

# Economics 136

## ... financial markets and modelling

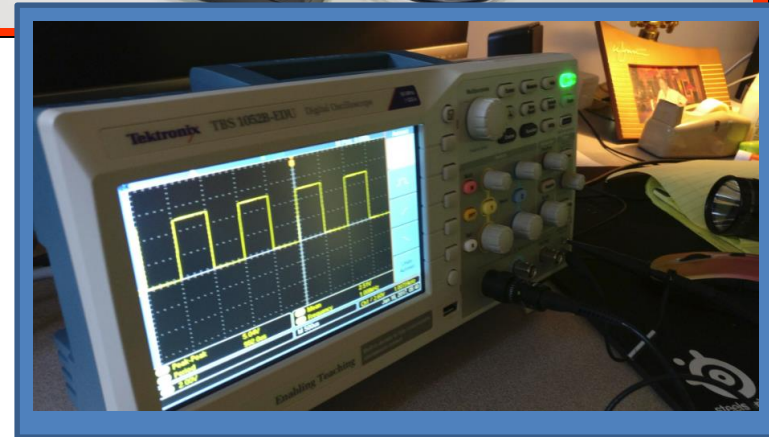
This is a formal financial modelling course teaching both traditional and experimental methods for modeling stocks, options, and futures, with an emphasis on trading them for profitability.

Most applied modeling is done in Python/Numpy.



# Your instructor

- **Prof. Gary R. Evans**
- Background/interests in macroeconomics, policy, finance, and entrepreneurship
- Trades markets heavily, manages portfolios (more used-to than still does)
- Believes that poker and competitive games like golf enhance trading skills and is fascinated by waves
- At Mudd since 1981
- Contact info:
  - [garyrevans@gmail.com](mailto:garyrevans@gmail.com)
- ~~Twitter trading information~~
  - ~~PITraders~~



<http://palmislandtraders.com/mudd/hmcgre.html>

## Today review ...

- The course outline and calendar
  - What we will definitely do
  - What we might do
  - What we won't do
- Requirements for being in this class ...
- Working in teams on projects
- Trading during class .. how that will work
- Policy about attendance

# Primary course objectives

- Make students comfortable in the exotic world of derivatives and valuation
- Encourage students to develop a professional bearing in all that you do.
- Provide both a quantitative and qualitative basis for analysis – look at some interesting models
- Encourage students to think of career opportunities in finance (hedge funds, analysts, etc.)
- Encourage students to use your imagination in profitable ways.






# Reading material



Text (**optional** and **reference** only):

*Options, Futures, and Other Derivatives, 10<sup>th</sup> edition*, by John C. Hull, Pearson/Prentice Hall, ISBN 013447208X or any earlier edition used. **[Do not buy until this is discussed in lecture, and do not buy new].**

... but mostly, as you will see, we will use online sources plus my Jupyter Notebooks and other lectures.

Section	Topics
1st half	<b>1. Setup and Fundamentals</b>  1.1 Setting up Python, Numpy, Pandas, and Jupyter 1.2 Compounding, discounting, continuous growth, log growth 1.3 Extracting data from IEX with Python
	<b>2. Theoretical Frameworks</b>  2.1 Stock and option foundational assumptions 2.2 Classical proxies for risk and alpha calculation 2.3 Monte Carlo simulations 2.4 Monte Carlo simulations with Poisson jumps 2.5 Probability estimators and range estimators 2.6 Roll_the_Dice (conceptual Python model)  2.7 Gaussian to log-linear transformations
	<b>3. Classical option modeling</b>  3.1 Black-Scholes-Merton (classical model and applications) 3.2 Python-modified BSM model 3.3 The Greeks (classical)
	<b>4. Let's Trade!</b>  4.1 Elementary spread arbitrage 4.2 Basic options strategies 4.3 IB elementary algorithms and API design issues 4.4 Anomaly trading  4.5 Momentum and DITM calls
	<b>Important dates!</b>  First online 20-minute quiz: Thursday, February 13. <b>Mid-term examination on Thursday, March 12!</b> 

 problematic may not happen Note these dates!!

