Practice Lecture 3: The likelihood function

Erlis Ruli (ruli@stat.unipd.it)

26 October 2021

1 The Bernoulli model

Consider again the example in Lecture 3, Section 3.2. In which we assume that the observed sample

$$(0, 1, 1, 1, 0, 1, 1, 0, 1, 1)$$

is a realisation of the random sample $X_i \overset{i.i.d.}{\sim} \text{Ber}(\theta)$, where θ is an unknown parameter. We show how to define the likelihood function in `R` and how to plot it.

Call `yobs` the vector with the observed sample

```
> yobs <- c(0,1,1,1,0,1,1,0,1,1)
```

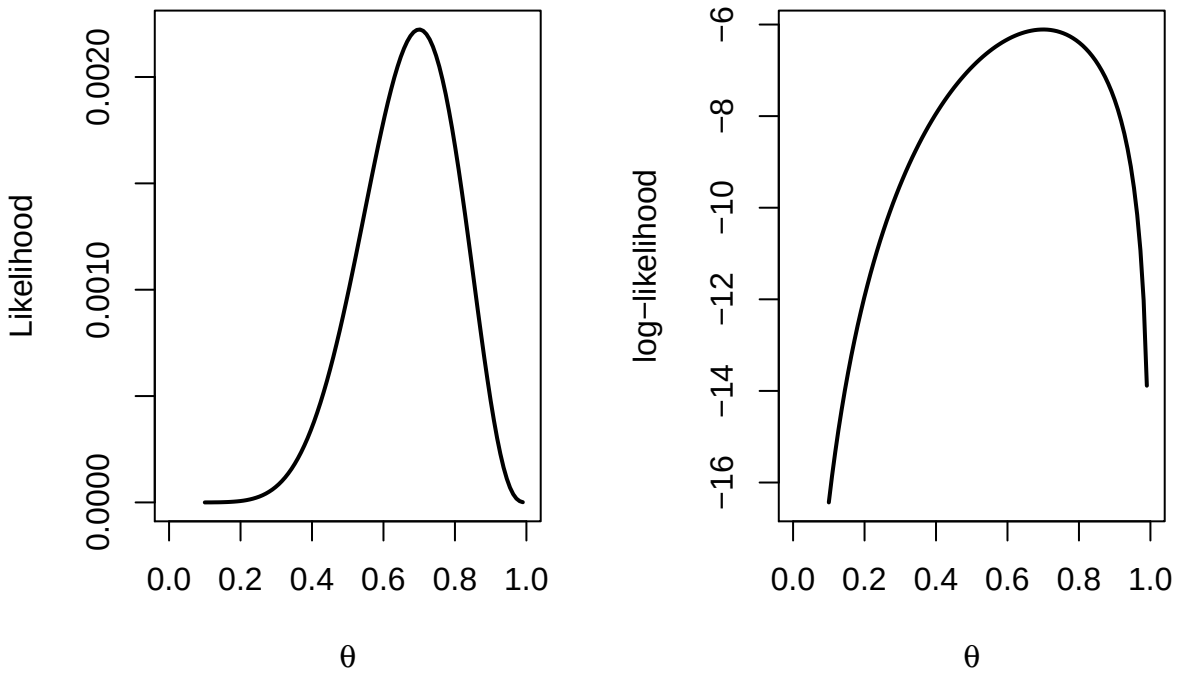
To define a likelihood function in `R` we can either implement everything manually or resort to the `R`'s implementation of the p.d.f. of the involved r.v.'s. Whenever possible, option 2 is less error-prone.

```
> # likelihood function: option 1
> Lik1.ber <- function(theta, y){
+   n = length(y)
+   vi = theta^y  * (1-theta)^(1-y)
+   return(prod(vi))
+ }
>
> # log-likelihood function: option 1
> lLik1.ber <- function(theta, y){
+   n = length(y)
+   vi = y * log(theta)  + (1-y)*log(1-theta)
+   return(sum(vi))
+ }
>
>
> # likelihood function: option 2
> Lik2.ber <- function(theta, y){
+   vi = dbinom(y, size=1, prob = theta)
+   return(prod(vi))
+ }
```

