# Implied Volatility using Python's Pandas Library

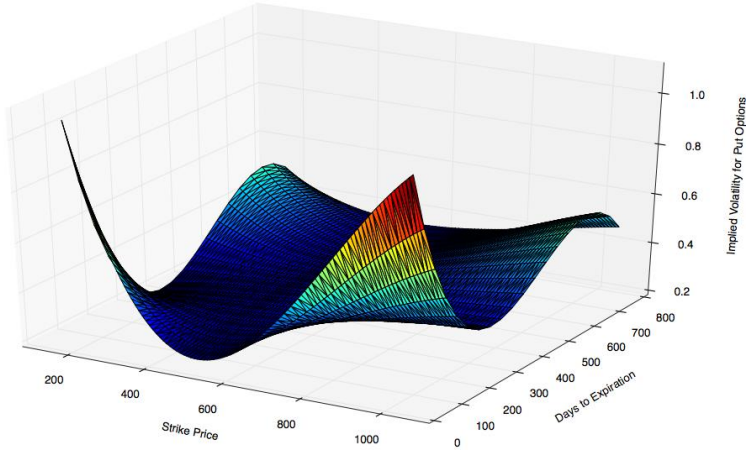
Brian Spector

*New York Quantitative Python Users Group*
*March 6th 2014*

**Experts in numerical algorithms and HPC services**

# Overview

- Introduction
- Motivation
- Python
- Pandas
- Implied Volatility
  - Timings in python
  - Different Volatility Curves
  - Fitting data points



# Numerical Algorithms Group

- Not-for-profit organization committed to research & development
- NAG provides mathematical and statistical algorithm libraries and services widely used in industry and academia
- Library code written and contributed by some of the world's most renowned mathematicians and computer scientists
- NAG Libraries available in C, MATLAB, .NET, Fortran, Java, SMP/Multicore, Excel, Python

# NAG Library Contents

- Root Finding
- Summation of Series
- Quadrature
- Ordinary Differential Equations
- Partial Differential Equations
- Numerical Differentiation
- Integral Equations
- Mesh Generation
- Interpolation
- Curve and Surface Fitting
- Optimization
- Approximations of Special Functions
- Dense Linear Algebra
- Sparse Linear Algebra
- Correlation & Regression Analysis
- Multivariate Methods
- Analysis of Variance
- Random Number Generators
- Univariate Estimation
- Nonparametric Statistics
- Smoothing in Statistics
- Contingency Table Analysis
- Survival Analysis
- Time Series Analysis
- Operations Research

# Motivation

- Data available from CBOE:
  - https://www.cboe.com/delayedquote/QuoteTableDownload.aspx

