Monte Carlo Simulation applied to Portfolio Management

Posted by Niko G. on April 1, 2019

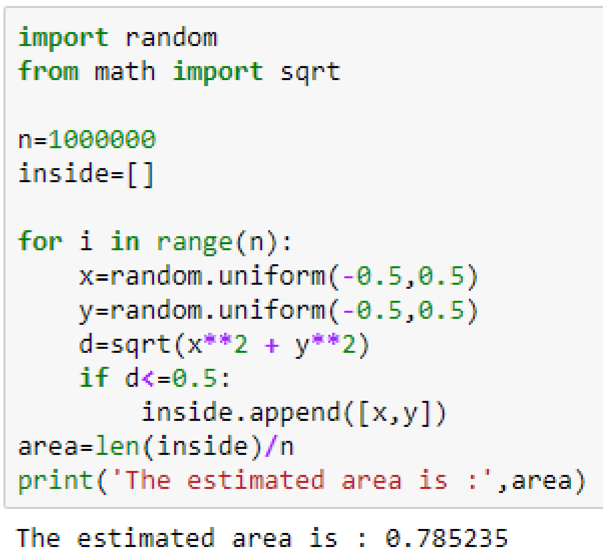
“To reduce risk it is necessary to avoid a portfolio whose securities are all highly correlated with each other. One hundred securities whose returns rise and fall in near unison afford little protection than the uncertain return of a single security.” - *Harry Markowitz*

A basic understanding of Monte Carlo simulations

Monte Carlo simulations perform risk analysis by simulating models of possible outcomes according to a chosen probability distribution for a parameter that has an inherent uncertainty. To get a basic understating of how it works we can use a simple example and calculate the area of a circle inside a square with a binary outcome experiment. Let say that we don’t know the value of Pi and want to calculate the area of a circle of diameter of one. We can draw a square around that circle and start randomly plotting dots inside that square. In this case we would be following a uniform distribution for the coordinates of the dots we are plotting. If the dot falls inside the circle we will label it a success and at the end we will estimate the area of the circle by dividing the number of dots inside the square by the total number of dots plotted.

As we will see in the illustration below, the number of dots plotted is crucial to obtaining an accurate result. If we don’t plot enough dots the value is likely to be far from the actual value but with a large number of random values it appears very likely that we will get a number very close to the right answer. Python allows us to generate more than one hundred thousand outcomes in a significantly short period of time which makes it a powerful tool to run Monte Carlo simulations. By using the formula A = πr^2 the area of a circle with a diameter of one is approximately 0.785. Let’s use Python to code the experiment and see how close to this value we get with different numbers of random dots or different number of paths to say it in Monte Carlo words.

All we are going to do is go through a loop of ‘n’ number of paths and for each iteration we will generate two random values between -0.5 and 0.5 because our circle is centered on the origin. The first of this values will be the x axis value for our dot and the second one will be the y axis value. Using this coordinates we calculate the Euclidian distance of the dot to the origin. If the distance is inferior or equal to 0.5, our dot is in the circle, otherwise it is outside. Finally we divide the number of dots inside the circle by ‘n’ to calculate the area. The following few lines of code are enough to run the entire experiment.



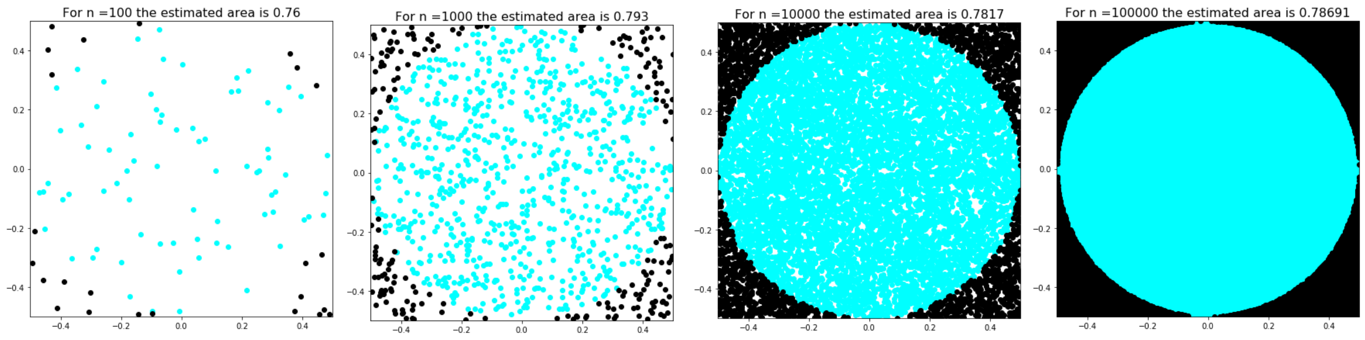
```
import random
from math import sqrt

n=1000000
inside=[]

for i in range(n):
    x=random.uniform(-0.5,0.5)
    y=random.uniform(-0.5,0.5)
    d=sqrt(x**2 + y**2)
    if d<=0.5:
        inside.append([x,y])
area=len(inside)/n
print('The estimated area is :',area)
```

```
The estimated area is : 0.785235
```

