RISK_MNG_HW6_problem2,4

October 20, 2016

In [ ]:
```
##### HW6 Problem 2###### PART 1: unweighted mu and vols for VaR and 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 =[]


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


        if i<6:
            ES_years = VaR_years
        else:
            sum_loss = []
            sum_loss_value = 0
            for k in range(4):
                sum_loss_value = sum_loss_value + A_VaR_years[i-k]
                sum_loss.append(sum_loss_value/(k+1))
                s = pd.Series(sum_loss)
                ES_years = s.quantile(.975)
        A_ES_years.append(ES_years)

    return(A_vol_years, A_mu_years, A_VaR_years, A_ES_years)


A_VaR_2years = vol_and_mu(2)[2][1:252*20]
A_ES_2years = vol_and_mu(2)[3][1:252*20]


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

A_VaR_10years = vol_and_mu(10)[2][1:252*20]
A_ES_10years = vol_and_mu(10)[3][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_VaR_2years, 'y-', label='VaR 2 year')
ax.plot(timeline, A_VaR_5years,'g-', label='VaR 5 year')
ax.plot(timeline, A_VaR_10years,'m-', label='VaR 10 year')
ax.plot(timeline, A_ES_2years,'r-',lw=0.5, label='ES 2 year')
ax.plot(timeline, A_ES_5years,'c-',lw=0.5, label='ES 5 year')
ax.plot(timeline, A_ES_10years,'b-',lw=0.5, label='ES 10 year')


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

In [4]:
```
########## HW6 Problem2 ############ PART 2: exponentially weighted VaRs and 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
```

```
## exponential weighting paramater lambda
lambda1 = 0.9972531953
lambda2 = 0.9989003714
lambda3 = 0.9994500345


### 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_lambda1 = list_lambdas(lambda1)
list_lambda2 = list_lambdas(lambda2)
list_lambda3 = list_lambdas(lambda3)



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_lamb
        #wgt_A_vol_lambda.append(wgt_vol_lambda)
        wgt_mu_lambda =252 * wgt_mu_lambda0 + (wgt_vol_lambda**2)/2
        #wgt_A_mu_lambda.append(wgt_mu_lambda)
```

```
        #wgt_mu_lambda = sum(wgt_log_rtn[j:j+252*years])/sum(list_lambda[j:j+252*years])
        #wgt_vol_lambda = sum(wgt_log_rtn_sq[j:j+252*years])/sum(list_lambda[j:j+252*years])- w
        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_2 = 2
wgt_A_VaR_2years = weigthed_VaR_and_ES(list_lambda1, years_2)[0][1: 252*20]
wgt_A_ES_2years = weigthed_VaR_and_ES(list_lambda1, years_2)[1][1: 252*20]

years_5 = 5
wgt_A_VaR_5years = weigthed_VaR_and_ES(list_lambda2, years_5)[0][1: 252*20]
wgt_A_ES_5years = weigthed_VaR_and_ES(list_lambda2, years_5)[1][1: 252*20]

years_10 = 10
wgt_A_VaR_10years = weigthed_VaR_and_ES(list_lambda3, years_10)[0][1: 252*20]
wgt_A_ES_10years = weigthed_VaR_and_ES(list_lambda3, years_10)[1][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, wgt_A_VaR_2years, 'r-', label='expw VaR 2 year')
ax.plot(timeline, wgt_A_ES_2years, 'y-', label='expw ES 2 year')

ax.plot(timeline, wgt_A_VaR_5years, 'b-', label='expw VaR 5 year')
ax.plot(timeline, wgt_A_ES_5years, 'g-', label='expw ES 5 year')

ax.plot(timeline, wgt_A_VaR_10years, 'm-', label='expw VaR 10 year')
ax.plot(timeline, wgt_A_ES_10years, 'c-', label='expw ES 10 year')


legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('VaR and ES with lognormal assumption, exponential weighting')
```

```
plt.show()
```

In [6]:
```
########## HW6 Problem2 ############ PART 3: exponentially weighted VaRs and ES VS windows in 2
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


## 2,5, 10 windows VaR and ES

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 =[]


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


        if i<6:
            ES_years = VaR_years
        else:
            sum_loss = []
            sum_loss_value = 0
            for k in range(4):
                sum_loss_value = sum_loss_value + A_VaR_years[i-k]
                sum_loss.append(sum_loss_value/(k+1))
                s = pd.Series(sum_loss)
                ES_years = s.quantile(.975)
        A_ES_years.append(ES_years)

    return(A_vol_years, A_mu_years, A_VaR_years, A_ES_years)


