554.488/688 Financial Computing I
Fall 2019
Homework #1
Due Tuesday Sept 10th 11:59PM

Introduction

This exercise is meant to be an easy and friendly one to get you started with Monte-Carlo simulation at a very basic level, and to develop a little bit of simple code, including the creation of user-defined functions in Python. Before proceeding to the actual exercises, let's go over some basic concepts and code.

We use the numpy package to produce pseudo-random numbers. We'll see more advanced uses later, but for now, the following Python code produces a numpy array (which we can later treat as a vector) whose entries are a realization of a sequence of Bernoulli random variables with success probability $p = .75$. We will not need the array properties of the `np.random.choice()` output, so we immediately convert the array to a list.

```
import numpy as np
L=list(np.random.choice(2,100,p=[.25,.75]))
print(L)
```

Here is an example of how we attack a particular problem using Monte-Carlo simulation. Suppose for Bernoulli trials X_1, X_2, X_3 with success probability $p = 0.5$ we want to know the expected value of the random variable $Y = (1 + X_1)/(1 + X_2 + X_3)$. Here is a Python program for getting an approximation to $E(Y)$ and $\mathrm{Var}(Y)$ based on $N = 100,000$ trials. This is just for illustrative purposes. The calculation can easily be done exactly since we can easily find the probability mass function of Y.

```
import numpy as np
N=100000
Yvalues=[]
for i in range(N):
    X=np.random.choice(2,3,[.5,.5])
    Y=(1+X[0])/(1+X[1]+X[2])
    Yvalues.append(Y)
Yvalues=np.array(Yvalues)
print("estimated mean = " + str(np.mean(Yvalues)))
print("estimated variance = " + str(np.var(Yvalues)))
```

In the code above, we use the `for i in range(N):` to repeatedly carry out our sampling N times. In the block of code associated with the `for` loop, we generate X which is, by default an np.array. We calculate the Y value for this particular Bernoulli trial sequence, and append that value to our list of Y values. Once the `for` loop is complete, we want to calculate the sample mean and sample variance for that list of sampled Y values. There is not a function that calculates the mean and variance of a *list*, but we can create a numpy array from that list `Yvalues=np.array(Yvalues)`, then we can use the numpy `mean` and `var` functions to get the mean and variance.

We can also use Monte-Carlo to approximate a probability. For example, what is the chance that Y defined above, is at least 1. We can use this code.

```
import numpy as np
N=100000
EventIndicators=[]
for i in range(N):
    X=np.random.choice(2,3,[.5,.5])
    Y=(1+X[0])/(1+X[1]+X[2])
    if Y>=1:
        EventIndicators.append(1)
    else:
        EventIndicators.append(0)
EventIndicators=np.array(EventIndicators)
print("estimated probability = " + str(np.mean(EventIndicators)))
```

Note that if we know p we can estimate the variance of our event indicator using $p(1-p)$.

Make sure you understand the above code before proceeding.

The Exercises

A *run* in a binary list is a maximal sequence of contiguous equal values. For example, in the list $[0,0,1,1,1,1,0,0,1,1,1,0,0,0,0,0,0,1]$ we have the following runs, 2 0's, 4 1's, 2 0's, 3 1's, and 6 0's, and 1 1. The sequence of run lengths is therefore 2,4,2,3,6,1.

Here is *pseudo-code* for computing the longest run in a binary sequence. Make sure you understand this code.

```
input: a list of binary values b[0],b[1],...,b[n-1]
output: the length of the longest run
```

```
Set v=b[0]
Set runlength=1
Set maxrunlength=0

For i=1,...,n-1 do
    If b[i]=v
        Set runlength=runlength+1
    Else
        if runlength>maxrunlength
            Set maxrunlength=runlength
        Set v=b[i]
        Set runlength=1
If runlength>maxrunlength
    Set maxrunlength=runlength
output maxrunlength
```

And here is the corresponding Python code for a function that takes a binary list as input and computes the longest run.

```
def longest_run(L):
    N=len(L)
    v=L[0]
    runlength=1
    maxrunlength=0
    for i in range(1,N):
        if L[i]==v:
            runlength+=1
        else:
            if runlength>maxrunlength:
                maxrunlength=runlength
            v=L[i]
            runlength=1
    if runlength>maxrunlength:
        maxrunlength=runlength
    return(maxrunlength)
```

Here is an example of use of this function:

```
L=[1,0,1,1,1,0,0,1,1,1,1,1,0,0,0,1,0,1,1,0,0,0,1]
print(longest_run(L))
```

The following should be provided in a Python notebook.

Important: Please copy numerical results into a text cell so that the grader does not have to run your code to see what the results are.

1) For a sequence of 100 Bernoulli random variables with success probability $p = 0.5$ use Monte-Carlo simulation with 100,000 trials to estimate the expected value and variance of the longest run. Do the same for $p = 0.55$. In your notebook, please provide code that runs but put your answers in a text box.

2) Revise the longest run function so that instead of outputting the length of the longest run, the output is the list of run lengths (in the order of appearance) as described above. Call this function: *runlengths*.

3) For a sequence of 100 Bernoulli random variables with success probability $p = 0.5$ use Monte-Carlo simulation with 100,000 trials to estimate the chance that at least three runs of length 6 occur.