Fixed Income HW5

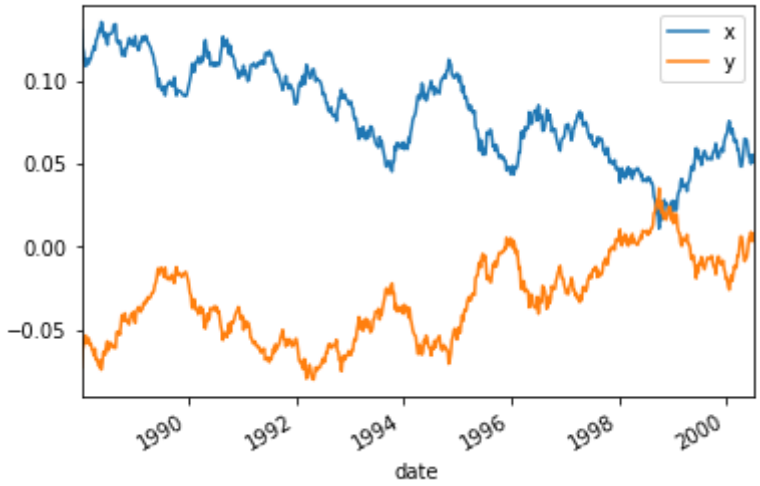
Group Members: Huanyu Liu, Justin Tan, Tongsu Peng, Sajel Bharati

Q1-4

In [6]:

