SOME EXERCISES ABOUT LIKELIHOOD

Henrique Ap. Laureano
henriquelaureano@outlook.com ∧ www.leg.ufpr.br/~henrique/
/UFPR/DEST/LEG/

June & July, 2019

Contents

1 Uniform distribution

Let X be a r.v. with uniform distribution $X \sim U(0, \theta)$.
A random sample gave the following values: 5.5; 3.2; 4, 8, 5.3; 3.8 and 5.0. Obtain the likelihood function and one or more appropriate interval(s) specifying the form of obtaining.

<r code>

```
data <- c(5.5, 3.2, 4.8, 5.3, 3.8, 5)
```

The likelihood of a Uniform$(0, \theta)$ is

$$\begin{aligned} L(\theta) &= \theta^{-n}, \quad \text{for } x_i < \theta \text{ for all } i \\ &= \theta^{-n}, \quad \text{for } \theta > x_{(n)}, \end{aligned}$$

and equal to zero otherwise.

Given the data above, we get $x_{(n)} = 5.5$ and the likelihood is shown in Figure 1, together with some intervals.

<r code>

```
lkl.unif <- function(theta, data) {
    n <- length(data) ; xn <- max(data)
    # this next step, in the way presented, isn't very efficient, since I'll
    # compute the likelihood for all theta's and then, if necessary, convert to
    # zero. however, since we have here a very simple likelihood (and this
    # notation is extremely clear, btw), I keep in this way
    lkl = theta**(-n)*{theta >= xn}
    return(lkl)
}
theta.seq <- seq(4, 12, length.out = 100)
lklseq.unif <- lkl.unif(theta.seq, data)

par(mfrow = c(1, 2)) # plot window definition (one row, two columns)
# plotting the normalized likelihood function to have unit maximum
plot(theta.seq, lklseq.unif/max(lklseq.unif), type = "l",
     xlab = expression(theta), ylab = "Likelihood")
# probability-based interval ------------------------------------------------
# cutoff's corresponding to a 95\% and 99\% confidence interval for the mean
cuts <- c(.15, .04) * lkl.unif(max(data), data) / max(lklseq.unif)
abline(h = cuts, lty = 2:3)
# uniroot.all finds the zeros, so we need to subtract the cutoff point from
# the likelihood to be able to find the points where the likelihood is cut
ic.lkl.unif <- function(theta, cut, ...) {
    lkl.unif(theta, ...)/max(lklseq.unif) - cut
}
ic.95.prob <- rootSolve::uniroot.all(ic.lkl.unif, range(theta.seq),
```

```
                                    data = data, cut = cuts[1])
ic.99.prob <- rootSolve::uniroot.all(ic.lkl.unif, range(theta.seq),
                                    data = data, cut = cuts[2])
abline(v = ic.95.prob, lty = 2) ; abline(v = ic.99.prob, lty = 3)

plot(theta.seq, lklseq.unif/max(lklseq.unif), type = "l",
     xlab = expression(theta), ylab = "Likelihood")
# pure likelihood interval ------------------------------------------------------
# cutoff's corresponding to a 95\% and 99\% confidence interval for the mean
cuts <- c(.05, .01) * lkl.unif(max(data), data) / max(lklseq.unif)
abline(h = cuts, lty = 2:3)

ic.95.pure <- rootSolve::uniroot.all(ic.lkl.unif, range(theta.seq),
                                    data = data, cut = cuts[1])
ic.99.pure <- rootSolve::uniroot.all(ic.lkl.unif, range(theta.seq),
                                    data = data, cut = cuts[2])
abline(v = ic.95.pure, lty = 2) ; abline(v = ic.99.pure, lty = 3)
```

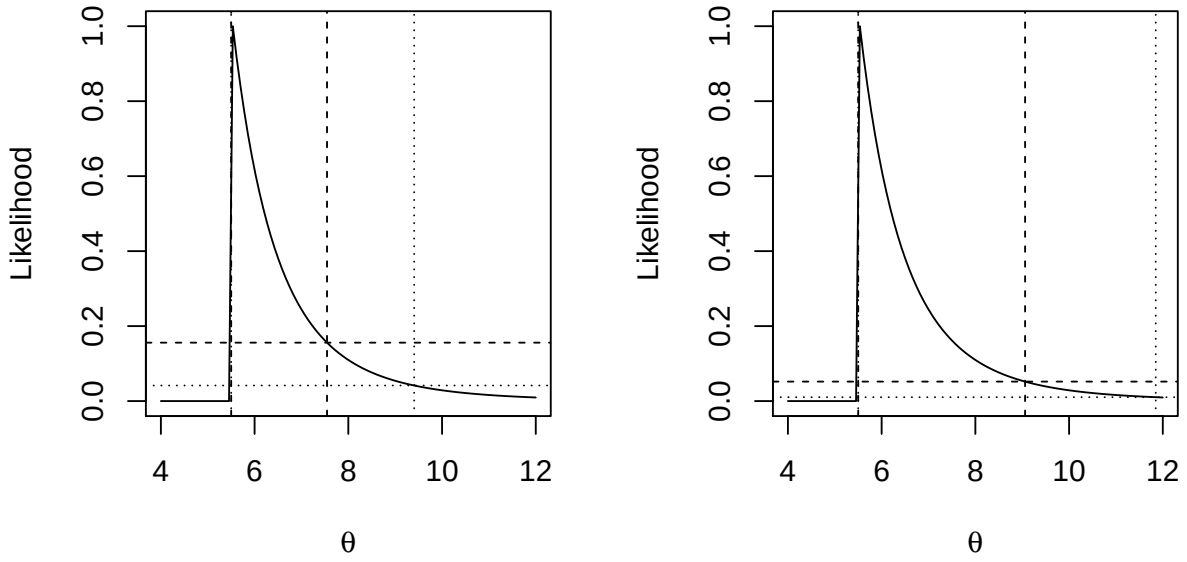

Figure 1: Likelihood function of θ in Uniform(0, θ) based on $x_{(6)} = 5.5$. In the left, likelihood intervals at 15% (dashed line) and 4% (dotted line) cutoff. In the right, likelihood intervals at 5% (dashed line) and 1% (dotted line) cutoff. These cutoffs correspond, in both sides, to a 95% and 99% confidence interval, respectively.

In both graphs of Figure 1 we have two confidence intervals for θ, one corresponding to a 95% confidence interval (dashed line) and one corresponding to a 99% confidence interval for the mean (dotted line). In the left we have a probability-based likelihood interval, in the right we have a pure likelihood interval.

The left ones are bad intervals, since they're based in a large-sample theory that results in an exact interval for the Gaussian case, in a good approximation for a reasonably regular case, and as we can see by the figure... here we doesn't have a regular likelihood (the likelihood isn't well approximated by a quadratic function), thus, here this interval should not be so good.

Explaining better how we arrived in these intervals:

For a normalized likelihood, $L(\theta)/L(\hat{\theta})$, we have the following Wilk's likelihood ratio statistic, defined as W

$$W \equiv 2\log\frac{L(\hat{\theta})}{L(\theta)} \sim \chi^2_1.$$

Its χ^2 distribution is exact only in the normal mean model, and approximately true when the likelihood is reasonably regular.

Based in this Wilk's statistic we are able to find the probability that the likelihood interval covers θ,

$$\Pr\left\{\frac{L(\theta)}{L(\hat{\theta})} > c\right\} = \Pr\left\{2\log\frac{L(\theta)}{L(\hat{\theta})} < -2\log c\right\} = \Pr\{\chi^2_1 < -2\log c\}.$$

So, if for some $0 < \alpha < 1$ we choose a cutoff

$$c = e^{-\frac{1}{2}\chi^2_{1,(1-\alpha)}},$$

where $\chi^2_{1,(1-\alpha)}$ is the $100(1-\alpha)$ percentile of χ^2_1.
For $\alpha = 0.05$ and 0.01 we have a cutoff $c = 0.15$ and 0.04, that corresponds to a 95% and 99% confidence interval, respectively. □

Having the cutoff values we need now to find the interval values, i.e., in which points the cutoff horizontal line cuts the likelihood.
For this purpose we use the function `rootSolve::uniroot.all`.

A likelihood interval at 15% and 4% cutoff for θ are (5.5, 7.545) and (5.5, 9.405).

Already on the right side of Figure 1 we have a more coherent, let's say, confidence interval for the Uniform likelihood. While the likelihood isn't regular, it is still possible to provide an exact theoretical justification for a confidence interval interpretation. Now

$$\Pr\left\{\frac{L(\theta)}{L(\hat{\theta})} > c\right\} = \Pr\left\{\frac{X_{(n)}}{\theta} > c^{1/n}\right\} = 1 - \Pr\left\{\frac{X_{(n)}}{\theta} < c^{1/n}\right\} = 1 - (c^{1/n})^n = 1 - c.$$

So the likelihood interval with cutoff c is a $100(1-c)\%$ confidence interval.

A likelihood interval at 15% and 4% cutoff for θ are (5.5, 9.062) and (5.5, 11.849).

Comparing the obtained intervals we see a broader range with the pure likelihood intervals. In other words, with the pure likelihood intervals we see a higher uncertainty than with the (not so recommended here) probability-based likelihood intervals.

For doing this exercise I read and used it Pawitan's book:

```
@book{pawitan,
  author    = {Yudi Pawitan},
  title     = {In All Likelihood:
               Statistical Modelling and Inference Using Likelihood},
  year      = {1991},
  publisher = {Oxford University Press},
  address   = {Great Clarendon Street, Oxford OX2 6DP},
}
```

2 Poisson distribution

Let X be a r.v. of a Poisson distribution ($X \sim P(\lambda)$) for which the following random sample was obtained: (3, 1, 0, 2, 1, 1, 0, 0).

<r code>

```
data <- c(3, 1, 0, 2, 1, 1, 0, 0)
```

(a) **Obtain the likelihood function, its quadratic approximation and intervals for λ.**

The likelihood and the log-likelihood of a Poisson(λ) is

$$L(\lambda) = \prod_{i=1}^{n} \frac{e^{-\lambda}\lambda^{x_i}}{x_i!} \quad \text{and} \quad \log L(\lambda) = l(\lambda) = -n\lambda + \log\lambda \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \log x_i!.$$

We work here with the log-likelihood to make our life easier, since the computations are much simpler in the log scale.

By computing the Score (derivative of $l(\lambda)$ wrt to λ) and making it equal to zero, we find the MLE $\hat{\lambda} = \bar{x}$. Computing the second derivative we find the observed information

$$I_O(\lambda) = \frac{n\bar{x}}{\lambda^2} \quad \rightarrow \quad I_O(\hat{\lambda}) = \frac{n}{\bar{x}}.$$

A graph of the Poisson log-likelihood is provided in black in Figure 2.

Doing a quadratic approximation (Taylor expansion of second order) in $l(\lambda)$ around $\hat{\lambda}$ we have

$$\begin{aligned} l(\lambda) &\approx l(\hat{\lambda}) + (\lambda - \hat{\lambda})l'(\hat{\lambda}) + \frac{1}{2}(\lambda - \hat{\lambda})^2 l''(\hat{\lambda}) \\ &\quad \text{(the Score is zero at the MLE)} \\ &= l(\hat{\lambda}) + \frac{1}{2}(\lambda - \hat{\lambda})^2 l''(\hat{\lambda}) = l(\hat{\lambda}) + \frac{1}{2}(\lambda - \hat{\lambda})^2 I_O(\hat{\lambda}). \end{aligned}$$

A graph of the quadratic approximation of the Poisson log-likelihood is provided in gray in Figure 2. There we can see how good is the approximation around the maximum likelihood estimator (MLE). Here the sample size is very small, the idea is that as the sample size increase the quality, in this case the range, of the approximation also increase. This should happen because as the sample size increase the Poisson likelihood should be more symmetric.

In the topright graph of Figure 2 we have two intervals for λ.

<r code>

```
# likelihood ------------------------------------------------------------------
lkl.poi <- function(lambda, data) {
    n <- length(data)
    # log-likelihood ignoring irrelevant constant terms
    lkl = -n * lambda + sum(data) * log(lambda)
    return(lkl)
}
lambda.seq <- seq(0, 4.5, length.out = 100)
lklseq.poi <- lkl.poi(lambda.seq, data)

layout(matrix(c(1, 3, 2, 3), 2, 2), heights = c(1.5, 1)) # plot window definition
plot(lambda.seq, lklseq.poi, type = "l",
     xlab = expression(lambda), ylab = "log-likelihood")
lambda.est <- mean(data) # MLE
abline(v = lambda.est, lty = 3)
# quadratic approximation -----------------------------------------------------
quadprox.poi <- function(lambda, lambda.est, data) {
    n <- length(data)
    obs.info <- n / lambda.est # observed information
    lkl.poi(lambda.est, data) - .5 * obs.info * (lambda - lambda.est)**2
}
curve(quadprox.poi(x, lambda.est, data), col = "darkgray", add = TRUE)
legend(2.4, -6.25, c("log-like", "Quadratic\napprox.", "MLE"),
       col = c(1, "darkgray", 1), lty = c(1, 1, 3), bty = "n")
# intervals for lambda --------------------------------------------------------
plot(lambda.seq, lklseq.poi, type = "l",
     xlab = expression(lambda), ylab = "log-likelihood")
curve(quadprox.poi(x, lambda.est, data), col = "darkgray", add = TRUE)
abline(v = lambda.est, lty = 3)
## probability-based interval -------------------------------------------------
# cutoff's corresponding to a 95\% confidence interval for the mean
cut <- log(.15) + lkl.poi(lambda.est, data) ; abline(h = cut, lty = 2)
ic.lkl.poi <- function(lambda, cut, ...) { lkl.poi(lambda, ...) - cut }
ic.95.prob <- rootSolve::uniroot.all(ic.lkl.poi, range(lambda.seq),
                                     data = data, cut = cut)
arrows(x0 = ic.95.prob, y0 = rep(cut, 2),
       x1 = ic.95.prob, y1 = rep(-25, 2), lty = 2, length = .1)
## wald confidence interval  --------------------------------------------------
```

```r
cut <- log(.15) + quadprox.poi(lambda.est, lambda.est, data)
abline(h = cut, lty = 2, col = "darkgray")
se.lambda.est <- sqrt(lambda.est / length(data))
wald.95 <- lambda.est + qnorm(c(.025, .975)) * se.lambda.est
arrows(x0 = wald.95, y0 = rep(cut, 2),
       x1 = wald.95, y1 = rep(-25, 2), lty = 2, length = .1, col = "darkgray")
# a better look in the approximation around the MLE ------------------------------
lambda.seq <- seq(.8, 1.2, length.out = 25)
plot(lambda.seq, lkl.poi(lambda.seq, data), type = "l",
     xlab = expression(lambda), ylab = "log-likelihood")
curve(quadprox.poi(x, lambda.est, data), col = "darkgray", add = TRUE)
abline(v = lambda.est, lty = 3)
```

