Example of MLE Computations, using R

First of all, do you really need R to compute the MLE? Please note that MLE in many cases have explicit formula. Second of all, for some common distributions even though there are no explicit formula, there are standard (existing) routines that can compute MLE. Example of this catergory include Weibull distribution with both scale and shape parameters, logistic regression, etc. If you still cannot find anything usable then the following notes may be useful.

We start with a simple example so that we can cross check the result. Suppose the observations $X_1, X_2, ..., X_n$ are from $N(\mu, \sigma^2)$ distribution (2 parameters: μ and σ^2).

The log likelihood function is

$$\sum -\frac{(X_i - \mu)^2}{2\sigma^2} - 1/2 \log 2\pi - 1/2 \log \sigma^2 + \log dX_i$$

(actually we do not have to keep the terms $-1/2 \log 2\pi$ and $\log dX_i$ since they are constants.

In R software we first store the data in a vector called `xvec`

```
xvec <- c(2,5,3,7,-3,-2,0)      # or some other numbers
```

then define a function (which is negative of the log lik)

```
fn <- function(theta) {
sum ( 0.5*(xvec - theta[1])^2/theta[2] + 0.5* log(theta[2]) )
}
```

where there are two parameters: `theta[1]` and `theta[2]`. They are components of a vector `theta`. then we try to find the max (actually the min of negative log lik)

```
nlm(fn, theta <- c(0,1), hessian=TRUE)
```

or

```
optim(theta <- c(0,1), fn, hessian=TRUE)
```

You may need to try several starting values (here we used `c(0,1)`) for the `theta`. (i.e. `theta[1]=0, theta[2]=1.`)

Actual R output session:

```
> xvec <- c(2,5,3,7,-3,-2,0)                 # you may try other values
> fn                                         # I have pre-defined fn
function(theta) {
sum( 0.5*(xvec-theta[1])^2/theta[2] + 0.5* log(theta[2]) )
}
> nlm(fn, theta <- c(0,2), hessian=TRUE)         # minimization
$minimum
[1] 12.00132

$estimate
[1]  1.714284 11.346933

$gradient
[1] -3.709628e-07 -5.166134e-09

$hessian
              [,1]          [,2]
[1,]  6.169069e-01 -4.566031e-06
[2,] -4.566031e-06  2.717301e-02

$code
[1] 1

$iterations
[1] 12

> mean(xvec)
[1] 1.714286                               # this checks out with estimate[1]
> sum( (xvec -mean(xvec))^2 )/7
[1] 11.34694                               # this also checks out w/ estimate[2]
> output1 <- nlm(fn, theta <- c(2,10), hessian=TRUE)
> solve(output1$hessian)                   # to compute the inverse of hessian
                                           # which is the approx. var-cor matrix
              [,1]         [,2]
[1,] 1.6209919201 3.028906e-04
[2,] 0.0003028906 3.680137e+01
> sqrt( diag(solve(output1$hessian)) )
[1] 1.273182 6.066413
> 11.34694/7
[1] 1.620991
> sqrt(11.34694/7)
[1] 1.273182               # st. dev. of mean checks out

> optim( theta <- c(2,9), fn, hessian=TRUE)    # minimization, diff R function
$par
[1]  1.713956 11.347966

$value
[1] 12.00132

$counts
function gradient
      45       NA

$convergence
[1] 0

$message
NULL

$hessian
              [,1]         [,2]
[1,] 6.168506e-01 1.793543e-05
[2,] 1.793543e-05 2.717398e-02
```

Comment: We know long ago the variance of $\bar{x}$ can be estimated by s^2/n. (or replace s^2 by the MLE of σ^2) (may be even this is news to you? then you need to review some basic stat).

But how many of you know (or remember) the variance/standard deviation of the MLE of σ^2 (or s^2)? (by above calculation we know its standard deviation is approx. equal to 6.066413)

How about the covariance between $\bar{x}$ and v? here it is approx. 0.0003028 (very small). Theory say they are independent, so the true covariance should equal to 0.

Example of inverting the (Wilks) likelihood ratio test to get confidence interval

Suppose independent observations $X_1, X_2, ..., X_n$ are from $N(\mu, \sigma^2)$ distribution (one parameter: σ). μ assumed known, for example $\mu = 2$.

The log likelihood function is

$$\sum -\frac{(X_i - \mu)^2}{2\sigma^2} - 1/2 \log 2\pi - 1/2 \log \sigma^2 + \log dX_i$$

We know the log likelihood function is maximized when

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{n}}$$

This is the MLE of σ.