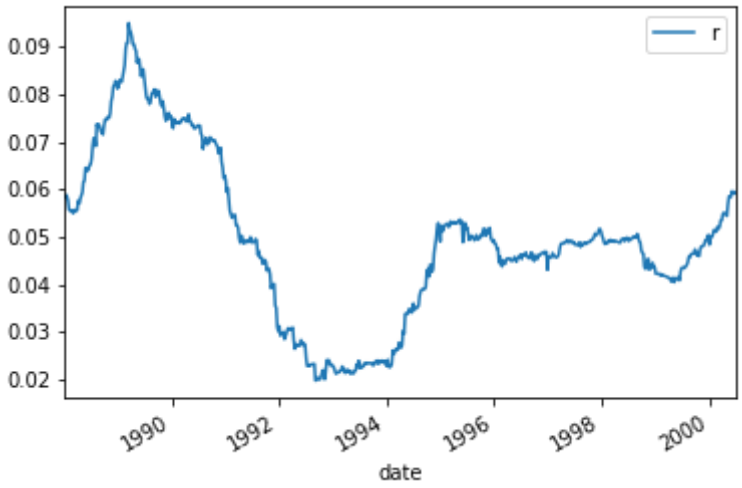
```
#%matplotlib inline
import pandas as pd
import numpy as np
import scipy.optimize as opt
import matplotlib.pyplot as plt

# 1-4

data = pd.read_excel('/Users/huanyu/Desktop/FixedIncome/hw5/Homework_5.x
lsx')
data[['cmt0.25','cmt2','cmt3','cmt5','cmt7','cmt10']] = data[['cmt0.25',
'cmt2','cmt3','cmt5','cmt7','cmt10']] / 100
data['date'] = np.vectorize(pd.datetime)(data['year'],data['month'],data
['day'])
def A_T(alpha, beta, sigma, T):
    temp1 = (sigma ** 2 / (2 * beta * beta) - alpha / beta) * T
    temp2 = (alpha / (beta * beta) - sigma * sigma / beta ** 3) * (1 - n
p.exp(-beta * T))
    temp3 = sigma * sigma / (4 * beta ** 3) * (1 - np.exp(-2 * beta * T
))
    return np.exp(temp1 + temp2 + temp3)

def B_T(beta, T):
    return 1 / beta * (1 - np.exp(-beta * T))

def solve(x, y, alphax, betax, sigmax, betay, sigmay):
    ax0_25 = A_T(alphax, betax, sigmax, 0.25)
    bx0_25 = B_T(betax, 0.25)
    ay0_25 = A_T(0, betay, sigmay, 0.25)
    by0_25 = B_T(betay, 0.25)
    ax10 = A_T(alphax, betax, sigmax, 10)
    bx10 = B_T(betax, 10)
    ay10 = A_T(0, betay, sigmay, 10)
    by10 = B_T(betay, 10)
    a1 = bx0_25 / 0.25
    b1 = by0_25 / 0.25
    c1 = np.log(ax0_25) / 0.25 + np.log(ay0_25) / 0.25
    a2 = bx10 / 10
    b2 = by10 / 10
    c2 = np.log(ax10) / 10 + np.log(ay10) / 10
    a = np.array([[a1,b1],[a2,b2]])
    b = np.array([c1 + x, c2 + y])
    return np.linalg.solve(a, b)

def D_T(x, y, alphax, betax, sigmax, betay, sigmay, T):
    axT = A_T(alphax, betax, sigmax, T)
    ayT = A_T(0, betay, sigmay, T)
    bxT = B_T(betax, T)
    byT = B_T(betay, T)
    return axT * ayT * np.exp(-bxT * x - byT * y)
def par_rate(x, y, alphax, betax, sigmax, betay, sigmay, T):
    DT = D_T(x, y, alphax, betax, sigmax, betay, sigmay, T)
    denominator = 0
    for i in range(1,2*T + 1):
        denominator += D_T(x, y, alphax, betax, sigmax, betay, sigmay, i
/2)
```

```
    return 2 * (1 - DT) / denominator

def RMSE(parameters, data):
    alphax = parameters[0]
    betax = parameters[1]
    sigmax = parameters[2]
    betay = parameters[3]
    sigmay = parameters[4]
    if sigmax < 0 or sigmay < 0:
        return 10000
    try:
        x, y = solve(data['cmt0.25'], data['cmt10'], alphax, betax, sigm
ax, betay, sigmay)
    except np.linalg.LinAlgError as err:
        if 'Singular matrix' in str(err):
            return 100000
        else:
            raise
    par2 = par_rate(x, y, alphax, betax, sigmax, betay, sigmay, 2)
    par3 = par_rate(x, y, alphax, betax, sigmax, betay, sigmay, 3)
    par5 = par_rate(x, y, alphax, betax, sigmax, betay, sigmay, 5)
    par7 = par_rate(x, y, alphax, betax, sigmax, betay, sigmay, 7)
    output = np.sum(np.square(data['cmt2'] - par2)) + np.sum(np.square(d
ata['cmt3'] - par3)) + np.sum(np.square(data['cmt5'] - par5)) + np.sum(n
p.square(data['cmt7'] - par7))
    return np.sqrt(output) / 4
init_value = [0.1, 0.2, 0.1, 0.3, 0.4]
output2 = opt.minimize(RMSE,x0=[0.1, 0.3, 0.05, 0.15, 0.4],args=(data,),
method = 'L-BFGS-B',bounds=((0.00001,0.999),(0.00001,0.999),(0.00001,1),
(-0.999,None),(0.00001,1)))
alphax, betax, sigmax, betay, sigmay = output2.x
x,y = solve(data['cmt0.25'],data['cmt10'], alphax, betax, sigmax, betay,
sigmay)
data['x'] = x
data['y'] = y
data[['date','x','y']].plot(x='date')
plt.show()
print(".....................")
print("..........X..........")
print("alpha x = ", alphax)
print("beta x = ", betax)
print("sigma x = ", sigmax)
print("..........Y..........")
print("beta y = ", betay)
print("sigma y = ", sigmay)
print(".....................")
```




```
....................
..........X..........
alpha x =  0.062363366455294236
beta x =  0.13562973067080977
sigma x =  0.06854248714727425
..........Y..........
beta y =  0.5681399964666118
sigma y =  0.356965642567698
....................
```

In [2]:
```
data['r'] = data['x'] + data['y']
data[['date','r']].plot(x='date')
plt.show()
```



