RISK_MNGT_HW11

December 1, 2016

In [1]:
```
##### Problem 1 Backtesting, 5 year window Part I: AMD Long ##########

import operator
import itertools
from operator import add

import numpy as np
import scipy.stats as ss
import pandas as pd
import math
import matplotlib.pyplot as plt
import datetime as dt

path = '~/Documents/Semester3/M5320/homework/HW4/INTC-yahoo.csv'
sto_AMD= pd.read_csv(path, header = 1)

data1 = sto_AMD.values
AMD_close = list(data1[:,1])

'''
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
```

```
'''

##### formula VaR #########

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 =[]

##### GBM Formula VaR and ES Long #######
    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
        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) -

        A_vol_years.append(vol_years)
        A_mu_years.append(mu_years)
        A_VaR_years.append(VaR_years)
        A_ES_years.append(ES_years)

    return(A_vol_years, A_mu_years, A_VaR_years, A_ES_years)

A_VaR_5years = vol_and_mu(5)[2][1:252*20]


###################----------AMD Long--------###################3
S0 = 10000
nexcep = [ ]
for i in range(len(A_VaR_5years)):
    window_data = AMD_close[(len(A_VaR_5years)-i-252):(len(A_VaR_5years)-i-1)]
    excep = 0
    for j in range( len(window_data) - 4):
        share = S0/window_data[j]
        price0 = window_data[j]
        pricet = window_data[j+4]
        loss = (pricet - price0) * share
        if loss < - A_VaR_5years[i]:
            excep = excep + 1
        else:
```

```
                excep = excep

        nexcep.append(excep)



    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, nexcep, 'm-', label='INTC Long')
    legend = ax.legend(loc='upper right', shadow=True)
    ax.set_title('INTC Long(Horiz=5 days, Win=1 yr) Exceptions Per Year')
    plt.show()
```

In [2]:
```
##### Problem 1 Backtesting, 5 year window Part 2: AMD Long VS realized loss ##########

##################
#AMD Long (Horiz = 5 days, Window = 1 yrs) VaR vs Realized Losses
##################
loss=[ ]
for i in range(len(A_VaR_5years)):
    price0 = AMD_close[len(A_VaR_5years)-i+5]
    pricet = AMD_close[len(A_VaR_5years)-i+1]
    share = S0/price0
    loss.append((price0 - pricet)*share)

fig, ax = plt.subplots()

ax.plot(timeline, loss, 'm-', label='INTC Long')
ax.plot(timeline, A_VaR_5years, 'g' )
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title("INTC Long (Horiz = 5 days, Window = 1 yrs) VaR vs Realized Losses")

plt.show()
```

In [3]:
```
##### Problem 1 Backtesting, 5 year window Part 3: AMD Short  ##########

###################----------AMD Short--------####################3

def vol_and_mu_short(years):
    S0 = 10000
    T = 5/252
    p = 1 - 0.99

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

##### GBM Formula VaR and ES Short#######
    for i in range(len(data1)-252*years):
        vol_years = np.std(A_log_rtn[i:i+252*years]) * np.sqrt(252)
        #random value = vol_years * NORMSINV(RAND())
```

```
            mu_years = np.mean(A_log_rtn[i:i+252*years])*252 + (vol_years**2)/2
            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) -

            A_vol_years.append(vol_years)
            A_mu_years.append(mu_years)
            A_VaR_years.append(VaR_years)
            A_ES_years.append(ES_years)

        return(A_vol_years, A_mu_years, A_VaR_years, A_ES_years)

    A_VaR_5years_short = vol_and_mu_short(5)[2][1:252*20]



    S0=10000
    nexcep1 = [ ]
    for i in range(0, len(A_VaR_5years_short)):

        window_data = AMD_close[(len(A_VaR_5years_short)-i-252):(len(A_VaR_5years_short)-i)]
        excep = 0

        for j in range(1,len(window_data)-4):
            share = S0/window_data[j]
            price0 = window_data[j]
            pricet = window_data[j+4]
            loss = (pricet - price0)*share
            loss = -loss

            if loss < -A_VaR_5years_short[i]:
                excep = excep + 1
            else:
                excep = excep

        nexcep1.append(excep)


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

    len(nexcep1)
    len(timeline)
    fig, ax = plt.subplots()
    ax.plot(timeline, nexcep1, 'r-', label='INTC Short')
    legend = ax.legend(loc='upper right', shadow=True)
    ax.set_title('INTC Short(Horiz=5 days, Win=1 yr) Exceptions Per Year')
    plt.show()
```

