Assignment 2

1) Given, growth rate $f(y;\alpha) = \frac{1}{\alpha^2} y e^{-y/\alpha}$
with $\alpha \in (0,\infty)$ & $y \in [0,\infty)$

- assuming a sample of n independent observations $y_1, y_2 \cdots y_n$
The likelihood is the product of the individual PDFs.

- $L(\alpha) = \prod_{i=1}^{n} \frac{1}{\alpha^2} y e^{-y/\alpha}$

- Defining log-likelihood function $l(\alpha)$

$l(\alpha) = \log(L(\alpha)) = \sum_{i=1}^{n} \log\left[\frac{1}{\alpha^2} y e^{-y/\alpha}\right]$

$l(\alpha) = \sum_{i=1}^{n}\left(-2\log\alpha + \log y - \frac{y}{\alpha}\right)$

$l(\alpha) = -2n\log\alpha + \sum_{i=1}^{n}\log y_i - \frac{\sum_{i=1}^{n} y_i}{\alpha}$

$\frac{d(-l(\alpha))}{\partial\alpha} = 0$, to find max likelihood we will be minimizing the $-l(\alpha)$

$\frac{d}{d\alpha}\left[2n\log\alpha - \sum_{i=1}^{n}\log y_i + \frac{\sum_{i=1}^{n} y_i}{\alpha}\right) = 0$

$\frac{2n}{\alpha} - \frac{\sum_{i=1}^{n} y_i}{\alpha^2} = 0$ $<$ $1/\alpha = 0$, $2n - \frac{\sum_{i=1}^{n} y_i}{\alpha} = 0$

So, maximum likelihood estimate for the parameter is,

$\left[\alpha_{MLE} = \frac{\sum_{i=1}^{n} y_i}{2n} = \frac{\bar{y}}{2}\right]$

Question 2 (MM20B07)

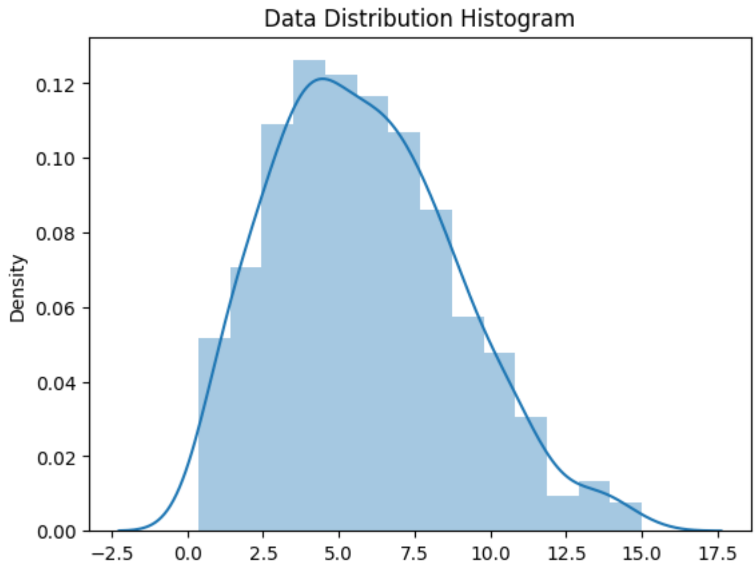
Importing required packages

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.optimize import minimize
```

```
df = pd.read_csv('/content/drive/MyDrive/sem 7/ID5055/Assignment 2/q2 Weibull.csv')
```

```
plt.title('Data Distribution Histogram')
sns.distplot(df)
```

```
<ipython-input-3-3fac749752ce>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df)
<Axes: title={'center': 'Data Distribution Histogram'}, ylabel='Density'>
```



Parameter estimation using MLE

The Weibull distribution is given by:

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}, & \text{if } x \geq 0, \\ 0, & \text{if } x < 0. \end{cases}$$

```
def log_likelihood_weibull(parameters, data):
    lambda_ = parameters[0]
    k = 2.0
    if lambda_ <= 0:
        return np.inf
    log_likelihood = np.sum(np.log((k / lambda_) * (data / lambda_)**(k-1) * np.exp(-(data / lambda_)**k)))
    return -log_likelihood

initail_parameters = [1.0]

result = minimize(log_likelihood_weibull, initail_parameters, args = (df,))

# Estimated parameter
estimated_lamda = result.x[0]
print("Estimated lambda:", estimated_lamda)
```

```
Estimated lambda: 6.581546500160183
```

So the estimated value of λ = 6.581546500160183.

To estimate the parameter σ of the Rayleigh distribution, which is given by:

$$f(x;\sigma) = \frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}, \quad x \geq 0$$

we can use the invariance property of MLE.

According to the Invariance property, if we have an estimate of parameter or set of parameters of a statistical distribution, then we can use this estimate to calculate MLE estimates for related paramters. In the context of the Rayleigh and Weibull distributions, there is a connection between the two distributions based on their parameters. For k = 2, Weibull distribution turns into Rayleigh distribution with **λ =** $\sqrt{2}\boldsymbol{\sigma}$ or we can say that Rayleigh distribution is a special case of Weibull distribution.

Now, since we have estimate for λ, which is 6.581546500160183, we can use the above relation to get σ = (6.581546500160183) / $\sqrt{2}$ = 4.653856039522699.

```
# Maximum Likelihood Estimation for σ in the Rayleigh distribution
def rayleigh_log_likelihood(sigma, data):
    if sigma <= 0:
        return np.inf
    log_likelihood = np.sum(np.log((data / sigma**2) * np.exp(-data**2 / (2 * sigma**2))))
    return -log_likelihood

result_rayleigh = minimize(rayleigh_log_likelihood, [1.0], args=(df,))
sigma_mle = result_rayleigh.x[0]

print(f"MLE estimate for σ in Rayleigh distribution: {sigma_mle}")
```

```
MLE estimate for σ in Rayleigh distribution: 4.653856039522699
```

The σ value after Maximum Likelihood estimation of Weibull distribution matches with the calculated value.

37) given $f(y;\theta) = \frac{3y^2}{\theta^3}$, with $\theta \in (0,\infty)$ and $y \in [0,\theta]$.

Let, we have a dataset of n points $y_1, y_2, y_3 \cdots y_n$.

- The likelihood function for this is $L(\theta)$.

$$L(\theta) = \prod_{i=1}^{n} \frac{3y_i^2}{\theta^3} = \left(\frac{3^n}{\theta^{3n}}\right) \prod_{i=1}^{n} y_i^2$$

now to find maximum likelihood estimate for θ we need to find out θ for which $L(\theta)$ maximises

Also it is given $y \in [0,\theta]$

So, we need small θ such that above condition is satisfied, which is only possible for

$$\left[\theta = \max\{y_i\} \quad i = 1,2,3\cdots n\right]$$

Question 4 (MM20B007)

Importing required packages

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

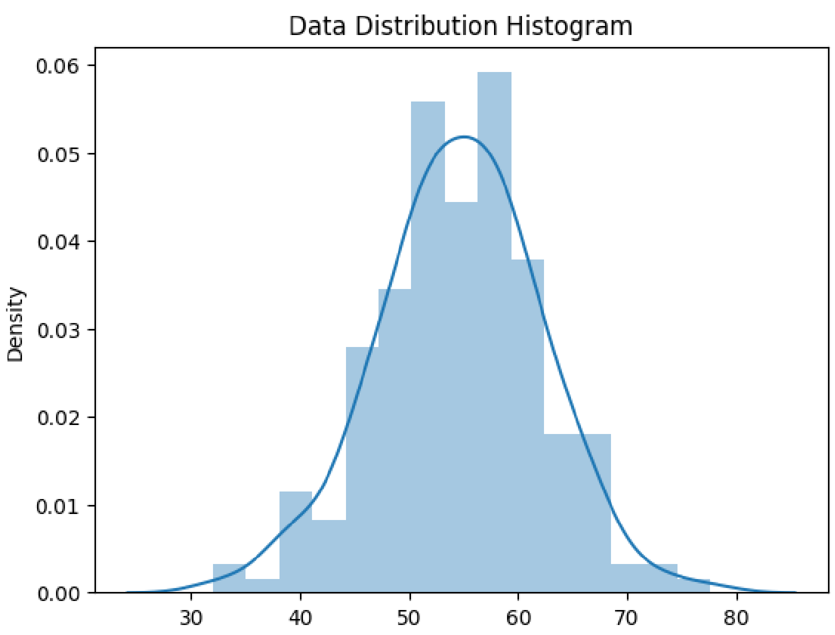
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.optimize import minimize
from scipy.stats import norm
```

```
df = pd.read_csv('/content/drive/MyDrive/sem 7/ID5055/Assignment 2/q4 Gene_expression.csv')
df
```