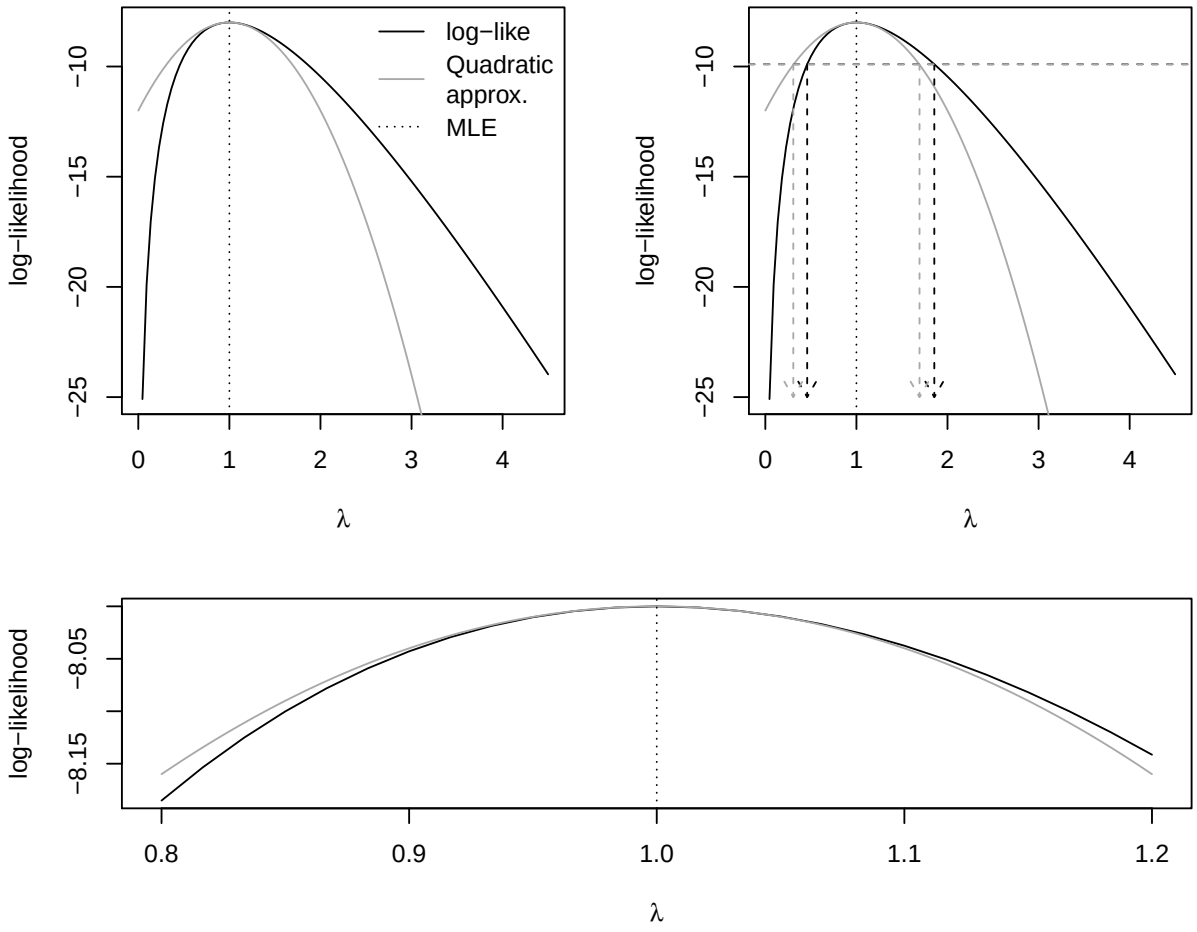

Figure 2: log-likelihood function and MLE of λ in Poisson(λ) based on `data`. In the topleft, a quadratic approximation in gray. In the topright, two intervals for λ - one based in the likelihood (in black) and one based in the quadratic approximation (in gray). In the bottom we provide a better look in the quadratic approximation.

In the topright, in black, we have a interval based in the likelihood, $\lambda \in (0.459, 1.855)$, but with a cutoff criterion based in a χ^2 distribution. Thus, to have a nominal 95% confidence interval we use a cutoff $c = 15\%$. As we saw before, this interval is based in a large-sample theory. Since we're dealing here with a reasonable regular case, this interval shows as a good approximation.

Still in the topright of Figure 2, in red we have a interval based in the quadratic approximation of the log-likelihood, $\lambda \in (0.307, 1.693)$. From the quadratic approximation we get

$$\log \frac{L(\lambda)}{L(\hat{\lambda})} \approx -\frac{1}{2} I_O(\hat{\lambda})(\lambda - \hat{\lambda})^2,$$

that also follows a χ^2 distribution, since we have a r.v. $\hat{\lambda}$ normalized (with its expected value subtracted and divided by its variance) and squared. From this we get the following exact (in the normal case) 95% confidence interval

$$\hat{\lambda} \pm 1.96\ I_O(\hat{\lambda})^{-1/2} \qquad (\hat{\lambda} \pm 1.96\ \mathrm{se}(\hat{\lambda})).$$

In the nonnormal cases this is an approximate 95% CI.

The actual variance is $I_E(\hat{\lambda})^{-1/2}$, but for the Poisson case the Fisher (expected) information is equal to the observed one, $I_O(\hat{\lambda})^{-1/2}$.

A very nice thing that we can see from this intervals is that a Wald interval (a $\pm$ interval) corresponds to a cut in the quadratic approximation exactly in the same point that the probability-based interval cuts the (log-)likelihood. Thus, as more regular the likelihood, better will be the fit of the approximation and more reliable will be the Wald interval.

(b) **Repeat the previous question for reparametrization $\theta = \log(\lambda)$.**

By the invariance property of the MLE we have

$$\hat{\lambda} = \bar{x} \quad \Rightarrow \quad g(\hat{\lambda}) = \log \hat{\lambda} = \hat{\theta} = \log \bar{x} = g(\bar{x}).$$

i.e., the MLE of $\hat{\theta}$ is $\log \bar{x}$.

By the Delta Method we compute the variance of $\hat{\theta}$,

$$V[\theta] = V[g(\lambda)] = \left[\frac{\partial}{\partial \lambda} g(\lambda)\right]^2 V[\lambda] = \left[\frac{1}{\lambda}\right]^2 \frac{\lambda}{n} = \frac{1}{\lambda\, n} \quad \rightarrow \quad V[\hat{\theta}] = \frac{1}{\bar{x}\, n}.$$

From this we can take the observed information for the reparametrization

$$V[\hat{\theta}] = I_O^{-1}(\hat{\theta}) = (\bar{x}\, n)^{-1}.$$

Now we do, in the same manner, everything that we did in the previous letter.

<r code>

```r
# likelihood ------------------------------------------------------------------
## we can still use the lkl.poi function, but now we do
## lambda = exp(theta),
## with theta being a real rate (positive or negative)
theta.seq <- seq(-2, 2, length.out = 100)
lklseq.poi <- lkl.poi(exp(theta.seq), data)

layout(matrix(c(1, 3, 2, 3), 2, 2), heights = c(1.5, 1)) # plot window definition
plot(theta.seq, lklseq.poi, type = "l",
     xlab = expression(theta), ylab = "log-likelihood")
theta.est <- log(mean(data)) # MLE
abline(v = theta.est, lty = 3)
# quadratic approximation for the reparametrization ---------------------------
quadprox.poi.repa <- function(theta, theta.est, data) {
    obs.info <- sum(data) # observed information
    lkl.poi(exp(theta.est), data) - .5 * obs.info * (theta - theta.est)**2
}
curve(quadprox.poi.repa(x, theta.est, data), col = "darkgray", add = TRUE)

legend(-2.15, -27.5, c("log-like", "Quadratic\napprox.", "MLE"),
       col = c(1, "darkgray", 1), lty = c(1, 1, 3), bty = "n")
# intervals for lambda --------------------------------------------------------
plot(theta.seq, lklseq.poi, type = "l",
     xlab = expression(theta), ylab = "log-likelihood")
curve(quadprox.poi.repa(x, theta.est, data), col = "darkgray", add = TRUE)
abline(v = theta.est, lty = 3)
## probability-based interval --------------------------------------------------
# cutoff's corresponding to a 95\% confidence interval for the mean
cut <- log(.15) + lkl.poi(exp(theta.est), data) ; abline(h = cut, lty = 2)
ic.lkl.poi <- function(theta, cut, ...) {
    lambda <- exp(theta)
    lkl.poi(lambda, ...) - cut
}
ic.95.prob <- rootSolve::uniroot.all(ic.lkl.poi, range(theta.seq),
                                     data = data, cut = cut)
arrows(x0 = ic.95.prob, y0 = rep(cut, 2),
       x1 = ic.95.prob, y1 = rep(-43, 2), lty = 2, length = .1)
## wald confidence interval  ---------------------------------------------------
cut <- log(.15) + quadprox.poi.repa(theta.est, theta.est, data)
abline(h = cut, lty = 2, col = "darkgray")

se.theta.est <- sqrt(1 / sum(data))
wald.95 <- theta.est + qnorm(c(.025, .975)) * se.theta.est

arrows(x0 = wald.95, y0 = rep(cut, 2),
       x1 = wald.95, y1 = rep(-43, 2), lty = 2, length = .1, col = "darkgray")
```

```r
# a better look in the approximation around the MLE -----------------------------
theta.seq <- seq(-.75, .75, length.out = 25)
plot(theta.seq, lkl.poi(exp(theta.seq), data), type = "l",
     xlab = expression(theta), ylab = "log-likelihood")
curve(quadprox.poi.repa(x, theta.est, data), col = "darkgray", add = TRUE)
abline(v = theta.est, lty = 3)
```

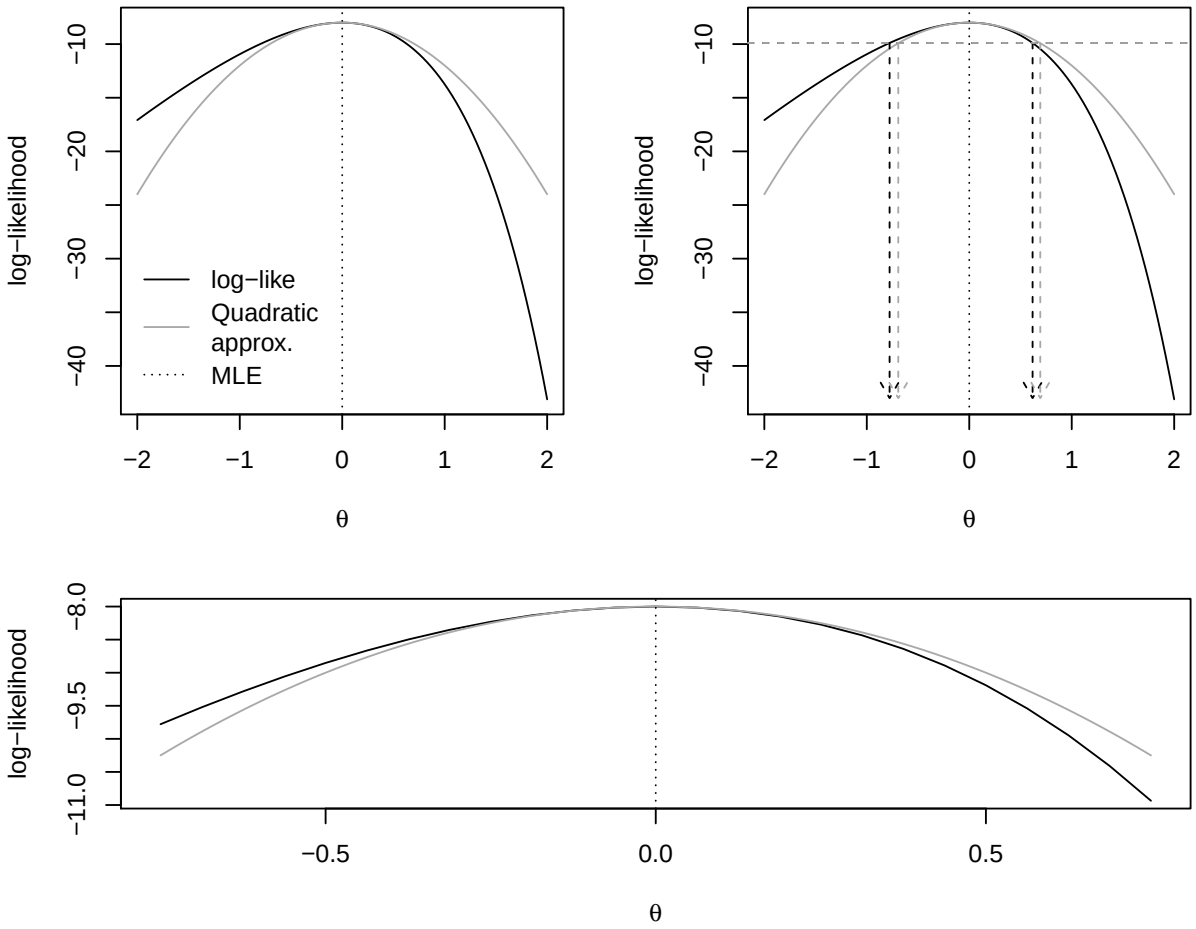

Figure 3: log-likelihood function and MLE of θ in Poisson($\lambda = e^{\theta}$) based on `data`. In the topleft, a quadratic approximation in gray. In the topright, two intervals for θ - one based in the likelihood (in black) and one based in the quadratic approximation (in gray). In the bottom we provide a better look in the quadratic approximation.

We obtained here two intervals for θ. One based in a cut in the likelihood, $\theta \in (-0.778, 0.618)$ - in black on the topright of Figure 3, and one based in a cut in the quadratic approximation of the likelihood, $\theta \in (-0.693, 0.693)$ - in gray on the topright of Figure 3.

With the parametrization $\theta = \log \lambda$ the two intervals are closer than the ones obtained for λ. i.e., for θ (with the use of the log) we get a more regular likelihood.

(c) **Also obtain (by at least two different methods) confidence intervals for the parameter λ from the likelihood function (approximate or not) of θ.**

<r code>

```
# likelihood ------------------------------------------------------------------
theta.seq <- seq(-2, 2, length.out = 100)
lklseq.poi <- lkl.poi(exp(theta.seq), data)

plot(theta.seq, lklseq.poi, type = "l",
     xlab = expression(theta), ylab = "log-likelihood")
abline(v = lambda.est, lty = 3)
# quadratic approximation for the reparametrization ---------------------------
curve(quadprox.poi.repa(x, theta.est, data), col = "darkgray", add = TRUE)
legend(-2.15, -25, c("log-like", "Quadratic\napprox.", "MLE"),
       col = c(1, "darkgray", 1), lty = c(1, 1, 3), bty = "n")
# intervals for lambda --------------------------------------------------------
## probability-based interval -------------------------------------------------
arrows(x0 = exp(ic.95.prob), y0 = c(-9, -36.5),
       x1 = exp(ic.95.prob), y1 = rep(-43, 2), lty = 2, length = .1)
## wald confidence interval  --------------------------------------------------
arrows(x0 = exp(wald.95), y0 = c(-9, -24),
       x1 = exp(wald.95), y1 = rep(-43, 2), lty = 2, length = .1, col = "darkgray")
```

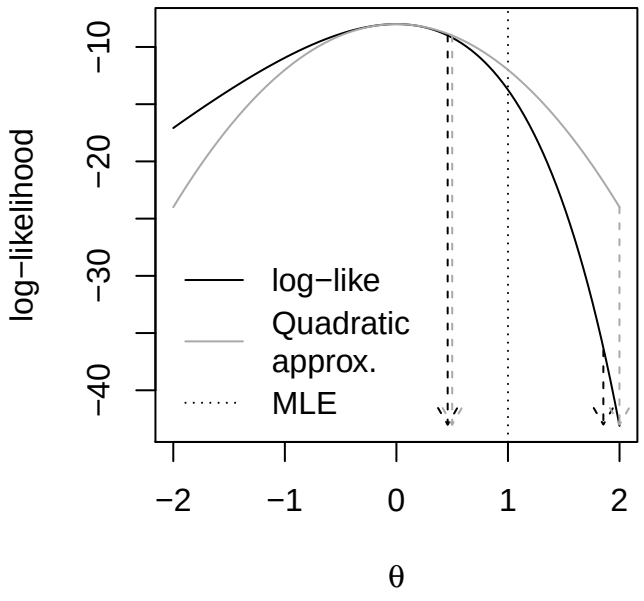

Figure 4: log-likelihood function and quadratic approximation of θ, and MLE of λ in Poisson($\lambda = e^{\theta}$) based on `data`. In dashed, 95% confidence intervals for λ.

