More about likelihood

Contents

1 Numerical maximization of the likelihood

In several interesting cases, the maximization problem has no analytical solution. In other words, it is not possible to write $\hat{\theta}$ explicitly as a function of the data. In these cases, it is necessary to resort to numerical algorithms for the maximization of the likelihood.

1.1 The Newton-Raphson (NR) method

One of the most used method for optimization in statistics is the Newton-Raphson method. It is based on approximating the log-likelihood $\ell_n(\theta)$ by a quadratic function. For a given ***starting point*** $\theta_0 \in \Theta$, such that $\ell_n''(\theta_0) < 0$, define

$$\tilde{\ell}_n(\theta) = \ell_n(\theta_0) + (\theta - \theta_0)\ell_n'(\theta_0) + (\theta - \theta_0)^2 \ell_n''(\theta_0)/2.$$

This is the second order Taylor serie approximation of $\ell_n(\theta)$ around θ_0.

The solution of the first-order condition for maximizing $\tilde{\ell}_n(\theta)$ is

$$\theta_1 = \theta_0 - \frac{\ell_n'(\theta_0)}{\ell_n''(\theta_0)}.$$

Since $\tilde{\ell}_n$ is an approximation of ℓ_n, θ_1, defined above, provides a guess value for the MLE.

We can try to improve the approximation by taking θ_1 as the new starting point and *keep repeating the process until convergence*. This suggests the following iterative procedure:

$$\theta_{k+1} = \theta_k - \frac{\ell_n'(\theta_k)}{\ell_n''(\theta_k)}, \ k = 0, 1, \ldots$$

Equivalently, this algorithm can also be written as

$$\theta_{k+1} = \theta_k + \frac{S_n(\theta_k)}{J_n(\theta_k)}, \ k = 0, 1, \ldots,$$

where $S_n(\theta) = \ell_n'(\theta)$ and $J_n(\theta) = -\ell_n''(\theta)$ are the score and the observed FI.

As we said, the procedure should be run until convergence, i.e. until there is no "significant" difference between θ_k and θ_{k+1}. No "significant" difference means that changes between consecutive iterations are less than a user-defined tolerance. For example, we may stop the algorithm whenever the difference $|\theta_{k+1} - \theta_k|$, or the relative difference $|\theta_{k+1} - \theta_k|/|\theta_k|$, is smaller than 10^{-8}.

Note that, $\theta_{k+1} = \theta_k$ is equivalent to $\ell_n'(\theta_k) = 0$. So, at the end of the process (i.e. when the iterations stop), the algorithm converges to (a neighborhood of) a stationary point of ℓ_n. This point could be a maximum point, a minimum point or even a inflection point. However, if $\ell_n''(\theta_k) < 0$, then the convergence point is a **local maximizer**. Moreover, if ℓ_n is strictly concave, then the convergence point is the (unique) global maximizer.

In all cases, when it converges, the NR algorithm reaches the stationary point closest to its starting point θ_0. *If several stationary points are present, the choice of the starting point becomes critical*. In many situations, the MoM can be used to obtain a reasonable starting point.

In the case where the likelihood is not strictly concave, it is recommended to:

- If possible, start by visually checking the graph of the (log-)likelihood function. Sometimes it's easy to see where an extremum occurs. But sometimes, local fluctuations of a relatively small scale can hide such points.
- Rerun the algorithm from different starting points, and then choose, among all the convergence points, the one that globally maximizes the likelihood.
- Perturb the convergence point by a "small" amount, and use this as a starting point for a new run of the algorithm. Then, see if it converges to a "better point", or "always" to the same one.

The arguments for deriving the NR algorithm for optimization in one dimension can be directly extended to multi-dimensional problems giving the multi-parameter NR method:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + J_n^{-1}(\boldsymbol{\theta}_k) S_n(\boldsymbol{\theta}_k), \; k = 0, 1, \ldots$$

where $S_n(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \ell_n(\boldsymbol{\theta}) = \sum_{i=1}^n \nabla_{\boldsymbol{\theta}} \log f(X_i, \boldsymbol{\theta})$ is the Score vector and $J_n(\boldsymbol{\theta}) = -\nabla^2_{\boldsymbol{\theta}} \ell_n(\boldsymbol{\theta}) = -\sum_{i=1}^n \nabla^2_{\boldsymbol{\theta}} \log f(X_i, \boldsymbol{\theta})$ is the observed FI matrix (i.e. Hessian of negative log-likelihood).