Here is the plot of the likelihood function and of the log-likelihood function

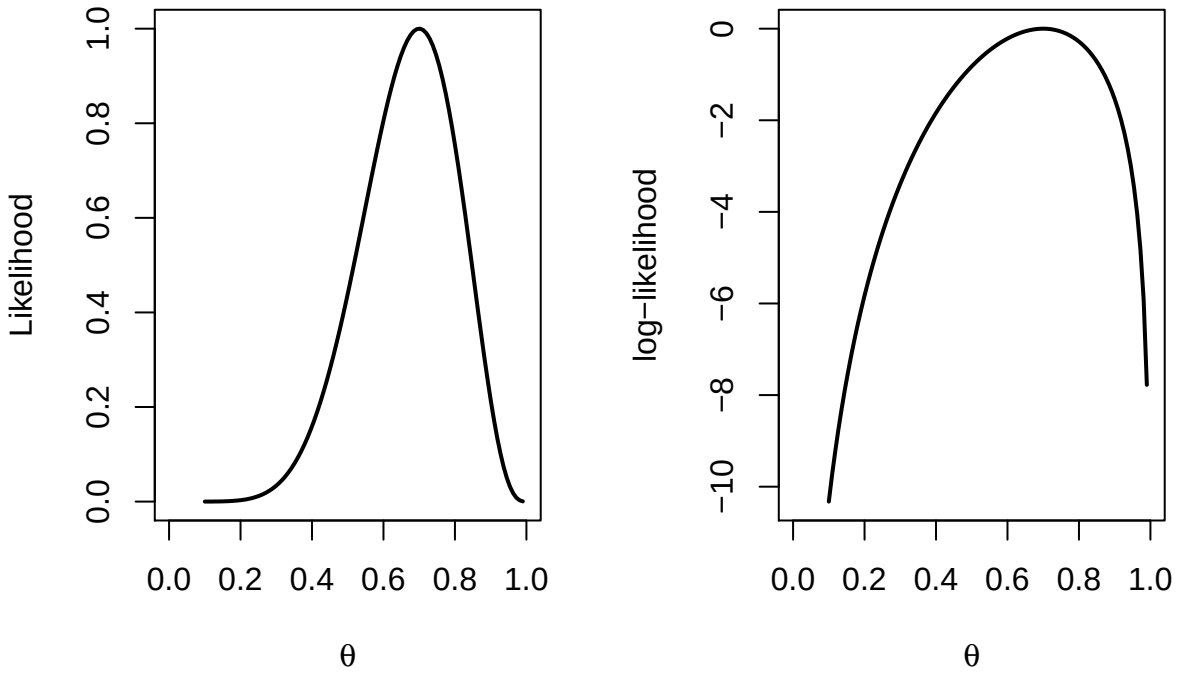
```
> par(mfrow=c(1,2))
> theta.gr <- seq(0.1,0.99, len=100)
> fLik <- sapply(theta.gr, Lik1.ber,  y=yobs)
> flLik <- sapply(theta.gr, lLik1.ber,  y=yobs)
> plot(theta.gr, fLik, xlim=c(0,1),lwd=2,
+      xlab=expression(theta),
```

```
+      type="l",ylab="Likelihood")
> plot(theta.gr, flLik, xlim=c(0,1),lwd=2,
+      xlab=expression(theta),
+      type="l", ylab="log-likelihood")
```



To plot the relative likelihood $RL(\theta)$ (and the relative log-likelihood, i.e. $\log RL(\theta)$) function we must scale the likelihood by its maximum, thus a quick (and dirty) way to do that is

```
> par(mfrow=c(1,2))
> plot(theta.gr, fLik/max(fLik), xlim=c(0,1),lwd=2,
+      xlab=expression(theta),
+      type="l",ylab="Likelihood")
> plot(theta.gr, flLik-max(flLik), xlim=c(0,1),lwd=2,
+      xlab=expression(theta),
+      type="l", ylab="log-likelihood")
```



Notice that while the maximum of the relative likelihood function is 1, the maximum value of the relative log-likelihood is $\log 1 = 0$.