Since the MLE has the invariance property, a very simple idea is: take the obtained interval for θ and apply a transformation, $\lambda = e^{\theta}$. This simple idea is true for the interval obtained via likelihood function, as we can see in Figure 4. The obtained interval is the same that the one in letter a). However, this idea doesn't work for the interval based in the quadratic approximation.

n the letter a), via the quadratic approximation we got $\lambda \in (0.307, 1.693)$. Here, applying the relation $\lambda = e^{\theta}$ we get $\lambda \in (0.5, 2)$. i.e., this shows that the invariance property applies to the likelihood, not to the quadratic approximation of the likelihood.

3 Hypergeometric distribution

In order to obtain an audience estimate of a game without using ticket sales data or stadium roulette records, special shirts were distributed to 300 supporters on condition that they were used during a game. During the game 250 fans were randomly selected and 12 of them had the shirt.

a) **Obtain the likelihood function for the total number of supporters.**

We have here a random variable, let's say, X, representing the number of observed successes, k. Here an observed success is select a fan using a special shirt. So, we have

$$X \sim \text{Hypergeometric}(N, K = 300, n = 250),$$

with probability $p = K/N$.

N is the population size, that we want estimate. K is the unknown number of fans using a special shirt, and n is the number of, randomly, selected fans.

$$\begin{aligned}
\Pr[X = k] &= \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}} \\
\Pr[X = 12] &= \frac{\binom{300}{12}\binom{N-300}{250-12}}{\binom{N}{250}} \\
&= \frac{300!\,250!}{288!\,238!\,12!}\frac{(N-300)!\,(N-250)!}{(N-538)!\,N!} \\
&= \text{constant} \times \frac{(N-300)!\,(N-250)!}{(N-538)!\,N!}.
\end{aligned}$$

Since we already have a k, the likelihood is equal to $\Pr[X = k]$.

$$L(N) = \Pr[X = 12] = \text{constant}\frac{(N-300)!\,(N-250)!}{(N-538)!\,N!}$$
$$l(N) = \log L(N) = \log\ \text{constant} + \log\ \frac{(N-300)!\,(N-250)!}{(N-538)!\,N!}$$
$$\approx \log\ \frac{(N-300)!\,(N-250)!}{(N-538)!\,N!}.$$

A plot of the likelihood is provided in Figure 5.

<r code>

```r
# likelihood ------------------------------------------------------------------
lkl.hypergeo <- function(N, N.init, n, K, k) {
    size <- N - N.init
    n1 <- n2 <- n3 <- n4 <- numeric(size)
    N <- N.init
    for (i in 1:size) {
        n1[i] = log(N - K)
        n2[i] = log(N - n)
        n3[i] = log(N - K - n + k)
        n4[i] = log(N)
        N = N + 1
    }
    lkl <- sum(n1) + sum(n2) - sum(n3) - sum(n4)
    return(lkl)
}
compute.lkl <- function(grid, N.init, n, K, k) {
    size.grid = length(grid)
    lkl.grid = numeric(size.grid)
    i = 1
    for (j in grid) {
        lkl.grid[i] = lkl.hypergeo(N = j, N.init = N.init, n = n, K = K, k = k)
        i = i + 1
    }
    return(lkl.grid)
}
# plotting
layout(matrix(c(0, 1, 1, 0,
                2, 2, 3, 3), nrow = 2, byrow = TRUE)) # plot window definition

n.grid <- seq(3000, 17000, by = 20)
plot(n.grid, compute.lkl(n.grid, N.init = 539, n = 250, K = 300, k = 12),
     type = "l", xlab = "N", ylab = "log-likelihood")
abline(v = 6250, lty = 2)

n.grid <- seq(539, 1e4, by = 20)
```

```r
plot(n.grid, compute.lkl(n.grid, N.init = 539, n = 250, K = 300, k = 12),
     type = "l", xlab = "N", ylab = "log-likelihood")
abline(v = 6250, lty = 2)

n.grid <- seq(5000, 8000, by = 10)
plot(n.grid, compute.lkl(n.grid, N.init = 539, n = 250, K = 300, k = 12),
     type = "l", xlab = "N", ylab = "log-likelihood")
abline(v = 6250, lty = 2)
```

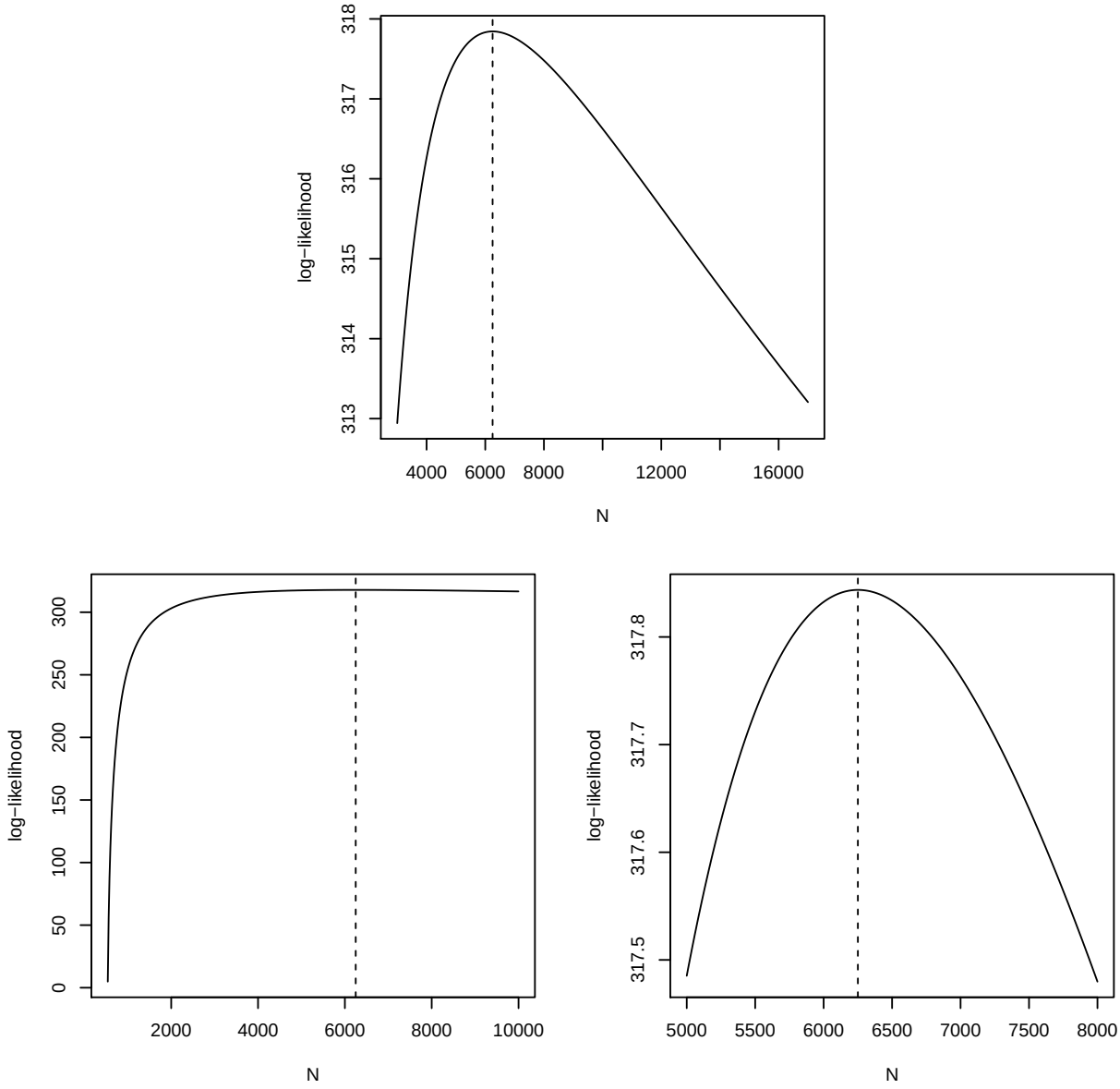

Figure 5: log-likelihood of N in Hypergeometric(N, K = 300, n = 250) based on $k = 12$. In dashed, the MLE. Different ranges of N to have a better understanding of the likelihood behaviour.

The MLE here is given by $\hat{N} = \lfloor Kn/k \rfloor = 6250$.

Figure 5 present some very interesting behaviours.

In the top, going from $N = 3000$ to $N = 17000$ we can see clearly the likelihood asymmetry. In the bottomleft, starting from the first valid possible value, 539, we see basically nothing. i.e., we don't see any possible curvature in the likelihood. In the bottomright we focus around the MLE to see better the curvature. Around that point we see a very small variation in the likelihood, 0.3, for a range of 3000 in N. This shows how smooth is the curvature around the MLE in a large region.

b) **Get the point estimate and the interval estimate, the latter by at least two different methods.**

<r code>

```
# point estimate ------------------------------------------------------------
optimize(lkl.hypergeo, N.init = 539, n = 250, K = 300, k = 12,
         lower = 539, upper = 2e4, maximum = TRUE)
```

```
$maximum
[1] 6251.045

$objective
[1] 317.8437
```

Using the `optimize` routine to do the optimization we obtain $\hat{N} = 6251$, and a log-likelihood of 317.85. This value for $\hat{N}$ is practically equal to what the ML theory says, $\hat{N} = \lfloor Kn/k \rfloor = 6250$.

Next, we compute a quadratic approximation for the hypergeometric (log-)likelihood followed by two intervals, one based in the likelihood, and a wald interval based in the quadratic approximation. All this is presented in Figure 6.

<r code>

```
# quadratic approximation of the likelihood ---------------------------------
hess <- numDeriv:::hessian(lkl.hypergeo, x = 6250,
                           N.init = 539, n = 250, K = 300, k = 12)
# quadratic approx.
quadprox.hypergeo <- function(N, N.est = 6250, N.init, n, K, k) {
    obs.info <- as.vector(-hess) # observed information
    lkl <- lkl.hypergeo(N.est, N.init, n, K, k)
    lkl - .5 * obs.info * (N - N.est)**2
}
layout(matrix(c(1, 2, 3, 3), nrow = 2, byrow = TRUE),
       heights = c(1.5, 1)) # plot window definition
# likelihood plot
n.grid <- seq(3000, 17000, by = 20)
```

```r
plot(n.grid, compute.lkl(n.grid, N.init = 539, n = 250, K = 300, k = 12),
     type = "l", xlab = "N", ylab = "log-likelihood")
abline(v = 6250, lty = 3)

# quadratic approximation plot
curve(quadprox.hypergeo(x, N.init = 539, n = 250, K = 300, k = 12),
      col = "darkgray", add = TRUE)

legend(10500, 318.25, c("log-like", "Quadratic\napprox.", "MLE"),
       col = c(1, "darkgray", 1), lty = c(1, 1, 3), bty = "n")
# intervals for N ------------------------------------------------------------
# first, we plot again the likelihood and the quadratic approx.
n.grid <- seq(2500, 12000, by = 20)

plot(n.grid, compute.lkl(n.grid, N.init = 539, n = 250, K = 300, k = 12),
     type = "l", xlab = "N", ylab = "log-likelihood")
abline(v = 6250, lty = 3)

curve(quadprox.hypergeo(x, N.init = 539, n = 250, K = 300, k = 12),
      col = "darkgray", add = TRUE)
## probability-based interval -----------------------------------------------
# cutoff's corresponding to a 95\% confidence interval for N
cut <- log(.15) + lkl.hypergeo(6250, N.init = 539, n = 250, K = 300, k = 12)
abline(h = cut, lty = 2)

ic.lkl.hypergeo <- function(grid, N.init, n = 250, K, k) {
    compute.lkl(grid, N.init, n, K, k) - cut
}
ic.95.prob <- rootSolve::uniroot.all(ic.lkl.hypergeo, range(n.grid),
                                     N.init = 539, K = 300, k = 12)
arrows(x0 = ic.95.prob, y0 = rep(cut, 2),
       x1 = ic.95.prob, y1 = rep(309.5, 2), lty = 2, length = .1)
## wald confidence interval -------------------------------------------------
cut <- log(.15) + quadprox.hypergeo(6250, N.init = 539, n = 250, K = 300, k = 12)
abline(h = cut, lty = 2, col = "darkgray")

se.n.est <- sqrt(as.vector(-1 / hess))
wald.95 <- 6250 + qnorm(c(.025, .975)) * se.n.est

arrows(x0 = wald.95, y0 = rep(cut, 2),
       x1 = wald.95, y1 = rep(309.5, 2), lty = 2, length = .1, col = "darkgray")
# a better look in the approximation around the MLE ------------------------------
n.grid <- seq(5000, 8000, by = 10)

plot(n.grid, compute.lkl(n.grid, N.init = 539, n = 250, K = 300, k = 12),
     type = "l", xlab = "N", ylab = "log-likelihood")
abline(v = 6250, lty = 3)
```

```
curve(quadprox.hypergeo(x, N.init = 539, n = 250, K = 300, k = 12),
      col = "darkgray", add = TRUE)
```