Q5

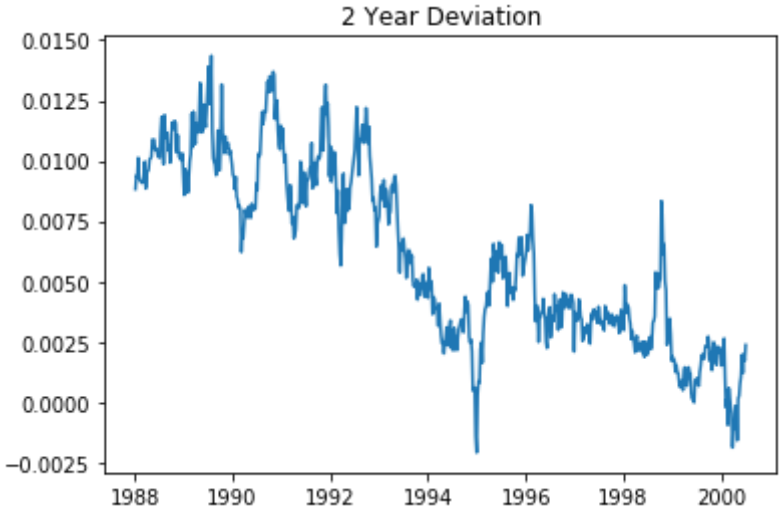
In [3]:
```
# 5

x_mean = data['x'].mean()
x_var = data['x'].var()
y_mean = data['y'].mean()
y_var = data['y'].var()
Ex = alphax / betax
Vx = sigmax * sigmax / (2 * betax)
Vy = sigmay * sigmay / (2 * betay)
# x_diff_expectation = abs(x_mean - Ex)
# x_diff_volatility = abs(x_std - Vx)
# y_diff_expectation = abs(x_mean - Ex)
# y_diff_volatility = abs(x_std - Vx)
result5 = np.array([[x_mean,x_var,y_mean,y_var],[Ex,Vx,0,Vy]])
result5_df = pd.DataFrame(result5,index=['Estimated','Risk neutral'],col
umns=['X mean','X variance','Y mean','Y variance'])
print(result5_df)
```

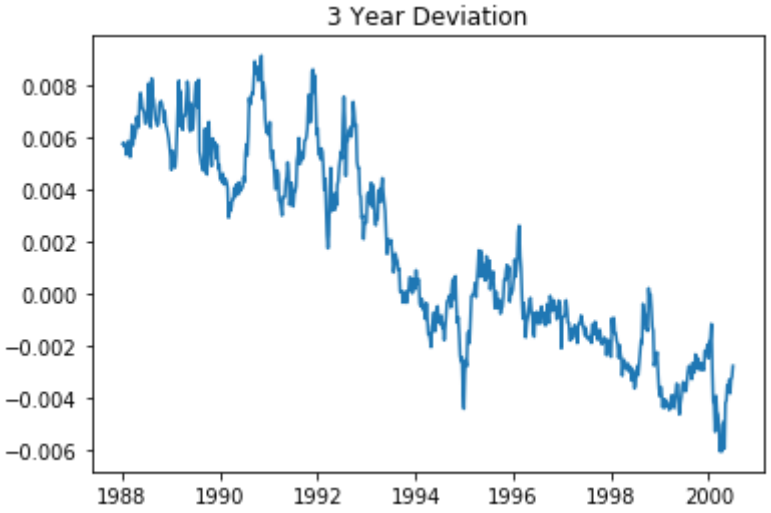
```
              X mean  X variance    Y mean  Y variance
Estimated   0.081232    0.000793 -0.031813    0.000654
Risk neutral  0.459806    0.017319  0.000000    0.112142
```