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Da
0	44.238570	46.327568	49.336706	49.930000	52.155010	56.082897	57.758317	45.890225	60.454725	51.5
1	65.909395	57.122918	51.468609	52.995711	56.320355	54.214907	38.064228	62.533744	53.105279	55.3
2	57.219155	59.881927	54.669539	51.552905	47.547845	38.944785	65.598596	55.011085	53.275810	57.2
3	46.615968	61.673988	43.844661	61.296659	48.107037	55.137401	43.697487	61.932580	49.239911	51.8
4	46.002237	56.666401	52.667228	57.554166	66.118988	57.574394	59.343295	55.084425	57.494767	56.6
5	58.754421	61.233917	56.667768	56.664684	46.098282	58.861785	56.822221	58.651242	58.277109	49.2
6	50.668883	54.414806	44.654714	40.298918	46.392419	46.404962	55.292177	64.875544	56.782074	45.1
7	55.138678	62.839336	46.072974	57.633362	49.270153	50.046526	53.087044	64.555079	54.328780	60.7
8	58.617063	66.785466	62.619731	50.624416	62.244634	46.770792	34.702778	73.110426	44.591585	52.6
9	50.267081	56.076615	56.433327	48.701990	51.109332	57.528858	64.996836	57.595148	51.066236	47.4
10	62.634627	65.546841	64.699085	63.181544	57.596187	54.734586	57.527183	61.722810	60.011010	51.4
11	40.352076	55.697653	65.903332	51.124695	58.147523	58.548908	69.460192	59.218951	64.552957	66.1
12	67.076471	52.936018	47.604358	60.060257	67.892842	54.356890	46.532354	60.278694	39.068312	77.6
13	58.916744	57.072397	52.261270	59.410175	46.200802	53.933091	55.526270	51.484975	61.941075	42.1
14	49.592512	51.061038	51.445816	56.100705	53.031448	52.344522	53.536417	52.670571	59.179352	49.4
15	69.319945	50.625160	54.049848	72.820496	39.102310	52.443711	52.331355	63.432274	50.182928	61.5
16	47.092987	31.965615	49.054014	53.164946	51.112784	60.814402	65.493070	58.604508	57.868255	54.7
17	43.233873	61.406949	43.213157	61.239765	49.521735	60.866254	59.666966	54.701077	49.912754	68.0
18	37.881003	52.166742	62.045807	49.114973	54.246561	47.262577	58.357507	53.182155	54.530043	44.8
19	51.413426	59.579842	50.013945	49.035201	55.850403	62.050039	54.122273	39.727245	51.863449	58.6

```
plt.title('Data Distribution Histogram')
sns.distplot(df)
```

```
<ipython-input-4-3fac749752ce>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df)
<Axes: title={'center': 'Data Distribution Histogram'}, ylabel='Density'>
```



Assume the prior distribution of μ to be a normal distribution. You can take the sample mean of Day 1 samples and variance as prior parameters.

