Financial Modelling
&
Simulation

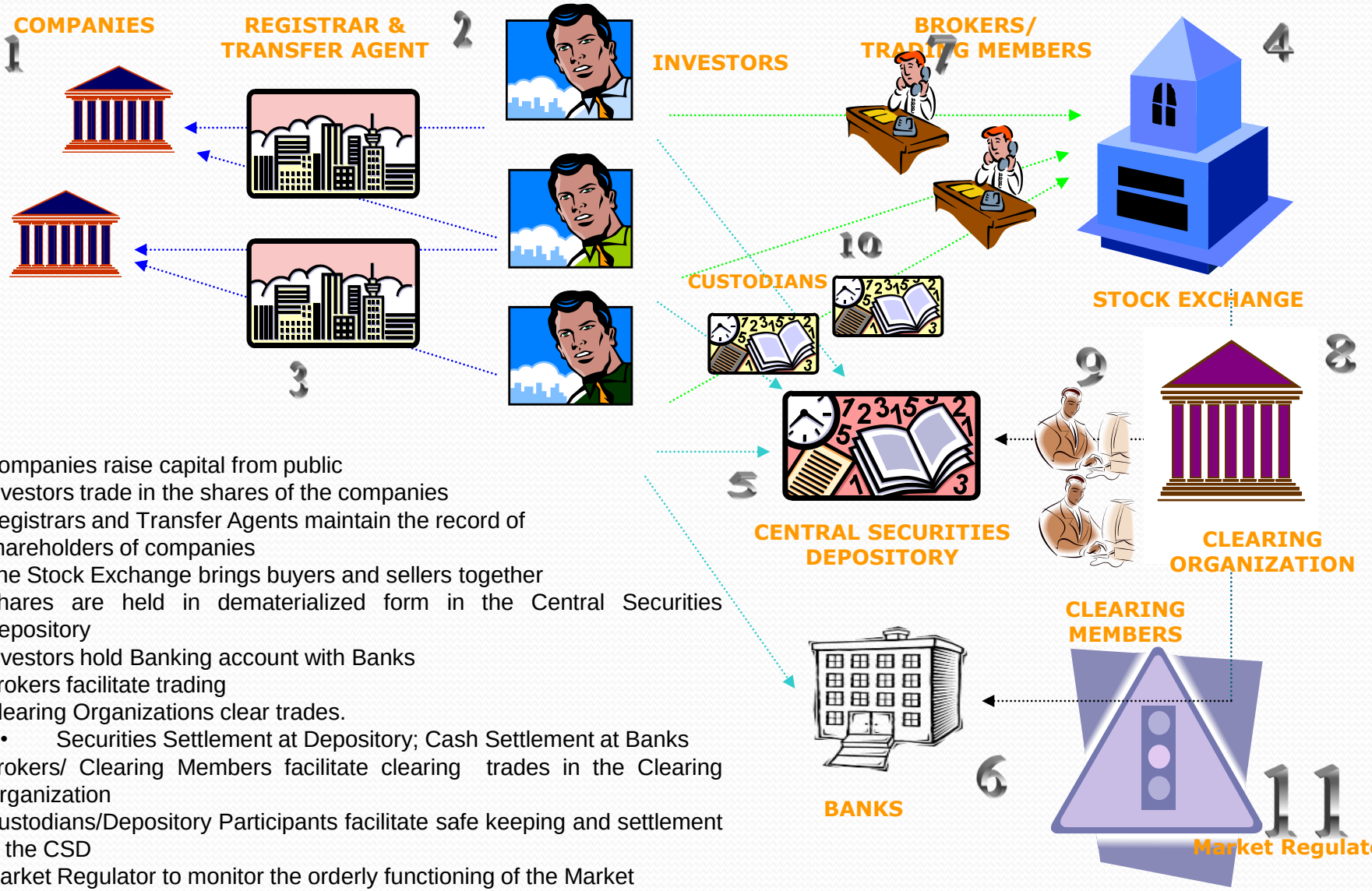
using Numpy, Scipy, Matplotlib and Pandas

-By Saurabh Jaiswal

Agenda

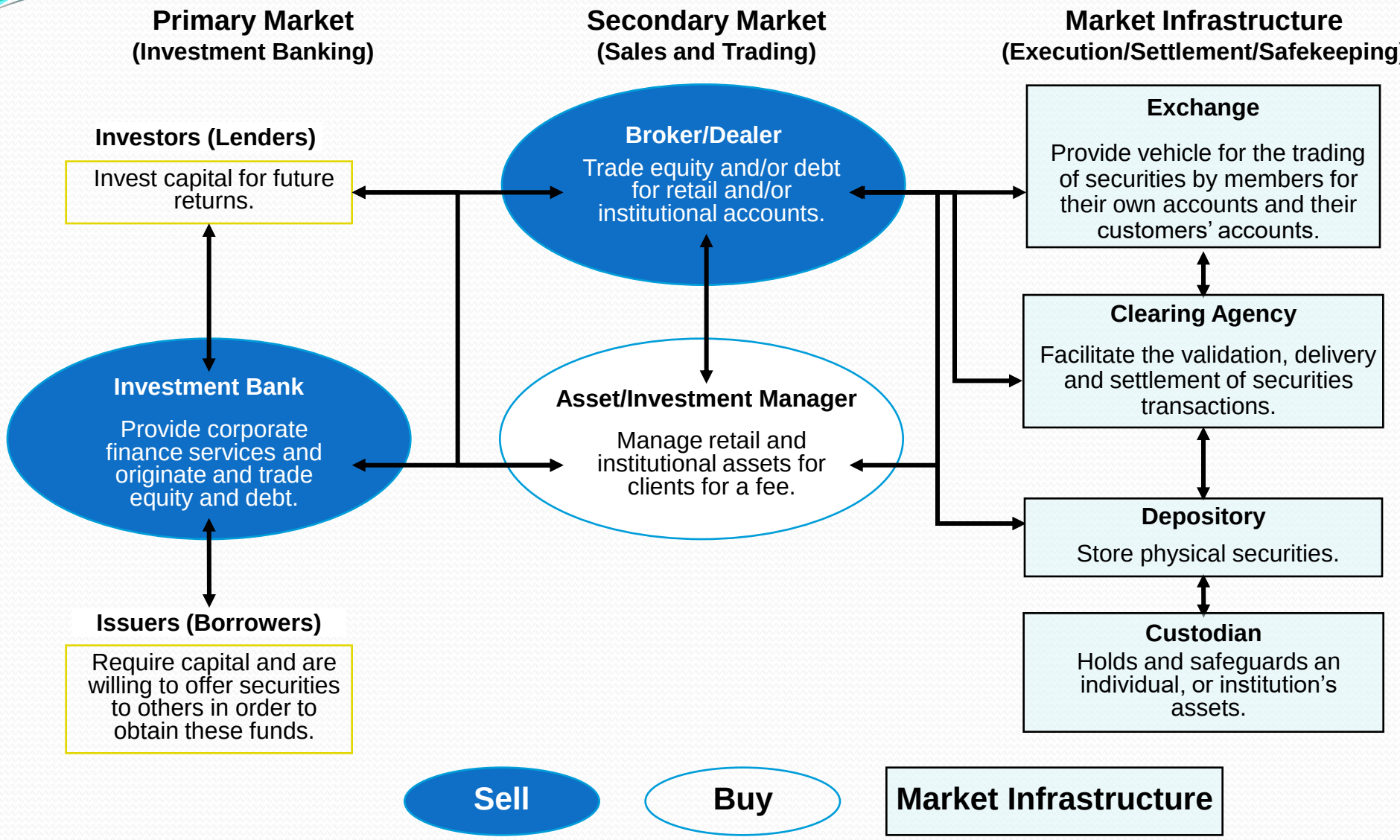
- Capital Markets Overview
- Python for Finance
- Black & Scholes for European Call Options
- Monte Carlo simulation for European Call Options
- VaR using Monte Carlo Simulation
- Geometric Brownian Motion
- Basics of Volatility and Normal function
- Basics of Correlation, Covariance and VaR
- Credit Valuation Adjustment (CVA)