Let's consider our previous example with $f(x;\theta) = \frac{1+\theta x}{2} I(-1 \leq x \leq 1)$; $-1 < \theta < 1$, and the observed data

```
x <- c(0.9852, 0.0450, -0.6123, -0.7518, -0.2824, 0.7085, -0.0711, 0.9625,
       -0.4746, 0.1617, -0.4592, -0.3113, 0.6800, -0.6694, 0.1512, -0.7048,
       0.3421, -0.9658, 0.9809, -0.1205, 0.4730, -0.1665, 0.9956, 0.8720,
       0.9849, -0.7650, 0.4528, 0.2190, 0.9611, -0.0257)
```

The following code gives the R functions needed for running the NR algorithm.

```
LogLik <- function(theta, x) {sum(log((1 + theta * x) / 2))}
LogLikGrad <- function(theta, x) {sum(x / (1 + theta * x))}
LogLikHess <- function(theta, x) {-sum((x / (1 + theta * x))^2)}
NRoptim <- function(theta0, x, eps = 1e-06, trace = FALSE) {
  diff <- Inf
  theta <- theta0
  LL <- LogLik(theta, x)
  grad <- LogLikGrad(theta, x)
  hess <- LogLikHess(theta, x)
  detail <- data.frame(theta = theta, LL = LL, grad = grad, hess = hess, diff = diff)
  while (diff > eps) {
    theta.old <- theta
    theta <- theta.old - grad / hess
    diff <- abs(theta - theta.old)
    LL <- LogLik(theta, x)
    grad <- LogLikGrad(theta, x)
    hess <- LogLikHess(theta, x)
    detail <- rbind(detail, c(theta = theta, LL = LL, grad = grad, hess = hess, diff = diff))
  }
  if(trace) print(detail)
  c(Estimate = theta, Std.Error = sqrt(-1/hess))
}
```

Let's run this function with starting point -0.3.

```
NRoptim(-0.3, x, trace = TRUE)
```

```
   theta     LL       grad   hess      diff
1 -0.300 -22.5  7.93e+00 -18.1       Inf
2  0.140 -20.4  1.96e+00 -11.5 4.40e-01
3  0.310 -20.2 -1.24e-02 -11.8 1.71e-01
4  0.309 -20.2 -3.66e-06 -11.8 1.05e-03
5  0.309 -20.2 -3.16e-13 -11.8 3.09e-07
```

```
 Estimate Std.Error
    0.309     0.291
```

Here, with `eps = 1e-06`, the NR algorithm reaches its target after only 4 iterations. And since the log-likelihood function is strictly concave, there is no need for further investigation, and the point of convergence (0.309) is certainly the MLE.

The two figures below show how ℓ_n is approximated by $\tilde{\ell}_n$ and how the algorithm moves to the maximum.

