()
fig.set_size_inches(12, 9)
fig.savefig('AAPL_plot.png', dpi=300)
plt.show()
```
Finance Data from Quandl

```python
# ! pip install quandl
import quandl

# quandl.ApiConfig.api_key = "YOURAPIKEY"

df = quandl.get("WIKI/AAPL", start_date="2016-01-01", end_date="2017-12-31")

df.to_csv('AAPL.csv')

df.from_csv('AAPL.csv')

df.tail()
```

### Output Table

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-12-22</td>
<td>174.68</td>
<td>175.424</td>
<td>174.500</td>
<td>175.01</td>
<td>16052615.0</td>
<td>0.0</td>
<td>1.0</td>
<td>174.68</td>
<td>175.424</td>
<td>174.500</td>
<td>175.01</td>
<td>16052615.0</td>
</tr>
<tr>
<td>2017-12-26</td>
<td>170.80</td>
<td>171.470</td>
<td>169.679</td>
<td>170.57</td>
<td>32968167.0</td>
<td>0.0</td>
<td>1.0</td>
<td>170.80</td>
<td>171.470</td>
<td>169.679</td>
<td>170.57</td>
<td>32968167.0</td>
</tr>
<tr>
<td>2017-12-27</td>
<td>170.10</td>
<td>170.780</td>
<td>169.710</td>
<td>170.60</td>
<td>21672062.0</td>
<td>0.0</td>
<td>1.0</td>
<td>170.10</td>
<td>170.780</td>
<td>169.710</td>
<td>170.60</td>
<td>21672062.0</td>
</tr>
<tr>
<td>2017-12-28</td>
<td>171.00</td>
<td>171.850</td>
<td>170.480</td>
<td>171.08</td>
<td>15997739.0</td>
<td>0.0</td>
<td>1.0</td>
<td>171.00</td>
<td>171.850</td>
<td>170.480</td>
<td>171.08</td>
<td>15997739.0</td>
</tr>
<tr>
<td>2017-12-29</td>
<td>170.52</td>
<td>170.590</td>
<td>169.220</td>
<td>169.23</td>
<td>25643711.0</td>
<td>0.0</td>
<td>1.0</td>
<td>170.52</td>
<td>170.590</td>
<td>169.220</td>
<td>169.23</td>
<td>25643711.0</td>
</tr>
</tbody>
</table>

Source: [https://www.quandl.com/tools/python](https://www.quandl.com/tools/python)
Yahoo Finance Symbols: AAPL Apple Inc. (AAPL)

S&P 500
2,344.02
-29.45 (-1.24%)

Dow 30
20,668.01
-237.85 (-1.14%)

Nasdaq
5,793.83
-107.70 (-1.83%)

Crude Oil
47.50
+0.16 (+0.34%)

Gold
1,245.40
-1.10 (-0.09%)

Quote Lookup

Symbols similar to 'aapl'

All (9)  Stocks (6)  Mutual Funds (0)  ETFs (1)  Indices (2)  Futures (0)  Currencies (0)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Company Name</th>
<th>Last Price</th>
<th>Industry / Category</th>
<th>Type</th>
<th>Exchange</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAPL</td>
<td>Apple Inc.</td>
<td>139.84</td>
<td>Electronic Equipment</td>
<td>Stocks</td>
<td>NMS</td>
</tr>
<tr>
<td>AAPL.SW</td>
<td>Apple Inc.</td>
<td>140.70</td>
<td>N/A</td>
<td>Stocks</td>
<td>EBS</td>
</tr>
<tr>
<td>AAPL.MX</td>
<td>Apple Inc.</td>
<td>2678.68</td>
<td>Electronic Equipment</td>
<td>Stocks</td>
<td>MEX</td>
</tr>
<tr>
<td>AAPL34F.SA</td>
<td>Apple Inc.</td>
<td>0.00</td>
<td>N/A</td>
<td>Stocks</td>
<td>SAO</td>
</tr>
<tr>
<td>AAPL34.SA</td>
<td>Apple Inc.</td>
<td>43.14</td>
<td>Electronic Equipment</td>
<td>Stocks</td>
<td>SAO</td>
</tr>
</tbody>
</table>

http://finance.yahoo.com/q?s=AAPL
Apple Inc. (AAPL) - NasdaqGS

139.84  -1.62 (-1.15%)  139.35  -0.49 (-0.35%)
At close: 4:00PM EDT  After hours: 7:59PM EDT

Summary

Previous Close  141.46
Open  142.11
Bid  139.31 x 100
Ask  139.40 x 1300
Day's Range  139.73 - 142.80
52 Week Range  89.47 - 142.80
Volume  39,529,912
Avg. Volume  26,889,183
Market Cap  733.68B
Beta  1.45
PE Ratio (TTM)  16.79
EPS (TTM)  8.33
Earnings Date  Apr 24, 2017 - Apr 28, 2017
Dividend & Yield  2.28 (1.63%)
Ex-Dividend Date  N/A
1y Target Est  143.29

Trade prices are not sourced from all markets

