Introduction to Maximum Likelihood Inference

Contents

1 Introduction to Maximum Likelihood Inference

These notes are derived primarily from Dobson and Barnett (2018) (mostly chapters 1-5).

Some material was also taken from McLachlan and Krishnan (2007) and Casella and Berger (2002).

Configuring R

Functions from these packages will be used throughout this document:

```
library(conflicted) # check for conflicting function definitions
# library(printr) # inserts help-file output into markdown output
library(rmarkdown) # Convert R Markdown documents into a variety of formats.
library(pander) # format tables for markdown
library(ggplot2) # graphics
library(ggfortify) # help with graphics
library(dplyr) # manipulate data
library(tibble) # `tibble`s extend `data.frame`s
library(magrittr) # `%>%` and other additional piping tools
library(haven) # import Stata files
library(knitr) # format R output for markdown
library(tidyr) # Tools to help to create tidy data
library(plotly) # interactive graphics
library(dobson) # datasets from Dobson and Barnett 2018
library(parameters) # format model output tables for markdown
library(haven) # import Stata files
library(latex2exp) # use LaTeX in R code (for figures and tables)
library(fs) # filesystem path manipulations
library(survival) # survival analysis
library(survminer) # survival analysis graphics
library(KMsurv) # datasets from Klein and Moeschberger
library(parameters) # format model output tables for
library(webshot2) # convert interactive content to static for pdf
library(forcats) # functions for categorical variables ("factors")
library(stringr) # functions for dealing with strings
library(lubridate) # functions for dealing with dates and times
```

Here are some R settings I use in this document:

```
rm(list = ls()) # delete any data that's already loaded into R

conflicts_prefer(dplyr::filter)
ggplot2::theme_set(
  ggplot2::theme_bw() +
        # ggplot2::labs(col = "") +
    ggplot2::theme(
      legend.position = "bottom",
      text = ggplot2::element_text(size = 12, family = "serif")))

knitr::opts_chunk$set(message = FALSE)
options('digits' = 6)

panderOptions("big.mark", ",")
pander::panderOptions("table.emphasize.rownames", FALSE)
pander::panderOptions("table.split.table", Inf)
conflicts_prefer(dplyr::filter) # use the `filter()` function from dplyr() by default
legend_text_size = 9
run_graphs = TRUE
```

1.1 Overview of maximum likelihood estimation

1.1.1 The likelihood function

Definition 1.1 (Likelihood of a single observation)**.** Let X be a random variable and let x be X's observed data value. Let $\mathrm{p}_\Theta(X = x)$ be a probability model for the distribution of X, with parameter vector Θ.

Then the **likelihood** of parameter value θ, for model $\mathrm{p}_\Theta(X = x)$ and data $X = x$, is simply the probability of the event $X = x$ given $\Theta = \theta$:

$$\mathcal{L}(\theta) \stackrel{\text{def}}{=} \mathrm{P}_\theta(X = x)$$

Definition 1.2 (Likelihood of a dataset)**.** Let $\tilde{x} \stackrel{\text{def}}{=} x_1, \ldots, x_n$ be a dataset with corresponding random variable $\tilde{X}$. Let $\mathrm{p}_\Theta(\tilde{X})$ be a probability model for the distribution of $\tilde{X}$ with unknown parameter vector Θ.

Then the **likelihood** of parameter value θ, for model $\mathrm{p}_\Theta(X)$ and data $\tilde{X} = \tilde{x}$, is the *joint probability* of $\tilde{X} = \tilde{x}$ given $\Theta = \theta$:

$$\begin{aligned}\mathcal{L}(\theta) &\stackrel{\text{def}}{=} \mathrm{p}(\tilde{X} = \tilde{x} | \Theta = \theta) \\ &= \mathrm{p}(X_1 = x_1, ..., X_n = x_n | \Theta = \theta)\end{aligned}$$

> **i** Notation for the likelihood function
>
> The likelihood function can be written as:
> - $\mathcal{L}(\theta)$
> - $\mathcal{L}(\tilde{x}; \theta)$
> - $\mathcal{L}(\theta; \tilde{x})$
> - $\mathcal{L}_{\tilde{x}}(\theta)$
> - $\mathcal{L}_\theta(\tilde{x})$
> - $\mathcal{L}(\tilde{x}|\theta)$
>
> All of these notations mean the same thing. The parameter vector θ is often listed first or solely, either to emphasize that we are interested in how this function varies with the parameters, given the data, or possibly to make the likelihood resemble the Bayesian posterior probability $\mathrm{p}(\theta|\tilde{x})$, hinting at the fact that if the prior probability $\mathrm{p}(\theta)$ is uniform over some finite parameter space, the posterior probability is proportional to the likelihood:
>
> $$\begin{aligned}\mathrm{p}(\theta|\tilde{x}) &= \frac{\mathrm{p}(\tilde{x}|\theta)\mathrm{p}(\theta)}{\mathrm{p}(\tilde{x})} \\ &= \frac{\mathcal{L}(\tilde{x}|\theta)\mathrm{p}(\theta)}{\mathrm{p}(\tilde{x})} \\ &= \mathcal{L}(\tilde{x}|\theta)\frac{\mathrm{p}(\theta)}{\mathrm{p}(\tilde{x})}\end{aligned}$$

The likelihood is a function that takes θ (and implicitly, $\tilde{X}$) as inputs and outputs a single number, the joint probability of $\tilde{x}$ for model $p_\Theta(\tilde{X} = \tilde{x})$ with $\Theta = \theta$.

Theorem 1.1 (Likelihood of an independent sample)**.** *For mutually independent*[1] *data* $X_1, ..., X_n$*:*

$$\mathcal{L}(\tilde{x}|\theta) = \prod_{i=1}^{n} p(X_i = x_i|\theta) \tag{1}$$

---

[1] probability.qmd#def-indpt

Proof.

$$\begin{aligned}\mathcal{L}(\tilde{x}|\theta) &\stackrel{\text{def}}{=} \text{p}(X_1 = x_1, ..., X_n = x_n|\theta) \\ &= \prod_{i=1}^{n} \text{p}(X_i = x_i|\theta)\end{aligned}$$

The second equality is by the definition of statistical independence. □

Definition 1.3 (Likelihood components)**.** Given an iid dataset $\tilde{x}$, the **likelihood component** or **likelihood factor** of observation $X_i = x_i$ is the marginal likelihood of $X_i = x_i$:

$$\mathcal{L}_i(\theta) = \text{P}(X_i = x_i)$$

Theorem 1.2. *For iid data $\tilde{x} \stackrel{def}{=} x_1, \ldots, x_n$, the likelihood of the dataset is equal to the product of the observation-specific likelihood factors:*

$$\mathcal{L}(\theta) = \prod_{i=1}^{n} \mathcal{L}_i(\theta)$$

1.1.2 Binary outcomes models - one group, no covariates

$$\begin{aligned}\text{P}(Y = 1) &= \pi \\ \text{P}(Y = 0) &= 1 - \pi \\ \text{P}(Y = y) &= \pi^y (1 - \pi)^{1-y}\end{aligned}$$

Exercise 1.1. Let $\tilde{y}$ represent a data set of mutually independent binary outcomes, all with the same event probability π:

$$\begin{aligned}\tilde{y} &= (y_1, ..., y_n) \\ y_i &\sim_{\perp\!\!\!\perp} \text{Ber}(\pi)\end{aligned}$$

Write the likelihood of $\tilde{y}$.

Solution 1.1. For iid data $\tilde{y} = (y_1, ..., y_n)$:

$$\begin{aligned}\mathcal{L}(\pi; \tilde{y}) &= \text{P}(Y_1 = y_1, \ldots, Y_n = y_n) \\ &= \prod_{i=1}^{n} \mathcal{L}_i(\pi_i) \\ &= \prod_{i=1}^{n} \text{P}(Y_i = y_i) \\ &= \prod_{i=1}^{n} \pi^{y_i} (1 - \pi)^{1-y_i} \\ &= \pi^{\sum_{i=1}^{n} y_i} (1 - \pi)^{n - \sum_{i=1}^{n} y_i} \\ &= \pi^{\tilde{1} \cdot \tilde{y}} (1 - \pi)^{\tilde{1} \cdot (\tilde{1} - \tilde{y})}\end{aligned}$$

Exercise 1.2. Write the log-likelihood of $\tilde{y}$.

