COMPUTATIONAL FINANCE

Lecture 4: Asset Allocation Simulation
Brief Introduction to Simulation

Philip H. Dybvig
Washington University
Saint Louis, Missouri

Some Leading Uses of Simulation in Finance

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

Some Simulation Strategies

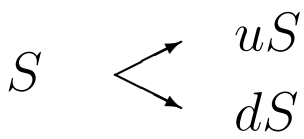
- Use pseudo-random numbers (Monte Carlo)
- Use quasi-random numbers
- Use historical data (back-testing)
- Use pseudo-random numbers to draw from historical data (bootstrapping)

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

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

both (1) as a list of values and probabilities and (2) as a distribution function. Assume that $d < u$ and that the constant p satisfies $0 < p < 1$.

Simulating a Uniform Distribution

The Java function Math.random() returns a pseudo-random double in the range from 0 to 1 that is actually deterministic but behaves so erratically from draw to draw that it seems random and independent of previous draws. This random variable generator is available on all platforms but I am not at all confident it is a good one (see, for example, Chapter 7 of the book Numerical Recipes in C, by Press et al, Cambridge, ISBN 0-521-35465-X, which discusses random number generators in C and provides code for good ones); Using Math.random() will be more than adequate for our purposes, but if you are going to use a lot of random draws I suggest looking into more reliable random number generator.

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*(Math.random()-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 buying or selling security i.

Then, absent taxes,

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

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

The HTML File *AssetAlloc.html*

```
<HTML>
<HEAD>
<TITLE>Asset Allocation Simulation</TITLE>
</HEAD>
<BODY>
<APPLET CODE=AssetAlloc.class WIDTH=300 HEIGHT=50>
</APPLET>
</BODY>
</HTML>
```

Asset Allocation File *AssetAlloc.java*

```
//
// Asset Allocation Applet
//
// This applet simulates asset allocation between risky and riskless
// assets.  The horizon is fixed at 10 years and initial wealth is
// normalized to be 100.  The interest rate is constant and stock
// returns are i.i.d. over time.
//

import java.applet.*;
import java.awt.*;

public class AssetAlloc extends Applet {
  ValuePlotFrame valuePlotFrame;
  Button startASimu;
    public AssetAlloc() {
      setLayout(new GridLayout(1,1));
      add(startASimu = new Button("Start a Simulation"));
      valuePlotFrame = new ValuePlotFrame();
```

```
      valuePlotFrame.setTitle("Asset Allocation Simulation");
      valuePlotFrame.pack();
      valuePlotFrame.resize(500,400);}
    public boolean action(Event e, Object arg) {
      if(e.target == startASimu) {
        valuePlotFrame.reset();
        return true;}
      return false;}}

class ValuePlotFrame extends Frame {
  BasicLinePlot blp;
  TextField r,mu,sigma,inrisky;
  Button newRandomDraws,resetInputs;
  double ttm = 10.0,ir;
  double[] times,values;
  int nper = 1200;
  int i;
  AssetAllocEngine vroom;
  double[] xgrid,ygrid;
  public ValuePlotFrame() {
    setLayout(new BorderLayout());
```

```
blp = new BasicLinePlot();
add("Center",blp);
Panel inputs = new Panel();
inputs.setLayout(new GridLayout(6,2));
  inputs.add(new Label("interest rate (%/yr)"));
  inputs.add(r = new TextField("5",10));
  inputs.add(new Label("mean stock return (%/yr)"));
  inputs.add(mu = new TextField("15",10));
  inputs.add(new Label("std dev of stock return (%/yr)"));
  inputs.add(sigma = new TextField("30",10));
  inputs.add(new Label("allocation to risky asset (%)"));
  inputs.add(inrisky = new TextField("70",10));
  inputs.add(newRandomDraws = new Button("Simulate again"));
  inputs.add(new Label(""));
  inputs.add(resetInputs = new Button("Reset and simulate again"));
add("South",inputs);
vroom = new AssetAllocEngine(nper);
times = new double[nper + 1];
for(i=0;i<=nper;i++) times[i] = ((double) i) * ttm / ((double) nper);
xgrid = new double[6];
for(i=0;i<6;i++) xgrid[i] = ((double) i) * 2.0;
```

```
  ygrid = new double[6];
  for(i=0;i<6;i++) ygrid[i] = ((double) i) * 100.0;}
public void startSimu() {
  vroom.newPars(text2double(r)/100.0,
      text2double(mu)/100.0,text2double(sigma)/100.0,ttm);
  ir = text2double(inrisky)/100.0;
  values = vroom.fixProps(ir,ir * 100.0,(1.0-ir)*100.0);
  blp.newXY(times,values,xgrid,ygrid,
    "Simulated Wealth, Fixed Proportions Startegy: 0-10 years out");
  show();}
public void reset() {
  r.setText("5");
  mu.setText("15");
  sigma.setText("30");
  inrisky.setText("70");
  startSimu();}
public boolean action(Event e, Object arg) {
  if(e.target == newRandomDraws) {
    startSimu();
    return true;}
  if(e.target == resetInputs) {
```

```
      reset();
      return true;}
    return false;}
  public boolean handleEvent(Event event) {
    if(event.id == Event.WINDOW_DESTROY) {
      dispose();}
    return super.handleEvent(event);}
  double text2double(TextField tf) {
    return Double.valueOf(tf.getText()).doubleValue();}}

class AssetAllocEngine {
  int nper;
  double tinc,r1per,mean1per,std1persqrt12;
  double[] values;
  public AssetAllocEngine(int nper) {
    values = new double[nper+1];
    this.nper = nper;}
  public void newPars(double r,double mu,double sigma,double ttm) {
    tinc = ttm/((double) nper);
    r1per = 1.0 + r * tinc;
    mean1per = 1.0 + mu * tinc;
```

```
  std1persqrt12 = sigma * Math.sqrt(12.0 * tinc);}
public double[] fixProps(double inrisky,double initstock,
  double initcash) {
  double stock,cash,wealth,stockret;
  int i;
  stock = initstock;
  cash = initcash;
  for(i=1,values[0]=stock + cash;i<=nper;i++) {
    wealth = stock + cash;
    stock = wealth * inrisky;
    cash = wealth - stock;
    stockret = stockTotRet();
    stock *= stockret;
    cash *= r1per;
    values[i] = cash + stock;}
  return values;}
double stockTotRet() {
  return mean1per + std1persqrt12 * (Math.random()-0.5);}}
```