Understanding of system flow



1. Companies raise capital from public
2. Investors trade in the shares of the companies
3. Registrars and Transfer Agents maintain the record of shareholders of companies
4. The Stock Exchange brings buyers and sellers together
5. Shares are held in dematerialized form in the Central Securities Depository
6. Investors hold Banking account with Banks
7. Brokers facilitate trading
8. Clearing Organizations clear trades.
   - Securities Settlement at Depository; Cash Settlement at Banks
9. Brokers/ Clearing Members facilitate clearing trades in the Clearing Organization
10. Custodians/Depository Participants facilitate safe keeping and settlement in the CSD
11. Market Regulator to monitor the orderly functioning of the Market

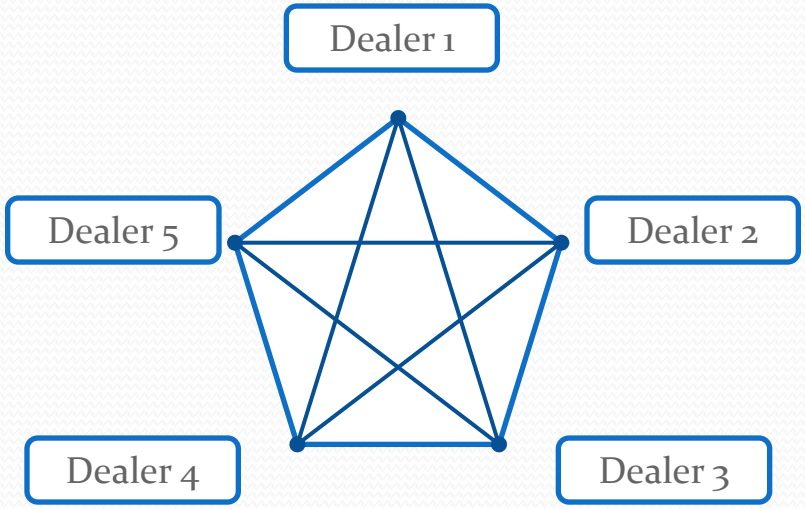
How they are Integrated



Bilateral Vs Centrally Cleared Markets – Netting concept

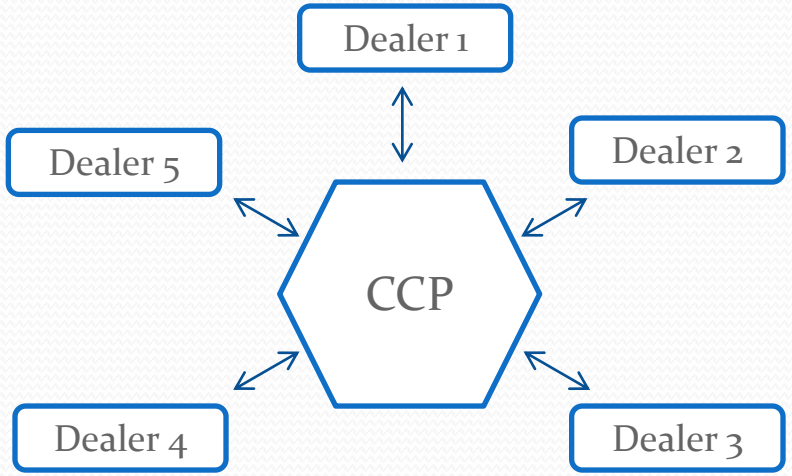
Bilateral Market

- Dealers book trades with each other
- Participants have direct credit exposure to each other



Centrally Cleared Market

- Each dealer books trade with CCP
- Participants have credit exposure to the CCP, not to each other



Risk Mitigation

Various types of Risk

- Credit Risk
- Liquidity Risk
- Market Risk
- Operational Risk

Regulations governing Risk Management

- Basel 3
- DoddFrank Act
- CCAR

How Risk is mitigated

- Margins
  - Initial Margin
  - Variation Margin
  - Maintenance Margins
  - Adhoc Margins
- Margins are covered using Collaterals
  - Cash
    - Specific currencies such as USD, GBP, JPY etc
  - Securities
  - Bank Guarantees
- Collateral Valuation
  - Limits, Excess haircuts
- Management of Excess collaterals
- Maintaining ratios as per Regulations

Agenda

- Capital Markets Overview
- Python for Finance
- Black & Scholes for European Call Options
- Monte Carlo simulation for European Call Options
- VaR using Monte Carlo Simulation
- Geometric Brownian Motion
- Basics of Volatility and Normal function
- Basics of Correlation, Covariance and VaR
- Credit Valuation Adjustment (CVA)

Python for Finance

Technology in Finance

1) Costs for Technology in Finance Industry
2) Technology as enabler for new business and innovation
3) Technology and talent as barrier to entry in the finance industry
4) Increasing speeds, frequencies and data volumes
5) The rise of real-time analytics

Translated to

Python

1) Python syntax close to mathematical syntax
2) Every mathematical & algorithmic statement can be translated into a single line of python code
3) Vectorization using NumPy allows for 100,000 calculations within a single line of code
4) Interactive data analytics
5) Comprehensive and specified set of libraries for scientific stack