Solution 1.2.

$$\begin{aligned}
\ell(\pi, \tilde{y}) &= \left(\sum_{i=1}^{n} y_i\right) \log\{\pi\} + \left(n - \sum y_i\right) \log\{1-\pi\} \\
&= \left(\sum_{i=1}^{n} y_i\right) \left(\log\{\pi\} - \log\{1-\pi\}\right) + n \cdot \log\{1-\pi\} \\
&= \left(\sum_{i=1}^{n} y_i\right) \log\left\{\frac{\pi}{1-\pi}\right\} + n \cdot \log\{1-\pi\} \\
&= \left(\sum_{i=1}^{n} y_i\right) \text{logit}(\pi) + n \cdot \log\{1-\pi\}
\end{aligned}$$

1.1.3 The maximum likelihood estimate

Definition 1.4 (Maximum likelihood estimate)**.** The **maximum likelihood estimate** of a parameter vector Θ, denoted $\hat{\theta}_{\text{ML}}$, is the value of Θ that maximizes the likelihood:

$$\hat{\theta}_{\text{ML}} \stackrel{\text{def}}{=} \arg\max_{\Theta} \mathcal{L}(\Theta) \tag{2}$$

1.1.4 Finding the maximum of a function

Recall from calculus: the maxima of a continuous function $f(x)$ over a range of input values $\mathcal{R}(x)$ can be found either:

- at the edges of the range of input values, *OR*:
- where the function is flat (i.e. where the gradient function $f'(x) = 0$) *AND* the second derivative is negative definite ($f''(x) < 0$).

1.1.5 Directly maximizing the likelihood function for independent data

To find the maximizer(s) of the likelihood function, we need to solve $\mathcal{L}'(\theta) = 0$ for θ. However, even for mutually independent data, we quickly run into a problem:

$$\begin{aligned}
\mathcal{L}'(\theta) &= \frac{\partial}{\partial\theta} \mathcal{L}(\theta) \\
&= \frac{\partial}{\partial\theta} \prod_{i=1}^{n} p(X_i = x_i|\theta)
\end{aligned} \tag{3}$$

The derivative of the likelihood of independent data is the derivative of a product. To evaluate this derivative, we will have to perform a massive application of the product rule[2].

1.1.6 The log-likelihood function

It is typically easier to work with the log of the likelihood function:

Definition 1.5 (Log-likelihood)**.** The **log-likelihood** of parameter value θ, for model $\text{p}_{\Theta}(\tilde{X})$ and data $\tilde{X} = \tilde{x}$, is the natural logarithm of the likelihood[3]:

$$\ell \stackrel{\text{def}}{=} \log\{\mathcal{L}(\tilde{x}|\theta)\} \tag{4}$$

Theorem 1.3. *The likelihood and log-likelihood have the same maximizer:*

$$\arg\max_{\theta} \mathcal{L}(\theta) = \arg\max_{\theta} \ell(\theta)$$

[2] intro-MLEs.html#thm-product-rule

[3] https://en.wikipedia.org/wiki/Does_exactly_what_it_says_on_the_tin

Proof. Left to the reader. □

Theorem 1.4 (Log-likelihood of an independent sample)**.** *For mutually independent data $X_1, ..., X_n$ with shared distribution $p(X = x)$:*

$$\ell(x|\theta) = \sum_{i=1}^{n} \log p(X = x_i|\theta) \tag{5}$$

Proof.

$$\begin{aligned}\ell(x|\theta) &\stackrel{\text{def}}{=} \log \mathcal{L}(\tilde{x}|\theta) \\ &= \log \prod_{i=1}^{n} \mathrm{p}(X_i = x_i|\theta) \\ &= \sum_{i=1}^{n} \log p(X = x_i|\theta)\end{aligned}$$

□

For iid data, we will have a much easier time taking the derivative of the log-likelihood:

Theorem 1.5 (Derivative of the log-likelihood function for iid data)**.** *For iid data:*

$$\ell'(\theta) = \sum_{i=1}^{n} \frac{\partial}{\partial\theta} \log p(X = x_i|\theta) \tag{6}$$

Proof.

$$\begin{aligned}\ell'(\theta) &= \frac{\partial}{\partial\theta}\ell(\theta) \\ &= \frac{\partial}{\partial\theta} \sum_{i=1}^{n} \log \mathrm{p}(X = x_i|\theta) \\ &= \sum_{i=1}^{n} \frac{\partial}{\partial\theta} \log \mathrm{p}(X = x_i|\theta)\end{aligned}$$

□

1.1.7 The score function

The first derivative[4] of the log-likelihood, $\ell'(\theta)$, is important enough to have its own name: the *score function.*

Definition 1.6 (Score function)**.** The **score function** of a statistical model $\mathrm{p}(\tilde{X} = \tilde{x})$ is the gradient (i.e., first derivative) of the log-likelihood of that model:

$$\ell' \stackrel{\text{def}}{=} \frac{\partial}{\partial\theta}\ell(\tilde{x}|\theta) \tag{7}$$

We often skip writing the arguments x and/or θ, so $\ell' \stackrel{\text{def}}{=} \ell'(\tilde{x}|\theta) \stackrel{\text{def}}{=} \ell'(\theta)$.

Some statisticians use U or S instead of ℓ'. We will use ℓ', both to save U and S for other uses and to avoid introducing unnecessary notation to memorize.

Exercise 1.3. Derive the score function for a single Bernoulli random variable X. In other words, differentiate the marginal log-likelihood of a single Bernoulli random variable X with respect to the event probability parameter, π. Simplify as much as possible.

[4] a.k.a. the gradient[5]

Solution 1.3. Starting from Solution 1.2:

$$
\begin{aligned}
\ell' &\stackrel{\text{def}}{=} \frac{\partial}{\partial \pi}\ell \\
&= \frac{\partial}{\partial \pi}(x\log\{\pi\} + (1-x)\log\{1-\pi\}) \\
&= \frac{\partial}{\partial \pi}x\log\{\pi\} + \frac{\partial}{\partial \pi}(1-x)\log\{1-\pi\} \\
&= x\frac{\partial}{\partial \pi}\log\{\pi\} + (1-x)\frac{\partial}{\partial \pi}\log\{1-\pi\} \\
&= x\frac{1}{\pi} - (1-x)\frac{1}{1-\pi} \\
&= x\frac{1-\pi}{\pi(1-\pi)} - (1-x)\frac{\pi}{\pi(1-\pi)} \\
&= \frac{x(1-\pi) - (1-x)\pi}{\pi(1-\pi)} \\
&= \frac{x - x\pi - \pi + x\pi}{\pi(1-\pi)} \\
&= \frac{x-\pi}{\pi(1-\pi)} \\
&= \frac{x-\mu}{\pi(1-\pi)} \\
&= \frac{\varepsilon}{\text{Var}(X)}
\end{aligned}
$$

Exercise 1.4. Derive the score function for a single Poisson random variable X.

Solution 1.4. The score function is the first derivative of the log-likelihood:

$$
\begin{aligned}
\ell' &= \frac{\partial}{\partial \lambda}(x\log\lambda - \lambda - \log x!) \\
&= \frac{\partial}{\partial \lambda}x\log\lambda - \frac{\partial}{\partial \lambda}n\lambda - \frac{\partial}{\partial \lambda}\log x! \\
&= x\frac{\partial}{\partial \lambda}\log\lambda - n\frac{\partial}{\partial \lambda}\lambda - \frac{\partial}{\partial \lambda}\log x! \\
&= x\frac{1}{\lambda} - 1 - 0 \\
&= \frac{1}{\lambda}x - 1 \\
&= \frac{x}{\lambda} - \frac{\lambda}{\lambda} \\
&= \frac{x-\lambda}{\lambda} \\
&= \frac{x-\mu}{\lambda} \\
&= \frac{\varepsilon}{\text{Var}(X)}
\end{aligned}
$$

Exercise 1.5. Derive the score function for a single Gaussian random variable X, with respect to the mean parameter μ.

Solution 1.5. The score function is the first derivative of the log-likelihood:

$$\begin{aligned}
\ell' &\stackrel{\text{def}}{=} \frac{\partial}{\partial \mu}\ell \\
&= \frac{\partial}{\partial \mu}\left(\frac{-1}{2}\left(\log\{2\pi\sigma^2\} + \frac{\varepsilon^2}{\sigma^2}\right)\right) \\
&= \frac{-1}{2}\frac{\partial}{\partial \mu}\left(\log\{2\pi\sigma^2\} + \frac{\varepsilon^2}{\sigma^2}\right) \\
&= \frac{-1}{2}\left(\frac{\partial}{\partial \mu}\log\{2\pi\sigma^2\} + \frac{\partial}{\partial \mu}\frac{\varepsilon^2}{\sigma^2}\right) \\
&= \frac{-1}{2}\left(0 + \frac{\partial}{\partial \mu}\frac{(x-\mu)^2}{\sigma^2}\right) \\
&= \frac{-1}{2}\left(-2\frac{x-\mu}{\sigma^2}\right) \\
&= \frac{x-\mu}{\sigma^2} \\
&= \frac{\varepsilon}{\text{Var}(X)}
\end{aligned}$$

Exercise 1.6. Derive the score function for a single exponential random variable X, with respect to the mean parameter μ.

Solution 1.6. The score function is the first derivative of the log-likelihood:

$$\begin{aligned}
\ell' &\stackrel{\text{def}}{=} \frac{\partial}{\partial \mu}\ell \\
&= \frac{\partial}{\partial \mu}\left(-\log\{\mu\} - \frac{x}{\mu}\right) \\
&= \frac{\partial}{\partial \mu}(-\log\{\mu\}) - \frac{\partial}{\partial \mu}\frac{x}{\mu} \\
&= -\frac{1}{\mu} + \frac{x}{\mu^2} \\
&= -\frac{\mu}{\mu^2} + \frac{x}{\mu^2} \\
&= \frac{x-\mu}{\mu^2} \\
&= \frac{\varepsilon}{\text{Var}(X)}
\end{aligned}$$

In all four cases above, the score function (with respect to the mean) turned out to be:

$$\ell' = \frac{\varepsilon}{\text{Var}(X)}$$

That is no coincidence. All four of these distributions belong to the exponential family/class of probability distributions[6]. The distributions in the exponential family share many special properties. For more details, see Hogg, Tanis, and Zimmerman (2015), Section 6.7 and Dobson and Barnett (2018), Chapter 3.

[6] https://en.wikipedia.org/wiki/Exponential_family

1.1.8 Asymptotic distribution of the maximum likelihood estimate

We learned how to quantify our uncertainty about these maximum likelihood estimates; with sufficient sample size, $\hat{\theta}_{\mathrm{ML}}$ has an approximately Gaussian distribution (Newey and McFadden 1994):

Theorem 1.6 (Central limit theorem for MLEs)**.**

$$\hat{\theta}_{ML} \dot{\sim} N\Big(\theta, \big[\mathcal{I}(\tilde{\theta})\big]^{-1}\Big) \tag{8}$$

Proof. See (Lehmann 1999), Theorem 7.3.2. □

Recall:

Definition 1.7 (Observed information matrix)**.** The **observed information matrix**, denoted I, is defined as the negative of the Hessian of the log-likelihood:

$$I \stackrel{\text{def}}{=} -\ell''(\tilde{x}|\tilde{\theta}) \tag{9}$$

Definition 1.8 (Expected information/Fisher information)**.** The **expected information matrix**, also known as the **Fisher information matrix**, is denoted $\mathcal{I}$ and is defined as the expected value of the observed information matrix:

$$\mathcal{I} \stackrel{\text{def}}{=} \mathrm{E}[I(\tilde{x}|\theta)] \tag{10}$$

We can estimate $\mathcal{I}(\theta)$ using either $\mathcal{I}(\hat{\theta}_{\mathrm{ML}})$ or $I(\tilde{x}; \hat{\theta}_{\mathrm{ML}})$.

So we can estimate the standard error of $\hat{\theta}_k$ as:

$$\widehat{\mathrm{SE}}\big(\hat{\theta}_k\big) = \sqrt{\Big[\big(\hat{\mathcal{I}}\big(\hat{\theta}_{\mathrm{ML}}\big)\big)^{-1}\Big]_{kk}}$$

1.1.9 The (Fisher) (expected) information matrix

The variance of $\ell'(x, \theta)$, $Cov\{\ell'(x, \theta)\}$, is also very important; we call it the "expected information matrix", "Fisher information matrix", or just "information matrix", and we represent it using the symbol $\mathcal{I}(I)$ (`\scriptI` in Unicode, `\mathcal{I}` in LaTeX).

$$\begin{aligned}\mathcal{I} &\stackrel{\text{def}}{=} \mathcal{I}(\theta) \\ &\stackrel{\text{def}}{=} \mathrm{Cov}(\ell'|\theta) \\ &= \mathrm{E}[\ell'\ell'^{\top}] - \mathrm{E}[\ell']\ \mathrm{E}[\ell']^{\top}\end{aligned}$$

The elements of $\mathcal{I}$ are:

$$\begin{aligned}\mathcal{I}_{ij} &\stackrel{\text{def}}{=} \mathrm{Cov}\big(\ell'_i, \ell'_j\big) \\ &= \mathrm{E}[\ell'_i\ell'_j] - \mathrm{E}[\ell'_i]\mathrm{E}[\ell'_j]\end{aligned}$$

Here,

$$
\begin{aligned}
\mathrm{E}[\ell'] &\stackrel{\text{def}}{=} \int_{x\in\mathcal{R}(x)} \ell'(x,\theta)\mathrm{p}(X=x|\theta)dx \\
&= \int_{x\in\mathcal{R}(X)} \Big(\frac{\partial}{\partial\theta}\log \mathrm{p}(X=x|\theta)\Big)\mathrm{p}(X=x|\theta)dx \\
&= \int_{x\in\mathcal{R}(X)} \frac{\frac{\partial}{\partial\theta}\mathrm{p}(X=x|\theta)}{\mathrm{p}(X=x|\theta)}\mathrm{p}(X=x|\theta)dx \\
&= \int_{x\in\mathcal{R}(X)} \frac{\partial}{\partial\theta}\mathrm{p}(X=x|\theta)dx
\end{aligned}
$$

And similarly

$$
\mathrm{E}[\ell'\ell'^{\top}] \stackrel{\text{def}}{=} \int_{x\in R(x)} \ell'(x,\theta)\ell'(x,\theta)^{\top}\ \mathrm{p}(X=x|\theta)\ dx
$$

Note that $\mathrm{E}[\ell']$ and $\mathrm{E}\left[\ell'\ell'^{\top}\right]$ are functions of θ but not of x; the expectation operator removed x.

Also note that for most of the distributions you are familiar with (including Gaussian, binomial, Poisson, exponential):

$$
\mathrm{E}[\ell'] = 0
$$

So

$$
\mathcal{J}(\theta) = \mathrm{E}\left[\ell'\ell'^{\top}\right]
$$

Moreover, for those distributions (called the "exponential family"), we have:

$$
\mathcal{J} = -\mathrm{E}[\ell''] = \mathrm{E}[-\ell'\ell'^{\top}]
$$

(see Dobson and Barnett (2018), §3.17).

Definition 1.9 (Hessian)**.** The matrix of second derivatives of the log-likelihood function is called the **Hessian matrix (of the log-likelihood function)** [7]:

$$
\ell'' \stackrel{\text{def}}{=} \frac{\partial}{\partial\tilde{\theta}}\frac{\partial}{\partial\tilde{\theta}^{\top}}\ell(\tilde{x}|\tilde{\theta}) \tag{11}
$$

Theorem 1.7 (Elements of the Hessian matrix)**.** *If $\tilde{\theta}$ is a $p\times 1$ vector, then the Hessian is a $p\times p$ matrix, whose ij^{th} entry is:*

$$
\ell''_{ij} = \frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}\ell(\tilde{X}=\tilde{x}|\tilde{\theta}) \tag{12}
$$

Theorem 1.8 (Hessian = derivative of transposed score)**.**

$$
\ell''(\tilde{x}|\tilde{\theta}) = \frac{\partial}{\partial\tilde{\theta}}\big(\ell'(\tilde{x}|\tilde{\theta})\big)^{\top}
$$

---

[7] named after mathematician Otto Hesse[8]

Observed information

Sometimes, we use $I(\theta; x) \stackrel{\text{def}}{=} -hess$ (note the standard-font "I" here). $I(\theta; x)$ is the observed information, precision, or concentration matrix (Negative Hessian).