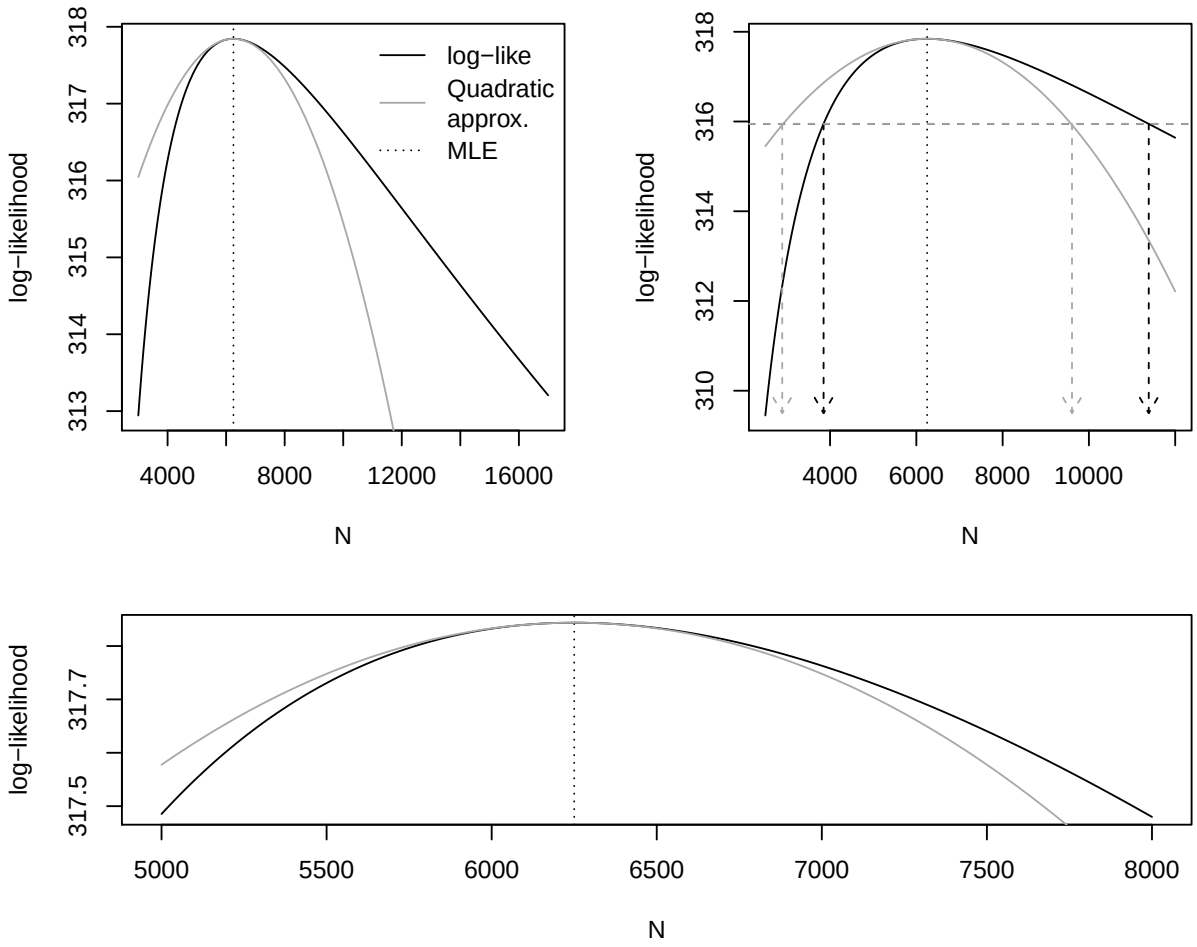

Figure 6: log-likelihood function and MLE of N in Hypergeometric(N, K = 300, n = 250) based on $k = 12$. In the topleft, a quadratic approximation in gray. In the topright, two intervals for N - one based in the likelihood (in black) and one based in the quadratic approximation (in gray). In the bottom we provide a better look in the quadratic approximation.

The quadratic approximation and the intervals were obtained in the same way that in the exercises before, with the only difference that here the hessian was obtained numerically, just for simplicity.

Based in a cut in the likelihood, we obtain $(3850, 11387)$ as a 95% interval for the total number of fans in the game. With the wald interval (quadratic approximation) we obtain $(2892, 9608)$ as a 95% interval. *A considerable difference*.

c) **Repeat and compare the results if there were 500 shirts and 20 with shirts out of the 250.**

The procedure here is exactly the same.

We have

$$\Pr[X=k] = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}, \qquad \Pr[X=20] = \frac{\binom{500}{20}\binom{N-500}{250-20}}{\binom{N}{250}}$$
$$= \frac{500!\,250!}{480!\,230!\,20!}\frac{(N-500)!\,(N-250)!}{(N-730)!\,N!}$$
$$= \text{constant} \times \frac{(N-500)!\,(N-250)!}{(N-730)!\,N!}.$$

With likelihood

$$L(N) = \Pr[X=20] = \text{constant}\frac{(N-500)!\,(N-250)!}{(N-730)!\,N!}$$
$$l(N) = \log L(N) = \log\ \text{constant} + \log\ \frac{(N-500)!\,(N-250)!}{(N-730)!\,N!}$$
$$\approx \log\ \frac{(N-500)!\,(N-250)!}{(N-730)!\,N!}.$$

In Figure 7 we provide the likelihood graphs and intervals for N.

The MLE is exactly the same that the one in the previous scenario, since

$$\hat{N} = \lfloor Kn/k \rfloor = \lfloor 300 \times 250\ /\ 12 \rfloor = \lfloor 500 \times 250\ /\ 20 \rfloor = 6250.$$

- Based in a cut in the likelihood, we obtain $(3850, 11387)$ as a 95% interval for the total number of fans in the game.
- With the wald interval (quadratic approximation) we obtain $(2892, 9608)$ as a 95% interval for the total number of fans.

A considerable difference.

<r code>

```
# plotting ----------------------------------------------------------------
layout(matrix(c(1, 2, 3, 3), nrow = 2, byrow = TRUE),
       heights = c(1.5, 1)) # plot window definition
## likelihood -------------------------------------------------------------
n.grid <- seq(3000, 17000, by = 20)

plot(n.grid, compute.lkl(n.grid, N.init = 731, n = 250, K = 500, k = 20),
     type = "l", xlab = "N", ylab = "log-likelihood")
```

```r
abline(v = 6250, lty = 3)
## quadratic approximation ---------------------------------------------------
hess <- numDeriv:::hessian(lkl.hypergeo, x = 6250,
                           N.init = 731, n = 250, K = 500, k = 20)
curve(quadprox.hypergeo(x, N.init = 731, n = 250, K = 500, k = 20),
      col = "darkgray", add = TRUE)
legend(10500, 382.5, c("log-like", "Quadratic\napprox.", "MLE"),
       col = c(1, "darkgray", 1), lty = c(1, 1, 3), bty = "n")
# intervals for N -----------------------------------------------------------
# first, we plot again the likelihood and the quadratic approx.
n.grid <- seq(2750, 11000, by = 20)

plot(n.grid, compute.lkl(n.grid, N.init = 731, n = 250, K = 500, k = 20),
     type = "l", xlab = "N", ylab = "log-likelihood")
abline(v = 6250, lty = 3)

curve(quadprox.hypergeo(N = x, N.init = 731, n = 250, K = 500, k = 20),
      col = "darkgray", add = TRUE)
## probability-based interval ------------------------------------------------
# cutoff's corresponding to a 95\% confidence interval for N
cut <- log(.15) + lkl.hypergeo(6250, N.init = 731, n = 250, K = 500, k = 20)
abline(h = cut, lty = 2)

ic.95.prob <- rootSolve::uniroot.all(ic.lkl.hypergeo, range(n.grid),
                                     N.init = 731, K = 500, k = 20)
arrows(x0 = ic.95.prob, y0 = rep(cut, 2),
       x1 = ic.95.prob, y1 = rep(370.25, 2), lty = 2, length = .1)
## wald confidence interval --------------------------------------------------
cut <- log(.15) + quadprox.hypergeo(6250, N.init = 731, n = 250, K = 500, k = 20)
abline(h = cut, lty = 2, col = "darkgray")

se.n.est <- sqrt(as.vector(-1 / hess))
wald.95 <- 6250 + qnorm(c(.025, .975)) * se.n.est

arrows(x0 = wald.95, y0 = rep(cut, 2),
       x1 = wald.95, y1 = rep(370.25, 2), lty = 2, length = .1, col = "darkgray")
# a better look in the approximation around the MLE -------------------------
n.grid <- seq(5000, 7500, by = 10)

plot(n.grid, compute.lkl(n.grid, N.init = 731, n = 250, K = 500, k = 20),
     type = "l", xlab = "N", ylab = "log-likelihood")
abline(v = 6250, lty = 3)
curve(quadprox.hypergeo(x, N.init = 731, n = 250, K = 500, k = 20),
      col = "darkgray", add = TRUE)
```

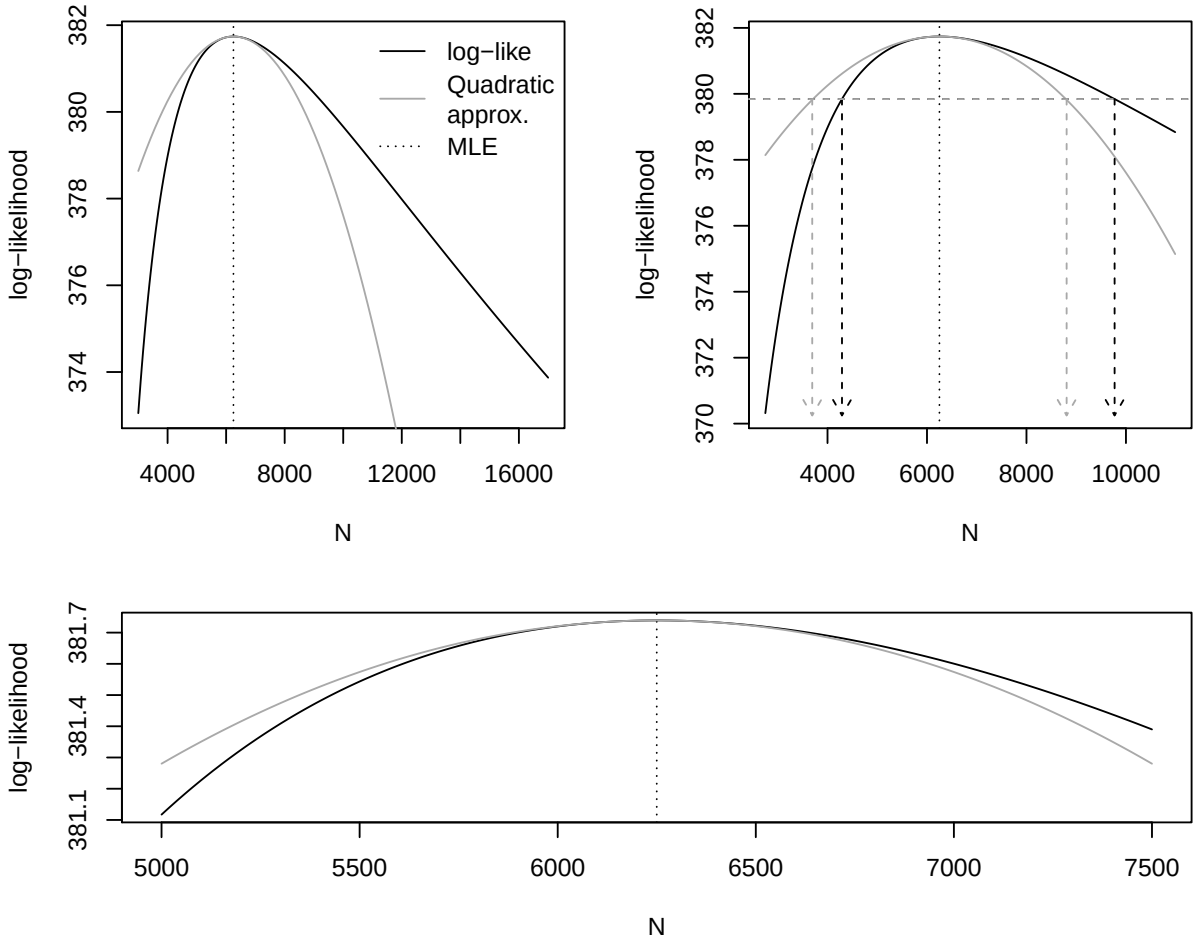

Figure 7: log-likelihood function and MLE of N in Hypergeometric(N, K = 500, n = 250) based on $k = 20$. In the topleft, a quadratic approximation in gray. In the topright, two intervals for N - one based in the likelihood (in black) and one based in the quadratic approximation (in gray). In the bottom we provide a better look in the quadratic approximation.

Comparing with the previous scenario the intervals changed considerably, even with MLE retain the same.

The likelihood shape is the same that before, just with a vertical increase. Before the maximum log-likelihood estimate was around 318, now is around 382.

4 Gaussian distribution with interval data

The following independent observations were taken from one r.v. $X \sim N(\mu, \sigma^2)$.

<r code>

- `[1] 56.4 54.2 65.3 49.2 50.1 56.9 58.9 62.5 70.0 61.0`

- **another 5 observations are known to be less than 50.**
- **it is known that 3 other observations are greater than 65.**

(a) **Write the likelihood function.**

The likelihood can be expressed as

$$
\begin{aligned}
L(\theta) &= \left(\prod_{i=1}^{10} f(x_i)\right) \times (F(50))^5 \times (1 - F(65))^3 \\
&= \left(\prod_{i=1}^{10} \phi\left(\frac{y_i - \mu}{\sigma}\right)\right) \times \left(\Phi\left(\frac{50 - \mu}{\sigma}\right)\right)^5 \times \left(1 - \Phi\left(\frac{65 - \mu}{\sigma}\right)\right)^3 \\
l(\theta) &= \sum_{i=1}^{10} \log \phi\left(\frac{y_i - \mu}{\sigma}\right) + 5 \log \Phi\left(\frac{50 - \mu}{\sigma}\right) + 3 \log\left(1 - \Phi\left(\frac{65 - \mu}{\sigma}\right)\right).
\end{aligned}
$$

<r code>

```r
# likelihood ------------------------------------------------------------------
## if logsigma = TRUE, we do the parametrization using log(sigma)
lkl.norm <- function(par, xs, less.than, more.than, logsigma = FALSE) {
    if (logsigma) par[2] <- exp(par[2])
    l1 <- sum(dnorm(xs, mean = par[1], sd = par[2], log = TRUE))
    l2 <- 5 * log(pnorm(less.than, mean = par[1], sd = par[2]))
    l3 <- 3 * log(1 - pnorm(more.than, mean = par[1], sd = par[2]))
    lkl <- l1 + l2 + l3
    # we return the negative of the log-likelihood,
    # since the optim routine performs a minimization
    return(-lkl)
}
```

(b) **Obtain estimates of maximum likelihood.**

<r code>

```r
# providing an initial guess based uniquely in the ten points sample
init.guess <- c(mean(data), sd(data))
# parameters estimation
(par.est <- optim(init.guess, lkl.norm, logsigma = FALSE,
                  xs = data, less.than = 65, more.than = 50)$par)
```

```
[1] 58.10630  5.66888
```

$$\hat{\mu} = 58.106, \quad \hat{\sigma} = 5.669.$$

FYI: In the ten points sample we have $\bar{x} = 58.45$ and $s^2 = 6.527$.

(c) **Obtain profiles of likelihood.**

First, we do the graph of the likelihood surface in the deviance scale, since this scale is much more convenient. This requires the value of the likelihood evaluated in the parameter estimates.

The deviance here is basically

$$D(\theta) = -2(l(\theta) - l(\hat{\theta})),$$

with θ being the vector of parameters μ and σ.

In Figure 8 we have the likelihood surfaces for two parametrizations: σ and $\log \sigma$.