The Wilks statistics is

$$-2 \log \frac{\max_{H_0} lik}{\max lik} = 2[\log \max Lik - \log \max_{H_0} Lik]$$

In R software we first store the data in a vector called xvec

```
xvec <- c(2,5,3,7,-3,-2,0)      # or some other numbers
```

then define a function (which is negative of log lik) (and omit some constants)

```
fn <- function(theta) {
sum ( 0.5*(xvec - theta[1])^2/theta[2] + 0.5* log(theta[2]) )
}
```

In R we can compute the Wilks statistics for testing $H_0 : \sigma = 1.5$ vs $H_a : \sigma \neq 1.5$ as follows:

assume we know $\mu = 2$ then the MLE of σ is

```
  mleSigma <- sqrt( sum( (xvec - 2)^2 ) /length(xvec))
```

The Wilks statistics is

```
WilksStat <- 2*( fn(c(2,1.5^2)) - fn(c(2,mleSigma^2))  )
```

The actual R session:

```
> xvec <- c(2,5,3,7,-3,-2,0)
> fn
function(theta) {
sum ( 0.5*(xvec-theta[1])^2/theta[2] + 0.5* log(theta[2]) )
}
> mleSigma <- sqrt((sum((xvec - 2)^2))/length(xvec))
> mleSigma
[1] 3.380617
> 2*( fn(c(2,1.5^2)) - fn(c(2,mleSigma^2))  )
[1] 17.17925
```

This is much larger then 3.84 ($= 5\%$ significance of a chi-square distribution), so we should reject the hypothesis of $\sigma = 1.5$.

After some trial and error we find

```
> 2*( fn(c(2,2.1635^2)) - fn(c(2,mleSigma^2))  )
[1] 3.842709
> 2*( fn(c(2,6.37^2)) - fn(c(2,mleSigma^2))  )
[1] 3.841142
```

So the 95% confidence interval for σ is (approximately)

[2.1635, 6.37]

We also see that the 95% confidence Interval for σ^2 is

$$[2.1635^2, 6.37^2]$$

sort of invariance property (for the confidence interval).

We point out that the confidence interval from the Wald construction do not have invariance property.

The Wald 95% confidence interval for sigma is (using formula we derived in the midterm exam)

```
   3.380617 +- 1.96*3.380617/sqrt(2*length(xvec))

=  [1.609742, 5.151492]
```

The Wald 95% confidence interval for σ^2 is (homework)

```
   (3.380617)^2 +- 1.96* ...
```

Define a function (the log lik of the multinomial distribution)

```
> loglik <- function(x, p) { sum( x * log(p) ) }
```

For the vector of observation x (integers) and probability proportion p (add up to one)

We know the MLE of the p is just x/N where N is the total number of trials $= sumx_i$.

Therefore the $-2[\log lik(H_0) - \log lik(H_0 + H_a)]$ is

```
> -2*(\loglik(c(3,5,8), c(0.2,0.3,0.5))-loglik(c(3,5,8),c(3/16,5/16,8/16)))
[1] 0.02098882
>
```

This is not significant (not larger then 5.99) The cut off values are obtained as follows:

```
> qchisq(0.95, df=1)
[1] 3.841459
> qchisq(0.95, df=2)
[1] 5.991465

> -2*(loglik(c(3,5,8),c(0.1,0.8,0.1))-lik(c(3,5,8),c(3/16,5/16,8/16)))
[1] 20.12259
```

This is significant, since it is larger then 5.99.

Now use Pearson's chi square:

```
> chisq.test(x=c(3,5,8), p= c(0.2,0.3,0.5))

        Chi-squared test for given probabilities

data:  c(3, 5, 8)
X-squared = 0.0208, df = 2, p-value = 0.9896

Warning message:
Chi-squared approximation may be incorrect in: chisq.test(x = c(3, 5, 8), p = c(
0.2, 0.3, 0.5))
```

```
> chisq.test(x=c( 3,5,8), p= c(0.1,0.8,0.1))

        Chi-squared test for given probabilities

data:  c(3, 5, 8)
X-squared = 31.5781, df = 2, p-value = 1.390e-07

Warning message:
Chi-squared approximation may be incorrect in: chisq.test(x = c(3, 5, 8), p = c(
0.1, 0.8, 0.1))
```

1 t-test and approximate Wilks test

Use the same function we defined before but now we always plug-in the MLE for the (nuisance parameter) σ^2. As for the mean μ, we plug the MLE for one and plug the value specified in H_0 in the other (numerator).

```
> xvec <- c(2,5,3,7,-3,-2,0)
> t.test(xvec, mu=1.2)

        One Sample t-test

data:  xvec
t = 0.374, df = 6, p-value = 0.7213
alternative hypothesis: true mean is not equal to 1.2
95 percent confidence interval:
 -1.650691  5.079262
sample estimates:
mean of x
 1.714286
```

Now use Wilks likelihood ratio:

```
> mleSigma <- sqrt((sum((xvec - mean(xvec) )^2))/length(xvec))
> mleSigma2 <- sqrt((sum((xvec - 1.2)^2))/length(xvec))
> 2*( fn(c(1.2,mleSigma2^2)) - fn(c(mean(xvec),mleSigma^2))  )
[1] 0.1612929
```

```
> pchisq(0.1612929, df=1)
[1] 0.3120310
```

P-value is 0.312031