Agenda

- Capital Markets Overview
- Python for Finance
- Black & Scholes for European Call Options
- Monte Carlo simulation for European Call Options
- VaR using Monte Carlo Simulation
- Geometric Brownian Motion
- Basics of Volatility and Normal function
- Basics of Correlation, Covariance and VaR
- Credit Valuation Adjustment (CVA)

Options

- Option is a contract that gives buyer the right, but not the obligation, to buy / sell an underlying asset at a specific price on or before certain date
- Suppose the investor feels that he has to purchase share of Accenture at 105$ after 12 months, which is currently trading at 100$ in the spot market
- However, the investor is confident that the price of Accenture share would move to 115$ in the next 12 months
- To lock his purchase of Accenture shares at 105$, the Investor wants to enter into a call option contract, which has expiration date of 1 year and the expiration / strike price as 105$
- As Options are traded on premium, the Investor has to pay the premium to own the options contract. The current option premium / value of the option is 9$ in the market
- However, how does Investor know whether the option premium is valued to its worth of the contract ?
- Black & Scholes model helps to value the options contact

Understanding Black & Scholes for European option pricing

- One of the most important concepts in modern financial theory
- Developed by Fisher Black, Robert Merton and Myron Scholes
- States – variation of price over time of financial instruments can be used to determine the price of European call option
- Model specifies:
  - Price of heavily traded assets follows Geometric Brownian motion with constant drift & volatility
  - When applied to stock option, the model incorporates
    - Constant price variation of the stock
    - Time value of money
    - Options strike price
    - Time to options expiry
- This model assumes:
  - No dividend is paid during options life
  - Options are European and can be exercised on expiration date
  - Efficient markets
  - No commissions
  - Risk free rate and volatility of underlying are known and constant
  - Follows a lognormal distribution (returns on the underlying are normally distributed)

Black & Scholes method for European Call Option

- Black & Scholes Options pricing formula:

$$C(S_t, K, t, T, r, \sigma) = S_t \cdot \mathbf{N}(d_1) - e^{-r(T-t)} \cdot K \cdot \mathbf{N}(d_2)$$

$$\mathbf{N}(d) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{d} e^{-\frac{1}{2}x^2} dx$$

$$d_1 = \frac{\log \frac{S_t}{K} + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}$$

$$d_2 = \frac{\log \frac{S_t}{K} + (r - \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}$$

  - Whereby S_t *is the price / level of underlying at time t*
  - *σ is the constant volatility*
  - *K is the strike price of the option*
  - *T is the maturity date of the option*
  - *r is constant risk rate*
- What would be the options value with St =100; K = 105; T = 1.0; r = 0.05; Sigma = 0.2; Ans: 8.021

OptionPricing_BS method.py

OptionPricing_BSmethod.py - C:\Users\Dell\Downloads\OptionPricing_BSmethod.py (2.7.12)

File Edit Format Run Options Window Help

```
from math import log, sqrt, exp
from scipy import stats

# Parameters
S0 = 100.  # initial value
K = 105.  # strike price
T = 1.0  # maturity
r = 0.05  # riskless short rate
sigma = 0.2  # volatility
d1 = (log(S0 / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * sqrt(T))
d2 = (log(S0 / K) + (r - 0.5 * sigma ** 2) * T) / (sigma * sqrt(T))
value = (S0 * stats.norm.cdf(d1, 0.0, 1.0)
        - K * exp(-r * T) * stats.norm.cdf(d2, 0.0, 1.0))
print "European Option Value %7.3f" % value
```

Ln: 1 Col: 0

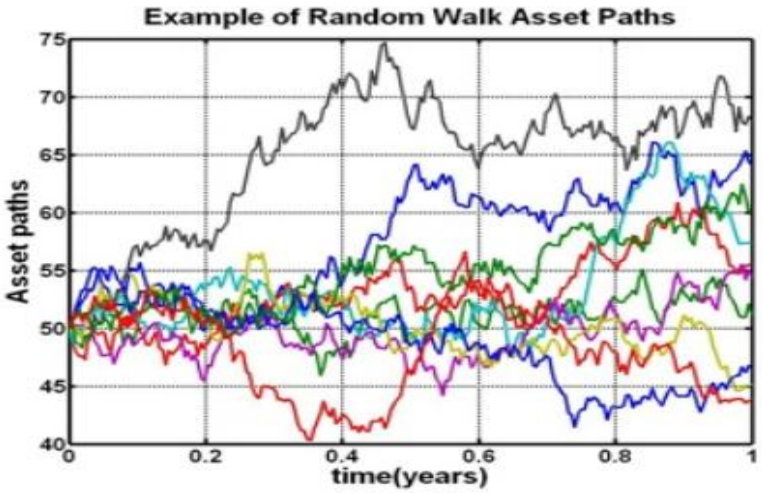
Agenda

- Capital Markets Overview
- Python for Finance
- Black & Scholes for European Call Options
- Monte Carlo simulation for European Call Options
- VaR using Monte Carlo Simulation
- Geometric Brownian Motion
- Basics of Volatility and Normal function
- Basics of Correlation, Covariance and VaR
- Credit Valuation Adjustment (CVA)

Monte Carlo Simulation

- Monte Carlo simulation helps to simulate the underlying process to arrive at the exact result
- In finance, Monte Carlo method is used to simulate various sources of uncertainty that affects the value of instrument / portfolio and then to calculate a representative value
- In valuing an option, the simulation generates several thousand possible price paths of the underlying share
- With the associated exercise value of the contract, the values are then averaged and discounted back to today, and this result i

Sample price path wrto time



- From the graph,
  - Current price of underlying value is 50 $
- The simulated price path of the underlyin

Asset at various time intervals is shown

Monte Carlo Simulation for European Call Option

- One of the most important algorithms in finance
- Used in Options pricing and Risk Management
- Use built-in python capabilities to implement Monte Carlo Simulation
- Include Time Steps & Paths
  - Divide the time interval between o & T, in equi-distant sub intervals Δ T
  - Determine the time T value of the index level ST(i) by applying the pseudo-random numbers time step by time step to the discretization scheme using the following equation, whereby z is a standard normal distributed random

$$S_t = S_{t-\Delta t} \exp\left((r - \frac{1}{2}\sigma^2)\Delta t + \sigma\sqrt{\Delta t}\, z_t\right)$$

  - Determine the inner value h_T of the European call option at T as $h_T(S_T(i)) = \max(S_T(i) - K, 0)$.
  - Iterate until $i = I$.
- Sum up the inner values, average, and discount them back with the riskless short rate accordin$C_0 \approx e^{-rT}\frac{1}{I}\sum_I h_T(S_T(i))$ion, which is Monte Carlo estimator for European Ca

- What would be the options value with So =100; K = 105; T = 1.0; r = 0.05; Sigma = 0.2; M=50; dt = T/M; I = 250000;

Option Price using MC

option_pricing_3.py

option_pricing_3.py - C:\Users\Dell\Downloads\Python_for_Finance_Code_&_Deck\option_pricin...