The Plotting Program File *BasicLinePlot.java*

```
import java.awt.*;

public class BasicLinePlot extends Canvas {
  double[] x,y,xgrid,ygrid;
  int i,nper;
  int ixleft,ixright,iytop,iybottom,ih,iw,ihalftic,iRise,fmHeight;
  double xleft,xright,ytop,ybottom;
  double lmarg = 0.10,rmarg = 0.05,tmarg = 0.10,bmarg = 0.10,
    halftic = 0.0035;
  double ixintercept,ixslope,iyintercept,iyslope;
  String thingy;
  public BasicLinePlot() {}
  String mainTitle;
  public void newXY(double[] x,double[] y,double[] xgrid,double[] ygrid,
    String mainTitle) {
    if(x.length != y.length) {
      System.err.println("Error: x and y must have the same length");
      System.exit(-1);}
    this.x = new double[x.length];
```

```
  this.y = new double[x.length];
  for(i=0;i<x.length;i++) {
    this.x[i] = x[i];
    this.y[i] = y[i];}
  this.xgrid = new double[xgrid.length];
  this.ygrid = new double[ygrid.length];
  for(i=0;i<xgrid.length;i++)
    this.xgrid[i] = xgrid[i];
  for(i=0;i<ygrid.length;i++)
    this.ygrid[i] = ygrid[i];
  this.mainTitle = mainTitle;
  repaint();}
public void paint(Graphics g) {
    if(xgrid != null) {
    xleft = xgrid[0];
    xright = xgrid[xgrid.length - 1];
    ybottom = ygrid[0];
    ytop = ygrid[ygrid.length - 1];

    Rectangle r = bounds();
    ih = r.height;
```

```
      iw = r.width;
      iytop = (int) (tmarg * ((double) ih));
      iybottom = (int) ((1.0-bmarg) * ((double) ih));
      ixleft = (int) (lmarg * ((double) iw));
      ixright = (int) ((1.0-rmarg) * ((double) iw));
      ihalftic = (int) (halftic * ((double) iw));

      ixslope = ((double) (ixright - ixleft))/(xright - xleft);
      ixintercept = 0.5 + (((double) ixright) * xleft -
        ((double) ixleft) * xright)/(xleft - xright);
      iyslope = ((double) (iybottom - iytop))/(ybottom - ytop);
      iyintercept = 0.5 + (((double) iybottom) * ytop -
        ((double) iytop) * ybottom)/(ytop - ybottom);

//
//  white background
//
      g.setColor(Color.white);
      g.fillRect(0,0,iw,ih);

//
```

```
//  draw the axes
//
      g.setColor(Color.black);
      g.drawLine(ixleft,iytop,ixleft,iybottom);
      g.drawLine(ixleft,iybottom,ixright,iybottom);

//
//  add tick marks
//
      g.setColor(Color.black);
      for(i=0;i<ygrid.length;i++)
        g.drawLine(ixleft-ihalftic,y2iy(ygrid[i]),
          ixleft+ihalftic,y2iy(ygrid[i]));
      for(i=0;i<xgrid.length;i++)
        g.drawLine(x2ix(xgrid[i]),iybottom-ihalftic,
          x2ix(xgrid[i]),iybottom+ihalftic);

//
//  add axis numbers
//
      g.setColor(Color.black);
```

```
      FontMetrics fm = g.getFontMetrics();
      iRise = fm.getAscent()/2;
      fmHeight = fm.getHeight();
      for(i=0;i<ygrid.length;i++) {
        thingy = Double.toString(ygrid[i]);
        g.drawString(thingy,ixleft - ihalftic -fm.stringWidth(thingy),
          y2iy(ygrid[i]) + iRise);}
      for(i=0;i<xgrid.length;i++) {
        thingy = Double.toString(xgrid[i]);
        g.drawString(thingy,x2ix(xgrid[i]) - fm.stringWidth(thingy)/2,
          iybottom + ihalftic + fmHeight);}

//
//  plot lines
//
      g.setColor(Color.blue);
      for(i=1;i<x.length;i++)
        g.drawLine(x2ix(x[i-1]),y2iy(y[i-1]),
          x2ix(x[i]),y2iy(y[i]));

//
```

```
//  add the mainTitle
//
      g.drawString(mainTitle,(ixright + ixleft)/2 -
        fm.stringWidth(mainTitle)/2,fmHeight);}}
  int x2ix(double x) {return (int) Math.floor(ixintercept + ixslope * x);}
  int y2iy(double y) {return (int) Math.floor(iyintercept + iyslope * y);}}
```