Hypoglycemia Prediction using R & Python:

Data Preparation using R:

```
rm()
> # Load required libraries
> library(lubridate)
> library(dplyr)


#load data
> dd=read.csv("D:/shweta/Project_5 Hypoglecemia/kalman_data.csv", header=T)
> head(dd)
        Date  Time Code Value
1 04-21-1991  9:09   58   100
2 04-21-1991  9:09   33     9
3 04-21-1991  9:09   34    13
4 04-21-1991 17:08   62   119
5 04-21-1991 17:08   33     7
6 04-21-1991 22:51   48   123

> str(dd)
'data.frame':    29330 obs. of  4 variables:
 $ Date : Factor w/ 1141 levels "01-01-1990","01-01-1991",..: 308 308 308 308 308 308 312 312 312 312 ...
 $ Time : Factor w/ 1295 levels "0:30","00:00",..: 1280 1280 1280 789 789 1133 1235 1235 1235 620 ...
 $ Code : int  58 33 34 62 33 48 58 33 34 33 ...
 $ Value: num  100 9 13 119 7 123 216 10 13 2 …


#create column datetime
> datetime.vec = paste(dd$Date, dd$Time)
> dd$datetime = as.POSIXct(strptime(datetime.vec, "%m-%d-%Y %H:%M"))
 #convert date column into specific formate(yy-mm-dd)
> date.vec = paste(dd$Date)
```

```
> dd$date = as.Date(strptime(date.vec, "%m-%d-%Y"))
```

#select code which were corresponding to **glucose**

```
> dd1=subset(dd, (Code %in% c(48,57:65)))
#extarct columns date, Code and Value
> dd2=dd1[,c(6,3,4)]
> colnames(dd2)[2] <- "Glucose_code"
> colnames(dd2)[3] <- "Glucose_value"

> head(dd2,3)
        date Glucose_code Glucose_value
1 1991-04-21          58           100
4 1991-04-21          62           119
6 1991-04-21          48           123
```

#select code which were corresponding to **Insulin**

```
> dd3=subset(dd, (Code %in% c(33,34,35,56)))
#extarct columns date, Code and Value
> dd4=dd3[,c(6,3,4)]
> colnames(dd4)[2] <- "Insulin_code"
> colnames(dd4)[3] <- "Insulin_value"
> head(dd4,3)
        date Insulin_code Insulin_value
2 1991-04-21          33             9
3 1991-04-21          34            13
5 1991-04-21          33             7

#sort data by date
dd2$date[order(dd2$date , decreasing=T)]

#sort data by date
dd4$date[order(dd4$date , decreasing=T)]


#Here we want data in continuous date formate
#combine date weekly
#Summerize Value by mean
#and Code by latest value
> a=dd2 %>% group_by(Week=floor_date(date, "week")) %>%
summarize(Glucose_code=last(Glucose_code), Glucose_value=mean(Glucose_value))
> a1=dd4 %>% group_by(Week=floor_date(date, "week")) %>%
summarize(Insulin_code=last(Insulin_code), Insulin_value=mean(Insulin_value))
```

```
#Extract relevent variables code and value for further analysis
> b=a[,c(1,3)]   #glucose
> dim(b)
[1] 184   2

> b1=a1[,c(1,3)] #insulin
> dim(b1)
[1] 184   2

> new=data.frame(b,b1$Insulin_value)
> names(new)
[1] "Week"            "Glucose_value"    "b1.Insulin_value"


 #residuals noise for process cov matrix
> mod1 = lm(Glucose_value~ b1.Insulin_value, data = new)
> mod2 = lm(b1.Insulin_value~ Glucose_value, data = new)
> res1=resid(mod1)
> res2=resid(mod2)
> new1=data.frame(b,b1$Insulin_value,res1,res2)
> names(new1)
[1] "Week"            "Glucose_value"    "b1.Insulin_value" "res1"
[5] "res2"


#export data as csv named as new2
```

Translated data for further analysis:

