

# Exponential model

We consider the following univariate Exponential model, parametrized by  $\lambda$ , a positive real number

$$\left\{ \mathcal{E}(\lambda); \lambda > 0 \right\}$$

The PDF of the Exponential distribution is given by:

$$f_{\lambda}(x) = \lambda e^{-\lambda x} \mathbb{1}_{\mathbb{R}_+^*}(x)$$

To obtain a **Maximum Likelihood Estimator** (MLE), denoted  $\hat{\lambda}$  for  $\lambda$  from a sample of  $n$  *i.i.d.* realizations of an Exponential r.v., we first write the likelihood function (joint PDF of the sample), take the logarithm of this function, differentiate it w.r.t. the parameter and then solve for the parameter.

# Maximum likelihood estimation

Likelihood and log-likelihood functions for an Exponential model

$$\mathcal{L}(\lambda \mid \mathbf{x}) = \prod_{i=1}^n f_{\lambda}(x_i) = \prod_{i=1}^n \lambda e^{-\lambda x_i} = \lambda^n e^{-(\lambda \sum_{i=1}^n x_i)}$$

$$l(\lambda \mid \mathbf{x}) = \ln(\mathcal{L}(\lambda \mid \mathbf{x})) = n \ln(\lambda) - \lambda \sum_{i=1}^n x_i$$

$$\frac{\partial \ln \mathcal{L}(\lambda \mid \mathbf{x})}{\partial \lambda} = n \frac{1}{\lambda} - \sum_{i=1}^n x_i$$

Setting the derivative equal to 0 and solving for  $\lambda$  then yields

$$\frac{n}{\lambda} - \sum_{i=1}^n x_i = 0 \quad \Leftrightarrow \quad \frac{n}{\lambda} = \sum_{i=1}^n x_i \quad \Leftrightarrow \quad \hat{\lambda}_{MLE} = \frac{n}{\sum_{i=1}^n x_i} = \frac{1}{\bar{x}}$$

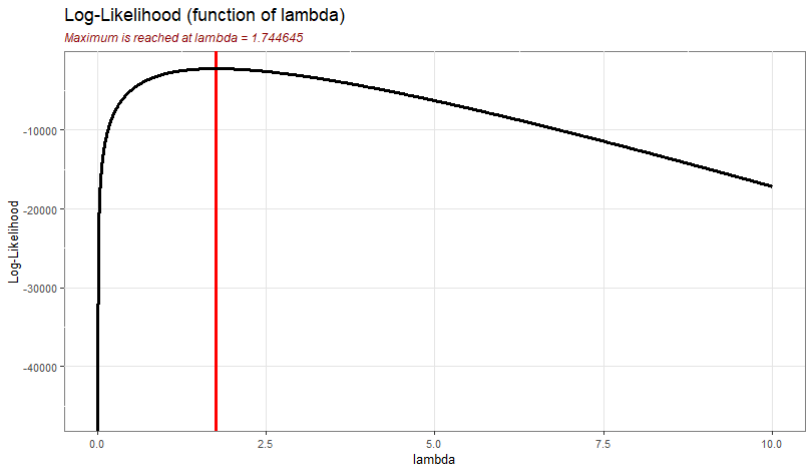
# Exponential MLE in R: closed-form formula

```
1 # generate a random sample of n = 5000 from an
2 # Exponential distribution
3 # with parameter lambda = 1.75 (argument name = 'rate')
4
5 set.seed(1986)
6 n = 5000
7 xi <- rexp(n = n, rate = 1.75) # rate assumed unknown
8 head(xi)
9 # [1] 0.4817649 0.5847904 0.1148642 0.1820848 0.9921011
   0.2747533
10
11 # Closed-form MLE
12 lambda_hat_formula = n / sum(xi)    # or 1 / mean(xi)
13 lambda_hat_formula
14 # [1] 1.744645
15
16 # the approximation is good with such a large sample.
```

# Exponential MLE in R: numerical approximation

```
1 # Numerical approximation of the MLE
2 mle = optimize(function(lambda){
3     sum(dexp(x = xi, rate = lambda, log = TRUE))
4 },
5     interval = c(0, 10),
6     maximum = TRUE,
7     tol = .Machine$double.eps^0.5)
8
9 lambda_hat = mle$maximum
10 lambda_hat
11 # [1] 1.744645
12
13 # the result is exactly the same as when using
14 # the closed-form formula
```

# Plot in R: log-likelihood function



Artificial dataset of size 5000

# Exponential MLE in Python: closed-form formula

```
1 from scipy import stats
2 import numpy as np
3
4 # generate a sample of size 5000
5 # in numpy, the parametrization is different from R
6 np.random.seed(1986)
7 n = 5000
8 Lambda = 1.75 # true parameter value, that we will estimate
9 xi = np.random.exponential(scale = 1/Lambda, size = n)
10 print(xi)
11 # [0.48077598 0.04599578 0.53583846 ... 0.31828503
12    0.05858069 0.0721613 ]
13 # Closed-form MLE
14 lambda_hat_formula = n/sum(xi) # or 1/statistics.mean(xi)
15 lambda_hat_formula
16 # [1] 1.71848223876711
```

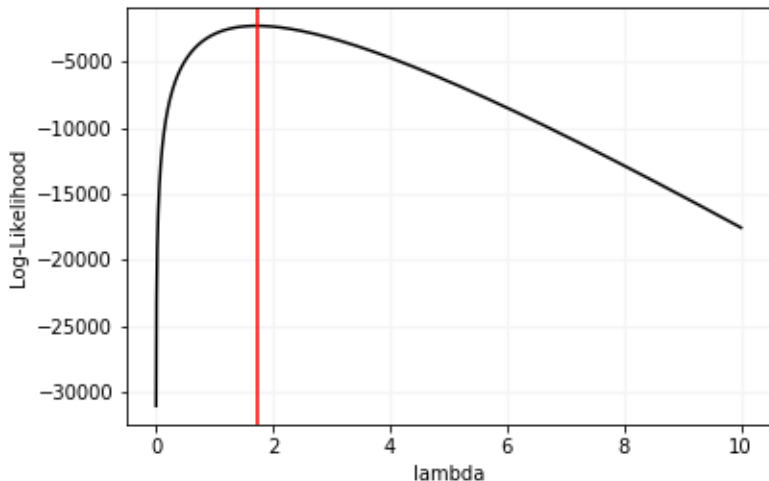
# Exponential MLE in Python: numerical approximation

```
1 def llikelihood(Lambda):
2     # log-likelihood function
3     ll = -np.sum(stats.expon.logpdf(xi, scale = 1/Lambda))
4     return ll
5
6 from scipy.optimize import minimize
7
8 # Numerical approximation of the MLE using minimize()
9 mle = minimize(llikelihood,
10                x0 = 4,
11                method = 'BFGS')
12 print(mle.x)
13 # [1.71848223]
```

# Plot in Python: log-likelihood function

Log-Likelihood (function of lambda)

Maximum is reached at  $\lambda = 1.71848223$



artificial dataset of size 5000



# Further reading and code

The R Project for Statistical Computing:

<https://www.r-project.org/>

Python:

<https://www.python.org/>

Accessing R and Python code:

<https://github.com/JRigh/Simple-maximum-likelihood-estimation/>