File Edit Format Run Options Window Help

```
from time import time
from math import exp, sqrt, log
from random import gauss, seed

seed(20000)
t0 = time()
# Parameters
S0 = 100.  # initial value
K = 105.  # strike price
T = 1.0  # maturity
r = 0.05  # riskless short rate
sigma = 0.2  # volatility
M = 50  # number of time steps
dt = T / M  # length of time interval
I = 250000  # number of paths

# Simulating I paths with M time steps
S = []
for i in range(I):
    path = []
    for t in range(M + 1):
        if t == 0:
            path.append(S0)
        else:
            z = gauss(0.0, 1.0)
            St = path[t - 1] * exp((r - 0.5 * sigma ** 2) * dt
                                   + sigma * sqrt(dt) * z)
            path.append(St)
    S.append(path)

# Calculating the Monte Carlo estimator
C0 = exp(-r * T) * sum([max(path[-1] - K, 0) for path in S]) / I

# Results output
tpy = time() - t0
print "European Option Value %7.3f" % C0
print "Duration in Seconds   %7.3f" % tpy
```

Agenda

- Capital Markets Overview
- Python for Finance
- Black & Scholes for European Call Options
- Monte Carlo simulation for European Call Options
- **VaR using Monte Carlo Simulation**
- Geometric Brownian Motion
- Basics of Volatility and Normal function
- Basics of Correlation, Covariance and VaR
- Credit Valuation Adjustment (CVA)

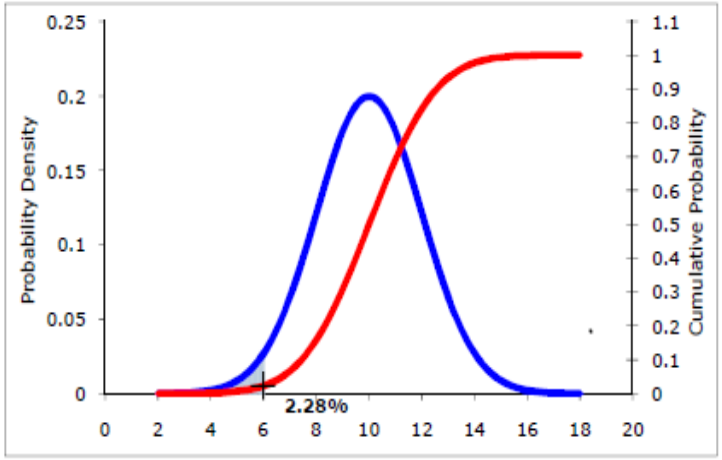
A Case of VaR: Supervisory Regulation

- The key basis of Regulation is to protect Banks from themselves, by setting aside money for a rainy day
- How much should they set aside is addressed by regulations like Basel 3
- If the Banks know how much they would loose with a certain confidence, over a certain interval of time, they would set aside that money to cover losses
- Things are not so simple, reality is complex, but there is still some hope
- Capital Adequacy is the term used to aside money to cover risks
- Regulators look for Banks to back-test their portfolio and penalize for exceptions
- With VaR banks can cover for Credit Risk, Market Risk and Interest Rate Risk

The Basics: VaR for a single security

- Review of what this means?
- Using historical values we can find the mean and standard deviation.
- If our mean is 10%, Probability of getting 10% or less return, comes to 50%.
- What is the probability that the return will fall to 6%
- We use the function NORMSDIST(Number of std deviations that fit) to determine the probability
- If we assume we have 10,00,000 in our portfolio with the same characteristics (Normal Distribution)
- We try to find out what is the maximum loss that we will have so that we can prepare for it.
- This has to be bounded by our confidence in the amount.
- We are confident that 95% of the time we will loose only 32,897

Observation	1	2	3	4	5
Mean Return	10%	10%	10%	10%	10%
Target Return	6%	7%	8%	10%	12%
Standard Deviation	2%	2%	2%	2%	2%
No. Standard Deviation	-2	-2	-1	0	1
Probability	2.28%	6.68%	15.87%	50.00%	84.13%



Mean Return	10%
Std Deviation	2%
Required VaR	95%
# Std Deviation	-1.644853627 =NORMSINV(1-B13)
Volatility	-3.29%
Portfolio Value	10,00,000
Maximum Loss	-32897

Value At Risk

- VaR has 3 components
  - Time period
  - Confidence level
  - Loss amount
- The maximum loss which could be incurred, over a target horizon (time) within a given confidence level
- VaR is the loss level that will not be exceeded with a specified probability
- Single number summarizes the portfolio's exposure to market risk as well as the probability of an adverse move
- Computation of VAR can be used to trim the Risk
- The following parameters needs to be specified for calculating VAR
  - Value of portfolio
  - Average return for a single time period
  - Standard deviation of returns for a single time period
  - Desired confidence level

Value At Risk calculations using Monte Carlo simulation

- Monte Carlo simulations corresponds to an algorithm that generates random numbers that are used to compute the formula
- Drawing random numbers over a large number of times, will have a good indication of what the output of the formula should be
- Computing VAR with Monte Carlo simulation is similar to Historical simulations
  - Instead of using the historical data for the price of the asset and assuming that this return can reoccur in next time interval, random price are generated
- Determine the time horizon t for analysis and divide it equally into small time periods (i.e) dt = t/m
- Draw a random number from the random number generator and update the price of the asset at the end of the first increment
- Repeat the above step until reaching the end of the analysis horizon T by walking along the M time intervals
- Repeat above 2 steps for I times to generate I different paths for the stock over T
- Rank the I terminal stock price from smallest to largest, read the simulated values that corresponds to the desired confidence level and deduce relevant VaR