2nd half

## 5. Advanced stock and options trading strategies

- 5.1 Strangles and earnings strangles
- 5.2 Aruba covered calls model
- 5.3 Credit and Debit spreads
- 5.4 Cash-secured option writing
- 5.5 Mean reversion and momentum trading



problematic

may not happen

## 6. Advanced modern options modeling (in Python)

- 6.1 Core Taboga (brute force) models
- 6.2 Adding drift and dividend estimations to the Taboga model
- 6.3 Converting Taboga to American expiries (non-European)
- 6.2 Brute force log-normal integration model
- 6.3 Core option strategy pricing
- 6.4 The Greeks (sensitivity analysis)



## 7. Futures Trading

- 7.1 Futures trading basics
- 7.2 Contango and backwardation trades
- 7.3 Indexes, interest rates, exchange rates, natural gas



## 8. Esoteric and experimental models

- 8.1 Correlation and cointegration models
- 8.2 Financial wave modeling with Fast Fourier Transforms
- 8.3 Propagation brute force models
- 8.4 Bond ETF trading



## Important dates!

Second online 20-minute quiz: Tuesday, April 7.

**Final examination, online and essay on Thursday, April 30!**

Note these dates!!

# What we won't do ...

... although I would like to someday or wish we had time:

- Machine-learning applications in finance
- ~~Direct hard-core algo~~
- ~~Spread arbitrage etc.~~
- ~~Bond and fixed income trading~~
- VIX trading
- Traditional structural analysis

• We do these now!!  
... at least get a taste of it.

• Mining and Crypto???  
... not any more ...

... ever!!

- Technical analysis
- Bollinger bands etc.
- Candlestick analysis etc.

• We actually look at a version of this as an example of mean-reversion.



# Background prep requirements ...

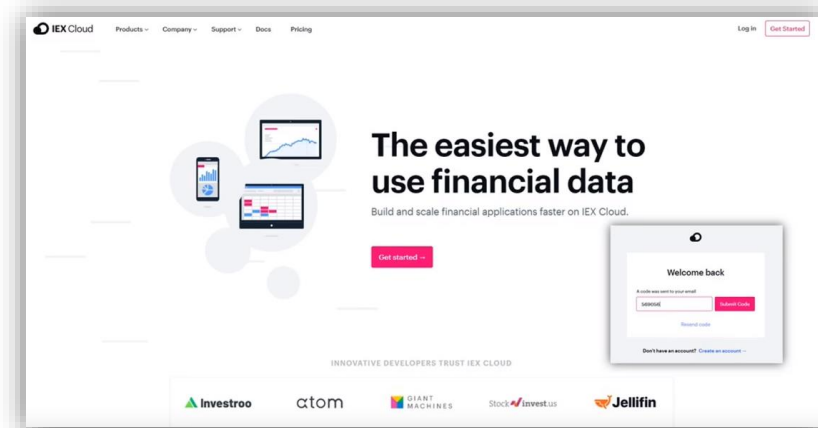
1. The Anaconda distribution of Python 3.X is **highly** recommended. Because in addition to Python 3.X you **are required** to have access to
  - a) Numpy
  - b) Pandas
  - c) asyncio
  - d) Jupyter Notebook
  - d) Seaborn
2. You must have a free data subscription to [iexcloud.io](https://iexcloud.io)
3. You may use whatever editor that you want, but your IDE must be capable of plotting (Sublime Text, PyCharm or VSCode is recommended).
4. You **must** have Jupyter Notebook up and running.