# Motivation

- Data available from CBOE:

```
AAPL (APPLE INC),531.03,+3.27,
Mar 04 2014 @ 12:18 ET,Bid,531.03,Ask,531.21,Size,5x1,Vol,3803030,
Calls,Last Sale,Net,Bid,Ask,Vol,Open Int,Puts,Last Sale,Net,Bid,Ask,Vol,Open Int,
14 Mar 400.00 (AAPL1407C400),132.00,0.0,130.10,132.00,0,3,14 Mar 400.00 (AAPL1407O400),0.01,0
14 Mar 400.00 (AAPL1407C400-4),0.0,0.0,129.35,133.05,0,3,14 Mar 400.00 (AAPL1407O400-4),0.0,0
14 Mar 400.00 (AAPL1407C400-8),0.0,0.0,130.10,132.00,0,3,14 Mar 400.00 (AAPL1407O400-8),0.02,(
14 Mar 400.00 (AAPL1407C400-A),0.0,0.0,130.10,132.00,0,3,14 Mar 400.00 (AAPL1407O400-A),0.0,0
14 Mar 400.00 (AAPL1407C400-B),0.0,0.0,129.35,133.20,0,3,14 Mar 400.00 (AAPL1407O400-B),0.0,0
14 Mar 400.00 (AAPL1407C400-E),144.33,0.0,129.35,133.60,0,3,14 Mar 400.00 (AAPL1407O400-E),0.
14 Mar 400.00 (AAPL1407C400-I),0.0,0.0,129.35,133.05,0,3,14 Mar 400.00 (AAPL1407O400-I),0.0,0
14 Mar 400.00 (AAPL1407C400-J),0.0,0.0,130.10,132.00,0,3,14 Mar 400.00 (AAPL1407O400-J),0.0,0
14 Mar 400.00 (AAPL1407C400-O),109.60,0.0,130.10,132.00,0,3,14 Mar 400.00 (AAPL1407O400-O),0.(
14 Mar 400.00 (AAPL1407C400-P),0.0,0.0,130.10,132.00,0,3,14 Mar 400.00 (AAPL1407O400-P),0.01,(
14 Mar 400.00 (AAPL1407C400-S),0.0,0.0,130.10,132.00,0,3,14 Mar 400.00 (AAPL1407O400-S),0.0,0
14 Mar 400.00 (AAPL1407C400-X),132.00,0.0,130.10,132.00,0,3,14 Mar 400.00 (AAPL1407O400-X),0.(
14 Mar 400.00 (AAPL1407C400-Y),0.0,0.0,130.10,132.00,0,3,14 Mar 400.00 (AAPL1407O400-Y),0.0,0
14 Mar 400.00 (AAPL71407C400),0.0,0.0,129.50,133.20,0,0,14 Mar 400.00 (AAPL71407O400),0.0,0.0
14 Mar 400.00 (AAPL71407C400-4),0.0,0.0,129.35,133.25,0,0,14 Mar 400.00 (AAPL71407O400-4),0.0
14 Mar 400.00 (AAPL71407C400-8),0.0,0.0,129.10,133.35,0,0,14 Mar 400.00 (AAPL71407O400-8),0.0
14 Mar 400.00 (AAPL71407C400-A),0.0,0.0,129.50,133.20,0,0,14 Mar 400.00 (AAPL71407O400-A),0.0
14 Mar 400.00 (AAPL71407C400-B),0.0,0.0,129.05,133.30,0,0,14 Mar 400.00 (AAPL71407O400-B),0.0
14 Mar 400.00 (AAPL71407C400-E),0.0,0.0,129.05,133.45,0,0,14 Mar 400.00 (AAPL71407O400-E),0.0
14 Mar 400.00 (AAPL71407C400-I),0.0,0.0,128.85,133.45,0,0,14 Mar 400.00 (AAPL71407O400-I),0.0
14 Mar 400.00 (AAPL71407C400-J),0.0,0.0,0.0,0.0,0,0,14 Mar 400.00 (AAPL71407O400-J),0.0,0.0,0
14 Mar 400.00 (AAPL71407C400-O),0.0,0.0,129.10,133.30,0,0,14 Mar 400.00 (AAPL71407O400-O),0.0
14 Mar 400.00 (AAPL71407C400-P),0.0,0.0,0.0,0.0,0,0,14 Mar 400.00 (AAPL71407O400-P),0.0,0.0,0
14 Mar 400.00 (AAPL71407C400-S),0.0,0.0,129.40,133.15,0,0,14 Mar 400.00 (AAPL71407O400-S),0.0
```

# Python

- Why use python?
  - Cheap
  - Easy to learn
  - Powerful

# Python

- Why use python?
  - Cheap
  - Easy to learn
  - Powerful
- Why use python over R?
  - “I would rather do math in a programming language than programming in a math language.”

# Python

- What python has:
  - Many built-in powerful packages
  - OO programming
    - Classes
    - Base + Derived Classes
  - Plotting
- What python does not have:
  - Multiple constructors
  - Pointers
  - ???

# numpy

- Has made numerical computing much easier in recent years.
- numpy matrices / arrays
- numpy.linalg
- Behind many of these functions are LAPACK + BLAS!