VaR using Monte Carlo simulation

VaR_MCS.py

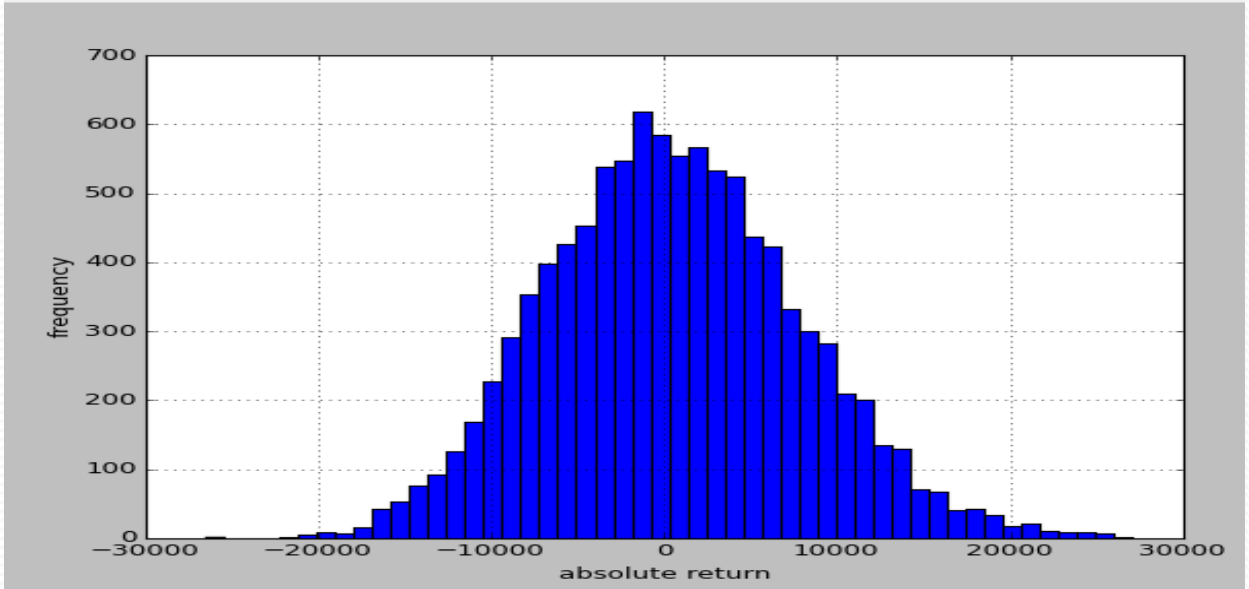
VaR_MCS.py - C:\Users\Dell\Downloads\Python_for_Finance_Code_&_Deck\VaR_MCS.py (2.7....

File Edit Format Run Options Window Help

```
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt
from scipy import stats
global log, sqrt, exp, stats, gen_sn
from time import time
from math import exp, sqrt, log
from random import gauss, seed
import scipy.stats as scs
"""Value at Risk"""
S0 = 100000
r = 0.05
sigma = 0.25
T = 30 / 365.
I = 10000
ST = S0 * np.exp((r - 0.5 * sigma ** 2) * T
             + sigma * np.sqrt(T) * npr.standard_normal(I))
R_gbm = np.sort(ST - S0)
print R_gbm
plt.hist(R_gbm, bins=50)
plt.xlabel('absolute return')
plt.ylabel('frequency')
plt.grid(True)
plt.show ()
percs = [0.01, 0.1, 1., 2.5, 5.0, 10.0]
var = scs.scoreatpercentile(R_gbm, percs)
print "%16s %16s" % ('Confidence Level', 'Value-at-Risk')
print 33 * "-"
for pair in zip(percs, var):
    print "%16.2f %16.3f" % (100 - pair[0], -pair[1])
```

Ln: 16 Col: 43

Output of the executed code

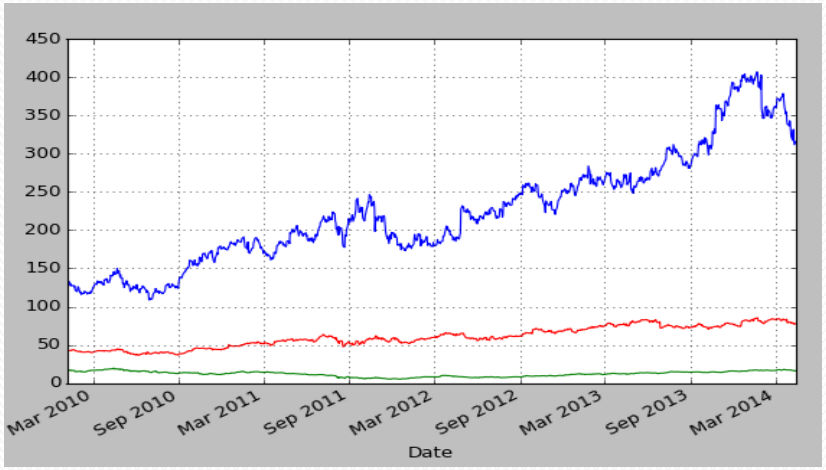



```
C:\Python27>Python VaR_MCS.py
[-24292.30222455 -22506.94738238 -22304.78169239 ...,  28192.0347064
  28934.93573841  30889.22772706]
Confidence Level    Value-at-Risk
--------------------------------
           99.99         22507.126
           99.90         19941.063
           99.00         14994.998
           97.50         12802.042
           95.00         10877.987
           90.00          8543.290

C:\Python27>_
```

Price comparison and 2D graphs

- The Institutional / Retail investor would be interested to know the relative price movements of individual stocks with respect to the other
- This would help to draw the correlation amongst the invested stocks and also to analyze performance of those stocks with respect to Index
- Lets assume the portfolio of the investor have the following industry stocks
  - Amazon
  - Bank of America
  - Accenture
- The investor would be interested to know the stock price movements from 1/1/2010 till 4/14/2014
- When a 2D graph is plotted, the following output is provided



Price Comparison & 2D graph

Pandas2Dplotting
.py

Pandas2Dplotting.py - C:\Users\Dell\Downloads\Python_for_Finance_Code_&_Deck\Pandas2...

File Edit Format Run Options Window Help

```
import numpy as np
import pandas as pd
import pandas.io.data as web
import matplotlib.pyplot as plt
"""sp500 = web.DataReader('^GSPC', data_source='yahoo',
                          start='1/1/2010', end='4/14/2014')
sp500.info()
sp500['Close'].plot(grid=True, figsize=(8, 5))
plt.show()"""
Amazon = web.DataReader('AMZN', data_source='yahoo',
                          start='1/1/2010', end='4/14/2014')
Amazon.info()
Amazon['Close'].plot(grid=True, figsize=(8, 5))
BankofAmerica = web.DataReader('BAC', data_source='yahoo',
                          start='1/1/2010', end='4/14/2014')
BankofAmerica.info()
BankofAmerica['Close'].plot(grid=True, figsize=(8, 5))
Accenture = web.DataReader('ACN', data_source='yahoo',
                          start='1/1/2010', end='4/14/2014')
Accenture.info()
Accenture['Close'].plot(grid=True, figsize=(8, 5))
plt.show()
```

Agenda

- Capital Markets Overview
- Python for Finance
- Black & Scholes for European Call Options
- Monte Carlo simulation for European Call Options
- VaR using Monte Carlo Simulation
- Geometric Brownian Motion
- Basics of Volatility and Normal function
- Basics of Correlation, Covariance and VaR
- Credit Valuation Adjustment (CVA)

Geometric Brownian Motion

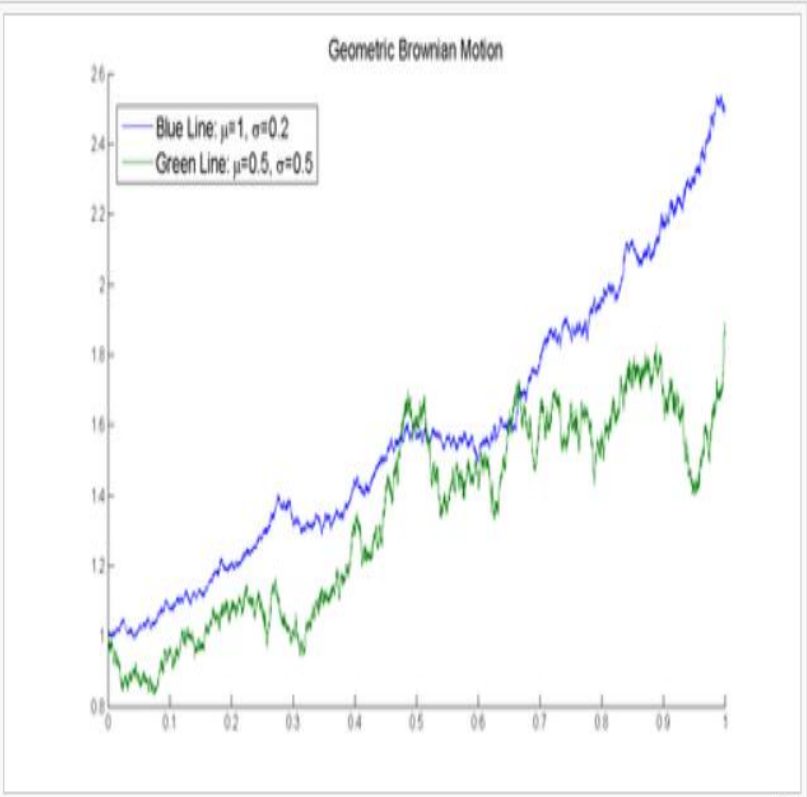
- GBM is continuous time stochastic process
- Random variable follows a Brownian motion with a drift
- A stochastic process St is said to follow a GBM if it satisfies the following stochastic differential equation (SDE):

$$dS_t = \mu S_t\,dt + \sigma S_t\,dW_t$$

where W_t is a Wiener process or Brownian motion, μ and ('the percentage drift') and σ ('the percentage volatility') are constants and St is the stock price at time t

- The above equation can be discretized by an Euler scheme, given below, whereby Δt being the fixed discretization and z_t and being a standard normally distributed

$$S_t = S_{t-\Delta t}\exp\left((r - \tfrac{1}{2}\sigma^2)\Delta t + \sigma\sqrt{\Delta t}\,z_t\right)$$



Two sample paths of Geometric Brownian motion, with different parameters. The blue line has larger drift, the green line has larger variance.

Generating GBM using MCS

GBM.py

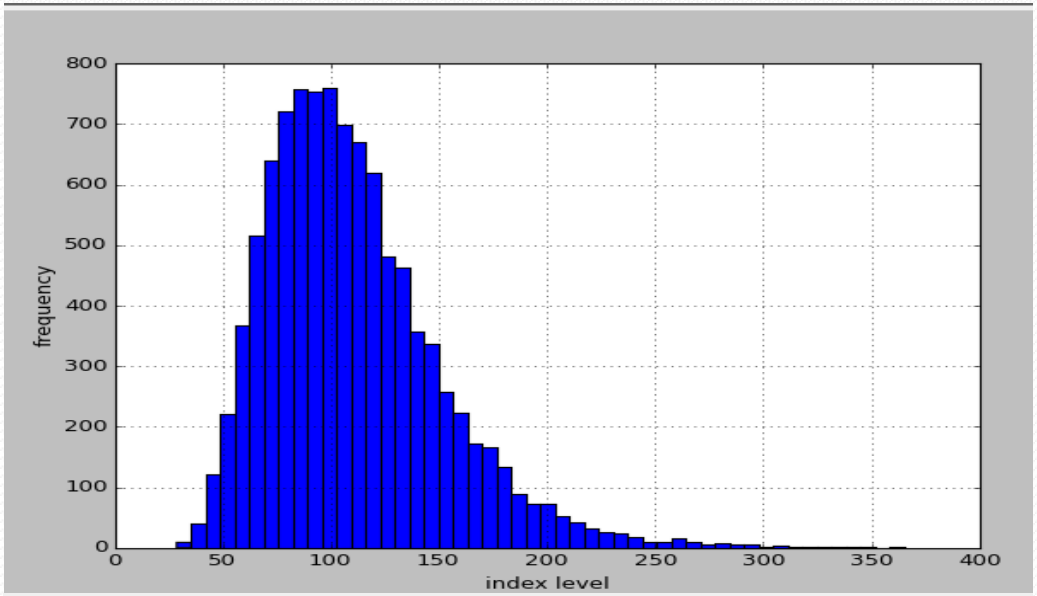
GBM.py - C:\Users\Dell\Downloads\Python_for_Finance_Code_&_Deck\GBM.py (2.7.12)