<r code>

```
# deviance ----------------------------------------------------------------
dev.norm <- function(theta, est, ...) {
    # remember: here we have the negative of the log-likelihoods,
    # so the sign the different
    return(2 * (lkl.norm(theta, ...) - lkl.norm(est, ...)))
}
dev.norm.surf <- Vectorize(function(x, y, ...) dev.norm(c(x, y), ...))

par(mfrow = c(1, 2)) # plot window denifition, one row and two columns
## parametrization: sigma ------------------------------------------------
mu.grid <- seq(52, 63.75, length.out = 100)
sigma.grid <- seq(3.25, 12.5, length.out = 100)

outer.grid <- outer(mu.grid, sigma.grid, FUN = dev.norm.surf, est = par.est,
                    xs = data, less.than = 65, more.than = 50)

contour.levels <- c(.99, .95, .9, .7, .5, .3, .1, .05)
contour(mu.grid, sigma.grid, outer.grid,
        levels = qchisq(contour.levels, df = 2), labels = contour.levels,
        xlab = expression(mu), ylab = expression(sigma))
# converting par.est to a matricial object
points(t(par.est), col = "darkgray", pch = 19, cex = .75)
## parametrization: log sigma --------------------------------------------
par.est <- optim(init.guess, lkl.norm, logsigma = TRUE,
                 xs = data, less.than = 65, more.than = 50)$par

outer.grid <- outer(mu.grid, log(sigma.grid), FUN = dev.norm.surf, est = par.est,
                    xs = data, logsigma = TRUE, less.than = 65, more.than = 50)
```

```r
contour.levels <- c(.99, .95, .9, .7, .5, .3, .1, .05)
contour(mu.grid, log(sigma.grid), outer.grid,
        levels = qchisq(contour.levels, df = 2), labels = contour.levels,
        xlab = expression(mu), ylab = expression(log(sigma)))
points(t(par.est), col = "darkgray", pch = 19, cex = .75)
```

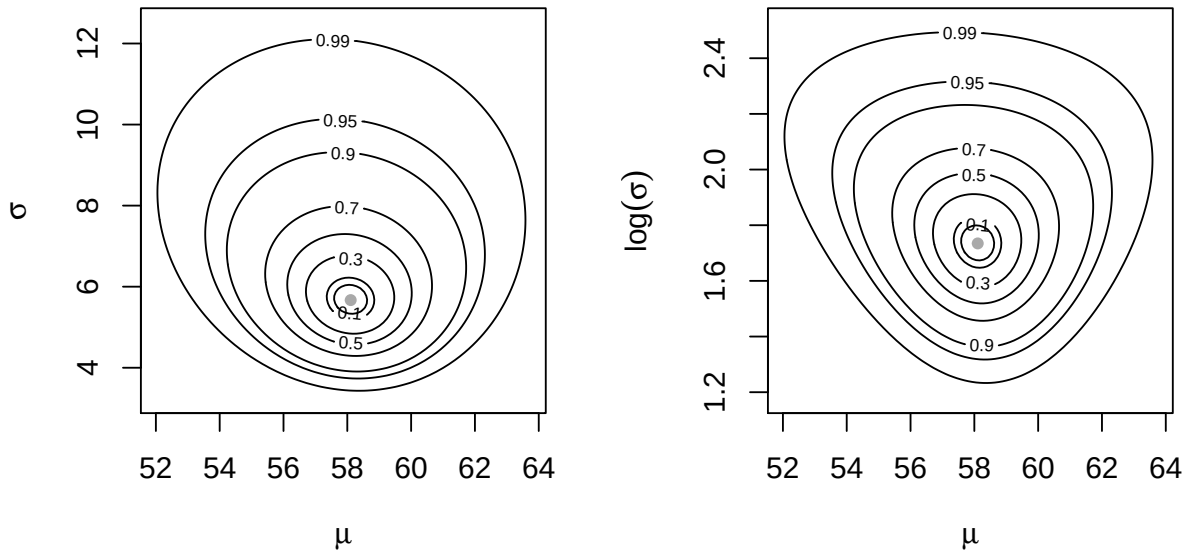

Figure 8: Deviances of (μ, σ) and $(\mu, \log \sigma)$ for a Gaussian sample made of interval data.

From Figure 8 two things are important to mention. First, with interval data, the surface doesn't exhibit orthogonally between the parameters. Second, in the log parametrization of sigma we have better behavior in the surface.

Now, we obtain the likelihood profiles for each parameter. The profiles are presented in Figure 9.

To obtain the likelihood profile for μ, let's say, we need to create a grid of values of μ and for each value of this grid find the value, let's say, $\hat{\sigma}_\mu$ that maximizes the profile likelihood. We do this below for μ and σ, the results can be seen in Figure 9.

<r code>

```r
# profiles ----------------------------------------------------------------------
# optimizing one parameter per time (optimize routine: single par. optimization)
# we need to split the par argument in two
lkl.norm.profiles <- function(mu, sigma, xs, less.than, more.than) {
    l1 <- sum(dnorm(xs, mean = mu, sd = sigma, log = TRUE))
    l2 <- 5 * log(pnorm(less.than, mean = mu, sd = sigma))
    l3 <- 3 * log(1 - pnorm(more.than, mean = mu, sd = sigma))
    lkl <- l1 + l2 + l3
```

```
    return(lkl)} ; par(mfrow = c(1, 2)) # plot window definition
## mu ---------------------------------------------------------------------------
mu.grid <- seq(45, 69.25, length.out = 100)
# with the optimize routine we obtain two numbers:
# the optimal sigma for the given value of mu.grid, and the log-likelihood value
mu.perf <- matrix(0, nrow = length(mu.grid), ncol = 2)
for (i in 1:length(mu.grid)) {
    mu.perf[i, ] <- unlist(
        optimize(lkl.norm.profiles, lower = 0, upper = 50, mu = mu.grid[i],
                 xs = data, less.than = 65, more.than = 50, maximum = TRUE))}
plot(mu.grid, mu.perf[ , 2], type = "l",
     xlab = expression(mu), ylab = expression(l(mu[sigma])))
abline(v = par.est[1], lty = 2)
## sigma ------------------------------------------------------------------------
sigma.grid <- seq(3.25, 13.75, length.out = 100)
sigma.perf <- matrix(0, nrow = length(sigma.grid), ncol = 2)
for (i in 1:length(sigma.grid)) {
    sigma.perf[i, ] <- unlist(
        optimize(lkl.norm.profiles, lower = 0, upper = 100, sigma = sigma.grid[i],
                 xs = data, less.than = 65, more.than = 50, maximum = TRUE))}
plot(sigma.grid, sigma.perf[ , 2], type = "l",
     xlab = expression(sigma), ylab = expression(l(sigma[mu])))
# we apply the exponential, since we did log sigma optimization by last
abline(v = exp(par.est[2]), lty = 2)
```

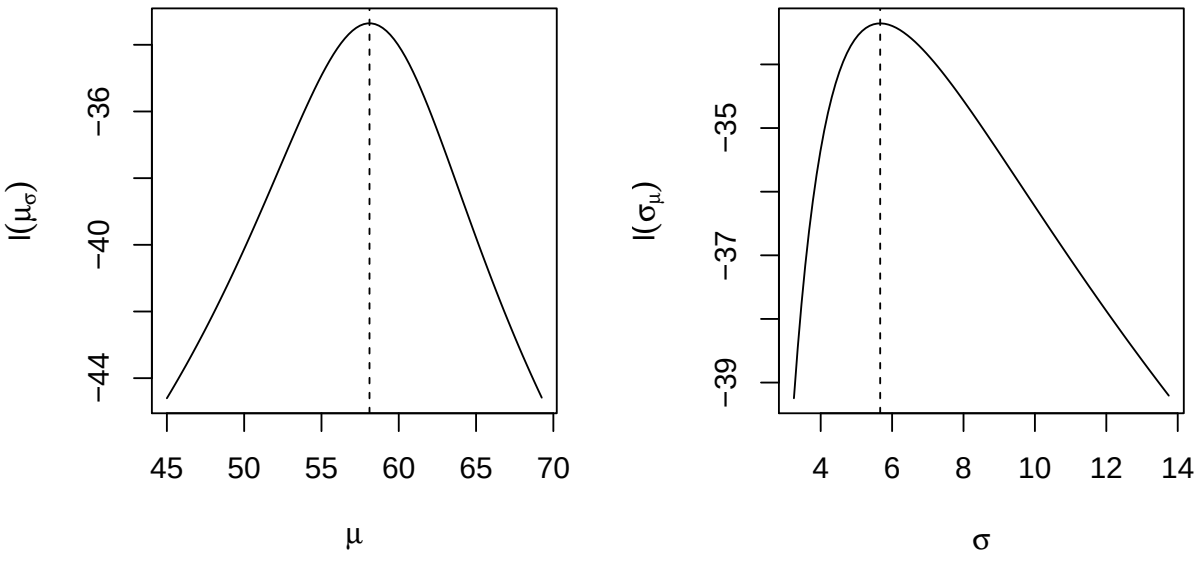

Figure 9: Profile log-likelihoods for μ and σ for a Gaussian sample made of interval data. In dashed, the respective MLE's.

(d) **Obtain the standard error estimates.**

With the `optim` routine we obtain the hessian.

<r code>

```
init.guess <- c(mean(data), sd(data)) # initial guess
# parameters estimation, now also computing the hessian
(hess <- optim(init.guess, lkl.norm, xs = data,
               less.than = 65, more.than = 50, hessian = TRUE)$hessian)
```

```
           [,1]       [,2]
[1,] 0.38190632 0.04403561
[2,] 0.04403561 0.80092515
```

And we know that the hessian, $H(\theta)$, is the negative of the observed information, $I_O(\theta)$, and that asymptotically the variance of $\hat{\theta}$ is $I_O(\theta)^{-1}$. Thus,

<r code>

```
(se <- diag(sqrt(1 / hess))) # standard error
```

```
[1] 1.618160 1.117388
```

The standard errors, se, are

$$\text{se}(\hat{\mu}) = 1.618, \quad \text{se}(\hat{\sigma}) = 1.117.$$

5 Gaussian regression

Consider the following data table (adapted/modified from Montgomery & Runger, 1994) to which we wish to fit a simple linear regression model by relating the response variable Y (purity in %) to an explanatory variable X (hydrocarbon level).

X	0.99	1.02	1.15	1.29	1.46	1.36	0.87	1.23	1.55	1.40
Y	99.01	89.05	91.43	93.74	96.73	94.45	87.59	91.77	99.42	93.65
X	1.19	1.15	0.98	1.01	1.11	1.20	1.26	1.32	1.43	0.95
Y	93.54	92.52	90.56	89.54	89.85	90.39	93.25	93.41	94.98	87.33

(a) **Find the likelihood function.**

<r code>

```
data <- data.frame(
    x = c(.99, 1.02, 1.15, 1.29, 1.46, 1.36, .87, 1.23, 1.55, 1.4,
          1.19, 1.15, .98, 1.01, 1.11, 1.2, 1.26, 1.32, 1.43, .95),
```

```
    y = c(99.01, 89.05, 91.43, 93.74, 96.73, 94.45, 87.59, 91.77, 99.42,
          93.65, 93.54, 92.52, 90.56, 89.54, 89.85, 90.39, 93.25, 93.41,
          94.98, 87.33))
plot(y ~ x, data, xlab = "hydrocarbon level", ylab = expression("purity"~("%")))
```

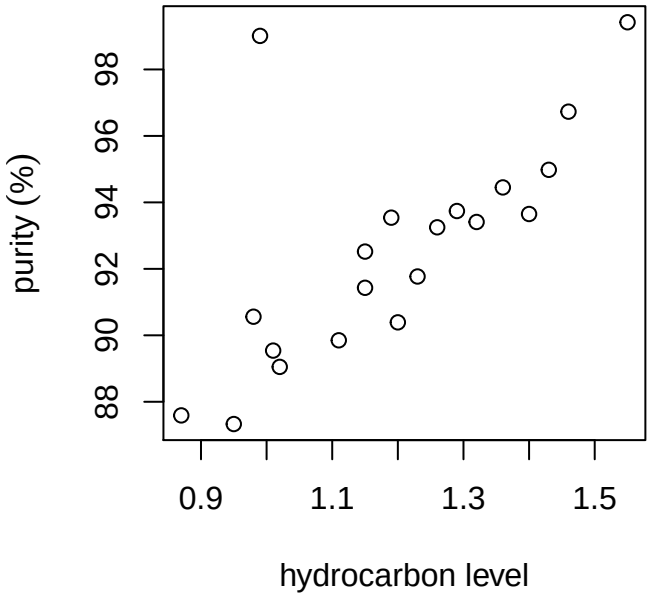

Figure 10: Dispersion between hydrocarbon level $\times$ purity (%).

Assuming a Normal distribution for $Y|X$, we have the following log-likelihood, in matricial form

$$\begin{aligned} l(\theta; Y) &= -\frac{n}{2}\log 2\pi - n\log\sigma - \frac{1}{2\sigma^2}\|Y - X\beta\|^2 \\ &= \text{constant} - n\log\sigma - \frac{1}{2\sigma^2}\|Y - X\beta\|^2, \end{aligned}$$

were $\theta = [\beta_0\ \beta_1\ \sigma]^\top$, $\beta = [\beta_0\ \beta_1]^\top$, and $\mu_i = \beta_0 + \beta_1 x_i$.

(b) **Find the estimates of maximum likelihood.**

<r code>

```
# likelihood ------------------------------------------------------------------
lkl.model.norm <- function(b0, b1, sigma, y, x, logsigma = FALSE) {
    ## if logsigma = TRUE, we do the parametrization using log(sigma)
    if (logsigma) sigma <- exp(sigma)
    mu <- b0 + b1 * x
```

```
    lkl <- sum(dnorm(y, mean = mu, sd = sigma, log = TRUE))
    # we return the negative of the log-likelihood,
    # since the optim routine performs a minimization
    return(-lkl)
}
library(bbmle)
par.est <- mle2(lkl.model.norm,
                start = list(b0 = min(data$y), b1 = 1.5, sigma = sd(data$y)),
                data = list(y = data$y, x = data$x), method = "BFGS")
par.est@coef
```

```
       b0        b1     sigma
77.988429 12.225489  2.343799
```

Just checking...

<r code>

```
lm(y ~ x, data)$coef # beta_0 and beta_1
```

```
(Intercept)           x
   77.98996    12.22453
```

```
par.est@coef[[3]]**2 # sigma^2
```

```
[1] 5.493392
```

```
(summary(lm(y ~ x, data))$sigma**2) * (nrow(data) - 2) / nrow(data)
```

```
[1] 5.493386
```

Checked, ✓.