> **!** Key point
>
> The asymptotics of MLEs gives us $\hat{\theta}_{ML} \sim N(\theta, \mathfrak{I}^{-1}(\theta))$, approximately, for large sample sizes.

We can estimate $\mathcal{I}^{-1}(\theta)$ by working out $\mathrm{E}[-\ell'']$ or $\mathrm{E}\left[\ell'\ell'^{\top}\right]$ and plugging in $\hat{\theta}_{\mathrm{ML}}$, but sometimes we instead use $I(\hat{\theta}_{\mathrm{ML}}, \tilde{x})$ for convenience; there are some cases where it's provably better according to some criteria (Efron and Hinkley (1978)).

1.1.10 Quantifying (un)certainty of MLEs

Confidence intervals for MLEs

An asymptotic approximation of a 95% confidence interval for θ_k is

$$\hat{\theta}_{\mathrm{ML}} \pm z_{0.975} \times \widehat{\mathrm{SE}}\left(\hat{\theta}_k\right)$$

where z_β the β quantile of the standard Gaussian distribution.

p-values and hypothesis tests for MLEs

(to add)

Likelihood ratio tests for MLEs

log(likelihood ratio) tests (c.f. Dobson and Barnett 2018, sec. 5.7):

$$2(\ell - \ell_{\mathrm{H0}}) \sim \chi^2(p - q)$$

See also https://online.stat.psu.edu/stat504/book/export/html/657

Prediction intervals for MLEs

$$\overline{X} \in [\hat{\mu} \pm z_{1-\alpha/2}\frac{\sigma}{m}]$$

Where m is the sample size of the new data to be predicted (typically 1, except for binary outcomes, where it needs to be bigger for prediction intervals to make sense).

1.2 Example: Maximum likelihood for Tropical Cyclones in Australia

(Adapted from Dobson and Barnett (2018) §1.6.5)

1.2.1 Data

The `cyclones` dataset in the `dobson` package (Table 1) records the number of tropical cyclones in Northeastern Australia during 13 November-to-April cyclone seasons (more details in Dobson and Barnett (2018) §1.6.5 and `help(cyclones, package = "dobson")`). Figure 1 graphs the number of cyclones (y-axis) by season (x-axis). Let's use Y_i to represent these counts, where i is an indexing variable for the seasons and Y_i is the number of cyclones in season i.

1.2.2 Exploratory analysis

Suppose we want to learn about how many cyclones to expect per season.

```
library(dobson)
library(dplyr)
data(cyclones)
library(pander)
pander(cyclones |> relocate(season, .before = everything()))
```

Table 1: Number of tropical cyclones during a season from November to April in Northeastern Australia

season	years	number
1	1956/7	6
2	1957/8	5
3	1958/9	4
4	1959/60	6
5	1960/1	6
6	1961/2	3
7	1962/3	12
8	1963/4	7
9	1964/5	4
10	1965/6	2
11	1966/7	6
12	1967/8	7
13	1968/9	4

```r
library(ggplot2)
library(dplyr)
cyclones |>
  mutate(years = factor(years, levels = years)) |>
  ggplot(aes(x = years, y = number, group = 1)) +
  geom_point() +
  geom_line() +
  xlab("Season") +
  ylab("Number of cyclones") +
  expand_limits(y = 0) +
  theme(axis.text.x = element_text(vjust = .5, angle = 45))
```

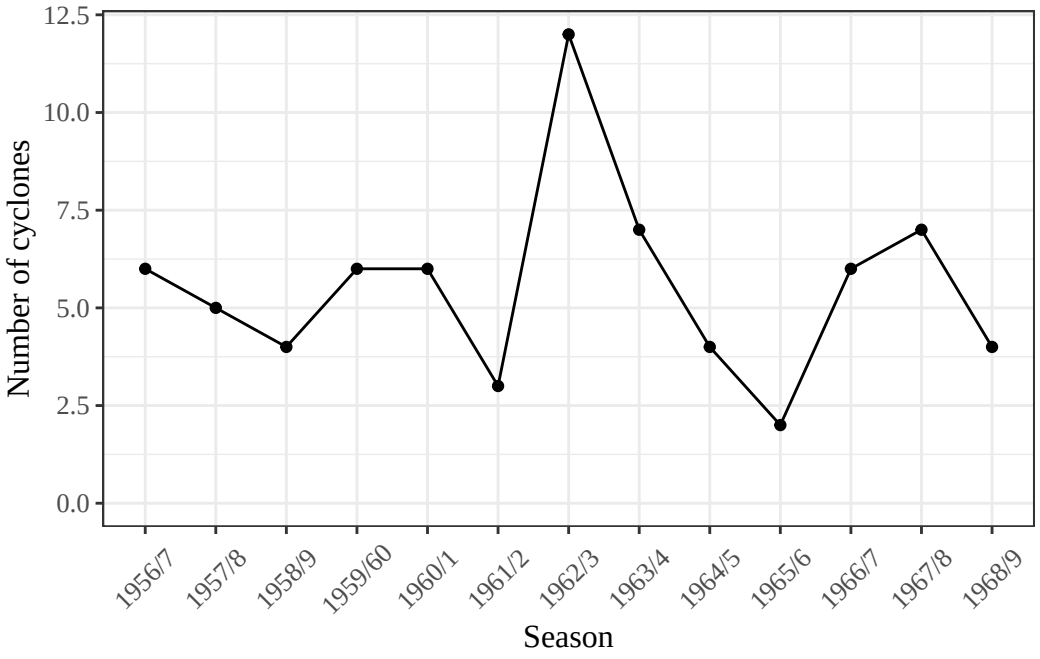


Figure 1: Number of tropical cyclones per season in northeastern Australia, 1956-1969

There's no obvious correlation between adjacent seasons, so let's assume that each season is independent of the others.

Let's also assume that they are identically distributed; let's denote this distribution as $P(Y = y)$. Note that there's no index i in this expression, since we are assuming the Y_is are identically distributed.

We can visualize the distribution using a bar plot (Figure 2).

```r
cyclones |>
  ggplot() +
  geom_histogram(aes(x = number)) +
  expand_limits(x = 0) +
  xlab("Number of cyclones") +
  ylab("Count (number of seasons)")
```

Table 2: Summary statistics for `cyclones` data

	Overall (N=13)
number	
Mean (SD)	5.54 (2.47)
Median [Min, Max]	6.00 [2.00, 12.0]

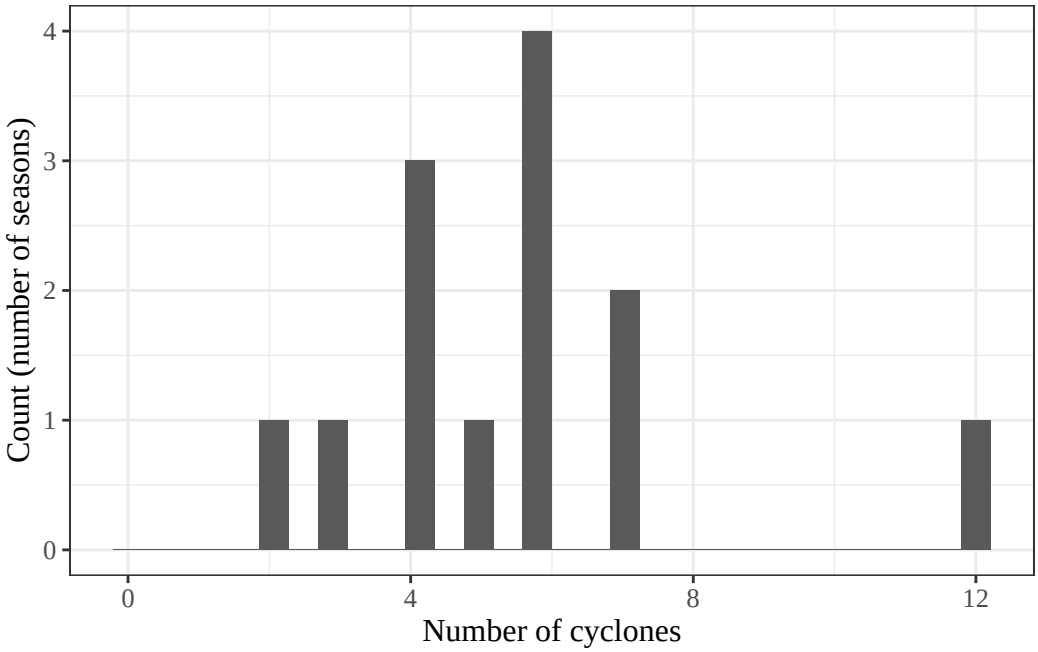


Figure 2: Bar plot of cyclones per season

Table 2 provides summary statistics.

```
n <- nrow(cyclones)
sumx <- cyclones |>
  pull(number) |>
  sum()
xbar <- cyclones |>
  pull(number) |>
  mean()

cyclones |> table1::table1(x = ~number)
```