As shown above, one million paths give us a very accurate estimate of the areas, but let’s use matplotlib to have a look at what it took to get there by plotting all the random dots for different values of the number of paths “n”.



Applying Monte Carlo simulations to Portfolio Management

Now that we have a good basic understanding of how it works, I am sure that calculating the area of a circle or even estimating the value of PI is not what most people reading this blog are looking to achieve. How can we apply Monte Carlo simulation to Data Science problems? Because of my background I will chose the finance industry to provide an example.

It is well known that the stock market exhibits very high dimensionality due to the almost unlimited number of factors that can affect it which makes it an almost random variable, very difficult to model and predict. One relatively straightforward application of Monte Carlo in finance is optimization of a portfolio of stocks. Portfolios are built based on the Modern Portfolio Theory (MPT) by Harry Markowitz’s groundbreaking, Nobel Prize winning paper “Portfolio Selection” in 1952.

I recommend to anybody interested in finance and portfolio theory to read about Modern Portfolio Theory by Markowitz but for the purposes of this blog we will keep it very simplistic. When picking a portfolio of stocks, you may be willing to take on different levels of risk depending on your goals. And here risk simply means standard deviation of returns. But regardless of your willingness to accept risk, you can maximize your returns by picking the combination of stocks that has the highest returns for this given level of risk or volatility. From here Markowitz’s idea is very intuitive: it only makes sense to invest in the portfolios, combinations of stocks, that have the highest returns for each level of risk and we are going to use a Monte Carlo simulation to simulate and plot thousands of portfolios and pick the ones with the highest returns by plotting returns against standard deviation. These portfolios are called by the author “The Efficient Frontier”.

To illustrate the efficient frontier concept we will use four of the most common tickers in the Communication Services and Information Technology sectors: Netflix, Apple, Facebook and Google Inc.

The idea is to use pandas to get historical prices for each of the above tickers and calculate daily returns and standard deviation. Then we will use a Monte Carlo simulation to assign random weights to each of these tickers in our portfolio (the sum of which will add to one) and plot the returns of each the portfolio against standard deviation. Note that from a return perspective, each stock contributes to the portfolio its daily returns times its weight in the portfolio. The formula for the standard deviation of the portfolio is explained here: https://en.wikipedia.org/wiki/Modern_portfolio_theory (https://en.wikipedia.org/wiki/Modern_portfolio_theory). The Efficient Frontier is the line drawn by the portfolios with the highest returns for each level of risk.

Let’s have a look at the code to run this Monte Carlo simulation and plot Markowitz Efficient Frontier.

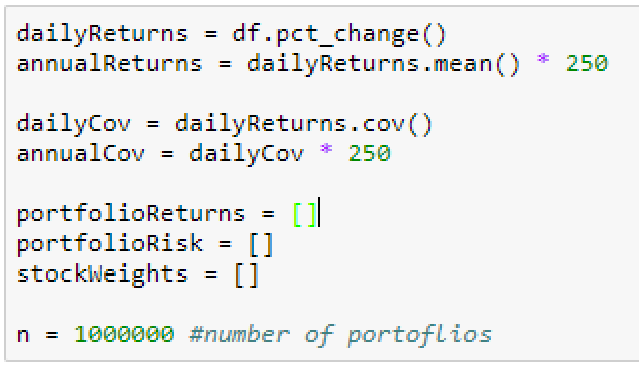
The First step is to import the libraries we will need:

```
import pandas_datareader.data as web
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

We then move forward to pick our tickers and create the data frame with pandas_datareader:

