RISK_MNGT_HW8

November 3, 2016

In [6]:
```
#### Problem 2 plot spread #######
import numpy as np
import matplotlib.pyplot as plt

def spread(t):
    y = (-1/t) * np.log(0.4 + 0.6* np.e**(-0.03*t))
    return(y)

t1 = np.arange(0.0, 30.0, 0.1)

plt.figure()
plt.plot(t1, spread(t1))
plt.title('Spot spread for lambda = 0.03')
plt.show()
```

```
/Users/mac/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:6: RuntimeWarning: divide by zero e
/Users/mac/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:6: RuntimeWarning: invalid value er
```

In [3]:
```
##### HW8 Problem 3###### PART 1: window & MC 5 year for VaR #######
import numpy as np
import scipy.stats as ss
import pandas as pd
import math
import matplotlib.pyplot as plt
import datetime as dt


import pandas as pd
import operator
import itertools
from operator import add

path1 = '~/Documents/Semester3/M5320/homework/HW4/AMD-yahoo.csv'
path2 = '~/Documents/Semester3/M5320/homework/HW4/INTC-yahoo.csv'

AMD= pd.read_csv(path1, header = 0)
INTC= pd.read_csv(path2, header = 0)
data1 = AMD.values
AMD_INTC = pd.concat([AMD, INTC], axis = 1, join='inner', keys = 'Date' )
AMD_INTC_price  = AMD_INTC[[('D', 'Date'),('D', 'Adj Close'), ('a', 'Adj Close')]]

df = pd.DataFrame(AMD_INTC_price)
```

```
amd = df[[1]].values.tolist()
intc = df[[2]].values.tolist()

flat_amd= list(itertools.chain.from_iterable(amd))
flat_intc = list(itertools.chain.from_iterable(intc))

list1 = [x*640 for x in flat_amd]
list2 = [x*546 for x in flat_intc]

portfolio = [sum(x) for x in zip(list1, list2)]
AMD_close = portfolio


A_log_rtn = []
A_log_rtn_sq = []

for i in range(1 , len(data1)-1):
    log_return = math.log( AMD_close[i]/AMD_close[i+1] )
    A_log_rtn.append(log_return)
    A_log_rtn_sq.append(log_return**2)

def vol_and_mu(years):
    S0 = 10000
    T = 5/252
    p = 0.99

    A_vol_years = []
    A_mu_years = []
    A_VaR_years = []
    A_ES_years =[]

    A_MC_VaR_years = []

    for i in range(len(data1)-252*years):
        vol_years = np.std(A_log_rtn[i:i+252*years]) * np.sqrt(252)
        mu_years = np.mean(A_log_rtn[i:i+252*years])*252 + (vol_years**2)/2

        random = vol_years * ss.norm.ppf(np.random.rand())

        VaR_years = S0 - S0 * np.exp( vol_years * T**(0.5)* ss.norm.ppf(1-p) + (mu_years - pow(v
        ES_years = S0 * (1 - np.exp(mu_years *T)/(1-0.975) * ss.norm.cdf(ss.norm.ppf(1-0.975) -
        MC_VaR_years = S0 - S0 * np.exp( vol_years * T**(0.5)* ss.norm.ppf(1-p) + (mu_years+ran

        A_vol_years.append(vol_years)
        A_mu_years.append(mu_years)
        A_VaR_years.append(VaR_years)
        A_ES_years.append(ES_years)
        A_MC_VaR_years.append(MC_VaR_years)
    return(A_vol_years, A_mu_years, A_VaR_years, A_ES_years, A_MC_VaR_years)


A_VaR_5years = vol_and_mu(5)[2][1:252*20]
A_ES_5years = vol_and_mu(5)[3][1:252*20]
A_MC_VaR_5years = vol_and_mu(5)[4][1:252*20]
```

```
timeline = data1[1:252*20,0]
timeline = [dt.datetime.strptime(d,'%Y-%m-%d').date() for d in timeline]

fig, ax = plt.subplots()


#ax.plot(timeline, A_ES_5years,'c-',lw=0.5, label='ES 5 year')
ax.plot(timeline, A_MC_VaR_5years,'b-', label='MC Portfolio GBM VaR')
ax.plot(timeline, A_VaR_5years,'r-', label='Formula VaR')

legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('Portfolio VaR, 5yr windows')
plt.show()
```