1.2.3 Model

We want to estimate $P(Y = y)$; that is, $P(Y = y)$ is our estimand[9].

We could estimate $P(Y = y)$ for each value of y in $0 : \infty$ separately ("nonparametrically") using the fraction of our data with $Y_i = y$, but then we would be estimating an infinitely large set of parameters, and we would have low precision. We will probably do better with a parametric model.

[9] estimation.qmd#def-estimand

Exercise 1.7. What parametric probability distribution family might we use to model this empirical distribution?

Solution. Let's use the Poisson. The Poisson distribution is appropriate for this data , because the data are counts that could theoretically take any integer value (discrete) in the range $0 : \infty$. Visually, the plot of our data closely resembles a Poisson or binomial distribution. Since cyclones do not have an "upper limit" on the number of events we could potentially observe in one season, the Poisson distribution is more appropriate than the binomial.

Exercise 1.8. Write down the Poisson distribution's probability mass function.

Solution.

$$P(Y = y) = \frac{\lambda^y e^{-\lambda}}{y!} \tag{13}$$

1.2.4 Estimating the model parameters using maximum likelihood

Now, we can estimate the parameter λ for this distribution using maximum likelihood estimation.

Exercise 1.9 (What is the likelihood?)**.** Write down the likelihood (probability mass function or probability density function) of a single observation x, according to your model.

Solution.

$$\begin{aligned}\mathcal{L}(\lambda; x) &= p(X = x|\Lambda = \lambda) \\ &= \frac{\lambda^x e^{-\lambda}}{x!}\end{aligned}$$

Exercise 1.10. Write down the vector of parameters in your model.

Solution. There is only one parameter, λ:

$$\theta = (\lambda)$$

Exercise 1.11. Write down the population mean and variance of a single observation from your chosen probability model, as a function of the parameters (extra credit - derive them).

Solution.

- Population mean: $\mathrm{E}[X] = \lambda$
- Population variance: $\mathrm{Var}(X) = \lambda$

Exercise 1.12. Write down the likelihood of the full dataset.

Solution.

$$\begin{aligned}\mathcal{L}(\lambda; \tilde{x}) &= \mathrm{P}(\tilde{X} = \tilde{x}) \\ &= \mathrm{P}(X_1 = x_1, X_2 = x_2, ..., X_{13} = x_{13}) \\ &= \prod_{i=1}^{13} \mathrm{P}(X_i = x_i) \\ &= \prod_{i=1}^{13} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!}\end{aligned}$$

Exercise 1.13. Graph the likelihood as a function of λ.

Solution.

```
lik <- function(lambda, y = cyclones$number, n = length(y)) {
  lambda^sum(y) * exp(-n * lambda) / prod(factorial(y))
}

library(ggplot2)
lik_plot <-
  ggplot() +
  geom_function(fun = lik, n = 1001) +
  xlim(min(cyclones$number), max(cyclones$number)) +
  ylab("likelihood") +
  xlab("lambda")

print(lik_plot)
```

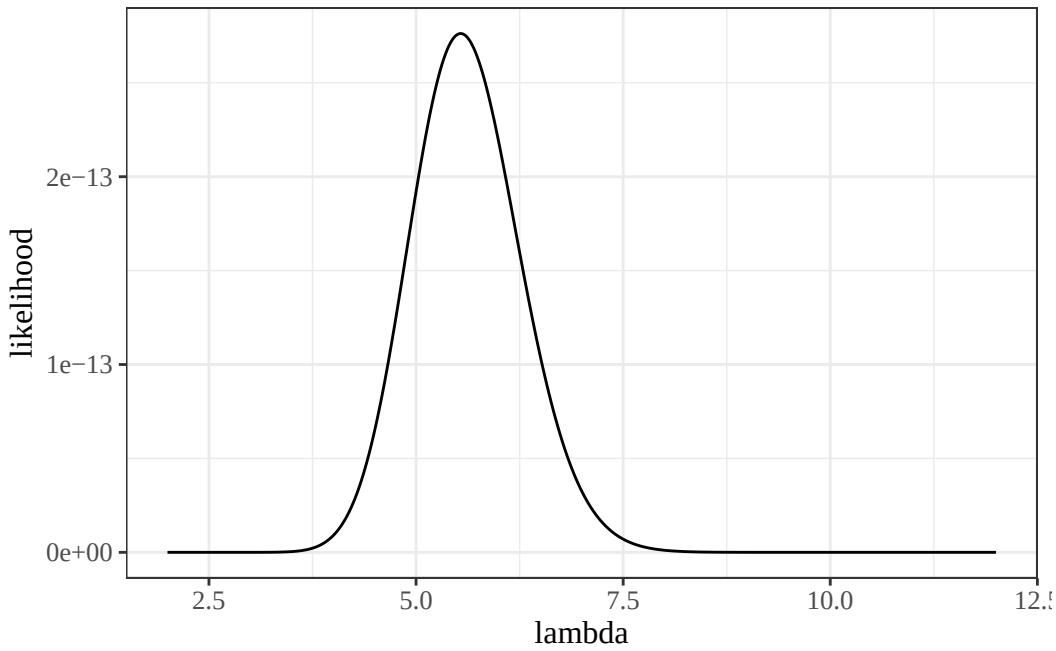


Figure 3: Likelihood of Dobson cyclone data

Exercise 1.14. Write down the log-likelihood of the full dataset.

Solution.

$$
\begin{aligned}
\ell(\lambda; \tilde{x}) &= \log \mathcal{L}(\lambda; \tilde{x}) \\
&= \log \prod_{i=1}^{n} \frac{\lambda^{x_i} \mathrm{e}^{-\lambda}}{x_i!} \\
&= \sum_{i=1}^{n} \log \frac{\lambda^{x_i} \mathrm{e}^{-\lambda}}{x_i!} \\
&= \sum_{i=1}^{n} \log \lambda^{x_i} + \log \mathrm{e}^{-\lambda} - \log x_i! \\
&= \sum_{i=1}^{n} x_i \log \lambda - \lambda - \log x_i! \\
&= \sum_{i=1}^{n} x_i \log \lambda - \sum_{i=1}^{n} \lambda - \sum_{i=1}^{n} \log x_i! \\
&= \sum_{i=1}^{n} x_i \log \lambda - n\lambda - \sum_{i=1}^{n} \log x_i!
\end{aligned}
$$

Exercise 1.15. Graph the log-likelihood as a function of λ.