**All of this must be done by  
Tuesday, January 28!**

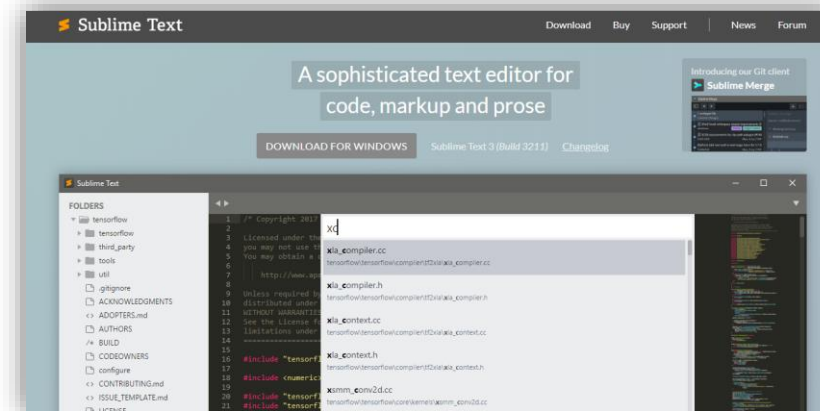
<https://code.visualstudio.com>



<https://www.anaconda.com/download/>



[iexcloud.io](https://iexcloud.io)

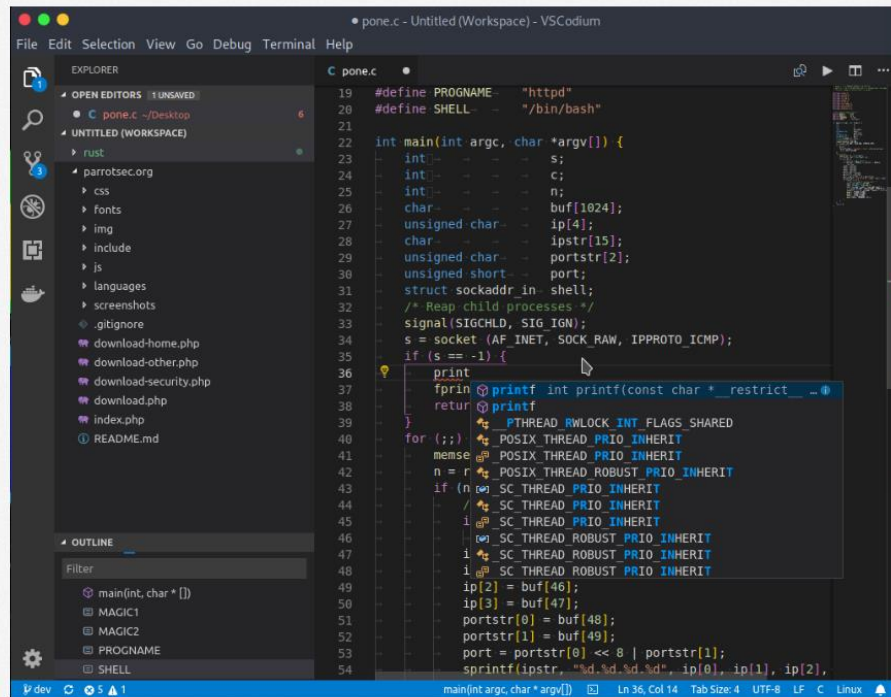


[sublimetext.com](https://sublimetext.com)


# If you want to use VS Code (which many recommend) ...

<https://vscodium.com>

## Free/Libre Open Source Software Binaries of VSCode



VSCodium is a community-driven, freely-licensed binary distribution of Microsoft's editor VSCode

release **v1.41.1**  Azure Pipelines **succeeded**  build **passing**  license **MIT**  chat **on gitter**

## Why Does This Exist

Microsoft's `vscode` source code is open source (MIT-licensed), but the the product available for download (Visual Studio Code) is licensed under [this not-FLOSS license](#) and contains telemetry/tracking. According to [this comment](#) from a Visual Studio Code maintainer:

*When we [Microsoft] build Visual Studio Code, we do exactly this. We clone the `vscode` repository, we lay down a customized `product.json` that has Microsoft specific functionality (telemetry, gallery, logo, etc.), and then produce a build that we release under our license.*


*When you clone and build from the `vscode` repo, none of these endpoints are configured in the default `product.json`. Therefore, you generate a "clean" build, without the Microsoft customizations, which is by default licensed under the MIT license*

The VSCodium project exists so that you don't have to download+build from source. This project includes special build scripts that clone Microsoft's `vscode` repo, run the build commands, and upload the resulting binaries for you to [GitHub releases](#). **These binaries are licensed under the MIT license. Telemetry is disabled.**

If you want to build from source yourself, head over to [Microsoft's `vscode` repo](#) and follow their [instructions](#). VSCodium exists to make it easier to get the latest version of MIT-licensed VSCode.

