AI for Investment Analysis

Portfolio Optimization and Algorithmic Trading

Min-Yuh Day
Associate Professor
Dept. of Information Management, Tamkang University

http://mail.tku.edu.tw/myday/
2020-06-10
# 課程大綱 (Syllabus)

<table>
<thead>
<tr>
<th>週次 (Week)</th>
<th>日期 (Date)</th>
<th>內容 (Subject/Topics)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2020/03/04</td>
<td>人工智慧投資分析課程介紹 (Course Orientation on AI for Investment Analysis)</td>
</tr>
<tr>
<td>2</td>
<td>2020/03/11</td>
<td>AI 金融科技: 金融服務創新應用 (AI in FinTech: Financial Services Innovation and Application)</td>
</tr>
<tr>
<td>3</td>
<td>2020/03/18</td>
<td>機器人理財顧問與AI交談機器人 (Robo-Advisors and AI Chatbots)</td>
</tr>
<tr>
<td>4</td>
<td>2020/03/25</td>
<td>投資心理學與行為財務學 (Investing Psychology and Behavioral Finance)</td>
</tr>
<tr>
<td>5</td>
<td>2020/04/01</td>
<td>財務金融事件研究法 (Event Studies in Finance)</td>
</tr>
<tr>
<td>6</td>
<td>2020/04/08</td>
<td>人工智慧投資分析個案研究 I (Case Study on AI for Investment Analysis I)</td>
</tr>
<tr>
<td>週次 (Week)</td>
<td>日期 (Date)</td>
<td>內容 (Subject/Topics)</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>7 2020/04/15</td>
<td>2020/04/15</td>
<td>Python AI投資分析基礎&lt;br&gt;(Foundations of AI Investment Analysis in Python)</td>
</tr>
<tr>
<td>8 2020/04/22</td>
<td>2020/04/22</td>
<td>Python Pandas 量化投資分析&lt;br&gt;(Quantitative Investing with Pandas in Python)</td>
</tr>
<tr>
<td>9 2020/04/29</td>
<td>2020/04/29</td>
<td>期中報告 (Midterm Project Report)</td>
</tr>
<tr>
<td>10 2020/05/06</td>
<td>2020/05/06</td>
<td>Python Scikit-Learn 機器學習投資分析&lt;br&gt;(Machine Learning for Investment Analysis with Scikit-Learn in Python)</td>
</tr>
<tr>
<td>11 2020/05/13</td>
<td>2020/05/13</td>
<td>TensorFlow 深度學習投資分析 I&lt;br&gt;(Deep Learning for Investment Analysis with TensorFlow I)</td>
</tr>
<tr>
<td>12 2020/05/20</td>
<td>2020/05/20</td>
<td>TensorFlow 深度學習投資分析 II&lt;br&gt;(Deep Learning for Investment Analysis with TensorFlow II)</td>
</tr>
</tbody>
</table>
課程大綱 (Syllabus)

週次 (Week)  日期 (Date)  內容 (Subject/Topics)
13 2020/05/27  人工智慧投資分析個案研究 II
               (Case Study on Artificial Intelligence for Investment Analysis II)
14 2020/06/03  TensorFlow 深度學習投資分析 III
               (Deep Learning for Investment Analysis with TensorFlow III)
15 2020/06/10  投資組合最佳化與程式交易
               (Portfolio Optimization and Algorithmic Trading)
16 2020/06/17  期末報告 I (Final Project Presentation I)
17 2020/06/24  期末報告 II (Final Project Presentation II)
18 2020/07/01  教師彈性補充教學
Portfolio Optimization and Algorithmic Trading
Outline

• Portfolio Optimization

• Algorithmic Trading
Portfolio Optimization
Algorithmic Trading
Yves Hilpisch (2015), Derivatives Analytics with Python: Data Analysis, Models, Simulation, Calibration and Hedging, Wiley
Algorithmic Trading

Historical Finance Market Data

Live Finance Market Data

Computer Program

Broker API

Broker’s Server

Backtest Results

Order

Order Status

Order

Order Status

FinTech Innovation: From Robo-Advisors to Goal Based Investing and Gamification, Paolo Sironi, Wiley, 2016
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
5 FinTech: Investment Management

Source: http://www3.weforum.org/docs/WEF_The_future__of_financial_services.pdf
FinTech: Investment Management
Empowered Investors
Process Externalization