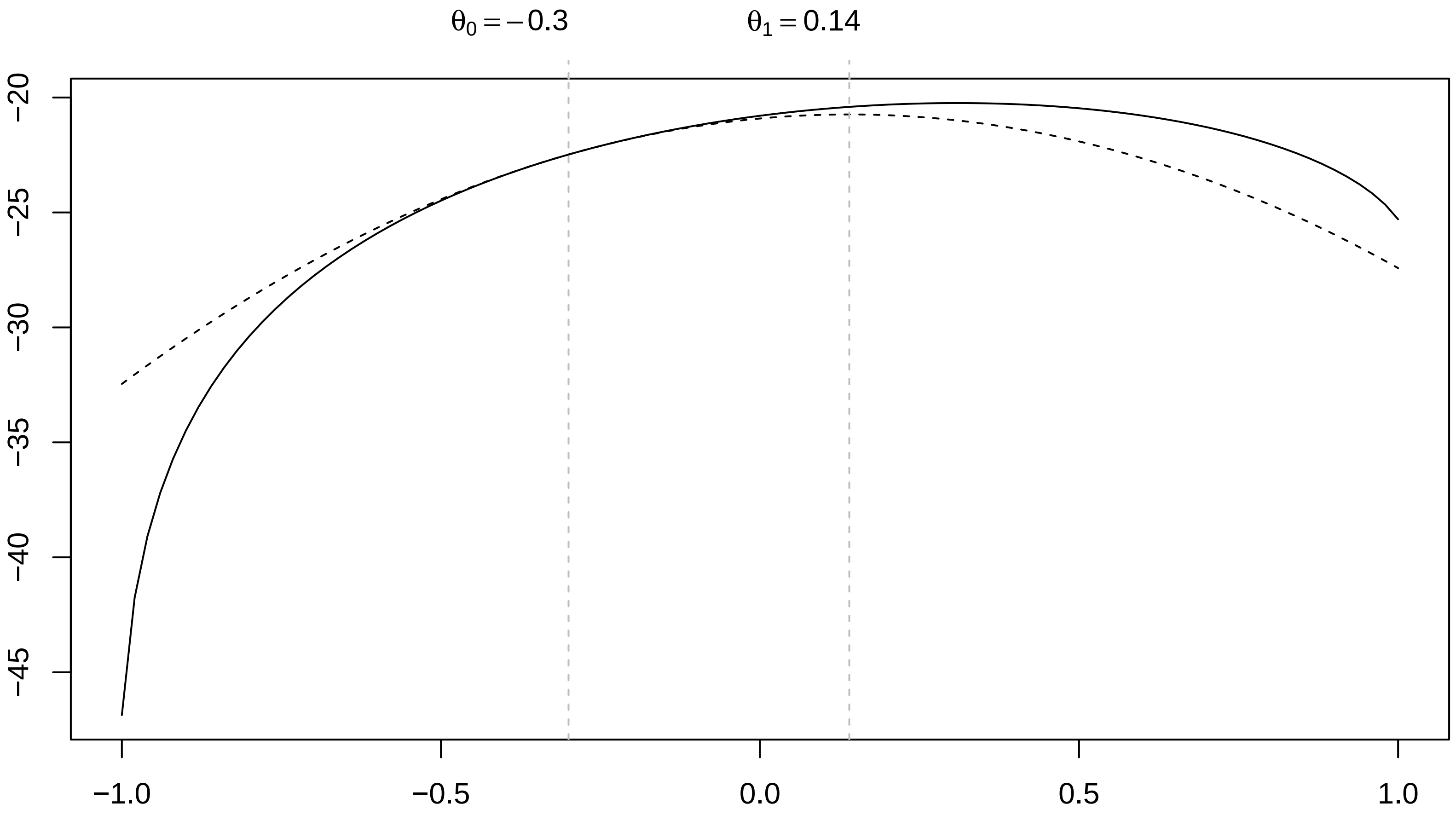


Figure 1: ℓ_n (solid line) and its quadratic approximation $\tilde{\ell}_n$ (dashed line) at $\theta_0 = -0.3$