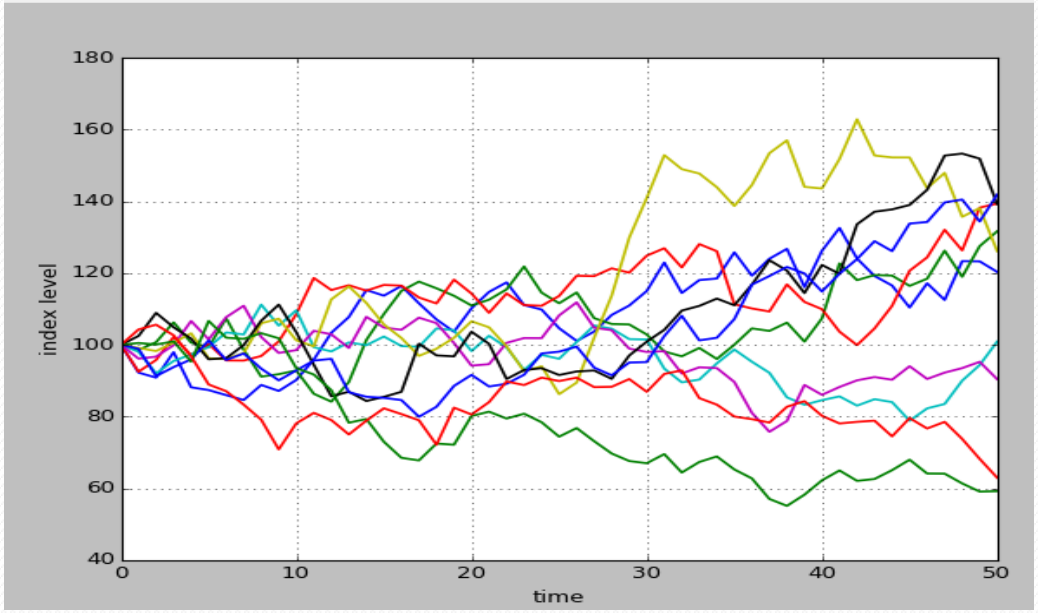
File Edit Format Run Options Window Help

```
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt
from scipy import stats
global log, sqrt, exp, stats, gen_sn
from time import time
from math import exp, sqrt, log
from random import gauss, seed
import scipy.stats as scs
"""Simulating index levels dynamically in Black-Scholes-Merton set up"""
S0 = 100  # initial value
I = 10000 # number of random draws
r = 0.05  # constant short rate
sigma = 0.25 # constant volatility
M = 50
T = 2.0  # in years
dt = T / M
S = np.zeros((M + 1, I))
S[0] = S0
for t in range(1, M + 1):
    S[t] = S[t - 1] * np.exp((r - 0.5 * sigma ** 2) * dt
           + sigma * np.sqrt(dt) * npr.standard_normal(I))
ST2 = S0 * npr.lognormal((r - 0.5 * sigma ** 2) * T,
                         sigma * np.sqrt(T), size=I)
def print_statistics(a1, a2):
    sta1 = scs.describe(a1)
    sta2 = scs.describe(a2)
    print "%14s %14s %14s" % \
        ('statistic', 'data set 1', 'data set 2')
    print 45 * "-"
    print "%14s %14.3f %14.3f" % ('size', sta1[0], sta2[0])
    print "%14s %14.3f %14.3f" % ('min', sta1[1][0], sta2[1][0])
    print "%14s %14.3f %14.3f" % ('max', sta1[1][1], sta2[1][1])
    print "%14s %14.3f %14.3f" % ('mean', sta1[2], sta2[2])
    print "%14s %14.3f %14.3f" % ('std', np.sqrt(sta1[3]),
                                  np.sqrt(sta2[3]))
    print "%14s %14.3f %14.3f" % ('skew', sta1[4], sta2[4])
    print "%14s %14.3f %14.3f" % ('kurtosis', sta1[5], sta2[5])
plt.hist(S[-1], bins=50)
plt.xlabel('index level')
```

Ln: 16 Col: 0

Output of the executed code





Agenda

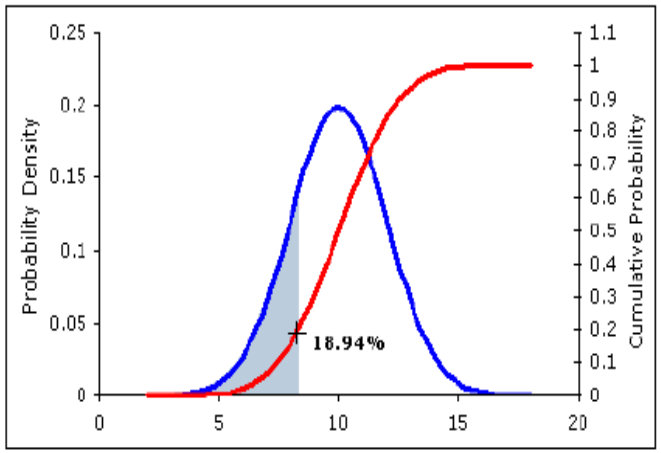
- Capital Markets Overview
- Python for Finance
- Black & Scholes for European Call Options
- Monte Carlo simulation for European Call Options
- VaR using Monte Carlo Simulation
- Geometric Brownian Motion
- Basics of Volatility and Normal function
- Basics of Correlation, Covariance and VaR
- Credit Valuation Adjustment (CVA)

The Basics: Volatility and Normal Function

- Mean is the same of the two portfolios.
- Standard deviation gives us a measure of how volatile the asset is.
- Normal Distribution: High probability that observation will be close to the mean.
- If the mean and standard deviation are known and a random variable (return) follows normal distribution we can determine the probability that the return will fall in a range.
- The key challenge in using Normal Distribution is that in reality tail reaches quite fast , when prices fall there is pressure on owners to cut losses and sell, which pushes the

Observation Period	Portfolio A	Portfolio B
1	11.50%	7.00%
2	11.20%	9.00%
3	11.00%	11.00%
4	10.30%	13.00%
5	11.00%	15.00%
Mean	11.00%	11.00%
Standard Deviation	0.00441588	0.031623