```
tickers = ['NFLX', 'AAPL', 'FB', 'GOOG']
df= web.DataReader(tickers,'yahoo','2015-01-01','2019-09-29')['Adj Close']
```

Let’s now calculate returns and covariance of these returns as well as create an empty list to store data from our Monte Carlo simulation and pick the number of paths, in this case one million.



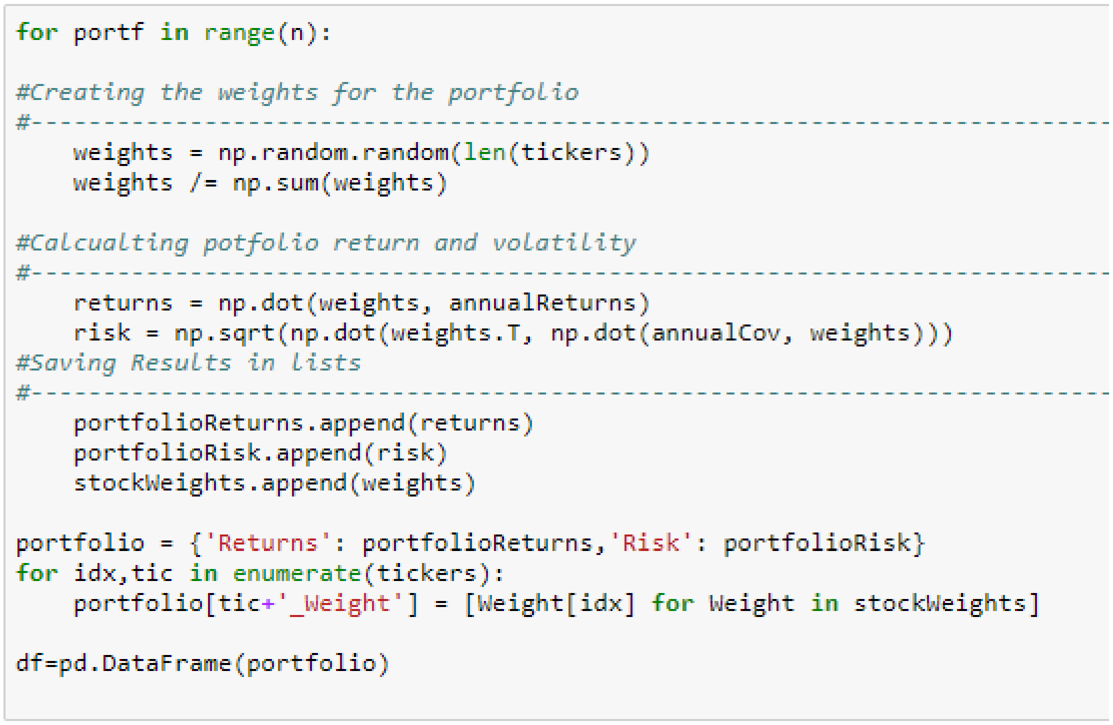
```
dailyReturns = df.pct_change()
annualReturns = dailyReturns.mean() * 250

dailyCov = dailyReturns.cov()
annualCov = dailyCov * 250

portfolioReturns = []
portfolioRisk = []
stockWeights = []

n = 1000000 #number of portoflios
```

We are now ready to run the Monte Carlo Simulation for one million different portfolios:



```
for portf in range(n):

#Creating the weights for the portfolio
#--------------------------------------------------------------------------
    weights = np.random.random(len(tickers))
    weights /= np.sum(weights)

#Calcualting potfolio return and volatility
#--------------------------------------------------------------------------
    returns = np.dot(weights, annualReturns)
    risk = np.sqrt(np.dot(weights.T, np.dot(annualCov, weights)))
#Saving Results in lists
#--------------------------------------------------------------------------
    portfolioReturns.append(returns)
    portfolioRisk.append(risk)
    stockWeights.append(weights)

portfolio = {'Returns': portfolioReturns,'Risk': portfolioRisk}
for idx,tic in enumerate(tickers):
    portfolio[tic+'_Weight'] = [Weight[idx] for Weight in stockWeights]