Given:

1. mean of the distribution μ_0 = mean of the Day 1 samples

2. variance of the distribution σ_0^2 = variance of the Day 1 samples

```
day1 = df['Day 1']
mu_0 = np.mean(day1, axis = 0)
sigma_0 = np.std(day1, axis = 0)

print(f"mean of the prior distribution is {mu_0}")
print(f"stanadrd deviation of the prior distribution is {sigma_0}")
```

```
mean of the prior distribution is 53.0472556695
stanadrd deviation of the prior distribution is 8.826980817204479
```

Estimate the posterior distribution of μ using samples from Day 1

Gene expression levels distribution is given by:

$$f(x; \mu, \sigma = 8) = \frac{1}{8\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{128}\right)$$

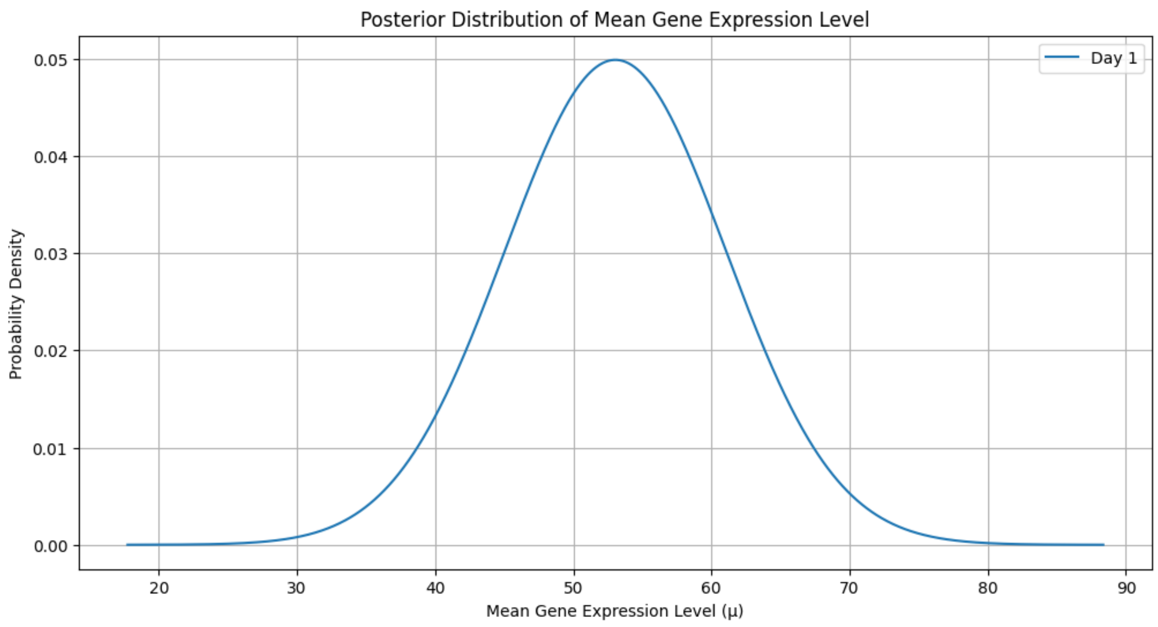
The prior distribution is given by:

$$f(\mu; \mu_0, \sigma_0) = \frac{1}{\sigma_0\sqrt{2\pi}} \exp\left(-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}\right)$$

where μ_0 = 6.0337401566764175 and standard deviation σ_0 = 3.5503391410788043.

```
def map_log_likelihood(parameters, data, prior_mu, prior_std):
    mu = parameters
    if mu < 0:
        return np.inf
    log_likelihood = np.sum(np.log((1/(8*(2**0.5)*(np.pi)))*np.exp(-((data - mu)**2)/128))) + np.sum(np.log((1/(prior_std*(2**0.5)*(np.pi)))*np.exp(-((mu - prior_mu)**2)/(2*(prior_s
    return -log_likelihood

initail_parameters = [1.0]

result = minimize(map_log_likelihood, initail_parameters, args = (df['Day 1'], mu_0, sigma_0))

# Estimated parameter
estimated_mu = result.x[0]
print("Estimated mu:", estimated_mu)

# Plot the probability distribution of the mean gene expression level each time after the update
plt.figure(figsize=(12, 6))
x = np.linspace(estimated_mu - 4*sigma_0, estimated_mu + 4*sigma_0, 1000)
y = norm.pdf(x, loc=estimated_mu, scale=8)
plt.plot(x, y, label=f'Day 1')

plt.xlabel('Mean Gene Expression Level (μ)')
plt.ylabel('Probability Density')
plt.title('Posterior Distribution of Mean Gene Expression Level')
plt.legend()
plt.grid()
plt.show()
```

```
Estimated mu: 53.04723720260668
```

