金融科技

FinTech: Financial Technology

Tamkang University
淡江大學

投資組合最佳化與程式交易

(Portfolio Optimization and Algorithmic Trading)

1052FinTech10
MIS EMBA (M2263) (8595)
Fri, 12,13,14 (19:20-22:10) (D409)

Min-Yuh Day
戴敏育
Assistant Professor
專任助理教授
Dept. of Information Management, Tamkang University
淡江大學 資訊管理學系
http://mail. tku.edu.tw/myday/

2017-05-26

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2017/02/17	Fintech 金融科技課程介紹 (Course Orientation for Fintech: Financial Technology)
2	2017/02/24	Fintech 金融科技的演進：貨幣與金融服務 (Evolution of Fintech: Money and Financial Services)
3	2017/03/03	Fintech 金融科技：金融服務科技創新 (Fintech: Technology Innovation in Financial Services)
4	2017/03/10	Fintech 金融科技與金融服務價值鏈 (Fintech and Financial Services Value Chain)
5	2017/03/17	Fintech 金融科技商業模式創新 (Fintech Business Models Innovation)
6	2017/03/24	Fintech 金融科技個案研究 I (Case Study on Fintech I)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
7	2017/03/31	金融服務消費者心理與行為 (Consumer Psychology and Behavior on Financial Services)
8	2017/04/07	教學行政觀摩日 (Off-campus study)
9	2017/04/14	區塊鏈技術 (Blockchain Technology) [Invited Speaker: Dr. Raymund Lin, IBM (林俊叡 博士，IBM)]
10	2017/04/21	期中報告 (Midterm Project Report)
11	2017/04/28	Python Pandas財務大數據分析 (Finance Big Data Analytics with Pandas in Python)
12	2017/05/05	人工智慧與深度學習金融科技 (Artificial Intelligence and Deep Learning for Fintech)

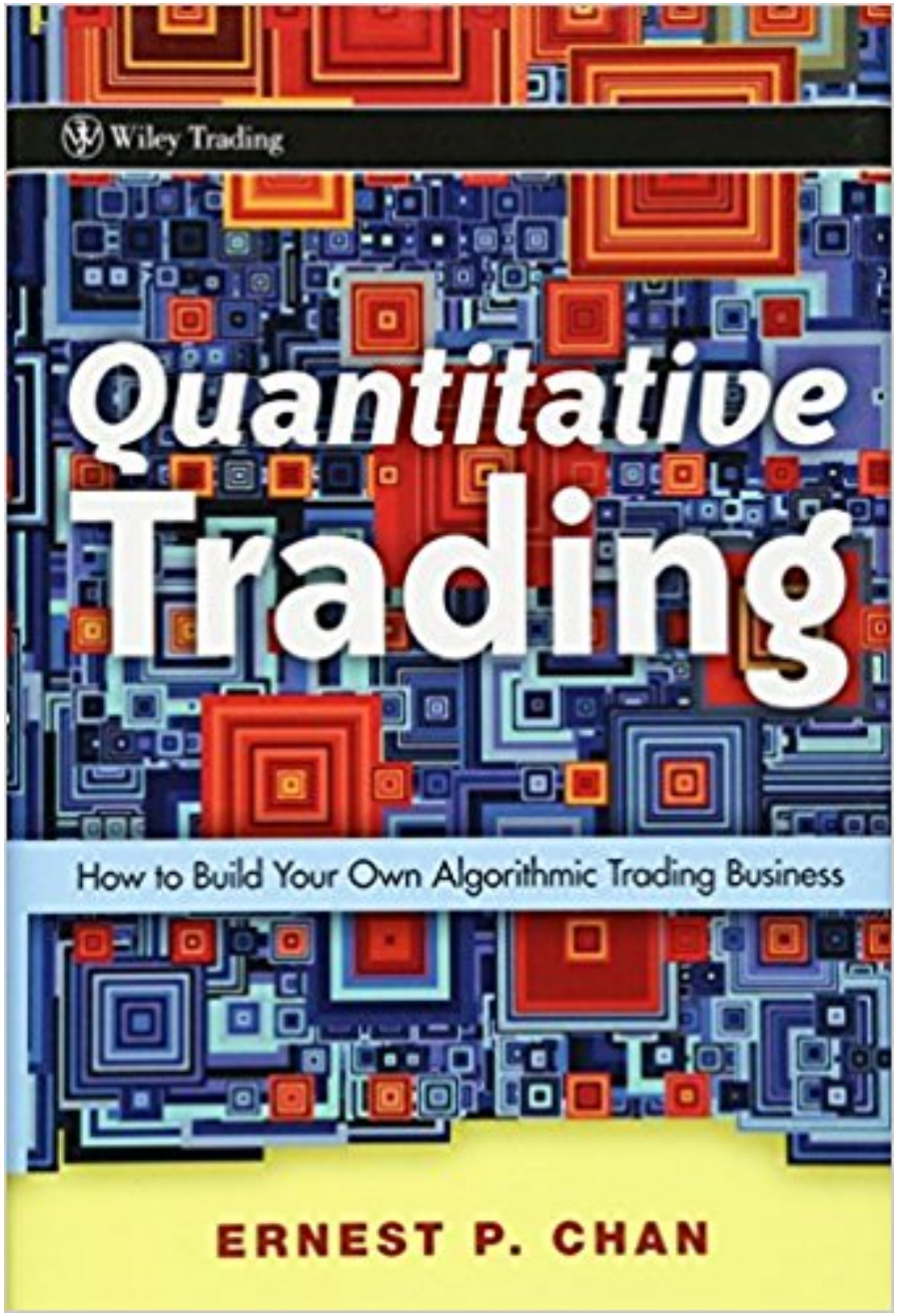
課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
13	2017/05/12	Fintech 金融科技個案研究 II (Case Study on Fintech II)
14	2017/05/19	金融科技財富管理：機器人理財顧問 (Robo-Advisors for Wealth Management in Fintech)
15	2017/05/26	投資組合最佳化與程式交易 (Portfolio Optimization and Algorithmic Trading)
16	2017/06/02	金融科技智慧問答系統 (Intelligent Question Answering System for Fintech)
17	2017/06/09	期末報告 I (Final Project Presentation I)
18	2017/06/16	期末報告 II (Final Project Presentation II)

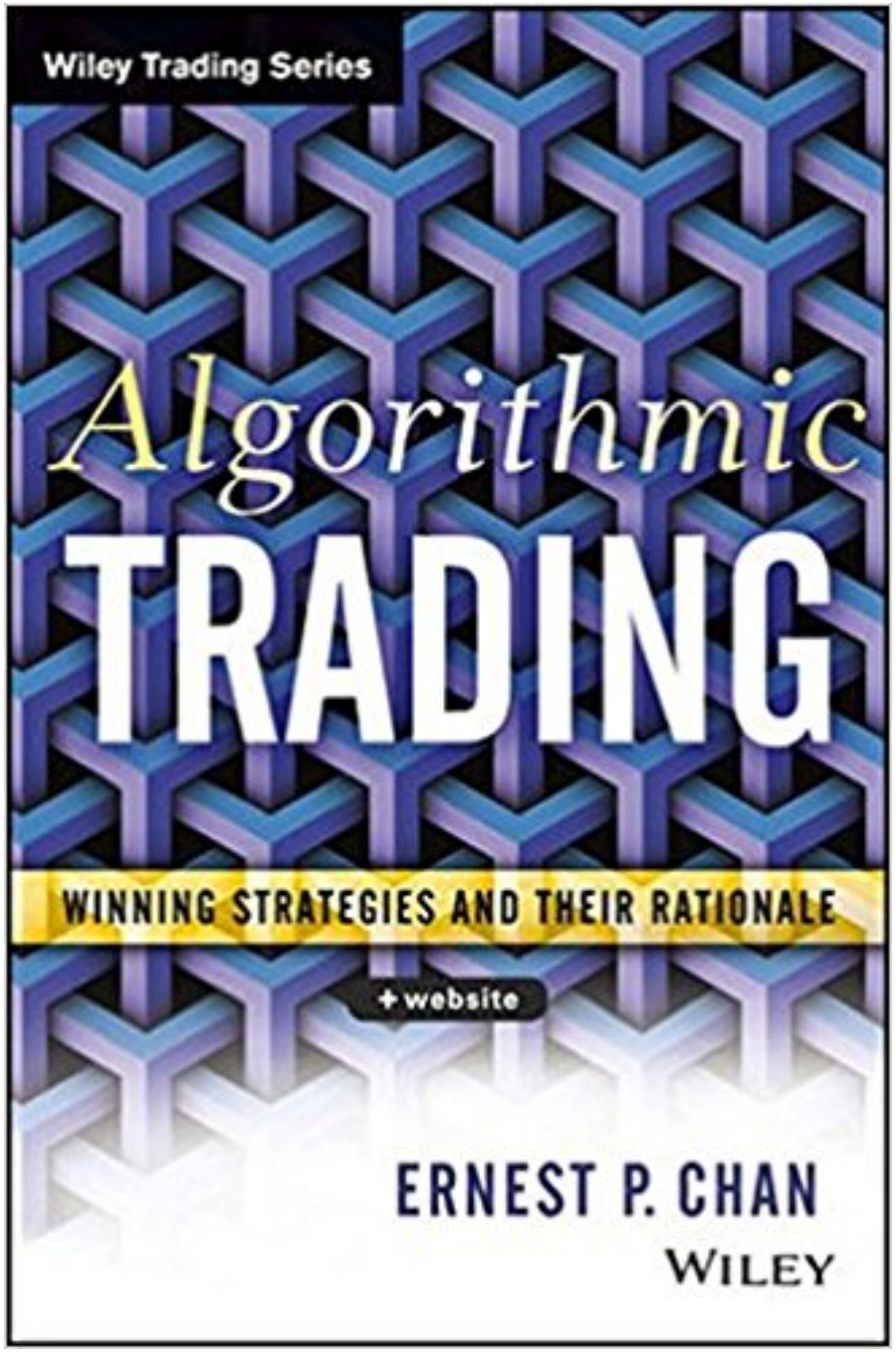
Portfolio Optimization and Algorithmic Trading

Portfolio Optimization

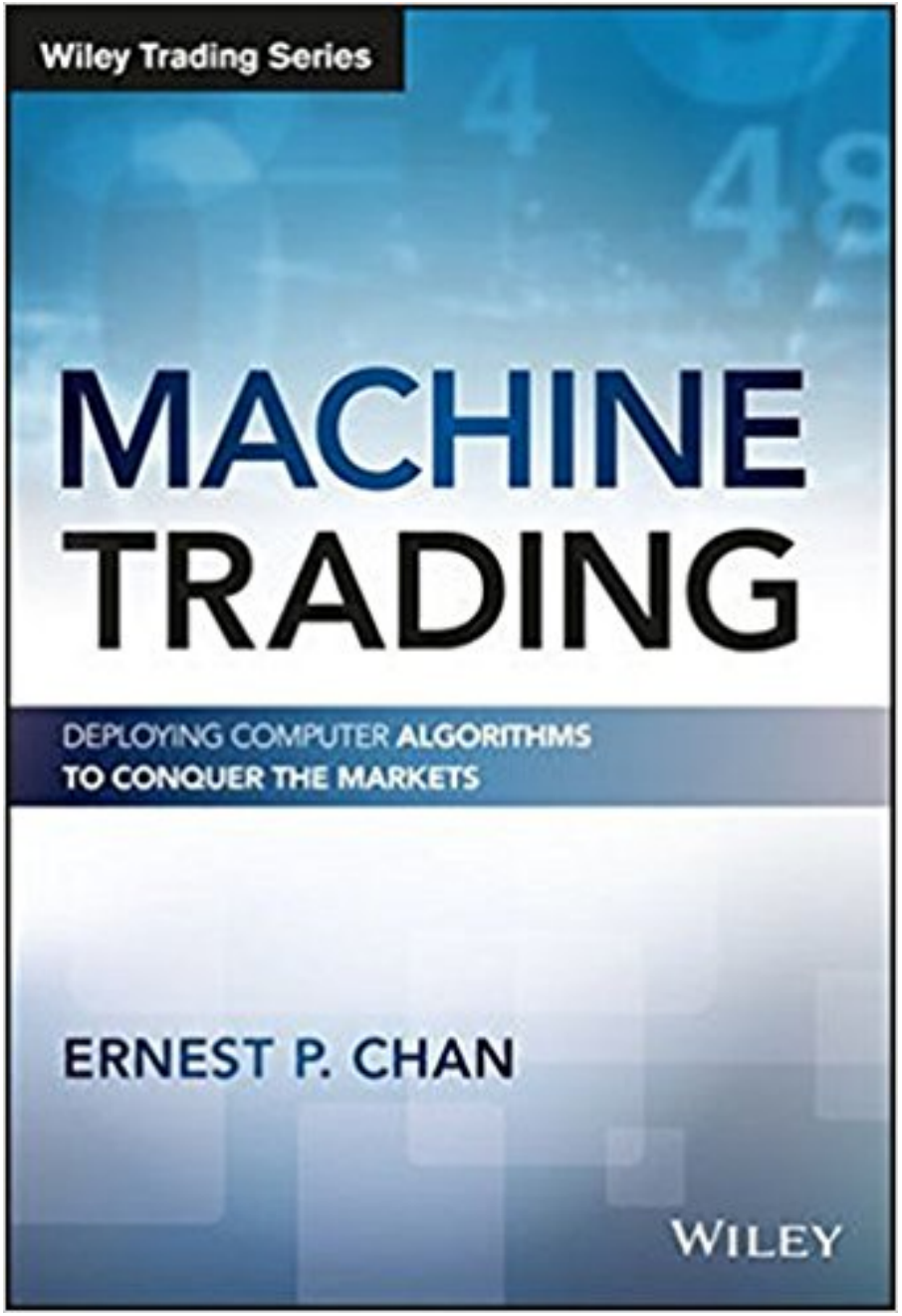
Algorithmic Trading



Source: Ernie Chan (2008), “Quantitative Trading: How to Build Your Own Algorithmic Trading Business”, Wiley

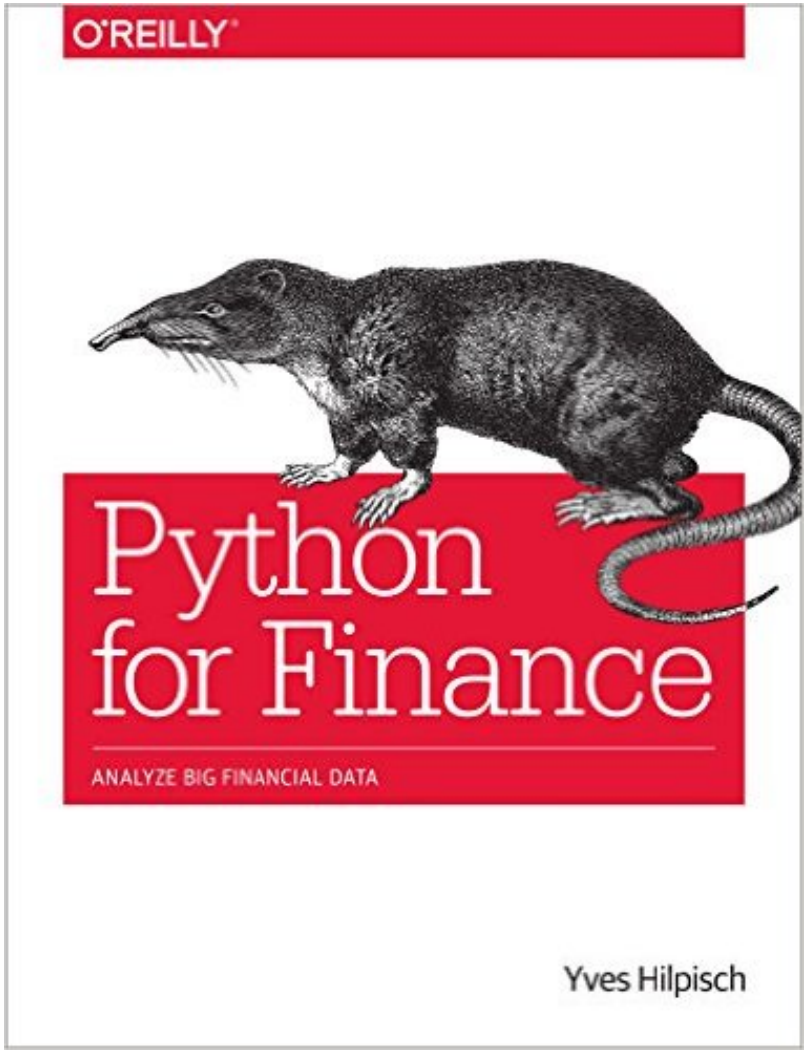


Source: Ernie Chan (2013), “Algorithmic Trading: Winning Strategies and Their Rationale”, Wiley



Source: Ernest P. Chan (2017), “Machine Trading: Deploying Computer Algorithms to Conquer the Markets”, Wiley

Yves Hilpisch, Python for Finance: Analyze Big Financial Data, O'Reilly, 2014

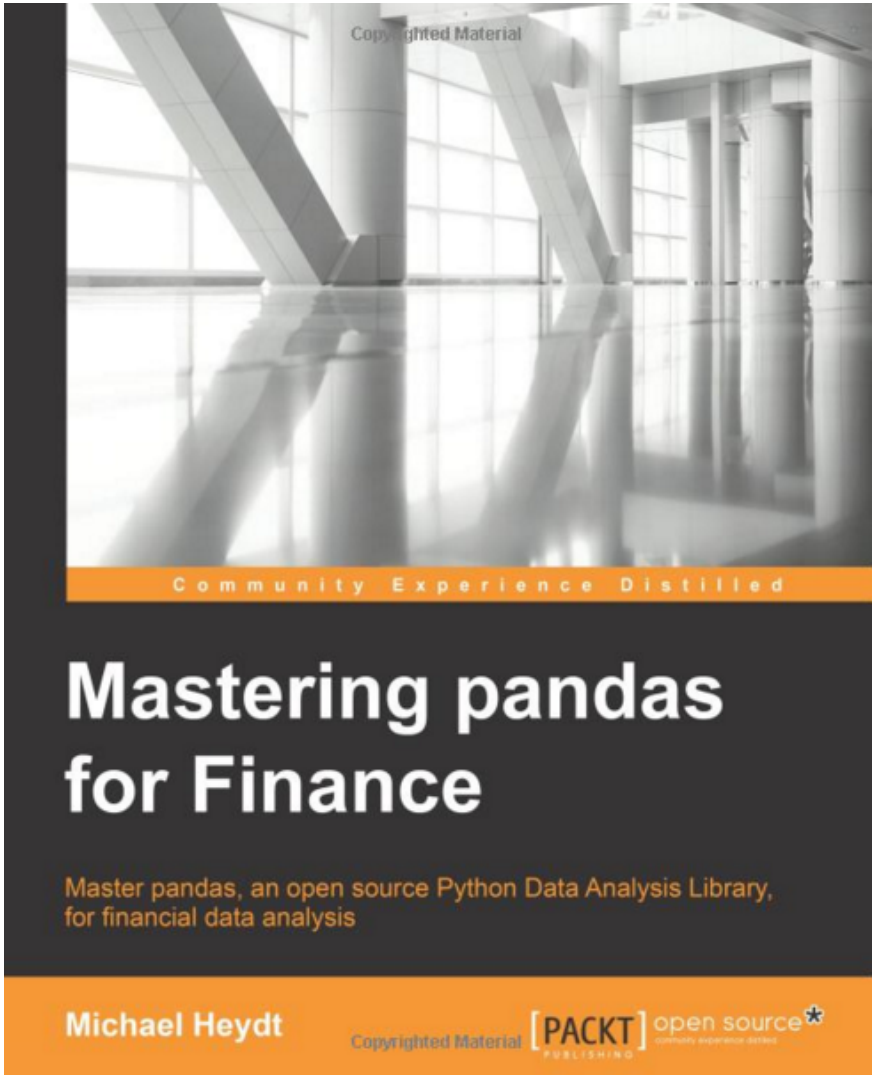


Source: http://www.amazon.com/Python-Finance-Analyze-Financial-Data/dp/1491945281