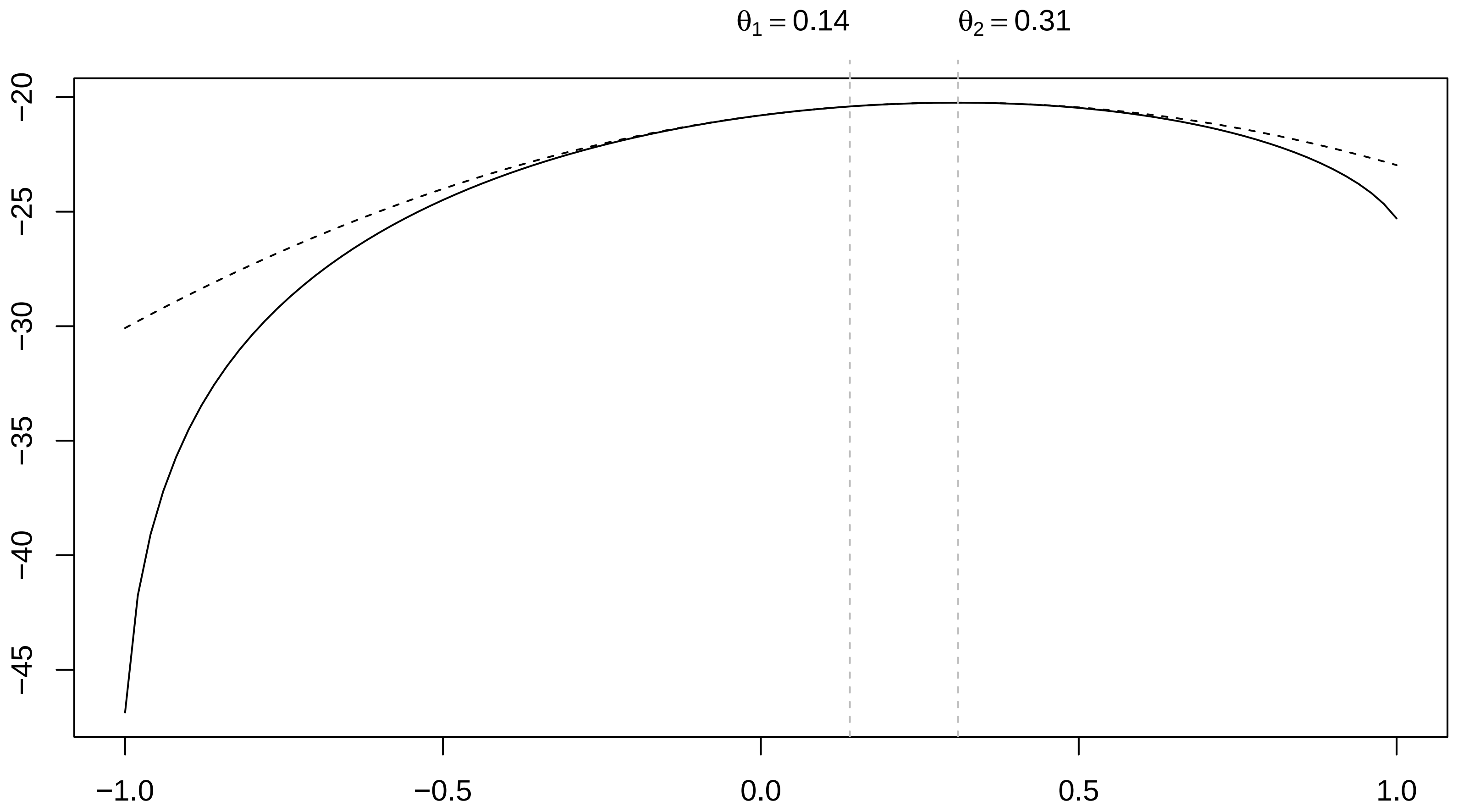


Figure 2: ℓ_n (solid line) and its quadratic approximation $\tilde{\ell}_n$ (dashed line) at $\theta_1 = 0.14$

As starting point, we could used the MoM estimator of θ. In fact, it is easy to check that $E(X) = \theta/3$, so the MoM of θ is $\hat{\theta}_0 = 3\bar{x}_n = 0.36$.

```r
NRoptim(0.36, x)
```

```
 Estimate Std.Error
    0.309     0.291
```

When it works, the NR method converges very quickly to the maximum, especially when the starting point is not very far from the maximizer. However, this method has some serious problems, especially if the likelihood is non-concave:

- NR is sensitive to the starting point.
- A NR step may jump far away from the target.
- As the number of parameters increases, the NR method becomes very computationally expensive.

A large number of enhanced alternative methods (as for example the Quasi-Newton methods) can be found in the literature, but this is beyond the scope of this course.

1.2 Maximum Likelihood in R

R provides a function called `optim()` which, by default, ***performs minimization***. To maximize the likelihood, provide `optim()` with *the negative of the log-likelihood* (for any function f, minimizing $(-f)$ maximizes f).

As main arguments, `optim()` takes: `par`, the starting vector point, i.e. the initial values for the parameters to be optimized over; `fn`: the function to be minimized, *with first argument the vector of parameters over which minimization is to take place*; and some other optional arguments.

```r
optim(par, fn, gr = NULL, ..., # Further arguments to be passed to fn and gr.
      method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN", "Brent"),
      lower = -Inf, upper = Inf, control = list(), hessian = FALSE)
```

The default algorithm (`method`) for `optim()` is a derivative-free optimization routine called the “Nelder-Mead” simplex algorithm. The other optimization methods are: BFGS, CG, L-BFGS-B, ... Here we will use the L-BFGS-B method. L-BFGS-B is a variant of BFGS method, an optimization algorithm in the family of quasi-Newton methods. L-BFGS-B is more memory-efficient than BFGS and allows the incorporation of “box” constraints, i.e. constraints of the form $a < \theta < b$; see the the Help of `optim()` for more details.

`optim()` returns a list of values of which `par`: the local minimizer, `value`: the target function evaluated at the solution found, `convergence`: an integer code indicating successful convergence (code 0) or a warning or an error code (see the Help), and `hessian`: the Hessian at the solution found (if `hessian = TRUE`).

Let's consider again our previous example with $f(x;\theta) = \frac{1+\theta x}{2} I(-1 \leq x \leq 1)$; $-1 < \theta < 1$, and the observed data

```
x <- c(0.9852, 0.0450, -0.6123, -0.7518, -0.2824, 0.7085, -0.0711, 0.9625,
       -0.4746, 0.1617, -0.4592, -0.3113, 0.6800, -0.6694, 0.1512, -0.7048,
       0.3421, -0.9658, 0.9809, -0.1205, 0.4730, -0.1665, 0.9956, 0.8720,
       0.9849, -0.7650, 0.4528, 0.2190, 0.9611, -0.0257)
```

```
negLogLik <- function(theta, x) {-sum(log((1 + theta * x) / 2))}
optim(-0.3, fn = negLogLik, x = x, method = "L-BFGS-B", lower = -1, upper = 1, hessian = TRUE)
```

```
$par
[1] 0.309


$value
[1] 20.2
```

```
$counts
function gradient
       6        6

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"

$hessian
     [,1]
[1,] 11.8
```