(c) **Obtain the joint likelihood for the parameters β_0 and β_1:**

i. **considering σ fixed with value equal to its estimate.**

<r code>

```
# deviance ----------------------------------------------------------------
dev.model.norm <- function(beta, theta.est, ...) {
    b0 <- beta[1] ; b1 <- beta[2]
    b0.est <- theta.est[[1]] ; b1.est <- theta.est[[2]]
    sigma.est <- theta.est[[3]]
    lkl.grid <- lkl.model.norm(b0, b1, sigma.est, ...)
    lkl.est <- lkl.model.norm(b0.est, b1.est, sigma.est, ...)
    lkl.diff <- lkl.grid - lkl.est
```

```
    # remember: here we have the negative of the log-likelihoods,
    # so the sign is different
    return(2 * lkl.diff)
}
# deviance, vectorized version ----------------------------------------------------
dev.model.norm.surf <- Vectorize(function(a, b, ...) dev.model.norm(c(a, b), ...))
# deviance contour, plotting ------------------------------------------------------
b0.grid <- seq(67.5, 88.5, length.out = 100)
b1.grid <- seq(3.5, 21, length.out = 100)

outer.grid <- outer(b0.grid, b1.grid, FUN = dev.model.norm.surf,
                    theta.est = par.est@coef, y = data$y, x = data$x)

contour.levels <- c(.99, .95, .9, .7, .5, .3, .1, .05)
contour(b0.grid, b1.grid, outer.grid,
        levels = qchisq(contour.levels, df = 2), labels = contour.levels,
        xlab = expression(beta[0]), ylab = expression(beta[1]))
# converting par.est@coef to a matricial object
points(t(par.est@coef[1:2]), col = 1, pch = 19, cex = .75)
```

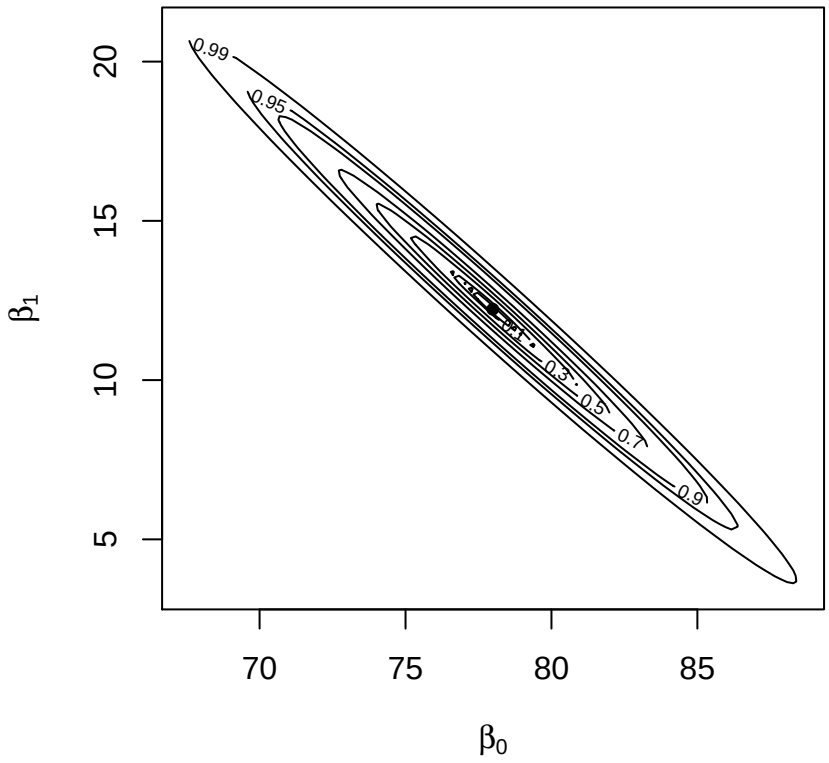

Figure 11: Deviance of (β_0, β_1) with σ fixed in it's estimative, for a Gaussian regression.

ii. **obtaining the profile likelihood (joint - 2D) with respect to σ.**

<r code>
```r
# profile ----------------------------------------------------------------------
sigma.grid <- seq(1.7, 3.5, length.out = 100)
# with the optim routine we obtain three numbers:
# the estimates of beta_0, beta_1, and the log-likelihood value
beta.perf <- matrix(0, nrow = length(sigma.grid), ncol = 3)
for (i in 1:length(sigma.grid)) {
    beta.perf[i, ] <- unlist(
        mle2(lkl.model.norm, start = list(b0 = min(data$y), b1 = 1.5),
             data = list(y = data$y, x = data$x, sigma = sigma.grid[i]),
             method = "BFGS"
        )@details[1:2])
}
plot(sigma.grid, -1 * beta.perf[ , 3], type = "l",
     xlab = expression(sigma), ylab = "log-likelihood")
abline(v = par.est@coef[[3]], lty = 2)
```

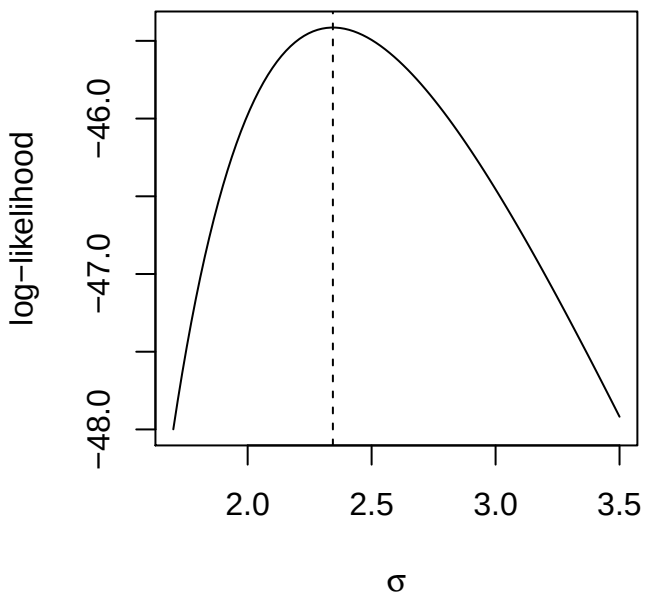

Figure 12: Profile log-likelihood for σ, for a Gaussian regression. In dashed, the MLE.

And how the β's fluctuates across the grid of σ?

<r code>
```r
summary(beta.perf[ , 1]) # beta_0
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  77.98   77.99   77.99   77.99   78.00   78.01
```

```r
summary(beta.perf[ , 2]) # beta_1
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  12.21   12.22   12.22   12.22   12.22   12.23
```

Very few, as expected. Since the estimation of β doesn't depend on σ.

(d) **Obtain the profile likelihood for the parameters β_0 and β_1 individually.**

<r code>

```r
par(mfrow = c(1, 2)) # plot window definition, one row and two columns
# profiling ------------------------------------------------------------------
coef.profile <- function(grid, coef = c("b0", "b1")) {
    n <- length(grid)
    prof <- numeric(n)
    for (i in 1:n) {
        prof[i] <- switch(
            coef,
            "b0" = mle2(lkl.model.norm, start = list(b1 = 1.5),
                        data = list(y = data$y, x = data$x, b0 = grid[i],
                                    sigma = par.est@coef[[3]]),
                        method = "BFGS"
            )@details$value,
            "b1" = mle2(lkl.model.norm, start = list(b0 = min(data$y)),
                        data = list(y = data$y, x = data$x, b1 = grid[i],
                                    sigma = par.est@coef[[3]]),
                        method = "BFGS"
            )@details$value)
    }
    return((-1) * prof)
}
# beta_0 ---------------------------------------------------------------------
b0.grid <- seq(67.5, 88.5, length.out = 100)

plot(b0.grid, coef.profile(b0.grid, "b0"), type = "l",
     xlab = expression(beta[0]), ylab = "log-likelihood")

abline(v = par.est@coef[[1]], lty = 2)
# beta_1 ---------------------------------------------------------------------
b1.grid <- seq(3.5, 21, length.out = 100)

plot(b1.grid, coef.profile(b1.grid, "b1"), type = "l",
     xlab = expression(beta[1]), ylab = "log-likelihood")

abline(v = par.est@coef[[2]], lty = 2)
```

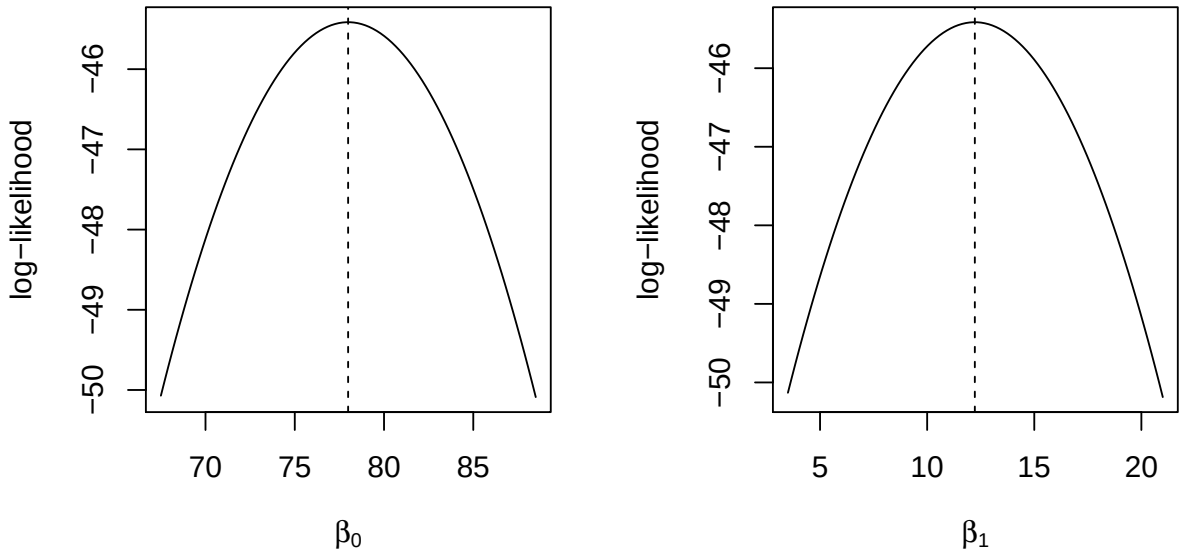

Figure 13: Profile log-likelihoods for β_0 and β_1 for a Gaussian regression. In dashed, MLE's.

6 Poisson regression

Consider a sample of a r.v. Y where it is assumed that $Y_i \sim P(\lambda_i)$ where the parameter λ_i is described by a function of an explanatory variable $\log(\lambda_i) = \beta_0 + \beta_1 x_i$ with values known from x_i.
Simulated data with ($\beta_0 = 2, \beta_1 = 0.5$) are shown below.

Y	10	15	11	37	70	19	12	12	13	88
X	1.7	1.5	0.5	2.8	4.4	1.8	0.4	0.7	1.3	5.1

<r code>

```
data <- data.frame(y = c(10, 15, 11, 37, 70, 19, 12, 12, 13, 88),
                   x = c(1.7, 1.5, .5, 2.8, 4.4, 1.8, .4, .7, 1.3, 5.1))
plot(y ~ x, data)
```

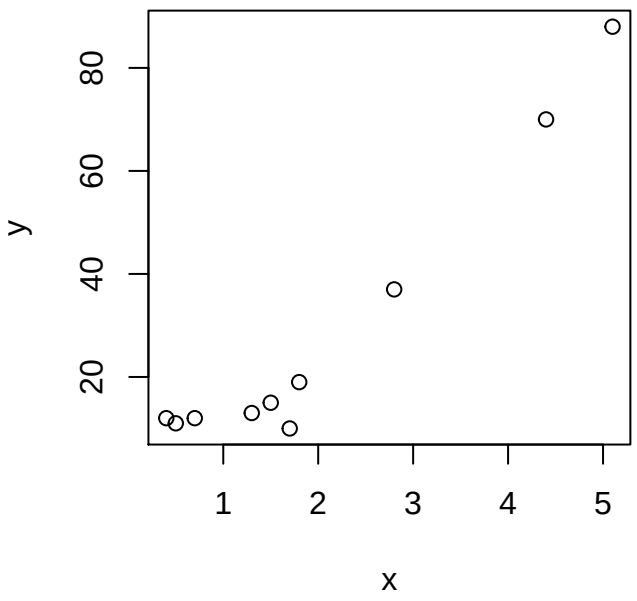

Figure 14: Dispersion between the variables Y and X.

(a) **Obtain the graph of the likelihood function indicating the position of the true values of the parameters.**

The log-likelihood is given by

$$l(\lambda) = -n\lambda + \log\lambda \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} \log y_i!\,, \quad \lambda_i = \exp\{\beta_0 + \beta_1 x_i\}.$$

<r code>

```r
# likelihood ------------------------------------------------------------------
lkl.model.poi <- function(b0, b1, data) {
    n <- nrow(data)
    lambda <- exp(b0 + b1 * data$x)
    lkl <- sum(dpois(data$y, lambda = lambda, log = TRUE))
    # we return the negative of the log-likelihood,
    # since the optim routine performs a minimization
    return(-lkl)
}
# deviance --------------------------------------------------------------------
dev.model.poi <- function(beta, beta.est, ...) {
    b0 <- beta[1] ; b1 <- beta[2]
    b0.est <- beta.est[1] ; b1.est <- beta.est[2]
    lkl.grid <- lkl.model.poi(b0, b1, ...)
    lkl.est <- lkl.model.poi(b0.est, b1.est, ...)
```

```
    lkl.diff <- lkl.grid - lkl.est
    # remember: here we have the negative of the log-likelihoods,
    # so the sign the different
    return(2 * lkl.diff)
}
# deviance, vectorized version ----------------------------------------------------
dev.model.poi.surf <- Vectorize( function(a, b, ...) dev.model.poi(c(a, b), ...) )
# deviance contour, plotting ------------------------------------------------------
b0.grid <- seq(1.625, 2.475, length.out = 100)
b1.grid <- seq(.375, .6, length.out = 100)

outer.grid <- outer(b0.grid, b1.grid, FUN = dev.model.poi.surf,
                    beta.est = c(2, .5), data = data)

contour.levels <- c(.99, .95, .9, .7, .5, .3, .1, .05)
contour(b0.grid, b1.grid, outer.grid,
        levels = qchisq(contour.levels, df = 2), labels = contour.levels,
        xlab = expression(beta[0]), ylab = expression(beta[1]))
# true parameter values
points(c(2, .5), pch = 19, cex = .75)
```

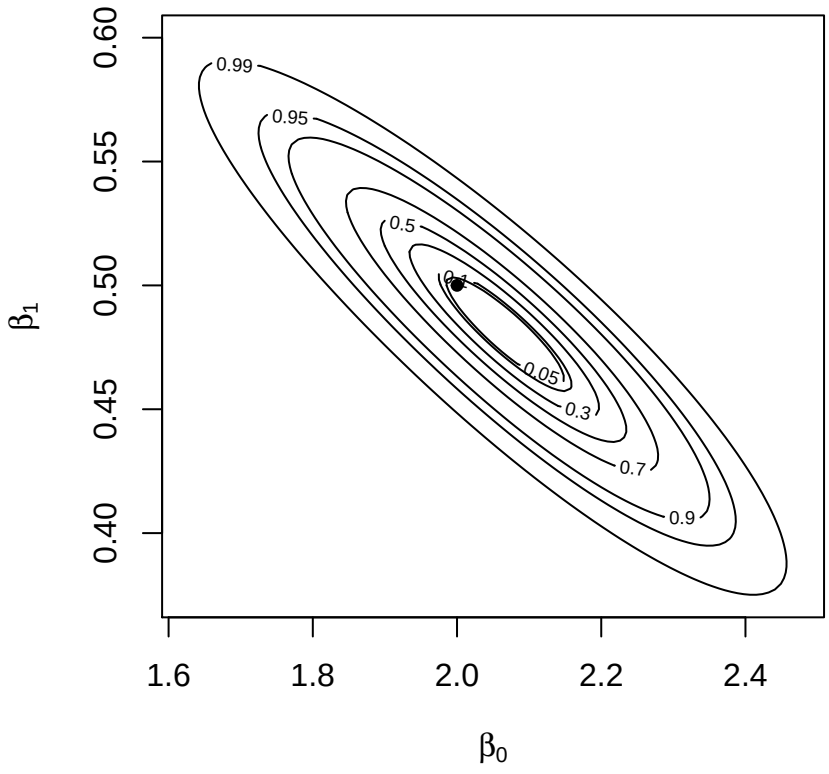

Figure 15: Deviance of (β_0, β_1) for a Poisson regression based on `data`.

(b) **Obtain the likelihood profiles of the parameters.**

<r code>

```r
par(mfrow = c(1, 2)) # plot window definition, one row and two columns
# profiling ---------------------------------------------------------------------
coef.profile <- function(grid, coef = c("b0", "b1")) {
    n <- length(grid) ; prof <- numeric(n)
    for (i in 1:n) {
        # for some points can happen that the hessian not be invertible
        tryCatch({ # with tryCatch we can skip these points
            prof[i] <- switch(
                coef,
                "b0" = mle2(lkl.model.poi, start = list(b1 = .5),
                            data = list(data = data, b0 = grid[i]),
                            method = "BFGS")@details$value,
                "b1" = mle2(lkl.model.poi, start = list(b0 = min(data$y)),
                            data = list(data = data, b1 = grid[i]),
                            method = "BFGS")@details$value)},
            warning = function(w) w)}
    return((-1) * prof)}
# beta_0 ------------------------------------------------------------------------
b0.grid <- seq(1.25, 2.75, length.out = 100)
plot(b0.grid, coef.profile(b0.grid, "b0"), type = "l", xlab = expression(beta[0]),
     ylab = "log-likelihood") ; abline(v = 2, lty = 2)
# beta_1 ------------------------------------------------------------------------
b1.grid <- seq(.3, .675, length.out = 100)
plot(b1.grid, coef.profile(b1.grid, "b1"), type = "l", xlab = expression(beta[1]),
     ylab = "log-likelihood") ; abline(v = .5, lty = 2)
```

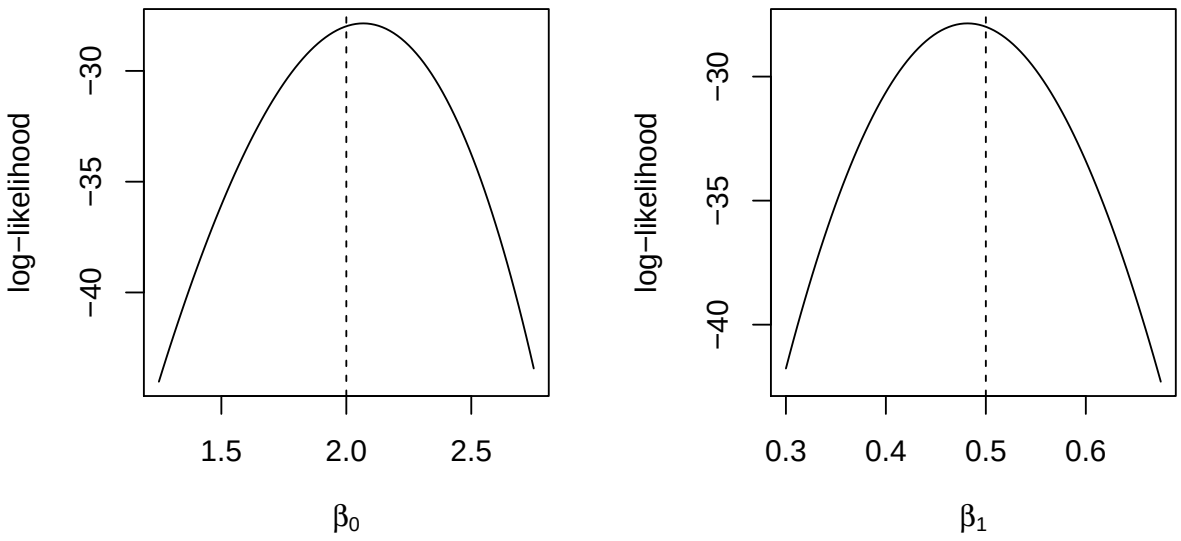

Figure 16: Profile log-likelihoods for β_0 and β_1 for a Poisson regression. In dashed, MLE's.

(c) **Get estimates.**

<r code>

```
(par.est <- mle2(lkl.model.poi, start = list(b0 = min(data$y), b1 = 0),
                 data = list(data = data), method = "BFGS"))
```

```
Call:
mle2(minuslogl = lkl.model.poi, start = list(b0 = min(data$y),
    b1 = 0), method = "BFGS", data = list(data = data))

Coefficients:
       b0        b1
2.0675572 0.4816563

Log-likelihood: -27.86
```

(d) **Get intervals (use different methods).**

<r code>

```
par(mfrow = c(2, 1)) # plot window definition, two rows and one column
library(rootSolve) # uniroot.all function
library(numDeriv) # hessian function
# beta_0 ---------------------------------------------------------------------
## first, plotting the profile with point estimates ----------------------------
plot(b0.grid, coef.profile(b0.grid, "b0"), type = "l",
     xlab = expression(beta[0]), ylab = "log-likelihood")
abline(v = 2, lty = 3) ; abline(v = par.est@coef[1], lty = 2)
## intervals ------------------------------------------------------------------
### probability-based interval -----------------------------------------------
# cutoff's corresponding to a 95\% confidence interval for beta
# minus, since lkl.model.poi return -lkl
cut <- log(.15) - par.est@details$value ; abline(h = cut, lty = 2)
# finding the cutoff values
ic.poi <- function(grid, coef = c("b0", "b1")) {
    lkl.prof <- switch(
        coef,
        "b0" = mle2(lkl.model.poi, start = list(b1 = 0),
                    data = list(data = data, b0 = grid), method = "BFGS"
        )@details$value,
        "b1" = mle2(lkl.model.poi, start = list(b0 = min(data$y)),
                    data = list(data = data, b1 = grid), method = "BFGS"
        )@details$value)
    (-1) * lkl.prof - cut}
# for some "mysterious" reason the uniroot.all
# doesn't work here, but uniroot works perfectly
```

```
ic.95.prob.b0 <- c(uniroot(ic.poi, c(1, par.est@coef[1]), "b0")$root,
                   uniroot(ic.poi, c(par.est@coef[1], 3), "b0")$root)
arrows(x0 = ic.95.prob.b0, y0 = rep(cut, 2),
       x1 = ic.95.prob.b0, y1 = rep(-43.5, 2), lty = 2, length = .1)
### wald confidence interval ----------------------------------------------------
# quadratic approximation
qa.coef.poi <- function(grid, coef = c("b0", "b1")) {
    coef.hat <- switch(coef, "b0" = par.est@coef[1], "b1" = par.est@coef[2])
    lkl.prof <- coef.profile(coef.hat, coef)
    hess <- hessian(coef.profile, x = coef.hat, coef = coef)
    n <- length(grid) ; qa <- numeric(n)
    for (i in 1:n) qa[i] <- lkl.prof + .5 * hess * (grid[i] - coef.hat)**2
    return(qa)}
qa.b0.poi <- qa.coef.poi(b0.grid, "b0") ; lines(b0.grid, qa.b0.poi, col = "darkgray")

cut <- log(.15) + qa.coef.poi(par.est@coef[1], "b0")
abline(h = cut, lty = 2, col = "darkgray")
# now, finding the cutoff values
ic.coef.qa.poi <- function(grid, coef) qa.coef.poi(grid, coef) - cut

wald.95.b0 <- uniroot.all(ic.coef.qa.poi, range(b0.grid), coef = "b0")
arrows(x0 = wald.95.b0, y0 = rep(cut, 2),
       x1 = wald.95.b0, y1 = rep(-43.5, 2), lty = 2, length = .1, col = "darkgray")
### from the mle2 output we get the standard error of each parameter -------------
se.b0 <- summary(par.est)@coef[1, 2]
mle2.95.b0 <- par.est@coef[1] + qnorm(c(.025, .975)) * se.b0

arrows(x0 = mle2.95.b0, y0 = rep(cut, 2),
       x1 = mle2.95.b0, y1 = rep(-43.5, 2), lty = 3, length = .1)
# beta_1 ------------------------------------------------------------------------
plot(b1.grid, coef.profile(b1.grid, "b1"), type = "l",
     xlab = expression(beta[1]), ylab = "log-likelihood")
abline(v = .5, lty = 3) ; abline(v = par.est@coef[2], lty = 2)
## intervals ------------------------------------------------------------------
### probability-based interval ---------------------------------------------------
abline(h = cut, lty = 2) # remember, the cutoff is the same
# for some "mysterious" reason the uniroot.all
# doesn't work here, but uniroot works perfectly
ic.95.prob.b1 <- c(uniroot(ic.poi, c(.2, par.est@coef[2]), "b1")$root,
                   uniroot(ic.poi, c(par.est@coef[2], .8), "b1")$root)
arrows(x0 = ic.95.prob.b1, y0 = rep(cut, 2),
       x1 = ic.95.prob.b1, y1 = rep(-42, 2), lty = 2, length = .1)
### wald confidence interval ----------------------------------------------------
qa.b1.poi <- qa.coef.poi(b1.grid, "b1") ; lines(b1.grid, qa.b1.poi, col = "darkgray")
cut <- log(.15) + qa.coef.poi(par.est@coef[2], "b1")
abline(h = cut, lty = 2, col = "darkgray")
```

```
wald.95.b1 <- uniroot.all(ic.coef.qa.poi, range(b1.grid), coef = "b1")
arrows(x0 = wald.95.b1, y0 = rep(cut, 2),
       x1 = wald.95.b1, y1 = rep(-42, 2), lty = 2, length = .1, col = "darkgray")
### mle2 standard error of ----------------------------------------------------
se.b1 <- summary(par.est)@coef[2, 2]
mle2.95.b1 <- par.est@coef[2] + qnorm(c(.025, .975)) * se.b1

arrows(x0 = mle2.95.b1, y0 = rep(cut, 2),
       x1 = mle2.95.b1, y1 = rep(-42, 2), lty = 3, length = .1)
```

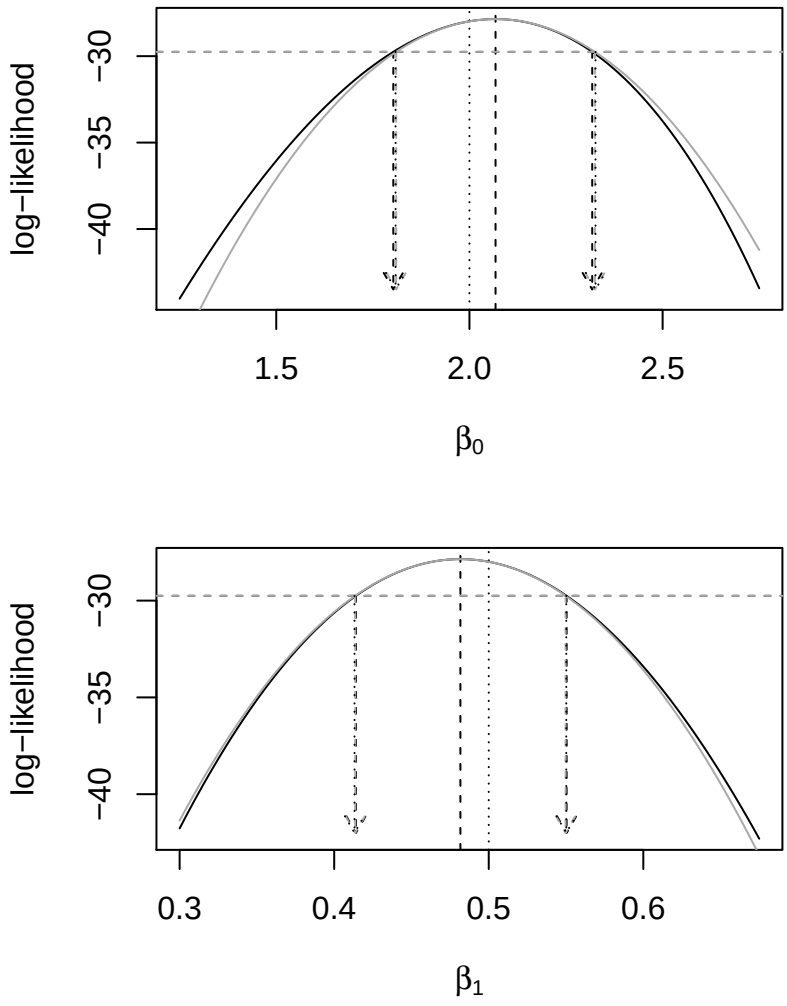

Figure 17: Profile log-likelihoods, in black, for β_0 and β_1 in a Poisson regression with some intervals. In black dotted, the original parameter value. In black dashed, the MLE. The gray curve is a quadratic approx. for the profile, with the gray dashed arrows corresponding to its 95% interval. The black dashed arrows correspond to the 95% interval based in a cut in the likelihood. The black dotted arrows correspond to the 95% interval based in the `mle2` output.

By the amount of code provided in this question, we can see that isn't so straightforward to obtain intervals and quadratic approximations of profiled likelihoods of parameters.

A very nice and interesting thing that we can see in Figure 17 is how all the three obtained intervals match. A brief, but enough, description is already given in the Figure caption, and for more details, you have the provided code.

(e) **Compare the previous results with those provided by the `glm()` function and discuss the findings.**

<r code>

```
# fitting the model via glm() ---------------------------------------------------
model.glm <- glm(y ~ x, data, family = poisson)
# parameter estimates -----------------------------------------------------------
model.glm$coef
```

```
(Intercept)           x
  2.0675451   0.4816599
```

```
# at three decimal cases, the estimates are equal
round(model.glm$coef, 3) == round(par.est@coef, 3)
```

```
(Intercept)           x
       TRUE        TRUE
```

```
# maximum log-likelihood --------------------------------------------------------
logLik(model.glm)
```

```
'log Lik.' -27.85527 (df=2)
```

```
# at eight decimal cases, the max. log-likes are equal
round(logLik(model.glm), 8) == round(par.est@details[[2]] * (-1), 8)
```

```
[1] TRUE
```

```
# standard erros ----------------------------------------------------------------
summary(model.glm)$coef[ , 2]
```

```
(Intercept)           x
 0.13201821  0.03494989
```

```
## at six decimal cases, the standard errors are equal
round(summary(model.glm)$coef[ , 2], 6) == round(summary(par.est)@coef[ , 2], 6)
```

```
(Intercept)           x
       TRUE        TRUE
```

As a general reference guide for questions 2 to 6,
I read and used (in portuguese): `www.leg.ufpr.br/mcie`

```
@conference{mcie,
  author    = {Wagner Hugo Bonat and Paulo Justiniano RIbeiro Jr and
               Elias Teixeira Krainski and Walmes Marques Zeviani},
  title     = {Computational Methods in Statistical Inference},
  year      = {2012},
  publisher = {20th National Symposium on Probability and Statistics (SINAPE)},
  address   = {Jo\~{a}o Pessoa - PB - Brazil},
}
```

7 Poisson regression (*focus* on the profile likelihood)

In the previous exercise, we computed the profile likelihood for each parameter of a simple Poisson regression with just one covariate, i.e., an intercept, β_0, and an angular coefficient, β_1.

