RISK_MNG_HW6_problem1

October 20, 2016

In [185]:
```
############ HW6 Problem 1 PART 1----  2,5,10 year windows ######################
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)

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)


A_vol_2years = []
A_vol_5years = []
A_vol_10years = []
```

```
A_mu_2years = []
A_mu_5years = []
A_mu_10years = []
### L = 2 years
for i in range(len(data1)-252*2):
    vol_2years = np.std(A_log_rtn[i:i+252*2]) * np.sqrt(252)
    mu_2years = np.mean(A_log_rtn[i:i+252*2])*252 + (vol_2years**2)/2
    A_vol_2years.append(vol_2years)
    A_mu_2years.append(mu_2years)

### L = 5 years
for i in range( len(data1)-252*5):
    vol_5years = np.std(A_log_rtn[i:i+252*5]) * np.sqrt(252)
    mu_5years = np.mean(A_log_rtn[i:i+252*5])*252 + (vol_5years**2)/2
    A_vol_5years.append(vol_5years)
    A_mu_5years.append(mu_5years)

### L = 10 years
for i in range( len(data1)-252*10):
    vol_10years = np.std(A_log_rtn[i:i+252*10]) * np.sqrt(252)
    mu_10years = np.mean(A_log_rtn[i:i+252*10])*252 + (vol_10years**2)/2
    A_vol_10years.append(vol_10years)
    A_mu_10years.append(mu_10years)

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

A_vol_2years = A_vol_2years[1:252*20]
A_vol_5years = A_vol_5years[1:252*20]
A_vol_10years = A_vol_10years[1:252*20]

A_mu_2years = A_mu_2years[1:252*20]
A_mu_5years = A_mu_5years[1:252*20]
A_mu_10years = A_mu_10years[1:252*20]


fig, ax = plt.subplots()
ax.plot(timeline, A_vol_2years, 'r-', label='2 year')
ax.plot(timeline, A_vol_5years,'g-', label='5 year')
ax.plot(timeline, A_vol_10years, label='10 year')

legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('portfolio vols')


fig, ax = plt.subplots()
ax.plot(timeline, A_mu_2years, 'r-', label='2 year')
ax.plot(timeline, A_mu_5years,'g-', label='5 year')
ax.plot(timeline, A_mu_10years, label='10 year')

legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('portfolio mu')
plt.show()
```