Solution.

```
loglik <- function(lambda, y = cyclones$number, n = length(y)) {
  sum(y) * log(lambda) - n * lambda - sum(log(factorial(y)))
}

ll_plot <- ggplot() +
  geom_function(fun = loglik, n = 1001) +
  xlim(min(cyclones$number), max(cyclones$number)) +
  ylab("log-likelihood") +
  xlab("lambda")
ll_plot
```

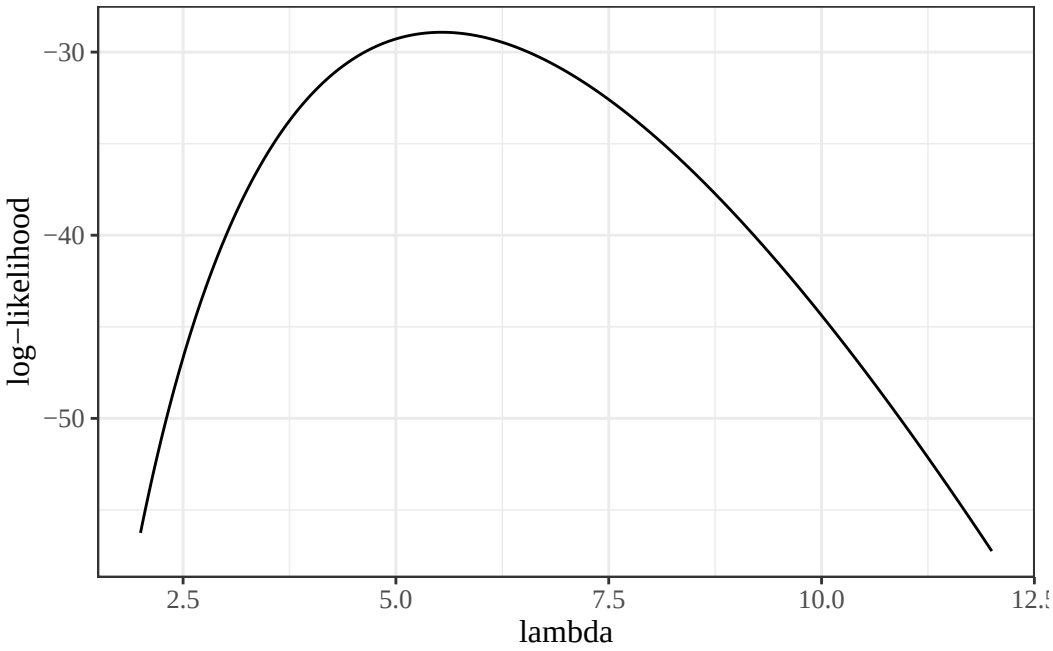


Figure 4: log-likelihood of Dobson cyclone data

The score function

Exercise 1.16. Derive the score function for the dataset.

Solution. The score function is the first derivative of the log-likelihood:

$$\begin{aligned}
\ell'(\lambda; \tilde{x}) &= \frac{\partial}{\partial \lambda}\left(\sum_{i=1}^{n} x_i \log \lambda - n\lambda - \sum_{i=1}^{n} \log x_i!\right) \\
&= \frac{\partial}{\partial \lambda}\sum_{i=1}^{n} x_i \log \lambda - \frac{\partial}{\partial \lambda} n\lambda - \frac{\partial}{\partial \lambda}\sum_{i=1}^{n} \log x_i! \\
&= \sum_{i=1}^{n} x_i \frac{\partial}{\partial \lambda} \log \lambda - n\frac{\partial}{\partial \lambda}\lambda - \sum_{i=1}^{n} \frac{\partial}{\partial \lambda} \log x_i! \\
&= \sum_{i=1}^{n} x_i \frac{1}{\lambda} - n - 0 \\
&= \frac{1}{\lambda}\sum_{i=1}^{n} x_i - n \\
&= \left(\frac{1}{\lambda} n\bar{x}\right) - n \\
&= \left(\frac{1}{\lambda} 72\right) - 13
\end{aligned}$$

Exercise 1.17. Graph the score function.

Solution.

```
score <- function(lambda, y = cyclones$number, n = length(y)) {
  (sum(y) / lambda) - n
}

ggplot() +
  geom_function(fun = score, n = 1001) +
  xlim(min(cyclones$number), max(cyclones$number)) +
  ylab("l'(lambda)") +
  xlab("lambda") +
  geom_hline(yintercept = 0, col = "red")
```

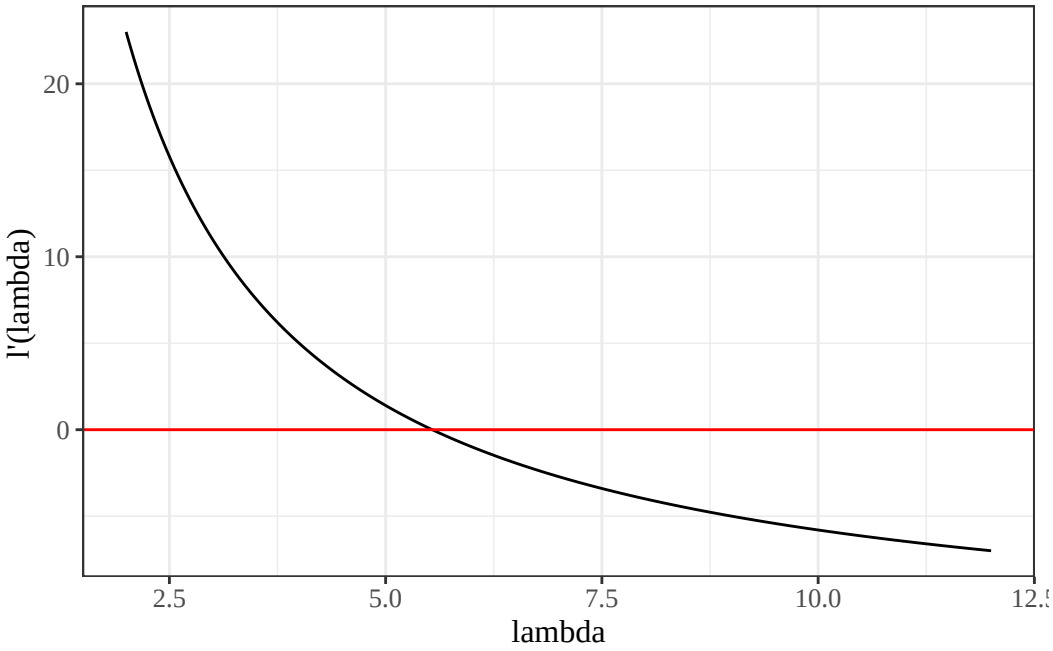


Figure 5: score function of Dobson cyclone data

The Hessian matrix

Exercise 1.18. Derive the Hessian matrix.

Solution. The Hessian function for an iid sample is the 2nd derivative(s) of the log-likelihood:

$$\begin{aligned}
\ell''(\lambda; \tilde{x}) &= \frac{\partial}{\partial \lambda}\left(\frac{1}{\lambda}\sum_{i=1}^{n} x_i - n\right) \\
&= \frac{\partial}{\partial \lambda}\frac{1}{\lambda}\sum_{i=1}^{n} x_i - \frac{\partial}{\partial \lambda} n \\
&= -\frac{1}{\lambda^2}\sum_{i=1}^{n} x_i \\
&= -\frac{1}{\lambda^2} n\bar{x} \\
&= -\frac{1}{\lambda^2} \cdot 72
\end{aligned}$$

Exercise 1.19. Graph the Hessian.

Solution.

```
hessian <- function(lambda, y = cyclones$number, n = length(y)) {
  -sum(y) / (lambda^2)
}

ggplot() +
  geom_function(fun = hessian, n = 1001) +
  xlim(min(cyclones$number), max(cyclones$number)) +
```

```
  ylab("l''(lambda)") +
  xlab("lambda") +
  geom_hline(yintercept = 0, col = "red")
```

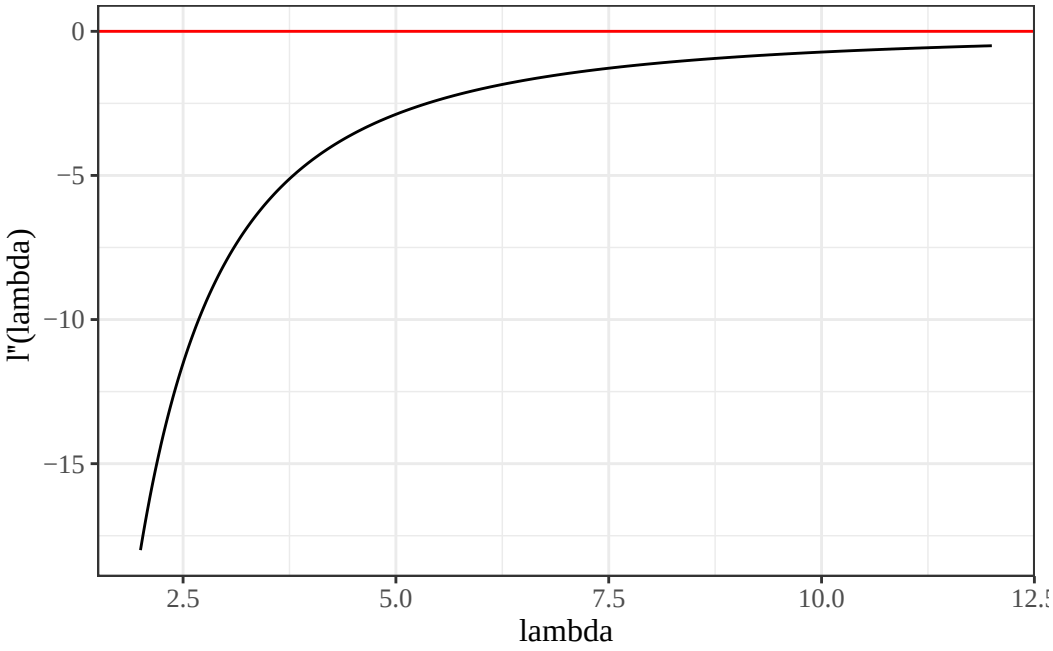


Figure 6: Hessian function of Dobson cyclone data

Exercise 1.20. Write the score equation (estimating equation).

Solution.

$$\ell'(\lambda; \tilde{x}) = 0$$

1.2.5 Finding the MLE analytically

In this case, we can find the MLE of λ by solving the score equation for λ analytically (using algebra):

Exercise 1.21. Solve the estimating equation for λ:

Solution.

$$\begin{aligned} 0 &= \frac{1}{\lambda}\sum_{i=1}^{n} x_i - n \\ n &= \frac{1}{\lambda}\sum_{i=1}^{n} x_i \\ n\lambda &= \sum_{i=1}^{n} x_i \\ \lambda &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ &= \bar{x} \end{aligned}$$

Let's call this solution of the estimating equation $\tilde{\lambda}$ for now:

$$\tilde{\lambda} \stackrel{\text{def}}{=} \bar{x}$$

Exercise 1.22. Confirm that the Hessian $\ell''(\lambda; \tilde{x})$ is negative when evaluated at $\tilde{\lambda}$.

Solution.

$$\begin{aligned} \ell''(\tilde{\lambda}; \tilde{x}) &= -\frac{1}{\tilde{\lambda}^2} n\bar{x} \\ &= -\frac{1}{\bar{x}^2} n\bar{x} \\ &= -\frac{n}{\bar{x}} \\ &< 0 \end{aligned}$$

Exercise 1.23. Draw conclusions about the MLE of λ.

Solution. Since $\ell''(\tilde{\lambda}; \tilde{x}) < 0$, $\tilde{\lambda}$ is at least a local maximizer of the likelihood function $\mathcal{L}(\lambda)$. Since there is only one solution to the estimating equation and the Hessian is negative definite everywhere, $\tilde{\lambda}$ must also be the global maximizer of $\mathcal{L}(\lambda; \tilde{x})$:

```
mle <- mean(cyclones$number)
```

$$\hat{\lambda}_{\text{ML}} = \bar{x} = 5.538462$$

Exercise 1.24. Graph the log-likelihood with the MLE superimposed.

Solution.

```
library(dplyr)

mle_data <- tibble(x = mle, y = loglik(mle))
ll_plot + geom_point(data = mle_data, aes(x = x, y = y), col = "red")
```

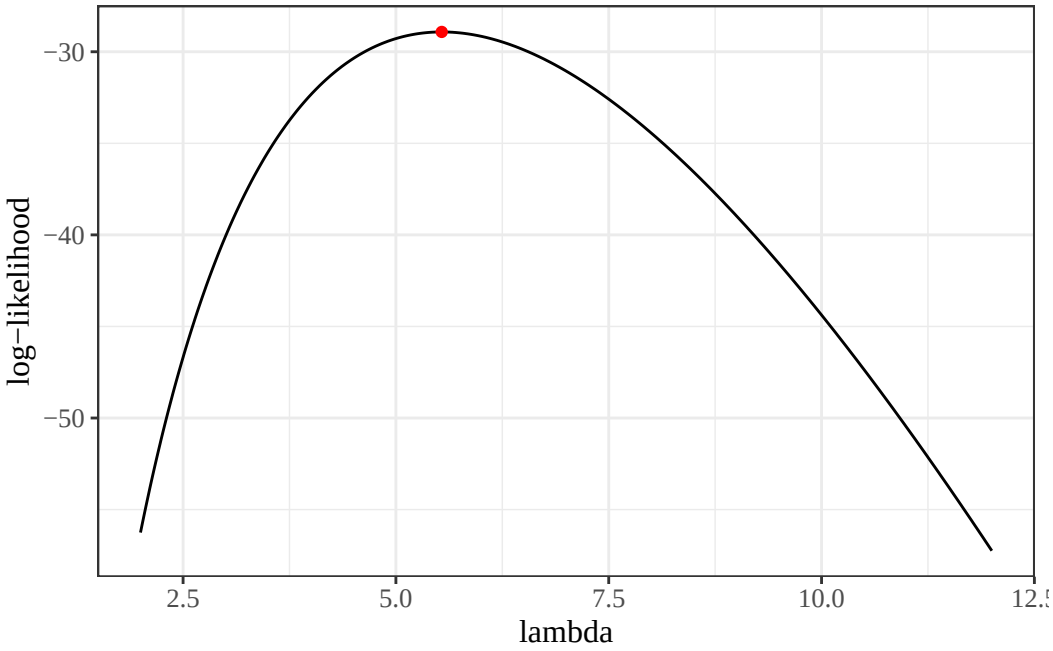


Figure 7: log-likelihood of Dobson cyclone data with MLE

Information matrices

```r
obs_inf <- function(...) -hessian(...)
ggplot() +
  geom_function(fun = obs_inf, n = 1001) +
  xlim(min(cyclones$number), max(cyclones$number)) +
  ylab("I(lambda)") +
  xlab("lambda") +
  geom_hline(yintercept = 0, col = "red")
```

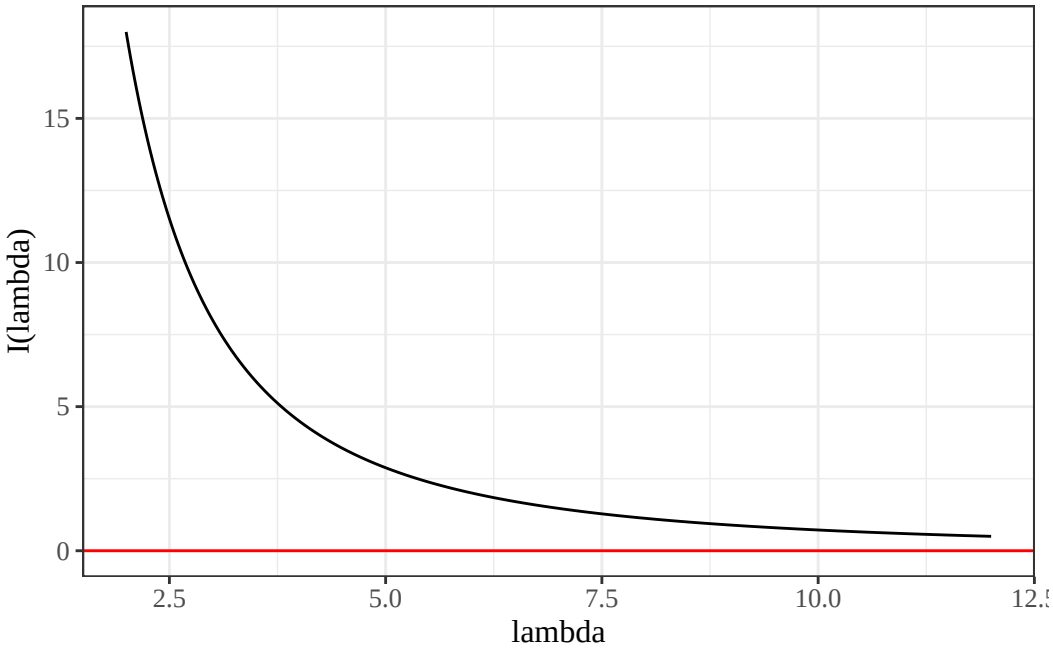


Figure 8: Observed information function of Dobson cyclone data

1.3 Finding the MLE using the Newton-Raphson algorithm

1.3.1 Iterative maximization

(c.f., Dobson and Barnett (2018), Chapter 4)

