individual-assignment

September 11, 2024

1 Archit Murkunde - 2845576 - a.u.murkunde@student.vu.nl

```
[2]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
```

```
[3]: data = pd.read_csv(r"C:\Users\archi\OneDrive\Desktop\VU\My VU\P1\Adv␣
       ↪Econ\Assignment 1 - IA\heart_rates.csv", index_col = "10min_Interval")
```

```
[4]: data = data[:2000] #first 2000 obs for Question 1 to 6
```

1.1 Question 1

```
[6]: df = pd.DataFrame({"No of Obs": data.count(),"Mean": data.mean(), "Median":␣
       ↪data.median(), "Std Dev": data.std(), "Skewness": data.skew(), "Excess␣
       ↪Kurtosis": data.kurt(), "Min":data.min(), "Max":data.max()})
```

```
[7]: df
```

```
[7]:     No of Obs       Mean     Median   Std Dev  Skewness  Excess Kurtosis  \
     V1       2000  69.211311  69.026500  6.157390  0.073437        -0.252180
     V2       2000  74.506205  74.592051  5.782674  0.018712        -0.069098
     V3       2000  86.092599  85.948986  5.225615  0.049079        -0.000813
     V4       2000  78.620294  78.552662  5.156608 -0.042983        -0.109199

               Min         Max
     V1  51.041275   87.324332
     V2  56.678226   93.344119
     V3  68.504259  102.644232
     V4  61.628978   94.387767
```

```
[ ]:
```

1.1.1 Question 2

[9]:

```
data_1 = data.copy()
```

Theta 1

[11]:

```
x1_mean = np.round(data[:100].mean(),2)
#print(x1_mean)
```

[12]:

```
#Creating DataFrame for Theta_2
theta_1 = np.ones((4, 4))
theta_1[0] = theta_1[0]*x1_mean*0.01
theta_1[1] = theta_1[1]*0.15
theta_1[2] = theta_1[2]*0.84
theta_1[3] = theta_1[3]*4
theta_1 = pd.DataFrame(theta_1)
theta_1.columns = data_1.columns
theta_1.index = ["omega_i", "alpha_i", "beta_i", "sigma_i"]
```

[13]:

```
theta_1
```

[13]:

```
             V1     V2      V3      V4
omega_i  0.6968  0.707  0.8643  0.7928
alpha_i  0.1500  0.150  0.1500  0.1500
beta_i   0.8400  0.840  0.8400  0.8400
sigma_i  4.0000  4.000  4.0000  4.0000
```

[14]:

```
# Generating random numbers
eps = np.ones((2000,4))
np.random.seed(42)
n = 2000
for i in range(4):
    eps[:,i] = eps[:,i]*np.round(np.random.normal(0,theta_1.iloc[3,3],size =␣
 ↪n),4)
```

[15]:

```
#Creating an array of 1's for calculation of filtered heart rates
avg_x = np.ones((2000,4))
avg_x[0] = avg_x[0]*x1_mean

#Applying the given formula
for i in range(1,2000):
    avg_x[i] = np.round(np.array(theta_1.iloc[0]) + np.array(theta_1.
 ↪iloc[1])*(avg_x[i-1] + eps[i]) + np.array(theta_1.iloc[2])*avg_x[i-1],4)

avg_x = pd.DataFrame(avg_x)
avg_x.columns = data_1.columns + "_avg_heart" # I have named it average heart␣
 ↪rate, but it is the filtered hr based on the given formula
avg_x.index = data_1.index
```

```
[16]: new_df_t1 = pd.concat([data_1,avg_x], axis =1 )
```

Theta 2

```
[18]: x2_mean = np.round(data[:100].mean(),2)
      #print(x1_mean)
```

```
[19]: #Creating DataFrame for Theta_2
      theta_2 = np.ones((4, 4))
      theta_2[0] = theta_2[0]*x2_mean*0.01
      theta_2[1] = theta_2[1]*0.84
      theta_2[2] = theta_2[2]*0.15
      theta_2[3] = theta_2[3]*4
      theta_2 = pd.DataFrame(theta_2)
      theta_2.columns = data_1.columns
      theta_2.index = ["omega_i", "alpha_i", "beta_i", "sigma_i"]
      theta_2
```

```
[19]:              V1     V2      V3      V4
      omega_i  0.6968  0.707  0.8643  0.7928
      alpha_i  0.8400  0.840  0.8400  0.8400
      beta_i   0.1500  0.150  0.1500  0.1500
      sigma_i  4.0000  4.000  4.0000  4.0000
```

