Stock Market Analysis

Gaurav Goel
CSC - 9010

Project Description

- In this project I would be looking at data from the stock market, particularly some technology stocks. I will learn how to use pandas to get stock information, visualize different aspects of it, and finally will look at a few ways of analyzing the risk of a stock, based on its previous performance history. I will use Monte Carlo method! to predict future stock prices

Problem Statement

- What is the change in price of the stock over time?
- What is the daily return of the stock on average?
- What is the moving average of the various stocks?
- What is the correlation between different stocks' closing prices ?
- What was the correlation between different stocks' daily returns ?

Problem Statement contd.

- How much value do we put at risk by investing in a particular stock?
- How can we attempt to predict future stock behavior?
- Some outperforming stocks in given sector

Data setup

- For reading stock data from yahoo we use

pandas.io.data import DataReader

globals()[stock] =
DataReader(stock,'yahoo',start,end) where end is given by

end = datetime.now()

and start = datetime(end.year - 1,end.month,end.day)

Data setup contd.

- To get the value of the stock at any time , we use
- Summary Stats -->AAPL.describe()
- General Info ---->AAPL.info()
- For calculating closing price of AAP → AAPL['Adj Close']
- For calculating Volume of AAPL→ AAPL['Volume']
- For calculating rolling_mean
- For calculating 'Daily Return use AAPL['Daily Return'] = AAPL['Adj Close'].pct_change()

Data setup contd.

- To get the value of the different stocks, we create a dataframe closing_df = DataReader(['AAPL','GOOG','MSFT','AMZN'],'yahoo',start,end)['Adj Close‘
- tech_rets = closing_df.pct_change()] gives us the returns associated with any stock

Analysis

- We plot the values on the graph

We'll use joinplot to compare the daily returns of Google and Microsoft

sns.jointplot('GOOG','MSFT',tech_rets,kind='scatter')

The advantage of seaborn package is that it gives us the value of Pearson's coeffecient of Co-relation

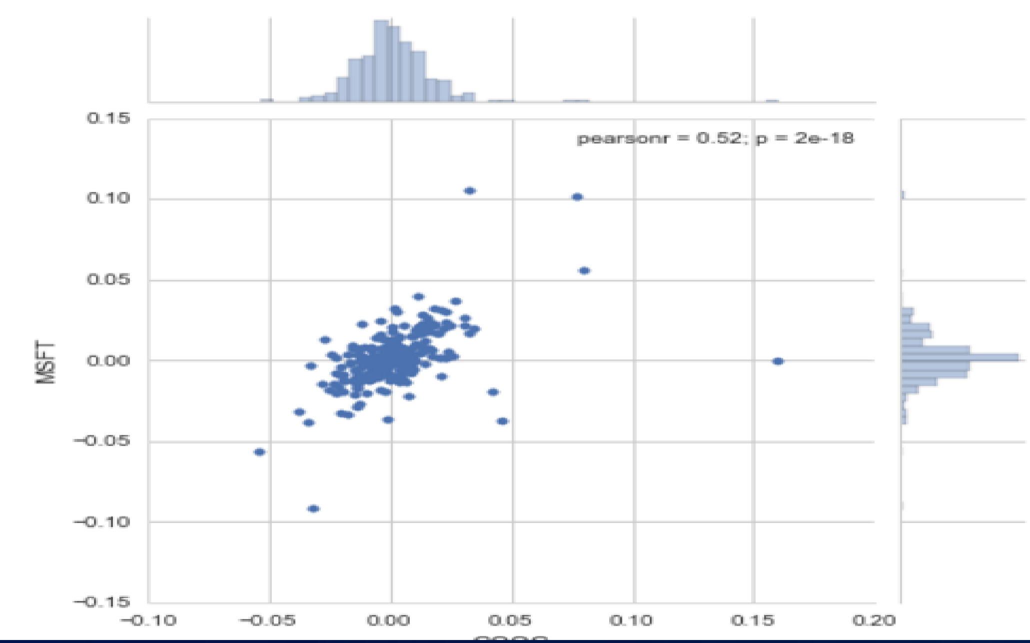
sns.corrplot(tech_rets.dropna(),annot=True)

Analysis

The advantage of seaborn package is that it gives us the value of Pearson's coeffecient of Co-relation

sns.jointplot('GOOG','MSFT',tech_rets,kind='scatter')