Source: https://www.stockfeel.com.tw/2015年世界經濟論壇－未來的金融服務/
FinTech: Market Provisioning
Smarter, Faster Machines
New Market Platforms

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

Quantopian

PyThalesians

Zipline

DX Analytics

PyAlgoTrade

QuantLib

StatsModels

Statistics in Python

scikit-learn

scikit-image

image processing in python

matplotlib

pandas

\[ y_{it} = \beta x_{it} + \mu_i + \epsilon_{it} \]

SciPy

NumPy

SymPy

IPython

Python

Jupyter

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/
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/
Become an Expert in Quant Finance

Quantopian provides free education, data, and tools so anyone can pursue quantitative finance. Select members license their algorithms and share in the profits.

Start Learning

Community Achievements

All numbers are as of June 1, 2018

https://www.quantopian.com/
Sign up for Quantopian

Sign up for Quantopian

Research and Develop Your Investment Ideas

First name

Last name

Email address

Create a password

Get started

I accept the Terms Of Use and Privacy Policy.

https://www.quantopian.com/users/sign_up
Sample Mean Reversion Algorithm

Settings: From 2015-03-27 to 2017-05-24 with $1,000,000 initial capital
Calendar: US Equities
Status: Backtest complete

<table>
<thead>
<tr>
<th>Total Returns</th>
<th>Benchmark Returns</th>
<th>Alpha</th>
<th>Beta</th>
<th>Sharpe</th>
<th>Sortino</th>
<th>Volatility</th>
<th>Max Drawdown</th>
</tr>
</thead>
<tbody>
<tr>
<td>-13.4%</td>
<td>22.2%</td>
<td>-0.08</td>
<td>0.13</td>
<td>-0.82</td>
<td>-1.15</td>
<td>0.08</td>
<td>-17.3%</td>
</tr>
</tbody>
</table>

Cumulative performance: Algorithm -13.24% Benchmark (SPY) 21.9%

Source: https://www.quantopian.com/algorithms/59274d707875d1000d41937d/5927506f3cf7da6fec5951b6#backtest
Quantopian Sample Mean Reversion Algorithm

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

Calendar: US Equities
Status: Backtest complete

Total Returns: -13.4%
Benchmark Returns: 22.2%
Alpha: -0.08
Beta: 0.13
Sharpe: -0.82
Sortino: -1.15
Volatility: 0.08
Max Drawdown: -17.3%

Cumulative performance:
- Algorithm: -3.3%
- Benchmark (SPY): -5.02%

Weekly returns: $37,746

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

Sample Mean Reversion Algorithm

Source: https://www.quantopian.com/algorithms/59274d707875d1000d41937d/5927506f3cf7da6fec5951b6#backtest
## Sample Mean Reversion Algorithm

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

**Calendar:** US Equities

**Status:** Backtest complete

### Transaction Details

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

Settings: From 2007-01-01 to 2016-12-31 with $1,000,000 initial capital
Calendar: US Equities

Cumulative performance: **Algorithm 10.97%**  **Benchmark (SPY) 94.12%**

Total Returns 93.2%
Alpha 0.01
Beta 0.07
Sharpe 0.16

Source: https://www.quantopian.com/algorithms/59274d707875d1000d41937d/592756885310d86a415cb0e1#backtest
Quantopian
Sample Mean Reversion Algorithm

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

Cumulative performance:  Algorithm 10.97%  Benchmark (SPY) 94.12%

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

Sharpe Ratio

\[
\text{Sharpe Ratio} = \frac{\text{Portfolio Return} - \text{Risk Free Return}}{\text{Portfolio Risk}}
\]

Sharpe Ratio

\[ SR = \frac{r_P - r_F}{\sigma_P} \]

Where
- \( r_P \) = portfolio return
- \( r_F \) = risk free rate
- \( \sigma_P \) = portfolio risk (variability, standard deviation of return)

**Sortino Ratio**

\[
\text{Sortino Ratio} = \frac{r_P - r_T}{\sigma_D}
\]

Where

- \(r_P\) = portfolio return
- \(r_T\) = Minimum Target Return
- \(\sigma_D\) = Downside Risk

**Downside Risk**

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

Max Drawdown

Quantopian
Sample Mean Reversion Algorithm

This is a sample mean-reversion algorithm on Quantopian for you to test and adopt.

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.