We saw in Section 3.2 that the maximum of the likelihood function is reached at the value $\hat{\theta} = 0.7$. To compute this maximum numerically we use the log-likelihood function since the latter is numerically more stable.

```
> # note: by default optimize performs numerical minimisation,
> # thus we use the option maximum=TRUE
> optimize(f = lLik1.ber, interval = c(0.1, 0.99),
+          y = yobs,
+          maximum = TRUE)
```

```
## $maximum
## [1] 0.7000096
##
## $objective
## [1] -6.108643
```

This numerical result agrees with the analytical solution.

R also has routines for approximating numerical derivatives of the given function. These are provided by an external package that has to be installed and loaded. For instance, if we wished to compute the observed information, i.e. the negative of the second derivative of the log-likelihood, evaluated at $\theta = 0.7$ we could do the following.

```
> # if not already installed
> #install.packages("numDeriv")
>
> # after installation, load the package
> library(numDeriv)
>
> # check first that the first-order cond. is satisfied
> grad(lLik1.ber,0.7, y=yobs)
```

```
## [1] -2.71399e-11
```

```
> # observed information at theta=0.7
> -hessian(lLik1.ber,0.7, y=yobs)
```

```
##           [,1]
## [1,] 47.61905
```

```
> # check that this agrees with the analytic version
> 3/(1-0.7)^2 + 7/0.7^2
```

```
## [1] 47.61905
```

In the code above we also showed how to compute the first derivative of the log-likelihood function, by means of the `grad` function.

2 The Weibull model

Consider the example in Lecture 3, Section 3.3. In which we assume that the observed sample

$$(5.1, 7.4, 10.9, 21.3, 12.3, 15.4, 25.4, 18.2, 17.4, 22.5)$$

is a realisation of the random sample $X_i \overset{i.i.d.}{\sim} \text{Wei}(\alpha, \beta)$, where α, β are the unknown parameters. We show how to define the likelihood function in `R` and how to plot it. For numerical stability reasons it is better to work with a slightly different version of the Weibull distribution, which p.d.f. is given by

$$f(y; \alpha, \lambda) = \frac{\alpha}{\lambda^\alpha} \left(\frac{y}{\lambda}\right)^{\alpha-1} e^{-(y/\lambda)^\alpha}.$$

This version of the Weibull distribution corresponds to the one given in the Lecture 1 and Lecture 3 with $\beta = \lambda^\alpha$. Furthermore, this version is also the one implemented in `R`.

Set the observed sample in the memory of `R`

```
> yobs=c(5.1, 7.4, 10.9, 21.3, 12.3, 15.4,
+        25.4, 18.2, 17.4, 22.5)
```