There are many packages and functions in R designed to facilitate MLE calculation, most of which call the `optim()` function, in the background, to perform the optimization. This includes the functions `stats4::mle()`, `maxLik::maxLik()` and `MASS::fitdistr()`, to name but a few. In general, these functions make things a little easier to code and perform additional processing to make the results more useful for estimation and/or inference. In the following, we'll look at the `mle()` function.

```r
library(stats4)
mle.negLogLik <- mle(\(theta) negLogLik(theta, x = x), # set the arguments that must be held fixed.
                     start = -0.3, lower = -1, upper = 1, method = "L-BFGS-B")
mle.negLogLik
```

```

Call:
mle(minuslogl = function(theta) negLogLik(theta, x = x), start = -0.3,
    method = "L-BFGS-B", lower = -1, upper = 1)

Coefficients:
theta
0.309
```

The objective returned by the `mle()` function can be used by some useful well-known generic `R` functions like `summary()`, `logLik()`, `vcov()`. This latter gives the asymptotic variance-covariance matrix of the estimated vector. The following example provides an illustration.

Example 1.1 (Two dimensional case)**.** Here we consider the problem of estimating $\boldsymbol{\theta} = (l, s)$ the location-scale parameters of the Cauchy distribution:

$$f(x; l, s) = \frac{1}{\pi s \left[1 + \left(\frac{x-l}{s}\right)^2\right]}, \, l \in \mathbb{R}, s > 0.$$

We start by generating 30 observations with true values $l = 0$ and $s = 1$.

```
set.seed(1)
y <- rcauchy(30)
y
```

```
 [1]   1.1025   2.3538  -4.2926  -0.2966   0.7346  -0.3305  -0.1756  -1.8082
 [9]  -2.3286   0.1966   0.7556   0.6195  -1.5015   2.6241  -0.8825 138.3476
[17]  -1.2274  -0.0254   2.5265  -0.8409  -0.2081   0.7865  -1.9374   0.4163
[25]   1.1145   2.6747   0.0421   2.5821  -0.4339   1.8237
```

```r
neglogLi2 <- function(l, s, y) {-sum(dcauchy(y, location = l, scale = s, log = TRUE))}
mle.negLogLik2 <- mle(\(l, s) neglogLi2(l, s, y = y),
                      start = list(l = median(y), s = IQR(y)),
                      lower = c(-Inf, 0), upper = c(Inf, Inf),
                      method = "L-BFGS-B")
summary(mle.negLogLik2)
```

```
Maximum likelihood estimation

Call:
mle(minuslogl = function(l, s) neglogLi2(l, s, y = y), start = list(l = median(y),
    s = IQR(y)), method = "L-BFGS-B", lower = c(-Inf, 0), upper = c(Inf,
    Inf))

Coefficients:
  Estimate Std. Error
l    0.129      0.266
s    0.962      0.235
```

```
-2 log L: 141
```

```r
# the maximum log-likelihood value,
# i.e. the log-likelihood function evaluated at the MLE
logLik(mle.negLogLik2)
```

```
'log Lik.' -70.6 (df=2)
```

```r
# asymptotic variance-covariance matrix
vcov(mle.negLogLik2)
```

```
        l       s
l 0.07082 0.00425
s 0.00425 0.05520
```

This latter is nothing but the inverse of the negative of the Hessian matrix (of the log-likelihood) evaluated at the maximum likelihood.

```r
solve(mle.negLogLik2@details$hessian)
```

```
        l       s
l 0.07082 0.00425
s 0.00425 0.05520
```

Remarks

- In the example above, the same solution can be found directly via `optim()` as follows

  ```r
  neglogLi2 <- function(ls, y) {
    -sum(dcauchy(y, location = ls[1], scale = ls[2], log = TRUE))
  }
  optim(c(l = median(y), s = IQR(y)), fn = neglogLi2, y = y, method = "L-BFGS-B",
    lower = c(-Inf, 0), upper = c(Inf, Inf), hessian = TRUE)
  ```