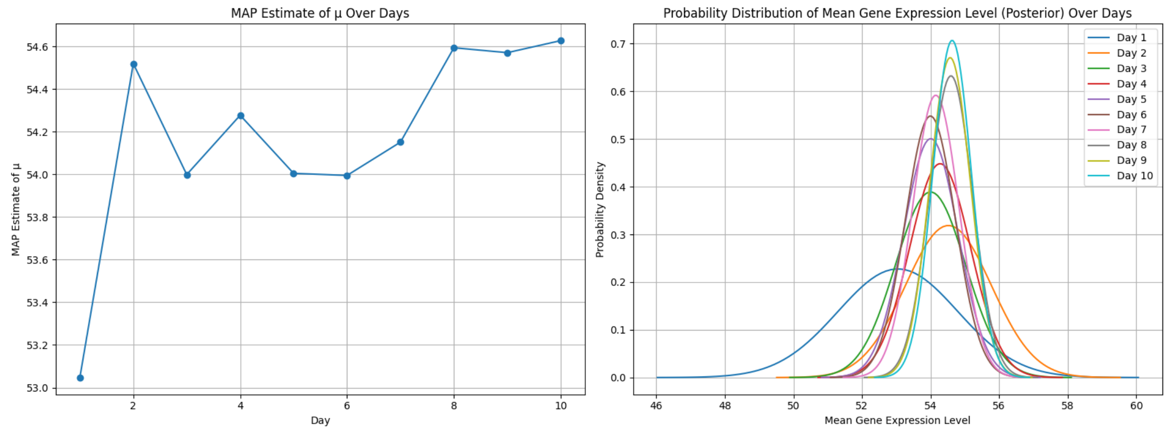


Update the priors and repeat the process using data from each of the days. Plot the probability distribution of the mean of gene expression level each time after the update

Using posterior of previous day as prior of the next next day.

```
# Prior parameters (initialized using Day 1 data)
prior_mean = np.mean(df['Day 1'])
prior_variance = np.var(df['Day 1'])

# Lists to store MAP estimates and posterior parameters
map_estimates = []
posterior_means = [prior_mean]
posterior_variances = [prior_variance]
```

```
# Perform Bayesian estimation for each day
for i in range(10):
    # Update prior parameters based on the previous posterior
    prior_mean = posterior_means[-1]
    prior_variance = posterior_variances[-1]

    # Likelihood parameters (known standard deviation)
    likelihood_stddev = 8

    # Compute posterior parameters using Bayesian update
    posterior_variance = 1 / (1 / prior_variance + len(df['Day ' + str(i+1)]) / (likelihood_stddev**2))
    posterior_mean = (prior_mean / prior_variance + np.sum(df['Day ' + str(i+1)]) / (likelihood_stddev**2))/(len(df['Day ' + str(i+1)])/likelihood_stddev**2 + 1/prior_variance)

    # Calculate the MAP estimate for μ
    map_estimate = posterior_mean

    # Store posterior parameters and MAP estimate
    posterior_means.append(posterior_mean)
    posterior_variances.append(posterior_variance)
    map_estimates.append(map_estimate)


fig, ax = plt.subplots(1, 2, figsize=(16, 6))

days = np.arange(1, 11)
ax[0].plot(days, map_estimates, marker='o', linestyle='-')
ax[0].set_xlabel("Day")
ax[0].set_ylabel("MAP Estimate of μ")
ax[0].set_title("MAP Estimate of μ Over Days")
ax[0].grid(True)

days = np.arange(1, 11)
for i in range(1, 11):
    posterior_mean = posterior_means[i]
    posterior_variance = posterior_variances[i]
    posterior_stddev = np.sqrt(posterior_variance)

    x = np.linspace(posterior_mean - 4 * posterior_stddev, posterior_mean + 4 * posterior_stddev, 100)
    ax[1].plot(x, norm.pdf(x, posterior_mean, posterior_stddev), label=f'Day {i}')

ax[1].set_xlabel("Mean Gene Expression Level")
ax[1].set_ylabel("Probability Density")
ax[1].set_title("Probability Distribution of Mean Gene Expression Level (Posterior) Over Days")
ax[1].legend()
ax[1].grid(True)

plt.tight_layout()
plt.show()
```



