Maximum Likelihood Estimation

08/27/2025

The maximum likelihood estimator (MLE) is a statistical method for parameter estimation. The other widely used method is the so-called method of moments (MoM) where the mean and standard deviation of a distribution are estimated using the first and second moments. MLE was first proposed by R. A. Fisher in 1912.

Let $f(X = x|\mu, \sigma)$ be the probability distribution of a random variable, X, for observations, x_i, with the mean μ and standard deviation σ. For a set of n independent observations, $x_1, x_2, \ldots, x_n$, the joint probability distribution is

$$f(x_1, x_2, \ldots, x_n|\mu, \sigma) = \prod_{i=1}^{n} f(x_i|\mu, \sigma)$$

The interpretation of above expression is the probability of the random variable, X, taking on various values x_i for a given set of parameters μ, σ. The likelihood function, $\mathcal{L}()$, although looks very much like the joint probability function, has a very different interpretation. It is the likelihood of the parameters μ, σ, taking on different possible values for given a set of observations, $x_1, x_2, \ldots, x_n$.

$$\mathcal{L}(\mu, \sigma|x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} f(x_i|\mu, \sigma)$$

The estimated parameter values are those when the likelihood function, $\mathcal{L}()$, is at its maximum. It is extremely advantageous to work with the logarithmic form of $\mathcal{L}()$ to avoid the dominance of the function by a few $f(x_i|\mu, \sigma)$ of large values. Many optimization procedures would minimize a function value rather than maximize it. Therefore, it is also often to see the likelihood function written in the negative logarithmic form, *i.e.*, to minimize a function is also to maximize its negation.

Maximization Procedures in R

There are several function maximization procedures in R. Most of them are based on the optim() function which does minimization of a target function not maximization. Therefore, the target function is formulated in its negative value before being submitted to optim().

optim() in the stats Library

One can use optim() directly for MLE. Its input requires the target function to be minimized and a list of initial guess of the parameters -- optim(fn, par, method, hessian). An example is shown in Figure 1.

One hundred normally distributed data points are generated using the rnorm() function with the mean of 1 and standard deviation of 2. Please note that the data, x_i, are in a vector form which is a global variable therefore passing the values to another function is not required.

NegLogLik1 is a function to be called by optim() as target function (fn). Only a parameter vector, par, is required for this function. Inside the function, a negative sum is used and the logarithmic form of dnorm() is specified (`log=TRUE`). The big pi product in the likelihood function, $\mathcal{L}()$, is therefore replaced by the summation of the logarithmic function.

```
## normal distribution
x=rnorm(100,mean=1,sd=2)

### use optim
NegLogLik1=function(par){-
sum(dnorm(x=x,mean=par[1],sd=par[2],log=TRUE))}

mle1=optim(fn=NegLogLik1,par=c(mean(x),sd(x)),method="Nelder-
Mead",hessian=TRUE)

cov=solve(mle1$hessian)

data.frame(param=c("mu","sigma"),estimate=mle1$par,SE=sqrt(diag(cov
)))

  param estimate        SE
1    mu 1.045098 0.2009087
2 sigma 2.009087 0.1420719
```

Figure 1. Use of optim() for MLE

Object mle1 holds the output from the optim() function. The NegLogLik1 function is passed to optim() along with the initial estimated parameters using the MoM. Nelder-Mead is the default optimization method used by optim(); calling it explicitly here is for clarity. Hessian matrix is later used to compute the standard errors of the estimates from the covariance matrix which in turn is the inverse of the Hessian matrix. To compute the inverse, one needs to “solve for” it. The standard errors are the square roots of the diagonals of the covariance matrix.

mle() in the stats4 Library

mle() from stats4 library hides the complexities in computing the standard errors. In addition, it offers confidence interval and AIC computations for the estimates. However, the NegLogLik1 function can no longer take a parameter vector as the input but each parameter must be spelled out separately. Inside the function, the negative sum is nonetheless similar to the one used by optim(). When calling the mle() function, it is clear that the input parameters are the initial guesses, not the variables (`start=c(mu=mean(x),sigma=sd(x))`). Again, MoM estimates are conveniently used as the guesses.

```
### use mle from stats4
NegLogLik1=function(mu,sigma){-
sum(dnorm(x,mean=mu,sd=sigma,log=TRUE))}

mle2=mle(minuslogl=NegLogLik1, start=c(mu=mean(x),sigma=sd(x)),
method="Nelder-Mead")

summary(mle2)

confint(mle2)

AIC(mle2)
```

Figure 2. Use of mle() from the stats4 library.