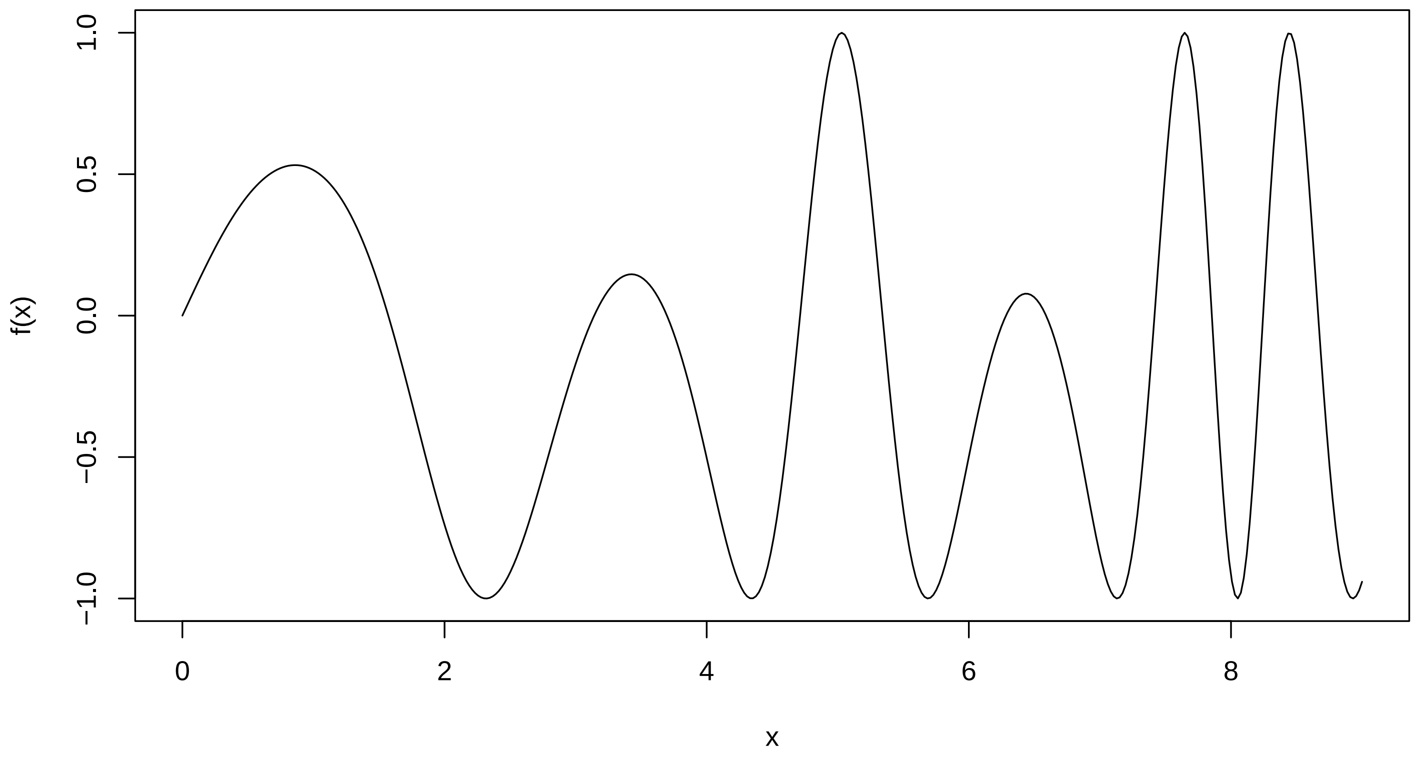
- The same can be done as follows; see the `Help` for more details

```r
MASS::fitdistr(y, densfun = "cauchy")
```

```
  location    scale
  0.129       0.962
 (0.266)     (0.235)
```

- Be careful when performing numerical optimization, as there is no guarantee that the resulting convergence point is actually the global maximizer. Here is an illustration.

```r
f <- function(x) sin(x * cos(x))
curve(f, 0, 9, n = 400)
```



```
mle(\(x) -f(x), start = 2)@coef
[1] 0.86
mle(\(x) -f(x), start = 4)@coef
[1] 3.43
mle(\(x) -f(x), start = 5)@coef
```

```
[1] 5.03
mle(\(x) -f(x), start = 6)@coef
[1] 8.45
```

The problem becomes more and more difficult as the number of parameters increases.

2 Profile likelihood

Although the definition of likelihood covers the *multiparameter case*, the resulting multidimensional likelihood function can be difficult to deal with. Furthermore, in many practical multi-parameter problems, only a subset of the parameters is of interest; in the normal model, we might be interested only in the mean μ, while σ^2 is a “nuisance“, which is there only to make the model correct. And even if we are interested in several parameters, it is always easier to study a single parameter or only a very few at once.

For a given model, let (θ, η) be the full set of parameters (θ and η may be vectors), where both θ and η are unknown. Let’s say that θ is our primary parameter of interest and η is a nuisance parameter.

Definition 2.1 (Profile likelihood)**.** Given a model with (full) likelihood $L(\theta, \eta)$, *the profile likelihood for* θ is

$$L_p(\theta) = max_\eta L(\theta, \eta).$$

Let $\hat{\eta}(\theta) = \arg\max_\eta L(\theta, \eta)$, i.e. $\hat{\eta}(\theta)$ is the maximizer of the function $\eta \mapsto L(\theta, \eta)$ when θ *is regarded as known*. In this way, the definition of the profile likelihood can be made more explicit using

$$L_p(\theta) = L(\theta, \hat{\eta}(\theta)).$$

Let $\hat{\theta} = \arg\max_\theta L_p(\theta)$, a bit of logical deduction shows that $(\hat{\theta}, \hat{\eta}(\hat{\theta}))$ is the MLE of (θ, η). In fact,

$$L(\hat{\theta}, \hat{\eta}(\hat{\theta})) = L_p(\hat{\theta}) \geq L_p(\theta) = max_\eta L(\theta, \eta) \geq L(\theta, \eta), \; \forall (\theta, \eta).$$

This procedure is illustrated in the following example.