	A	B	C	
1	Date	Glucose_value	Insulin_value	re
2	1988-03-27	148.5	20	
3	1988-04-03	129.7272727273	18.25	
4	1988-04-10	115.2222222222	20	
5	1988-04-17	124.4166666667	20	
6	1988-04-24	115.4444444444	20	
7	1988-05-01	112.4166666667	20	
8	1988-05-08	116.1111111111	20	
9	1988-05-15	128.2857142857	20	
10	1988-05-22	136.8	20	
11	1988-05-29	113.4090909091	20	
12	1988-06-05	102.8571428571	16.4285714286	
13	1988-06-12	114.2857142857	14.5714285714	
14	1988-06-19	110.7142857143	12	
15	1988-06-26	133.3157894737	12	
16	1988-07-03	131.15	12	
17	1988-07-10	143.8648648649	17.0909090909	
18	1988-07-17	139.3142857143	18.2	
19	1988-07-24	124.76	17.7142857143	
20	1988-07-31	144	16.2857142857	
21	1988-08-07	141.0476190476	17	
22	1988-08-14	154.7391304348	17.7142857143	
23	1988-08-21	149.5714285714	17	
24	1988-08-28	126	16.2857142857	
25	1988-09-04	125.4	17	
26	1988-09-11	127.6111111111	15.3333333333	
27	1988-09-18	144.0833333333	17.7142857143	
28	1988-09-25	127.0384615385	15.75	
29	1988-10-02	164.9545454545	14.2222222222	
30	1988-10-09	146.375	7.9375	
31	1988-10-16	112.75	8	
32	1988-10-23	111.2631578947	9.75	
33	1988-10-30	125.3	8.9090909091	
34	1988-11-06	113.8666666667	12	
35	1988-11-13	121.9166666667	12	

Here we have two states glucose_value and Insulin value as input vector.

Kalman filter in python:

```
# Importing libraries
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
from scipy.stats import norm
import math

%matplotlib inline

import seaborn as sns

import statsmodels.api as sm
import statsmodels.tsa.api as smt
sns.set(style='ticks', context='talk')
```

```
In [55]: #Load data
         data1= pd.read_csv('D:/shweta/Project_5 Hypoglecemia/new2.csv',sep=';')
         data1.shape

Out[55]: (184, 5)
```

```
In [56]: M=data[data.columns].mean()
         M

Out[56]: Glucose_value       1.541637e+02
         b1.Insulin_value    1.198956e+01
         res1               -1.083945e-12
         res2               -1.630359e-12
         dtype: float64
```

```
In [57]: #Covariance
         Cov=data[data.columns].cov()
         Cov

Out[57]:
```

	Glucose_value	b1.Insulin_value	res1	res2
Glucose_value	6.583621e+02	-1.262094e+01	6.493633e+02	-4.390504e-12
b1.Insulin_value	-1.262094e+01	1.770100e+01	-3.919740e-11	1.745906e+01
res1	6.493633e+02	-3.919740e-11	6.493633e+02	1.244844e+01
res2	-4.390504e-12	1.745906e+01	1.244844e+01	1.745906e+01

In [58]:
```
#Initial state
x = np.matrix([[0.0, 0.0, 0.0, 0.0]]).T
print(x, x.shape)
```

```
[[ 0.]
 [ 0.]
 [ 0.]
 [ 0.]] (4, 1)
```

In [59]:
```
#Initial matrix
P=100.0*np.eye(4)
print(P, P.shape)
```

```
[[ 100.    0.    0.    0.]
 [   0.  100.    0.    0.]
 [   0.    0.  100.    0.]
 [   0.    0.    0.  100.]] (4, 4)
```

In [60]:
```
#Dinamic matrix
dt = 0.1 # Time Step between Filter Steps

A = np.matrix([[1.0, 0.0, dt, 0.0],
              [0.0, 1.0, 0.0, dt],
              [0.0, 0.0, 1.0, 0.0],
              [0.0, 0.0, 0.0, 1.0]])
print(A, A.shape)
```

```
[[ 1.   0.   0.1  0. ]
 [ 0.   1.   0.   0.1]
 [ 0.   0.   1.   0. ]
 [ 0.   0.   0.   1. ]] (4, 4)
```

In [61]:
```
#measurement matrix
H = np.matrix([[0.0, 0.0, 1.0, 0.0],
              [0.0, 0.0, 0.0, 1.0]])
print(H, H.shape)
```

```
[[ 0.  0.  1.  0.]
 [ 0.  0.  0.  1.]] (2, 4)
```

In [62]:
```python
# R matrix
R = np.matrix([[6.493633e+02,0.0],
              [0.0, 1.745906e+01]])
print(R, R.shape)
```

```
[[ 649.3633     0.      ]
 [   0.        17.45906]] (2, 2)
```

In [64]:
```python
#Process Noise Covariance Q

#Covariance
Cov1=Cov
Cov1
```

Out[64]:

	Glucose_value	b1.Insulin_value	res1	res2
Glucose_value	6.583621e+02	-1.262094e+01	6.493633e+02	-4.390504e-12
b1.Insulin_value	-1.262094e+01	1.770100e+01	-3.919740e-11	1.745906e+01
res1	6.493633e+02	-3.919740e-11	6.493633e+02	1.244844e+01
res2	-4.390504e-12	1.745906e+01	1.244844e+01	1.745906e+01

In [65]:
```python
#matrix Q
Q=np.matrix([[6.583621e+02, -1.262094e+01, 6.493633e+02, -4.390504e-12],
             [-1.262094e+01, 1.770100e+01,  -3.919740e-11, 1.745906e+01],
             [6.493633e+02, -3.919740e-11, 6.493633e+02, 1.244844e+01],
             [-4.390504e-12, 1.745906e+01, 1.244844e+01, 1.745906e+01]])
print(Q, Q.shape)
```

```
[[  6.58362100e+02  -1.26209400e+01   6.49363300e+02  -4.39050400e-12]
 [ -1.26209400e+01   1.77010000e+01  -3.91974000e-11   1.74590600e+01]
 [  6.49363300e+02  -3.91974000e-11   6.49363300e+02   1.24484400e+01]
 [ -4.39050400e-12   1.74590600e+01   1.24484400e+01   1.74590600e+01]] (4, 4)
```

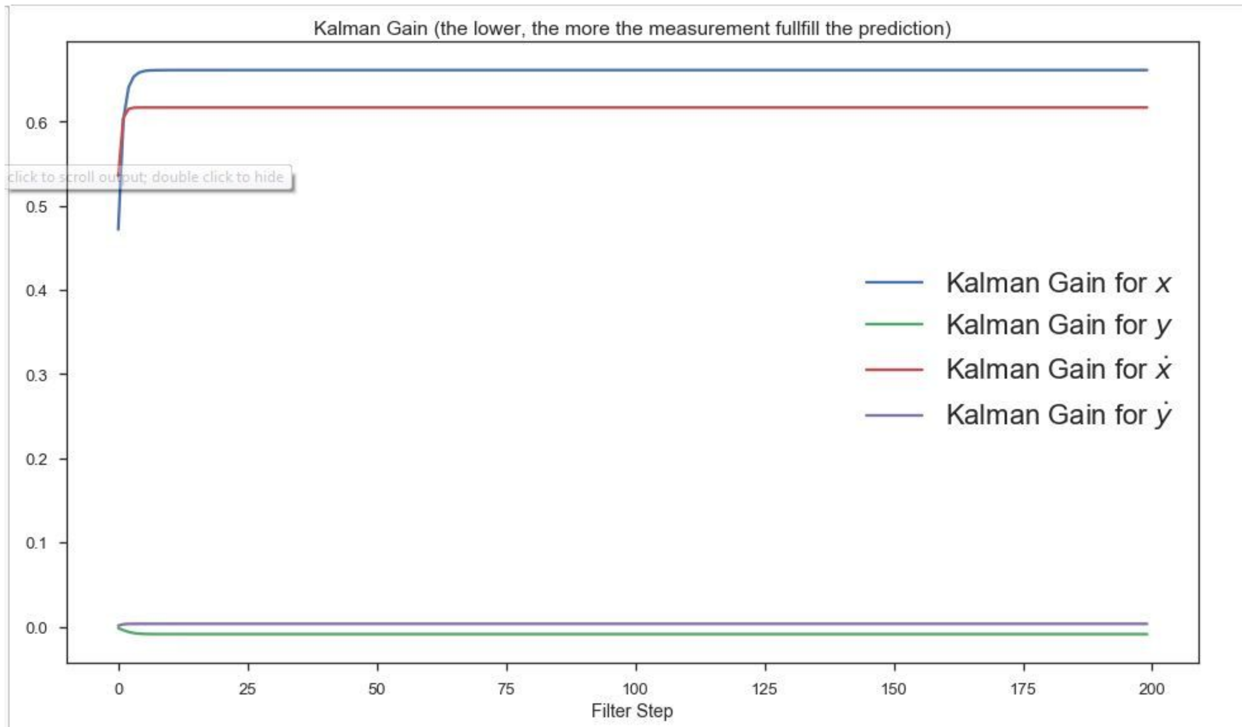
In [66]:
```python
I = np.eye(4)
print(I, I.shape)
```

```
[[ 1.  0.  0.  0.]
 [ 0.  1.  0.  0.]
 [ 0.  0.  1.  0.]
 [ 0.  0.  0.  1.]] (4, 4)
```

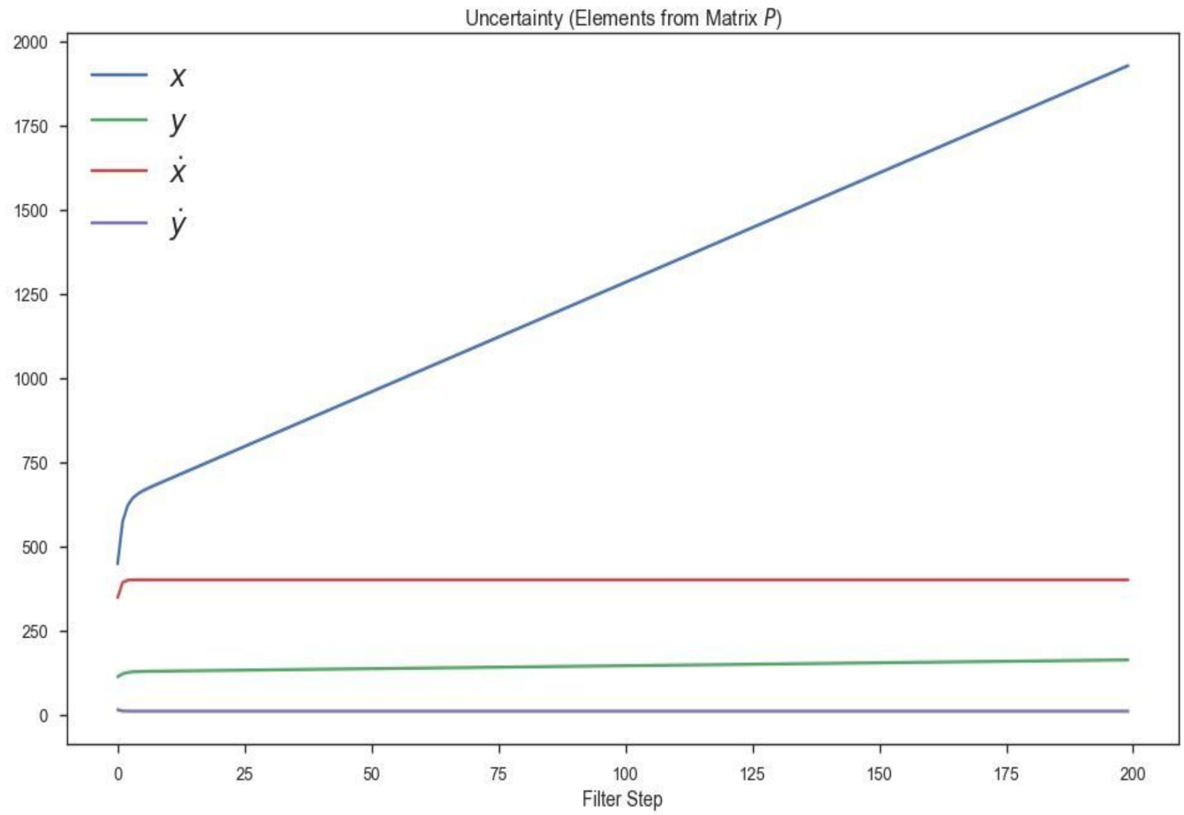
```
# Preallocation for Plotting
xt = []
yt = []
dxt= []
dyt= []
Zx = []
Zy = []
```

```
Px = []
Py = []
Pdx= []
Pdy= []
Rdx= []
Rdy= []
Kx = []
Ky = []
Kdx= []
Kdy= []

def savestates(x, Z, P, R, K):
    xt.append(float(x[0]))
    yt.append(float(x[1]))
    dxt.append(float(x[2]))
    dyt.append(float(x[3]))
    Zx.append(float(Z[0]))
    Zy.append(float(Z[1]))
    Px.append(float(P[0,0]))
    Py.append(float(P[1,1]))
    Pdx.append(float(P[2,2]))
    Pdy.append(float(P[3,3]))
    Rdx.append(float(R[0,0]))
    Rdy.append(float(R[1,1]))
    Kx.append(float(K[0,0]))
    Ky.append(float(K[1,0]))
    Kdx.append(float(K[2,0]))
    Kdy.append(float(K[3,0]))



for n in range(len(measurements[0])):

    # Time Update (Prediction)
    # ========================
    # Project the state ahead
    x = A*x

    # Project the error covariance ahead
    P = A*P*A.T + Q


    # Measurement Update (Correction)
```

```
    # ===============================
    # Compute the Kalman Gain
    S = H*P*H.T + R
    K = (P*H.T) * np.linalg.pinv(S)


    # Update the estimate via z
    Z = measurements[:,n].reshape(2,1)
    y = Z - (H*x)                            # Innovation or Residual
    x = x + (K*y)

    # Update the error covariance
    P = (I - (K*H))*P