The real power of mle() function is that the standard errors of the parameter estimates are part of the standard output (see Figure 2). Further, it allows to compute the 95% confidence intervals and AIC, which is computed using:

$$AIC = -2 \times log(\mathcal{L}) + 2 \times k$$

where k is the number of parameters. Therefore AIC = 423.316+2 $\times$ 2 = 427.316 (see Figure 3). BIC, on the other hand, is defined as:

$$BIC = -2 \times log(\mathcal{L}) + log(n) \times k$$

where n is the number of samples (n=100). As such, BIC = 423.3161 + 2.303 $\times$ 2 = 427.921.

```
Coefficients:
      Estimate Std. Error
mu    1.045098  0.2009087
sigma 2.009087  0.1420719

-2 log L: 423.3161

> confint(mle2)
         2.5 %   97.5 %
mu    0.647727 1.442889
sigma 1.759760 2.323237

> AIC(mle2)
[1] 427.3161
```

Figure 3. Output of mle() from the stats4 library.

<ins>maxLik() in the maxLik Library</ins>

maxLik() further simplifies the MLE procedure by providing the p-values in the standard output (see Figure 5). It also allows computation of AIC and BIC for the model estimation.

```
### use maxLik
library(maxLik)

LogLik1=function(par)
{sum(dnorm(x=x,mean=par[1],sd=par[2],log=TRUE))}

mle3=maxLik(logLik=LogLik1, start=c(mu=mean(x),sigma=sd(x)),
method="Nelder-Mead")

summary(mle3)
```

Figure 4. Use of maxLik() from the maxLik library.

It is important to note that the LogLik1 function is similar to that used by optim() but the negative sign is no longer required (see Figure 4). As such, the function is named LogLik1() not NegLogLik1(). The use of maxLik() function is similar to that of mle() function with the initial estimates clearly spelled out.

```
Maximum Likelihood estimation
Nelder-Mead maximization, 39 iterations
Return code 0: successful convergence
Log-Likelihood: -211.6581
2  free parameters
Estimates:
      Estimate Std. error t value  Pr(> t)
mu      1.0451     0.2009   5.203 1.96e-07 ***
sigma   2.0091     0.1421  14.141  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’
```

Figure 5. Output of maxLik() from the maxLik library.

A Complex Example

MLE can also be used for non-linear parameter estimation. In these cases, the parameters of the distribution are no longer all constants, but a function or even solution to an equation. In turn, the function contains various parameters that are to be estimated. The example below shows a binomial distribution with one of its parameters (p) being a logistic regression of a non-linear function.

The resistance of gypsy moth to the nucleopolyhedrovirus (NPV) is affected by the crowding that the insects experienced as larvae. The script in Figure 6 shows how the rearing density (D) affects the survival (S) of the moth (population = N) after exposure to a given dose of the virus.

The survival probability (p) is a logistic regression function of the rearing density, D, and constants a, b, and delta that are to be estimated (see Figure 7).

$$p = \frac{e^u}{1+e^u}$$

where $u = a + b \times D^\delta$. The binomial distribution is used for the survival probability.

```
D=c(1,5,10,15,20); N=c(90,90,89,87,93); S=c(60,60,56,41,31)

MothSurvNLL=function(a,b,delta){u=a+b*D^delta; p=exp(u)/(1+exp(u))
  -sum(dbinom(x=S,size=N,prob=p,log=TRUE))}

MothSurv=mle(minuslogl=MothSurvNLL,start=c(a=0.7,b=-0.001,delta=2),
method="Nelder-Mead")

u=MothSurv@coef[1]+MothSurv@coef[2]*D^MothSurv@coef[3]
p=exp(u)/(1+exp(u))
plot(D,p,type="b"); grid(lty=2)
```

Figure 6. Use of mle() for nonlinear parameter estimation. Note mle() gives a S4 object and access to the names follows "@" rather than "$" convention.

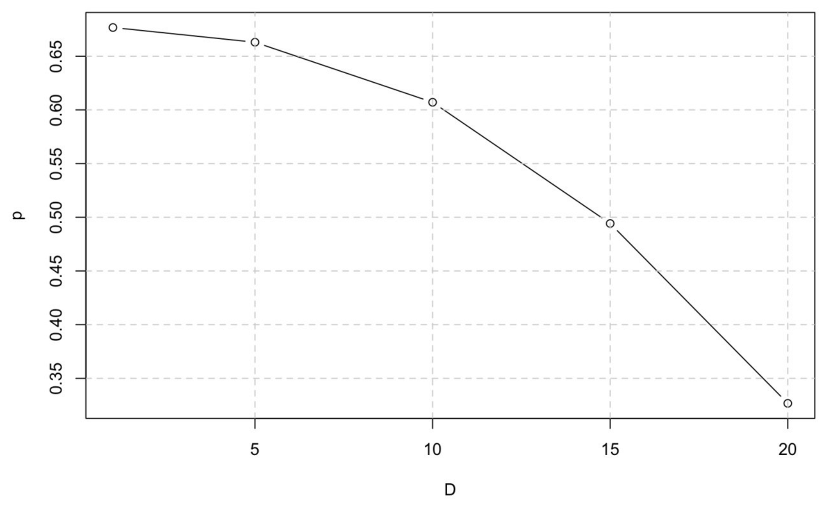

Figure 7. Gypsy moth survival probability (p) in relation to the larvae rearing density (D).