df=pd.DataFrame(portfolio)
```

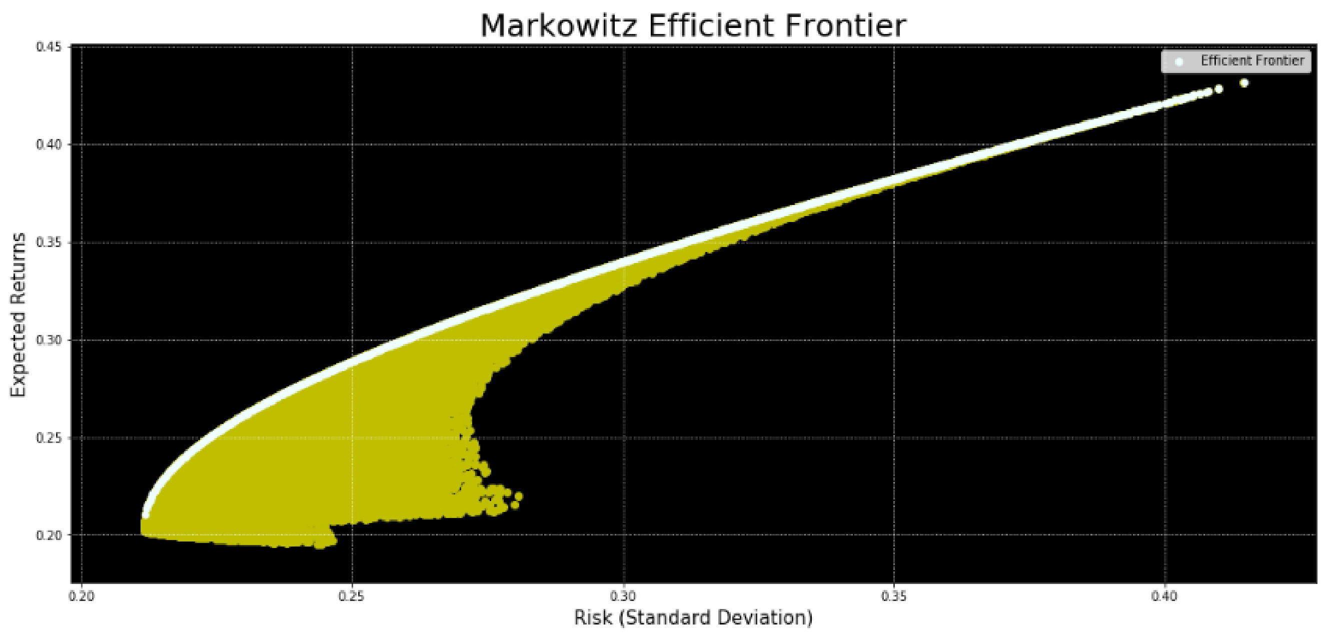
And finally some code matplotlib to plot all portfolios in one color and the efficient frontier in another color:

```
#Creating lists wiht data for Efficient Frontier
#-------------------------------------------------------------------------------------
data=pd.concat([df['Returns'],df['Risk']],axis=1)
beg=data['Risk'].min()
end=data['Risk'].max()
step = data['Risk'].min()*(1/700)
step

X,Y=[],[]
for i in range(700):
    y=data['Returns'][data['Risk']>=beg+(i*step)][data['Risk']<=beg+(i+1)*step].max()
    x=(beg+(i*step) + beg+(i+1)*step)/2
    X.append(x)
    Y.append(y)
```

```
#Plotting
#-------------------------------------------------------------------------------------
fig = plt.figure(figsize = (18,8))
ax = plt.subplot2grid((4,4),(0,0), rowspan =4, colspan = 4, facecolor='k')
ax.scatter(df['Risk'],df['Returns'],color ='y', label ='')
ax.scatter(X,Y, color = 'azure', s =30, label = 'Efficient Frontier')
ax.grid(True, color = 'w', linestyle = ':')
plt.xlabel('Risk (Standard Deviation)',fontsize =15)
plt.ylabel('Expected Returns',fontsize =15)
plt.legend()
plt.title('Markowitz Efficient Frontier',fontsize =25)
```

This is the resulting Graph:



This graph shows all the portfolios resulting from the one million randomly generated combinations of weights for our four stocks. For example we can see that there is a very large number of portfolios that will exhibit a risk of 25% but expected returns for these portfolios will range from 21% to 29%. The portfolio with 29% expected return for a level of risk of 25% is located on the white line along with every other portfolio that has the highest expect return for each given level of risk. This white line is called the Efficient Frontier.

← **PREVIOUS POST (/WEB_SCRAPING_YAHOO_FINANCE)**

NEXT POST → (/MONITORING_TOXIC_AND_HATE_SPEECH_IN_ONLINE_COMMENTS)

● (https://learn.co/ganevniko)

● (https://github.com/ganevniko)