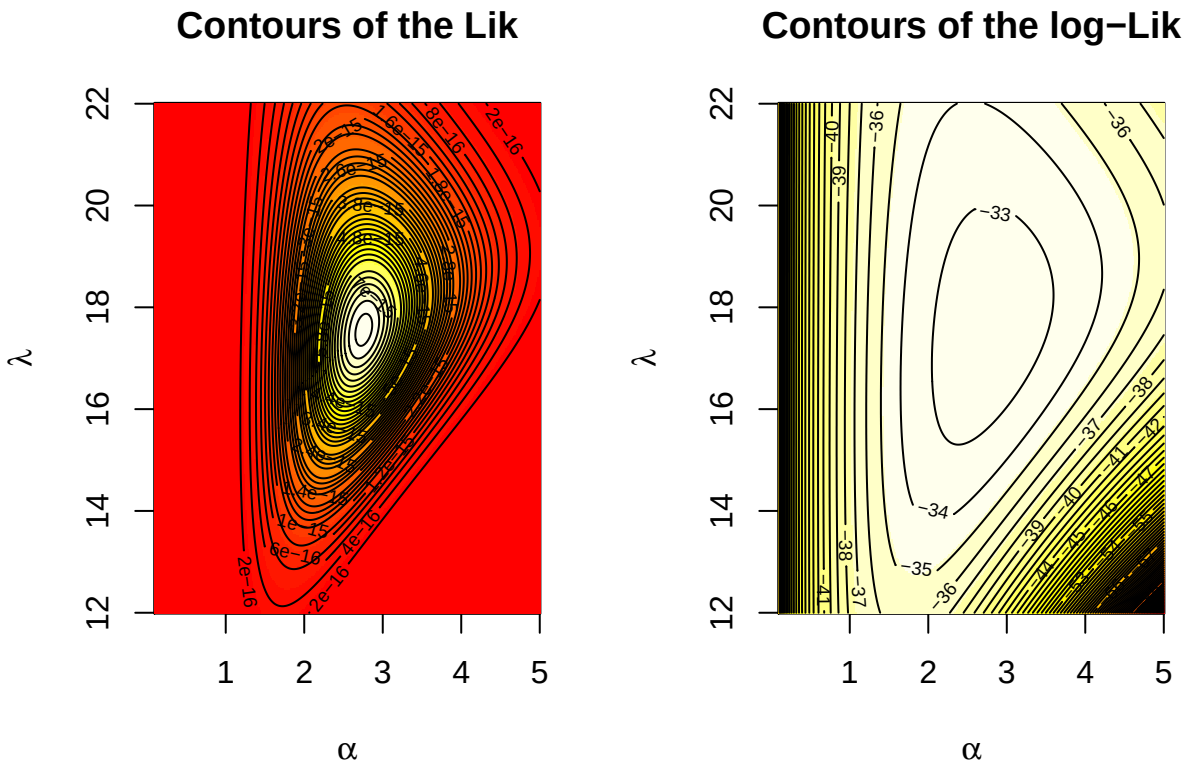
To define the likelihood function, this time we use the second option.

```
> # likelihood function
> Lik.wei <- function(theta, y){
+   # using dwibull(x, alpha, beta)
+   vi = dweibull(y, shape = theta[1], scale = theta[2])
+   return(prod(vi))
+ }
>
> # log-likelihood function
> lLik.wei <- function(theta, y){
+   n = length(y)
+   vi = dweibull(y, shape = theta[1], scale = theta[2], log = TRUE)
+
+   return(sum(vi))
+ }
```

Now we plot the likelihood surface. Actually we consider two versions, one plots the contour levels of the likelihood function and the other plots the contour levels of the log-likelihood function.

```
> # set a regular grid for alpha and beta
> theta1 <- seq(0.1, 5, len=300)
> theta2 <- seq(12,22, len=300)
```

```
>
> # build the bivariate grid
> theta12 <- expand.grid(theta1,theta2)
>
> # evaluate the log-Lik over the grid
> fLik <-apply(theta12,1,Lik.wei,y=yobs)
> flLik <- apply(theta12,1,lLik.wei,y=yobs)
>
> # transform the previous objects in matrices
> Lzmat <- matrix(fLik, ncol=300, byrow = F)
> lLzmat <-matrix(flLik, ncol=300, byrow = F)
>
> par(mfrow = c(1,2))
> image(theta1,theta2, -Lzmat, col = heat.colors(30, rev = TRUE),
+       xlab=expression(alpha),
+         ylab=expression(lambda), main="Contours of the Lik")
> contour(theta1,theta2, Lzmat, nlevels = 30, add = TRUE)
> image(theta1,theta2, -lLzmat,col = heat.colors(30, rev = TRUE),
+       xlab=expression(alpha),
+         ylab=expression(lambda),main="Contours of the log-Lik")
> contour(theta1,theta2, lLzmat, nlevels = 100, add=TRUE)
```



In Lecture 3 we saw that the maximum the likelihood (or log-likelihood function) is reached at the point $\hat{\theta}$ with coordinates $(2.8, 17.6)$. To get this value, we first solved $\partial\ell(\alpha,\lambda)/\partial\lambda = 0$ in λ and found the partial solution for a fixed α, i.e. $\hat{\lambda}(\alpha) = \left(\frac{1}{n}\sum_{i=1}^{n} y_i^{\alpha}\right)^{1/\alpha}$. Then, replacing λ by $\hat{\lambda}(\alpha)$ in $\partial\ell(\alpha,\lambda)/\partial\alpha = 0$ gives the equation

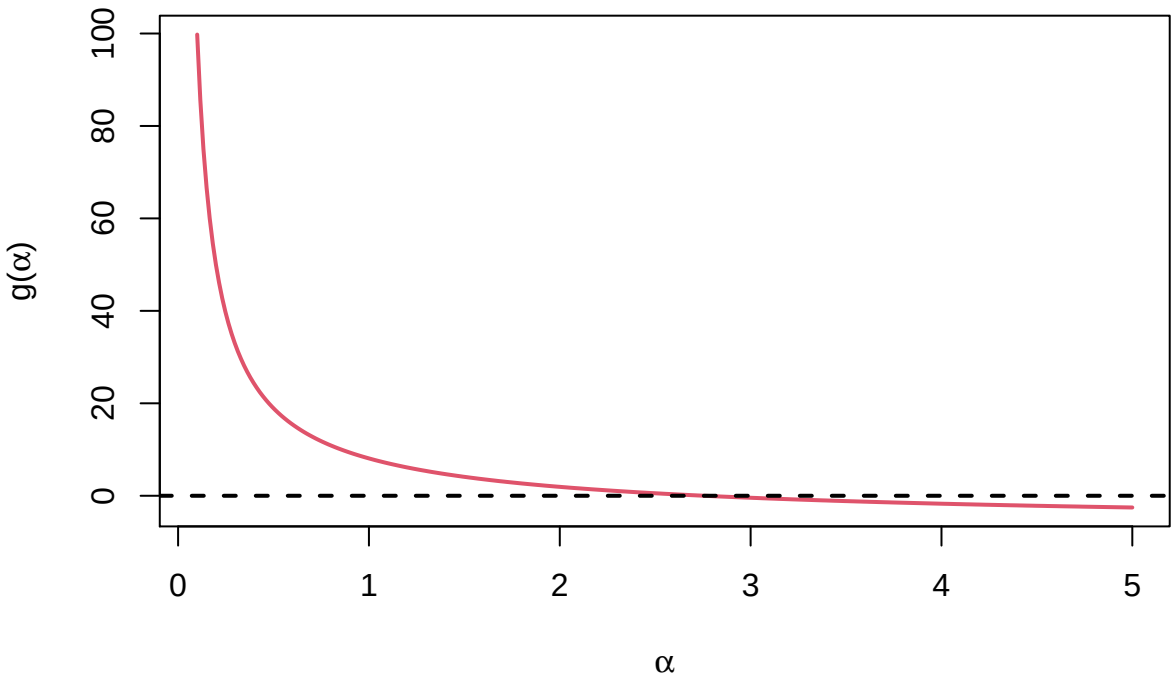
$$g(\alpha) = \sum_{i=1}^{n} \log y_i + \frac{n}{\alpha} - \frac{n\sum_{i=1}^{n} y_i^{\alpha}\log y_i}{\sum_{i=1}^{n} y_i^{\alpha}} = 0$$

Let' define g first.

```
> # define the function g(alpha)
> g.a <- function(a, y){
+   n = length(y)
+   oo = sum(log(y)) + n/a - n*sum(y^a * log(y))/sum(y^a)
+
+   return(oo)
+ }
```

Here is how the function looks like

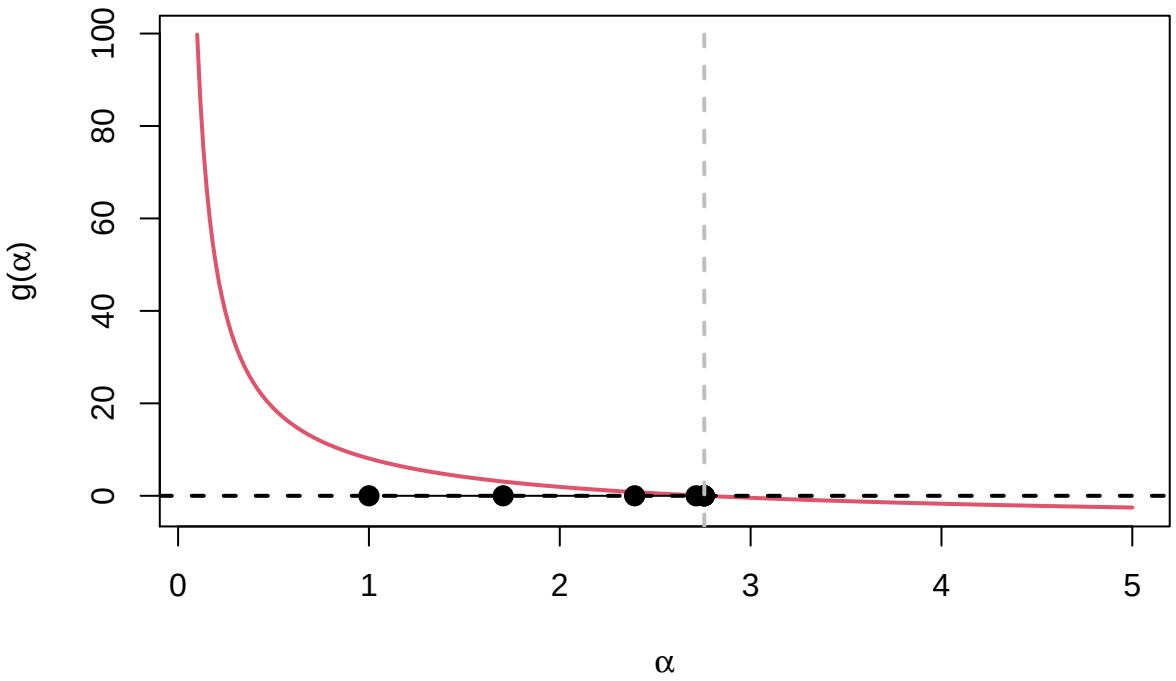
```
> ga <- sapply(theta1, g.a, y=yobs)
> plot(theta1, ga, xlim=c(0.1, 5),
+      xlab=expression(alpha), ylab=expression(paste("g(",alpha,")")),
+      type="l", lwd=2, col=2)
> abline(h=0, lwd=2, lty=2)
```



Since this cannot be solved analytically, let's try to get the first terms of the sequence given by the Newton-Raphson method, in which we approximate $g'(\alpha)$ numerically

```
> theta1. <- theta1
> # set the starting point alpha0
> theta0 <- 1
>
> # this is a numericall approximation of dg(a)/da
> g.a.prime <- function(a,y) {
+   dd <- grad(function(x) g.a(x, y), a)
+ }
>
>
> # compute the first 7 iterations
> theta1 <- theta0 - g.a(theta0, yobs)/g.a.prime(theta0, yobs)
> theta2 <- theta1 - g.a(theta1, yobs)/g.a.prime(theta1, yobs)
```

```
> theta3 <- theta2 - g.a(theta2, yobs)/g.a.prime(theta2, yobs)
> theta4 <- theta3 - g.a(theta3, yobs)/g.a.prime(theta3, yobs)
> theta5 <- theta4 - g.a(theta4, yobs)/g.a.prime(theta4, yobs)
> theta6 <- theta5 - g.a(theta5, yobs)/g.a.prime(theta5, yobs)
> theta7 <- theta6 - g.a(theta6, yobs)/g.a.prime(theta6, yobs)
>
> plot(theta1., ga, xlim=c(0.1, 5),
+      xlab=expression(alpha), ylab=expression(paste("g(",alpha,")")),
+      type="l", lwd=2, col=2)
> abline(h=0, lwd=2, lty=2)
>
> points(y = rep(0,8),
+      x = c(theta0,theta1, theta2,theta3,theta4,theta5,theta6,theta7),
+      xlab="iteration",
+      ylab = "hat.alpha at iteration", pch=20, cex=2,
+      type="b", main="Newton-Raphson iterations")
> abline(v = 2.75717, lty=2, lwd=2, col="grey")
```



From the plot we wee that at iteration 7 the Newton-Raphson algorithm has already reached convergence since the sequence of $\hat{\alpha}_i$ seem to have converged to the the horizontal line at the point 2.75717, thus we conclude that $\hat{\alpha} = 2.75717$. Obviously we need to a criteria for stopping the sequence, there are many possibilities to accomplish this. However, instead of doing this by hand, in order to find the solution of $g(\alpha) = 0$, we will use the command `uniroot` as follows

```
> (oo <- uniroot(g.a, interval = c(0.1, 10), y=yobs))
```

```
## $root
## [1] 2.757155
##
## $f.root
## [1] -9.16928e-07
##
## $iter
```

```
## [1] 7
##
## $init.it
## [1] NA
##
## $estim.prec
## [1] 8.872539e-05
```

Form the output of the command we see that the solution, i.e. the root of $g(\alpha) = 0$, is found to be $\hat{\alpha} = 2.757154$. Plugging in $\hat{\alpha}$ in $\hat{\lambda}(\alpha)$ we get $\hat{\lambda}(\hat{\alpha}) = \hat{\lambda}$, i.e.

```
> hat.alpha <- oo$root
> hat.lambda <- (mean(yobs^hat.alpha))^(1/hat.alpha)
```

Note that $\hat{\beta} = \hat{\lambda}^{\hat{\alpha}}$.

To compute the maximum of the likelihood function we can also use a hill-climbing approach. However, since we have more than one parameter, the numerical optimisation routine `optimize` cannot be used. This time we use `optim` with the option `L-BFGS-B` which is the bound and low-memory version of the Broyden–Fletcher–Goldfarb–Shanno algorithm.

```
> # we let the legthy output of optim be placed
> # in the object oo and not printed on the screen
> # par = c(2,16): is the starting value of the optimizer
> # fn = function(x) -lLik.wei(x,yobs) redefines the log-likelihood function
> #  multiplying it by -1.
> oo <- optim(par = c(2,1),
+             fn = function(x) -lLik.wei(x,yobs),
+             method="L-BFGS-B",
+             lower = c(0.5,0.1),
+             upper = c(4,100),
+             hessian = TRUE) # with this option we get the J_n matrix
> oo
```

```
## $par
## [1]  2.757154 17.556445
##
## $value
## [1] 32.42347
##
## $counts
## function gradient
##       30       30
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
##
## $hessian
##            [,1]       [,2]
## [1,]  2.1202361 -0.2200252
## [2,] -0.2200252  0.2466318
```

The result of optim agrees with the one provided previously. The object `oo` also reports the Hessian matrix at the maximum. Since we miminised the negative log-likelihood function, this Hessian matrix is also the

observed information matrix evaluated at $\hat{\theta}$, i.e. $J_n(\hat{\theta})$. To check the second-order condition it is sufficient o check that the eigenvalues of $J_n(\hat{\theta})$ are all positive.

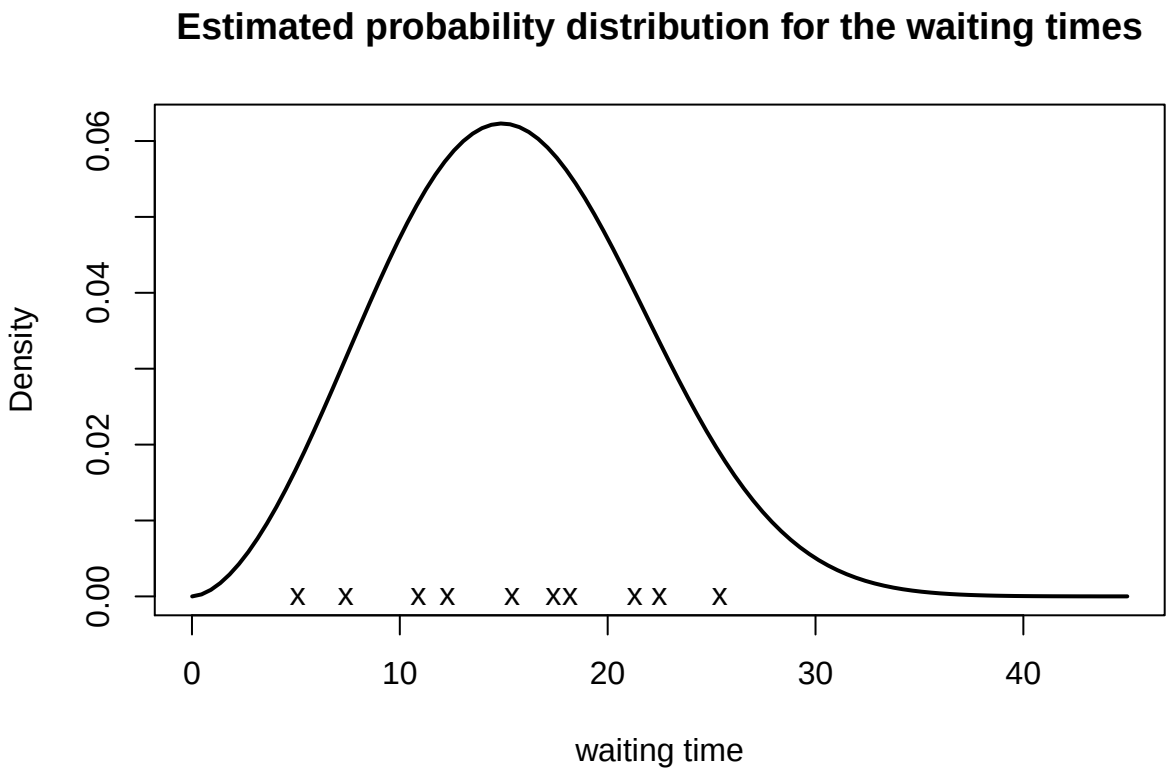
```
> eigen(oo$hessian)
```

```
## eigen() decomposition
## $values
## [1] 2.1457278 0.2211401
##
## $vectors
##            [,1]       [,2]
## [1,] -0.9933553 -0.1150880
## [2,]  0.1150880 -0.9933553
```

Thus we conclude that the value $\hat{\theta}$ found is a local maximum.

We have thus found that the most likely model is the Wei$(\hat{\alpha}, \hat{\lambda})$, where $\hat{\alpha} = 2.76$ and $\hat{\lambda} = 17.56$. Let's plot this most likely model and let's compute some summaries from it.

```
> plot(function(x) dweibull(x, shape = oo$par[1],
+                           scale = oo$par[2]),
+      xlim=c(0.01, 45), lwd=2,
+      ylab="Density", xlab="waiting time",
+      main= "Estimated probability distribution for the waiting times")
> points(x=yobs,y=rep(0,length(yobs)), pch="x")
```



For instance the estimated average weiting time is

```
> oo$par[2]*gamma(1+1/oo$par[1])
```

```
## [1] 15.62419
```

which is very close but not exactly equal to the empirical average

```
> mean(yobs)
```

```
## [1] 15.59
```

The probability of waiting more than 30 minutes is

```
> 1-pweibull(30, shape = oo$par[1], scale = oo$par[2])
```

```
## [1] 0.0125161
```

Whereas the estimated probability of waiting between 5 and 15 minutes is

```
> pweibull(15, shape = oo$par[1], scale = oo$par[2])-
+   pweibull(5, shape = oo$par[1], scale = oo$par[2])
```

```
## [1] 0.4460469
```

Exercise

Assume now that $X_i \overset{iid}{\sim} \text{Ga}(\alpha, \beta)$, $i = 1, \ldots, n$. Repeat all the computation performed in Section 2.