# Immediate homework assignment ...

Go to this link: <https://youtu.be/M7Ge0zi0oS4> and watch this, then use the instructions to set up an iexcloud.io site to extract historical data. If you want to get a little ahead, use the program discussed to actually download some sample data. Keep in mind that there is a limit to how much data they allow you to capture.



## Economics 136


Harvey Mudd College

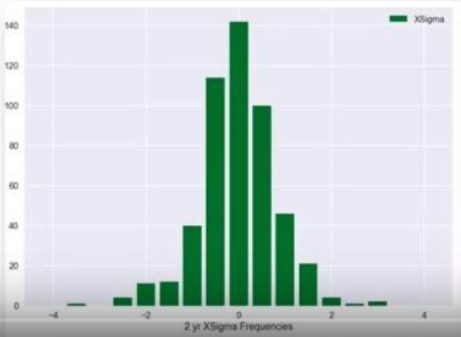
Using Python/Pandas, the IEX API, and Jupyter Notebooks to

# Download historical stock data from IEX Cloud ...


(how hard can it be??)

© 2019 Gary R. Evans. This video and slide set by Gary R. Evans is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.





A histogram showing the frequency distribution of 2-year XSigma values. The x-axis is labeled '2 yr XSigma Frequencies' and ranges from -4 to 4. The y-axis ranges from 0 to 140. The distribution is bell-shaped and centered around 0, with a peak frequency of approximately 140. A legend in the top right corner indicates 'XSigma' with a green square.



0:03 / 16:35

# ... in contrast to the tortured 2018 reversion market

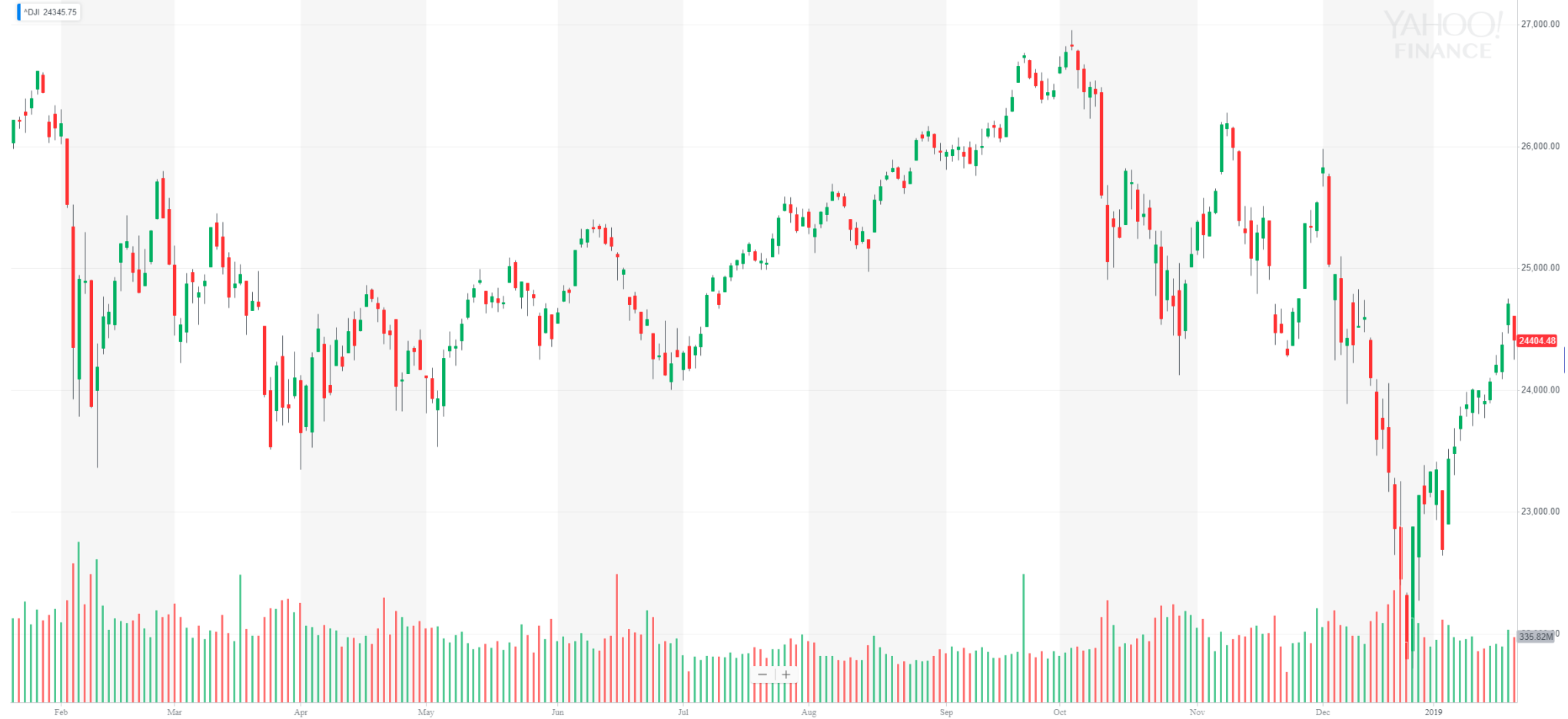
Dow Jones Industrial Average (^DJI) [☆ Add to watchlist](#)