Yves Hilpisch (2015),
Derivatives Analytics with Python:
Data Analysis, Models, Simulation, Calibration and Hedging, Wiley

Source: http://www.amazon.com/Derivatives-Analytics-Python-Simulation-Calibration/dp/1119037999/

Michael Heydt ,
Mastering Pandas for Finance,
Packt Publishing, 2015



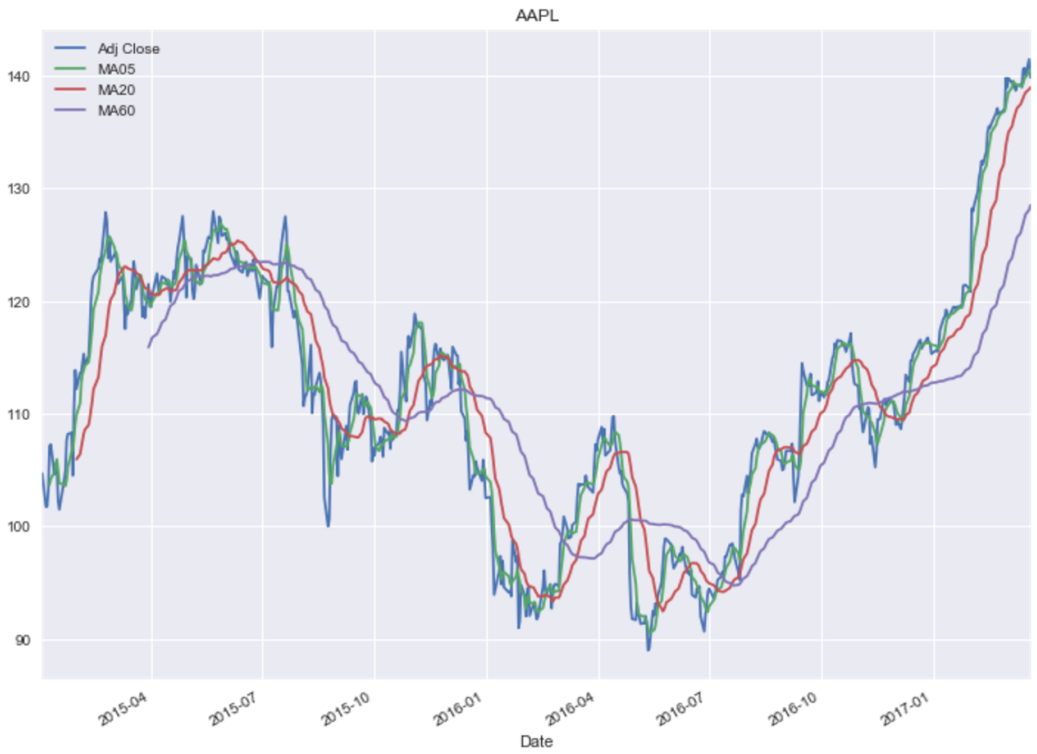
Source: http://www.amazon.com/Mastering-Pandas-Finance-Michael-Heydt/dp/1783985100

Yahoo Finance Charts: Apple Inc. (AAPL)

http://finance.yahoo.com/chart/AAPL

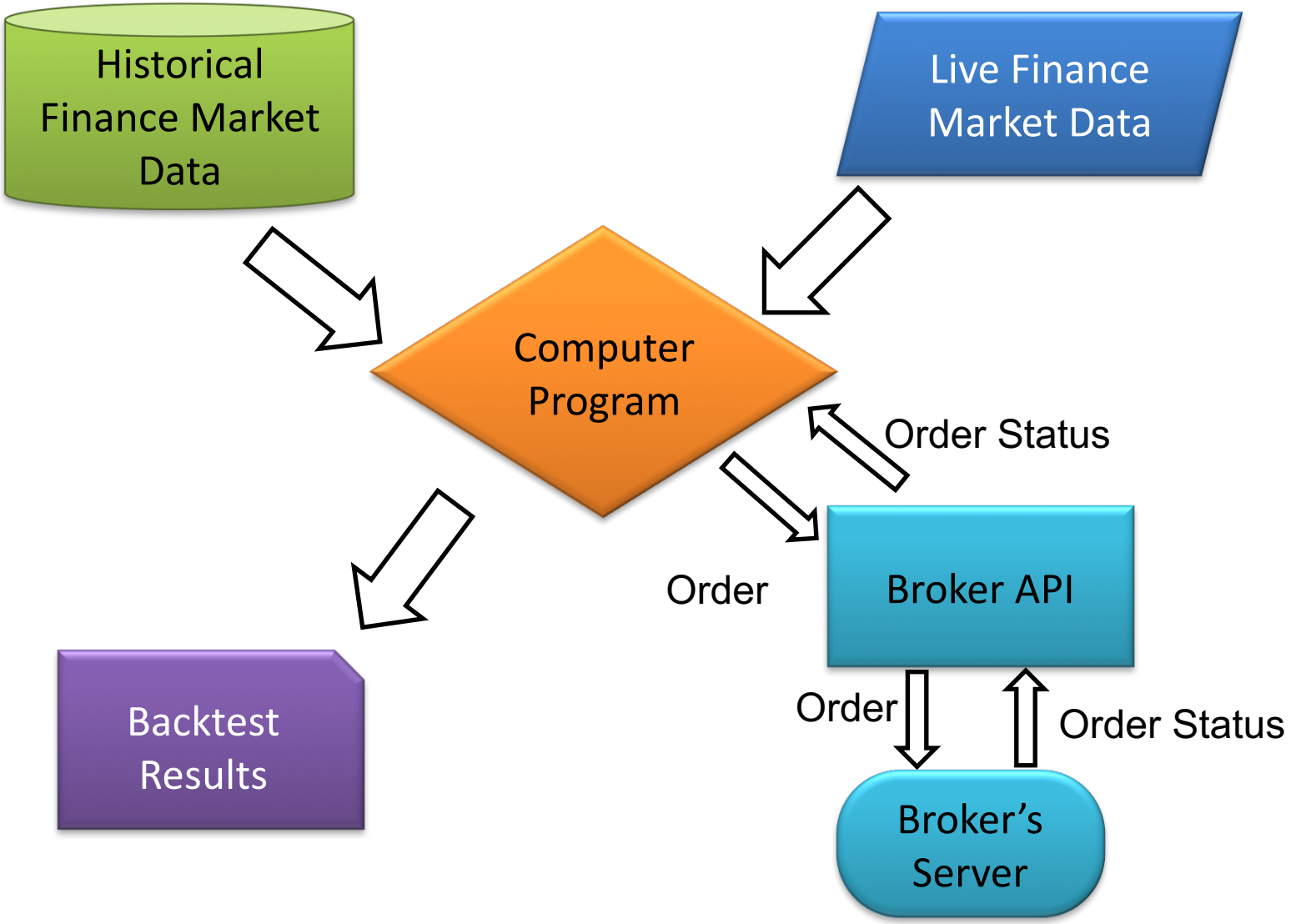
Python Pandas for Finance

```
# 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)
```



Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

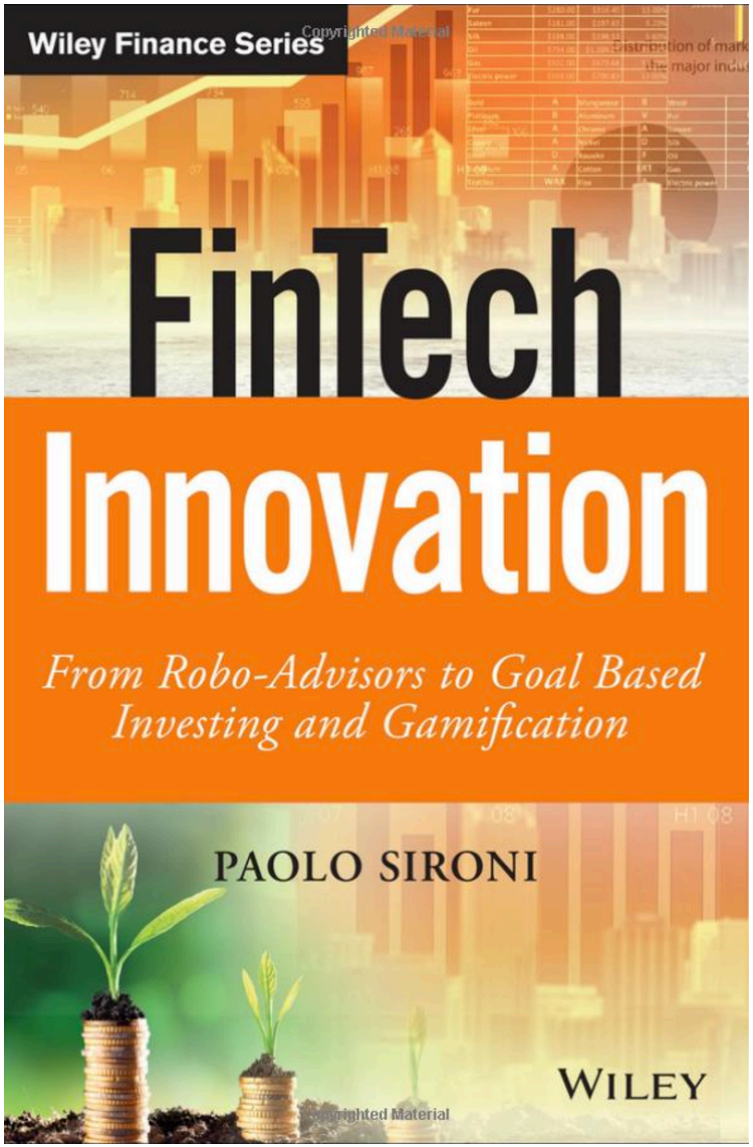
Algorithmic Trading



Source: Ernest P. Chan (2017), Machine Trading: Deploying Computer Algorithms to Conquer the Markets, Wiley

FinTech Innovation:
From Robo-Advisors to Goal Based Investing and Gamification,

Paolo Sironi, Wiley, 2016



Source: https://www.amazon.com/FinTech-Innovation-Robo-Advisors-Investing-Gamification/dp/1119226988

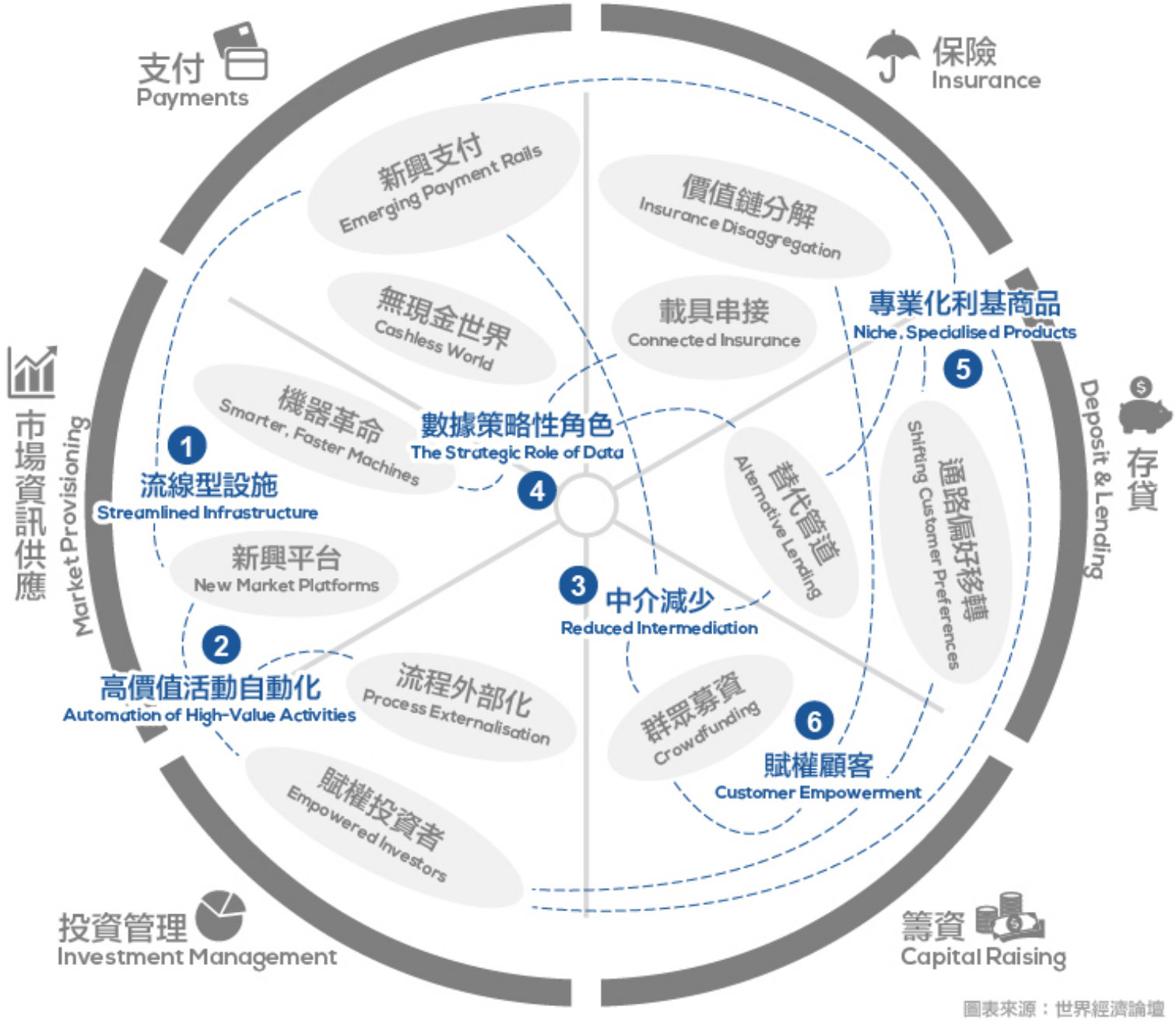
FinTech: Financial Services Innovation

Source: http://www3.weforum.org/docs/WEF_The_future__of_financial_services.pdf

FinTech:
Financial Services Innovation

1. Payments
2. Insurance
3. Deposits & Lending
4. Capital Raising
5. Investment Management
6. Market Provisioning

Source: http://www3.weforum.org/docs/WEF_The_future__of_financial_services.pdf



圖表來源：世界經濟論壇

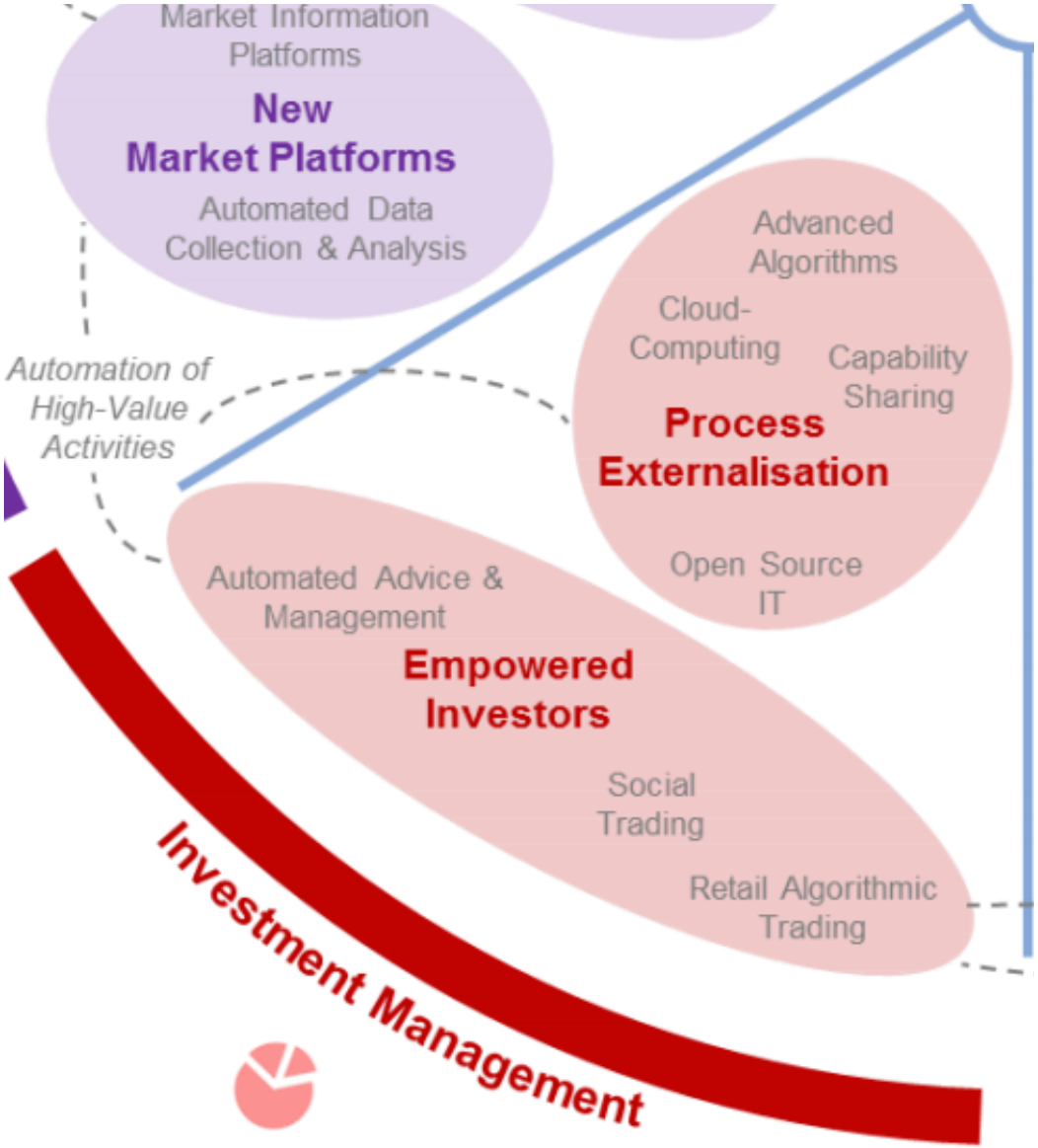
Source: https://www.stockfeel.com.tw/2015年世界經濟論壇－未來的金融服務/