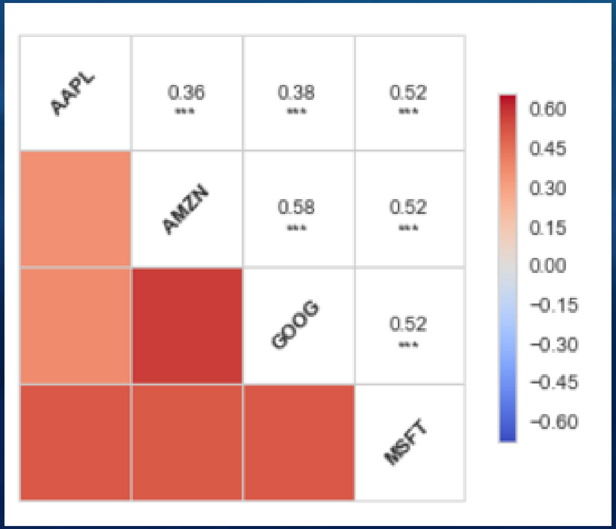
```
Out[13]:  <seaborn.axisgrid.JointGrid at 0x19a79e80>
```



Analysis

sns.corrplot(tech_rets.dropna(),annot=True) gives the below graph which indicates that the coeffecient Of corelation is 0.58 which is maximum between Amazon

and Google.



Risk Analysis

One of the most basic ways using the information we've gathered on daily percentage returns is by comparing the expected average return of the year with the standard deviation of the daily returns.

```
rets = tech_rets.dropna()
#print rets
print rets.mean()
print rets.std()
```

Algorithm Explanation –Monte Carlo Analysis

We define a value at risk parameter for our stocks. We can treat value at risk as the amount of money we could expect to lose (aka putting at risk) for a given confidence interval. There are several methods we can use for estimating a value at risk. Let's go ahead and see some of them in action.

```
# The 0.05 empirical quantile of daily returns
rets['AAPL'].quantile(0.05)
```

Algorithm Explanation –Monte Carlo Analysis

The 0.05 empirical quantile of daily returns is at -0.026. That means that with 95% confidence, our worst daily loss will not exceed 2.6%. If we have a 1 million dollar investment, our one-day 5% VaR is 0.026 * 1,000,000 = $26,000.

Algorithm Explanation –Monte Carlo Analysis

It is a computerized mathematical technique that allows people to account for risk in quantitative analysis and decision making. The technique is used by professionals in such widely disparate fields as finance,construction etc.

Probabilistic Results. Results show not only what could happen, but how likely each outcome is.

Algorithm Explanation –Monte Carlo Analysis contd.

To design a better process, we would need to collect a mountain of data in order to determine how input variability relates to output variability under a variety of conditions. However, if we understand the typical distribution of the input values and we have an equation that models the process, we can easily generate a vast amount of simulated input values and enter them into the process equation to produce a simulated distribution of the process outputs. Probabilistic Results. Results show not only what could happen, but how likely each outcome is.

Algorithm Explanation –Monte Carlo Analysis contd.

- **Value at Risk using the Monte Carlo method**

Using the Monte Carlo to run many trials with random market conditions, then we'll calculate portfolio losses for each trial. After this, we'll use the aggregation of all these simulations to establish how risky the stock is.

It uses geometric Brownian motion (GBM)

$\Delta S/S=\mu\Delta t+\sigma\epsilon(\mathrm{sqrt}\Delta t)$

Algorithm Explanation –Monte Carlo Analysis contd.

Where S is the stock price, mu is the expected return (which we calculated earlier),sigma is the standard deviation of the returns, t is time, and epsilon is the random variable.

The first term is known as "drift", which is the average daily return multiplied by the change of time. The second term is known as "shock", for each item period the stock will "drift" and then experience a "shock" which will randomly push the stock price up or down.

Algorithm Explanation –Monte Carlo Analysis

By simulating this series of steps of drift and shock thousands of times, we can begin to do a simulation of where we might expect the stock price to be.

Algorithm Explanation –Monte Carlo Analysis

```
# Set up our time horizon
days = 365

# Now our delta
dt = 1/days

# Now we grab our mu (drift) from the expected return data we got for AAPL
mu = rets.mean()['GOOG']

# Now  we grab the volatility of the stock from the std() of the average return
sigma = rets.std()['GOOG']
```

Algorithm Explanation –Monte Carlo Analysis

```
def stock_monte_carlo(start_price,days,mu,sigma):
    ''' This function takes in starting stock price, days of simulation,mu,sigma, and returns simulated price array'''

    # Define a price array
    price = np.zeros(days)
    price[0] = start_price
    # Schok and Drift
    shock = np.zeros(days)
    drift = np.zeros(days)

    # Run price array for number of days
    for x in xrange(1,days):

        # Calculate Schock
        shock[x] = np.random.normal(loc=mu * dt, scale=sigma * np.sqrt(dt))
        # Calculate Drift
        drift[x] = mu * dt
        # Calculate Price
        price[x] = price[x-1] + (price[x-1] * (drift[x] + shock[x]))

    return price
```