Example 2.1. Let X_i, $i = 1, \ldots, n$, be an iid sample from $N(\mu, \sigma^2)$, where $\mu \in (-\infty, \infty)$, and $\sigma^2 \in (0, \infty)$ are unknown.

The log-likelihood is

$$\ell(\mu, \sigma^2) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\big(\sigma^2\big) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2$$

We know that if we consider μ as known and maximize this function with respect to σ^2 we get the following explicit expression for the MLE of σ^2

$$\hat{\sigma}^2(\mu) = n^{-1}\sum_{i=1}^{n}(x_i - \mu)^2.$$

So, the profile log-likelihood for μ is

$$\ell_p(\mu) = \ell(\mu, \hat{\sigma}^2(\mu)) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2\right) - \frac{n}{2}.$$

Clearly, the $\arg\max_\mu \ell_p(\mu)$ is $\bar{x}_n$, and so the MLE of (μ, σ^2) is $\big(\overline{X}_n, n^{-1}\sum_{i=1}^{n}(X_i - \overline{X}_n)^2\big)$. $\square$

In addition to reducing the dimension of the problem, profile likelihood functions can be used in the same way as (ordinary) likelihood functions not only to estimate efficiently the parameter(s) of interest, but also to obtain their asymptotic variance, which is equal to the inverse of the negative of the second derivative (or the Hessian) of the profile log-likelihood. Profile likelihood is also very useful when it comes to inference; more on this later. Note, however, that in practice it is rarely possible to calculate the profile likelihood explicitly (as we did above for the Normal density); numerical evaluation is typically the way to proceed.

Exercise 2.1. Let X_i, $i = 1, \dots, n$, be an iid sample from the gamma distribution with pd

$$f(x; \alpha, \sigma) = \frac{1}{\sigma^\alpha \Gamma(\alpha)} x^{\alpha-1} \exp(-x/\sigma) I(x > 0),$$

where $\alpha > 0$ (shape), $\sigma > 0$ (scale) are unknown, and $\Gamma(\cdot)$ is the gamma function.

Show that the maximum likelihood estimator of (α, σ) is $(\hat{\alpha}, \hat{\alpha}^{-1}\overline{X}_n)$ where $\hat{\alpha}$ is the arg max of

$$\ell_p(\alpha) = -n \log \Gamma(\alpha) - n\alpha \log\left(\alpha^{-1}\bar{X}_n\right) + (\alpha - 1) \sum_{i=1}^{n} \log(X_i) - n\alpha.$$

Use the above result to calculate the MLE of (α, σ) from the following simulated data. Estimate the asymptotic standard deviation of $\hat{\alpha}$.

```r
set.seed(1)
x <- rgamma(n = 35, shape = 5, scale = 1/3)
x
```

```
 [1] 1.090 2.588 2.535 1.808 0.609 1.864 2.068 1.935 1.292 0.988 1.093 1.296
[13] 1.468 0.936 2.137 1.949 2.220 2.104 1.553 0.423 1.877 2.366 1.051 1.181
[25] 1.338 1.428 1.076 0.684 2.284 1.212 2.379 2.088 1.386 1.197 1.919
```

3 Likelihood for regression models

Consider the simple linear regression model with Gaussian noise, defined as $Y = \beta_0 + \beta_1 x + \epsilon$. ϵ is an unobservable noise/error variable with $N(0, \sigma^2)$ distribution. The aim is to estimate and make inference about $\boldsymbol{\theta} = (\beta_0, \beta_1, \sigma^2)$ on the basis of an iid sample (Y_i, x_i) of (Y, x). For now, we assume that x_i are *known non-random* quantities (fixed design). According to this model $Y_i \equiv Y|X = x_i \sim N(\beta_0 + \beta_1 x_i, \sigma^2)$, so the log-likelihood function is given by

$$\ell_n(\boldsymbol{\theta}) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2.$$

It's easy to see that maximizing this function is equivalent to resolve the following optimization problems

$$(1)\ \ (\hat\beta_0, \hat\beta_0) = \arg\min_{\beta_0,\beta_1} \sum_{i=1}^{n} (x_i - (\beta_0 + \beta_1 x_i))^2,$$
$$(2)\ \ \hat\sigma^2 = \arg\min_{\sigma^2} \ell_n(\hat\beta_0, \hat\beta_1, \sigma^2).$$

Equation (1) above is identical to the optimization equation that defines linear least-squares (LS) regression. So the MLE of (β_0, β_1) is the same the least squares estimator. In fact, setting the derivatives to zero and solving produces

$$\hat\beta_0 = \overline{Y} - \hat\beta_1 \bar{x},\ \hat\beta_1 = \frac{\overline{xY} - \overline{x}\overline{Y}}{\overline{x^2} - \overline{x}^2},\ \text{and}\ \hat\sigma^2 = n^{-1} \sum_{i=1}^{n} (Y_i - (\hat\beta_0 + \hat\beta_1 x_i))^2.$$

Maximum likelihood theory can of course be applied in this situation (Fisher information, asymptotic normality/efficiency, etc.).

For a random sampling design in which X_i are also sampled, we have that $Y_i|X_i \sim N(\beta_0 + \beta_1 X_i, \sigma^2)$ and the (full or "joint") likelihood is

$$\prod_i f_{Y|X}(y_i|x_i; \boldsymbol{\theta}) f_X(x_i),$$

where $f_{Y|X}$ is the pd of $Y|X$, and f_X is the pd of X. If the latter does not depend on $\boldsymbol{\theta}$, as is commonly supposed, then f_X plays no role in the likelihood function. In this case, it is equivalent to work with the "joint" log-likelihood or with the "conditional" log-likelihood as given by $\sum_i \log f_{Y|X}(y_i|x_i;\boldsymbol{\theta}) = \ell_n(\boldsymbol{\theta})$, i.e. the same as for fixed design. Consequently, the estimators and information matrix are the same as before, and the asymptotic results are the same also.

```r
set.seed(5)
x <- 1:10
y <- 10 + 20 * x + rnorm(10, sd = 10)

neglogLiReg <- function(a, b, s, y) {-sum(dnorm(y, mean = a + b * x, sd = s, log = TRUE))}
mle(\(a, b, s) neglogLiReg(a, b, s, y = y),
    start = list(a = mean(y), b = 0, s = sd(y))) |> coef()
     a      b      s
10.445 19.775  9.011

lm(y ~ x) |> coef()
(Intercept)           x
      10.45       19.77
```

The same theory applies to multiple linear regression. Without loss of generality, let's consider the case of 2 covariates X_1 and X_2. The linear equation is $Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon$, with $\epsilon \sim N(0, \sigma^2)$. The log-likelihood (under fixed or random design) is

$$-\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\left(\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mu_i)^2,$$

where $\mu_i = \boldsymbol{\beta}^t \boldsymbol{x}_i$, $\boldsymbol{\beta}^t = (\beta_0, \beta_1, \beta_2)$ and $\boldsymbol{x}_i^t = (x_{i0}, x_{i1}, x_{i2})$, with $x_{i0} = 1$, $\forall i$. Here again, the maximum likelihood method and the LS method lead to the same estimators. $\hat{\boldsymbol{\beta}}$ is the root of the system of equations

$$\boldsymbol{\nabla}_{\boldsymbol{\beta}} \sum_{i=1}^{n}(y_i - \mu_i)^2 := \left(\partial_{\beta_1}\sum_{i=1}^{n}(y_i - \mu_i)^2, \partial_{\beta_2}\sum_{i=1}^{n}(y_i - \mu_i)^2, \partial_{\beta_3}\sum_{i=1}^{n}(y_i - \mu_i)^2\right) = \mathbf{0}.$$

Since, $\partial_{\beta_k}\sum_{i=1}^{n}(y_i - \mu_i)^2 = -2\sum_i (y_i - \mu_i)x_{ik}$,

$$\boldsymbol{\nabla}_{\boldsymbol{\beta}} \sum_{i=1}^{n}(y_i - \mu_i)^2 = -2(\boldsymbol{X}^t \boldsymbol{y} - \boldsymbol{X}^t \boldsymbol{X}\boldsymbol{\beta}),$$

where $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^t \\ \vdots \\ \boldsymbol{x}_n^t \end{bmatrix}$ and $\boldsymbol{y}^t = (y_1, \ldots, y_n)$. It fallows that $\hat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^t \boldsymbol{X}\right)^t \boldsymbol{X}^t \boldsymbol{Y}$. The maximum likelihood estimator of σ^2 has the same form as in the one-covariate case, i.e., the average squared residual $n^{-1} \sum_i (Y_i - \hat{\mu}_i)^2$, with $\hat{\mu}_i = \hat{\boldsymbol{\beta}}^t \boldsymbol{x}_i$.

Linear regression is the simplest regression model that can be estimated and inferred using likelihood theory. This theory can be applied to a very wide range of other regression setting such as, for example, non-linear regression models, generalized linear models (based on the exponential family), generalized linear mixed models, censored regression models, to name only a few.