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$



Random Variable path of BSM using MCS

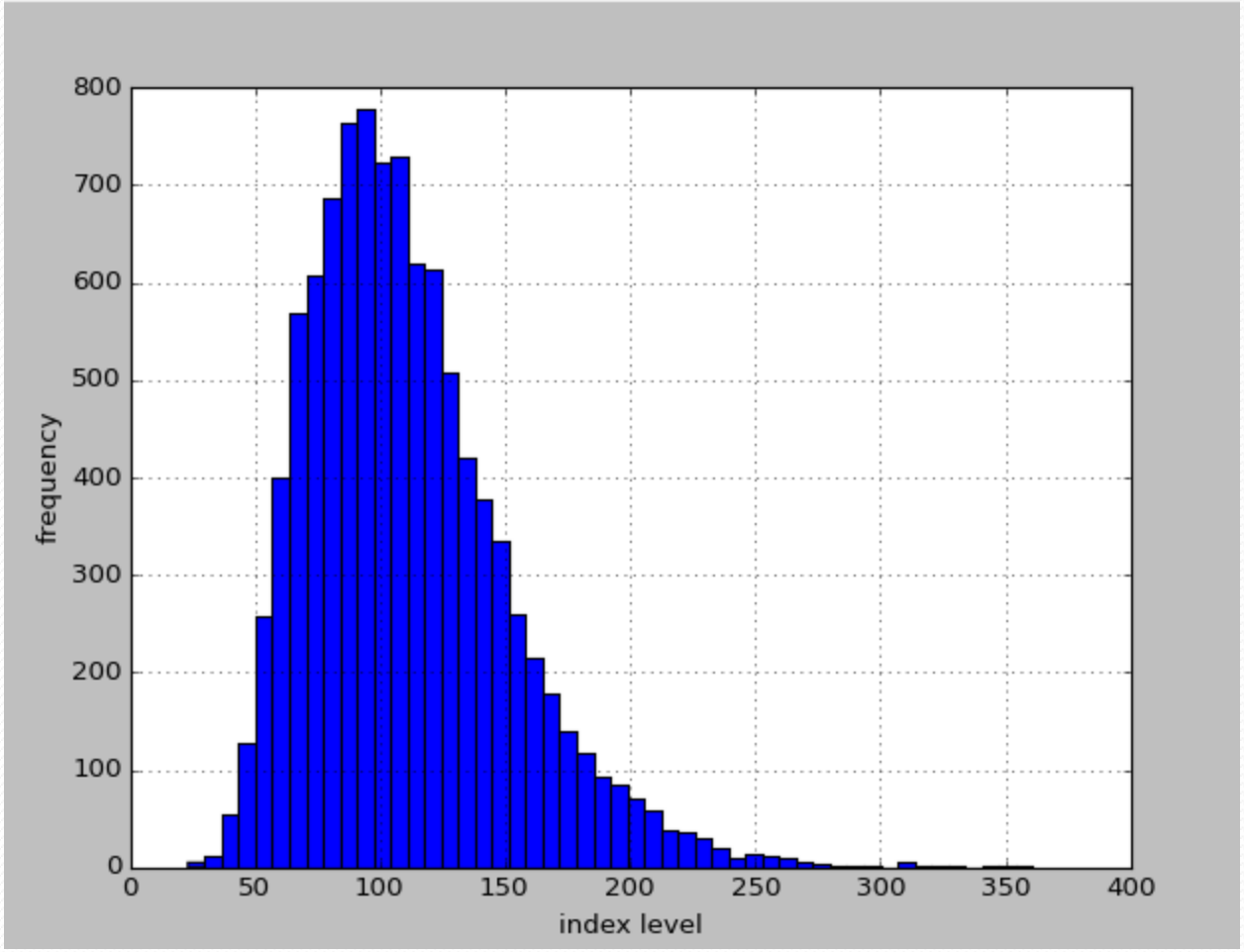
RandomVariable_SD.py

RandomVariable_SD.py - C:\Users\Dell\Downloads\Python_for_Finance_Code_&_Deck\Rando...

File Edit Format Run Options Window Help

```
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt
from scipy import stats
global log, sqrt, exp, stats, gen_sn
from time import time
from math import exp, sqrt, log
from random import gauss, seed
"""simulating futures index level in BS setup, via standard normal"""
S0 = 100  # initial value
r = 0.05  # constant short rate
sigma = 0.25  # constant volatility
T = 2.0  # in years
I = 10000  # number of random draws
ST1 = S0 * np.exp((r - 0.5 * sigma ** 2) * T
             + sigma * np.sqrt(T) * npr.standard_normal(I))
plt.hist(ST1, bins=50)
plt.xlabel('index level')
plt.ylabel('frequency')
plt.grid(True)
plt.show()
```

Output of the executed code



Agenda

- Capital Markets Overview
- Python for Finance
- Black & Scholes for European Call Options
- Monte Carlo simulation for European Call Options
- VaR using Monte Carlo Simulation
- Geometric Brownian Motion
- Basics of Volatility and Normal function
- Basics of Correlation, Covariance and VaR
- Credit Valuation Adjustment (CVA)

The Basics: Correlation, Covariance and VaR

- There are asset classes that can be statistically co-related
- Correlation coefficient of -0.9 means the two companies stocks are negatively correlated
- This would help in strategizing the Risk aversion
- If we combine and diversify, the risk of loosing can be reduced
- Again taking the 10,00,000 and dividing them equally among the two securities we can see that if we consider the securities independent, i.e. correlation of one, we can expect to lose 12,500 in the first and 47,200 in the second.
- When diversification is considered we get the VaR of 39,100 only.

Observation	X	Xi - Mx	Y	Yi - My	Covariance
1	8%	-4%	28%	16%	-0.569%
2	9%	-3%	24%	12%	-0.306%
3	10%	-2%	11%	-1%	0.011%
4	11%	-1%	11%	-1%	0.004%
5	12%	0%	8%	-4%	-0.019%
6	13%	2%	7%	-5%	-0.071%
7	14%	3%	2%	-10%	-0.244%
8	15%	4%	3%	-9%	-0.306%
Total					-1.500%
Covariance					-0.21%
Correlation Coeffienct					-0.927
Mean Return	12%		12%		
Standard Deviation	0.024		0.094		

$$Covariance = \frac{[(\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})]}{(n-1)}$$

$$Correlation\ Coefficient = Covariance/(SDx * SDy)$$

$$Portfolio\ Variance = w1 * \sigma1 + w2 * \sigma2 + 2\,w1 * w2 * \sigma1 * \sigma2 * \rho$$

	Example 1	Example 2	Example 3
X			
Volatility	2.45%	2.45%	2.45%
Weight	50%	50%	50%
Y			
Volatility	9.44%	9.44%	9.44%
Weight	50%	50%	50%
Cor. Coeff	-0.927	0	1
Portfolio Variance	0.13%	0.24%	0.35%
Diversified VaR	3.61%	4.88%	5.94%
Undiversified VaR	5.94%	5.94%	5.94%

Financial Product Characteristics

- We can classify financial products based on their relationship to the profit and loss that they generate.
- Linear: For example Forex forward where the P&L is linearly related to movements of the underlying.
- Curvature: There is a curved relationship (bonds with convexity).
- Non-linear: Here we run into a kinked or a broken line (e.g. options).
- Duration: How a bond price changes when the underlying interest rate changes.
- Convexity: Duration is an approximate value of the bonds price changes, to get more precision we also add Convexity.

Duration Calculation

Year	Interest	10%	Weighted Time
1	10	9.091	9.091
2	10	8.264	16.529
3	10	7.513	22.539
4	10	6.830	27.321
5	110	68.301	341.507
	Total		416.987
	Market Value		100.000
	Duration		4.170
	Modified Duration		3.791

Convexity

Year	Interest	10%	Convexity Col. 1 (n(n+1))	Convexity Col. 2
1	10	9.091	2	18.18
2	10	8.264	6	49.59
3	10	7.513	12	90.16
4	10	6.830	20	136.60
5	110	68.301	30	2049.04
	Market Value	100.000	Total	2343.57
			Convexity	23.44
			Modified Convexity	19.37

$$\Sigma C(f) * n(n+1) / MV(1+r)^2$$

Queries

Thank you