Stocks, volatility, and diversification

(Sethna, "Entropy, Order Parameters, and Complexity", ex. 2.11)

Stock prices are fairly good approximations to random walks. The Standard and Poor's 500 Index is a weighted average of the prices of five hundred large companies in the United States stock market.

Make sure you download SandPConstantDollars.dat from the exercises Web site. Each line represents a working weekday. The first column is time t in days since mid-October 1982, and the second column is the Standard and Poor's index $SP(t)$ for that day, corrected for inflation.

Import packages

```
In [ ]: %pylab inline
        from scipy import *
```

Read in the file of Standard and Poor's average stock price (SP) in constant dollars (corrected for inflation), versus time t

Convert from "list" to "array" form (arrays can be multiplied and added, etc.)

```
In [ ]: t, SP = array(loadtxt("SandPConstantDollars.dat").transpose())
```

Are the random fluctuations in the stock-market due to external events?

Plot SP versus t

Note 9/11/01 is day 6903: is it the cause for the post-2000 drop?"

Zoom in and see: did the terrorist attach on the World Trade Center trigger the stock market downturn, or was it just a small extra dip in an overall pattern?

You can plot a subset of the data using 'slicing': for a list or array v, v[m:n] gives back the elements in the range [m,n) -- starting at m and ending at n-1. The interesting region around the attack at day 6903 is around trading day 4750 (remember weekends count as days, but not as trading days), and ends around 4900.

```
In [ ]: plot(...);
        figure()
        tradeDayMin, tradeDayMax = (...)
        plot(t[tradeDayMin:tradeDayMax], SP[...]);
```

Sometimes large fluctuations are due to external events; the fluctuations in ecological populations and species are also quite random, but the dinosaur extinction was surely caused by a meteor.

What is the distribution of step sizes in stock prices? This depends on how we define a step; do we ask how much it has changed after a year, a month, a week, or a day? A technical question arises: do we measure time in days, or trading days. We shall follow the finance community, and consider only trading days.

Define a function P(lag) giving the percentage change after a 'lag' of trading days.

Use slicing to do all the comparisons in one array operation. For a list or array v, v[m:n] gives back the elements in the range [m,n) -- starting at m and ending at n-1. Python also makes the useful convention that v[:,n] starts at the beginning of the list, v[m:] ends at the end, and v[:,-p] 'drops' the last p elements.

Also use the scipy convention that arithmetic acts element-by-element on arrays. So for two arrays w and v of the same length,

```
w*v = array([w[0]*v[0],w[1]*v[1], ...]),
```

and similarly for w/v, w+v, and w-v. (The last two are the same as the math conventions for vectors.) Also, adding and subtracting scalars from arrays is done elementwise, so

```
v+5 = array([v[0]+5, v[1]+5, ...]).
```

Loops are very slow in interpreted languages like Python, so learning to do things with array operations like these is strongly recommended.

```
In [ ]: def P(lag):
            """
            Function which returns a list of percentage changes after a numb
        er
            "lag" of trading days.
            P(1) gives the daily percentage changes, P(5) the weekly change
        s, P[252] the yearly changes, etc.

            To find the ratio of values of an array v of length N after a sh
        ift of t, one could use
                v[t:N]/v[0:N-t]
            or
                v[t:]/v[:-t]
            """
            ratios = .../...
            P = ...*(...-...)
            return P
```

Plot a normed histogram using 50 bins for daily, weekly, and yearly percentage changes. Which of the three represents a reasonable time to stay invested in the Standard and Poor's index (during which you have a mean percentage growth larger than a tiny fraction of the fluctuations)? Why do the yearly changes look so much more complicated than the other distributions?

In [ ]:
```
hist(P(252), bins=50, normed=True);
hist(...);
hist(...);
```

Some of the distributions you found above should look quite Gaussian, as predicted from our Green's function analysis of random walks (or more generally, to the central limit theorem). Here the random walk is in the percentage return -- in the logarithm of the fractional price change. The logarithm of a Gaussian distribution is an inverted parabola. Do a log plot of the histogram of weekly price changes:

```
hist(..., log=True)
```

Are there 'fat tails' (more large changes than predicted by the inverted parabola you would expect from a Gaussian)? (Hint: P(lag).mean() and P(lag).std() may be useful in generating the Gaussian fit to the data.)

In [ ]:
```
lag = 5
hist(P(...), ..., log=True);
x = arange(-7.,7.,0.1);
sig = P(lag). ...
x0 = ...
gauss = (1./...)*exp(...)
plot(x,gauss,'r');
```

Make an array of lag times running from one to 100, and an array of volatilities using std(). Plot volatility and volatility squared versus lag.

Hint: [blah(lag) for lag in lags] produces a list of blah(lag[0]), blah(lag[1]), ..., and array([blah(lag) for lag in lags]) will change the list to an array (which then can be squared using **2)

In [ ]:
```
lags = arange(1,100)
volatility = array([... for ...])
```

In [ ]:
```
plot(...);
```

In [ ]:
```
plot(...);
```