A_VaR_2years = vol_and_mu(2)[2][1:252*20]
A_ES_2years = vol_and_mu(2)[3][1:252*20]


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

A_VaR_10years = vol_and_mu(10)[2][1:252*20]
A_ES_10years = vol_and_mu(10)[3][1:252*20]



## exponential weighting paramater lambda
lambda1 = 0.9972531953
lambda2 = 0.9989003714
lambda3 = 0.9994500345


### 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_lambda1 = list_lambdas(lambda1)
list_lambda2 = list_lambdas(lambda2)
list_lambda3 = list_lambdas(lambda3)



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_mu_lambda = sum(wgt_log_rtn[j:j+252*years])/sum(list_lambda[j:j+252*years])
        #wgt_vol_lambda = sum(wgt_log_rtn_sq[j:j+252*years])/sum(list_lambda[j:j+252*years])- w
        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 = []
```

```python
            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_2 = 2
wgt_A_VaR_2years = weigthed_VaR_and_ES(list_lambda1, years_2)[0][1: 252*20]
wgt_A_ES_2years = weigthed_VaR_and_ES(list_lambda1, years_2)[1][1: 252*20]

years_5 = 5
wgt_A_VaR_5years = weigthed_VaR_and_ES(list_lambda2, years_5)[0][1: 252*20]
wgt_A_ES_5years = weigthed_VaR_and_ES(list_lambda2, years_5)[1][1: 252*20]

years_10 = 10
wgt_A_VaR_10years = weigthed_VaR_and_ES(list_lambda3, years_10)[0][1: 252*20]
wgt_A_ES_10years = weigthed_VaR_and_ES(list_lambda3, years_10)[1][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_VaR_2years, 'y-', label='VaR 2 year')
ax.plot(timeline, A_ES_2years,'r-',lw=0.5, label='ES 2 year')

ax.plot(timeline, wgt_A_VaR_2years, 'r-', label='expw VaR 2 year')
ax.plot(timeline, wgt_A_ES_2years, 'y-', label='expw ES 2 year')
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('VaR comparison, GBM, 2 yr vs lambda')
plt.show()



fig, ax = plt.subplots()
ax.plot(timeline, A_VaR_5years,'g-', label='VaR 5 year')
ax.plot(timeline, A_ES_5years,'c-',lw=0.5, label='ES 5 year')

ax.plot(timeline, wgt_A_VaR_5years, 'b-', label='expw VaR 5 year')
ax.plot(timeline, wgt_A_ES_5years, 'g-', label='expw ES 5 year')
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('VaR comparison, GBM, 5 yr vs lambda')
plt.show()
```

```
fig, ax = plt.subplots()
ax.plot(timeline, A_VaR_10years,'m-', label='VaR 10 year')
ax.plot(timeline, A_ES_10years,'b-',lw=0.5, label='ES 10 year')

ax.plot(timeline, wgt_A_VaR_10years, 'm-', label='expw VaR 10 year')
ax.plot(timeline, wgt_A_ES_10years, 'c-', label='expw ES 10 year')
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('VaR comparison, GBM, 10 yr vs lambda')
plt.show()
```

In [6]:
```
##### HW6 Problem 2###### PART 4:  Normal Dist: unweighted mu and vols for VaR and 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 =[]


    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 = mu_years + ss.norm.ppf(1-p) * vol_years
        A_vol_years.append(vol_years)
        A_mu_years.append(mu_years)
        A_VaR_years.append(VaR_years)

        if i<6:
            ES_years = VaR_years
        else:
            sum_loss = []
            sum_loss_value = 0
            for k in range(4):
                sum_loss_value = sum_loss_value + A_VaR_years[i-k]
                sum_loss.append(sum_loss_value/(k+1))
                s = pd.Series(sum_loss)
                ES_years = s.quantile(.975)
        A_ES_years.append(ES_years)

    return(A_vol_years, A_mu_years, A_VaR_years, A_ES_years)


A_VaR_2years = vol_and_mu(2)[2][1:252*20]
A_ES_2years = vol_and_mu(2)[3][1:252*20]


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

A_VaR_10years = vol_and_mu(10)[2][1:252*20]
A_ES_10years = vol_and_mu(10)[3][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_VaR_2years, 'y-', label='VaR 2 year')
ax.plot(timeline, A_VaR_5years,'g-', label='VaR 5 year')
ax.plot(timeline, A_VaR_10years,'m-', label='VaR 10 year')
ax.plot(timeline, A_ES_2years,'r-',lw=0.5, label='ES 2 year')
ax.plot(timeline, A_ES_5years,'c-',lw=0.5, label='ES 5 year')
ax.plot(timeline, A_ES_10years,'b-',lw=0.5, label='ES 10 year')


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

In [ ]:

```
########## HW6 Problem2 ############ PART 5: Normally weighted VaRs and 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



## exponential weighting paramater lambda
```

```
lambda1 = 0.9972531953
lambda2 = 0.9989003714
lambda3 = 0.9994500345


### 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_lambda1 = list_lambdas(lambda1)
list_lambda2 = list_lambdas(lambda2)
list_lambda3 = list_lambdas(lambda3)



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_lamb
        wgt_mu_lambda =252 * wgt_mu_lambda0 + (wgt_vol_lambda**2)/2

        wgt_VaR_years = wgt_mu_lambda + ss.norm.ppf(1-p) * wgt_vol_lambda
        #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_2 = 2
wgt_A_VaR_2years = weigthed_VaR_and_ES(list_lambda1, years_2)[0][1: 252*20]
wgt_A_ES_2years = weigthed_VaR_and_ES(list_lambda1, years_2)[1][1: 252*20]

years_5 = 5
wgt_A_VaR_5years = weigthed_VaR_and_ES(list_lambda2, years_5)[0][1: 252*20]
wgt_A_ES_5years = weigthed_VaR_and_ES(list_lambda2, years_5)[1][1: 252*20]

years_10 = 10
wgt_A_VaR_10years = weigthed_VaR_and_ES(list_lambda3, years_10)[0][1: 252*20]
wgt_A_ES_10years = weigthed_VaR_and_ES(list_lambda3, years_10)[1][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, wgt_A_VaR_2years, 'r-', label='expw VaR 2 year')
ax.plot(timeline, wgt_A_ES_2years, 'y-', label='expw ES 2 year')

ax.plot(timeline, wgt_A_VaR_5years, 'b-', label='expw VaR 5 year')
ax.plot(timeline, wgt_A_ES_5years, 'g-', label='expw ES 5 year')

ax.plot(timeline, wgt_A_VaR_10years, 'm-', label='expw VaR 10 year')
ax.plot(timeline, wgt_A_ES_10years, 'c-', label='expw ES 10 year')


legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('VaR and ES with Normal Assumption, exponential weighting')

plt.show()
```

In [1]:
```
########## HW6 Problem2 ############ PART 6: Normally weighted VaRs and ES VS windows in 2,5,10
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


## 2,5, 10 windows VaR and ES

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 =[]


    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 = mu_years + ss.norm.ppf(1-p) * vol_years
        A_vol_years.append(vol_years)
        A_mu_years.append(mu_years)
        A_VaR_years.append(VaR_years)


        if i<6:
            ES_years = VaR_years
        else:
            sum_loss = []
            sum_loss_value = 0
            for k in range(4):
                sum_loss_value = sum_loss_value + A_VaR_years[i-k]
                sum_loss.append(sum_loss_value/(k+1))
                s = pd.Series(sum_loss)
                ES_years = s.quantile(.975)
        A_ES_years.append(ES_years)

    return(A_vol_years, A_mu_years, A_VaR_years, A_ES_years)


A_VaR_2years = vol_and_mu(2)[2][1:252*20]
A_ES_2years = vol_and_mu(2)[3][1:252*20]


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

A_VaR_10years = vol_and_mu(10)[2][1:252*20]
A_ES_10years = vol_and_mu(10)[3][1:252*20]



## exponential weighting paramater lambda
lambda1 = 0.9972531953
lambda2 = 0.9989003714
lambda3 = 0.9994500345


### 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_lambda1 = list_lambdas(lambda1)
list_lambda2 = list_lambdas(lambda2)
list_lambda3 = list_lambdas(lambda3)



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 = wgt_mu_lambda + ss.norm.ppf(1-p) * wgt_vol_lambda

        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_2 = 2
wgt_A_VaR_2years = weigthed_VaR_and_ES(list_lambda1, years_2)[0][1: 252*20]
wgt_A_ES_2years = weigthed_VaR_and_ES(list_lambda1, years_2)[1][1: 252*20]

years_5 = 5
wgt_A_VaR_5years = weigthed_VaR_and_ES(list_lambda2, years_5)[0][1: 252*20]
wgt_A_ES_5years = weigthed_VaR_and_ES(list_lambda2, years_5)[1][1: 252*20]

years_10 = 10
wgt_A_VaR_10years = weigthed_VaR_and_ES(list_lambda3, years_10)[0][1: 252*20]
wgt_A_ES_10years = weigthed_VaR_and_ES(list_lambda3, years_10)[1][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_VaR_2years, 'y-', label='VaR 2 year')
ax.plot(timeline, A_ES_2years,'r-',lw=0.5, label='ES 2 year')

ax.plot(timeline, wgt_A_VaR_2years, 'r-', label='expw VaR 2 year')
ax.plot(timeline, wgt_A_ES_2years, 'y-', label='expw ES 2 year')
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('VaR comparison, Normal Assumption, 2 yr vs lambda')
plt.show()



fig, ax = plt.subplots()
ax.plot(timeline, A_VaR_5years,'g-', label='VaR 5 year')
ax.plot(timeline, A_ES_5years,'c-',lw=0.5, label='ES 5 year')

ax.plot(timeline, wgt_A_VaR_5years, 'b-', label='expw VaR 5 year')
ax.plot(timeline, wgt_A_ES_5years, 'g-', label='expw ES 5 year')
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('VaR comparison, Normal Assumption, 5 yr vs lambda')
plt.show()




fig, ax = plt.subplots()
ax.plot(timeline, A_VaR_10years,'m-', label='VaR 10 year')
ax.plot(timeline, A_ES_10years,'b-',lw=0.5, label='ES 10 year')

ax.plot(timeline, wgt_A_VaR_10years, 'm-', label='expw VaR 10 year')
```

```
    ax.plot(timeline, wgt_A_ES_10years, 'c-', label='expw ES 10 year')
    legend = ax.legend(loc='upper right', shadow=True)
    ax.set_title('VaR comparison, Normal Assumption, 10 yr vs lambda')
    plt.show()
```