DJI - DJI Real Time Price. Currency in USD

**24,404.48** -301.87 (-1.22%)

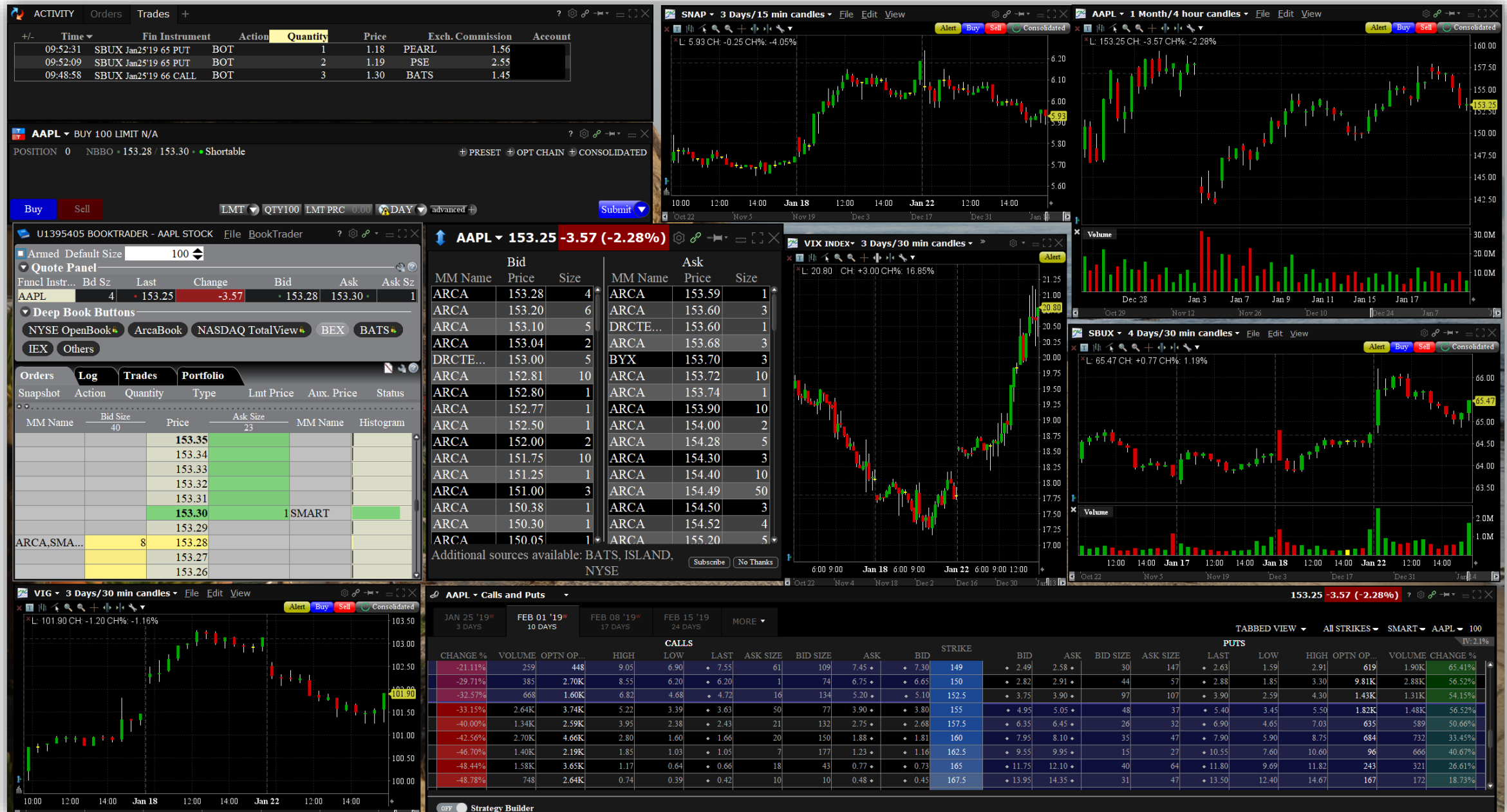
At close: 4:20PM EST

Indicators Comparison Date Range 1D 5D 1M 3M 6M YTD **1Y** 2Y 5Y Max Interval 1D Candle Draw





# The role played by Interactive Brokers ...





## HV Master IEX KS Jupyter

Make a copy of this then rename it if you want to use or test it. Do not write over this file. This downloads 2 years of historical data from IEX to calculate HV, perform and print a Kolmogorov-Smirnov test, and map the density distribution, and print the output to Excel. Modified on 7/24/2018 to accept earnings dates and to identify 2.5-sigma-plus moves Actual estimations should be saved using the following naming convention (because of the excel file): hv[symbol][day][month - 3 letters][year-2 digits].ipynb for example hvaapi7Jul18.ipynb

Version 2.4 modified on 1/8/2019. This needs a little checking and debugging but essentially is an operational file. This was done with NFLX for testing earnings dates. Keep for debugging reference for awhile in PyFi.

```
In [1]: 1 %matplotlib inline
```

```
In [2]: 1 import datetime
2 from datetime import date
3 import pandas as pd
4 import numpy as np
5 import math
6 import matplotlib.pyplot as plt
7 import matplotlib.mlab as mlab
8 import seaborn as sns
9 import sys
10 sys.path.append('c:/Users/Prof Gary Evans/Dropbox/PyGo/PyFi')
11 import finutil as fu
12 from IFython.display import display, HTML
```

## User-supplied information

The user need only supply the stock symbol and change check\_earnings to False if the symbol is an ETF like SPY.

```
In [3]: 1 stocksym = "AAPL"
2 check_earnings = True
```

Upper and lower case become issues when using the API and saving to Excel so we need to do this:

```
In [4]: 1 stocksymlo = stocksym.lower()
2 stocksymup = stocksym.upper()
3 stocksym = stocksymup
```

## Open the IEX API to extract the stock data

```
In [5]: 1 stock = pd.DataFrame()
2 stock = pd.read_json('https://api.iextrading.com/1.0/stock/'+stocksym+'/chart/2y')
```

```
In [6]: 1 length = len(stock)
2 length
```

```
Out[6]: 503
```