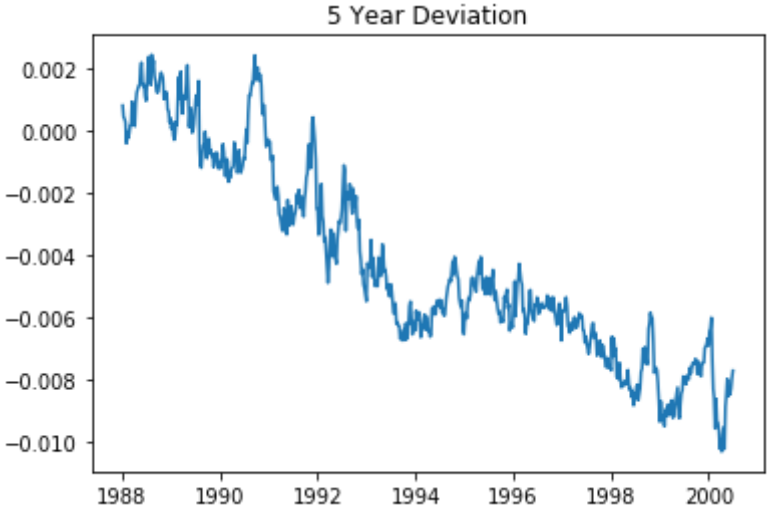
Q6

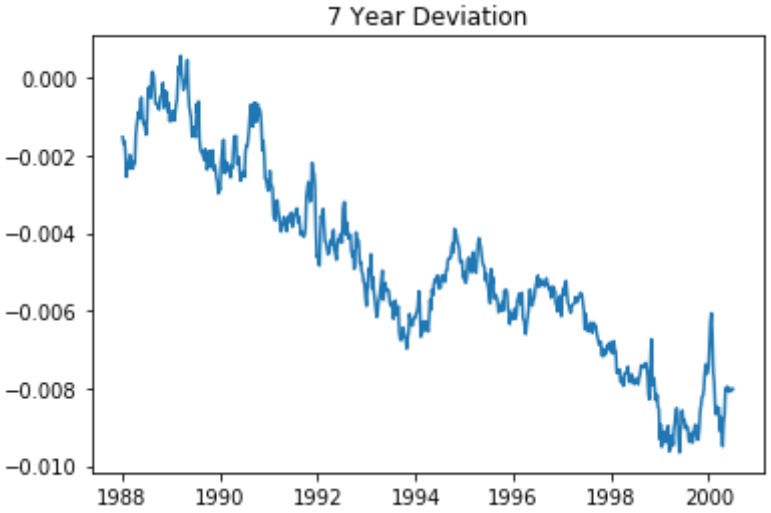
In [4]:

```
# 6

par2 = par_rate(x,y,alphax,betax,sigmax,betay,sigmay,2)
par3 = par_rate(x,y,alphax,betax,sigmax,betay,sigmay,3)
par5 = par_rate(x,y,alphax,betax,sigmax,betay,sigmay,5)
par7 = par_rate(x,y,alphax,betax,sigmax,betay,sigmay,7)
deviation2 = par2 - data['cmt2']
deviation3 = par3 - data['cmt3']
deviation5 = par5 - data['cmt5']
deviation7 = par7 - data['cmt7']

plt.plot(data['date'],deviation2)
plt.title('2 Year Deviation')
plt.show()
plt.plot(data['date'],deviation3)
plt.title('3 Year Deviation')
plt.show()
plt.plot(data['date'],deviation5)
plt.title('5 Year Deviation')
plt.show()
plt.plot(data['date'],deviation7)
plt.title('7 Year Deviation')
plt.show()
```


2 Year Deviation
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
−0.0025
1988
1990
1992
1994
1996
1998
2000


3 Year Deviation
0.008
0.006
0.004
0.002
0.000
−0.002
−0.004
−0.006
1988
1990
1992
1994
1996
1998
2000


5 Year Deviation
0.002
0.000
−0.002
−0.004
−0.006
−0.008
−0.010
1988
1990
1992
1994
1996
1998
2000



Q7

Use only the first week x, y as an example to illustrate the results.