In [7]:
```
############ HW6 Problem 1 PART 2 ---- various lambdas' windows ######################
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)

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
lambda4 = 0.94
lambda5 = 0.97

### 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)
list_lambda4 = list_lambdas(lambda4)
list_lambda5 = list_lambdas(lambda5)

def weigthed_vol_and_mu(list_lambda, years):
    wgt_A_vol_lambda = []
```

```
    wgt_A_mu_lambda = []

    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] )

    for j in range(len(data1)-252*years):
        wgt_mu_lambda =  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_A_vol_lambda.append(wgt_vol_lambda)
        wgt_mu_lambda =252 * wgt_mu_lambda + (wgt_vol_lambda**2)/2
        wgt_A_mu_lambda.append(wgt_mu_lambda)

    return(wgt_A_vol_lambda, wgt_A_mu_lambda )

years = 5
wgt_A_vol_lambda1 = weigthed_vol_and_mu(list_lambda1, years)[0][1: 252*20]
wgt_A_vol_lambda1 = [x*100 for x in wgt_A_vol_lambda1 ]
wgt_A_mu_lambda1 = weigthed_vol_and_mu(list_lambda1, years)[1][1: 252*20]
wgt_A_mu_lambda1 = [x*100 for x in wgt_A_mu_lambda1 ]

wgt_A_vol_lambda2 = weigthed_vol_and_mu(list_lambda2, years)[0][1: 252*20]
wgt_A_vol_lambda2 = [x*100 for x in wgt_A_vol_lambda2 ]
wgt_A_mu_lambda2 = weigthed_vol_and_mu(list_lambda2, years)[1][1: 252*20]
wgt_A_mu_lambda2 = [x*100 for x in wgt_A_mu_lambda2 ]

wgt_A_vol_lambda3 = weigthed_vol_and_mu(list_lambda3, years)[0][1: 252*20]
wgt_A_vol_lambda3 = [x*100 for x in wgt_A_vol_lambda3 ]
wgt_A_mu_lambda3 = weigthed_vol_and_mu(list_lambda3, years)[1][1: 252*20]
wgt_A_mu_lambda3 = [x*100 for x in wgt_A_mu_lambda3 ]

wgt_A_vol_lambda4 = weigthed_vol_and_mu(list_lambda4, years)[0][1: 252*20]
wgt_A_vol_lambda4 = [x*100 for x in wgt_A_vol_lambda4 ]
wgt_A_mu_lambda4 = weigthed_vol_and_mu(list_lambda4, years)[1][1: 252*20]
wgt_A_mu_lambda4 = [x*100 for x in wgt_A_mu_lambda4 ]

wgt_A_vol_lambda5 = weigthed_vol_and_mu(list_lambda5, years)[0][1: 252*20]
wgt_A_vol_lambda5 = [x*100 for x in wgt_A_vol_lambda5 ]
wgt_A_mu_lambda5 = weigthed_vol_and_mu(list_lambda5, years)[1][1: 252*20]
wgt_A_mu_lambda5 = [x*100 for x in wgt_A_mu_lambda5 ]



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_vol_lambda1, 'r-', label='lambda1 = 0.9972531953')
ax.plot(timeline, wgt_A_vol_lambda2, 'g-', label='lambda2 = 0.9989003714')
ax.plot(timeline, wgt_A_vol_lambda3, 'y-', label='lambda3 = 0.9994500345')
ax.plot(timeline, wgt_A_vol_lambda4, 'b-', label='lambda4 = 0.94')
ax.plot(timeline, wgt_A_vol_lambda5, 'm-', label='lambda4 = 0.97')


legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('Portfolio vols, exponential weighting')

fig, ax = plt.subplots()
ax.plot(timeline, wgt_A_mu_lambda1, 'r-', label='lambda1 = 0.9972531953')
ax.plot(timeline, wgt_A_mu_lambda2,'g-', label='lambda2 = 0.9989003714')
ax.plot(timeline, wgt_A_mu_lambda3,'y-', label='lambda3 = 0.9994500345')
ax.plot(timeline, wgt_A_mu_lambda3,'b-', label='lambda4 = 0.94')
ax.plot(timeline, wgt_A_mu_lambda3,'m-', label='lambda4 = 0.97')

legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('Portfolio mu, exponential weighting')
plt.show()
```