---

```python
from quantopian.algorithm import attach_pipeline, pipeline_output
from quantopian.pipeline import Pipeline
from quantopian.pipeline.data import USEquityPricing
from quantopian.pipeline.factors import Returns
from quantopian.pipeline.filters.morningstar import Q1500US

def initialize(context):
    # Import the libraries we will use here.
    from quantopian.pipeline import Pipeline
    from quantopian.pipeline.data import USEquityPricing
    from quantopian.pipeline.factors import Returns
    from quantopian.pipeline.filters.morningstar import Q1500US

# Define context variables that can be accessed in other methods of
# the algorithm.
context.long_leverage = 0.5
context.short_leverage = -0.5
context.return_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())
```

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

Sample Mean Reversion Algorithm

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.

```python

# 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.whole_leverage = 0.5

    # 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())
```

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()`.
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
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)
Investment Science: Portfolio Optimization

Three important portfolios on the Efficient Frontier

Source: Tucker Balch (2012), Investment Science: Portfolio Optimization, https://www.youtube.com/watch?v=5qbMhXXq0vI
Portfolio Optimization
Efficient Frontier

Source: Tucker Balch (2012), Investment Science: Portfolio Optimization, https://www.youtube.com/watch?v=5qbMhXXq0vI
Portfolio Optimization and Algorithmic Trading

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

https://tinyurl.com/imtkupython101
Portfolio Optimization

Efficient Frontier

Portfolio Optimization with Individual Stocks

- Efficient frontier
- Maximum Sharpe ratio
- Minimum volatility

Graph showing annualised returns vs. annualised volatility for AAPL, GOOG, FB, and AMZN.
优矿，您的私人量化平台
打破金融量化的壁垒，为量化研究者提供媲美华尔街专业机构的研究装备

了解专业版>>>
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/
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
RiceQuant

一个为你量身打造的量化策略平台

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+的分、日线级历史数据以及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/
策略研究

免费提供Python 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/
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/
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
# 可以自己import我们平台支持的第三方python模块，比如pandas、nump等。

```python
def init(context):
    context.sl = "00001.XSHE"
    # 实时打印日志
    logger.info("Interested at stock: " + str(context.sl))
```

# 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间做传递。

```python
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

```python
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)方法进行下单
    # T00D：开始编写你的算法吧！
    order_shares(context.sl, 1000)
```

Source: https://www.ricequant.com/algorithm/523081
```python
# 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
custom.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(custom.stocks[0], 20, "1d", "close")
        zz500 = history.bars(custom.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
        if hsIncrease < 0 and zzIncrease < 0:
            if hsQuality > 0:
                order_target_percent(custom.stocks[0], 0)
                logger.info("卖出沪深300")
            if zzQuality > 0:
                order_target_percent(custom.stocks[1], 0)
                logger.info("卖出中证500")
            if zzQuality <= 0.001:
                order_target_percent(custom.stocks[2], 0)
                logger.info("买入国债")
        elif hsIncrease < zzIncrease:
            if hsQuality > 0:
                order_target_percent(custom.stocks[0], 0)
                logger.info("卖出沪深300")
            if zzQuality > 0:
                order_target_percent(custom.stocks[2], 0)
                logger.info("卖出国债")
            if zzQuality <= 0.001:
                order_target_percent(custom.stocks[1], 0)
                logger.info("卖出中证500")
```

Source: [https://www.ricequant.com/algorithm/523092](https://www.ricequant.com/algorithm/523092)
MultiCharts 12
Advanced market analysis features for expert traders

- Native chart type for Time Price Opportunity analysis
- Matrix Optimization for the strategy re-optimization
- Evaluate Strategy Robustness with greater ease
- More Cryptocurrency data providers are now available
- Local order emulation has been enhanced
- New Kase Bar custom resolution plugin has been added
- Completely new optimization GUI and extended Report metrics
- Flush Cached Data to Database manually
- You can now view optimization reports on-the-go

More exciting features & improvements

https://www.multicharts.com/
Python in Google Colab (Python101)

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

Portfolio Optimization and Algorithmic Trading

