Use the adjusted daily closing prices for AAPL from August 10, 2016 to August 15, 2017.

## Part A

Similar to Problem #2, Python code was used to read in the data from finance.yahoo.com, calculate the returns, and then produce the standard deviation of the returns.

Python function for producing R:

def R(prices):

return (prices[1]-prices[0])/prices[0]

data\_df = pnda.read\_csv('final\_AAPL.csv')

data\_df['date\_formatted'] = pnda.to\_datetime(data\_df['Date'],infer\_datetime\_format=True)

data\_df.sort\_values(['date\_formatted'], ascending=[True], inplace=True)

data\_df['AAPL Return'] = pnda.rolling\_apply(arg=data\_df['AAPL Adj Close'], window=2, func=R,min\_periods=2)

historical\_stddev\_AAPL = np.std(np.array(data\_df['AAPL Return'][1:]), ddof=1, axis=0)

The daily standard deviation of the returns from AAPL between August 10, 2016 and August 15, 2017 is:

**0.0111931363057**

## Part B

Assuming that there are 250 trading days in a year:

Estimate = 0.0111931363057 \* **0.176979024433**

## yearly\_estimated\_stddev\_AAPL = historical\_stddev\_AAPL \* math.sqrt(250)

For the remaining questions, assume a portfolio containing one call at a strike price of and one put at a strike price of The time to maturity is *T = 12 months*. The annual risk-free rate *r = 0.0125* is used. The adjusted closing price for AAPL on August 15, 2017 is used as the initial stock price: *S = 161.600006*

## Part C

Price of the call: **18.9406014421**

Price of the put: **18.2400192418**

**Methodology:**

*N(x)* represents the cumulative distribution function for a standard normal. The Python *norm* function from the *scipy.stats* package is used.

By Black-Scholes, the price of a call option:

def priceCall(o):

d1, d2 = calculateCDFPoints(o)

price = o["S"] \* norm.cdf(d1) - (o["K"] \* math.exp(-o["r"] \* o["T-t"]) \* norm.cdf(d2))

return price

c = {}

c["S"] = S

c["K"] = 150

c["r"] = 0.0125

c["volatility"] = yearly\_estimated\_stddev\_AAPL

c["T-t"] = 12/12 - 0

print "Price of the call: {0}".format(priceCall(c))

The price of a European put option:

def pricePut(o):

d1, d2 = calculateCDFPoints(o)

price = o["K"] \* math.exp(-o["r"] \* o["T-t"]) \* norm.cdf(-d2) - (o["S"] \* norm.cdf(-d1))

return price

p = {}

p["S"] = S

p["K"] = 175

p["r"] = 0.0125

p["volatility"] = yearly\_estimated\_stddev\_AAPL

p["T-t"] = 12/12 – 0

print "Price of the put: {0}".format(pricePut(p))

Assume *r* is the annual risk-free rate = 0.0125.

Since the maturity is *T=12* months, starting at *t=0*:

*T – t = = 1* as the formulas are based upon yearly parameters.

-

def calculateCDFPoints(o):

numer = np.log(o["S"]/o["K"]) + (o["r"] + math.pow(o["volatility"], 2) / 2) \* o["T-t"]

denom = o["volatility"] \* math.sqrt(o["T-t"])

d1 = numer / denom

d2 = d1 - o["volatility"] \* math.sqrt(o["T-t"])

return d1, d2

d1, d2 = calculateCDFPoints(c)

print "d1 and d2 for the call : {0} {1}".format(d1, d2)

d1 and d2 for the call : 0.580010410798 0.403031386365

d1, d2 = calculateCDFPoints(p)

print "d1 and d2 for the put : {0} {1}".format(d1, d2)

d1 and d2 for the put : -0.291000604892 -0.467979629325

## Part D

The delta of the portfolio is **0.104571629844**

**Methodology:**

= 0.719046201403

def calculateDeltaForCall(o):

d1, d2 = calculateCDFPoints(o)

delta = norm.cdf(d1)

return delta

= -0.614474571559

def calculateDeltaForPut(o):

d1, d2 = calculateCDFPoints(o)

delta = -norm.cdf(-d1)

return delta

Since one call and one put are used in this problem:

0.719046201403 - 0.614474571559 = **0.104571629844**

## Part E

The gamma of the portfolio is **0.0251603514623**

**Methodology:**

The formula for Gamma is the same for both a call and a put.

def calculateGamma(o):

d1, d2 = calculateCDFPoints(o)

cdfPrimeValue\_d1 = (1/math.sqrt(2\*math.pi)) \* math.exp(-math.pow(d1,2)/2)

gamma = cdfPrimeValue\_d1 / (o["S"] \* o["volatility"] \* math.sqrt(o["T-t"]))

return gamma

Since one call and one put are used in this problem:

0.0117895150773

0.0133708363849

0.0117895150773 + 0.0133708363849 = **0.0251603514623**

## Part F

Determine what value of *n* (if any) will make a portfolio composed of 3 call options with strike price = and *n* put options with strike price = delta neutral.

Delta neutral requires:

where:

0.719046201403

= -0.614474571559

The is no integer value of *n* that will make the portfolio delta neutral:

Options are purchased in integer quantities (where each amount is a contract entitling the purchaser to buy/sell 100 shares upon exercise). *n = 3.510542* produces a delta neutral portfolio, but this implies buying *n = 4* put options. *n = 4* does not produce a delta neutral portfolio.

## Part G

Assume that on August 22, 2017, AAPL closes at $165 per share. Estimate the value of the call option on that date using

## Estimated value of the call option is **21.3327508062**

**Methodology:**

def estimateCallPrice(o, changeInS, changeInT):

current\_price = priceCall(o)

delta = calculateDeltaForCall(o)

gamma = calculateGamma(o)

theta = calculateThetaForCall(o)

estimate = current\_price+(delta\*changeInS)+(0.5\*gamma\*math.pow(changeInS,2))+(theta\*changeInT)

return estimate

# There are 5 trading days starting on August 16 thru August 22, 2017

# Assuming 250 trading days in a year, this suggests:

changeInTime =  5/250

# Assuming the closing price of $165 on August 22, this suggests:

changeInPrice = 165 – S

c = {}

c["S"] = S

c["K"] = 150

c["r"] = 0.0125

c["volatility"] = yearly\_estimated\_stddev\_AAPL

c["T-t"] = 12/12 - 0

print "Change in price : {0}".format(changeInPrice)

print "Delta for call option: {0}".format(calculateDeltaForCall(c))

print "Gamma for call option: {0}".format(calculateGamma(c))

print "Theta for call option: {0}".format(calculateThetaForCall(c))

estimated\_new\_price = estimateCallPrice(c, changeInPrice, changeInTime)

print "Call Price Estimation : {0}\n".format(estimated\_new\_price)

Change in price : 3.399994

Delta for call option: 0.719046201403

Gamma for call option: 0.0117895150773

Theta for call option: -6.03732815054

Call Price Estimation : 21.3327508062

#!/usr/bin/python

from \_\_future\_\_ import division

from warnings import filterwarnings

import math

import numpy as np

import pandas as pnda

from scipy.stats import norm

filterwarnings("ignore")

def logR(prices):

return math.log(prices[1]/prices[0])

def R(prices):

return (prices[1]-prices[0])/prices[0]

def calculateCDFPoints(o):

numer = np.log(o["S"]/o["K"]) + (o["r"] + math.pow(o["volatility"], 2) / 2) \* o["T-t"]

denom = o["volatility"] \* math.sqrt(o["T-t"])

d1 = numer / denom

d2 = d1 - o["volatility"] \* math.sqrt(o["T-t"])

return d1, d2

def calculateDeltaForCall(o):

d1, d2 = calculateCDFPoints(o)

delta = norm.cdf(d1)

return delta

def calculateDeltaForPut(o):

d1, d2 = calculateCDFPoints(o)

delta = -norm.cdf(-d1)

return delta

def calculateGamma(o):

d1, d2 = calculateCDFPoints(o)

cdfPrimeValue\_d1 = (1/math.sqrt(2\*math.pi)) \* math.exp(-math.pow(d1,2)/2)

gamma = cdfPrimeValue\_d1 / (o["S"] \* o["volatility"] \* math.sqrt(o["T-t"]))

return gamma

def calculateThetaForCall(o):

d1, d2 = calculateCDFPoints(o)

cdfPrimeValue\_d1 = (1/math.sqrt(2\*math.pi)) \* math.exp(-math.pow(d1,2)/2)

term1 = -(o["S"] \* cdfPrimeValue\_d1 \* o["volatility"]) / (2 \* math.sqrt(o["T-t"]))

term2 = o["r"] \* o["K"] \* math.exp(-o["r"]\*o["T-t"]) \* norm.cdf(d2)

theta = term1 - term2

return theta

def priceCall(o):

d1, d2 = calculateCDFPoints(o)

price = o["S"] \* norm.cdf(d1) - (o["K"] \* math.exp(-o["r"] \* o["T-t"]) \* norm.cdf(d2))

return price

def pricePut(o):

d1, d2 = calculateCDFPoints(o)

price = o["K"] \* math.exp(-o["r"] \* o["T-t"]) \* norm.cdf(-d2) - (o["S"] \* norm.cdf(-d1))

return price

def estimateCallPrice(o, changeInS, changeInT):

current\_price = priceCall(o)