Yahoo Finance Charts: Apple Inc. (AAPL)

S&P 500 2,344.02 -29.45 (-1.24%)
Dow 30 20,668.01 -237.85 (-1.14%)
Nasdaq 5,793.83 -107.70 (-1.83%)
Crude Oil 47.50 +0.16 (+0.34%)
Gold 1,245.50 -1.00 (-0.08%)

Apple Inc. (AAPL) 139.84 -1.62 (-1.15%) As of 4:00PM EDT. Market closed.

Open 142.11
Close 139.84
Low 139.73
High 142.80
Vol 39.53M
% Chg 63.27%

http://finance.yahoo.com/chart/AAPL
Apple Inc. (AAPL) Historical Data

[Yahoo Finance page with historical data for Apple Inc. (AAPL) displayed]

Thank you for helping us improve your Yahoo experience

Learn more about your feedback.

Time Period: Mar 22, 2016 - Mar 22, 2017
Show: Historical Prices
Frequency: Daily

Currency in USD

<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Adj Close*</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mar 21, 2017</td>
<td>142.11</td>
<td>142.80</td>
<td>139.73</td>
<td>139.84</td>
<td>139.84</td>
<td>39,116,800</td>
</tr>
</tbody>
</table>

http://finance.yahoo.com/q/hp?s=AAPL+Historical+Prices
Yahoo Finance Historical Prices
Apple Inc. (AAPL)

<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Adj Close*</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mar 21, 2017</td>
<td>142.11</td>
<td>142.80</td>
<td>139.73</td>
<td>139.84</td>
<td>139.84</td>
<td>39,116,800</td>
</tr>
<tr>
<td>Mar 20, 2017</td>
<td>140.40</td>
<td>141.50</td>
<td>140.23</td>
<td>141.46</td>
<td>141.46</td>
<td>21,542,000</td>
</tr>
<tr>
<td>Mar 17, 2017</td>
<td>141.00</td>
<td>141.00</td>
<td>139.89</td>
<td>139.99</td>
<td>139.99</td>
<td>43,885,000</td>
</tr>
<tr>
<td>Mar 16, 2017</td>
<td>140.72</td>
<td>141.02</td>
<td>140.26</td>
<td>140.69</td>
<td>140.69</td>
<td>19,232,000</td>
</tr>
<tr>
<td>Mar 15, 2017</td>
<td>139.41</td>
<td>140.75</td>
<td>139.03</td>
<td>140.46</td>
<td>140.46</td>
<td>25,691,800</td>
</tr>
<tr>
<td>Mar 14, 2017</td>
<td>139.30</td>
<td>139.65</td>
<td>138.84</td>
<td>138.99</td>
<td>138.99</td>
<td>15,309,100</td>
</tr>
<tr>
<td>Mar 13, 2017</td>
<td>138.85</td>
<td>139.43</td>
<td>138.82</td>
<td>139.20</td>
<td>139.20</td>
<td>17,421,700</td>
</tr>
<tr>
<td>Mar 10, 2017</td>
<td>139.25</td>
<td>139.36</td>
<td>138.64</td>
<td>139.14</td>
<td>139.14</td>
<td>19,612,800</td>
</tr>
<tr>
<td>Mar 09, 2017</td>
<td>138.74</td>
<td>138.79</td>
<td>137.05</td>
<td>138.68</td>
<td>138.68</td>
<td>22,155,900</td>
</tr>
<tr>
<td>Mar 08, 2017</td>
<td>138.95</td>
<td>139.80</td>
<td>138.82</td>
<td>139.00</td>
<td>139.00</td>
<td>18,707,200</td>
</tr>
<tr>
<td>Mar 07, 2017</td>
<td>139.06</td>
<td>139.98</td>
<td>138.79</td>
<td>139.52</td>
<td>139.52</td>
<td>17,446,300</td>
</tr>
<tr>
<td>Mar 06, 2017</td>
<td>139.37</td>
<td>139.77</td>
<td>138.60</td>
<td>139.34</td>
<td>139.34</td>
<td>21,750,000</td>
</tr>
<tr>
<td>Mar 03, 2017</td>
<td>138.78</td>
<td>139.83</td>
<td>138.59</td>
<td>139.78</td>
<td>139.78</td>
<td>21,108,100</td>
</tr>
</tbody>
</table>

http://finance.yahoo.com/quote/AAPL/history
### Yahoo Finance Historical Prices

#### Apple Inc. (AAPL)

**Time Period:** Dec 12, 1980 - Mar 22, 2017
**Show:** Historical Prices
**Frequency:** Daily

<table>
<thead>
<tr>
<th>Date</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Adj Close</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mar 21, 2017</td>
<td>142.17</td>
<td>130.68</td>
<td>139.73</td>
<td>139.84</td>
<td>39,116,800</td>
</tr>
<tr>
<td>Mar 20, 2017</td>
<td>141.50</td>
<td>140.23</td>
<td>141.46</td>
<td>141.46</td>
<td>21,542,000</td>
</tr>
<tr>
<td>Mar 17, 2017</td>
<td>139.99</td>
<td>139.89</td>
<td>139.99</td>
<td>139.99</td>
<td>43,885,000</td>
</tr>
<tr>
<td>Mar 16, 2017</td>
<td>140.72</td>
<td>141.02</td>
<td>140.26</td>
<td>140.69</td>
<td>19,232,000</td>
</tr>
<tr>
<td>Mar 15, 2017</td>
<td>139.41</td>
<td>140.75</td>
<td>139.03</td>
<td>140.46</td>
<td>25,691,800</td>
</tr>
<tr>
<td>Mar 14, 2017</td>
<td>139.30</td>
<td>139.65</td>
<td>138.84</td>
<td>138.99</td>
<td>15,309,100</td>
</tr>
<tr>
<td>Mar 13, 2017</td>
<td>138.85</td>
<td>139.43</td>
<td>138.82</td>
<td>139.20</td>
<td>17,421,700</td>
</tr>
<tr>
<td>Mar 10, 2017</td>
<td>139.25</td>
<td>139.36</td>
<td>138.64</td>
<td>139.14</td>
<td>19,612,800</td>
</tr>
<tr>
<td>Mar 09, 2017</td>
<td>138.74</td>
<td>138.79</td>
<td>137.05</td>
<td>138.68</td>
<td>22,155,900</td>
</tr>
<tr>
<td>Mar 08, 2017</td>
<td>138.95</td>
<td>139.80</td>
<td>138.82</td>
<td>139.00</td>
<td>18,707,200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Adj Close*</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mar 21, 2017</td>
<td>142.11</td>
<td>142.80</td>
<td>139.73</td>
<td>139.84</td>
<td>139.84</td>
<td>39,116,800</td>
</tr>
<tr>
<td>Mar 20, 2017</td>
<td>140.40</td>
<td>141.50</td>
<td>140.23</td>
<td>141.46</td>
<td>141.46</td>
<td>21,542,000</td>
</tr>
<tr>
<td>Mar 17, 2017</td>
<td>141.00</td>
<td>141.00</td>
<td>139.89</td>
<td>139.99</td>
<td>139.99</td>
<td>43,885,000</td>
</tr>
<tr>
<td>Mar 16, 2017</td>
<td>140.72</td>
<td>141.02</td>
<td>140.26</td>
<td>140.69</td>
<td>140.69</td>
<td>19,232,000</td>
</tr>
<tr>
<td>Mar 15, 2017</td>
<td>139.41</td>
<td>140.75</td>
<td>139.03</td>
<td>140.46</td>
<td>140.46</td>
<td>25,691,800</td>
</tr>
<tr>
<td>Mar 14, 2017</td>
<td>139.30</td>
<td>139.65</td>
<td>138.84</td>
<td>138.99</td>
<td>138.99</td>
<td>15,309,100</td>
</tr>
<tr>
<td>Mar 13, 2017</td>
<td>138.85</td>
<td>139.43</td>
<td>138.82</td>
<td>139.20</td>
<td>139.20</td>
<td>17,421,700</td>
</tr>
<tr>
<td>Mar 10, 2017</td>
<td>139.25</td>
<td>139.36</td>
<td>138.64</td>
<td>139.14</td>
<td>139.14</td>
<td>19,612,800</td>
</tr>
<tr>
<td>Mar 09, 2017</td>
<td>138.74</td>
<td>138.79</td>
<td>137.05</td>
<td>138.68</td>
<td>138.68</td>
<td>22,155,900</td>
</tr>
<tr>
<td>Mar 08, 2017</td>
<td>138.95</td>
<td>139.80</td>
<td>138.82</td>
<td>139.00</td>
<td>139.00</td>
<td>18,707,200</td>
</tr>
</tbody>
</table>

### Yahoo Finance Historical Prices

http://ichart.finance.yahoo.com/table.csv?s=AAPL

<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Volume</th>
<th>Adj Close</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-03-21</td>
<td>142.11</td>
<td>142.80</td>
<td>139.72</td>
<td>139.83</td>
<td>39116800</td>
<td>139.84</td>
</tr>
<tr>
<td>2017-03-20</td>
<td>140.39</td>
<td>141.50</td>
<td>140.23</td>
<td>141.46</td>
<td>20213100</td>
<td>141.46</td>
</tr>
<tr>
<td>2017-03-17</td>
<td>141.00</td>
<td>141.00</td>
<td>139.89</td>
<td>139.99</td>
<td>43597400</td>
<td>139.99</td>
</tr>
<tr>
<td>2017-03-16</td>
<td>140.72</td>
<td>141.02</td>
<td>140.26</td>
<td>140.69</td>
<td>19132500</td>
<td>140.69</td>
</tr>
<tr>
<td>2017-03-15</td>
<td>139.41</td>
<td>140.75</td>
<td>139.03</td>
<td>140.46</td>
<td>25566800</td>
<td>140.46</td>
</tr>
<tr>
<td>2017-03-14</td>