In [ ]:
```
######### HW6 Problem 4 Part1 ###########
import numpy as np
import matplotlib.pyplot as plt

def fun (x):
    if x <= 1:
        return 0.015
    elif 1 < x <= 2:
        return 0.02
    return 0.025

vfun = np.vectorize(fun)

x = np.linspace(0, 5, 1000)
y = vfun(x)

plt.plot(x, y, '-')
plt.title('Lambda Graph')
plt.xlabel('t')
plt.ylabel('lambda')
plt.show()
```

In [1]:
```
######### HW6 Problem 4 Part2 --- Survival probability graph ###########
import numpy as np
import matplotlib.pyplot as plt

def fun (x):
    if x <= 1:
        return np.exp(-0.015*x)
    elif 1 < x <= 2:
        return np.exp(-0.02*x + 0.005)
    return np.exp(-0.025*x + 0.015)

vfun = np.vectorize(fun)

x = np.linspace(0, 5, 1000)
y = vfun(x)

plt.plot(x, y, '-')
plt.title('Lambda Graph')
plt.xlabel('t')
plt.ylabel('probability')
plt.show()
```

In [2]:
```
######### HW6 Problem 4 Part2 --- Default time PDF graph ###########
import numpy as np
import matplotlib.pyplot as plt

def fun (x):
    if x <= 1:
```

```
            return 0.015 * np.exp(-0.015*x)
        elif 1 < x <= 2:
            return 0.02 * np.exp(-0.02*x + 0.005)
        return 0.025 * np.exp(-0.025*x + 0.015)

    vfun = np.vectorize(fun)

    x = np.linspace(0, 5, 1000)
    y = vfun(x)

    plt.plot(x, y, '-')
    plt.title('Default time PDF Graph')
    plt.xlabel('t')
    plt.show()
```

In [10]:
```
######### HW6 Problem 4 Part3 --- spread graph ###########
import numpy as np
import matplotlib.pyplot as plt
import math


def fun (x):
    R = 0.4
    if x <= 1:
        return  -(1/x)* math.log( 1- (1-R)*(1 - np.exp(-0.015*x)))
    elif 1 < x <= 2:
        return  -(1/x)* math.log( 1- (1-R)*(1 -  np.exp(-0.02*x + 0.005) ))
    return  -(1/x)* math.log( 1- (1-R)*(1 -  np.exp(-0.025*x + 0.015) ))

vfun = np.vectorize(fun)

x = np.linspace(0.01, 10, 10000)
y = vfun(x)

plt.plot(x, y, '-')
plt.title('Implied spot spread (in bp)')
plt.xlabel('t')
plt.show()
```

In [ ]: