Worksheet: Estimating parameters from data using likelihoods

Nathan Green, Imperial College London

14/01/2020

1 Introduction

This practical will use the statistical programming language R. Make sure you have it installed on your system before you start.

Follow along the code below and then answer the questions.

2 Maximum likelihood estimation

Recall from the lecture the log-likelihood plots show the MLE at the mode. We will recreate the analysis in the lectures and experiment with different models and data.

2.1 Logistic regression MLE

First, assume that the data are generated from a Binomial distribution $X \sim Bin(n, p)$. We specify the parameter values of the distribution using the assignment operator `<<-`. This binds the value on the right-hand side to the variable name on the left-hand side.

```
x <<- 8
n <<- 12
```

Note that we have use the global assignment operator `<<-` to ensure that these are the values used throughout. If we only want the binding to be local, i.e. in the same scope as the statement, then we can use `<-`.

Define the associated likelihood function

$$l = x \log(p) + (n - x) \log(1 - p)$$

We do this by creating function. This takes the form of

```
<name_of_function> <- function(<arguments>) {
  <function body>
}
```

So the above equation would be

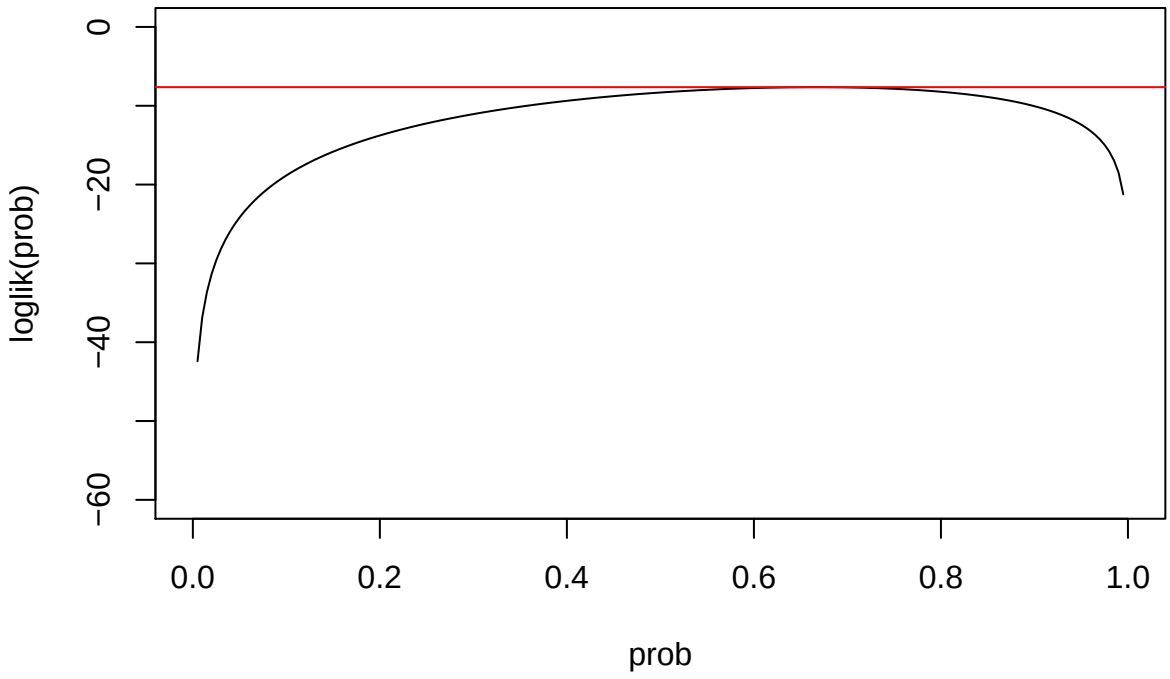
```
lik <- function(p) p^x * (1-p)^(n-x)

loglik <- function(p)
  log(lik(p))
```

When the function body is only a single line we don't need to use the curly parentheses `{}`. Note that we have used our first function `lik()` inside our second function `loglik()`. Also we could have alternatively used the in-built base function `dbin()`. Now let's plot the log-likelihood to see where the MLE is. We first define a sequence of x-axis values (`prob`) at which to calculate the log-likelihood. Because these are probailities we define a sequence from 0 to 1 by steps of 0.005.

```
prob <- seq(0, 1, by = 0.005)
```

We can now plot probabilities against log-likelihood. We select a line plot, `l`, and restrict the y-axis range to between -60 and 0.

```
plot(x = prob, y = loglik(prob), type = "l", ylim = c(-60, 0))
abline(h = loglik(8/12), col = "red")
```



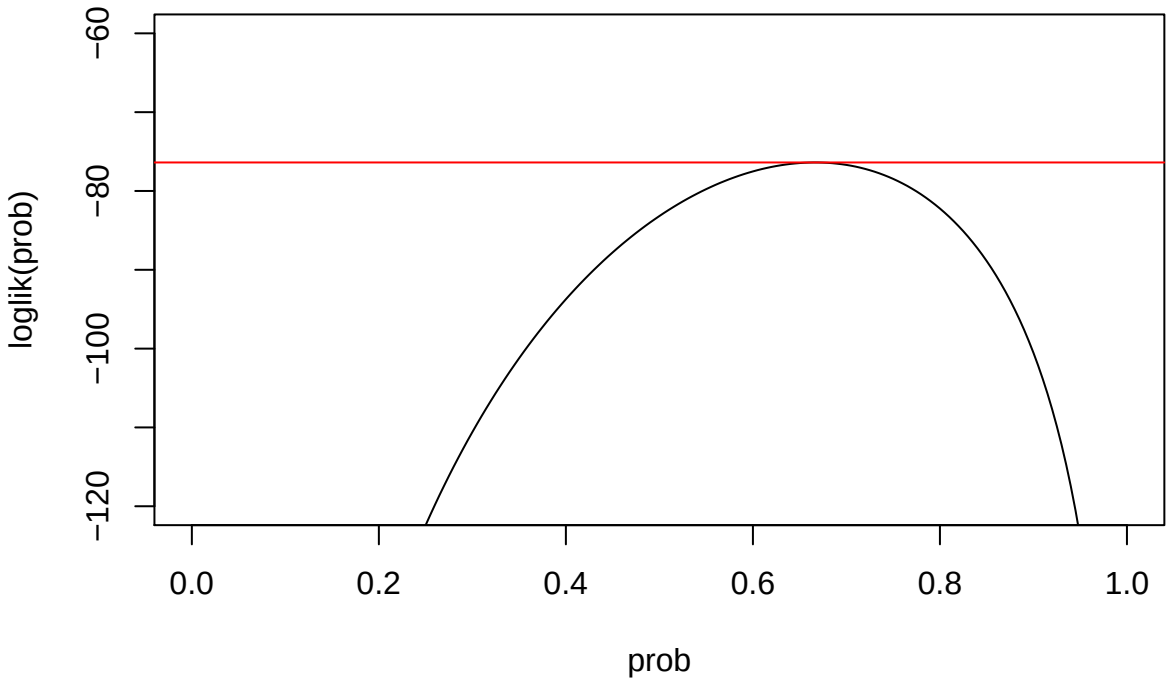
The red horizontal line indicates the location of the MLE $8/12 = 0.667$.

Next, let's increase the sample size whilst keeping the same proportion of observed successes.

```
x <<- 80
n <<- 120
```

Plotting the log-likelihood again for these data gives

```
plot(prob, loglik(prob), type = "l", ylim = c(-120, -60))
abline(h = loglik(80/120), col = "red")
```



We can see that the the mode of the curve is in the same place as previously but the spread of the curve is narrower for the larger sample size. We would expect this since there is more information in more data and so the degree of certainty about the true value is increase represented by greater curvature.

To determine the MLE we find the largest value in the vector of log-likelihood values and return the corresponding probability.

In mathematical notation we would write this as

$$\hat{p} = \text{argmax}_p\{loglik(p) : 0 \leq p \leq 1\}$$

In R, we can find the maximum log-likelihood value and check where in the vector of loglikelihood this is. We can do this by checking each value and seeing if it's equal to the maximum and returning `FALSE` if not and `TRUE` if it is the maximum. The `TRUE` entry is at the 134th position.

```
which_max <- loglik(prob) >= max(loglik(prob))
which_max
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [34] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [45] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [56] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [67] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [78] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [89] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## [100] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [111] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [122] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [144] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [155] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [166] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [177] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [188] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [199] FALSE FALSE FALSE
```

We can now ask R to return the log-likelihood and probability at this index.

```
loglik(prob)[which_max]
```

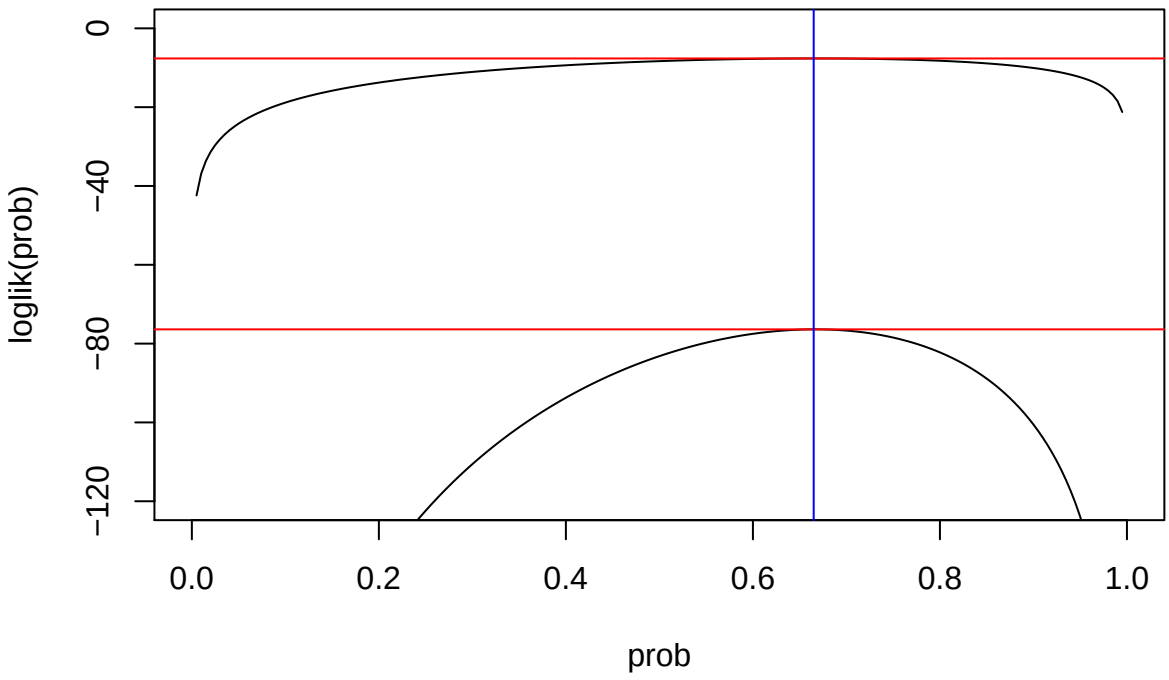
```
## [1] -76.38245
```

```
MLE <- prob[which_max]
MLE
```

```
## [1] 0.665
```

We can see this by plotting both log-likelihood together. The vertical blue line indicates the value of the MLE $\hat{p} = 0.665$.

```
x <<- 8; n <<- 12
plot(prob, loglik(prob), type = "l", ylim = c(-120, 0))
abline(h = loglik(8/12), col = "red")
x <<- 80; n <<- 120
lines(prob, loglik(prob), type = "l")
abline(h = loglik(80/120), col = "red")
abline(v = MLE, col = "blue")
```



Notice here that in `abline()` the argument `h` is for a horizontal line and `v` is for a vertical line.

2.2 Poisson model MLE

We shall duplicate the above analysis for data generated from a Poisson model, $X \sim Poisson(\mu)$. the log-likelihood if given by

$$l = \log(\mu) \sum x - n\mu - \sum \log(x!)$$

In the example in the lecture there were 20 cities surveyed, and 134 infected people were recorded.

We could create a vector of the number of counts for each city x. Using the `rep()` function we can replicate a number `times` times.

```
x <<- rep(134/20, times = 20)
n <<- length(x)

x
```

```
##  [1] 6.7 6.7 6.7 6.7 6.7 6.7 6.7 6.7 6.7 6.7 6.7 6.7 6.7 6.7 6.7 6.7 6.7 6.7
## [18] 6.7 6.7 6.7
```

```
sum(x)
```

```
## [1] 134
```

However, the sum total of x is sufficient to calculate the log-likelihood so this is not necessary. As for the Binomial case, we define the log-likelihood function. Note that we can ignore the constant term that doesn't depend on μ.

```
loglik_count <- function(mu)
  sum(x)*log(mu) - n*mu #- log(prod(map_dbl(x, factorial)))

ll_max <- loglik_count(134/20)
counts <- seq(0, 14, 0.1)
```

As before, let's find the index of the maximum log-likelihood and the MLE.

```
which_max <- loglik_count(counts) >= max(loglik_count(counts))

loglik_count(counts)[which_max]
```

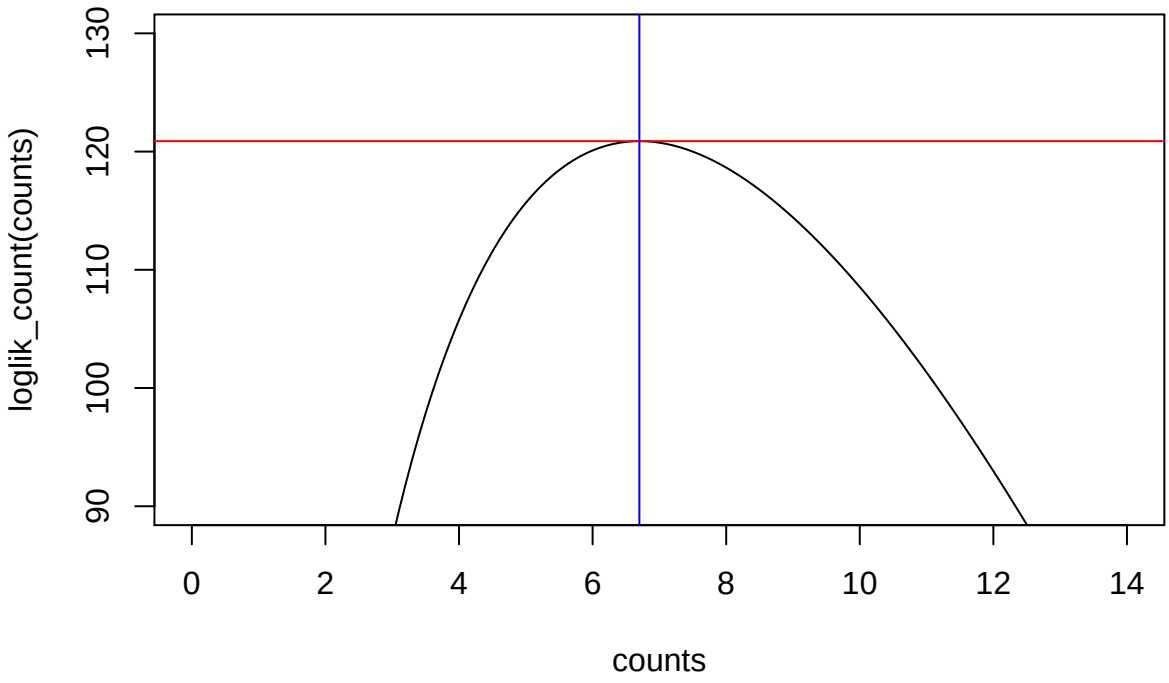
```
## [1] 120.8824
```

```
MLE <- counts[which_max]
MLE
```

```
## [1] 6.7
```

Plotting this model confirms this.

```
plot(counts, loglik_count(counts), type = "l", ylim = c(90,130))
abline(h = loglik_count(134/20), col = "red")
abline(v = MLE, col = "blue")
```



3 Likelihood-based confidence intervals

Further to finding the MLE point value we are interested in the uncertainty about this estimate represented as a confidence interval (CI). In order to do this we will write a function. Recall from the lecture that the 95% CI contains all those parameter values with log likelihood values within $\chi^2_p/2$ of the maximum log likelihood (where p is the number of parameters being estimated). So we will take a sequence of possible parameter values and find out which of these are within this interval. We do this with a `for` loop which iterate from the smallest to the largest value checking each time. When the lol-likelihood of a particular parameter value is within $\chi^2_p/2$ of the MLE then we record `TRUE`, otherwise `FALSE`. Just for convenience the function also print the upper and lowe values of the interval.

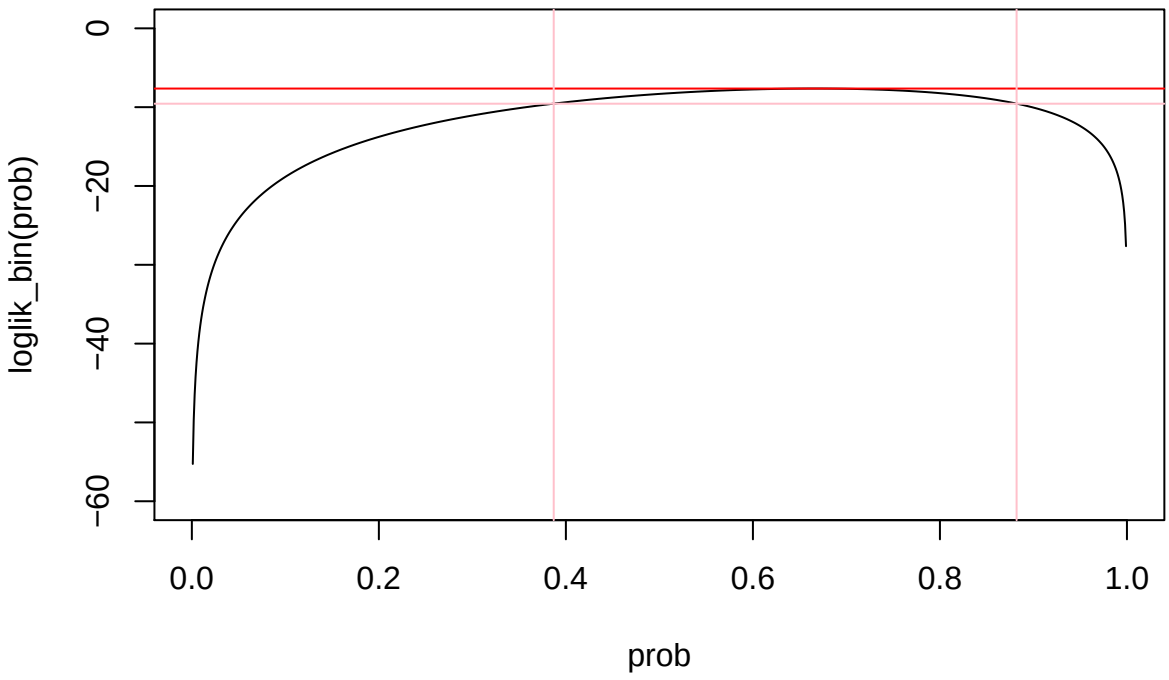
```
find_CI <- function(n,
                    x,
                    loglik,
                    param = seq(0, 1, by = 0.001)) {

  n <<- n
  x <<- x
  mle <- x/n
  ll_max <- loglik(mle)
  ll95 <- ll_max - qchisq(0.95, 1)/2

  interval <- rep(FALSE, length(param))
```

```
  for (i in seq_along(param)) {
    if (loglik(param[i]) > ll95)
      interval[i] <- TRUE
  }

  print(
    sprintf("95%% CI is (%.3f, %.3f).",
            min(param[interval]), max(param[interval])))

  invisible(c(min(param[interval]),
              max(param[interval])))
}
```

3.1 Binomial examples

Repeating the Binomial example above, we can add horizontal line associated with the 95% CI to the log-likelihood plot. With the function `find_CI()` we can now estimated CIs for the models. Vertical lines are added to the lol-likelihood plot indicating the CI upper and lower limits.

```
x <<- 8
n <<- 12

lik <- function(p) p^x * (1-p)^(n-x)
loglik_bin <- function(p) log(lik(p))

mle <- 8/12
ll_max <- loglik_bin(mle)
ll95 <- ll_max - qchisq(0.95, 1)/2
prob <- seq(0, 1, by = 0.001)

plot(prob, loglik_bin(prob), type = "l", ylim = c(-60, 0))
abline(h = ll_max, col = "red")
abline(h = ll95, col = "pink")
abline(v = find_CI(12, 8, loglik_bin), col = "pink")
```




```
## [1] "95% CI is (0.387, 0.882)."
```

The estimated values for the small sample are

```
find_CI(12, 8, loglik_bin)
```

```
## [1] "95% CI is (0.387, 0.882)."
```

and for the larger sample we see that the 95% CI is smaller

```
find_CI(120, 80, loglik_bin)
```

```
## [1] "95% CI is (0.580, 0.746)."
```

3.2 Poisson example

Lastly, find the 95% CI for the Poisson example were 20 cities surveyed, and 134 infected people were recorded.

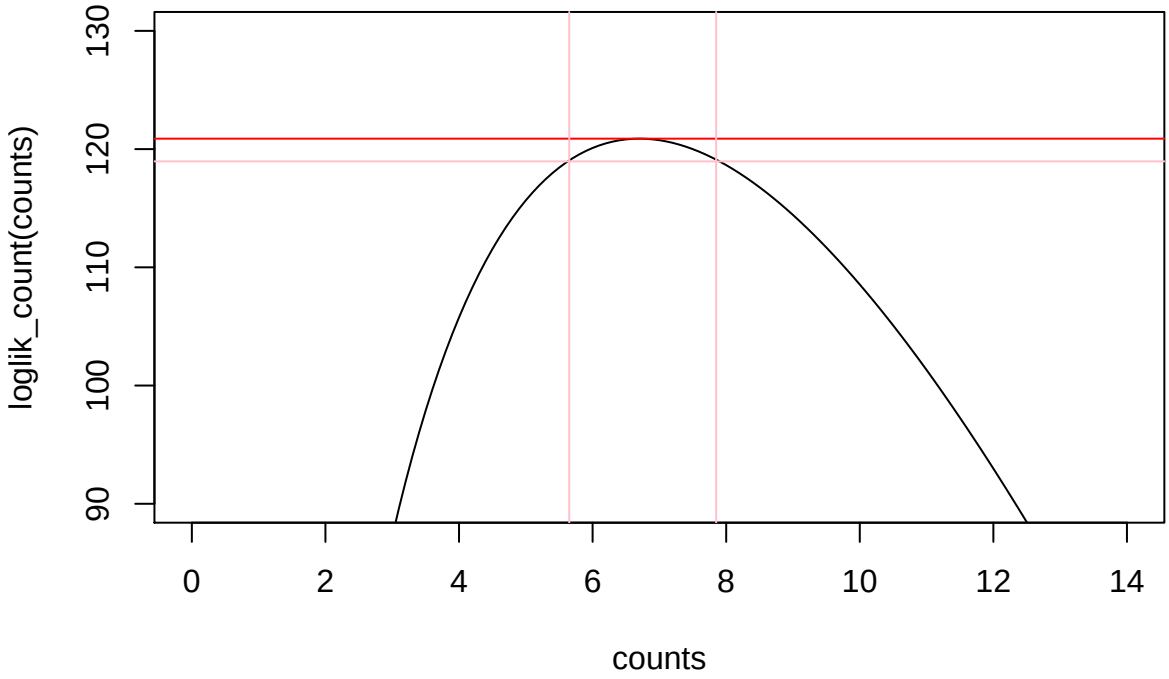
```
loglik_count <- function(mu)
  sum(x)*log(mu) - n*mu


find_CI(20, 134, param = seq(0, 14, 0.05), loglik_count)
```

```
## [1] "95% CI is (5.650, 7.850)."
```

```
ll_max <- loglik_count(134/20)
ll95 <- ll_max - qchisq(0.95, 1)/2
```

The log-likelihood plot with the 95% CI upper and lower bounds calculated using `find_CI()` is given below.

```
counts <- seq(0, 14, 0.1)
plot(counts, loglik_count(counts), type = "l", ylim = c(90,130))
abline(h = loglik_count(134/20), col = "red")
abline(h = ll95, col = "pink")
abline(v = find_CI(20, 134, param = seq(0, 14, 0.05), loglik_count), col = "pink")
```




```
## [1] "95% CI is (5.650, 7.850)."
```

4 Further examples

Imagine that we obtain more data about the infection. Calculate and plot the log-likelihood, MLE and 95% CIs for the following data:

- 200 cities surveys and 1340 cases recorded.
- 400 cities surveys and 2680 cases recorded.
- 1000 cities surveys and 6700 cases recorded.

What do you find?

```
find_CI(200, 1340, param = seq(0, 14, 0.05), loglik_count)
```

```
## [1] "95% CI is (6.350, 7.050)."
```

```
find_CI(400, 2680, param = seq(0, 14, 0.05), loglik_count)
```

```
## [1] "95% CI is (6.450, 6.950)."
```

```
find_CI(1000, 6700, param = seq(0, 14, 0.05), loglik_count)
```

```
## [1] "95% CI is (6.550, 6.850)."
```

```
ll_max1 <- loglik_count(1340/200)
ll_max2 <- loglik_count(2680/400)
ll_max3 <- loglik_count(6700/1000)

ll951 <- ll_max1 - qchisq(0.95, 1)/2
ll952 <- ll_max2 - qchisq(0.95, 1)/2
ll953 <- ll_max3 - qchisq(0.95, 1)/2
```

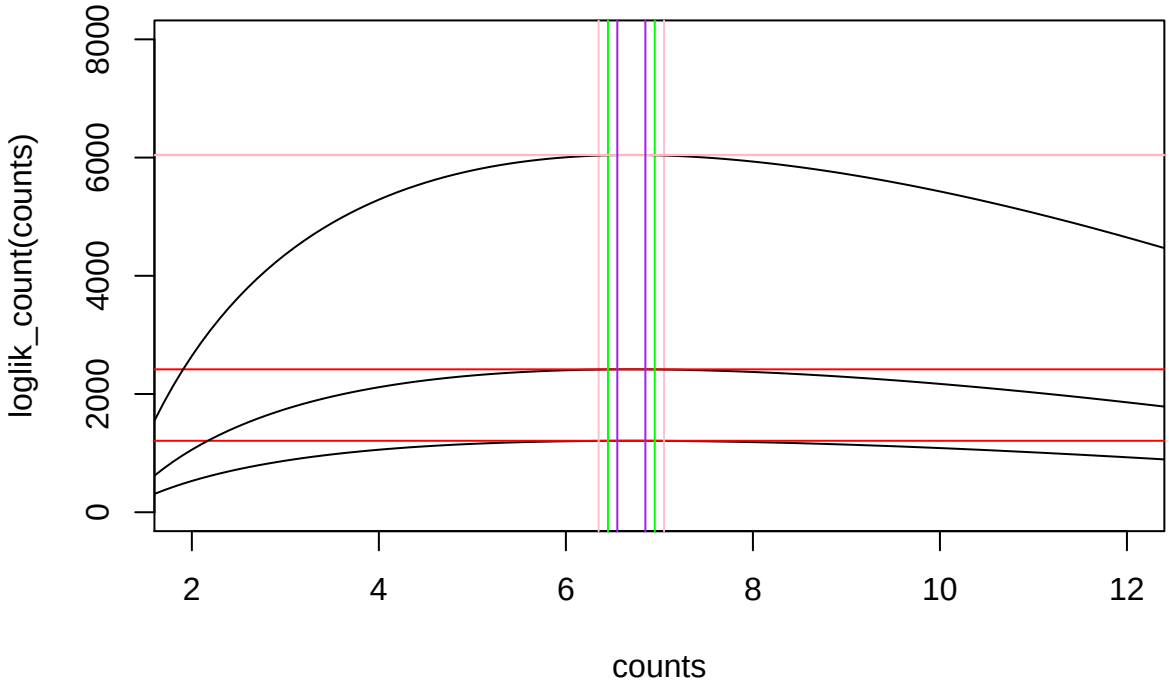
```
# The log-likelihood plot with the 95% CI upper and lower bounds calculated using `find_CI()` is given

x <<- 1340
n <<- 200

counts <- seq(0, 14, 0.1)
plot(counts, loglik_count(counts), type = "l", ylim = c(0,8000), xlim = c(2,12))
abline(h = loglik_count(x/n), col = "red")
abline(h = ll951, col = "pink")
abline(v = find_CI(200, 1340, param = seq(0, 14, 0.05), loglik_count), col = "pink")
```

```
## [1] "95% CI is (6.350, 7.050)."
```

```
x <<- 2680
n <<- 400

lines(counts, loglik_count(counts), type = "l")
abline(h = loglik_count(x/n), col = "red")
abline(h = ll952, col = "pink")
abline(v = find_CI(n, x, param = seq(0, 14, 0.05), loglik_count), col = "green")
```

```
## [1] "95% CI is (6.450, 6.950)."
```

```
x <<- 6700
n <<- 1000

lines(counts, loglik_count(counts), type = "l")
abline(h = loglik_count(x/n), col = "red")
abline(h = ll953, col = "pink")
abline(v = find_CI(n, x, param = seq(0, 14, 0.05), loglik_count), col = "purple")
```




```
## [1] "95% CI is (6.550, 6.850)."
```

```
library(dplyr)
library(knitr)

# we can collect the results in to a single table
# the `paste()` function combines the upper and lower CI limits into a single column entry
# the `kable()` function in the `knitr` package formats the data in to a table with borders.
kable(data.frame(Cities = c(200, 400, 1000),
                 Cases  = c(1340, 2680, 6700)) %>%
        mutate(MLE = Cases/Cities,
               CI95 =
                 c(paste(find_CI(200, 1340, param = seq(0, 14, 0.05), loglik_count),
                         collapse = ", "),
                   paste(find_CI(400, 2680, param = seq(0, 14, 0.05), loglik_count),
                         collapse = ", "),
                   paste(find_CI(1000, 6700, param = seq(0, 14, 0.05), loglik_count),
                         collapse = ", "))))
```

```
## [1] "95% CI is (6.350, 7.050)."
## [1] "95% CI is (6.450, 6.950)."
## [1] "95% CI is (6.550, 6.850)."
```

Cities	Cases	MLE	CI95
200	1340	6.7	6.35, 7.05
400	2680	6.7	6.45, 6.95
1000	6700	6.7	6.55, 6.85