Author: Milesh B
Github: https://github.com/milesh-b

Vanilla Put option pricing using Monte Carlo in AWK.

The main advantage of using AWK scripting over more popular programming languages like Python or R is execution speed. Programming in AWK does indeed take a considerable investment of time but its definitely worth it especially for those who work with “Big Data”.

Demo.

- We need to first make the “Euro_Binomial.sh” executable by running the terminal command:

```
chmod u+x *.sh.
```

- As an example, consider the following input parameters:
  - s0=144
  - vol=0.315573
  - r=0.19062
  - x=144
  - div=0
  - time=1.5
  - tree=97
  - sim=50000

- We are also going to time how long the script takes to run:

```
$ time ./Euro_Binomial.sh 144 0.315573 0.19062 144 0 1.5 97 50000
```

```
You have inputted the following parameters:
s0= 144
vol= 0.315573
r= 0.19062
x= 144
div= 0
time= 1.5
tree= 97
sim= 50000
Value of option=  6.44179

real    0m13.088s
user    0m7.249s
sys     0m1.283s
```

From the above, the value of the vanilla put option is 6.44179. The script took approximately 7.249 seconds to execute.

Comparing this with the equivalent Python script (MCExotics):

```
import MCExotics as Exotics #importing the calculator
import time
start=time.time()
derivative=Exotics.MCEuropean(144,0.315573,0.19062,144,0,1.5,97,50000) #initialization
derivative.put() #value of the option
end=time.time()
print(end-start)
```

```
Put option value: 6.32
 Input parameters:
s0=144.0
vol=0.315573
r=0.19062
x=144.0
div=0.0
time=1.5
tree=97
sim=50000
54.52375912666321
```

From the above, the vanilla put option is 6.32. The script took approximately 54.524 second to execute.

Conclusion.

Using Monte Carlo simulations are indeed powerful techniques for solving mathematical problems. One major disadvantage of this method is that it requires many simulations to converge depending on the problem at hand. As the number of simulations increase, so does the computation time and therefore the analyst needs to consider other methods that would speed up the computation time. In this simple example, the AWK script took 7.249 seconds while the python script took 54.524 seconds which illustrates the power of AWK especially when the scales are magnified beyond this simple illustration.