In [4]:
```
##### Problem 1 Backtesting, 5 year window Part 4: AMD Short VS realized loss ##########


################
# AMD Short (Horiz = 5 days, Window = 1 yrs) VaR vs Realized Losses
################
```

```
loss=[]
for i in range(0,len(A_VaR_5years)):
    price0 = AMD_close[(len(A_VaR_5years)-i+5)]
    pricet = AMD_close[(len(A_VaR_5years)-i+1)]
    share = S0/price0
    loss.append(-(price0 - pricet)*share)


fig, ax = plt.subplots()

ax.plot(timeline, loss, 'r-', label='AMD Short')
ax.plot(timeline,A_VaR_5years, 'g' )
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title("AMD Short (Horiz = 5 days, Window = 1 yrs) VaR vs Realized Losses")

plt.show()
```

In [6]:
```
################### HW11 Problem 2  ######################

############################################################

## 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)


def weigthed_VaR_and_ES(list_lambda, years):
    wgt_A_vol_years = []
    wgt_A_mu_years = []

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

        log_return_sq = log_return ** 2
        wgt_log_rtn_sq.append( log_return_sq * list_lambda[i] )
    S0 = 10000
    T = 5/252
    p = .99
    wgt_A_VaR_years = []
```

```
    wgt_A_ES_years =[]

    for j in range(len(data1)-252*years):
        wgt_mu_lambda0 =  sum(wgt_log_rtn[j:j+252*years])/sum(list_lambda[j:j+252*years])
        wgt_vol_lambda = np.sqrt(252) * np.sqrt(sum(wgt_log_rtn_sq[j:j+252*years])/sum(list_lam
        wgt_mu_lambda =252 * wgt_mu_lambda0 + (wgt_vol_lambda**2)/2
        wgt_VaR_years = S0 - S0 * np.exp( wgt_vol_lambda * T**(0.5)* ss.norm.ppf(1-p) + (wgt_mu
        wgt_A_vol_years.append(wgt_vol_lambda)
        wgt_A_mu_years.append(wgt_mu_lambda)
        wgt_A_VaR_years.append(wgt_VaR_years)

        if j<6:
            wgt_ES_years = wgt_VaR_years
        else:
            wgt_sum_loss = []
            wgt_sum_loss_value = 0
            for k in range(4):
                wgt_sum_loss_value = wgt_sum_loss_value + wgt_A_VaR_years[j-k]
                wgt_sum_loss.append(wgt_sum_loss_value/(k+1))
                s = pd.Series(wgt_sum_loss)
                wgt_ES_years = s.quantile(.975)
        wgt_A_ES_years.append(wgt_ES_years)

    return(wgt_A_VaR_years, wgt_A_ES_years)

years_5 = 5
wgt_A_VaR_5years = weigthed_VaR_and_ES(list_lambda2, years_5)[0][1: 252*20]


###################----------AMD Long Expo Case----------##################3
S0 = 10000
nexcep = [ ]
for i in range(len(wgt_A_VaR_5years)):
    window_data = AMD_close[(len(A_VaR_5years)-i-252):(len(A_VaR_5years)-i-1)]
    excep = 0
    for j in range( len(window_data) - 4):
        share = S0/window_data[j]
        price0 = window_data[j]
        pricet = window_data[j+4]
        loss = (pricet - price0) * share
        if loss < - wgt_A_VaR_5years[i]:
            excep = excep + 1
        else:
            excep = excep

    nexcep.append(excep)



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, nexcep, 'r-', label='AMD Long Expo')
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('AMD Long Expo(Horiz=5 days, Win=1 yr) Exceptions Per Year')
plt.show()
```

