John Gianakopoulos
4/16 Scribe Notes

Homework due next Wednesday. The last homework is due the following Monday.

Download R scripts Portfolio.R and MonteCarloIntro.R

- Load Mosaic Library
- Starting with a seemingly easy task. Calculating compound interest using a for loop to simulate scenarios in which this year's results depend on last year's results.
- 4 year investment horizon and starting wealth of 100. How much would I be worth in year 2?
    - Year 1: $100 (old wealth)
    - Interest (% Return): 5% or 0.05
    - New Wealth = (1 + 0.05) * Old Wealth = $105
    - Year 2:
    - Interest (% Return) : 0.05
    - New Wealth = (1+0.05) * Old Wealth = $110.25
- Reproduce this series of calculations using a **"for" loop**
    - Chains a sequence of calculations together
    - Line 9 of Rscript is the for loop part
        - For(year in 1:Horizon) {
          This Return = ReturnAvg
          Wealth = Wealth * (1+ThisReturn)
          }
        - New wealth = Old wealth * (1+Interest Rate)
        - Every year you calculate new wealth value

Running the Rscript you get a final wealth value of 703.9989. This can be verified using compound interest calculating programs online.

- It is important to bake in year-to-year volatility into returns. You cannot assume that you will get a 5% return every year. Assume that you have 5% volatility. This could be modeled by a bell curve or normal distribution with a standard deviation of 5%.

Rinput
# Now a risky asset with a positive expected return
ReturnAvg = 0.05
ReturnSD = 0.05
Horizon = 40

Wealth = 100
# Sweep through each year and update the value of wealth

```
for(year in 1:Horizon) {
        # Generate a random return
        ThisReturn = rnorm(1, ReturnAvg, ReturnSD)

        # Update wealth
        Wealth = Wealth * (1 + ThisReturn)
}
Wealth
```

This input is saying, simulate a random normal return with a standard deviation and mean of 5%. Apply update formula. Take old wealth and multiply it by (1+ThisReturn). Seeking to show how volatility in an interest rate impacts return year after year.

Repeat this simulation multiple times to get a sense of average or expected return of total wealth after 40 years if you invest in a portfolio where returns are on average 5% and volatility is 5%. This building block of code, which gets us one value, can be run 1000 times to get a range of values you might achieve after 40 years.

```
sim1 = do(1000)*{
        Wealth = 100
        # Sweep through each year and update the value of wealth
        for(year in 1:Horizon) {
                # Generate a random return
                ThisReturn = rnorm(1, ReturnAvg, ReturnSD)

                # Update wealth
                Wealth = Wealth * (1 + ThisReturn)
        }
        # Output the value of wealth for each simulated scenario
        Wealth
}
hist(sim1$result)
```

- Sim1 is 1000 rows where every entry is a draw from the distribution of possible final wealth after 40 years.
- Take the mean of the simulated values.
- By increasing the volatility to 10% you get a right skewed histogram

With a higher return, but also higher risk scenario (Return Avg = 0.08, Return SD = 0.12) there is a huge range of final outcomes.

A Monte Carlo simulation is taking a simulated outcome and repeating it multiple times to get an expected range of values.

To see the growth of your portfolio over time you use a placeholder. **Analogy:** To put 40 garments in a closet you must build a closet and have a hanger for all 40 pieces of clothing.

```
RunningWealth = rep(0, Horizon)  ← This is the closet of empty hangers

# Sweep through each year and update the value of wealth
for(year in 1:Horizon) {
    # Generate a random return
    ThisReturn = rnorm(1, ReturnAvg, ReturnSD)

    # Update wealth
    Wealth = Wealth * (1 + ThisReturn)

    # Save this year's wealth in the corresponding place in
RunningWealth
    RunningWealth[year] = Wealth
}
```

This will simulate 40 year investment career and show running wealth.
To see plot of running wealth you can just use the plot function in R.

Now, do this 1000 times. To plot individual simulations use the following code:

```
plot(1:Horizon, sim1[1,], type='l')
lines(1:Horizon, sim1[2,], type='l')
lines(1:Horizon, sim1[3,], type='l')
```

To find terminal wealth at a certain year use the following code:

```
hist(sim1[,40])
mean(sim1[,40])
```

```
    # Sweep through each year and update the value of wealth
    for(year in 1:Horizon) {
        # Generate a random return
        ThisReturn = rnorm(1, ReturnAvg, ReturnSD)

        # Update wealth
        Wealth = Wealth * (1 + ThisReturn)
    }
    # Output utility of wealth for this scenario
    log(Wealth)
}
```

Utility functions grow more slowly as you get more money. Log(x) has the same shape as this archetypal utility function so using log is representative of this. The

units of the simulation are now units of utility. It is useful to compare these units of utility with another simulation.

With a higher return and higher risk as well, expected utility increases from 6.52 → 7.57. i.e., the extra risk is compensated by the expected higher return.

- To make decisions based on utility functions this allows you to model various scenarios.

1. Compounding interest
2. Compounding interest with random rates
3. Monte Carlo simulation
4. Plotting trajectories
5. Expected utilities

In the previous simulation, we only consider one asset, which is unrealistic. Additionally, you assume knowledge on volatility and average returns. (The need to make explicit probabilistic assumptions on how this asset behaves year after year. Not necessarily described by a normal distribution.)

You must find a way to model a joint distribution. That technique is called Bootstrap resampling. (Sound familiar bro?)

Bootstrap Resampling is the legally required way for banks to quantify risk. We will learn this technique.

Use Portfolio.R script

You need a new library. Install package fImport into R.
fImport is a suite of functions and software tools to play with financial data.

Function: yahooSeries downloads data and takes closing price, trading volume, and other useful information and sticks it into a matrix for you.

```
# Import a few stocks
mystocks = c("ORCL", "JNJ", "WMT", "XOM", "MRK")
myprices = yahooSeries(mystocks, from='2009-01-01', to='2013-12-31')

# The first few rows
head(myprices)
```

- Must highlight bottom function computerreturns and hit command enter so that the rscript above will work.
- Now if you hit myreturns = computerreturns(myprices) then you will get a list of percentage returns.

The magic of Bootstrapping Sampling: For these 5 stocks every day there is a joint distribution. We can describe them, but it is difficult to put a mathematical model on it.

If you have samples of the joint distribution that bake in dependence structures of different stocks you can use bootstrapping.

Assume you have $10,000 of total wealth and that 20% of your wealth is in each of the five stocks listed on the Rscript.

```
# Simulate a one-day change in your portfolio
totalwealth = 10000
weights = c(0.2, 0.2, 0.2, 0.2, 0.2)
# What percentage of your wealth will you put in each stock?
```

Holdings calculates your portfolio:

```
holdings = holdings + holdings*return.today
```

Now, you want to build a model that simulates the growth or decline of portfolio over time and returns come from bootstrapped metrics.

Assumes that the past returns form a good model of joint distribution and we can use that information to model future returns.