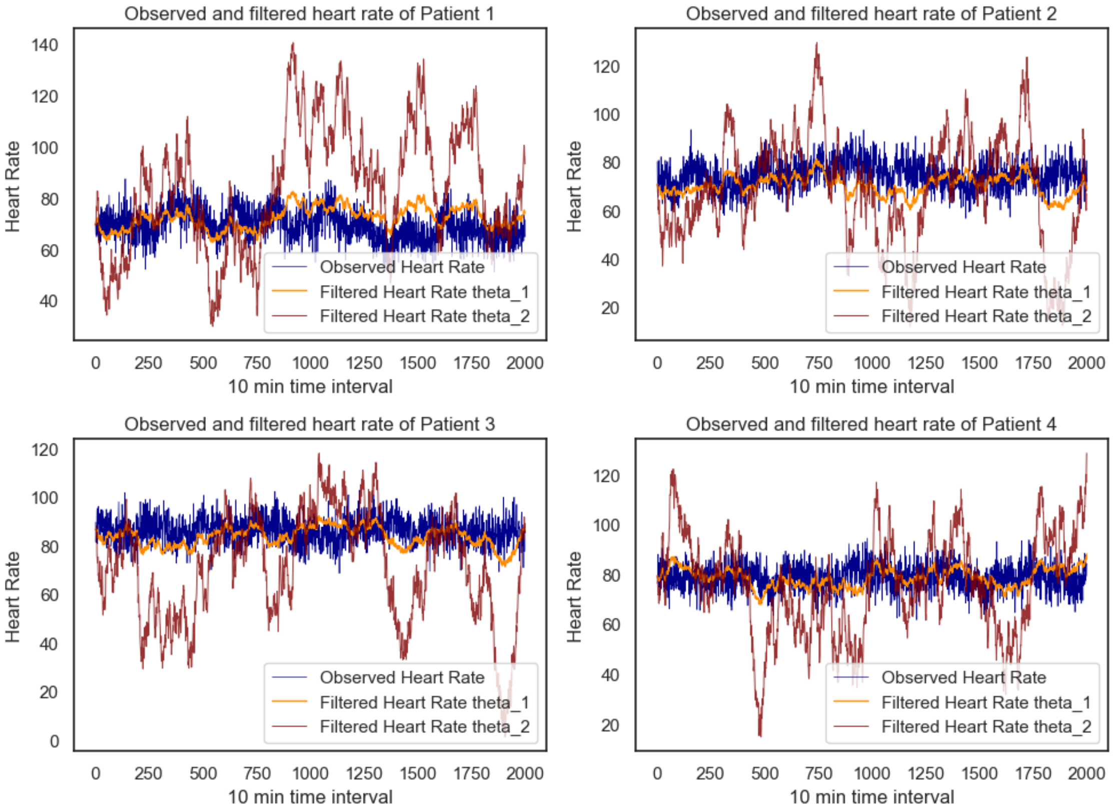
```
[20]: #Creating an array of 1's for calculation of filtered heart rates
      avg_x_2 = np.ones((2000,4))
      avg_x_2[0] = avg_x_2[0]*x1_mean
      for i in range(1,2000):
          avg_x_2[i] = np.round(np.array(theta_2.iloc[0]) + np.array(theta_2.
       ↪iloc[1])*(avg_x_2[i-1] + eps[i]) + np.array(theta_2.iloc[2])*avg_x_2[i-1],4)
```

```
[21]: avg_x_2 = pd.DataFrame(avg_x_2)
      avg_x_2.columns = data_1.columns + "_avg_heart"
      avg_x_2.index = data_1.index
      new_df_t2 = pd.concat([data_1, avg_x_2], axis =1 )
```

1.1.2 Plots

```
[91]: fig, axs = plt.subplots(2, 2, figsize=(10, 8))

      axs[0, 0].plot(new_df_t2.index, new_df_t2['V1'], label='Observed Heart Rate',␣
       ↪color='#00008B', linewidth=0.5)
      axs[0, 0].plot(new_df_t1.index, new_df_t1['V1_avg_heart'], label='Filtered␣
       ↪Heart Rate theta_1', color='#FF8C00', linewidth=1)
      axs[0, 0].plot(new_df_t2.index, new_df_t2['V1_avg_heart'], label='Filtered␣
       ↪Heart Rate theta_2', color='maroon', linewidth=0.7, alpha=0.8)
      axs[0, 0].set_title('Observed and filtered heart rate of Patient 1')
```

```
axs[0, 0].set_ylabel('Heart Rate')
axs[0, 0].set_xlabel('10 min time interval')
axs[0, 0].legend(loc='lower right')

axs[0, 1].plot(new_df_t2.index, new_df_t2['V2'], label='Observed Heart Rate',␣
  ↪color='#00008B', linewidth=0.5)
axs[0, 1].plot(new_df_t1.index, new_df_t1['V2_avg_heart'], label='Filtered␣
  ↪Heart Rate theta_1', color='#FF8C00', linewidth=1)
axs[0, 1].plot(new_df_t2.index, new_df_t2['V2_avg_heart'], label='Filtered␣
  ↪Heart Rate theta_2', color='maroon', linewidth=0.7, alpha=0.8)
axs[0, 1].set_title('Observed and filtered heart rate of Patient 2')
axs[0, 1].set_ylabel('Heart Rate')
axs[0, 1].set_xlabel('10 min time interval')
axs[0, 1].legend(loc='lower right')

axs[1, 0].plot(new_df_t2.index, new_df_t2['V3'], label='Observed Heart Rate',␣
  ↪color='#00008B', linewidth=0.5)
axs[1, 0].plot(new_df_t1.index, new_df_t1['V3_avg_heart'], label='Filtered␣
  ↪Heart Rate theta_1', color='#FF8C00', linewidth=1)
axs[1, 0].plot(new_df_t2.index, new_df_t2['V3_avg_heart'], label='Filtered␣
  ↪Heart Rate theta_2', color='maroon', linewidth=0.7, alpha=0.8)
axs[1, 0].set_title('Observed and filtered heart rate of Patient 3')
axs[1, 0].set_ylabel('Heart Rate')
axs[1, 0].set_xlabel('10 min time interval')
axs[1, 0].legend(loc='lower right')

axs[1, 1].plot(new_df_t2.index, new_df_t2['V4'], label='Observed Heart Rate',␣
  ↪color='#00008B', linewidth=0.5)
axs[1, 1].plot(new_df_t1.index, new_df_t1['V4_avg_heart'], label='Filtered␣
  ↪Heart Rate theta_1', color='#FF8C00', linewidth=1)
axs[1, 1].plot(new_df_t2.index, new_df_t2['V4_avg_heart'], label='Filtered␣
  ↪Heart Rate theta_2', color='maroon', linewidth=0.7, alpha=0.8)
axs[1, 1].set_title('Observed and filtered heart rate of Patient 4')
axs[1, 1].set_ylabel('Heart Rate')
axs[1, 1].set_xlabel('10 min time interval')
axs[1, 1].legend(loc='lower right')

#fig.suptitle('Comparison of Observed and Filtered Heart Rates for 4 Patients',␣
  ↪fontsize=14)

fig.text(0.5, 0.01, 'Filtered heart rate signals use two different parameter␣
  ↪sets.', ha='center', fontsize=10)

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



Filtered heart rate signals use two different parameter sets.

```
[ ]:
```

1.1.3 Question 4

```
[25]: data_3 = data.copy()
```

```
[26]: omega = 0.7
      alpha = 0.15
      beta = 0.84
      sigma = 4
      np.random.seed(42)
      n = 2000
      eps = np.round(np.random.normal(0, theta_1.iloc[3,3], size = n),4)
```

```
[27]: x_mean = np.round(data_3[:100].mean(),2)
      avg_x = np.ones((2000,4))
      avg_x[0] = avg_x[0]*x_mean
```

[28]:

```
ml_df = pd.DataFrame(avg_x)
ml_df.columns = data.columns + "_fil_heart"
ml_df.index = data.index
ml_df = pd.concat([data_3, ml_df], axis = 1)
```

[29]:

```
# Generating Filtered heart rate for each patient
for i in range(2, len(ml_df) + 1):
    ml_df.loc[i, 'V1_fil_heart'] = omega + alpha * ml_df.loc[i-1, 'V1'] + beta␣
 ↪* ml_df.loc[i-1, 'V1_fil_heart']

for i in range(2, len(ml_df) + 1):
    ml_df.loc[i, 'V2_fil_heart'] = omega + alpha * ml_df.loc[i-1, 'V2'] + beta␣
 ↪* ml_df.loc[i-1, 'V2_fil_heart']

for i in range(2, len(ml_df) + 1):
    ml_df.loc[i, 'V3_fil_heart'] = omega + alpha * ml_df.loc[i-1, 'V3'] + beta␣
 ↪* ml_df.loc[i-1, 'V3_fil_heart']

for i in range(2, len(ml_df) + 1):
    ml_df.loc[i, 'V4_fil_heart'] = omega + alpha * ml_df.loc[i-1, 'V4'] + beta␣
 ↪* ml_df.loc[i-1, 'V4_fil_heart']
```

[30]:

```
### ML Estimator function
def ml_estimator(x,mu,sigma):
    ml_estimator = 0
    for i in range(1,len(x)+1):
        ml_estimator = ml_estimator - np.log(np.sqrt(2*np.pi*sigma*sigma)) - 0.
 ↪5 * (((x[i] - mu[i])**2)/(sigma*sigma))

    return ml_estimator
```

[31]:

```
d = {}
for i in range(1,5):
    ml = ml_estimator(ml_df["V"+str(i)], mu =␣
 ↪ml_df['V'+str(i)+"_fil_heart"],sigma =  4)
    d["ml_" + str(i)] = np.round(ml,4)
```

[32]:

```
# ML Estimators for all the patients
d
```

[32]:

```
{'ml_1': -6173.7754,
 'ml_2': -6206.3222,
 'ml_3': -6283.3303,
 'ml_4': -6161.6393}
```

[ ]:

1.1.4 Question 5

[34]:

```python
from scipy.optimize import minimize
```

[35]:

```python
data_3 = data_3.reset_index()
```

[36]:

```python
import numpy as np
from scipy.optimize import minimize
# defining ml estimator function for minimization
def ml_estimator(x, omega, alpha, beta, sigma):
    ml_estimator = 0
    mu = np.zeros(len(x))
    mu[0] = np.mean(x[:100])

    for i in range(1, len(x)):
        mu[i] = omega + alpha * x[i-1] + beta * mu[i-1]
        ml_estimator -= np.log(np.sqrt(2 * np.pi * sigma ** 2))
        ml_estimator -= 0.5 * ((x[i] - mu[i]) ** 2) / (sigma ** 2)


    return -ml_estimator  # Return the negative log-likelihood for minimization

# Function to optimize parameters for a given patient
def estimate_params(x):
    initial_guess = [0.7, 0.15, 0.84, 4]  # Initial values

    result = minimize(lambda params: ml_estimator(x, params[0], params[1], params[2], params[3]),
                      x0=initial_guess,
                      bounds=[(None, None), (0, 1), (0, 1), (1e-6, None)])

    if result.success:
        optimal_params = result.x
        optimal_log_likelihood = -result.fun
        return optimal_params, optimal_log_likelihood
    else:
        print("Optimization failed")
        return None, None

patients = ['V1', 'V2', 'V3', 'V4']

results = {}

for patient in patients:
    x = ml_df[patient].values
    params, log_likelihood = estimate_params(x)
    results[patient] = {'omega': params[0], 'alpha': params[1], 'beta': params[2], 'sigma': params[3], 'log_likelihood': log_likelihood}

# Convert results to a pandas DataFrame
import pandas as pd
df_results = pd.DataFrame(results).T
df_results.columns = ["Omega", "Alpha", "Beta", "Sigma", "Optimal Log-Likelihood"]

df_results = df_results.T
df_results.columns = ["Patient_"+str(i) for i in range(1,5)]
df_styled = df_results.style.format("{:.4f}").background_gradient(cmap='coolwarm')
df_styled
```

[36]:

```
<pandas.io.formats.style.Styler at 0x27d2acb9130>
```

[103]:

```
df_results
```

[103]:

```
                        Patient_1    Patient_2    Patient_3    Patient_4
Omega                    0.571170     1.437848     2.950768     1.740650
Alpha                    0.092252     0.090447     0.067152     0.069794
Beta                     0.899493     0.890277     0.898564     0.908069
Sigma                    4.967744     4.982734     4.974354     4.892178
Optimal Log-Likelihood -6040.597582 -6056.416574 -6055.707718 -6010.104030
```

1.1.5 Question 6

[38]:

```
data_4 = data.copy()
```

[39]:

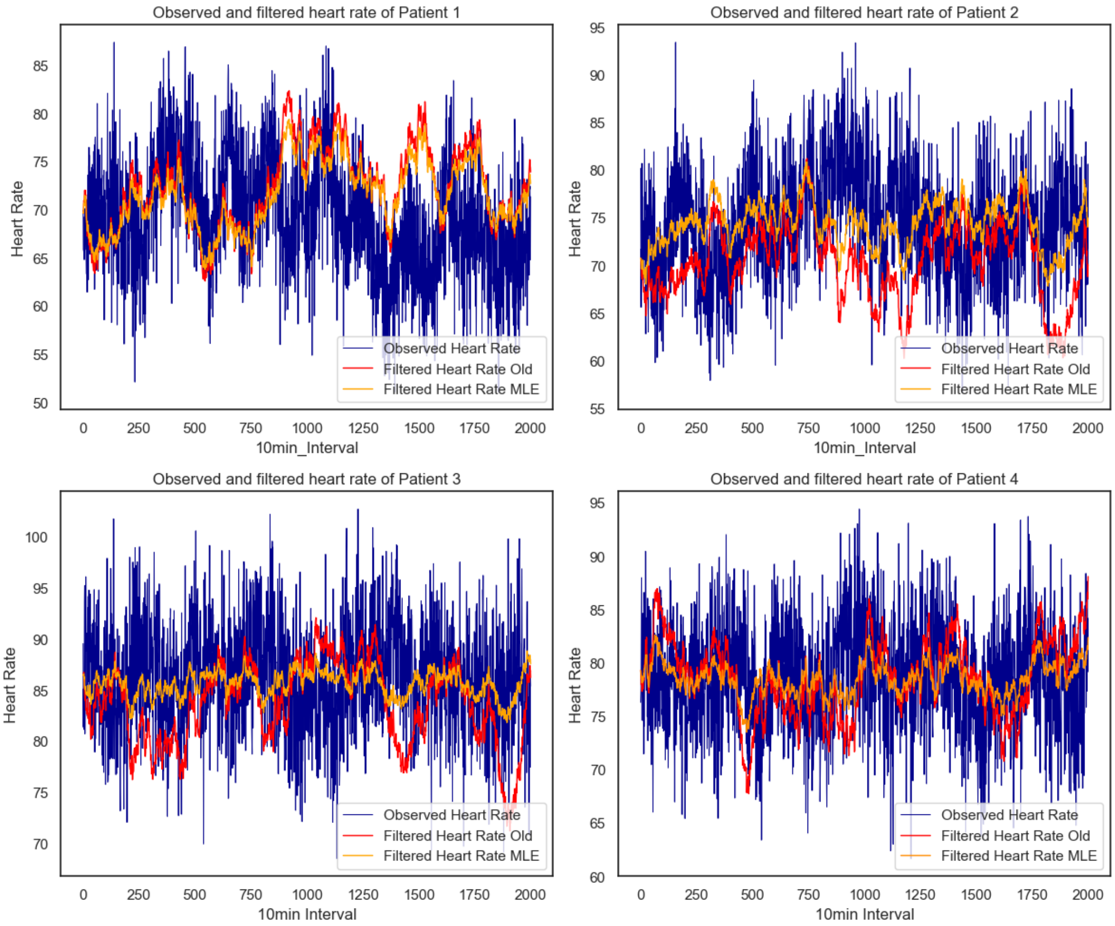
```
eps = np.ones((2000,4))
np.random.seed(42)
n = 2000
for i in range(4):
    eps[:,i] = eps[:,i]*np.round(np.random.normal(0,df_results.iloc[3,i],size = n),4)
```

[40]:

```
x_mean = np.round(data_4[:100].mean(),2)
avg_x = np.ones((2000,4))
avg_x[0] = avg_x[0]*x_mean
for i in range(1,2000):
    avg_x[i] = np.round(np.array(df_results.iloc[0]) + np.array(df_results.iloc[1])*(avg_x[i-1] + eps[i]) + np.array(df_results.iloc[2])*avg_x[i-1],4)

avg_x = pd.DataFrame(avg_x)
avg_x.columns = data_4.columns + "_filtered_new"
avg_x.index = data_4.index
new_df_t3 = pd.concat([new_df_t1,avg_x], axis = 1)
```

[101]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="white")
fig, axs = plt.subplots(2, 2, figsize=(12, 10))


sns.lineplot(x=new_df_t3.index, y=new_df_t3['V1'], ax=axs[0, 0],␣
 ↪label='Observed Heart Rate', color='#00008B', linewidth=0.7)
sns.lineplot(x=new_df_t3.index, y=new_df_t3['V1_avg_heart'], ax=axs[0, 0],␣
 ↪label='Filtered Heart Rate Old', color='red', linewidth=1)
sns.lineplot(x=new_df_t3.index, y=new_df_t3['V1_filtered_new'], ax=axs[0, 0],␣
 ↪label='Filtered Heart Rate MLE', color='orange', linewidth=1)
axs[0, 0].set_title('Observed and filtered heart rate of Patient 1')
axs[0, 0].set_ylabel('Heart Rate')
axs[0, 0].legend(loc='lower right')

sns.lineplot(x=new_df_t3.index, y=new_df_t3['V2'], ax=axs[0, 1],␣
 ↪label='Observed Heart Rate', color='#00008B', linewidth=0.7)
sns.lineplot(x=new_df_t3.index, y=new_df_t3['V2_avg_heart'], ax=axs[0, 1],␣
 ↪label='Filtered Heart Rate Old', color='red', linewidth=1)
sns.lineplot(x=new_df_t3.index, y=new_df_t3['V2_filtered_new'], ax=axs[0, 1],␣
 ↪label='Filtered Heart Rate MLE', color='orange', linewidth=1)
axs[0, 1].set_title('Observed and filtered heart rate of Patient 2')
axs[0, 1].set_ylabel('Heart Rate')
axs[0, 1].legend(loc='lower right')

sns.lineplot(x=new_df_t3.index, y=new_df_t3['V3'], ax=axs[1, 0],␣
 ↪label='Observed Heart Rate', color='#00008B', linewidth=0.7)
sns.lineplot(x=new_df_t3.index, y=new_df_t3['V3_avg_heart'], ax=axs[1, 0],␣
 ↪label='Filtered Heart Rate Old', color='red', linewidth=1)
sns.lineplot(x=new_df_t3.index, y=new_df_t3['V3_filtered_new'], ax=axs[1, 0],␣
 ↪label='Filtered Heart Rate MLE', color='orange', linewidth=1)
axs[1, 0].set_title('Observed and filtered heart rate of Patient 3')
axs[1, 0].set_xlabel('10min Interval')
axs[1, 0].set_ylabel('Heart Rate')
axs[1, 0].legend(loc='lower right')


sns.lineplot(x=new_df_t3.index, y=new_df_t3['V4'], ax=axs[1, 1],␣
 ↪label='Observed Heart Rate', color='#00008B', linewidth=0.7)
sns.lineplot(x=new_df_t3.index, y=new_df_t3['V4_avg_heart'], ax=axs[1, 1],␣
 ↪label='Filtered Heart Rate Old', color='red', linewidth=1)
```

```
sns.lineplot(x=new_df_t3.index, y=new_df_t3['V4_filtered_new'], ax=axs[1, 1],␣
 ↪label='Filtered Heart Rate MLE', color='#FF8C00', linewidth=1)
axs[1, 1].set_title('Observed and filtered heart rate of Patient 4')
axs[1, 1].set_xlabel('10min Interval')
axs[1, 1].set_ylabel('Heart Rate')
axs[1, 1].legend(loc='lower right')

fig.text(0.5, -0.01, 'The plots show the observed and the filtered heart rate␣
 ↪based on MLE and default parameter set.', ha='center', fontsize=12)
plt.tight_layout()
plt.show()
```



The plots show the observed and the filtered heart rate based on MLE and default parameter set.

```
[ ]:
```

1.1.6 Question 7

[43]:

```
data = pd.read_csv(r"C:\Users\archi\OneDrive\Desktop\VU\My VU\P1\Adv␣
  ↪Econ\Assignment 1 - IA\heart_rates.csv", index_col = "10min_Interval")
```

[44]:

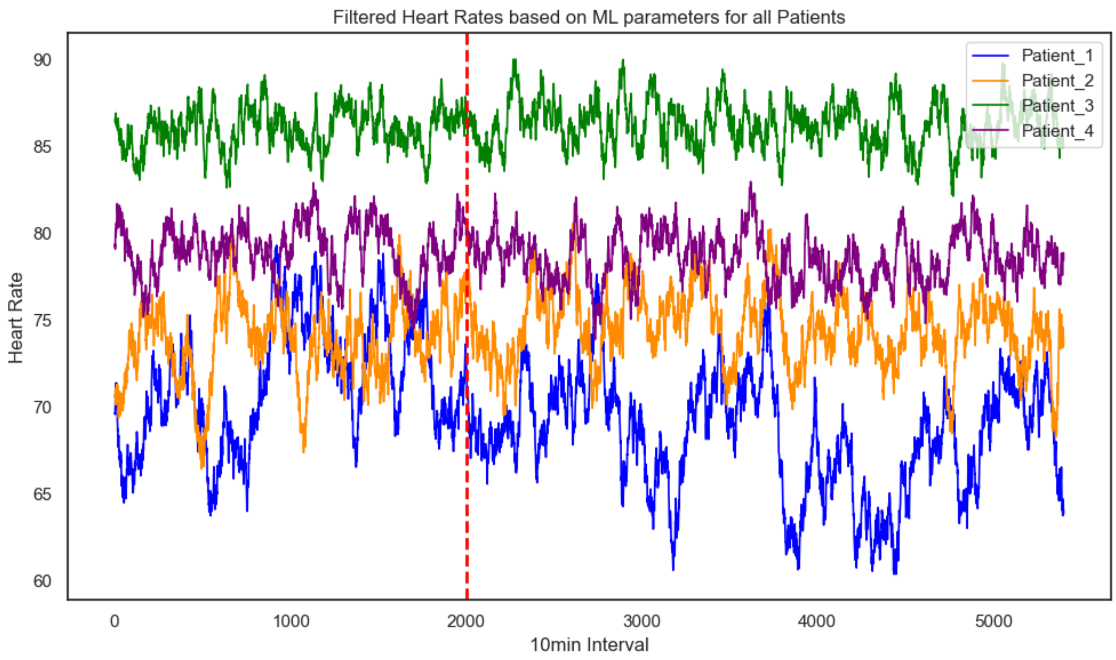
```
eps = np.ones((len(data),4))
np.random.seed(42)
n = len(data)
for i in range(4):
    eps[:,i] = eps[:,i]*np.round(np.random.normal(0,df_results.iloc[3,i],size =␣
  ↪n),4)
```

[45]:

```
x_mean = np.round(data[:100].mean(),2)
avg_x = np.ones((n,4))
avg_x[0] = avg_x[0]*x_mean
for i in range(1,n):
    avg_x[i] = np.round(np.array(df_results.iloc[0]) + np.array(df_results.
  ↪iloc[1])*(avg_x[i-1] + eps[i]) + np.array(df_results.iloc[2])*avg_x[i-1],4)

avg_x = pd.DataFrame(avg_x)
avg_x.columns = ["Patient_"+str(i) for i in range(1,5)]
avg_x.index = data.index
```

[46]:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="white")
plt.figure(figsize=(10, 6))
sns.lineplot(x=avg_x.index, y=avg_x['Patient_1'], label='Patient_1',␣
  ↪color='Blue', linewidth=1.2)
sns.lineplot(x=avg_x.index, y=avg_x['Patient_2'], label='Patient_2',␣
  ↪color='#FF8C00', linewidth=1.2)
sns.lineplot(x=avg_x.index, y=avg_x['Patient_3'], label='Patient_3',␣
  ↪color='green', linewidth=1.2)
sns.lineplot(x=avg_x.index, y=avg_x['Patient_4'], label='Patient_4',␣
  ↪color='purple', linewidth=1.2)

plt.axvline(x=2000, color='red', linestyle='--', linewidth=2)
plt.title('Filtered Heart Rates based on ML parameters for all Patients')
plt.xlabel('10min Interval')
plt.ylabel('Heart Rate')
plt.legend(loc='upper right')
plt.tight_layout()
plt.show()
```


Filtered Heart Rates based on ML parameters for all Patients
Patient_1
Patient_2
Patient_3
Patient_4
Heart Rate
90
85
80
75
70
65
60
0
1000
2000
3000
4000
5000
10min Interval

```
[ ]:
```