In [8]:
```
########################
########################

loss=[ ]
for i in range(len(A_VaR_5years)):
    price0 = AMD_close[len(A_VaR_5years)-i+5]
    pricet = AMD_close[len(A_VaR_5years)-i+1]
    share = S0/price0
    loss.append((price0 - pricet)*share)

fig, ax = plt.subplots()

ax.plot(timeline, loss, 'r-', label='AMD Long Expo')
ax.plot(timeline, wgt_A_VaR_5years, 'g' )
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title("AMD Long Expo(Horiz = 5 days, Window = 1 yrs) VaR vs Realized Losses")

plt.show()
```

In [9]:
```
###################----------AMD Short Expo Short--------###################3
#############################################################################################
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)


def weigthed_VaR_and_ES_short(list_lambda, years):
    wgt_A_vol_years = []
    wgt_A_mu_years = []

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

        log_return_sq = log_return ** 2
        wgt_log_rtn_sq.append( log_return_sq * list_lambda[i] )
    S0 = 10000
    T = 5/252
```

```
    p =1 - 0.99
    wgt_A_VaR_years = []
    wgt_A_ES_years =[]

    for j in range(len(data1)-252*years):
        wgt_mu_lambda0 =  sum(wgt_log_rtn[j:j+252*years])/sum(list_lambda[j:j+252*years])
        wgt_vol_lambda = np.sqrt(252) * np.sqrt(sum(wgt_log_rtn_sq[j:j+252*years])/sum(list_lamb
        wgt_mu_lambda =252 * wgt_mu_lambda0 + (wgt_vol_lambda**2)/2
        wgt_VaR_years = -(S0 - S0 * np.exp( wgt_vol_lambda * T**(0.5)* ss.norm.ppf(1-p) + (wgt_m
        wgt_A_vol_years.append(wgt_vol_lambda)
        wgt_A_mu_years.append(wgt_mu_lambda)
        wgt_A_VaR_years.append(wgt_VaR_years)

        if j<6:
            wgt_ES_years = wgt_VaR_years
        else:
            wgt_sum_loss = []
            wgt_sum_loss_value = 0
            for k in range(4):
                wgt_sum_loss_value = wgt_sum_loss_value + wgt_A_VaR_years[j-k]
                wgt_sum_loss.append(wgt_sum_loss_value/(k+1))
                s = pd.Series(wgt_sum_loss)
                wgt_ES_years = s.quantile(.975)
        wgt_A_ES_years.append(wgt_ES_years)

    return(wgt_A_VaR_years, wgt_A_ES_years)

years_5 = 5
wgt_A_VaR_5years_short = weigthed_VaR_and_ES_short(list_lambda2, years_5)[0][1: 252*20]

############################## AMD Short Expo Case ######################

S0=10000
nexcep1 = [ ]
for i in range(0, len(A_VaR_5years_short)):

    window_data = AMD_close[(len(A_VaR_5years_short)-i-252):(len(A_VaR_5years_short)-i)]
    excep = 0

    for j in range(1,len(window_data)-4):
        share = S0/window_data[j]
        price0 = window_data[j]
        pricet = window_data[j+4]
        loss = (pricet - price0)*share
        loss = -loss

        if loss < -wgt_A_VaR_5years_short[i]:
            excep = excep + 1
        else:
            excep = excep

    nexcep1.append(excep)
```

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

len(nexcep1)
len(timeline)
fig, ax = plt.subplots()
ax.plot(timeline, nexcep1, 'r-', label='AMD Short Expo')
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('AMD Short Expo(Horiz=5 days, Win=1 yr) Exceptions Per Year')
plt.show()
```

In [10]:
```
########################
########################
loss=[ ]
for i in range(len(A_VaR_5years)):
    price0 = AMD_close[len(A_VaR_5years)-i+5]
    pricet = AMD_close[len(A_VaR_5years)-i+1]
    share = S0/price0
    loss.append((price0 - pricet)*share)

fig, ax = plt.subplots()

ax.plot(timeline, loss, 'r-', label='AMD Short')
ax.plot(timeline, wgt_A_VaR_5years_short, 'g' )
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title("AMD Short Expo(Horiz = 5 days, Window = 1 yrs) VaR vs Realized Losses")

plt.show()
```

In [ ]: