COMPUTATIONAL FINANCE

Lecture 3: Asset Allocation Simulation
Brief Introduction to Simulation

Philip H. Dybvig
Washington University
Saint Louis, Missouri

Copyright © Philip H. Dybvig 1996

Some Leading Uses of Simulation in Finance

- Analysis of portfolio strategies
- Option pricing
- Stochastic *pro forma* analysis

Some Simulation Strategies

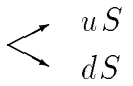
- Use pseudo-random numbers (Monte Carlo)
- Use quasi-random numbers
- Use historical data (back-testing)
- Use pseudo-random numbers to draw from historical data (bootstrapping)

Representing Discrete Random Variables

A discrete random variable's distribution can be represented by listing the values and their probabilities:

value	probability
uS	p
dS	1-p

This is implicit in our notation



where the up move has probability p.

Representing a general random variable

For random variables that can take on a continuum of values, listing all outcomes is not feasible and we use instead the *distribution function*. The distribution function of a random variable y is defined by $F(x) = prob(y \leq x)$, i.e., the value of the distribution function at x is the probability that y is less than or equal to x.

Distribution Function: Example

Suppose $dS < uS$. Then the random variable

value	probability
uS	p
dS	1-p

has the distribution function

$$F(x) = \begin{cases} 0 & x < dS \\ 1-p & dS \le x < uS \\ 1 & uS \le x \end{cases}$$

Uniform Distribution

A uniform distribution is distributed uniformly on some interval (by default [0,1]). We have the shorthand y $\sim$ U(a,b) to indicated that y is distributed uniformly on the interval [a,b]. For y $\sim$ U(0,1), the distribution function is

$$F(x) = \begin{cases} 0 & x < 0 \\ x & 0 \leq x < 1 \\ 1 & 1 \leq x \end{cases}$$

Therefore, for $0 \leq a < b \leq 1$, The probability that y lies in the interval $[a, b]$ is simply the length $b - a$ of the interval.

In-class Exercise: Distribution Functions

Take y $\sim$ U(0,1). Express the distribution of the random variable

$$z = \begin{cases} u & \text{when } y < p \\ d & \text{when } y > p \end{cases}$$

both (1) as a list of values and probabilities and (2) as a distribution function. Assume that $d < u$.

Simulating a Uniform Distribution

The standard function rand returns a random int in the range from 0 to RAND_MAX. This random variable generator is available on all platforms but has a bit of a bad reputation (see, for example, Chapter 7 of the book Numerical Recipes in C, by Press et al, Cambridge, ISBN 0-521-35465-X); many compilers have better random number generators built-in. Using rand will be more than adequate for our purposes.

The expression (double) rand()/(double) RAND_MAX simulates a random variable distributed uniformly on [0,1]. A variable that is U(0,1) has a mean of 1/2 and a variance of 1/12, so we can generate a uniform variate with mean m and standard deviation s as

```
m+s*((double)rand()/(double)RAND_MAX)-0.5)*sqrt((double)12).
```

This is implicit in our program this week.

Portfolio "Accounting"

Make the following definitions.

- C_t is the amount of cash after trading at t.
- S_t^i is the position in stock i after trading at t.
- r_t is the total return (1 + interest rate) on cash at t.
- r_t^i is the total return (1 plus rate of return) on stock i at t.
- d_i is the proportional cost for trading security i.

Then, absent taxes,

$$C_t = C_{t-1} r_t + \sum_i (S_{t-1}^i r_t^i - S_t^i) - \sum_i d_i |S_{t-1}^i r_t^i - S_t^i|. \tag{1}$$

With taxes, things are much messier. (See my paper with Hyeng-Keun Koo.)

The Header File simu1.h

```
//
// simu1.h Asset Allocation Header
//

struct termvals {
  double termstock;
  double termwealth;};

class aa {
  public:
    aa(double ttm=1.0,int nper=120,double r=.05,double mean=.15,double stddev=.2
    int restart(double ttm,int nper,double r,double mean,double stddev);
    termvals fixprops(double inrisky,double initstock=0.0,double initcash=100.0)
  private:
    int npers;
    double tinc, r1per, mean1per, std1persqrt12, wealth, stockpos, cash, stockP
      stockret;
```

```
double stocktotret();};
```

The Test File simu1test.cc

```
//
// simu1test.cc Asset Allocation test file
//
#include <iostream.h>
#include "simu1.h"

main() {
  int i;
  aa sim1;
  termvals y;
  for(i=0;i<100;i++) {
    y = sim1.fixprops(.5);
    cout << y.termstock << " " << y.termwealth << endl;}
  return(0);}
```

The Implementation File simu1.cc: Part 1

```
//
// simu1.cc Asset Allocation Simulation
//
#include <math.h>
#include <stdlib.h>
#include <iostream.h>
#include "simu1.h"
#define unifrand() ((double) rand()/((double) RAND_MAX))
```

The Implementation File simu1.cc: Part 2

```
aa::aa(double ttm,int nper,double r,double mean,double stddev) {
  restart(ttm,nper,r,mean,stddev);}

int aa::restart(double ttm,int nper,double r,double mean,double stddev) {
  npers = nper;
  tinc = ttm/(double) nper;
  r1per = 1.0 + r*tinc;
  mean1per = 1.0 + mean*tinc;
  std1persqrt12 = sqrt((double) 12)*stddev*sqrt(tinc);
//  cout << " " << npers << " " << tinc << " " << r1per << " "
//    << mean1per << " " << std1persqrt12 << endl;
  return(0);}
```

The Implementation File simu1.cc: Part 3

```
termvals aa::fixprops(double inrisky,double initstock,double initcash) {
  int i;
  termvals x;
  stockpos = initstock;
  cash = initcash;
  stockP = 100.0;
//  cout << npers << endl;
  for(i=0;i<npers;i++) {
    wealth = stockpos + cash;
    stockpos = inrisky * wealth;
    cash = wealth - stockpos;
    stockret = stocktotret();
//cout << stockret << " " << tinc << " " << mean1per << " " << std1persqrt12 <<
//" " << unifrand() <<endl;
    stockP *= stockret;
    stockpos *= stockret;
    cash *= r1per;
```

```
//    cout << stockret << " " << stockP << " " << stockpos << " " << cash << end
    }
  x.termstock = stockP;
  x.termwealth = stockpos + cash;
  return(x);}

double aa::stocktotret() {
  return(mean1per + std1persqrt12 * (unifrand()-0.5));}
```