Together with the profile and an interval based in a cut in this profile likelihood (let's say, a probability-based interval), we did a quadratic approximation in this profile likelihood. In the "real" world with "real" problems this isn't a very smart thing to do, but here just as an exercise, I believe that is extremely valid.

Using the same functions created before, we first show, in black, in Figure 18, the profiles likelihood together with its respective quadratic approximations.

Now, something that we can look at is to the conditional, or estimated likelihood (as `Pawitan` calls). In the profiled one, for β_0 let's say, in a grid of β_0's, we estimate the β_1's and then get the corresponding likelihoods. In the conditional one, as the name suggests, we fix (conditionate) β_1 in a value, and in a grid of β_0's, we get the likelihoods.

The conditional likelihood for each parameter and its correspondent quadratic approximation is computed and presented, in gray, in Figure 18.

<r code>

```r
# lkl.est.poi: estimated (conditional) poisson likelihood ------------------------
lkl.est.poi <- function(grid, coef = c("b0", "b1")) {
    coef.hat <- switch(coef, "b0" = par.est@coef[2], "b1" = par.est@coef[1])
    n <- length(grid)
    lkl.est <- numeric(n)
    for (i in 1:n) {
        lkl.est[i] <- switch(
            coef,
            "b0" = lkl.model.poi(b0 = grid[i], b1 = coef.hat, data = data),
            "b1" = lkl.model.poi(b0 = coef.hat, b1 = grid[i], data = data))
    }
    return((-1) * lkl.est)
}
```

```
# qa.est.poi: quadratic approximation of the estimated poisson likelihood --------
qa.est.poi <- function(grid, coef = c("b0", "b1")) {
    coef.hat <- switch(coef, "b0" = par.est@coef[1], "b1" = par.est@coef[2])
    lkl.hat <- -par.est@min
    hess <- switch(coef,
                   "b0" = -par.est@details$hessian[1],
                   "b1" = -par.est@details$hessian[4])
    n <- length(grid)
    qa <- numeric(n)
    for (i in 1:n) {
        qa[i] <- switch(coef,
                        "b0" = lkl.hat + .5 * hess * (grid[i] - coef.hat)**2,
                        "b1" = lkl.hat + .5 * hess * (grid[i] - coef.hat)**2)
    }
    return(qa)
}
# ------------------------------------------------------------------------------
par(mfrow = c(2, 1)) # plot window definition, two rows and one column
# beta_0 -----------------------------------------------------------------------
## first, plotting the profile with point estimate -----------------------------
b0.profile <- coef.profile(b0.grid, "b0")

plot(b0.grid, b0.profile, type = "l",
     xlab = expression(beta[0]), ylab = "log-likelihood")
abline(v = par.est@coef[1], lty = 3)
## now, the quadratic approximation of the profile -----------------------------
lines(b0.grid, qa.b0.poi, lty = 2)
## performing and plotting the conditional likelihood --------------------------
lkl.est.b0.poi <- lkl.est.poi(b0.grid, "b0")
lines(b0.grid, lkl.est.b0.poi, col = "darkgray")
## performing and plotting the quadratic approx. of the cond. likelihood --------
qa.est.b0.poi <- qa.est.poi(b0.grid, "b0")
lines(b0.grid, qa.est.b0.poi, col = "darkgray", lty = 2)
# beta_1 -----------------------------------------------------------------------
b1.profile <- coef.profile(b1.grid, "b1")

plot(b1.grid, b1.profile, type = "l",
     xlab = expression(beta[1]), ylab = "log-likelihood")
abline(v = par.est@coef[2], lty = 3)

lines(b1.grid, qa.b1.poi, lty = 2)

lkl.est.b1.poi <- lkl.est.poi(b1.grid, "b1")
lines(b1.grid, lkl.est.b1.poi, col = "darkgray")

qa.est.b1.poi <- qa.est.poi(b1.grid, "b1")
lines(b1.grid, qa.est.b1.poi, col = "darkgray", lty = 2)
```

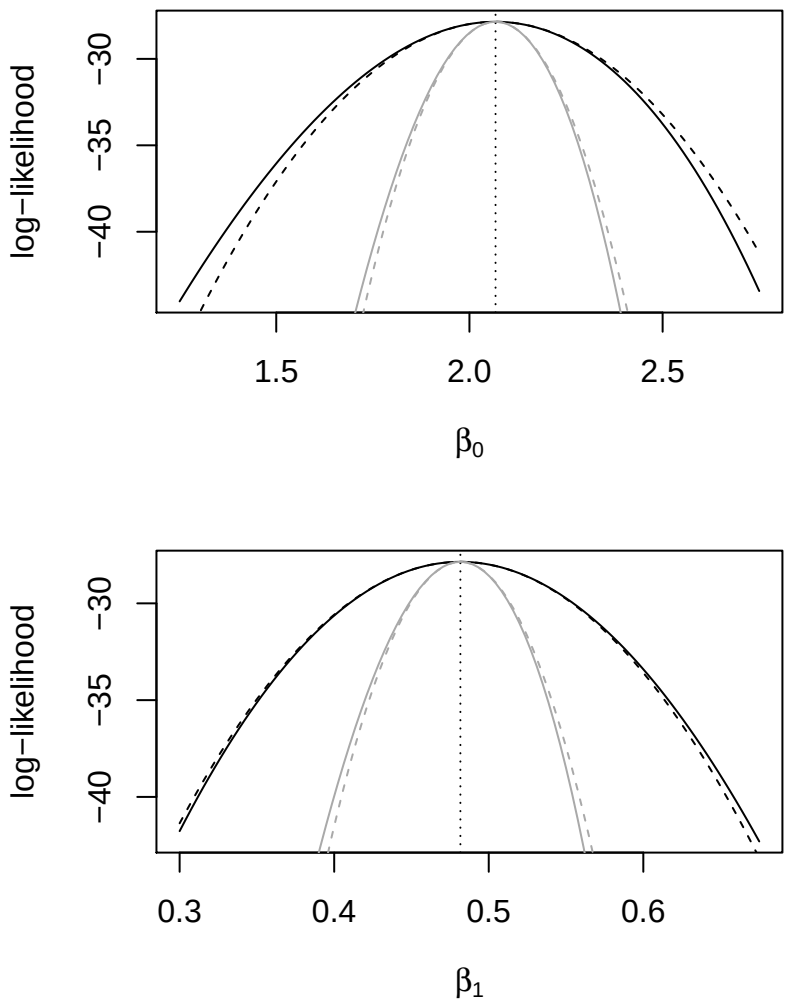

Figure 18: Profile log-likelihoods and quadratic approximations, in black (solid and dashed, respectively), for β_0 and β_1 in a Poisson regression. In dotted, the MLE. Estimated (conditional) log-likelihoods and quadratic approximations, in gray (solid and dashed, respectively).

The first and important note that we can take from Figure 18, and that makes all the possible sense, is how the estimated likelihood is narrow (compared with the profiled one). The explanation for this is that in the estimated (conditional) situation it's assumed as known one of the coefficients, fixing it at the MLE. In the other hand, the profile likelihood takes into account that one of the coefficients is unknown. Hence, it's sensible that the profile is greater than the estimated.

Another important point to mention is about the manner that we compute the profile likelihood. A function with this goal was made in the previous question, `qa.coef.poi`. The important thing to note about these function is that the hessian there is computed from the profile likelihood. This is inefficient, and not necessary. Instead, we can just take the corresponding element in the variance-covariance matrix, invert, and use it. In other

words, we don't need to use the profile likelihood to compute a quadratic approximation to the profile likelihood. We implement this in `smart.qa.coef.poi`.

<r code>

```
# qa: quadratic approximation ----------------------------------------------------
smart.qa.coef.poi <- function(model.mle2, coef = c("b0", "b1"), grid) {
    coef.hat <- switch(coef, "b0" = model.mle2@coef[1], "b1" = model.mle2@coef[2])
    lkl.hat <- (-1) * model.mle2@details$value
    hess <- switch(coef,
                   "b0" = (-1) / model.mle2@vcov[1],
                   "b1" = (-1) / model.mle2@vcov[4])
    n <- length(grid) ; qa <- numeric(n)
    for (i in 1:n) {
        qa[i] <- switch(coef,
                        "b0" = lkl.hat + .5 * hess * (grid[i] - coef.hat)**2,
                        "b1" = lkl.hat + .5 * hess * (grid[i] - coef.hat)**2)}
    return(qa)}
# ------------------------------------------------------------------------------
layout(matrix(1:4, 2, 2), widths = c(1.5, 1)) # plot window definition
# beta_0 -----------------------------------------------------------------------
## first, plotting the profile with the point estimate --------------------------
plot(b0.grid, b0.profile, type = "l", xlab = expression(beta[0]),
     ylab = "log-likelihood") ; abline(v = par.est@coef[1], lty = 3)
## now, the "old" quadratic approximation of the profile ------------------------
lines(b0.grid, qa.b0.poi, lty = 2)
## and finally, the new and smart quadratic approximation of the profile --------
sqa.b0.poi <- smart.qa.coef.poi(model.mle2 = par.est, coef = "b0", grid = b0.grid)
lines(b0.grid, sqa.b0.poi, col = "gray")
# beta_1 -----------------------------------------------------------------------
plot(b1.grid, b1.profile, type = "l", xlab = expression(beta[1]),
     ylab = "log-likelihood") ; abline(v = par.est@coef[2], lty = 3)
lines(b1.grid, qa.b1.poi, lty = 2)

sqa.b1.poi <- smart.qa.coef.poi(model.mle2 = par.est, coef = "b1", grid = b1.grid)
lines(b1.grid, sqa.b1.poi, col = "gray")
# how much the quadratic approximations are similar? ---------------------------
compare.qas <- function(dp, coef = c("b0", "b1")) {
    compare <- numeric(dp)
    for (i in 1:dp) {
        compare[i] <- switch(
            coef,
            "b0" = table(round(sqa.b0.poi, i) == round(qa.b0.poi, i))["TRUE"],
            "b1" = table(round(sqa.b1.poi, i) == round(qa.b1.poi, i))["TRUE"])}
    compare[is.na(compare)] <- 0
    graph <- plot(compare, type = "h",
                  xlab = "decimal places", ylab = "% of matching")
    return(invisible(graph))}
compare.qas(6, "b0") ; compare.qas(6, "b1")
```

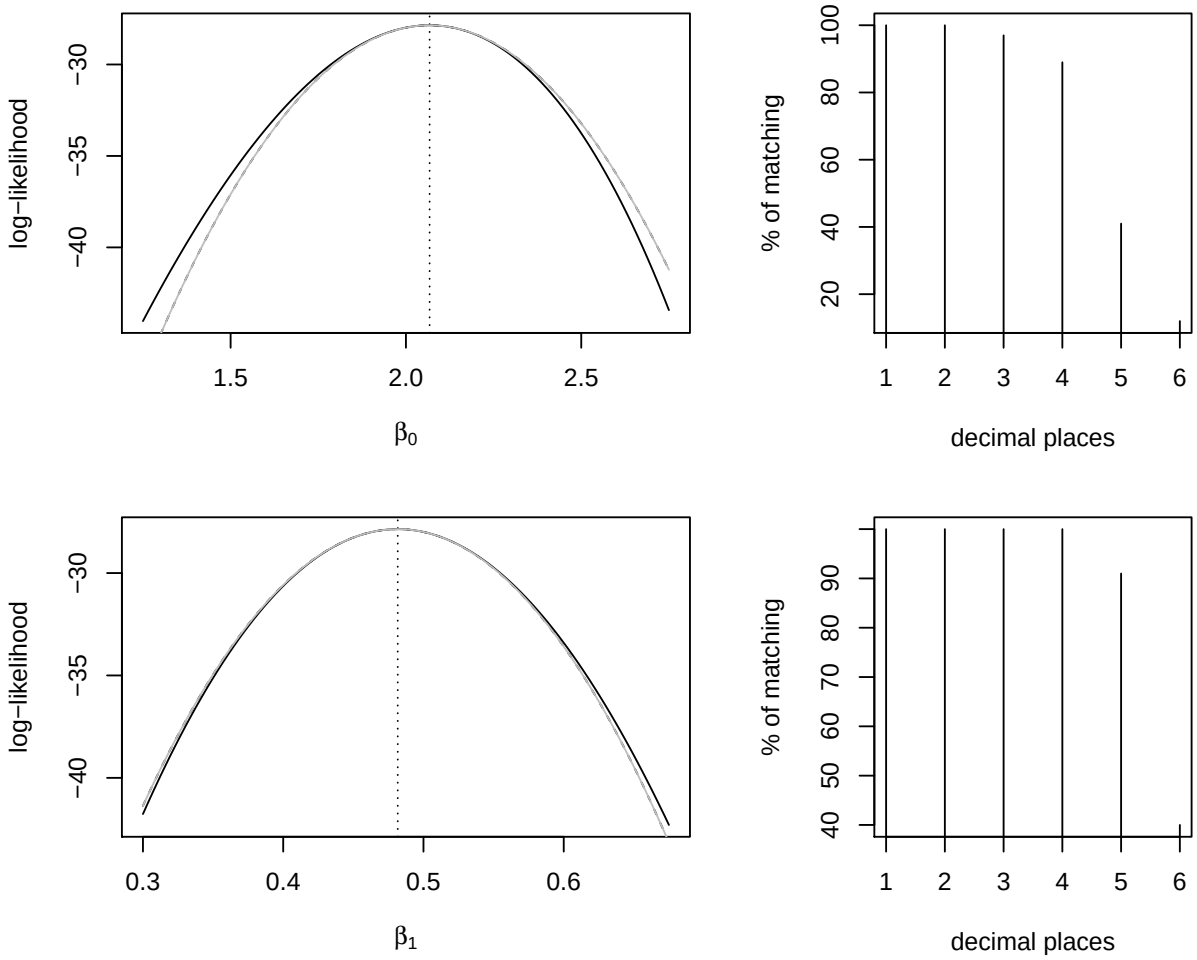

Figure 19: Profile log-likelihoods, solid line, for β_0 and β_1 in a Poisson regression. In dotted, MLEs. Quadratic approx.'s of the profiles in black dashed and gray, with a matching percentage between them considering six different decimal places.

In Figure 19 we see that the dashed matches with the gray line, i.e., the quadratic approximations are equal. To see better how equal they're, the right graphs of Figure 19 show the percentage of matching values (quadratic approximations of the profile log-likelihoods) considering different levels of rounding. As a conclusion statement, they're quite equal. This completes and complement the last paragraph.

A third and last note is that to compute the quadratic approx. for the estimated/conditional likelihood we use the corresponding element of the information matrix. To compute the quadratic approx. for the profile likelihood we use the inverse of that element in the variance-covariance matrix, that is the inverse of the information matrix.

The insights for this exercise was taken from `Pawitan`*'s book.*

Last modification on ...

```
[1] "2019-07-10 15:22:40 -03"
```