    # Save states (for Plotting)
    savestates(x, Z, P, R, K)




#Let's take a look at the filter performance
#Kalman Gains K


def plot_K():
    fig = plt.figure(figsize=(16,9))
    plt.plot(range(len(measurements[0])),Kx, label='Kalman Gain for $x$')
    plt.plot(range(len(measurements[0])),Ky, label='Kalman Gain for $y$')
    plt.plot(range(len(measurements[0])),Kdx, label='Kalman Gain for $\dot x$')
    plt.plot(range(len(measurements[0])),Kdy, label='Kalman Gain for $\dot y$')

    plt.xlabel('Filter Step')
    plt.ylabel('')
    plt.title('Kalman Gain (the lower, the more the measurement fullfill the prediction)')
    plt.legend(loc='best',prop={'size':22})

plot_K()
```




```
#Uncertainty Matrix P

def plot_P():
    fig = plt.figure(figsize=(16,9))
    plt.plot(range(len(measurements[0])),Px, label='$x$')
    plt.plot(range(len(measurements[0])),Py, label='$y$')
    plt.plot(range(len(measurements[0])),Pdx, label='$\dot x$')
    plt.plot(range(len(measurements[0])),Pdy, label='$\dot y$')

    plt.xlabel('Filter Step')
    plt.ylabel('')
    plt.title('Uncertainty (Elements from Matrix $P$)')
    plt.legend(loc='best',prop={'size':22})

plot_P()
```


Uncertainty (Elements from Matrix P)
x
y
$\dot{x}$
$\dot{y}$
2000
1750
1500
1250
1000
750
500
250
0
0
25
50
75
100
125
150
175
200
Filter Step