<td>139.30</td>
<td>139.65</td>
<td>138.84</td>
<td>138.90</td>
<td>15187000</td>
<td>138.90</td>
</tr>
<tr>
<td>2017-03-13</td>
<td>138.85</td>
<td>139.43</td>
<td>138.82</td>
<td>139.19</td>
<td>17042400</td>
<td>139.19</td>
</tr>
<tr>
<td>2017-03-10</td>
<td>139.25</td>
<td>139.36</td>
<td>138.64</td>
<td>139.14</td>
<td>19488000</td>
<td>139.14</td>
</tr>
<tr>
<td>2017-03-09</td>
<td>138.74</td>
<td>138.79</td>
<td>138.68</td>
<td>138.76</td>
<td>22065200</td>
<td>138.76</td>
</tr>
<tr>
<td>2017-03-08</td>
<td>138.95</td>
<td>139.80</td>
<td>138.82</td>
<td>139.00</td>
<td>18681800</td>
<td>139.00</td>
</tr>
<tr>
<td>2017-03-07</td>
<td>139.06</td>
<td>138.79</td>
<td>138.52</td>
<td>138.52</td>
<td>17267500</td>
<td>138.52</td>
</tr>
<tr>
<td>2017-03-06</td>
<td>138.39</td>
<td>138.77</td>
<td>138.60</td>
<td>138.34</td>
<td>21153300</td>
<td>138.34</td>
</tr>
<tr>
<td>2017-03-03</td>
<td>138.78</td>
<td>139.83</td>
<td>138.58</td>
<td>139.78</td>
<td>21108100</td>
<td>139.78</td>
</tr>
<tr>
<td>2017-03-02</td>
<td>140.00</td>
<td>140.28</td>
<td>138.76</td>
<td>139.96</td>
<td>26153300</td>
<td>139.96</td>
</tr>
<tr>
<td>2017-03-01</td>
<td>137.89</td>
<td>140.14</td>
<td>137.60</td>
<td>139.78</td>
<td>36272400</td>
<td>139.78</td>
</tr>
<tr>
<td>2017-02-28</td>
<td>137.08</td>
<td>137.44</td>
<td>136.69</td>
<td>136.99</td>
<td>23403500</td>
<td>136.99</td>
</tr>
<tr>
<td>2017-02-27</td>
<td>137.14</td>
<td>137.44</td>
<td>136.99</td>
<td>136.92</td>
<td>20164000</td>
<td>136.92</td>
</tr>
<tr>
<td>2017-02-24</td>
<td>135.91</td>
<td>136.66</td>
<td>135.28</td>
<td>136.66</td>
<td>21690900</td>
<td>136.66</td>
</tr>
<tr>
<td>2017-02-23</td>
<td>137.38</td>
<td>137.48</td>
<td>136.30</td>
<td>136.53</td>
<td>20704100</td>
<td>136.53</td>
</tr>
<tr>
<td>2017-02-22</td>
<td>136.43</td>
<td>137.12</td>
<td>136.11</td>
<td>137.11</td>
<td>20745300</td>
<td>137.11</td>
</tr>
</tbody>
</table>
Dow Jones Industrial Average (^DJI)

Dow Jones Industrial Average (^DJI) 20,668.01  -237.85 (-1.14%)  As of 4:36PM EDT. Market closed.

Open  20956.33
Close  20668.01
Low   20653.26
High  20970.04
Vol   369.32M
% Chg 56.43%

http://finance.yahoo.com/chart/^DJI
TSEC weighted index (^TWII) - Taiwan

http://finance.yahoo.com/chart/^TWII
Taiwan Semiconductor Manufacturing Company Limited (2330.TW)

192.50 -2.50 (-1.28%)
As of 9:52 AM CST. Market open.

Summary

| Previous Close | 195.00 | Market Cap | 4.98T |
| Open           | 192.50 | Beta       | N/A   |
| Bid            | 192.00 | PE Ratio (TTM) | 14.90 |
| Ask            | 192.50 | EPS (TTM)  | 12.89 |
| Day’s Range    | 191.50 - 193.00 | Earnings Date | Apr 13, 2017 |
| 52 Week Range  | 154.00 - 193.00 | Dividend & Yield | N/A (N/A) |
| Volume         | 6,977,000 | Ex-Dividend Date | N/A |

http://finance.yahoo.com/q?s=2330.TW
Yahoo Finance Charts
TSMC (2330.TW)

Taiwan Semiconductor Manufacturing Company Limited (2330.TW) 192.00 -3.00 (-1.54%)
As of 10:29AM CST. Taiwan Delayed Price. Market open.

Open 192.50
Close 192.00
Low 191.50
High 193.00
Vol 9.33M
% Chg 301.44%

http://finance.yahoo.com/chart/2330.TW
Yahoo Finance Charts
US Dollar/USDX - Index - Cash (DX-Y.NYB)

97.35  +0.11 (+0.12%)
As of 9:52AM EST. Market open.

Yahoo Finance Charts
USD/TWD (USDTWD=X)

https://finance.yahoo.com/quote/USDTWD%3DX/chart?p=USDTWD%3DX
US Dollar/USDX - Index - Cash (DX-Y.NYB)

```python
# !pip install pandas_datareader
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
%matplotlib inline

# Read Stock Data from Yahoo Finance
end = dt.datetime.now()
# start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2017, 1, 1)
df = web.DataReader("DX-Y.NYB", 'yahoo', start, end)
df.to_csv('DX-Y.NYB.csv')
print(df.tail())
df2 = pd.read_csv('DX-Y.NYB.csv')
print(df2.tail())
df["Adj Close"].plot(legend=True, figsize=(12, 8), title='DX-Y.NYB', label='Adj Close')
```
US Dollar/USDX - Index - Cash (DX-Y.NYB)
import pandas as pd
import pandas_datareader.data as web

df = web.DataReader('AAPL', data_source='yahoo', start='1/1/2010', end='3/21/2017')
df.to_csv('AAPL.csv')
df.tail()
```python
df = web.DataReader('GOOG', data_source='yahoo', start='1/1/1980', end='3/21/2017')
df.head(10)
```

<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Volume</th>
<th>Adj Close</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004-08-19</td>
<td>100.000168</td>
<td>104.060182</td>
<td>95.960165</td>
<td>100.340176</td>
<td>44871300</td>
<td>50.119968</td>
</tr>
<tr>
<td>2004-08-20</td>
<td>101.010175</td>
<td>109.080187</td>
<td>100.500174</td>
<td>108.310183</td>
<td>22942800</td>
<td>54.100990</td>
</tr>
<tr>
<td>2004-08-23</td>
<td>110.750191</td>
<td>113.480193</td>
<td>109.050183</td>
<td>109.400185</td>
<td>18342800</td>
<td>54.645447</td>
</tr>
<tr>
<td>2004-08-24</td>
<td>111.240189</td>
<td>111.600192</td>
<td>103.570177</td>
<td>104.870176</td>
<td>15319700</td>
<td>52.382705</td>
</tr>
<tr>
<td>2004-08-25</td>
<td>104.960181</td>
<td>108.000187</td>
<td>103.880180</td>
<td>106.000184</td>
<td>9232100</td>
<td>52.947145</td>
</tr>
<tr>
<td>2004-08-26</td>
<td>104.950180</td>
<td>107.950188</td>
<td>104.660179</td>
<td>107.910182</td>
<td>7128600</td>
<td>53.901190</td>
</tr>
<tr>
<td>2004-08-27</td>
<td>108.100185</td>
<td>108.620186</td>
<td>105.690180</td>
<td>106.150181</td>
<td>6241200</td>
<td>53.022069</td>
</tr>
<tr>
<td>2004-08-30</td>
<td>105.280178</td>
<td>105.490184</td>
<td>102.010172</td>
<td>102.010172</td>
<td>5221400</td>
<td>50.954132</td>
</tr>
<tr>
<td>2004-08-31</td>
<td>102.300173</td>
<td>103.710180</td>
<td>102.160177</td>
<td>102.370175</td>
<td>4941200</td>
<td>51.133953</td>
</tr>
<tr>
<td>2004-09-01</td>
<td>102.700174</td>
<td>102.970180</td>
<td>99.670169</td>
<td>100.250171</td>
<td>9181600</td>
<td>50.075011</td>
</tr>
</tbody>
</table>
### df.tail(10)

<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Volume</th>
<th>Adj Close</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-03-08</td>
<td>833.510010</td>
<td>838.150024</td>
<td>831.789978</td>
<td>835.369995</td>
<td>988700</td>
<td>835.369995</td>
</tr>
<tr>
<td>2017-03-09</td>
<td>836.000000</td>
<td>842.000000</td>
<td>834.210022</td>
<td>838.679993</td>
<td>1259900</td>
<td>838.679993</td>
</tr>
<tr>
<td>2017-03-10</td>
<td>843.280029</td>
<td>844.909973</td>
<td>839.500000</td>
<td>843.250000</td>
<td>1701100</td>
<td>843.250000</td>
</tr>
<tr>
<td>2017-03-13</td>
<td>844.000000</td>
<td>848.684998</td>
<td>843.250000</td>
<td>845.539978</td>
<td>1149500</td>
<td>845.539978</td>
</tr>
<tr>
<td>2017-03-14</td>
<td>843.640015</td>
<td>847.239990</td>
<td>840.799988</td>
<td>845.619995</td>
<td>779900</td>
<td>845.619995</td>
</tr>
<tr>
<td>2017-03-15</td>
<td>847.590027</td>
<td>848.630005</td>
<td>840.770020</td>
<td>847.200012</td>
<td>1379600</td>
<td>847.200012</td>
</tr>
<tr>
<td>2017-03-16</td>
<td>849.030029</td>
<td>850.849976</td>
<td>846.130005</td>
<td>848.780029</td>
<td>970400</td>
<td>848.780029</td>
</tr>
<tr>
<td>2017-03-17</td>
<td>851.609985</td>
<td>853.400024</td>
<td>847.109985</td>
<td>852.119995</td>
<td>1712300</td>
<td>852.119995</td>
</tr>
<tr>
<td>2017-03-20</td>
<td>850.010010</td>
<td>850.219971</td>
<td>845.150024</td>
<td>848.400024</td>
<td>1190300</td>
<td>848.400024</td>
</tr>
<tr>
<td>2017-03-21</td>
<td>851.400024</td>
<td>853.500000</td>
<td>829.020020</td>
<td>830.460022</td>
<td>2442900</td>
<td>830.460022</td>
</tr>
</tbody>
</table>
```python
df.count()
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>3169</td>
</tr>
<tr>
<td>High</td>
<td>3169</td>
</tr>
<tr>
<td>Low</td>
<td>3169</td>
</tr>
<tr>
<td>Close</td>
<td>3169</td>
</tr>
<tr>
<td>Volume</td>
<td>3169</td>
</tr>
<tr>
<td>Adj Close</td>
<td>3169</td>
</tr>
<tr>
<td>dtype:</td>
<td>int64</td>
</tr>
</tbody>
</table>
df.ix['2015-12-31']

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>7.695000e+02</td>
</tr>
<tr>
<td>High</td>
<td>7.695000e+02</td>
</tr>
<tr>
<td>Low</td>
<td>7.583400e+02</td>
</tr>
<tr>
<td>Close</td>
<td>7.588800e+02</td>
</tr>
<tr>
<td>Volume</td>
<td>1.489600e+06</td>
</tr>
<tr>
<td>Adj Close</td>
<td>7.588800e+02</td>
</tr>
</tbody>
</table>

Name: 2015-12-31 00:00:00, dtype: float64
df.to_csv('2330.TW.Yahoo.Finance.Data.csv')

<table>
<thead>
<tr>
<th></th>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Volume</th>
<th>Adj Close</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2010-01-01</td>
<td>64.5</td>
<td>64.5</td>
<td>64.5</td>
<td>64.5</td>
<td>0</td>
<td>52.8308</td>
</tr>
<tr>
<td>2</td>
<td>2010-01-04</td>
<td>65.0</td>
<td>65.0</td>
<td>64.0</td>
<td>64.9</td>
<td>39407000</td>
<td>53.1584</td>
</tr>
<tr>
<td>3</td>
<td>2010-01-05</td>
<td>65.0</td>
<td>65.1</td>
<td>63.9</td>
<td>64.5</td>
<td>37138000</td>
<td>52.8308</td>
</tr>
<tr>
<td>4</td>
<td>2010-01-06</td>
<td>64.5</td>
<td>64.9</td>
<td>63.7</td>
<td>64.9</td>
<td>49261000</td>
<td>53.1584</td>
</tr>
<tr>
<td>5</td>
<td>2010-01-07</td>
<td>64.9</td>
<td>65.0</td>
<td>64.2</td>
<td>64.2</td>
<td>42134000</td>
<td>52.5851</td>
</tr>
<tr>
<td>6</td>
<td>2010-01-08</td>
<td>63.5</td>
<td>64.3</td>
<td>63.5</td>
<td>64.0</td>
<td>46076000</td>
<td>52.4213</td>
</tr>
<tr>
<td>7</td>
<td>2010-01-11</td>
<td>64.0</td>
<td>64.9</td>
<td>63.5</td>
<td>64.5</td>
<td>36799000</td>
<td>52.8308</td>
</tr>
<tr>
<td>8</td>
<td>2010-01-12</td>
<td>64.4</td>
<td>64.4</td>
<td>63.3</td>
<td>63.6</td>
<td>49853000</td>
<td>52.0936</td>
</tr>
<tr>
<td>9</td>
<td>2010-01-13</td>
<td>63.0</td>
<td>63.1</td>
<td>62.6</td>
<td>62.8</td>
<td>47976000</td>
<td>51.4384</td>
</tr>
<tr>
<td>10</td>
<td>2010-01-14</td>
<td>63.6</td>
<td>63.6</td>
<td>63.0</td>
<td>63.2</td>
<td>36149000</td>
<td>51.766</td>
</tr>
<tr>
<td>11</td>
<td>2010-01-15</td>
<td>62.9</td>
<td>63.5</td>
<td>62.8</td>
<td>63.5</td>
<td>47852000</td>
<td>52.0117</td>
</tr>
<tr>
<td>12</td>
<td>2010-01-18</td>
<td>62.8</td>
<td>63.1</td>
<td>62.8</td>
<td>62.9</td>
<td>30136000</td>
<td>51.5203</td>
</tr>
<tr>
<td>13</td>
<td>2010-01-19</td>
<td>63.0</td>
<td>63.2</td>
<td>62.0</td>
<td>62.5</td>
<td>47202000</td>
<td>51.1926</td>
</tr>
<tr>
<td>14</td>
<td>2010-01-20</td>
<td>62.9</td>
<td>63.2</td>
<td>62.2</td>
<td>63.0</td>
<td>52281000</td>
<td>51.6022</td>
</tr>
</tbody>
</table>
```python
import fix_yahoo_finance as yf
data = yf.download("^TWII", start="2017-07-01", end="2017-11-15")
data.to_csv('TWII_201707_201711.csv')
data.tail()
```

```
import fix_yahoo_finance as yf
data = yf.download("^TWII", start="2017-07-01", end="2017-11-15")
data.to_csv('TWII_201707_201711.csv')
data.tail()
```

```
[******************************100%******************************] 1 of 1 downloaded

<table>
<thead>
<tr>
<th>Date</th>
<th>Open</th>
<th>High</th>
<th>Low</th>
<th>Close</th>
<th>Adj Close</th>
<th>Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-11-08</td>
<td>10839.440430</td>
<td>10844.740234</td>
<td>10806.009766</td>
<td>10818.990234</td>
<td>10818.990234</td>
<td>2438000</td>
</tr>
<tr>
<td>2017-11-09</td>
<td>10802.950195</td>
<td>10831.379883</td>
<td>10721.870117</td>
<td>10743.269531</td>
<td>10743.269531</td>
<td>2917800</td>
</tr>
<tr>
<td>2017-11-10</td>
<td>10713.669922</td>
<td>10742.610352</td>
<td>10659.290039</td>
<td>10732.669922</td>
<td>10732.669922</td>
<td>2277000</td>
</tr>
<tr>
<td>2017-11-13</td>
<td>10728.219727</td>
<td>10749.389648</td>
<td>10683.919922</td>
<td>10683.919922</td>
<td>10683.919922</td>
<td>2765100</td>
</tr>
<tr>
<td>2017-11-14</td>
<td>10716.589844</td>
<td>10735.080078</td>
<td>10654.580078</td>
<td>10687.179688</td>
<td>10687.179688</td>
<td>2596600</td>
</tr>
</tbody>
</table>
```
df = df.loc['2017-10-01':'2017-11-15']
import matplotlib.pyplot as plt
%matplotlib inline
import fix_yahoo_finance as yf

df = yf.download('^TWII', start='2000-01-01', end='2017-11-15')
df.to_csv('YF_TWII_2000_2017.csv')
print(df.head())

fig = plt.figure(figsize=(16,9))
df['Adj Close'].plot()
fig.show()
import matplotlib.pyplot as plt
from matplotliblib.finance import candlestick_ohlc
import matplotlib.pyplot as plt
from matplotlib.finance import candlestick_ohlc
# Convert Daily Data to Weekly Data

```python
def daily_to_weekly(df):
    # dfWeekly = daily_to_weekly(df)
    df.sort_index(axis=0, level=None, ascending=True, inplace=True)
    Open = df.Open.resample('W-Fri').first()  # W #W-MON #W-Fri
    High = df.High.resample('W-Fri').max()
    Low = df.Low.resample('W-Fri').min()
    Close = df.Close.resample('W-Fri').last()
    Volume = df.Volume.resample('W-Fri').sum()
    Adj_Close = df["Adj Close"].resample('W-Fri').last()
    dfWeekly = pd.concat([Open, High, Low, Close, Volume, Adj_Close], axis=1)
    dfWeekly = dfWeekly[pd.notnull(dfWeekly['Adj Close'])]
    return dfWeekly
```
def daily_to_monthly(df):
    dfMonthly = daily_to_monthly(df)
    Open = df.Open.resample('M').first()
    High = df.High.resample('M').max()
    Low = df.Low.resample('M').min()
    Close = df.Close.resample('M').last()
    Volume = df.Volume.resample('M').sum()
    Adj_Close = df['Adj Close'].resample('M').last()
    dfMonthly = pd.concat([Open, High, Low, Close, Volume, Adj_Close], axis=1)
    dfMonthly = dfMonthly[pd.notnull(dfMonthly['Adj Close'])]
    return dfMonthly
Multi-Platform Tools for Market Analysis ...
TA-Lib is widely used by trading software developers requiring to perform technical analysis of financial market data.

- Includes 200 indicators such as ADX, MACD, RSI, Stochastic, Bollinger Bands etc... (more info)
- Candlestick pattern recognition
- Open-source API for C/C++, Java, Perl, Python and 100% Managed .NET

Free Open-Source Library
TA-Lib is available under a BSD License allowing it to be integrated in your own open-source or commercial application. (more info)

Commercial Application
TA-Lib is also available as an easy to install Excel Add-Ins. Try it for free!
# Stochastic oscillator %D

def KDJ(df, n, m1, m2):
    KDJ_n = n
    KDJ_m1 = m1
    KDJ_m2 = m2

    df['Low_n'] = df['Low'].rolling(min_periods=n, win_size=n).min()
    df['Low_n'].fillna(value=df['Low'].expanding().min(), inplace=True)
    df['High_n'] = df['High'].rolling(min_periods=n, win_size=n).max()
    df['High_n'].fillna(value=df['High'].expanding().max(), inplace=True)

    df['RSV'] = (df['Close'] - df['Low_n']) / (df['High_n'] - df['Low_n']) * 100

    df['KDJ_K'] = df['RSV'].ewm(span=m1).mean()
    df['KDJ_D'] = df['KDJ_K'].ewm(span=m2).mean()
    df['KDJ_J'] = 3 * df['KDJ_K'] - 2 * df['KDJ_D']

    return df

Source: https://www.quantopian.com/posts/technical-analysis-indicators-without-talib-code
# Bollinger Bands

def BBANDS20(df, n):
    MA = pd.Series(pd.rolling_mean(df['Close'], n))
    MSD = pd.Series(pd.rolling_std(df['Close'], n))
    b1 = 4 * MSD / MA
    B1 = pd.Series(b1, name = 'BollingerB_' + str(n))
    df = df.join(B1)
    b2 = (df['Close'] - MA + 2 * MSD) / (4 * MSD)
    B2 = pd.Series(b2, name = 'Bollinger%b_' + str(n))
    df = df.join(B2)
    return df

Bollinger Bands

```python
#BB Bollinger Bands BB_20
def BB_20(df):
    df['BB_MA20'] = pd.stats.moments.rolling_mean(df['Adj Close'], 20)
    df['BB_SD20'] = pd.stats.moments.rolling_std(df['Adj Close'], 20)
    df['BB_UpperBand'] = df['BB_MA20'] + (df['BB_SD20'])*2)  # Default 2*SD
    df['BB_LowerBand'] = df['BB_MA20'] - (df['BB_SD20'])*2)
    df['BB_PB'] = (df['Adj Close'] - df['BB_LowerBand']) / (df['BB_UpperBand'] - df['BB_LowerBand'])
    df['BB_BW'] = (df['BB_UpperBand'] - df['BB_LowerBand']) / df['BB_MA20']
    df['BB_UpperBand_1SD'] = df['BB_MA20'] + (df['BB_SD20'])
    df['BB_LowerBand_1SD'] = df['BB_MA20'] - (df['BB_SD20'])

#BB_PB: Bollinger Band Percent b (PB)
#BB_BW: Bollinger Band Band Width (BW)
return df
```

Quantopian inspires talented people everywhere to write investment algorithms. Select authors may license their algorithms to us and get paid based on performance.

Start Coding

https://www.quantopian.com/
Summary

• Quantitative Investing with Pandas in Python
  – Numpy
  – Pandas
References

- Ties de Kok (2017), Learn Python for Research, https://github.com/TiesdeKok/LearnPythonforResearch
- Python Programming, https://pythonprogramming.net/
- Python, https://www.python.org/
- Pandas, http://pandas.pydata.org/
- Skikit-learn, http://scikit-learn.org/
- Data School (2015), Machine learning in Python with scikit-learn, https://www.youtube.com/playlist?list=PL5-da3qGB5ICeMbQuqbbCOQWcS6OYBr5A