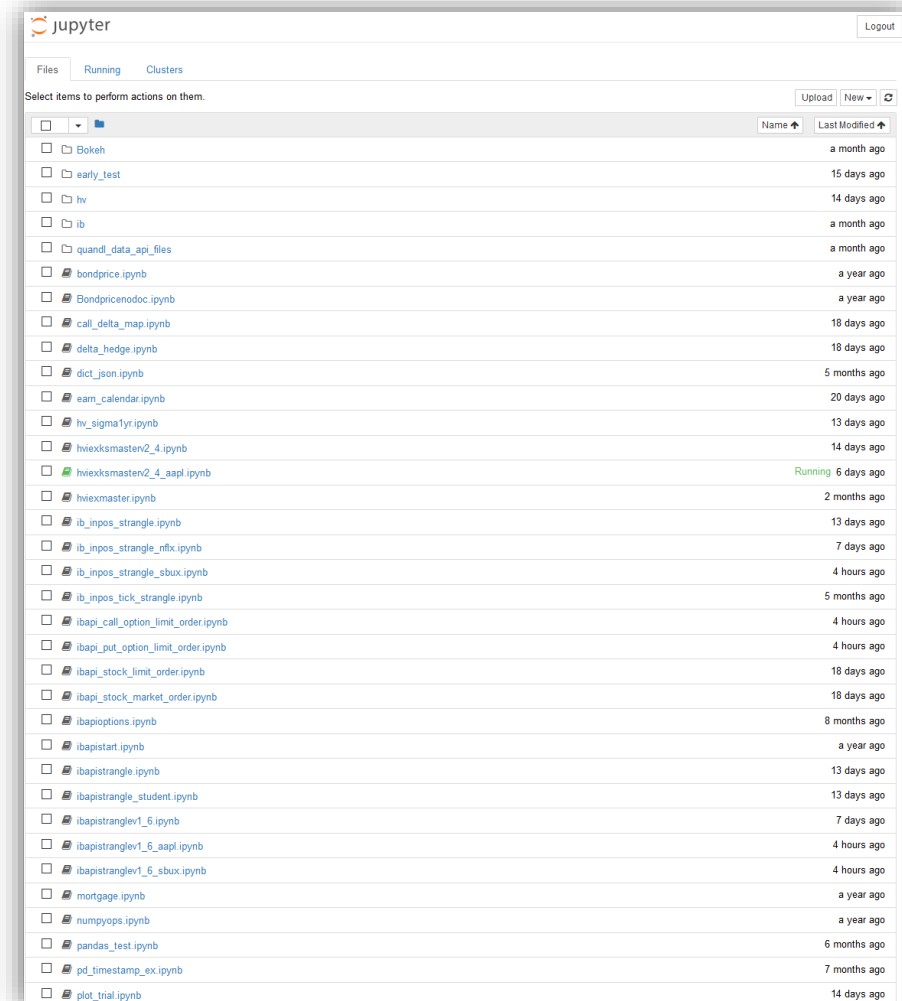
## Discard and rearrange data

Below we discard all downloaded data except four columns, Date, Volume, Open and Close. Date is no longer an index. Also with IEX data we want to use Close rather than Adjusted Close, in contrast to finance yahoo.

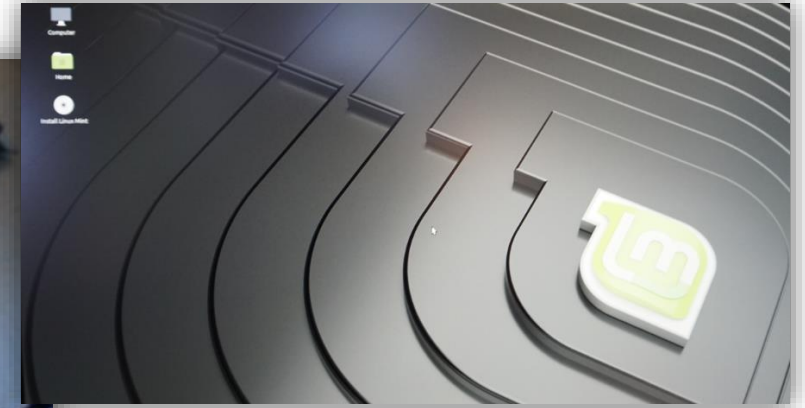
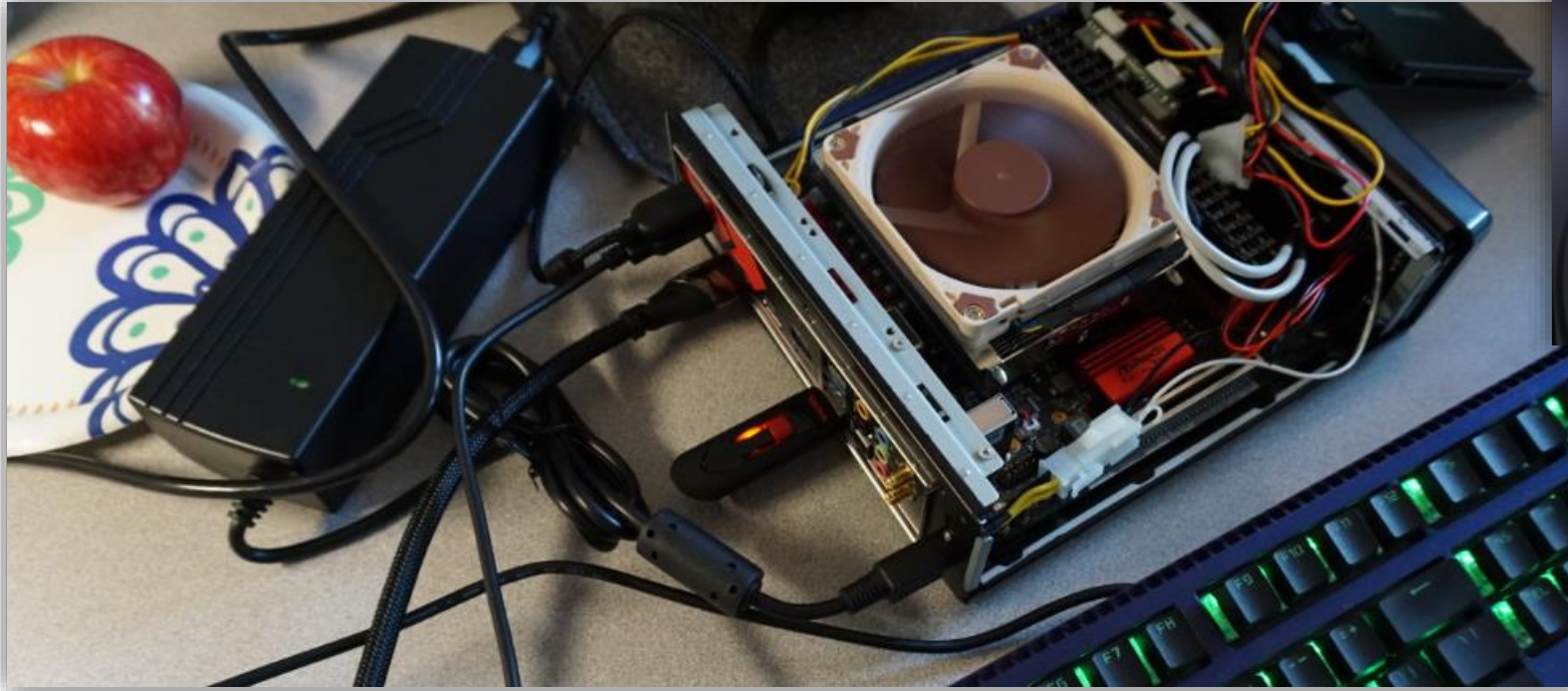
```
In [7]: 1 stock = stock.reset_index()
2 stock = stock[['date', 'volume', 'open', 'close']]
```

## Calculate log growth rates, drift and volatility estimates and add to data

# The role played by Jupyter Notebooks ...



# New task: Encouraging you to move to GNU-Linux



- FOSS, Creative Commons, and GNU Linux are healthier and less intrusive upon civil liberties than the alternatives.
- FOSS, Creative Commons and GNU Linux and nearly all software that runs on them, are free.
- GNU Linux and BSD are more suitable operating systems for (everything) modern data science.
- GNU Linux and BSD are more suitable for large-scale server-side operations.
- GNU Linux allows you to get very close to your machines. A scientist needs to be close to her machines!
- GNU Linux allows **us in this class** to run many of our Python programs as daemons!

# The beautiful 2017 momentum market ...

Mudd (Palm Island Traders) finance



... and back to the explosive mixed momentum market in 2019!