delta = calculateDeltaForCall(o)

gamma = calculateGamma(o)

theta = calculateThetaForCall(o)

estimate = current\_price+(delta\*changeInS)+(0.5\*gamma\*math.pow(changeInS,2))+(theta\*changeInT)

return estimate

data\_df = pnda.read\_csv('final\_AAPL.csv')

data\_df['date\_formatted'] = pnda.to\_datetime(data\_df['Date'],infer\_datetime\_format=True)

data\_df.sort\_values(['date\_formatted'], ascending=[True], inplace=True)

data\_df['AAPL Return'] = pnda.rolling\_apply(arg=data\_df['AAPL Adj Close'], window=2, func=R, min\_periods=2)

historical\_stddev\_AAPL = np.std(np.array(data\_df['AAPL Return'][1:]), ddof=1, axis=0)

yearly\_estimated\_stddev\_AAPL = historical\_stddev\_AAPL \* math.sqrt(250)

data\_df['AAPL Logarithmic Return'] = pnda.rolling\_apply(arg=data\_df['AAPL Adj Close'], window=2, func=logR, min\_periods=2)

historical\_stddev\_AAPL\_log = np.std(np.array(data\_df['AAPL Logarithmic Return'][1:]), ddof=1, axis=0)

yearly\_estimated\_stddev\_AAPL\_log = historical\_stddev\_AAPL\_log \* math.sqrt(250)

print "Standard Deviation - AAPL Returns\n{0}\n".format(historical\_stddev\_AAPL)

print "Yearly Estimated Standard Deviation - AAPL Returns\n{0}\n".format(yearly\_estimated\_stddev\_AAPL)

print "Standard Deviation - AAPL Logarithmic Returns\n{0}\n".format(historical\_stddev\_AAPL\_log)

print "Yearly Estimated Standard Deviation - AAPL Logarithmic Returns\n{0}\n".format(yearly\_estimated\_stddev\_AAPL\_log)

n = len(data\_df['AAPL Adj Close'][0:])

S = data\_df['AAPL Adj Close'][0:][n-1]

print "The adjusted closing price of August 15, 2017 : {0}".format(S)

c = {}

c["S"] = S

c["K"] = 150

c["r"] = 0.0125

c["volatility"] = yearly\_estimated\_stddev\_AAPL

c["T-t"] = 12/12 - 0

print "Price of the call: {0}".format(priceCall(c))

p = {}

p["S"] = S

p["K"] = 175

p["r"] = 0.0125

p["volatility"] = yearly\_estimated\_stddev\_AAPL

p["T-t"] = 12/12 - 0

print "Price of the put: {0}".format(pricePut(p))

d1, d2 = calculateCDFPoints(c)

print "d1 and d2 for the call : {0} {1}".format(d1, d2)

d1, d2 = calculateCDFPoints(p)

print "d1 and d2 for the put : {0} {1}".format(d1, d2)

deltaPortfolio = calculateDeltaForCall(c) + calculateDeltaForPut(p)

print "Delta of the call : {0}".format(calculateDeltaForCall(c))

print "Delta of the put : {0}".format(calculateDeltaForPut(p))

print "Delta of the portfolio : {0}".format(deltaPortfolio)

gammaPortfolio = calculateGamma(c) + calculateGamma(p)

print "Gamma of the call : {0}".format(calculateGamma(c))

print "Gamma of the put : {0}".format(calculateGamma(p))

print "Gamma of the portfolio : {0}".format(gammaPortfolio)

# There are 5 trading days starting on August 16 thru August 22, 2017

# Assuming 250 trading days in a year, this suggests:

changeInTime = 5/250

# Assuming the closing price of $165 on August 22, this suggests:

changeInPrice = 165 - S

print "Change in price : {0}".format(changeInPrice)

print "Delta for call option: {0}".format(calculateDeltaForCall(c))

print "Gamma for call option: {0}".format(calculateGamma(c))

print "Theta for call option: {0}".format(calculateThetaForCall(c))

estimated\_new\_price = estimateCallPrice(c, changeInPrice, changeInTime)

print "Call Price Estimation : {0}\n".format(estimated\_new\_price)

print "\n"

print "##########################################################################"

print "Example 15.3\n"

print "##########################################################################"

e = {}

e["S"] = 43

e["K"] = 40

e["r"] = 0.10

e["volatility"] = 0.20

e["T-t"] = 0.5

print "Price of the call : {0}\n".format(priceCall(e))

print "Delta : {0}\n".format(calculateDeltaForCall(e))

print "Gamma : {0}\n".format(calculateGamma(e))

print "Theta : {0}\n".format(calculateThetaForCall(e))

print "Call Price Estimation : {0}\n".format(estimateCallPrice(e, 1, 1/26))