Question 5 (MM20B007)

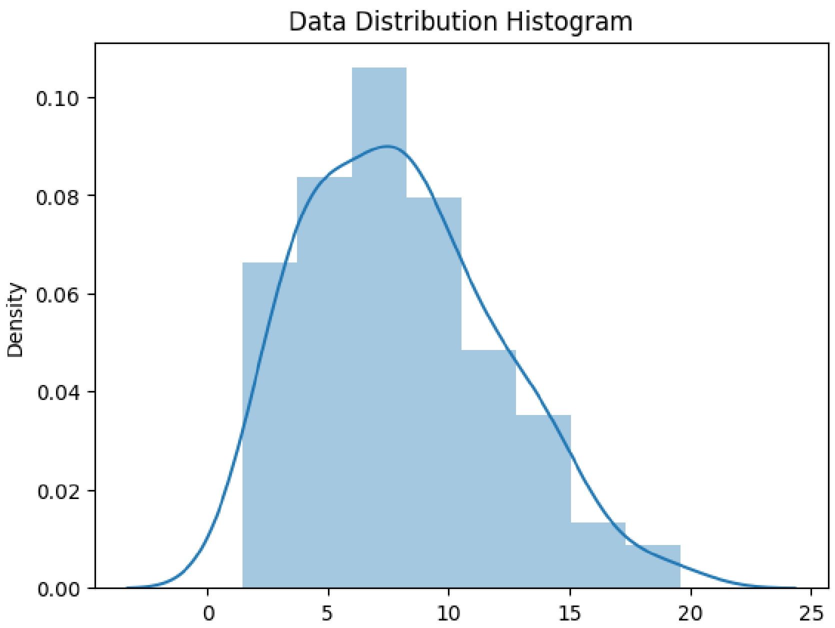
Importing required packages

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.optimize import minimize
```

```
df = pd.read_csv('/content/drive/MyDrive/sem 7/ID5055/Assignment 2/q5 Rayleigh.csv')
```

```
plt.title('Data Distribution Histogram')
sns.distplot(df)
```

```
<ipython-input-3-3fac749752ce>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df)
<Axes: title={'center': 'Data Distribution Histogram'}, ylabel='Density'>
```



Parameter estimation using Maximum A Posteriori

To estimate the parameter σ of the Rayleigh distribution, which is given by:

$$f(x;\sigma) = \frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}, \quad x \geq 0$$

we are provided with a normal distributed prior of σ, which is $P(\sigma) \sim \mathcal{N}(\mu = 15, \sigma = 3)$.

```
# Maximum a Posteriori Estimation for σ in the Rayleigh distribution
def rayleigh_map_likelihood(sigma, data):
    if sigma <= 0:
        return np.inf
    log_likelihood = np.sum(np.log((data / sigma**2) * np.exp(-data**2 / (2 * sigma**2)))) + np.log((np.exp(-((sigma - 15)**2)/18))/(3*(2**0.5)*np.pi))
    return -log_likelihood

result_rayleigh = minimize(rayleigh_map_likelihood, [1.0], args=(df,))
sigma_mle = result_rayleigh.x[0]

print(f"MAP estimate for σ in Rayleigh distribution: {sigma_mle}")
```

```
MAP estimate for σ in Rayleigh distribution: 6.461205508911481
```