```python
1  ! pip install pandas_datareader
2  import pandas as pd
3  import pandas_datareader.data as web
4  import matplotlib.pyplot as plt
5  import seaborn as sns
6  import datetime as dt
7  %matplotlib inline
8  
9  # Read Stock Data from Yahoo Finance
10  end = dt.datetime.now()
11  # start = dt.datetime(end.year-2, end.month, end.day)
12  start = dt.datetime(2010, 1, 1)
13  df = web.DataReader("AAPL", 'yahoo', start, end)
14  df.to_csv('AAPL.csv')
15  # df = pd.read_csv('AAPL.csv')
16  print(df.head())
17  print(df.tail())
18  print(df.describe())
19  
20  df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
21  plt.figure(figsize=(12, 9))
22  top = plt.subplot2grid((12, 9), (0, 0), rowspan=10, colspan=9)
23  bottom = plt.subplot2grid((12, 9), (10, 0), rowspan=2, colspan=9)
24  top.plot(df.index, df['Adj Close'], color='blue')  # df.index gives the dates
25  bottom.bar(df.index, df['Volume'])
26  
27  # set the labels
28  top.axes.get_xaxis().set_visible(False)
29  top.set_title('AAPL')
30  top.set_ylabel('Adj Close')
31  bottom.set_ylabel('Volume')
32  
33  plt.figure(figsize=(12, 9))
```

https://tinyurl.com/imtkupython101
Python in Google Colab (Python101)

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

![Python code snippet]

```python
# 1pip install plotly
# 3 import plotly.graph_objects as go
4
# 5 import pandas as pd
6 from datetime import datetime
7 df = pd.read_csv('AAPL.csv')
8 fig = go.Figure(data=[go.Candlestick(x=df['Date'],
9   open=df['Open'],
10   high=df['High'],
11   low=df['Low'],
12   close=df['Close']))
13
14 fig.show()
```

---

**Requirement already satisfied:** plotly in /usr/local/lib/python3.6/dist-packages (4.4.1)
**Requirement already satisfied:** retrying>=1.3.3 in /usr/local/lib/python3.6/dist-packages (from plotly) (1.3.3)
**Requirement already satisfied:** six in /usr/local/lib/python3.6/dist-packages (from plotly) (1.12.0)

---

https://tinyurl.com/imtkupython101
Python in Google Colab (Python101)

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

https://tinyurl.com/imtkupython101
Annualised Return: 0.18

<table>
<thead>
<tr>
<th>Asset</th>
<th>AAPL</th>
<th>AMZN</th>
<th>FB</th>
<th>GOOGL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocation</td>
<td>44.67</td>
<td>29.05</td>
<td>26.28</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Minimum Volatility Portfolio Allocation

Annualised Return: 0.22
Annualised Volatility: 0.16

<table>
<thead>
<tr>
<th>Asset</th>
<th>AAPL</th>
<th>AMZN</th>
<th>FB</th>
<th>GOOGL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocation</td>
<td>34.02</td>
<td>0.73</td>
<td>6.98</td>
<td>58.26</td>
</tr>
</tbody>
</table>

Calculated Portfolio Optimization based on Efficient Frontier
Annualised Return: 0.22
Annualised Volatility: 0.16

<table>
<thead>
<tr>
<th></th>
<th>AAPL</th>
<th>AMZN</th>
<th>FB</th>
<th>GOOGL</th>
</tr>
</thead>
<tbody>
<tr>
<td>allocation</td>
<td>34.02</td>
<td>0.73</td>
<td>6.98</td>
<td>58.26</td>
</tr>
</tbody>
</table>

Individual Stock Returns and Volatility

AAPL: annualised return 0.28, annualised volatility: 0.21
AMZN: annualised return 0.34, annualised volatility: 0.25
FB: annualised return 0.3, annualised volatility: 0.23
GOOGL: annualised return 0.18, annualised volatility: 0.18

Portfolio Optimization with Individual Stocks

https://tinyurl.com/imtkuppython101
Summary

• Portfolio Optimization

• Algorithmic Trading
References

• Michael Heydt (2015) , Mastering Pandas for Finance, Packt Publishing
• Tucker Balch (2012), Investment Science: Portfolio Optimization, https://www.youtube.com/watch?v=5qbMhXXq0vl
• Quantopian, https://www.quantopian.com/
• Zipline, https://github.com/quantopian/zipline
• Pyfolio, https://github.com/quantopian/pyfolio
• UQER, https://uqer.io/
• Joinquant, https://www.joinquant.com/
• Ricequant, https://www.ricequant.com/
• MultiCharts, https://www.multicharts.com/