# scipy

- Special functions (scipy.special)
- Integration (scipy.integrate)
- Optimization (scipy.optimize)
- Interpolation (scipy.interpolate)
- Fourier Transforms (scipy.fftpack)
- Signal Processing (scipy.signal)
- Linear Algebra (scipy.linalg)
- Sparse Eigenvalue Problems with ARPACK
- Compressed Sparse Graph Routines scipy.sparse.csgraph
- Spatial data structures and algorithms (scipy.spatial)
- Statistics (scipy.stats)
- Multidimensional image processing (scipy.ndimage)

# nag4py

- nag4py (The NAG Library for Python)
- Built on top of NAG C Library + Documentation
- 1600 NAG functions easily accessible from python
- 15 examples programs to help users call NAG functions

```
from nag4py.c05 import c05ayc

from nag4py.util import NagError,Nag_Comm
```

# pandas

- Data Analysis Package
- Many nice built in functions
- Common tools:
  - Series / DataFrame
  - Reading + Writing CSVs
  - Indexing, missing data, reshaping
  - Common time series functionality

(Examples)

# Implied Volatility

- Black Scholes Formula for pricing a call/put option is a function of 6 variables:
  - $C(S_0, K, T, \sigma, r, d) = S_0 N(d_1) - K e^{-rT} N(d_2)$
- Where
  - $d_{1,2} = \frac{1}{\sigma\sqrt{T}}\left[ln\left(\frac{S}{K}\right) + T\left(r \pm \frac{\sigma^2}{2}\right)\right]$
  - $N(x) =$ Standard Normal CDF

# Implied Volatility

- We can observe the following in the market:
- $C(S_0, K, T, \sigma, r, d) = C$
- But what is σ?
- $\sigma_{imp} \rightarrow C_{BS}(S_0, K, T, \sigma_{imp}, r, d) = Market\ Price$

# Implied Volatility

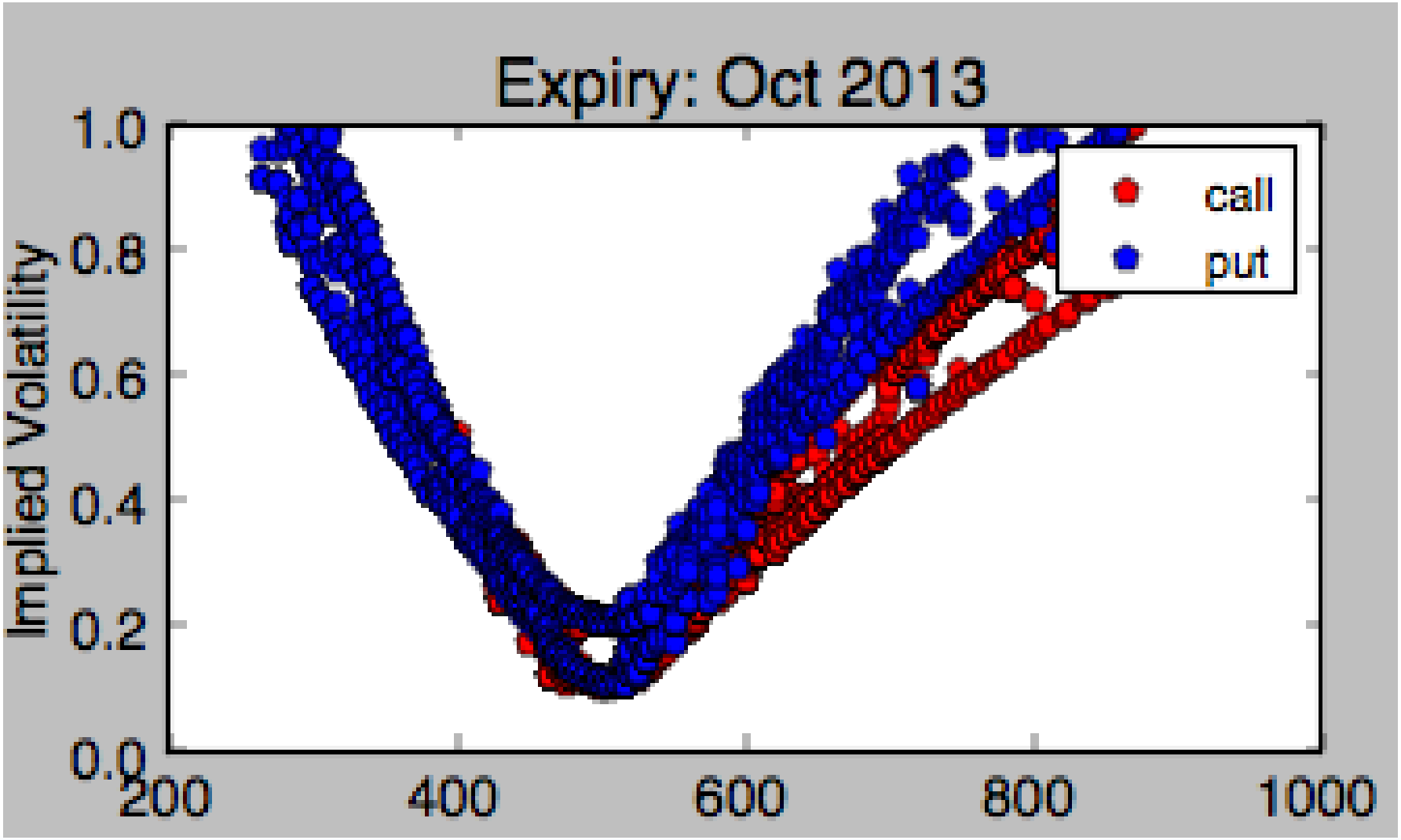
- We can observe the following in the market:
- $C(S_0, K, T, \sigma, r, d) = C$
- But what is σ?
- $\sigma_{imp} \rightarrow C_{BS}(S_0, K, T, \sigma_{imp}, r, d) = Market\ Price$
- Does $\sigma_{imp}$ exist?

# Implied Volatility

- We can observe the following in the market:
- $C(S_0, K, T, \sigma, r, d) = C$
- But what is σ?
- $\sigma_{imp} \rightarrow C_{BS}(S_0, K, T, \sigma_{imp}, r, d) = Market\ Price$
- Does $\sigma_{imp}$ exist?
  - Yes

(Examples)

# Implied Volatility – Different Curves?



# Implied Volatility – Different Curves?

- **No hyphen or letter present** = Composite
  - **A** = AMEX American Stock Exchange
  - **B** = BOX Boston Stock Exchange - Options
  - **E** = CBOE Chicago Board Options Exchange
  - **I** = BATS
  - **J** = NASDAQ OMX BX
  - **O** = NASDAQ OMX
  - **P** = NYSE Arca
  - **X** = PHLX Philadelphia Stock Exchange
  - **Y** = C2 Exchange
  - **4** = Miami Options Exchange
  - **8** = ISE International Securities Exchange

# Implied Volatility

- Reasons for skews/smiles?
  - Risk Preferences
  - Fat tailed distributions

# Implied Volatility Timings

| Method | Timing |
| --- | --- |
| fsolve + python BSM | |
| fsolve + NAG BSM | |
| nag4py | |
| NAG C | |

# Implied Volatility Timings

| Method | Timing |
| --- | --- |
| **fsolve + python BSM** | **~60 seconds** |
| fsolve + NAG BSM | |
| nag4py | |
| NAG C | |

# Implied Volatility Timings

| Method | Timing |
|---|---|
| fsolve + python BSM | ~60 seconds |
| **fsolve + NAG BSM** | **~10 seconds** |
| nag4py | |
| NAG C | |

# Implied Volatility Timings

| Method | Timing |
|---|---|
| fsolve + python BSM | ~60 seconds |
| fsolve + NAG BSM | ~10 seconds |
| **nag4py** | **~3 seconds** |
| NAG C | |

# Implied Volatility Timings

| Method | Timing |
|---|---|
| fsolve + python BSM | ~60 seconds |
| fsolve + NAG BSM | ~10 seconds |
| nag4py | ~3 seconds |
| **NAG C** | **~.15 seconds** |

# Implied Volatility Timings

| Method | Timing |
|---|---|
| fsolve + python BSM | ~60 seconds |
| fsolve + NAG BSM | ~10 seconds |
| nag4py | ~3 seconds |
| NAG C | ~.15 seconds |

- **Derivatives?**
- **We have the derivative, vega**
  - $\frac{\partial C}{\partial \sigma} = S * T * N'(d_1)$

# Fitting Data Points

- In our script we had k = l = 3…
  - What if we try different values?

# Fitting Data Points

- In our script we had k = l = 3…
  - What if we try different values?
    - Poor results, can we do better?
    - Two dimensional spline

# Thank You

## Questions?

- Further reading see:
- http://pandas.pydata.org/
- http://www.nag.co.uk/python.asp
- http://blog.nag.com/2013/10/implied-volatility-using-pythons-pandas.html