Algorithm Explanation –Monte Carlo Analysis

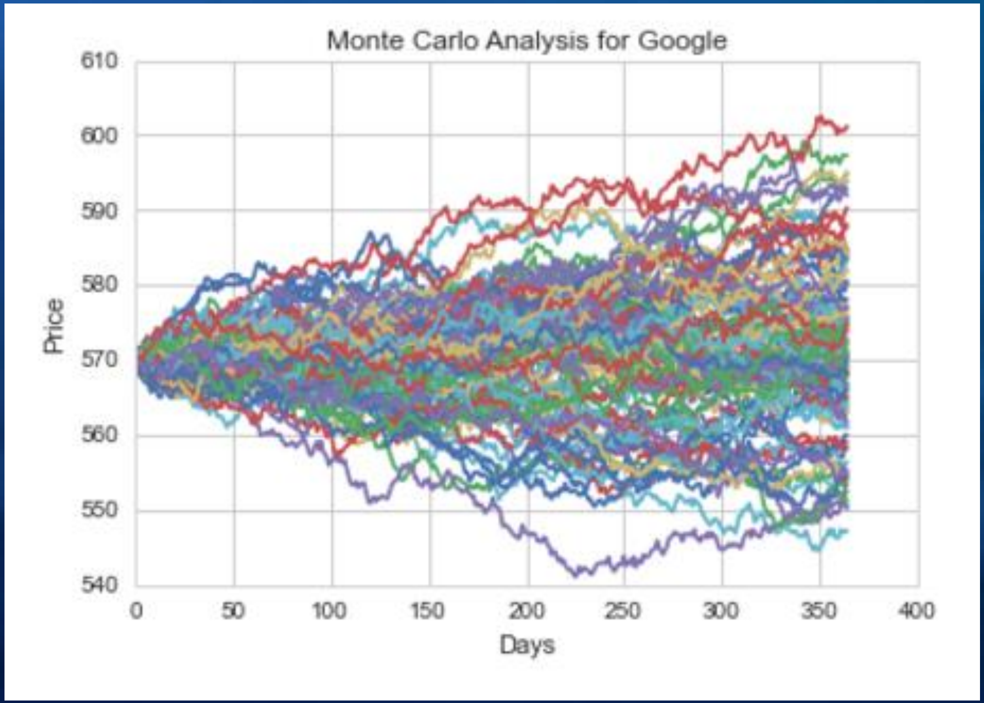
```
# Get start price from GOOG.head()

start_price = 569.85

for run in xrange(100):

plt.plot(stock_monte_carlo(start_price,days,mu,sigma))
plt.xlabel("Days")
plt.ylabel("Price")
plt.title('Monte Carlo Analysis for Google')
```

Algorithm Explanation –Monte Carlo Analysis contd.



Algorithm Explanation –Monte Carlo Analysis

```
# Set a large numebr of runs
runs = 10000

# Create an empty matrix to hold the end price data
simulations = np.zeros(runs)

# Set the print options of numpy to only display 0-5 points from an array
to suppress output
np.set_printoptions(threshold=5)

for run in xrange(runs):
    # Set the simulation data point as the last stock price for that run
    simulations[run] =
stock_monte_carlo(start_price,days,mu,sigma)[days-1];
```

Algorithm Explanation –Monte Carlo Analysis

Now that we have our array of simulations, we can go ahead and plot a histogram ,as well as use quantile to define our risk for this stock.

Algorithm Explanation –Monte Carlo Analysis contd.

```
# Now we'lll define q as the 1% empirical qunatile, this basically means that 99% of the values should fall between here
q = np.percentile(simulations, 1)

# Now let's plot the distribution of the end prices
plt.hist(simulations,bins=200)

# Using plt.figtext to fill in some additional information onto the plot

# Starting Price
plt.figtext(0.6, 0.8, s="Start price: $%.2f" %start_price)
# Mean ending price
plt.figtext(0.6, 0.7, "Mean final price: $%.2f" % simulations.mean())

# Variance of the price (within 99% confidence interval)
plt.figtext(0.6, 0.6, "VaR(0.99): $%.2f" % (start_price - q,))

# Display 1% quantile
plt.figtext(0.15, 0.6, "q(0.99): $%.2f" % q)

# Plot a line at the 1% quantile result
plt.axvline(x=q, linewidth=4, color='r')

# Title
plt.title(u"Final price distribution for Google Stock after %s days" % days, weight='bold');
```

Algorithm Explanation –Monte Carlo Analysis contd.

Results and Review

- We plotted the change in price of the stock over time for eg Google,Apple etc.
- We calculated the daily return of the stock on average for eg Google,Apple etc
- We calculated the moving average of the various stocks ..for eg Google,Apple etc
- We calculated the correlation between different stocks closing prices.
- We calculated the correlation between different stocks' daily returns ie Google and Amazon.

Bibliography

- http://blog.minitab.com/blog/adventures-in-statistics/understanding-monte-carlo-simulation-with-an-example.
- https://simplypython.wordpress.com/2014/12/24/basic-stock-technical-analysis-with-python/
- http://nakamuraseminars.org/nsblog/2014/06/21/monte-carlo-in-python-an-example