FinTech: Financial Services Innovation

圖表來源：Fugle團隊整理

Source: https://www.stockfeel.com.tw/2015年世界經濟論壇－未來的金融服務/

5 FinTech: Investment Management



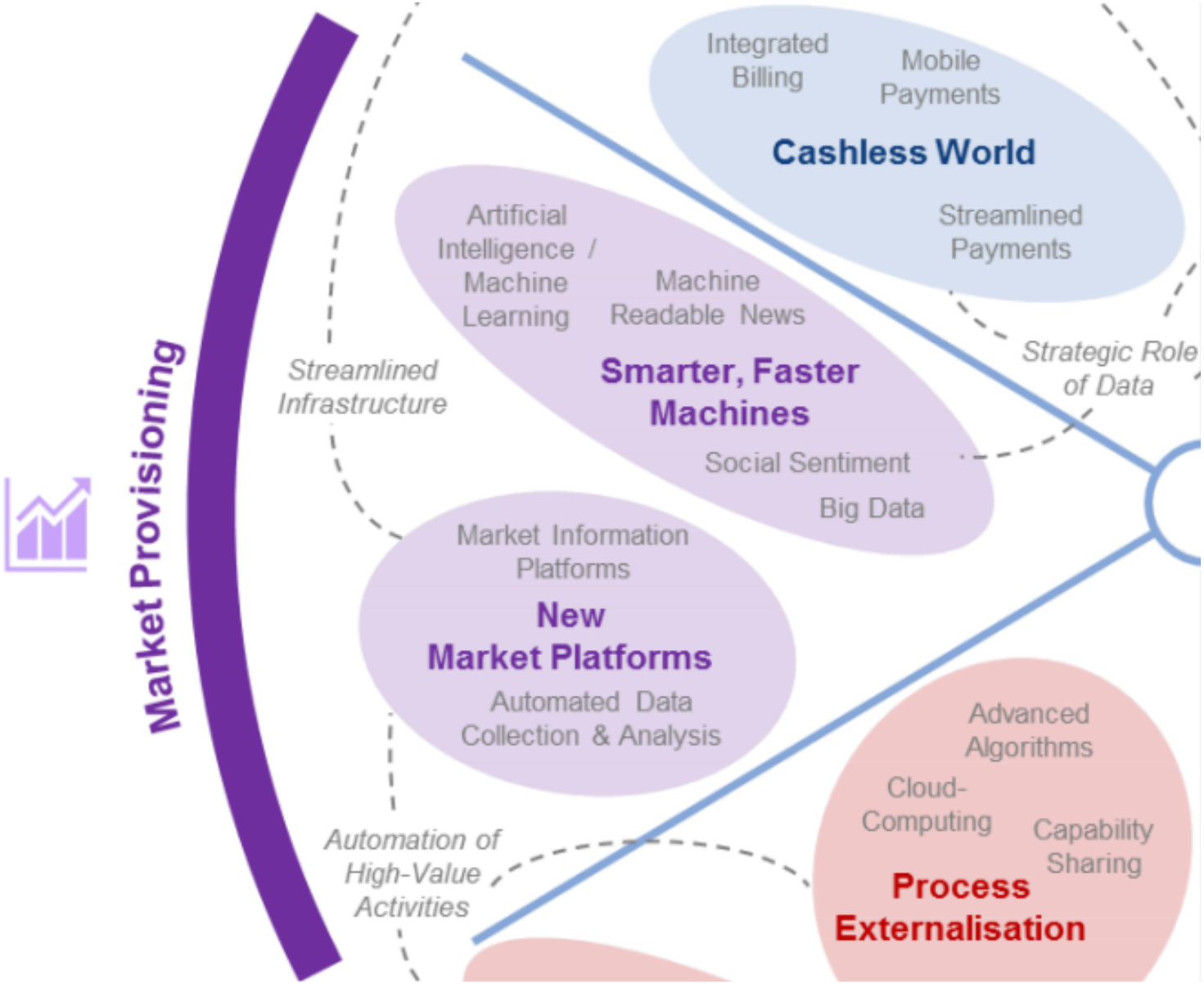
Source: http://www3.weforum.org/docs/WEF_The_future__of_financial_services.pdf

5 FinTech: Investment Management Empowered Investors Process Externalization

圖表來源：Fugle團隊整理

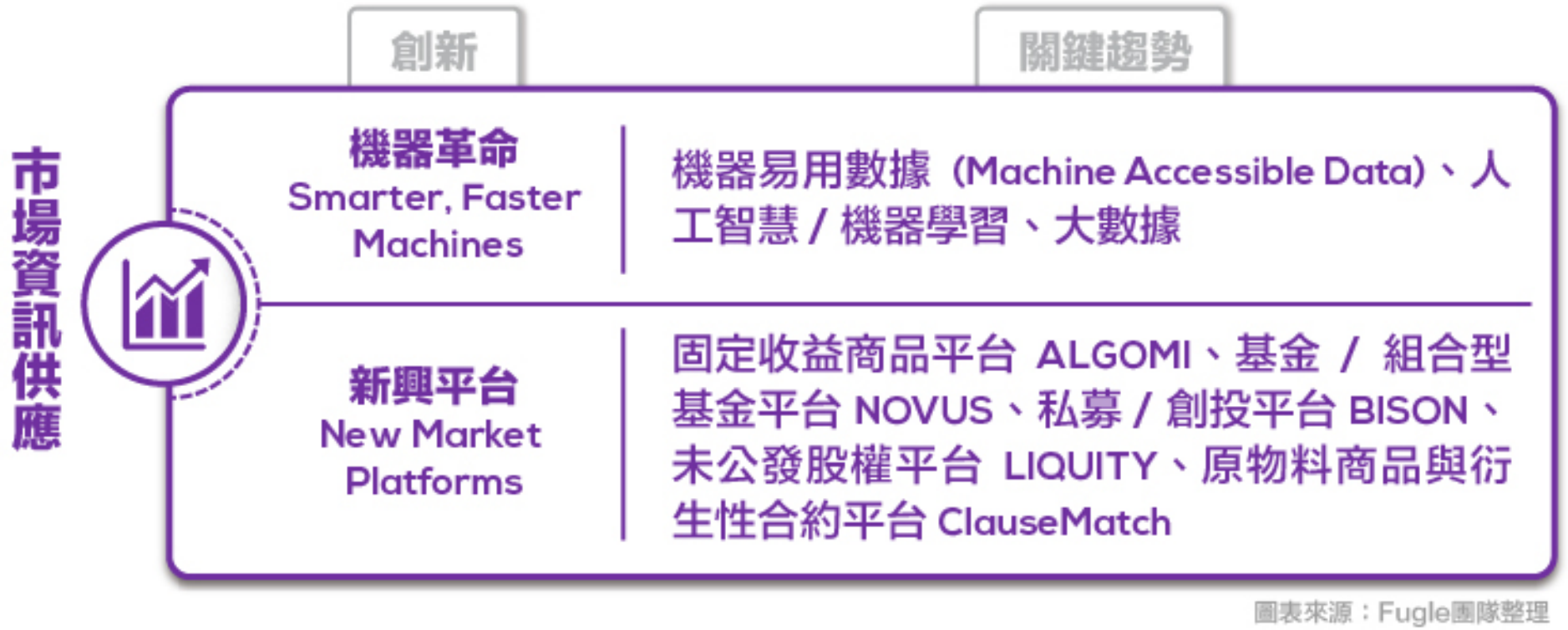
Source: https://www.stockfeel.com.tw/2015年世界經濟論壇－未來的金融服務/

6 FinTech: Market Provisioning



Source: http://www3.weforum.org/docs/WEF_The_future__of_financial_services.pdf

6 FinTech: Market Provisioning Smarter, Faster Machines New Market Platforms



圖表來源：Fugle團隊整理

Source: https://www.stockfeel.com.tw/2015年世界經濟論壇－未來的金融服務/

AI and Big Data Analytics in Finance

- 財務金融大數據分析
  - Big Data Analytics in Finance
- 投資大數據分析
  - Big Data Analytics in Investment
- 人工智慧與財務應用
  - Artificial Intelligence and Financial Application
- 人工智慧與投資分析
  - Artificial Intelligence and Investment Analysis

財務金融大數據分析

Big Data Analytics in Finance

投資大數據分析

Big Data Analytics in Investment

人工智慧與財務應用

Artificial Intelligence and Financial Application

人工智慧與投資分析

Artificial Intelligence and Investment Analysis

```
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()
```

```
import pandas as pd
import pandas_datareader.data as web
#df = web.DataReader('AAPL', 'yahoo')
df = web.DataReader('AAPL', data_source='yahoo', start='1/1/2010', end='3/21/2017')
#df = web.DataReader('AAPL', data_source='google', start='1/1/2010', end='3/21/2017')
df.to_csv('AAPL.csv')
df.tail()
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2017-03-15	139.410004	140.750000	139.029999	140.460007	25566800	140.460007
2017-03-16	140.720001	141.020004	140.259995	140.690002	19132500	140.690002
2017-03-17	141.000000	141.000000	139.889999	139.990005	43597400	139.990005
2017-03-20	140.399994	141.500000	140.229996	141.460007	20213100	141.460007
2017-03-21	142.110001	142.800003	139.729996	139.839996	39116800	139.839996

```
df = web.DataReader('GOOG',
data_source='yahoo', start='1/1/1980',
end='3/21/2017')
df.head(10)
```

```
df = web.DataReader('GOOG', data_source='yahoo', start='1/1/1980', end='3/21/2017')
df.head(10)
```

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

df.tail(10)

```
df.tail(10)
```

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

df.count()

```
df.count()
```

```
Open          3169
High          3169
Low           3169
Close         3169
Volume        3169
Adj Close     3169
dtype: int64
```

df.ix['2015-12-31']

```
df.ix['2015-12-31']
```

```
Open          7.695000e+02
High          7.695000e+02
Low           7.583400e+02
Close         7.588800e+02
Volume        1.489600e+06
Adj Close     7.588800e+02
Name: 2015-12-31 00:00:00, dtype: float64
```

```
df.to_csv('2330.TW.Yahoo.Finance.Data.csv')
```

2330.TW.Yahoo.Finance.Data.csv

```
Date,Open,High,Low,Close,Volume,Adj Close
2010-01-01,64.5,64.5,64.5,64.5,0,52.8308
2010-01-04,65.0,65.0,64.0,64.9,39407000,53.1584
2010-01-05,65.0,65.1,63.9,64.5,37138000,52.8308
2010-01-06,64.5,64.9,63.7,64.9,49261000,53.1584
2010-01-07,64.9,65.0,64.2,64.2,42134000,52.5851
2010-01-08,63.5,64.3,63.5,64.0,46076000,52.4213
2010-01-11,64.0,64.9,63.5,64.5,36799000,52.8308
2010-01-12,64.4,64.4,63.3,63.6,49853000,52.0936
2010-01-13,63.0,63.1,62.6,62.8,47976000,51.4384
2010-01-14,63.6,63.6,63.0,63.2,36149000,51.766
2010-01-15,62.9,63.5,62.8,63.5,47852000,52.0117
2010-01-18,62.8,63.1,62.8,62.9,30136000,51.5203
2010-01-19,63.0,63.2,62.0,62.5,47202000,51.1926
2010-01-20,62.9,63.2,62.2,63.0,52281000,51.6022
```

Python Pandas for Finance

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Python Pandas for Finance

```
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
```

```
#Python for Stocks: 1
#Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/
#Import Packages Required
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
```

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Python Pandas for Finance

```
#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2015, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

```
#Read Stock Data from Yahoo Finance
end = dt.datetime.now()
#start = dt.datetime(end.year-2, end.month, end.day)
start = dt.datetime(2015, 1, 1)
df = web.DataReader("AAPL", 'yahoo', start, end)
df.to_csv('AAPL.csv')
df.from_csv('AAPL.csv')
df.tail()
```

	Open	High	Low	Close	Volume	Adj Close
Date						
2017-03-15	139.410004	140.750000	139.029999	140.460007	25566800	140.460007
2017-03-16	140.720001	141.020004	140.259995	140.690002	19132500	140.690002
2017-03-17	141.000000	141.000000	139.889999	139.990005	43597400	139.990005
2017-03-20	140.399994	141.500000	140.229996	141.460007	20213100	141.460007
2017-03-21	142.110001	142.800003	139.729996	139.839996	39116800	139.839996

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Python Pandas for Finance

```
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
```

```
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
<matplotlib.axes._subplots.AxesSubplot at 0x1150bac88>
```

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Python Pandas for Finance

```
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')
```

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Python Pandas for Finance

```
<matplotlib.text.Text at 0x115630860>
```

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Python Pandas for Finance

```
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')
```

<matplotlib.text.Text at 0x1153630b60>

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

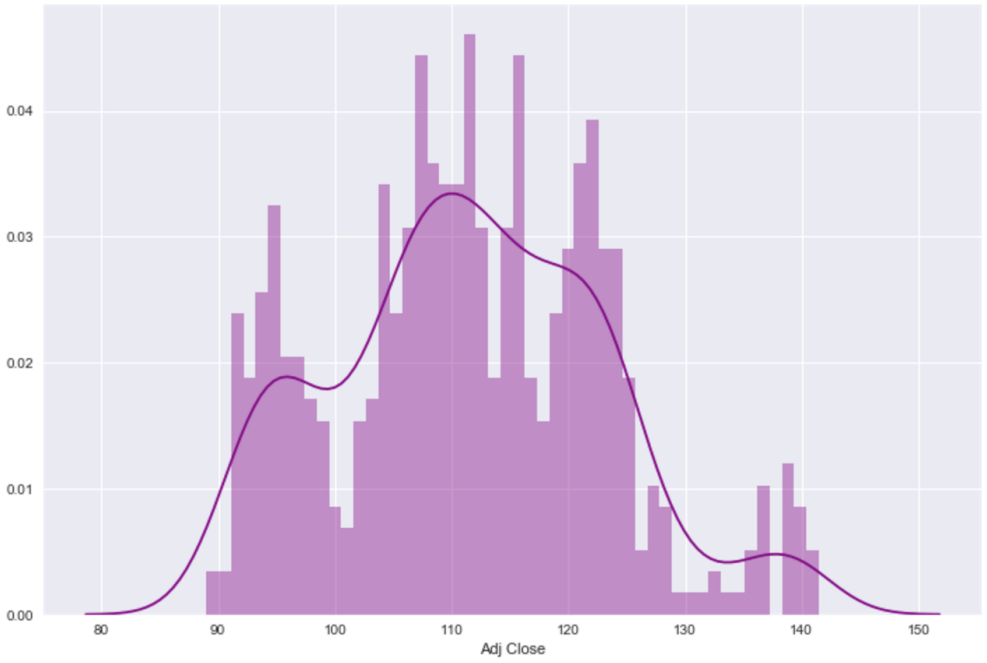
Python Pandas for Finance

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

```
plt.figure(figsize=(12,8))
sns.distplot(df['Adj Close'].dropna(), bins=50, color='purple')
```

```
/Users/imyday/anaconda/lib/python3.6/site-packages/statsmodels/nonparametric/kdetools
g: using a non-integer number instead of an integer will result in an error in the fu
  y = X[:m/2+1] + np.r_[0,X[m/2+1:],0]*1j
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x116309780>
```



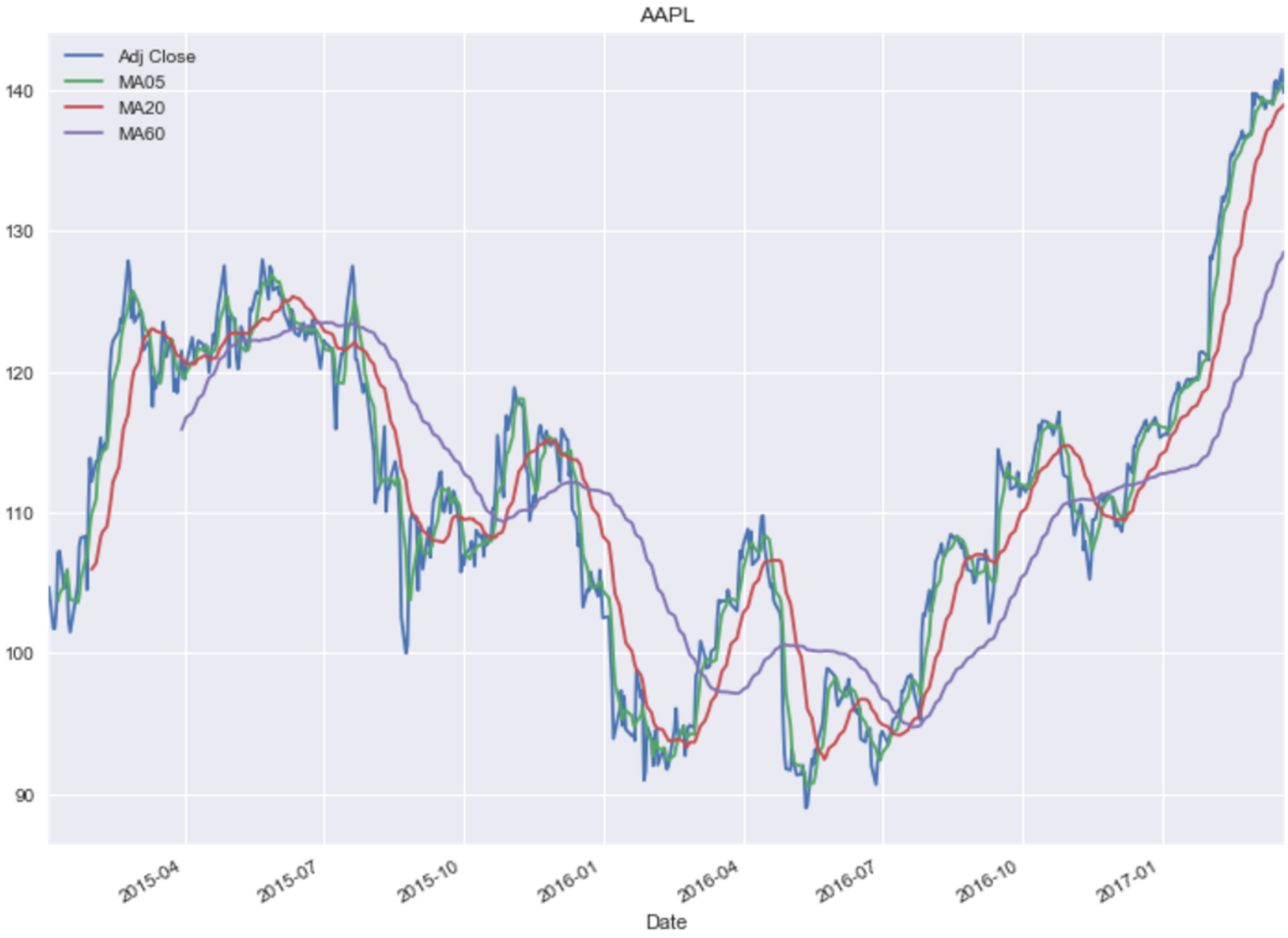
Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Python Pandas for Finance

```
# simple moving averages
df['MA05'] = df['Adj Close'].rolling(5).mean()
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()
```