In [5]:
```
##### HW8 Problem 3###### PART 2: window & MC 5 year for ES #######
import numpy as np
import scipy.stats as ss
import pandas as pd
import math
import matplotlib.pyplot as plt
import datetime as dt


import pandas as pd
import operator
import itertools
from operator import add

path1 = '~/Documents/Semester3/M5320/homework/HW4/AMD-yahoo.csv'
path2 = '~/Documents/Semester3/M5320/homework/HW4/INTC-yahoo.csv'

AMD= pd.read_csv(path1, header = 0)
INTC= pd.read_csv(path2, header = 0)
data1 = AMD.values
AMD_INTC = pd.concat([AMD, INTC], axis = 1, join='inner', keys = 'Date' )
AMD_INTC_price  = AMD_INTC[[('D', 'Date'),('D', 'Adj Close'), ('a', 'Adj Close')]]

df = pd.DataFrame(AMD_INTC_price)


amd = df[[1]].values.tolist()
intc = df[[2]].values.tolist()

flat_amd= list(itertools.chain.from_iterable(amd))
flat_intc = list(itertools.chain.from_iterable(intc))

list1 = [x*640 for x in flat_amd]
list2 = [x*546 for x in flat_intc]

portfolio = [sum(x) for x in zip(list1, list2)]
AMD_close = portfolio
```

```
A_log_rtn = []
A_log_rtn_sq = []

for i in range(1 , len(data1)-1):
    log_return = math.log( AMD_close[i]/AMD_close[i+1] )
    A_log_rtn.append(log_return)
    A_log_rtn_sq.append(log_return**2)

def vol_and_mu(years):
    S0 = 10000
    T = 5/252
    p = 0.99

    A_vol_years = []
    A_mu_years = []
    A_VaR_years = []
    A_ES_years =[]

    A_MC_VaR_years = []
    A_MC_ES_years = []

    for i in range(len(data1)-252*years):
        vol_years = np.std(A_log_rtn[i:i+252*years]) * np.sqrt(252)
        mu_years = np.mean(A_log_rtn[i:i+252*years])*252 + (vol_years**2)/2

        random = vol_years * ss.norm.ppf(np.random.rand())

        VaR_years = S0 - S0 * np.exp( vol_years * T**(0.5)* ss.norm.ppf(1-p) + (mu_years - pow(
        ES_years = S0 * (1 - np.exp(mu_years *T)/(1-0.975) * ss.norm.cdf(ss.norm.ppf(1-0.975) -
        MC_VaR_years = S0 - S0 * np.exp( vol_years * T**(0.5)* ss.norm.ppf(1-p) + (mu_years+ran
        MC_ES_years = S0 * (1 - np.exp((mu_years + random) *T)/(1-0.975) * ss.norm.cdf(ss.norm.

        A_vol_years.append(vol_years)
        A_mu_years.append(mu_years)
        A_VaR_years.append(VaR_years)
        A_ES_years.append(ES_years)
        A_MC_VaR_years.append(MC_VaR_years)
        A_MC_ES_years.append(MC_ES_years)
    return(A_vol_years, A_mu_years, A_VaR_years, A_ES_years, A_MC_VaR_years, A_MC_ES_years)


A_VaR_5years = vol_and_mu(5)[2][1:252*20]
A_ES_5years = vol_and_mu(5)[3][1:252*20]
A_MC_VaR_5years = vol_and_mu(5)[4][1:252*20]
A_MC_ES_5years = vol_and_mu(5)[5][1:252*20]

timeline = data1[1:252*20,0]
timeline = [dt.datetime.strptime(d,'%Y-%m-%d').date() for d in timeline]

fig, ax = plt.subplots()

ax.plot(timeline, A_MC_ES_5years,'m-',lw=0.5, label='MC Portfolio GBM ES')
ax.plot(timeline, A_ES_5years,'b-',lw=0.5, label='Formula ES')
#ax.plot(timeline, A_MC_VaR_5years,'b-', label='VaR 5 year')
```

```
        #ax.plot(timeline, A_VaR_5years,'r-', label='VaR 5 year')

        legend = ax.legend(loc='upper right', shadow=True)
        ax.set_title('Portfolio ES, 5yr windows')
        plt.show()
```

In [6]:
```
        ######## Problem 4 part1--Normal VaR stocks #########
        import numpy as np
        import scipy.stats as ss
        import pandas as pd
        import math
        import matplotlib.pyplot as plt
        import datetime as dt
        import itertools
        from operator import add
        import operator

        path1 = '~/Documents/Semester3/M5320/homework/HW4/AMD-yahoo.csv'
        path2 = '~/Documents/Semester3/M5320/homework/HW4/INTC-yahoo.csv'

        AMD = pd.read_csv(path1, header = 0)
        INTC = pd.read_csv(path2, header = 0)
        data1 = AMD.values
        data2 = INTC.values
        AMD_close = list(data1[:,6])
        INTC_close = list(data2[:,6])


        A_log_rtn = []
        A_log_rtn_sq = []

        for i in range(1 , len(data1)-1):
            log_return = math.log(AMD_close[i] / AMD_close[i+1])
            A_log_rtn.append(log_return)
            A_log_rtn_sq.append(log_return**2)


        I_log_rtn = []
        I_log_rtn_sq = []

        for i in range(1 , len(data1)-1):
            log_return = math.log(INTC_close[i] / INTC_close[i+1])
            I_log_rtn.append(log_return)
            I_log_rtn_sq.append(log_return**2)

        #########  Window Vol and Mu ###############

        S0 = 10000
        T = 5/252
        p = 0.99
        years = 5

        A_vol_years = []
        A_mu_years = []
```

```
I_vol_years = []
I_mu_years = []

corr = []

for i in range(len(data1)-252*years):
    vol_years1= np.std(A_log_rtn[i:i+252*years]) * np.sqrt(252)
    mu_years1 = np.mean(A_log_rtn[i:i+252*years])*252 + (vol_years1**2)/2

    vol_years2= np.std(I_log_rtn[i:i+252*years]) * np.sqrt(252)
    mu_years2 = np.mean(I_log_rtn[i:i+252*years])*252 + (vol_years2**2)/2

    x = np.asarray(A_log_rtn[i:i+252*years])
    y = np.asarray(I_log_rtn[i:i+252*years])
    X = np.hstack((x, y))

    cov = np.cov(X)
    corr_val = cov * 252 / (vol_years1 * vol_years2)
    A_vol_years.append(vol_years1)
    A_mu_years.append(mu_years1)


    I_vol_years.append(vol_years2)
    I_mu_years.append(mu_years2)

    corr.append(corr_val)


## exponential weighting paramater lambda
lambda2 = 0.9989003714

### List the lambda values

def list_lambdas(lambda_k):
    list_lambda = []
    for i in range(len(data1)):
        lambda_value = (lambda_k**i)
        list_lambda.append(lambda_value)
    return(list_lambda)


list_lambda2 = list_lambdas(lambda2)

wgt_log_rtn_A = []
wgt_log_rtn_sq_A = []


for i in range( len(data1)-1 ):
    log_return = math.log(AMD_close[i]/AMD_close[i+1] )
    wgt_log_return = log_return *  list_lambda2[i]
    wgt_log_rtn_A.append(wgt_log_return)

    log_return_sq = log_return ** 2
```

```
    wgt_log_rtn_sq_A.append( log_return_sq * list_lambda2[i] )

wgt_log_rtn_I = []
wgt_log_rtn_sq_I = []

for i in range( len(data1)-1 ):
    log_return = math.log( INTC_close[i]/INTC_close[i+1] )
    wgt_log_return = log_return *  list_lambda2[i]
    wgt_log_rtn_I.append(wgt_log_return)

    log_return_sq = log_return ** 2
    wgt_log_rtn_sq_I.append( log_return_sq * list_lambda2[i] )

#########  Weighterd Vol and Mu ###############
A_I_log_rtn = []
for i in range(len(data1)-252*years):
    x = A_log_rtn[i] * I_log_rtn[i] * list_lambda2[i]
    A_I_log_rtn.append(x)


wgt_A_vol_years = []
wgt_A_mu_years = []
wgt_I_vol_years = []
wgt_I_mu_years = []

wgt_corr = []

for j in range(len(data1)-252*years):
    A_wgt_mu_lambda0 =  sum(wgt_log_rtn_A[j:j+252*years])/sum(list_lambda2[j:j+252*years])
    A_wgt_vol_lambda = np.sqrt(252) * np.sqrt(sum(wgt_log_rtn_sq_A[j:j+252*years])/sum(list_lamb
    A_wgt_mu_lambda =252 * A_wgt_mu_lambda0 + (A_wgt_vol_lambda**2)/2

    I_wgt_mu_lambda0 =  sum(wgt_log_rtn_I[j:j+252*years])/sum(list_lambda2[j:j+252*years])
    I_wgt_vol_lambda = np.sqrt(252) * np.sqrt(sum(wgt_log_rtn_sq_I[j:j+252*years])/sum(list_lamb