In [5]:

```
# 7

def duration(rate,T):
    face_value = 100
    coupon = face_value * rate / 2
    period = T * 2
    dur = 0
    for i in range(1,period+1):
        dur += coupon * D_T(x[0],y[0],alphax,betax,sigmax,betay,sigmay,i
/2) / face_value * i * 0.5
    dur +=  D_T(x[0],y[0],alphax,betax,sigmax,betay,sigmay,T) * T
    return dur * D_T(x[0],y[0],alphax,betax,sigmax,betay,sigmay,0.5)

def convexity(rate,T):
    price = 100
    numerator = 0
    cf = price * rate / 2
    k = 2
    for i in range(1,2*T + 1):
        D_i = D_T(x[0],y[0],alphax,betax,sigmax,betay,sigmay,i)
        numerator += i * (i + 1) * cf *  D_i
    return D_i / ((1 + rate/k) ** 2 * 4 * price)

dur2 = duration(0.0792,2)
dur10 = duration(0.0897,10)


convex2 = convexity(0.0792,2)
convex10 = convexity(0.0897,10)

def derivative(rate,alphax, betax, sigmax, betay, sigmay,beta,T):
    coupon = 100 * rate / 2
    sum = 0
    for i in range(1,2 * T + 1):
        dt = D_T(0.88327,-0.81815,alphax, betax, sigmax, betay, sigmay,i
/2)
        sum += B_T(beta,i/2) * dt
    output = -coupon * sum - D_T(.88327,-0.81815,alphax, betax, sigmax,
betay, sigmay,T) * B_T(beta,T)
    return output

dx2 = derivative(0.0792,alphax, betax, sigmax, betay, sigmay,betax,2)
dx10 = derivative(0.0897,alphax, betax, sigmax, betay, sigmay,betax,10)
dy2 = derivative(0.0792,alphax, betax, sigmax, betay, sigmay,betay,2)
dy10 = derivative(0.0897,alphax, betax, sigmax, betay, sigmay,betay,10)
result = list()

for i in range(1,31):
    first_dayx = x[0]
    first_dayy = y[0]
    par = par_rate(first_dayx, first_dayy, alphax, betax, sigmax, betay,
sigmay, i)
    dur = duration(par,i)
    convex = convexity(par,i)
    N2a, N10a = np.linalg.solve([[dur2, dur10], [convex2, convex10]], [-
dur, -convex])
```

```
    dx = derivative(par,alphax, betax, sigmax, betay, sigmay,betax,i)
    dy = derivative(par,alphax, betax, sigmax, betay, sigmay,betay,i)
    N2b, N10b = np.linalg.solve([[dx2, dx10], [dy2, dy10]], [-dx, -dy])
    result.append([N2a,N2b,N2a - N2b, N10a, N10b, N10a - N10b])

result_df = pd.DataFrame(result,index=[x for x in range(1,31)],columns=[
'N2a','N2b','N2a - N2b','N10a','N10b','N10a - N10b'])
print(result_df)
result_df[['N2a','N2b','N10a','N10b']].plot()
plt.show()
```

	N2a	N2b	N2a - N2b	N10a	N10b	N10a - N10b
1	-1.204906	-0.633266	-0.571640	0.186332	0.055696	0.130636
2	-0.990649	-1.106580	0.115931	-0.005453	0.000941	-0.006395
3	-0.810651	-1.184356	0.373706	-0.177319	-0.145231	-0.032088
4	-0.641871	-1.045674	0.403803	-0.337053	-0.314806	-0.022247
5	-0.486652	-0.836845	0.350193	-0.484159	-0.474357	-0.009802
6	-0.350152	-0.624450	0.274298	-0.616904	-0.614688	-0.002217
7	-0.234508	-0.431780	0.197272	-0.734476	-0.735609	0.001133
8	-0.139073	-0.264361	0.125288	-0.837113	-0.839263	0.002150
9	-0.061686	-0.121369	0.059684	-0.925759	-0.928035	0.002276
10	0.000347	-0.000055	0.000401	-1.001718	-1.004071	0.002353
11	0.049701	0.102663	-0.052962	-1.066419	-1.069219	0.002799
12	0.088782	0.189650	-0.100868	-1.121280	-1.125064	0.003784
13	0.119638	0.263413	-0.143775	-1.167625	-1.172967	0.005342
14	0.143958	0.326076	-0.182118	-1.206653	-1.214091	0.007438
15	0.163110	0.379422	-0.216312	-1.239428	-1.249430	0.010002
16	0.178188	0.424937	-0.246749	-1.266881	-1.279833	0.012951
17	0.190058	0.463855	-0.273797	-1.289818	-1.306019	0.016202
18	0.199404	0.497203	-0.297799	-1.308930	-1.328601	0.019671
19	0.206765	0.525836	-0.319071	-1.324813	-1.348099	0.023286
20	0.212563	0.550465	-0.337902	-1.337972	-1.364952	0.026980
21	0.217130	0.571688	-0.354558	-1.348841	-1.379536	0.030695
22	0.220727	0.590004	-0.369276	-1.357785	-1.392169	0.034384
23	0.223559	0.605834	-0.382274	-1.365116	-1.403122	0.038007
24	0.225786	0.619532	-0.393746	-1.371097	-1.412628	0.041531
25	0.227535	0.631400	-0.403865	-1.375952	-1.420883	0.044932
26	0.228906	0.641693	-0.412788	-1.379868	-1.428058	0.048190
27	0.229977	0.650628	-0.420651	-1.383005	-1.434298	0.051293
28	0.230812	0.658391	-0.427579	-1.385497	-1.439728	0.054231
29	0.231458	0.665139	-0.433681	-1.387455	-1.444455	0.057000
30	0.231956	0.671010	-0.439054	-1.388975	-1.448573	0.059598