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

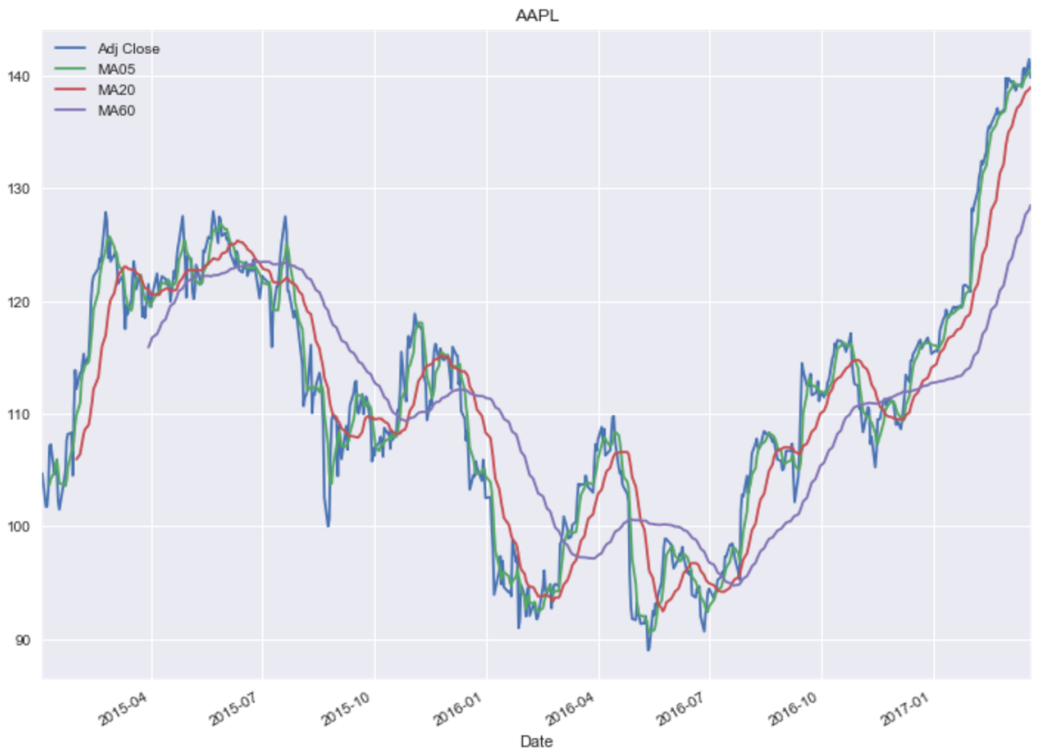
Python Pandas for Finance



Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Python Pandas for Finance

```
# 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)
```



Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

```
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(2015, 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)
plt.show()
```

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Examples: Python Pandas for Finance

In [11]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import pandas.io.data as web
import random

import datetime
import time
import timeit

import io
import os

import re
import codecs
import requests
get_ipython().magic('matplotlib inline')

from scipy import stats

#pd.set_option('display.notebook_repr_html', False)
pd.set_option('display.max_columns', 15)
pd.set_option('display.max_rows', 10)
pd.set_option('precision', 3)

def getDateTimeNow():
    strnow = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
    return strnow

print('Hello Pandas')
print(getDateTimeNow())
```

```
Hello Pandas
20160323_145708
```

```
sSymbol = "AAPL"
#sSymbol = "GOOG"
#sSymbol = "IBM"
#sSymbol = "MSFT"
#sSymbol = "^TWII"
#sSymbol = "000001.SS"
#sSymbol = "2330.TW"
#sSymbol = "2317.TW"

# sURL = "http://ichart.finance.yahoo.com/table.csv?s=AAPL"
# sBaseURL = "http://ichart.finance.yahoo.com/table.csv?s="
sURL = "http://ichart.finance.yahoo.com/table.csv?s=" + sSymbol
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=2330.TW")
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=AAPL")
req = requests.get(sURL)

sText = req.text
#print(sText)
#df = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
#df = web.DataReader("2330.TW", 'yahoo')

sPath = "data/"
sPathFilename = sPath + sSymbol + ".csv"
print(sPathFilename)

f = open(sPathFilename, 'w')
f.write(sText)
f.close()
sIOdata = io.StringIO(sText)
df = pd.DataFrame.from_csv(sIOdata)
df.head(5)
```

In [13]:
```
starttime = datetime.datetime(2000, 1, 1)
endtime = datetime.datetime(2015, 12, 31)
sSymbol = "AAPL"
#sSymbol = "GOOG"
#sSymbol = "IBM"
#sSymbol = "MSFT"
#sSymbol = "^TWII"
#sSymbol = "000001.SS"
#sSymbol = "2330.TW"
#sSymbol = "2317.TW"
#"^TWII"
#"AAPL"
#SHA:000016"
#"600000.SS"
#"2330.TW"
df = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
#df_01 = web.DataReader("2330.TW", 'yahoo')
sSymbol = sSymbol.replace(":","_")
sSymbol = sSymbol.replace("^","_")
sPath = "data/financedata/"
#sPath = "/users/imyday/SCDBA/data/"  #Mac OS X
#sPath = "C:\data\"                   #Windows

sPathFilename = sPath + sSymbol + "_Yahoo_Finance.csv"
print(sPathFilename)
df.to_csv(sPathFilename)
df.head(5)
```

```
data/financedata/AAPL_Yahoo_Finance.csv
```

Out[13]:

	Open	High	Low	Close	Volume	Adj Close
Date						
2000-01-03	104.875	112.500	101.688	111.938	133949200	3.702
2000-01-04	108.250	110.625	101.188	102.500	128094400	3.390

df.tail(5)

```
In [14]: df.tail(5)
```

Out[14]:

	Open	High	Low	Close	Volume	Adj Close
Date						
2015-12-24	109.00	109.00	107.95	108.03	13596700	107.447
2015-12-28	107.59	107.69	106.18	106.82	26704200	106.243
2015-12-29	106.96	109.43	106.86	108.74	30931200	108.153
2015-12-30	108.58	108.70	107.18	107.32	25213800	106.741
2015-12-31	107.01	107.03	104.82	105.26	40912300	104.692

```
sSymbol = "AAPL"

# sURL = "http://ichart.finance.yahoo.com/table.csv?s=AAPL"
sURL = "http://ichart.finance.yahoo.com/table.csv?s=" + sSymbol
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=AAPL")
req = requests.get(sURL)

sText = req.text
#print(sText)

sPath = "data/"
sPathFilename = sPath + sSymbol + ".csv"
print(sPathFilename)

f = open(sPathFilename, 'w')
f.write(sText)
f.close()
sIOdata = io.StringIO(sText)
df = pd.DataFrame.from_csv(sIOdata)
df.head(5)
```

In [15]:
```
sSymbol = "AAPL"
#sSymbol = "GOOG"
#sSymbol = "IBM"
#sSymbol = "MSFT"
#sSymbol = "^TWII"
#sSymbol = "000001.SS"
#sSymbol = "2330.TW"
#sSymbol = "2317.TW"

# sURL = "http://ichart.finance.yahoo.com/table.csv?s=AAPL"
# sBaseURL = "http://ichart.finance.yahoo.com/table.csv?s="
sURL = "http://ichart.finance.yahoo.com/table.csv?s=" + sSymbol
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=2330.TW")
#req = requests.get("http://ichart.finance.yahoo.com/table.csv?s=AAPL")
req = requests.get(sURL)

sText = req.text
#print(sText)
#df = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
#df = web.DataReader("2330.TW", 'yahoo')

sPath = "data/"
sPathFilename = sPath + sSymbol + ".csv"
print(sPathFilename)

f = open(sPathFilename, 'w')
f.write(sText)
f.close()
sIOdata = io.StringIO(sText)
df = pd.DataFrame.from_csv(sIOdata)
df.head(5)
```

```
data/AAPL.csv
```

```
data/AAPL.csv
```

Out[15]:

	Open	High	Low	Close	Volume	Adj Close
Date						
2016-03-22	105.25	107.29	105.21	106.72	32232600	106.72
2016-03-21	105.93	107.65	105.14	105.91	35180800	105.91
2016-03-18	106.34	106.50	105.19	105.92	43402300	105.92
2016-03-17	105.52	106.47	104.96	105.80	34244600	105.80
2016-03-16	104.61	106.31	104.59	105.97	37893800	105.97

```
def getYahooFinanceData(sSymbol, starttime, endtime, sDir):
    #GetMarketFinanceData_From_YahooFinance
    #"^TWII"
    #"000001.SS"
    #"AAPL"
    #SHA:000016"
    #"600000.SS"
    #"2330.TW"
    #sSymbol = "^TWII"
    starttime = datetime.datetime(2000, 1, 1)
    endtime = datetime.datetime(2015, 12, 31)
    sPath = sDir
    #sPath = "data/financedata/"
   df_YahooFinance = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
    #df_01 = web.DataReader("2330.TW", 'yahoo')
    sSymbol = sSymbol.replace(":","_")
    sSymbol = sSymbol.replace("^","_")
    sPathFilename = sPath + sSymbol + "_Yahoo_Finance.csv"
    df_YahooFinance.to_csv(sPathFilename)
    #df_YahooFinance.head(5)
    return sPathFilename
    #End def getYahooFinanceData(sSymbol, starttime, endtime, sDir):
```

In [16]:
```
def getYahooFinanceData(sSymbol, starttime, endtime, sDir):
    #GetMarketFinanceData_From_YahooFinance
    #"^TWII"
    #"000001.SS"
    #"AAPL"
    #SHA:000016"
    #"600000.SS"
    #"2330.TW"
    #sSymbol = "^TWII"
    starttime = datetime.datetime(2000, 1, 1)
    endtime = datetime.datetime(2015, 12, 31)
    sPath = sDir
    #sPath = "data/financedata/"
    df_YahooFinance = web.DataReader(sSymbol, 'yahoo', starttime, endtime)
    #df_01 = web.DataReader("2330.TW", 'yahoo')
    sSymbol = sSymbol.replace(":","_")
    sSymbol = sSymbol.replace("^","_")
    sPathFilename = sPath + sSymbol + "_Yahoo_Finance.csv"
    df_YahooFinance.to_csv(sPathFilename)
    #df_YahooFinance.head(5)
    return sPathFilename
    #End def getYahooFinanceData(sSymbol, starttime, endtime, sDir):
```

```
sSymbol = "AAPL”
starttime = datetime.datetime(2000, 1, 1)
endtime = datetime.datetime(2015, 12, 31)
sDir = "data/financedata/"

sPathFilename = getYahooFinanceData(sSymbol, starttime, endtime,
sDir)
print(sPathFilename)
```

In [17]:
```
sSymbol = "AAPL"
#sSymbol = "GOOG"
#sSymbol = "IBM"
#sSymbol = "MSFT"
#sSymbol = "^TWII"
#sSymbol = "000001.SS"
#sSymbol = "2330.TW"
#sSymbol = "2317.TW"
starttime = datetime.datetime(2000, 1, 1)
endtime = datetime.datetime(2015, 12, 31)
sDir = "data/financedata/"

sPathFilename = getYahooFinanceData(sSymbol, starttime, endtime, sDir)
print(sPathFilename)
```

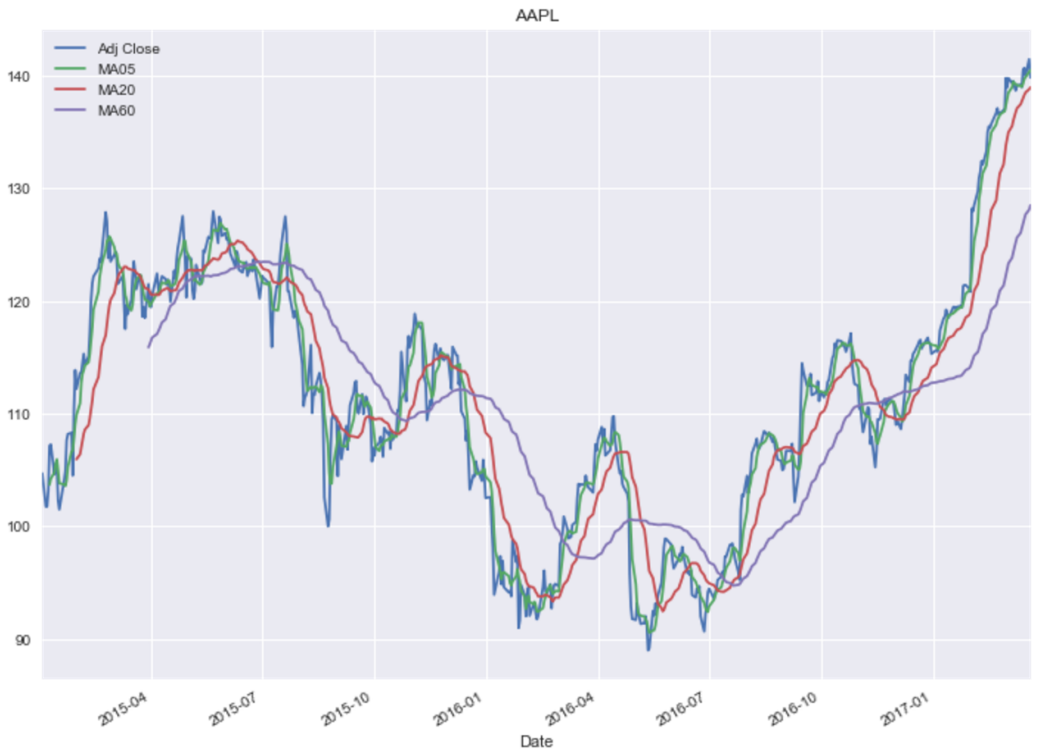
```
data/financedata/AAPL_Yahoo_Finance.csv
```

```
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(2015, 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)
plt.show()
```

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

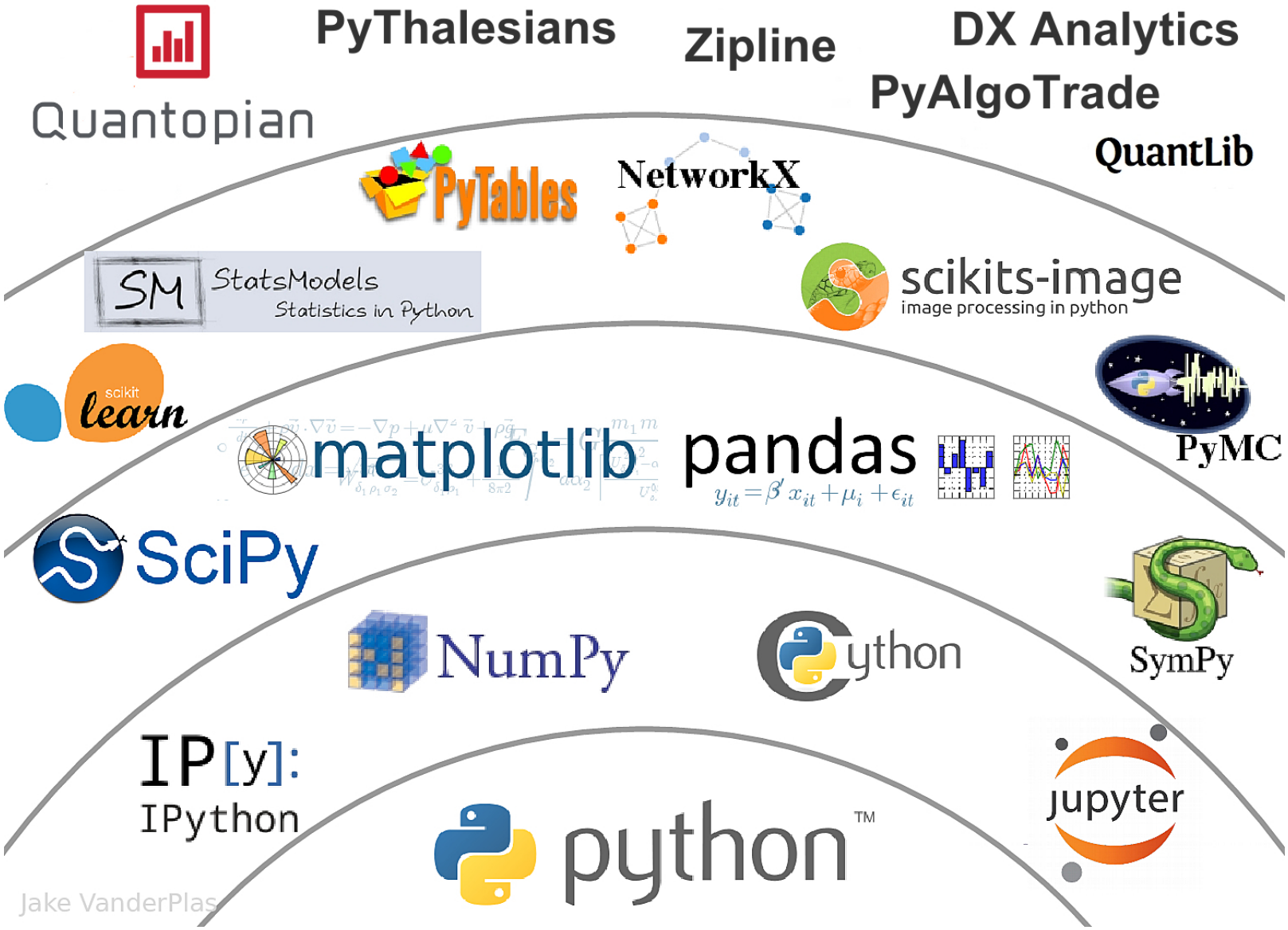
Python Pandas for Finance

```
# 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)
```



Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

The Quant Finance PyData Stack



Source: http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview_slides.ipynb#/5

Zipline

a Pythonic

Algorithmic Trading Library

http://www.zipline.io/

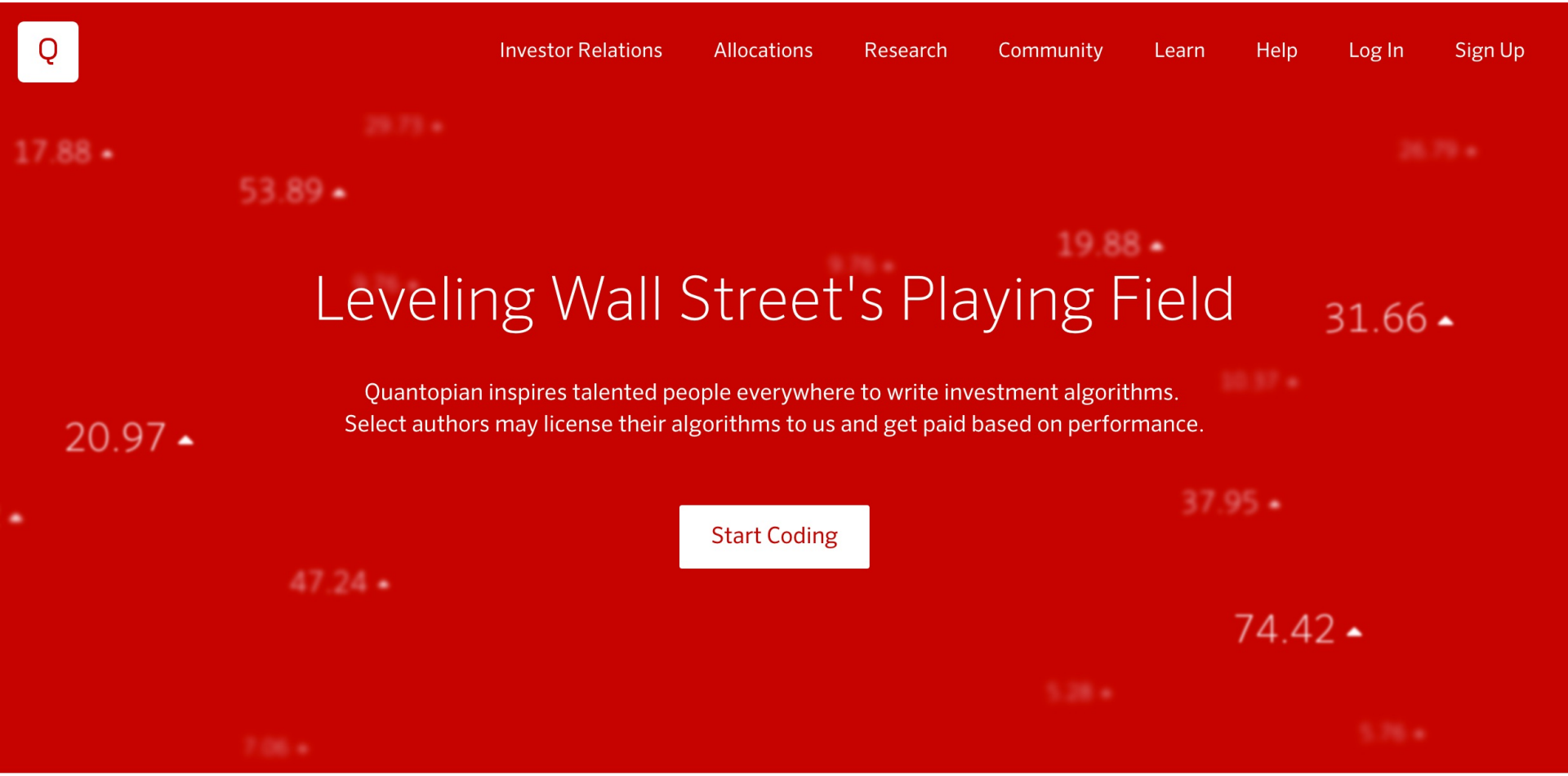
Source: https://github.com/quantopian/zipline

Zipline

- Zipline: Pythonic algorithmic trading library.
- Event-driven system
  - supports both backtesting and live-trading.
- Zipline is currently used in production as the backtesting and live-trading engine powering Quantopian
  - a free, community-centered, hosted platform for building and executing trading strategies.

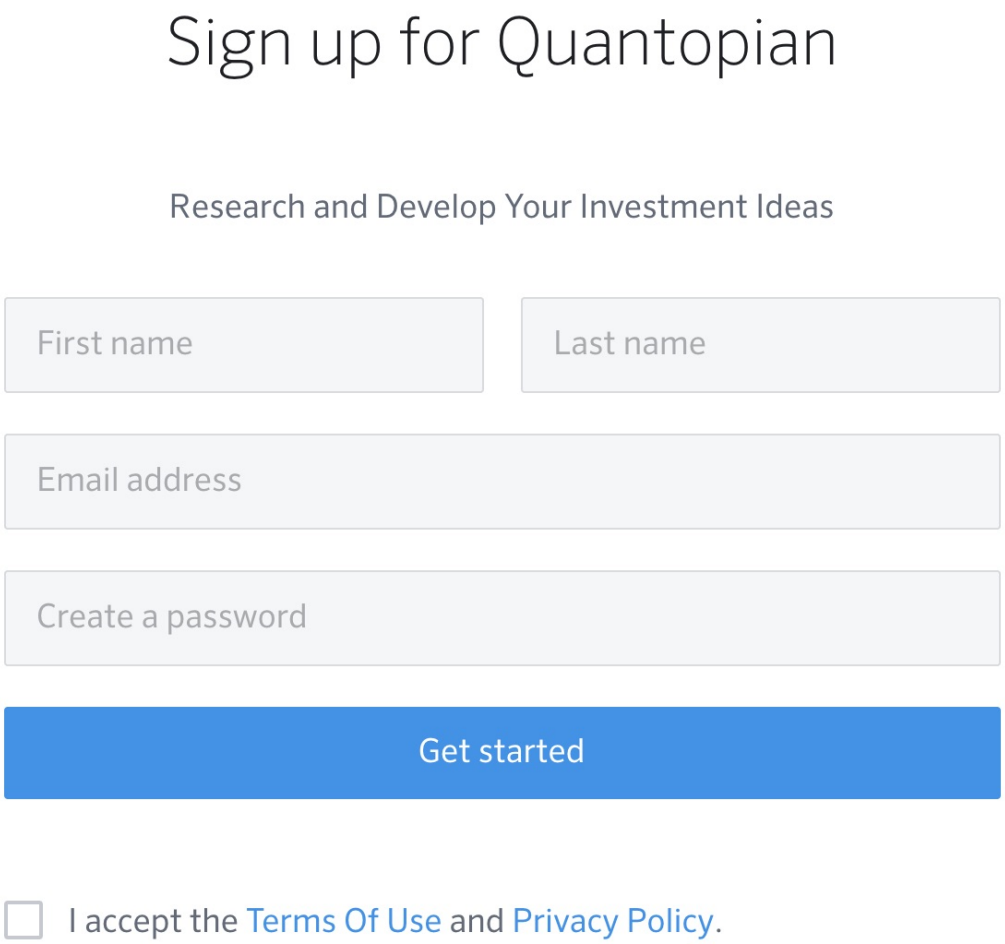
Source: http://www.zipline.io/

Quantopian



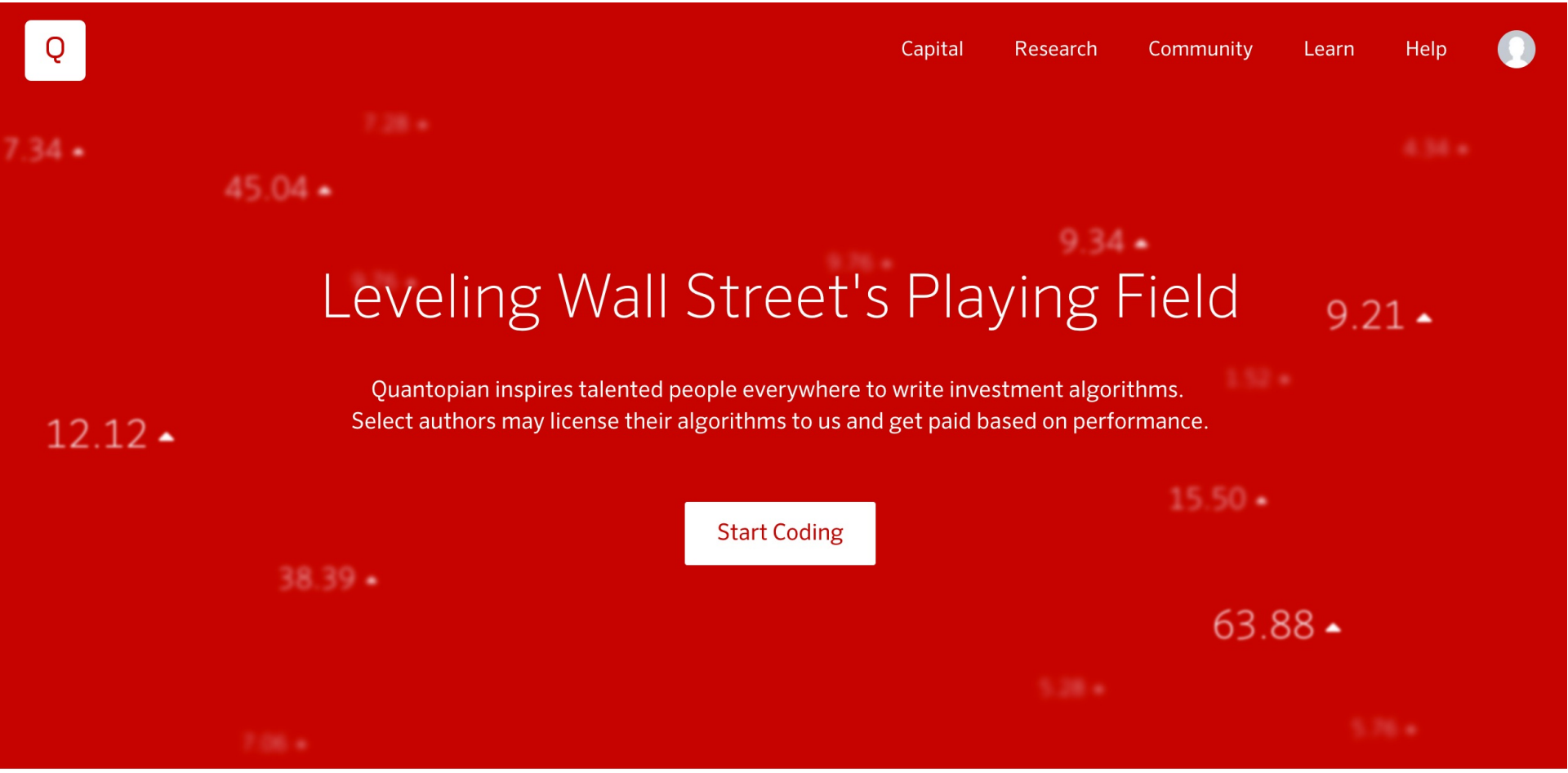
https://www.quantopian.com/

Sign up for Quantopian



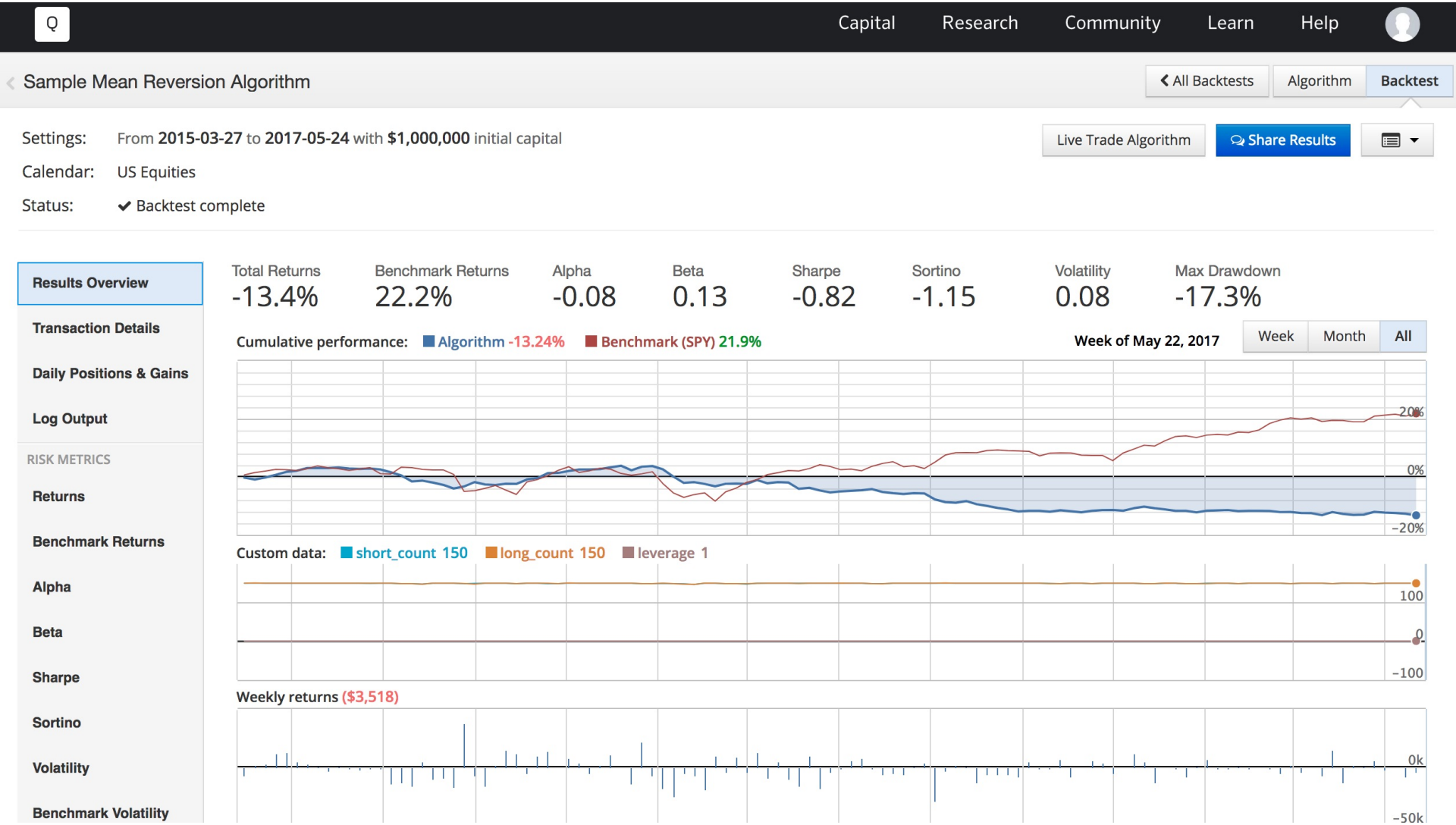
https://www.quantopian.com/users/sign_up

Quantopian



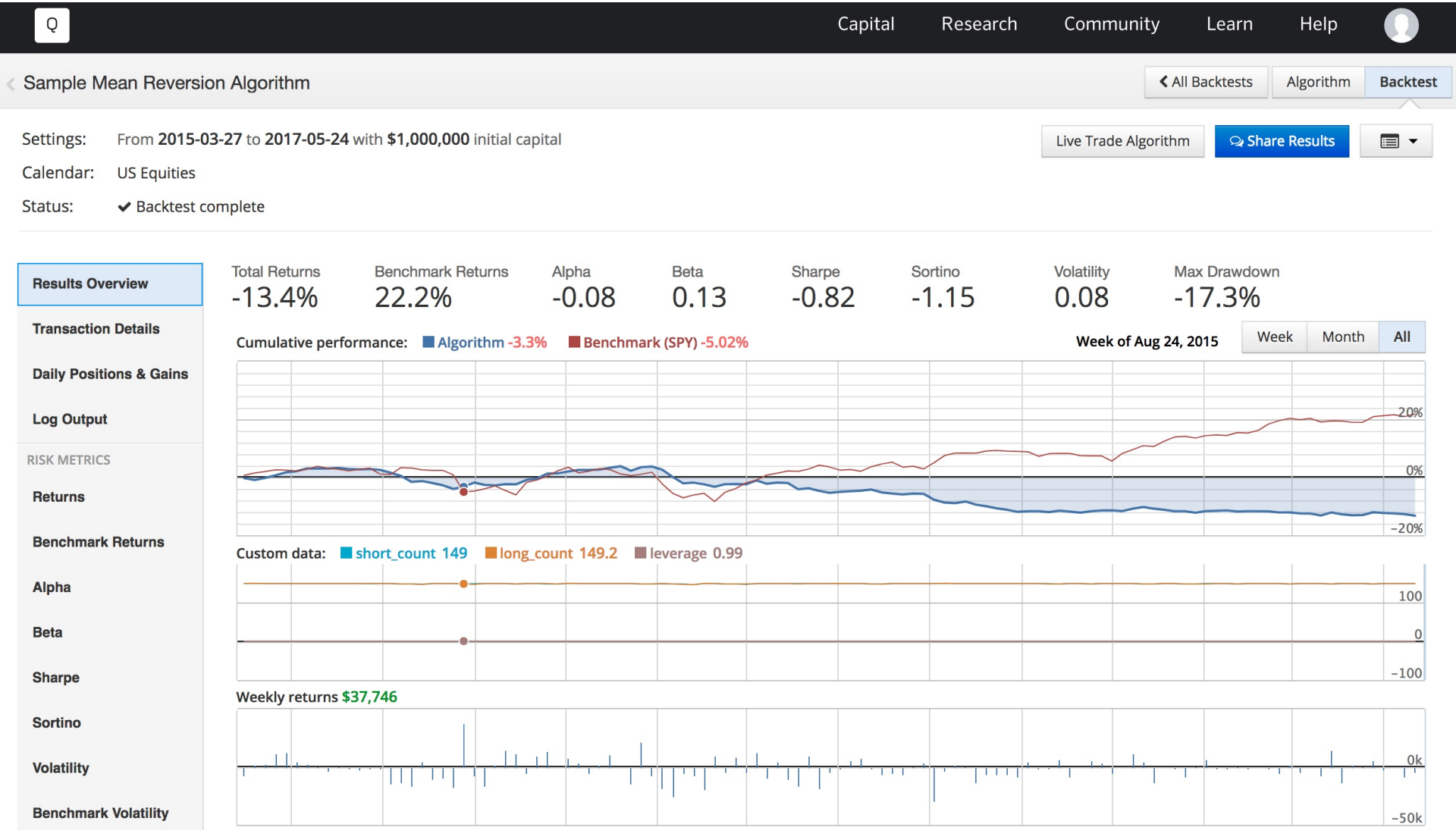
https://www.quantopian.com/

Quantopian
Sample Mean Reversion Algorithm



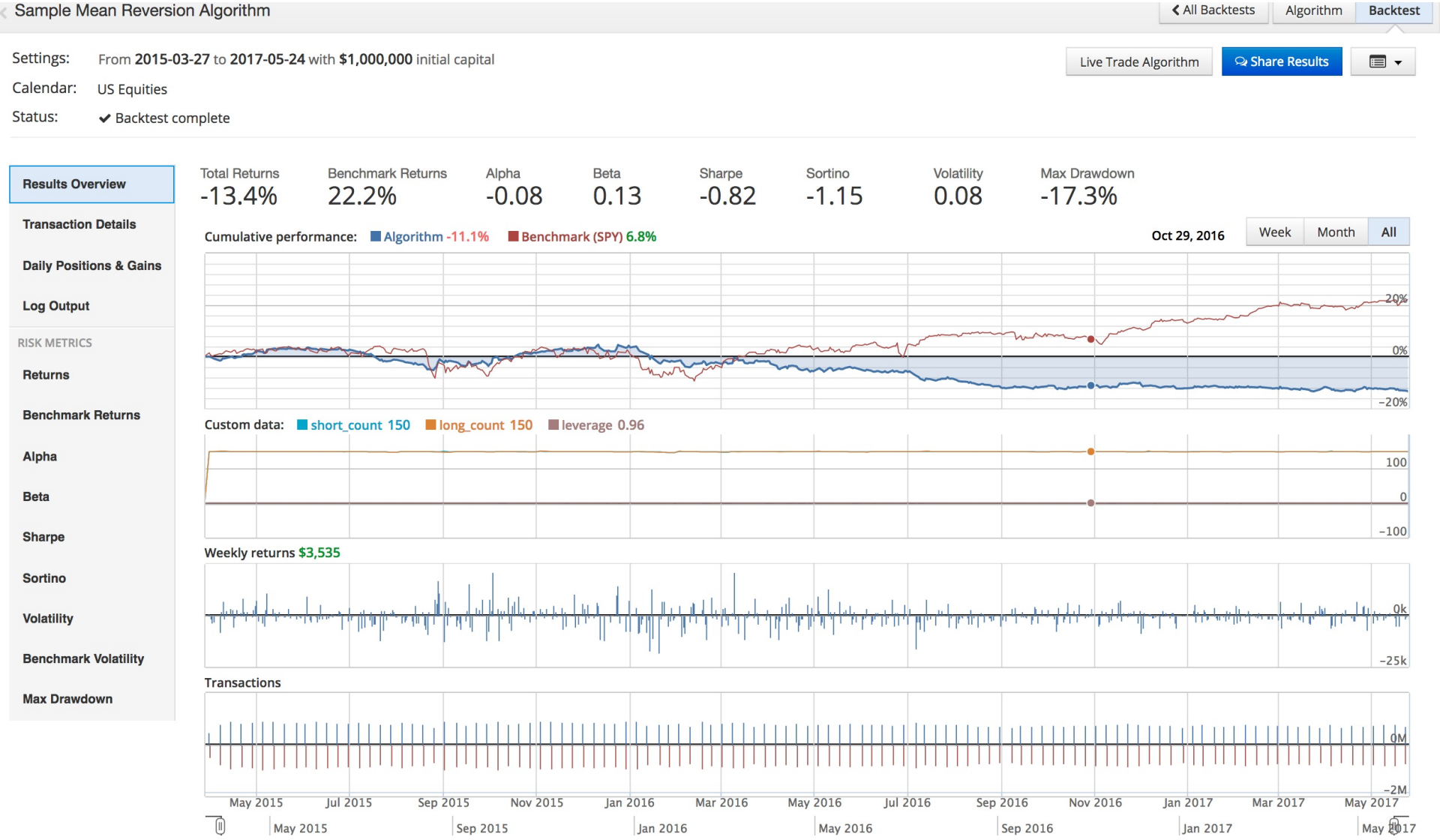
Source: https://www.quantopian.com/algorithms/59274d707875d1000d41937d/5927506f3cf7da6fec5951b6#backtest

Quantopian
Sample Mean Reversion Algorithm



Source: https://www.quantopian.com/algorithms/59274d707875d1000d41937d/5927506f3cf7da6fec5951b6#backtest

Quantopian
Sample Mean Reversion Algorithm



Source: https://www.quantopian.com/algorithms/59274d707875d1000d41937d/5927506f3cf7da6fec5951b6#backtest

Quantopian

Sample Mean Reversion Algorithm

Sample Mean Reversion Algorithm

All Backtests | Algorithm | Backtest

Settings: From **2015-03-27** to **2017-05-24** with **$1,000,000** initial capital

Calendar: US Equities

Status: ✔ Backtest complete

Live Trade Algorithm | Share Results

- Results Overview
- Transaction Details
- Daily Positions & Gains
- Log Output

RISK METRICS

- Returns
- Benchmark Returns
- Alpha
- Beta
- Sharpe
- Sortino
- Volatility
- Benchmark Volatility
- Max Drawdown

Transaction Details · Expand All · Collapse All

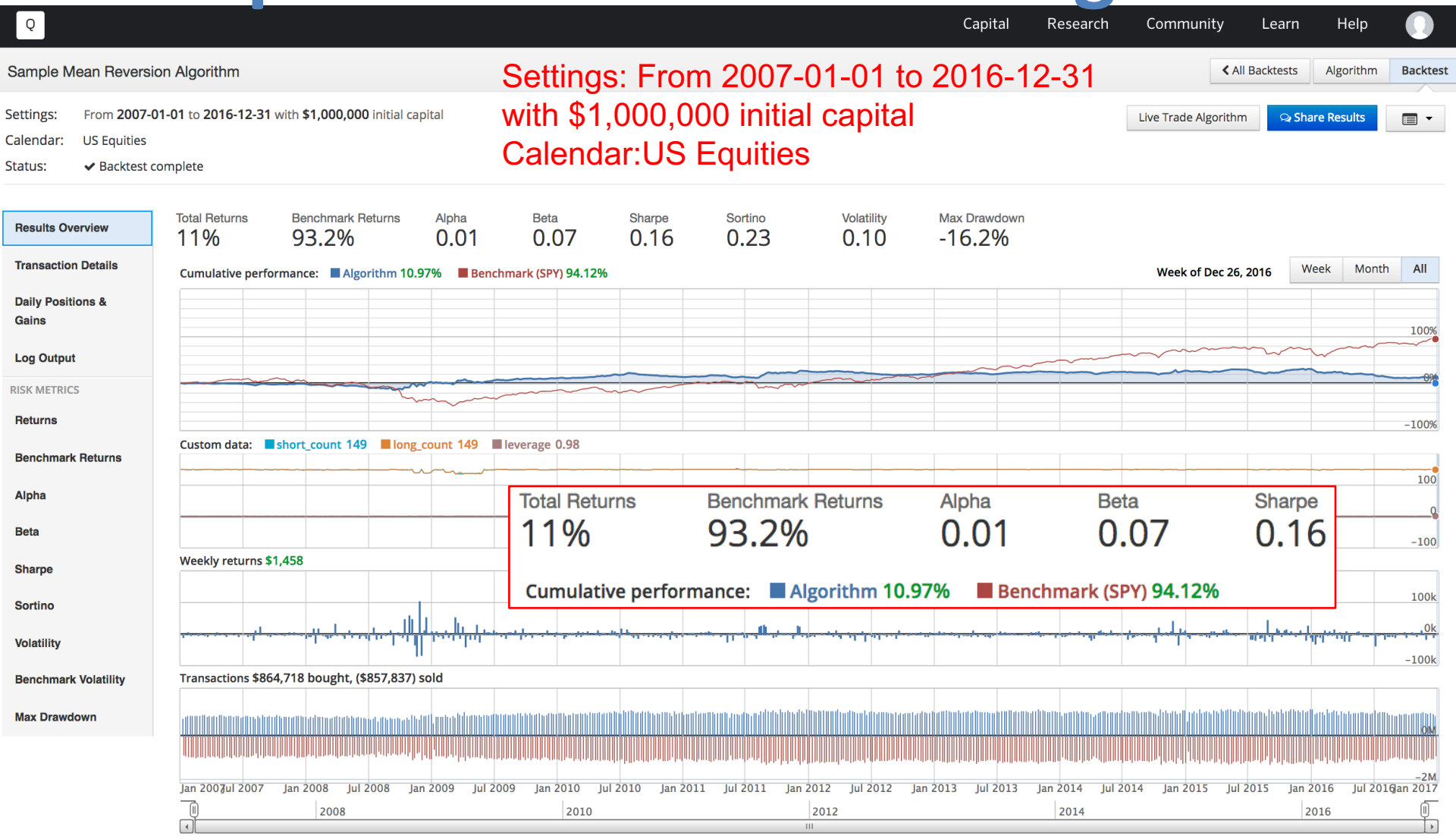
Group by day

Date	Asset	Transaction	Unit Price	Quantity	Position Value
2017-05-15 - 11:07 PM	PRAA	SELL	$37.32	-2	($74.65)
2017-05-15 - 11:07 PM	PRTA	SELL	$55.83	-31	($1,730.64)
2017-05-15 - 11:07 PM	PSTG	BUY	$11.68	44	$513.96
2017-05-15 - 11:07 PM	PTCT	SELL	$13.31	-10	($133.09)
2017-05-15 - 11:07 PM	QLYS	BUY	$43.50	10	$435.03
2017-05-15 - 11:07 PM	RGR	SELL	$64.07	-2	($128.14)
2017-05-15 - 11:07 PM	RRD	BUY	$13.30	62	$824.60
2017-05-15 - 11:07 PM	RXN	BUY	$23.45	9	$211.05
2017-05-15 - 11:07 PM	SUPN	SELL	$33.50	-12	($401.98)
2017-05-15 - 11:07 PM	TCO	SELL	$59.08	-7	($413.54)
2017-05-15 - 11:07 PM	TIVO	BUY	$17.15	2	$34.30
2017-05-15 - 11:07 PM	TLRD	BUY	$12.11	52	$629.77
2017-05-15 - 11:07 PM	TPC	SELL	$28.20	-5	($140.99)
2017-05-15 - 11:07 PM	TROX	SELL	$19.21	-60	($1,152.54)
2017-05-15 - 11:07 PM	TWNK	BUY	$15.69	17	$266.75

Source: https://www.quantopian.com/algorithms/59274d707875d1000d41937d/592756885310d86a415cb0e1#backtest

Quantopian

Sample Mean Reversion Algorithm



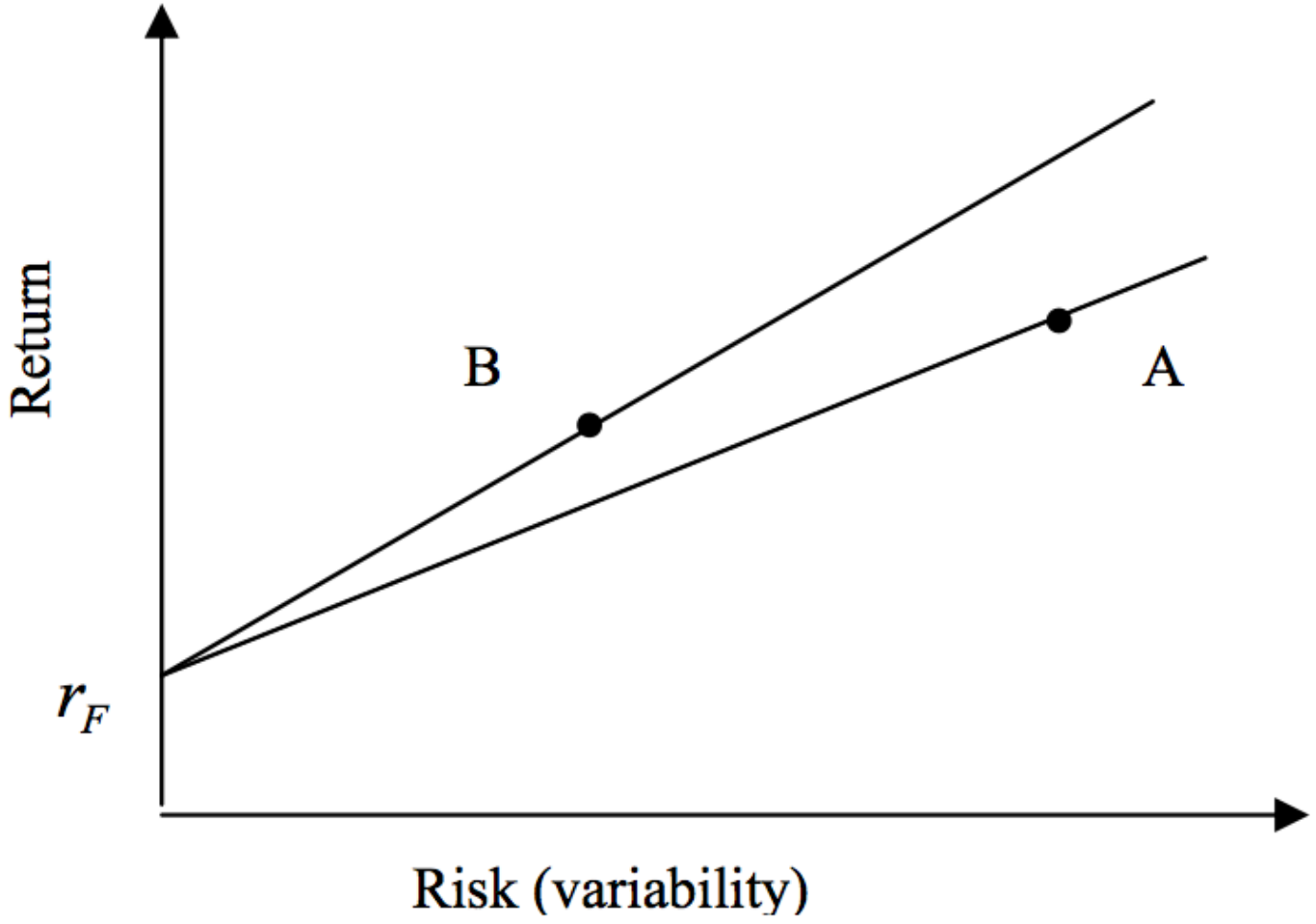
Source: https://www.quantopian.com/algorithms/59274d707875d1000d41937d/592756885310d86a415cb0e1#backtest

Quantopian
Sample Mean Reversion Algorithm

Total Returns	Benchmark Returns	Alpha	Beta	Sharpe
11%	93.2%	0.01	0.07	0.16

Source: https://www.quantopian.com/algorithms/59274d707875d1000d41937d/592756885310d86a415cb0e1#backtest

Risk and Return



Source: Bacon, Carl. "How sharp is the Sharpe-ratio?-Risk-adjusted Performance Measures." *Statpro White Paper* (2000).

Sharpe Ratio

$$\textbf{Sharpe Ratio} = \frac{Portofolio\ Return - Risk\ Free\ Return}{Portofolio\ Risk}$$

Source: Bacon, Carl. "How sharp is the Sharpe-ratio?-Risk-adjusted Performance Measures." *Statpro White Paper* (2000).

Sharpe Ratio

$$\textbf{Sharpe Ratio } SR = \frac{r_P - r_F}{\sigma_P}$$

Where

r_P = portfolio return

r_F = risk free rate

σ_P = portfolio risk (variability, standard deviation of return)

Source: Bacon, Carl. "How sharp is the Sharpe-ratio?-Risk-adjusted Performance Measures." *Statpro White Paper* (2000).

Sortino Ratio

$$\textbf{Sortino Ratio} = \frac{r_P - r_T}{\sigma_D}$$

Where

r_P = portfolio return

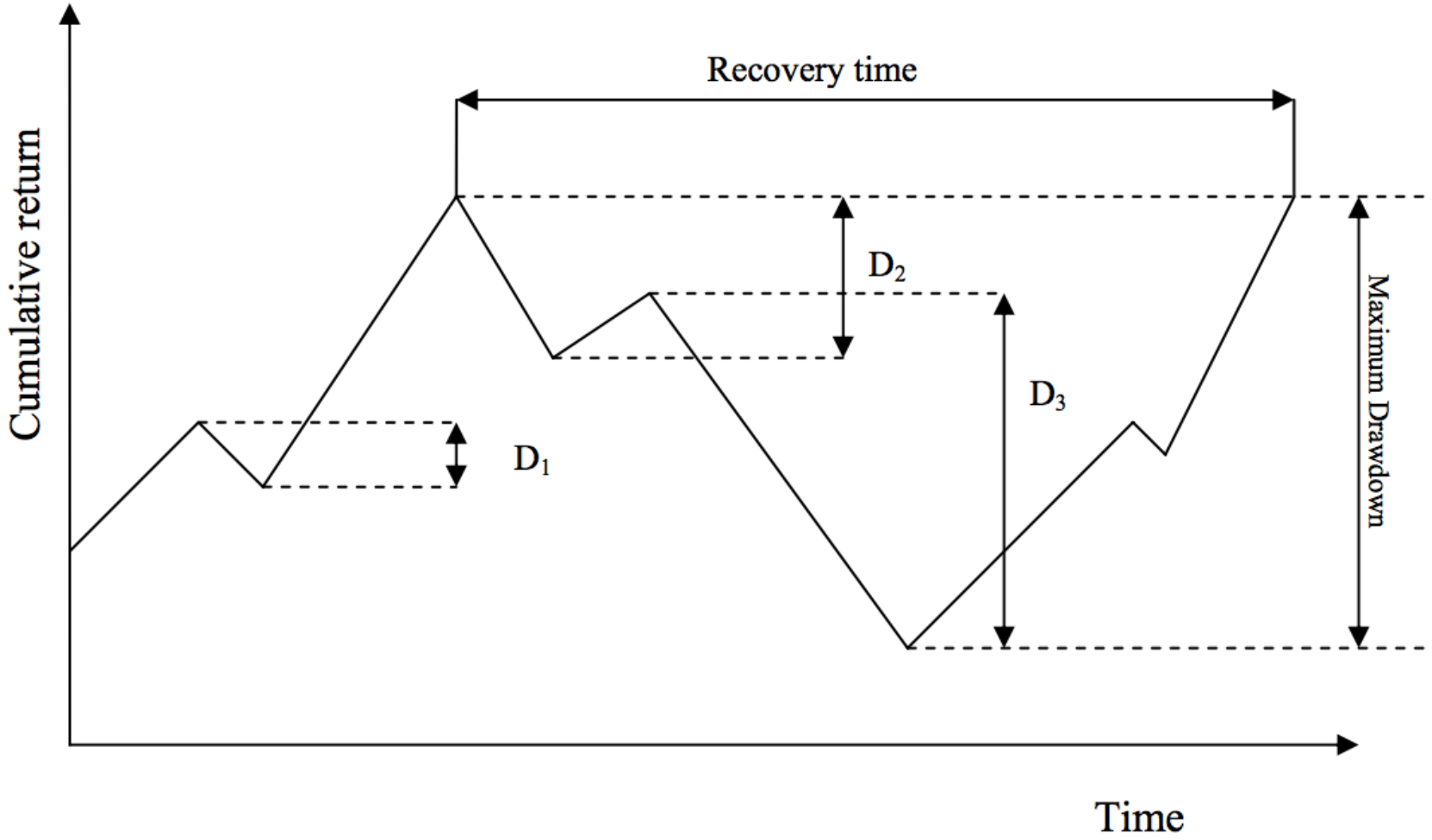
r_T = Minimum Target Return

σ_D = Downside Risk

$$\textbf{Downside Risk } \sigma_D = \sqrt{\sum_{i=1}^{n} \frac{\min[(r_i - r_T), 0]^2}{n}}$$

Source: Bacon, Carl. "How sharp is the Sharpe-ratio?-Risk-adjusted Performance Measures." *Statpro White Paper* (2000).

Max Drawdown



Source: Bacon, Carl. "How sharp is the Sharpe-ratio?-Risk-adjusted Performance Measures." *Statpro White Paper* (2000).

Quantopian

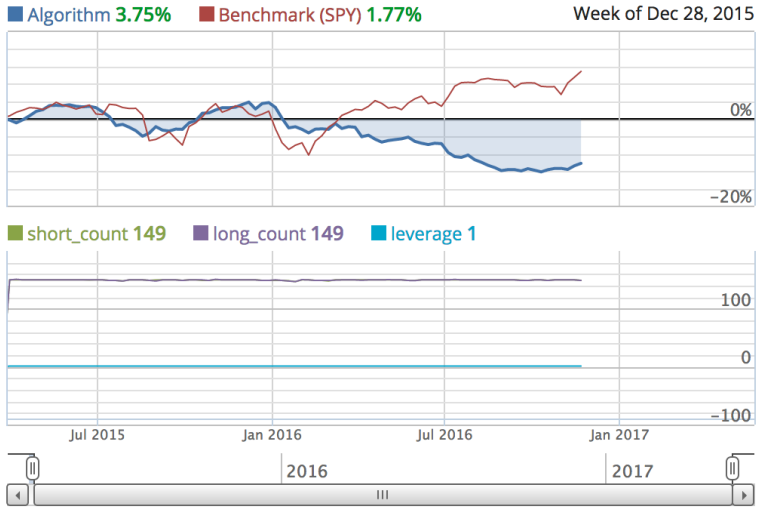
Sample Mean Reversion Algorithm

Save | Build Algorithm | Enter Contest | Collaborate | API Reference | Exit Fullscreen

```
"""
This is a sample mean-reversion algorithm on Quantopian for you to test and adapt.
This example uses a dynamic stock selector, pipeline, to select stocks to trade.
It orders stocks from the top 1% of the previous day's dollar-volume (liquid
stocks).

Algorithm investment thesis:
Top-performing stocks from last week will do worse this week, and vice-versa.

Every Monday, we rank high dollar-volume stocks based on their previous 5 day returns.
We long the bottom 10% of stocks with the WORST returns over the past 5 days.
We short the top 10% of stocks with the BEST returns over the past 5 days.

This type of algorithm may be used in live trading and in the Quantopian Open.
"""

# Import the libraries we will use here.
from quantopian.algorithm import attach_pipeline, pipeline_output
from quantopian.pipeline import Pipeline
from quantopian.pipeline.data.builtin import USEquityPricing
from quantopian.pipeline.factors import Returns
from quantopian.pipeline.filters.morningstar import Q1500US


def initialize(context):
    """
    Called once at the start of the program. Any one-time
    startup logic goes here.
    """
    # Define context variables that can be accessed in other methods of
    # the algorithm.
    context.long_leverage = 0.5
    context.short_leverage = -0.5
    context.returns_lookback = 5

    # Rebalance on the first trading day of each week at 11AM.
    schedule_function(rebalance,
                      date_rules.week_start(days_offset=0),
                      time_rules.market_open(hours=1, minutes=30))

    # Record tracking variables at the end of each day.
    schedule_function(record_vars,
                      date_rules.every_day()
```

Backtesting 77.6% Cancel

RETURNS	ALPHA	BETA	SHARPE	DRAWDOWN
-10%	-0.07	0.12	-0.72	-16.4%



Logs | Runtime Errors | More

```
2015-03-30 23:00 rebalance:146 INFO This week's longs: ARCB, ACXM, ARW, AVT, AXP,
BBY, CAMP, CDE, CMCS_A, CY, ATGE, S, HL, HMSY, HPQ, IDCC, BIIB, IDTI, IMGN, IONS,
JCP, KLAC, LRCX, MAT, MBI, MGM, MTG, REGN, RGLD, SMTC, SONC, TDS, TDW, TER, TRMB,
WDC, WERN, WOR, YRCW, ZBRA, CREE, RIG, GMCR, HST, BCRX, KNX, SSYS, FOXA, VECO, MYGN,
SNDK, NVAX, SCCO, ANDE, HUBG, CRR, NBIX, GWR, OCN, CIEN, TIVO, POWI, WAC, BRCM, AMKR,
NVDA, BRCD, LPNT, JNPR, VIAV, FNSR, ON, SGMO, CYH, ARNA, KERX, MDCO, ARRY, UTIW,
EXAS, SGMS, BTU, JOY, GME, LCI, BDSI, STX, CNX, APOL, ARRS, XPER, WLL, NRG, AGO,
ACAD, GNW, ALNY, DRH, HPY, IHS, CROX, ZIOP, SFLY, LDOS, ACHN, GSAT, AFSI, TWC, SMCI,
CZZ, DFT, CATM, SEM, PEB, PPC, FNGN, TSLA, SWFT, ZG, ZLTQ, GEVA, IMPV, HLSS, NSM,
PBYI, FRGI, WAGE, BLMN, RGLS, FLTX, ENTA, CSTM, PTCT, GOGO, AGIO, COMM, NMBL, RARE,
TRUE, ANET, TMST, ECR, KITE, MIK, SYF, MBLY, KEYS, VA, LC, QRVO
```

Source: https://www.quantopian.com/algorithms/59274d707875d1000d41937d/5927506f3cf7da6fec5951b6#backtest

Quantopian

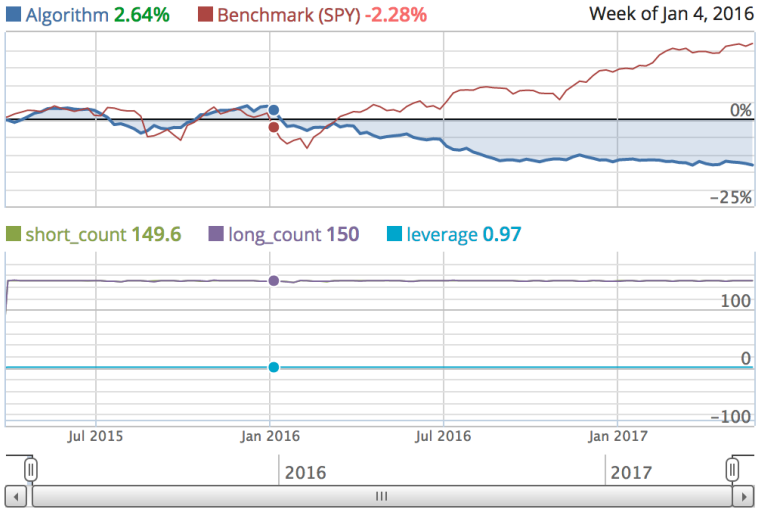
Sample Mean Reversion Algorithm

Save | Build Algorithm | Enter Contest | Collaborate | API Reference | Exit Fullscreen

```
"""
This is a sample mean-reversion algorithm on Quantopian for you to test and adapt.
This example uses a dynamic stock selector, pipeline, to select stocks to trade.
It orders stocks from the top 1% of the previous day's dollar-volume (liquid
stocks).

Algorithm investment thesis:
Top-performing stocks from last week will do worse this week, and vice-versa.

Every Monday, we rank high dollar-volume stocks based on their previous 5 day returns.
We long the bottom 10% of stocks with the WORST returns over the past 5 days.
We short the top 10% of stocks with the BEST returns over the past 5 days.

This type of algorithm may be used in live trading and in the Quantopian Open.
"""

# Import the libraries we will use here.
from quantopian.algorithm import attach_pipeline, pipeline_output
from quantopian.pipeline import Pipeline
from quantopian.pipeline.data.builtin import USEquityPricing
from quantopian.pipeline.factors import Returns
from quantopian.pipeline.filters.morningstar import Q1500US


def initialize(context):
    """
    Called once at the start of the program. Any one-time
    startup logic goes here.
    """
    # Define context variables that can be accessed in other methods of
    # the algorithm.
    context.long_leverage = 0.5
    context.short_leverage = -0.5
    context.returns_lookback = 5

    # Rebalance on the first trading day of each week at 11AM.
    schedule_function(rebalance,
                      date_rules.week_start(days_offset=0),
                      time_rules.market_open(hours=1, minutes=30))

    # Record tracking variables at the end of each day.
    schedule_function(record_vars,
```

03/27/2015 to 05/23/2017 | $ 1000000 | US Equities | Run Full Backtest

RETURNS	ALPHA	BETA	SHARPE	DRAWDOWN
-13.2%	-0.08	0.13	-0.81	-17.3%



Logs | Runtime Errors | More

```
2015-03-30 23:00 rebalance:146 INFO This week's longs: ARCB, ACXM, ARW, AVT, AXP,
BBY, CAMP, CDE, CMCS_A, CY, ATGE, S, HL, HMSY, HPQ, IDCC, BIIB, IDTI, IMGN, IONS,
JCP, KLAC, LRCX, MAT, MBI, MGM, MTG, REGN, RGLD, SMTC, SONC, TDS, TDW, TER, TRMB,
WDC, WERN, WOR, YRCW, ZBRA, CREE, RIG, GMCR, HST, BCRX, KNX, SSYS, FOXA, VECO, MYGN,
SNDK, NVAX, SCCO, ANDE, HUBG, CRR, NBIX, GWR, OCN, CIEN, TIVO, POWI, WAC, BRCM, AMKR,
NVDA, BRCD, LPNT, JNPR, VIAV, FNSR, ON, SGMO, CYH, ARNA, KERX, MDCO, ARRY, UTIW,
EXAS, SGMS, BTU, JOY, GME, LCI, BDSI, STX, CNX, APOL, ARRS, XPER, WLL, NRG, AGO,
ACAD, GNW, ALNY, DRH, HPY, IHS, CROX, ZIOP, SFLY, LDOS, ACHN, GSAT, AFSI, TWC, SMCI,
CZZ, DFT, CATM, SEM, PEB, PPC, FNGN, TSLA, SWFT, ZG, ZLTQ, GEVA, IMPV, HLSS, NSM,
PBYI, FRGI, WAGE, BLMN, RGLS, FLTX, ENTA, CSTM, PTCT, GOGO, AGIO, COMM, NMBL, RARE,
TRUE, ANET, TMST, ECR, KITE, MIK, SYF, MBLY, KEYS, VA, LC, QRVO
```

Source: https://www.quantopian.com/algorithms/59274d707875d1000d41937d/5927506f3cf7da6fec5951b6#backtest

Writing and Backtesting an Algorithm on Quantopian

Source: https://www.quantopian.com/tutorials/getting-started

What is a Trading Algorithm?

On Quantopian,
a trading algorithm
is a Python program
that defines two special functions:
initialize() and handle_data()

Source: https://www.quantopian.com/tutorials/getting-started

An example of an algorithm that allocates 100% of its portfolio in AAPL

```
def initialize(context):
    # Reference to AAPL
    context.aapl = sid(24)

def handle_data(context, data):
    # Position 100% of our portfolio to be long in AAPL
    order_target_percent(context.aapl, 1.00)
```

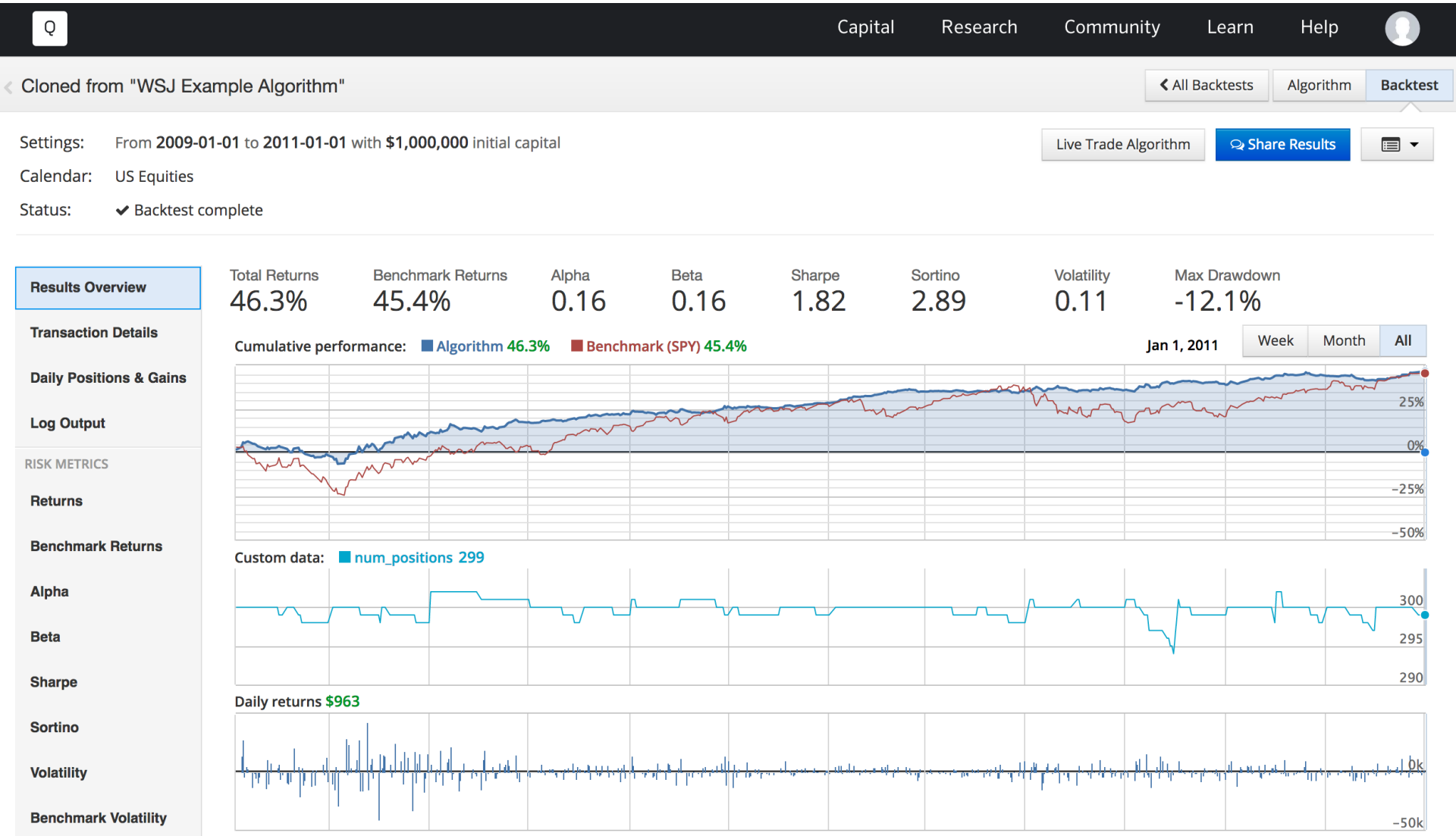
Source: https://www.quantopian.com/tutorials/getting-started

Moving Average

```
def initialize(context):
    context.security = symbol('AAPL')
    schedule_function(myfunc, date_rules.every_day(),  time_rules.market_open(minutes = 15))

def handle_data(context, data):
    MovingAvg1 = data[context.security].mavg(20)
    MovingAvg2 = data[context.security].mavg(60)

    current_positions = context.portfolio.positions[symbol('AAPL')].amount

    if (MovingAvg1 > MovingAvg2) and current_positions == 0:
        order_target_percent(context.security, 0.25)

    elif (MovingAvg1 < MovingAvg2) and current_positions != 0:
        order_target(context.security, 0)
```

Source: https://www.quantopian.com/tutorials/getting-started

Quantopian
WSJ Example Algorithm



Source: https://www.quantopian.com/algorithms/59274d707875d1000d41937d/5927506f3cf7da6fec5951b6#backtest

UQER

Source: https://uqer.io/home/

UQER

海量金融大数据

高质量的海量金融数据支撑，轻松实现大数据时代的交易策略

云端平台，高效研究，极速回测

稳定、安全、高可扩展的云平台，零门槛获得华尔街专业级别量化研究装备

Source: https://uqer.io/home/

UQER

模拟交易，赢取基金管理权

一键实盘模拟 ，云端托管，更有机会赢取500万实盘资金管理收益

Source: https://uqer.io/home/

JoinQuant

JoinQuant 聚宽 | 了解企业版

首页 | 策略擂台 HOT | 投资研究 | 我的策略 | 我的交易 | 数据 | 帮助 | 量化课堂 | 社区 | 登录 | 注册

十行代码，玩转聚宽

JOINQUANT

与7万+宽客一起

玩转量化投资

免费试用

立即登录

策略广场

赢率季胜季 模拟实盘	基于SVM的机器学习策略 策略回测	稳增高爆组合 模拟实盘
鏖戰先覺者	走得很慢的海龟	阴吹思婷

Source: https://www.joinquant.com/

JoinQuant

策略广场

赢率季胜季

模拟实盘


鏖戰先覺者

策略收益 基准收益

年化收益	最大回撤	初始资金
305.71%	13.31%	¥ 50000

已有540人订阅 免费订阅

基于SVM的机器学习策略

策略回测


走得很慢的海龟

策略收益 基准收益

年化收益	最大回撤	初始资金
10.05%	20.49%	¥ 1000000

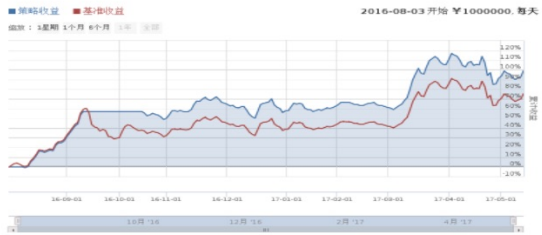
已有633人获取源码 获取源码

稳增高爆组合

模拟实盘


阴吹思婷

策略收益 基准收益



年化收益	最大回撤	初始资金
142.52%	15.88%	¥ 1000000

已有585人订阅 免费订阅

银行日内

模拟实盘


囚徒之爱

策略收益 基准收益

年化收益	最大回撤	初始资金
78.64%	4.51%	¥ 30000

【量化课堂】股指期货跨期套...

策略回测


JoinQuant量化课堂

策略收益 基准收益

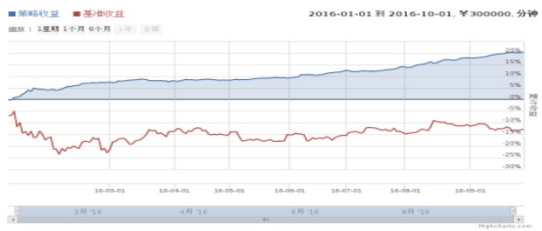
年化收益	最大回撤	初始资金
56.11%	17.13%	¥ 1000000

分级A轮动策略

策略回测


囚徒

策略收益 基准收益



年化收益	最大回撤	初始资金
18.58%	1.08%	¥ 300000

Source: https://www.joinquant.com/

RiceQuant

策略研究

历史回测

强大、易用的量化接口API，易于编写交易策略
免费提供10年+的日、分钟级历史数据以及400多项指标的财务数据
极速、精准的回测体验，快速开发和验证投资策略

Source: https://www.ricequant.com/

RiceQuant

策略研究

免费提供IPython Notebook研究平台以及强大的金融、数学等工具库
免费提供10年+的日、分钟级历史数据以及400多项指标的财务数据
灵活的文本编辑和绘图功能，提供无与伦比的交互式体验

Source: https://www.ricequant.com/

RiceQuant

历史回测

强大、易用的量化接口API，易于编写交易策略
免费提供10年+的日、分钟级历史数据以及400多项指标的财务数据
极速、精准的回测体验，快速开发和验证投资策略

Source: https://www.ricequant.com/

RiceQuant

实时模拟交易

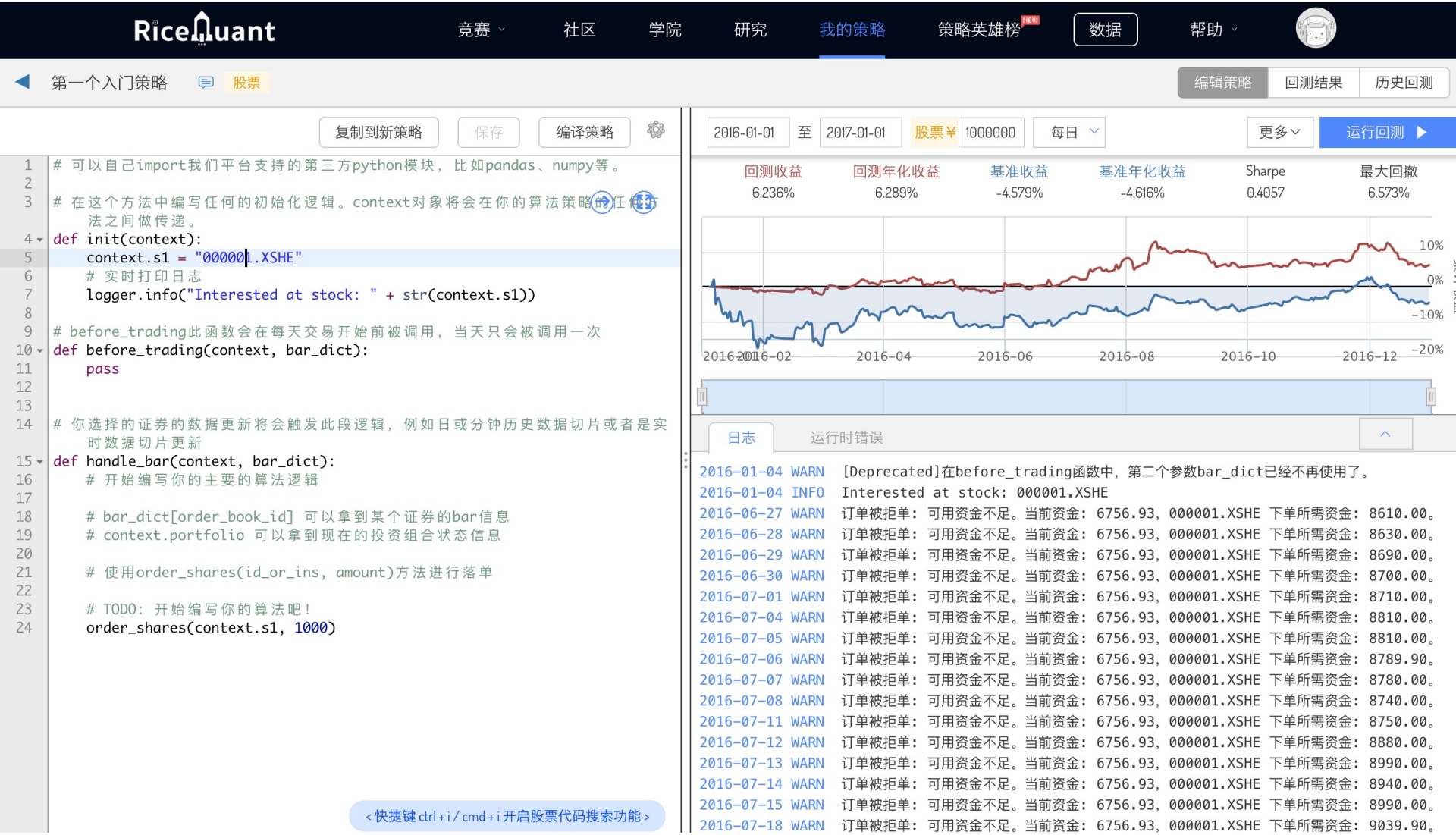
一键部署，云端永久运行

微秒级别实时数据推送计算

将会提供微信、邮件等交易信号推送

Source: https://www.ricequant.com/

RiceQuant



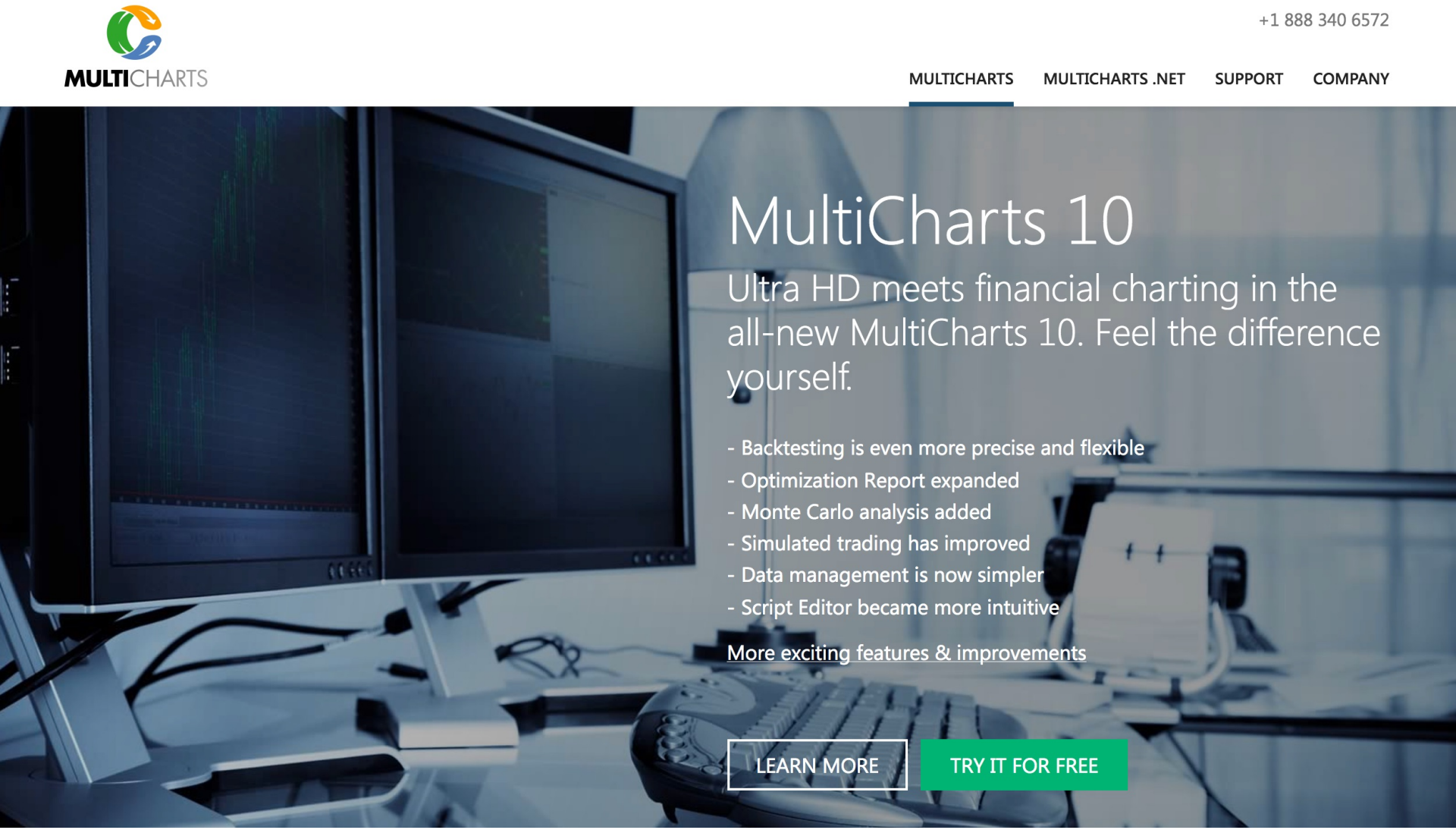

```
# 可以自己import我们平台支持的第三方python模块，比如pandas、numpy等。

# 在这个方法中编写任何的初始化逻辑。context对象将会在你的算法策略的任何方法之间做传递。
def init(context):
    context.s1 = "000001.XSHE"
    # 实时打印日志
    logger.info("Interested at stock: " + str(context.s1))

# before_trading此函数会在每天交易开始前被调用，当天只会被调用一次
def before_trading(context, bar_dict):
    pass


# 你选择的证券的数据更新将会触发此段逻辑，例如日或分钟历史数据切片或者是实时数据切片更新
def handle_bar(context, bar_dict):
    # 开始编写你的主要的算法逻辑

    # bar_dict[order_book_id] 可以拿到某个证券的bar信息
    # context.portfolio 可以拿到现在的投资组合状态信息

    # 使用order_shares(id_or_ins, amount)方法进行落单

    # TODO: 开始编写你的算法吧！
    order_shares(context.s1, 1000)
```

RiceQuant


```
# 在这个方法中编写任何的初始化逻辑。context对象将会在你的算法策略的任何方法之间做传递。
def init(context):
    #沪深300指数、中证500指数和国债指数
    context.stocks = ["000300.XSHG","000905.XSHG","000012.XSHG"]
# before_trading此函数会在每天交易开始前被调用，当天只会被调用一次
# 你选择的证券的数据更新将会触发此段逻辑，例如日或分钟历史数据切片或者是实时数据切片更新
def handle_bar(context, bar_dict):
    # 开始编写你的主要的算法逻辑
    hs300 = history_bars(context.stocks[0],20,"1d","close")
    zz500 = history_bars(context.stocks[1],20,"1d","close")
    hsIncrease = hs300[19] - hs300[0]
    zzIncrease = zz500[19] - zz500[0]
    p = context.portfolio.positions
    hsQuality = p[context.stocks[0]].quantity
    zzQuality = p[context.stocks[1]].quantity
    gzQuality = p[context.stocks[2]].quantity
    if hsIncrease < 0 and zzIncrease < 0:
        if hsQuality > 0:
            order_target_percent(context.stocks[0],0)
            logger.info("卖出沪深300")
        if zzQuality > 0:
            order_target_percent(context.stocks[1],0)
            logger.info("卖出中证500")
        if gzQuality <= 0.001:
            order_target_percent(context.stocks[2],1)
            logger.info("买入国债")
    elif hsIncrease < zzIncrease:
        if hsQuality > 0:
            order_target_percent(context.stocks[0],0)
            logger.info("卖出沪深300")
        if gzQuality > 0:
            order_target_percent(context.stocks[2],0)
            logger.info("卖出国债")
        if zzQuality <= 0.001:
```

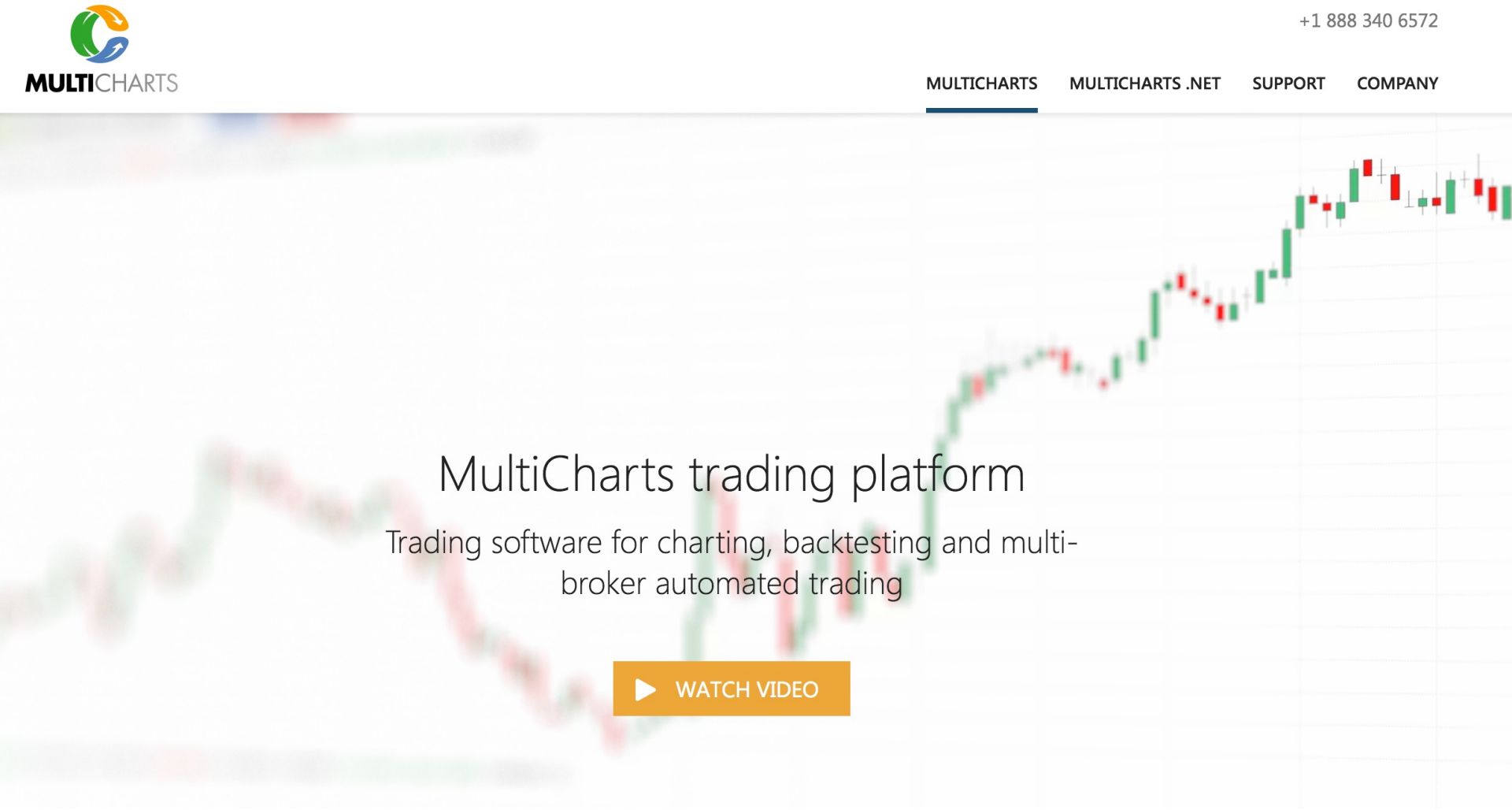
Source: https://www.ricequant.com/algorithm/523092

MultiCharts



https://www.multicharts.com/

MultiCharts



https://www.multicharts.com/

References

- Paolo Sironi (2016), “FinTech Innovation: From Robo-Advisors to Goal Based Investing and Gamification”, Wiley.
- Ernie Chan (2008), “Quantitative Trading: How to Build Your Own Algorithmic Trading Business”, Wiley
- Ernie Chan (2013), “Algorithmic Trading: Winning Strategies and Their Rationale”, Wiley
- Ernest P. Chan (2017), “Machine Trading: Deploying Computer Algorithms to Conquer the Markets”, Wiley
- Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly
- Yves Hilpisch (2015), Derivatives Analytics with Python: Data Analysis, Models, Simulation, Calibration and Hedging, Wiley
- Michael Heydt (2015) , Mastering Pandas for Finance, Packt Publishing
- Quantopian, https://www.quantopian.com/
- UQER, https://uqer.io/
- Joinquant, https://www.joinquant.com/
- Ricequant, https://www.ricequant.com/
- MultiCharts, https://www.multicharts.com/