Monte Carlo Simulation in Python

Rohit Garg (Email: f2005636@gmail.com)

1. Monte Carlo Simulation

In a Monte Carlo simulation, a random value is selected for each of the tasks based on the range of estimates. The model is calculated based on this random value. The result of the model is recorded and the process is repeated.

A typical Monte Carlo simulation calculates the model hundreds or thousands of times, each time using different randomly-selected values. The completed simulation yields a large results pool with each result based on random input values. These results are used to describe the likelihood, or probability, of reaching various results in the model.

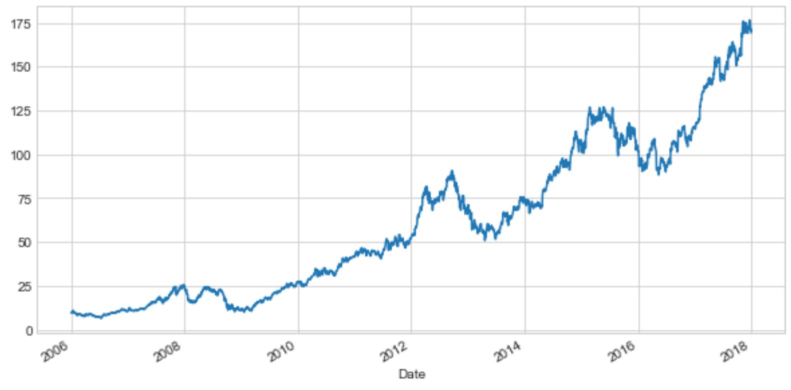
A Monte Carlo Simulation yields risk analysis by generating models of possible results through substituting a range of values (a probability distribution) for any factor that has inherent uncertainty. The Monte Carlo method is based on the generation of multiple trials to determine the expected value of a random variable

2. Python Codes

For this exercise the following modules are used: quandl, numpy, pandas, scipy.stats, and matplotlib.pyplot

2.1 Get the Data

- Download the data for Apple ('AAPL') for the period '2006-1-1' to '2017-12-31'
  data = AAPL = quandl.get('WIKI/AAPL',start_date='2006-01-01',end_date='2017-12-31')['Adj. Close']



- Obtain the log returns of Apple for the designated period.
  log_returns = np.log(1 + data.pct_change())

2.2 Calculations

- u - the mean value of the log returns
  var - the variance to the log returns
  std - standard deviation of the log returns
  u = log_returns.mean()
  var = log_returns.var()
  stdev = log_returns.std()

- Calculate the drift, using the following formula: drift = u - 0.5 * var
  *drift = u - (0.5 * var)*

2.3 Forecasting Future Stock Prices

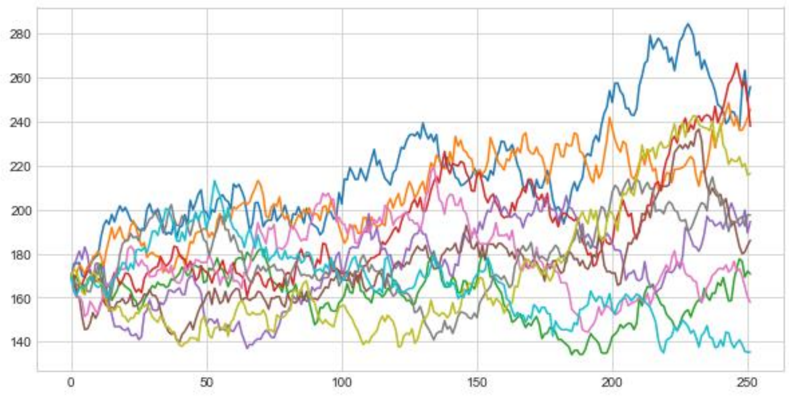
- Forecast future stock prices for every trading day a year ahead (assign 252 to “t_intervals”)
  Examine 10 possible outcomes (assign 10 to “iterations”)
  ```
  t_intervals = 252
  iterations = 10
  ```

- Using the below formula to calculate daily returns:
  daily returns = e(drift + stdev * Z)
  Z = norm.ppf(np.random.rand(t_intervals, iterations))
  ```
  daily_returns = np.exp(drift + stdev * norm.ppf(np.random.rand(t_intervals, iterations)))
  ```

- S0 - the last adjusted closing price of Apple
  price_list - the same dimension as the daily_returns matrix
  Set the values on the first row of the price_list array equal to S0
  ```
  S0 = data.iloc[-1]
  price_list = np.zeros_like(daily_returns)
  price_list[0] = S0
  ```

- Create a loop in the range (1 to t_intervals) that reassigns to the price in time t the product of the price in day (t-1) with the value of the daily returns in t
  ```
  for t in range(1, t_intervals):
      price_list[t] = price_list[t - 1] * daily_returns[t]
  ```

- Plots
  ```
  plt.figure(figsize=(10,5))
  plt.plot(price_list)
  ```


280
260
240
220
200
180
160
140
0
50
100
150
200
250