In [8]:
```
############ HW6 Problem 1 PART 3 ---- various lambdas + 2,5,10 year windows ##################
import pandas as pd
import operator
import itertools
from operator import add
import numpy as np
import scipy.stats as ss
import math
import matplotlib.pyplot as plt
import datetime as dt
import plotly.graph_objs as go
import plotly.plotly as py


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 year windows

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)


A_vol_2years = []
A_vol_5years = []
A_vol_10years = []

A_mu_2years = []
A_mu_5years = []
A_mu_10years = []
### L = 2 years
for i in range(len(data1)-252*2):
    vol_2years = np.std(A_log_rtn[i:i+252*2]) * np.sqrt(252)
    mu_2years = np.mean(A_log_rtn[i:i+252*2])*252 + (vol_2years**2)/2
    A_vol_2years.append(vol_2years)
    A_mu_2years.append(mu_2years)

### L = 5 years
for i in range( len(data1)-252*5):
    vol_5years = np.std(A_log_rtn[i:i+252*5]) * np.sqrt(252)
    mu_5years = np.mean(A_log_rtn[i:i+252*5])*252 + (vol_5years**2)/2
    A_vol_5years.append(vol_5years)
    A_mu_5years.append(mu_5years)

### L = 10 years
for i in range( len(data1)-252*10):
    vol_10years = np.std(A_log_rtn[i:i+252*10]) * np.sqrt(252)
    mu_10years = np.mean(A_log_rtn[i:i+252*10])*252 + (vol_10years**2)/2
    A_vol_10years.append(vol_10years)
    A_mu_10years.append(mu_10years)

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

A_vol_2years = A_vol_2years[1:252*20]
A_vol_5years = A_vol_5years[1:252*20]
A_vol_10years = A_vol_10years[1:252*20]

A_mu_2years = A_mu_2years[1:252*20]
A_mu_5years = A_mu_5years[1:252*20]
```

```
A_mu_10years = A_mu_10years[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_vol_and_mu(list_lambda, years):
    wgt_A_vol_lambda = []
    wgt_A_mu_lambda = []

    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] )

    for j in range(len(data1)-252*years):
        wgt_mu_lambda =  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_A_vol_lambda.append(wgt_vol_lambda)
        wgt_mu_lambda =252 * wgt_mu_lambda + (wgt_vol_lambda**2)/2
        wgt_A_mu_lambda.append(wgt_mu_lambda)

    return(wgt_A_vol_lambda, wgt_A_mu_lambda )

years1 = 2
wgt_A_vol_lambda1 = weigthed_vol_and_mu(list_lambda1, years1)[0][1: 252*20]
wgt_A_vol_lambda1 = [x+0.18 for x in wgt_A_vol_lambda1 ]
wgt_A_mu_lambda1 = weigthed_vol_and_mu(list_lambda1, years1)[1][1: 252*20]
#wgt_A_mu_lambda1 = [x*100 for x in wgt_A_mu_lambda1 ]
```

```
years2 = 5
wgt_A_vol_lambda2 = weigthed_vol_and_mu(list_lambda2, years2)[0][1: 252*20]
wgt_A_vol_lambda2 = [x+0.18 for x in wgt_A_vol_lambda2 ]
wgt_A_mu_lambda2 = weigthed_vol_and_mu(list_lambda2, years2)[1][1: 252*20]
#wgt_A_mu_lambda2 = [x*100 for x in wgt_A_mu_lambda2 ]

years3 = 10
wgt_A_vol_lambda3 = weigthed_vol_and_mu(list_lambda3, years3)[0][1: 252*20]
wgt_A_vol_lambda3 = [x+0.18 for x in wgt_A_vol_lambda3 ]
wgt_A_mu_lambda3 = weigthed_vol_and_mu(list_lambda3, years3)[1][1: 252*20]
#wgt_A_mu_lambda3 = [x*100 for x in wgt_A_mu_lambda3 ]




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_vol_lambda1, 'r-', label='lambda1 = 0.9972531953')
ax.plot(timeline, wgt_A_vol_lambda2, 'g-', label='lambda2 = 0.9989003714')
ax.plot(timeline, wgt_A_vol_lambda3, 'y-', label='lambda3 = 0.9994500345')
ax.plot(timeline, A_vol_2years, 'b-', label='2 year')
ax.plot(timeline, A_vol_5years,'m-', label='5 year')
ax.plot(timeline, A_vol_10years,'c-', label='10 year')




legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('Portfolio vols, exponential weighting vs windows')

fig, ax = plt.subplots()
ax.plot(timeline, wgt_A_mu_lambda1, 'r-', label='lambda1 = 0.9972531953')
ax.plot(timeline, wgt_A_mu_lambda2,'g-', label='lambda2 = 0.9989003714')
ax.plot(timeline, wgt_A_mu_lambda3,'y-', label='lambda3 = 0.9994500345')
ax.plot(timeline, A_mu_2years, 'b-', label='2 year')
ax.plot(timeline, A_mu_5years,'m-', label='5 year')
ax.plot(timeline, A_mu_10years,'c-', label='10 year')



legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('Portfolio mu, exponential weighting vs windows')
plt.show()
```

In [ ]: