Lecture 10:
Price Distributions and Binomial Trees

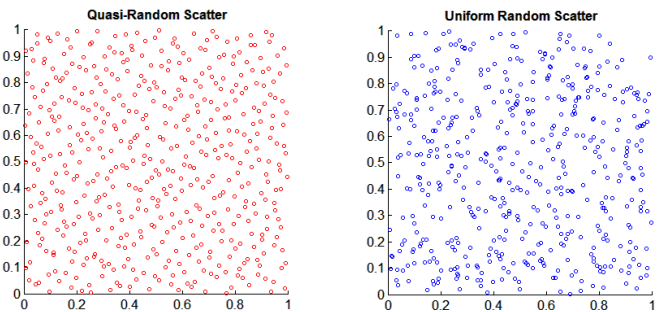
Steven Skiena

Department of Computer Science
State University of New York
Stony Brook, NY 11794–4400

http://www.cs.sunysb.edu/∼skiena

Quasi-random Number Generation

Quasi-random sequences fill space more uniformly than uncorrelated random points.



Intuition: on a line, consider repeatedly picking the next point in the middle of the largest remaining interval.

Note these are (1) not “randomly” ordered, and (2) will not be uniform if you don’t know the ultimate number in advance.

Random Walks with Memory

Successive movements in the random walks models to discussed to date are *independent*, which contradicts our perception about how markets move.

Hurst random walks are discrete random walks which reverse direction with probability h.

A value of $h = 0.5$ generates a coin flipping random walk, while a value of $h = 1.0$ generates a walk which moves in only one direction.

Intermediate values of h should generate walks more "driven" than simple coin-flipping, although the eye often mistakenly identifies trends in such walks.

Hurst walks arise in the analysis of *fractal* phenomenon.

Building a Price Distribution Model

Several design decisions remain to build a reasonable stock price distribution model:

- The number of steps per simulated time period.
- The up-tick and down-tick probabilities.
- The drift rate, perhaps the historical stock market average returns.
- The step size, which is a function of each given stock's volatility.
- The number of walks simulated per distribution.

Volatility Prediction

Stock volatility (measured by the absolute value of returns) tends to show much stronger short term correlation than returns itself.

Lag	Volatility Corr.	Return Corr.
1	0.441	0.021
2	0.371	-0.016
3	0.337	-0.024
4	0.311	-0.016
5	0.319	0.004
10	0.287	0.005
20	0.249	-0.012
30	0.264	-0.001
40	0.233	0.005
50	0.209	-0.002

We used an exponentially weighted moving average model to update the volatility σ_n^2 in response to each day's observation u, with $\lambda = 0.94$:

$$\sigma_n^2 = \lambda \sigma_{n-1}^2 + (1 - \lambda) u_{n-1}$$

Setting the Parameters

To incorporate the volatility prediction into our random walk model, we must map volatility to parameters for (1) the simulated number of steps per day, and (2) the step size.

We modeled each trading day by a walk of 1000 steps, and adjusted the step size so as to produce the step size to achieve the desired volatility.

We used a Hurst random walk model with $h = .57$, which gave us the best results.

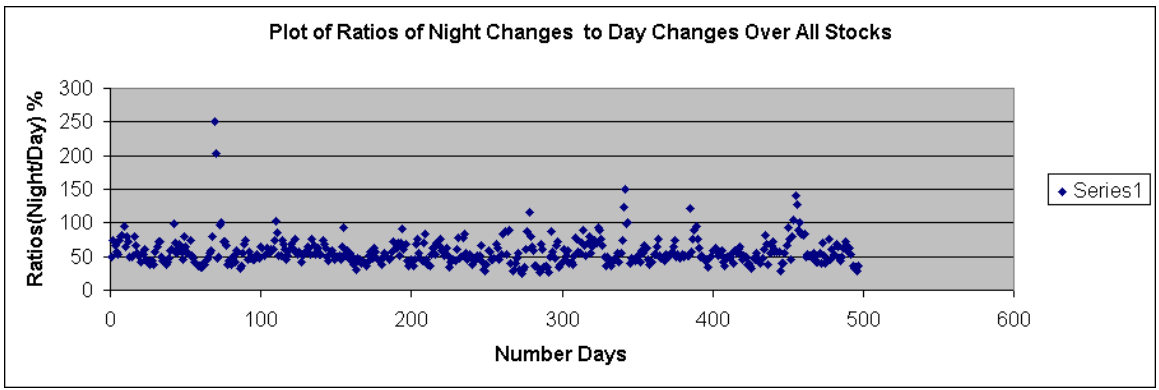
We did not model any drift, because we were interested in predictions over very short time intervals.

Night Moves

Usually there is a substantial difference between one days *closing* price and the next day's *opening price*, reflecting the news that occurred in the interim.

The NYSE is open 9:30AM-3:30PM each day. Does more or less activity happen in the 18 hours until the next session?

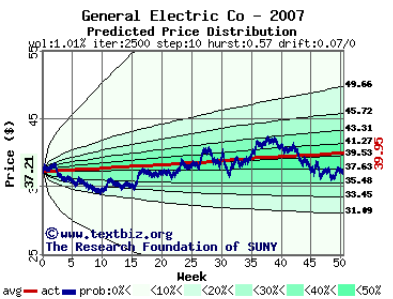
This can be established by plotting the average daily ratio of night-to-day changes for Dow stocks:

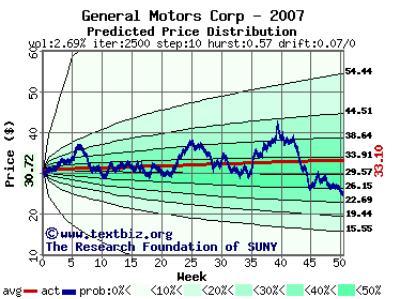


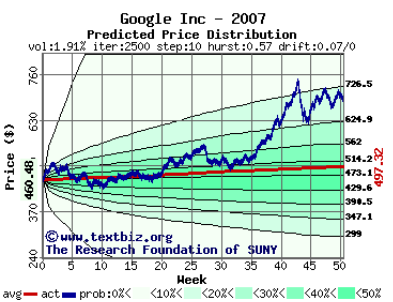
The average night-move over this period was 0.567 that of the day-move, with a mean of 0.527.

The impact of these moves can be simulated by running the random walk the equivalent of this many steps each night and starting the next day from there.

Price Distributions for Real Stocks (www.textbiz.org)

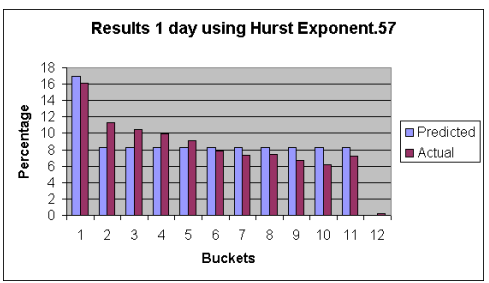


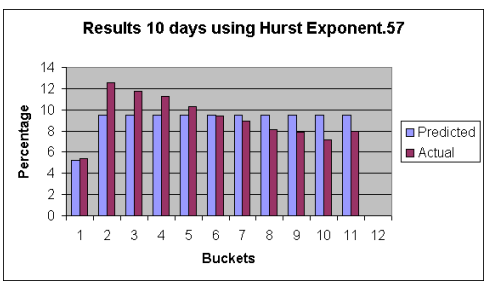




Results: Predicting the Expected High

We used our random walk to predict the range of the expected high achieved over the next 1 day and 10 days for a wide range of stocks:





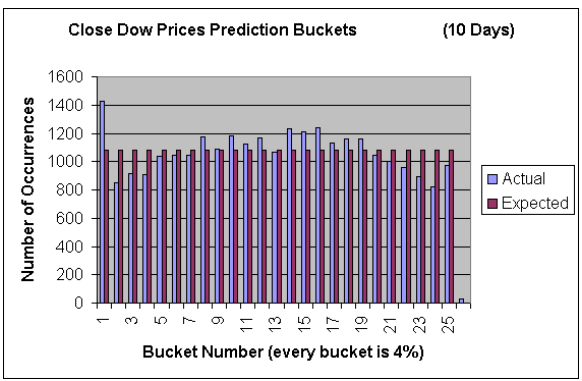
Each observed price maps to a cumulative probability in our distribution; the degree to which they fall into buckets as predicted are a measure of the accuracy of our model.

Interpreting the Results

The leftmost point on each plot records the frequency the actually high *never* exceeded the close of the start period.
The rightmost point records the frequency the actually high exceeded our prediction of what is possible in the time period.
The random walks do a good, but not perfect job, of predicting the actual distribution.

Results: Predicting Closing Prices

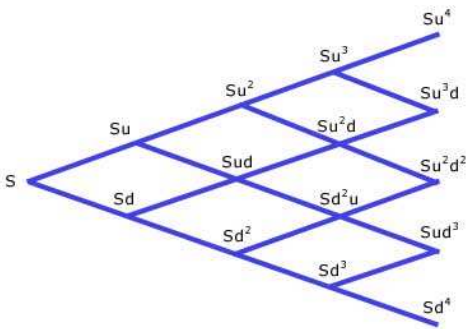
We also do a good, but not perfect, job of predicting closing prices:



The punch line is that simple random walk models can do a reasonable job of modeling future price distributions.

Binomial Trees

Additive or multiplicative walk models with *fixed up and down step sizes* gives rise to a graph of possible paths called a binomial tree within the finance literature:



Although there are 2^n distinct paths through this n-stage network there are only $\binom{n}{2}$ nodes, so we can efficiently compute the *exact* distribution of paths through this network!

Pascal's Triangle

No doubt you played with this arrangement of numbers in high school. Each number is the sum of the two numbers directly above it:

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1

Pascal's Recurrence

The binomial coefficient $\binom{n}{k}$ counts the number of ways to choose k things out of n possibilities.

A stable way to compute binomial coefficients uses the recurrence relation implicit in the construction of Pascal's triangle, namely, that

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

It works because the nth element either appears or does not appear in one of the $\binom{n}{k}$ subsets of k elements.

The basis cases are $\binom{n}{0} = 1$ and $\binom{k}{k} = 1$.

Binomial Coefficients Implementation

```
long binomial_coefficient(n,m)
int n,m; (* compute n choose m *)
{
        int i,j; (* counters *)
        long bc[MAXN][MAXN]; (* table of binomial coefficients *)

        for (i=0; i<=n; i++) bc[i][0] = 1;

        for (j=0; j<=n; j++) bc[j][j] = 1;

        for (i=1; i<=n; i++)
        for (j=1; j<i; j++)
                bc[i][j] = bc[i-1][j-1] + bc[i-1][j];

        return( bc[n][m] );
}
```

Computing Path Probability Distributions

The same idea can be used to efficiently compute the probability $Pr[n,k]$ of reaching the n-level node with exactly k up-moves in the binomial tree,

$$Pr[n,k] = p_d Pr[n-1,k] + p_u Pr[n-1,k-1]$$

Binomial trees compute option prices using this approach.
Note this algorithm works fine even if the up and down edge probabilites differ at each node.
But what is a principled way to set the up and down edge probabilities?

Risk-Neutral Probabilities

We can use an arbitrage argument to set the "right" probability of an upward move (β) as a function of the risk-free rate.

At any point, investors can either (a) hold \$1 stock or (b) invest \$1 at the risk-free rate r.

A *risk-neutral* investor would not care which portfolio they owned if they had the same return.

Setting equal the returns from the stock ($\beta\alpha + (1-\beta)/\alpha$) and the risk-free portfolio ($1+r$), we can solve for β to determine the *risk-neutral probability*.

But in truth, investors are not risk-neutral. In order to take the riskier investment they must be paid a premium.