    I_wgt_mu_lambda =252 * I_wgt_mu_lambda0 + (I_wgt_vol_lambda**2)/2

    wgt_A_vol_years.append(A_wgt_vol_lambda)
    wgt_A_mu_years.append(A_wgt_mu_lambda)
    wgt_I_vol_years.append(I_wgt_vol_lambda)
    wgt_I_mu_years.append(I_wgt_mu_lambda)

    x = [a*b for a,b in zip(A_log_rtn[i:i+252*years],I_log_rtn[i:i+252*years])]
    cov_bar = sum(A_I_log_rtn[j:j+252*years])/sum(list_lambda2[j:j+252*years]) - A_wgt_mu_lambda
    A_I_corr = cov_bar * 252/(A_wgt_vol_lambda * I_wgt_vol_lambda)
    wgt_corr.append(A_I_corr)

### Portfolio VaR and ES ############
## Construct portfolio
AMD_INTC = pd.concat([AMD, INTC], axis = 1, join='inner', keys = 'Date' )
AMD_INTC_price  = AMD_INTC[[('D', 'Date'),('D', 'Adj Close'), ('a', 'Adj Close')]]

df = pd.DataFrame(AMD_INTC_price)

amd = df[[1]].values.tolist()
```

```
intc = df[[2]].values.tolist()

flat_amd= list(itertools.chain.from_iterable(amd))
flat_intc = list(itertools.chain.from_iterable(intc))

list1 = [x*640*10000/19016 for x in flat_amd]
list2 = [x*546*10000/19016  for x in flat_intc]

portfolio = [sum(x) for x in zip(list1, list2)]

##################### Window VaR and ES ################
P_VaR_years = []
P_ES_years =[]

for i in range(len(data1)-252*years):
    mu = np.matrix([A_mu_years[i], I_mu_years[i]])
    sigma = np.matrix([A_vol_years[i], I_vol_years[i]])
    rho = np.matrix([[1, corr[i]],[corr[i],1]] )

    x0 = np.dot(sigma, rho)
    cov_mat = np.dot(x0, sigma.T )

    two_sto = np.matrix([list1[i], list2[i]])
    #horizon = np.matrix([5/252, 5/252])

    E_V_t = list1[i] * np.exp(A_mu_years[i] * 5/252) + list2[i] * np.exp(I_mu_years[i] * 5/252)

    x1 = np.matrix([np.exp(A_mu_years[i]*5/252), np.exp(I_mu_years[i]*5/252)])
    x2 = np.matrix(np.dot(two_sto.T, two_sto))
    x3 = np.exp(cov_mat.item(0) * 5/252) * x2

    E_V_t_sq = x1 * x3 * x1.T

    std_V_t = np.sqrt( E_V_t_sq.item(0) - E_V_t ** 2)
    VaR_val = list1[i] + list2[i] - (E_V_t + ss.norm.ppf(1-p) * std_V_t)
    ES_val = list1[i] + list2[i] - E_V_t + std_V_t * ss.norm.pdf(ss.norm.ppf(p))/(1-p)
    P_VaR_years.append(VaR_val)
    P_ES_years.append(ES_val)



##################### Weighted VaR and ES ################
P_wgt_VaR_years = []
P_wgt_ES_years =[]
for i in range(len(data1)-252*years):
    mu = np.matrix([wgt_A_mu_years[i], wgt_I_mu_years[i]])
    sigma = np.matrix([wgt_A_vol_years[i], wgt_I_vol_years[i]])
    rho = np.matrix([[1, wgt_corr[i]],[wgt_corr[i],1]] )

    x0 = np.dot(sigma, rho)
    cov_mat = np.dot(x0, sigma.T )

    two_sto = np.matrix([list1[i], list2[i]])
    E_V_t = list1[i] * np.exp(wgt_A_mu_years[i] * 5/252) + list2[i] * np.exp(wgt_I_mu_years[i]
```

```
        x1 = np.matrix([np.exp(wgt_A_mu_years[i]*5/252), np.exp(wgt_I_mu_years[i]*5/252)])
        x2 = np.matrix(np.dot(two_sto.T, two_sto))
        x3 = np.exp(cov_mat.item(0) * 5/252) * x2


        E_V_t_sq = x1 * x3 * x1.T

        std_V_t = np.sqrt( E_V_t_sq.item(0) - E_V_t ** 2)
        VaR_val = list1[i] + list2[i] - (E_V_t + ss.norm.ppf(1-p) * std_V_t)
        ES_val = list1[i] + list2[i] - E_V_t + std_V_t * ss.norm.pdf(ss.norm.ppf(p))/(1-p)


        P_wgt_VaR_years.append(VaR_val)
        P_wgt_ES_years.append(ES_val)



    timeline = data1[1:252*20,0]
    timeline = [dt.datetime.strptime(d,'%Y-%m-%d').date() for d in timeline]

    fig, ax = plt.subplots()
    ax.plot(timeline, P_wgt_VaR_years[1:252*20], 'm-', label='Weighted VaR Normal 2 year')
    #ax.plot(timeline, P_wgt_ES_years[1:252*20], 'r-', label='Weighted ES 2 year')

    ax.plot(timeline, P_VaR_years[1:252*20], 'b-', label='Window VaR Normal 2 year')
    #ax.plot(timeline, P_ES_years[1:252*20], 'g-', label='Window ES 2 year')



    legend = ax.legend(loc='upper right', shadow=True)
    ax.set_title('VaR and ES with Normal assumption, windowed data')
    plt.show()
```

In [ ]:

In [ ]: