ed20b045-assgt2

September 18, 2023

1 Q2

```
[1]: import numpy as np
     import pandas as pd
     from scipy.optimize import minimize
     import warnings
```

```
[2]: warnings.filterwarnings('ignore')
```

```
[3]: data = pd.read_csv('Weibull.csv')
```

```
[4]: k = 2
     n = len(data)
```

```
[5]: def likelihood_fn_weibull(lamda, data):
         if lamda <= 0:
             return np.inf
         log_likelihood = -np.sum(np.log(2) - 2 * np.log(lamda) + np.log(data) -␣
      ↪(data / lamda) ** 2)
         return log_likelihood
```

```
[6]: for i in range(10):
         bounds = [(1e-6, None)]
         lam_weibull = minimize(likelihood_fn_weibull, i, args=(data['X'],),␣
      ↪method='Powell', bounds=bounds)
         print(lam_weibull.x, i)
```

```
[6.58194043] 0
[6.58154638] 1
[6.58154641] 2
[6.58154637] 3
[6.58154635] 4
[6.58154633] 5
[6.58154639] 6
[6.58154364] 7
[6.58154364] 8
[6.58154364] 9
```

[7]:
```
sqr_sum = 0
for i in data.X:
  sqr_sum += i ** 2
lam_weibull_calc = (sqr_sum / n) ** 0.5
lam_weibull_calc
```

[7]:
```
6.581546404774335
```

[8]:
```
def likelihood_fn_rayleigh(sigma, data):
    if sigma <= 0:
        return -np.inf
    return -np.sum(-2 * np.log(sigma) + np.log(data) - 0.25 * (data / sigma) **␣
 ↪2)
```

[9]:
```
for i in range(0,10):
  bounds = [(1e-6, None)]
  sigma_rayleigh = minimize(likelihood_fn_rayleigh, i, args = (data['X'],),␣
 ↪method = 'Powell', bounds = bounds)
  print(sigma_rayleigh.x, i)
```

```
[3.29074687] 0
[3.2907732] 1
[3.2907732] 2
[3.29077318] 3
[3.29076151] 4
[3.29076151] 5
[3.29076151] 6
[3.29076151] 7
[3.29076152] 8
[3.29076151] 9
```

[10]:
```
sigma_rayleigh_calc = (0.5 * sqr_sum / n) ** 0.5
sigma_rayleigh_calc
```

[10]:
```
4.653856093509875
```

[11]:
```
print("MLE estimate of   (Weibull):", lam_weibull_calc)
print("MLE estimate of   (Rayleigh):", sigma_rayleigh_calc)
```

```
MLE estimate of   (Weibull): 6.581546404774335
MLE estimate of   (Rayleigh): 4.653856093509875
```

2 Q4

```
[12]: import numpy as np
      import pandas as pd
      import matplotlib.pyplot as plt
      from scipy.stats import norm
```

```
[13]: df = pd.read_csv('Gene_expression.csv')
```

```
[14]: # Convert the selected data to a NumPy array
      data = np.array(df)
```

```
[15]: data
```

```
[15]: array([[44.23856953, 46.32756761, 49.33670648, 49.93000048, 52.15501011,
              56.0828971 , 57.75831725, 45.89022541, 60.45472472, 51.59134855],
             [65.90939535, 57.12291845, 51.4686087 , 52.99571123, 56.32035513,
              54.21490678, 38.06422753, 62.53374416, 53.10527894, 55.36490963],
             [57.21915456, 59.88192727, 54.66953869, 51.55290527, 47.54784521,
              38.944785  , 65.59859592, 55.01108521, 53.2758098 , 57.24907549],
             [46.61596766, 61.67398822, 43.84466132, 61.29665911, 48.10703667,
              55.13740149, 43.69748715, 61.93258031, 49.23991059, 51.80810613],
             [46.00223746, 56.66640143, 52.6672275 , 57.55416647, 66.11898762,
              57.57439412, 59.34329535, 55.08442535, 57.49476686, 56.69855535],
             [58.75442126, 61.23391731, 56.66776774, 56.66468411, 46.09828197,
              58.86178489, 56.82222136, 58.65124218, 58.27710937, 49.21204761],
             [50.66888261, 54.41480639, 44.65471414, 40.29891803, 46.3924186 ,
              46.40496237, 55.29217732, 64.87554371, 56.78207354, 45.16831601],
             [55.13867845, 62.83933614, 46.07297446, 57.63336157, 49.27015271,
              50.04652638, 53.08704388, 64.55507937, 54.32877955, 60.78952829],
             [58.617063  , 66.7854659 , 62.61973083, 50.62441622, 62.24463438,
              46.77079199, 34.70277751, 73.11042621, 44.59158466, 52.6843219 ],
             [50.26708051, 56.07661452, 56.43332749, 48.70199002, 51.10933169,
              57.52885785, 64.99683578, 57.59514797, 51.06623574, 47.47890868],
             [62.63462652, 65.54684051, 64.69908482, 63.18154368, 57.59618726,
              54.73458573, 57.52718336, 61.72281047, 60.01101048, 51.49639027],
             [40.35207568, 55.69765264, 65.9033318 , 51.12469452, 58.1475235 ,
              58.54890805, 69.46019212, 59.21895093, 64.55295749, 66.19263129],
             [67.07647116, 52.9360184 , 47.60435784, 60.06025728, 67.89284215,
              54.35689007, 46.53235382, 60.27869398, 39.06831188, 77.62882834],
             [58.91674414, 57.07239705, 52.26127017, 59.41017495, 46.20080179,
              53.93309066, 55.5262696 , 51.48497454, 61.94107472, 42.12893687],
             [49.59251223, 51.06103776, 51.44581643, 56.10070541, 53.03144788,
              52.3445223 , 53.53641742, 52.67057142, 59.17935237, 49.43853141],
             [69.31994513, 50.62515967, 54.04984816, 72.8204957 , 39.10230996,
              52.44371102, 52.33135492, 63.43227387, 50.18292758, 61.53099591],
             [47.09298667, 31.96561531, 49.0540138 , 53.1649458 , 51.11278372,
         60.81440153, 65.49307035, 58.60450815, 57.86825485, 54.75421036],
        [43.2338728 , 61.40694866, 43.21315701, 61.23976479, 49.5217355 ,
         60.86625406, 59.66696591, 54.7010771 , 49.91275445, 68.06670656],
        [37.88100293, 52.16674221, 62.04580657, 49.11497345, 54.24656108,
         47.26257724, 58.35750665, 53.18215533, 54.5300433 , 44.88589082],
        [51.41342574, 59.57984217, 50.01394462, 49.03520128, 55.85040256,
         62.05003872, 54.12227327, 39.72724455, 51.86344913, 58.63693314]])
```

[16]:
```
# Initialise prior parameters using Day 1 data
prior_mean = np.mean(data[:,0])
prior_variance = np.var(data[:,0])
prior_mean
```

[16]:
```
53.0472556695
```

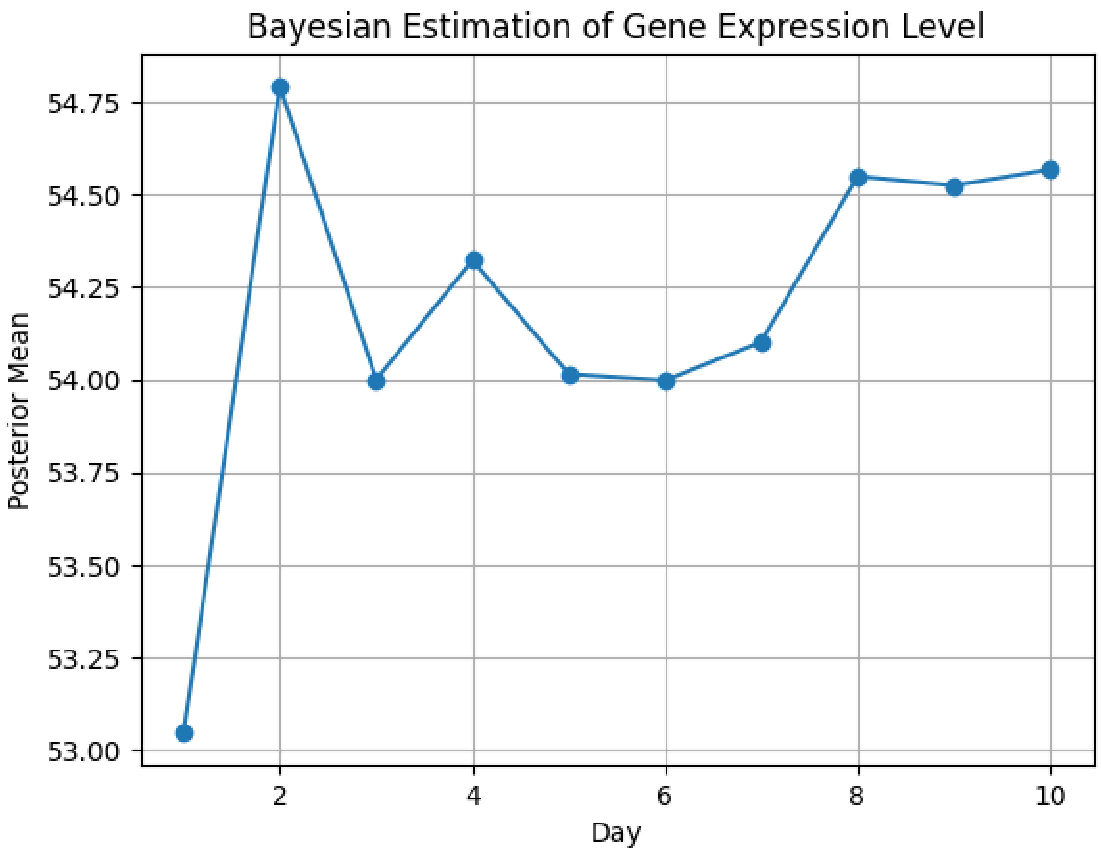
[17]:
```
# Array for posterior parameter for each day till day 10
posterior_mean = [prior_mean]
posterior_variance = [prior_variance]
```

[18]:
```
# Performing recursive Bayesian estimation
for d in range(1,10):
    curr_data = data[:,d]
    updated_mean = (prior_variance * np.mean(curr_data) + prior_mean * np.
 ↪var(curr_data)) / (prior_variance + np.var(curr_data))
    updated_variance = 1 / (1/prior_variance + 1/np.var(curr_data))
    posterior_mean.append(updated_mean)
    posterior_variance.append(updated_variance)
    prior_mean = updated_mean
    prior_variance = updated_variance
```

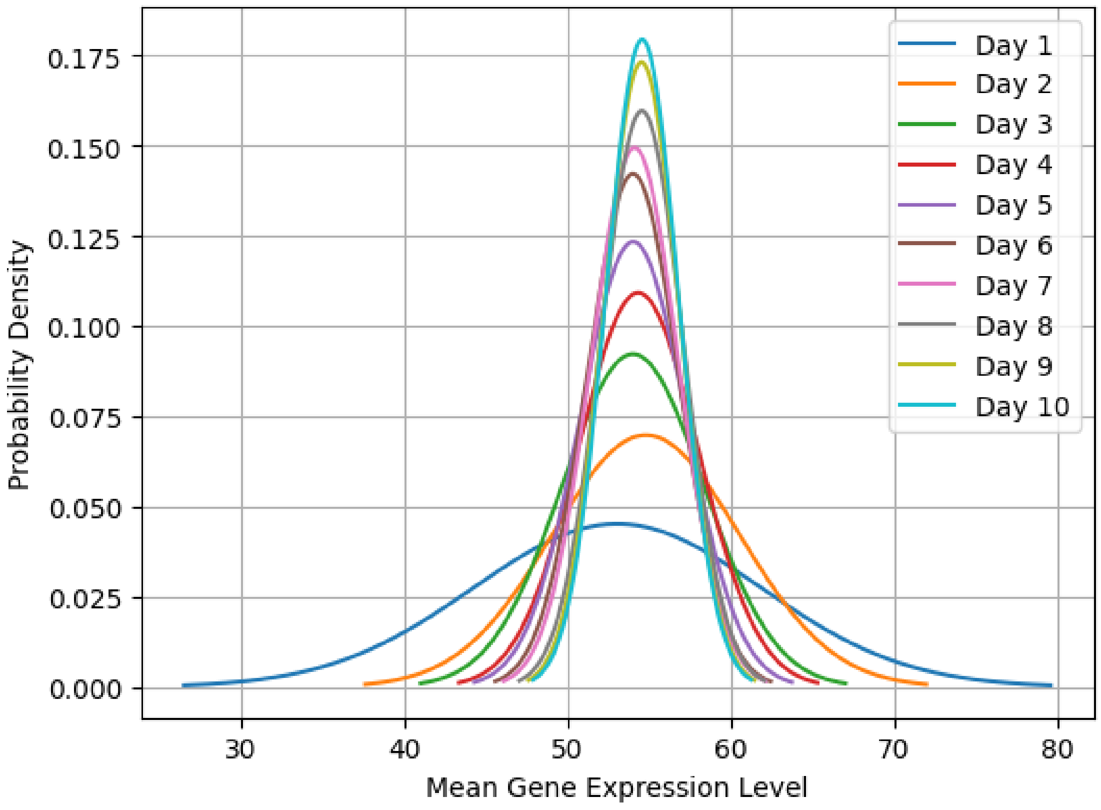
[19]:
```
print(posterior_mean)
```

```
[53.0472556695, 54.79347789504, 54.00033309003618, 54.32306365081989,
54.015625601429065, 53.9984463180054, 54.101884733435234, 54.54988316440594,
54.52540047396053, 54.567902688127454]
```

[20]:
```
plt.plot(range(1, 11), posterior_mean, marker='o')
plt.xlabel('Day')
plt.ylabel('Posterior Mean')
plt.title('Bayesian Estimation of Gene Expression Level')
plt.grid(True)
plt.show()
```



[21]:
```
# Plot the probability distributions for each day
for d in range(10):
    x = np.linspace(posterior_mean[d] - 3 * np.sqrt(posterior_variance[d]),
                    posterior_mean[d] + 3 * np.sqrt(posterior_variance[d]),␣
 ↪1000)
    plt.plot(x, norm.pdf(x, posterior_mean[d], np.sqrt(posterior_variance[d])),
             label=f'Day {d + 1}')
plt.xlabel('Mean Gene Expression Level')
plt.ylabel('Probability Density')
plt.legend()
plt.grid(True)
plt.show()
```



3 Q5

[22]:
```python
import numpy as np
import pandas as pd
from scipy.optimize import minimize
```

[23]:
```python
data = pd.read_csv('Rayleigh.csv')
```

[24]:
```python
def likelihood(sigma, data):
    return np.prod(data / sigma**2 * np.exp(-data**2 / (2 * sigma**2)))
```

[25]:
```python
def log_likelihood(sigma, data):
    return -np.sum(- np.log(data) + np.log(sigma**2) + data**2 / (2 * sigma**2))
```

[26]:
```python
def log_prior(sigma):
    mu = 15
    sigma_prior = 3
    return -0.5 * ((sigma - mu) / sigma_prior)**2 - np.log(sigma_prior * np.
 ↪sqrt(2 * np.pi))
```

[27]:
```
def log_posterior(sigma, data):
    return log_likelihood(sigma, data) + log_prior(sigma)
```

[28]:
```
# Initial guess for sigma
initial_sigma = 1
result = minimize(lambda sigma: -log_posterior(sigma, data), initial_sigma,␣
 ↪method='BFGS')
```

[29]:
```
# The MAP estimate for sigma is in result.x
map_estimate_sigma = result.x[0]

print("MAP estimate for sigma:", map_estimate_sigma)
```

```
MAP estimate for sigma: 6.461205744701753
```