Later, when we are trying to find MLEs for likelihoods which we can't easily differentiate, we will "hill-climb" using the Newton-Raphson algorithm:

$$\begin{aligned}\hat{\theta}^* &\leftarrow \hat{\theta}^* + \left(I\left(\tilde{y};\hat{\theta}^*\right)\right)^{-1}\ell'\left(\tilde{y};\hat{\theta}^*\right) \\ &= \hat{\theta}^* - \left(\ell''\left(\tilde{y};\hat{\theta}^*\right)\right)^{-1}\ell'\left(\tilde{y};\hat{\theta}^*\right)\end{aligned}$$

The reasoning for this algorithm is that we can approximate the score function near $\hat{\theta}^*$ using the first-order Taylor polynomial[10]:

$$\begin{aligned}\ell'(\theta) &\approx \ell'^*(\theta) \\ &\stackrel{\text{def}}{=} \ell'(\hat{\theta}^*) + \ell''(\hat{\theta}^*)(\theta - \hat{\theta}^*)\end{aligned}$$

The approximate score function, $\ell'^*(\theta)$, is a linear function of θ, so it is easy to solve the corresponding approximate score equation, $\ell'^*(\theta) = 0$, for θ:

$$\theta = \hat{\theta}^* - \ell'(\hat{\theta}^*) \cdot \left(\ell''(\hat{\theta}^*)\right)^{-1}$$

For computational simplicity, we will sometimes use $\mathfrak{I}^{-1}(\theta)$ in place of $I\left(\hat{\theta}, y\right)$; doing so is called "Fisher scoring" or the "method of scoring". Note: this substitution is the opposite of the substitution that we are making for estimating the variance of the MLE; this time we should technically use the observed information but we use the expected information instead.

[10] https://en.wikipedia.org/wiki/Taylor%27s_theorem

There's also an "empirical information matrix" (see McLachlan and Krishnan (2007)):

$$I_e(\theta, y) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \ell_i' \ \ell_i'^{\top} - \frac{1}{n} \ell' \ell'^{\top}$$

where ℓ_i is the log-likelihood of the ith observation. Note that $\ell' = \sum_{i=1}^{n} \ell_i'$.

$\frac{1}{n} I_e(\theta, y)$ is the sample equivalent of

$$\mathfrak{I} \stackrel{\text{def}}{=} \mathfrak{I}(\theta) \stackrel{\text{def}}{=} Cov\left(\ell' | \theta\right) = E[\ell' \ell'^{\top}] - E[\ell'] \ E[\ell']^{\top}$$

$$\left\{ \mathfrak{I}_{jk} \stackrel{\text{def}}{=} Cov\left(\ell'_j, \ell'_k\right) = E[\ell'_j \ell'_k] - E[\ell'_j] E[\ell'_k] \right\}$$

$I_e(\theta, y)$ is sometimes computationally easier to compute for Newton-Raphson-type maximization algorithms.

c.f. https://en.wikipedia.org/wiki/Newton%27s_method_in_optimization

---

Example 1.1 (Finding the MLE using the Newton-Raphson algorithm)**.**

We found that the MLE was $\hat{\lambda} = \bar{x}$, by solving the score equation $\ell'(\lambda) = 0$ for λ.

What if we hadn't been able to solve the score equation?

Then we could start with some initial guess $\hat{\lambda}^*$, such as $\hat{\lambda}^* = 3$, and use the Newton-Raphson algorithm.

```
# specify initial guess:
cur_lambda_est <- 3
```

In Exercise 1.16, we found that the score function was:

$$\ell'(\lambda; \tilde{x}) = \left(\frac{72}{\lambda}\right) - n$$

In Exercise 1.18, we found that the Hessian was:

$$\ell''(\lambda; \tilde{x}) = -\frac{72}{\lambda^2}$$

So we can approximate the score function using the first-order Taylor polynomial[11]:

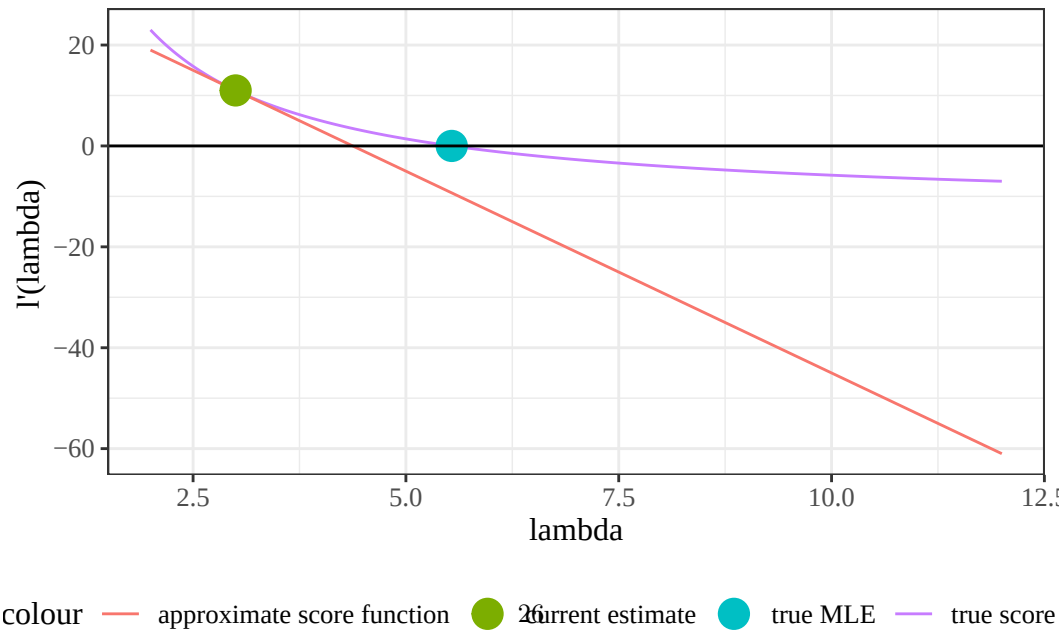
$$\begin{aligned} \ell'(\lambda) &\approx \ell'^*(\lambda) \\ &\stackrel{\text{def}}{=} \ell'(\hat{\lambda}^*) + \ell''(\hat{\lambda}^*)(\lambda - \hat{\lambda}^*) \\ &= \left(\frac{72}{\hat{\lambda}^*} - n\right) + \left(-\frac{72}{\left(\hat{\lambda}^*\right)^2}\right)(\lambda - \hat{\lambda}^*) \end{aligned}$$

---

[11] https://en.wikipedia.org/wiki/Taylor%27s_theorem

Figure 9 compares the true score function and the approximate score function at $\widehat{\lambda}^* = 3$.

```r
approx_score <- function(lambda, lhat, ...) {
  score(lambda = lhat, ...) +
    hessian(lambda = lhat, ...) * (lambda - lhat)
}

point_size <- 5

plot1 <- ggplot() +
  geom_function(
    fun = score,
    aes(col = "true score function"),
    n = 1001
  ) +
  geom_function(
    fun = approx_score,
    aes(col = "approximate score function"),
    n = 1001,
    args = list(lhat = cur_lambda_est)
  ) +
  geom_point(
    size = point_size,
    aes(
      x = cur_lambda_est, y = score(lambda = cur_lambda_est),
      col = "current estimate"
    )
  ) +
  geom_point(
    size = point_size,
    aes(
      x = xbar,
      y = 0,
      col = "true MLE"
    )
  ) +
  xlim(min(cyclones$number), max(cyclones$number)) +
  ylab("l'(lambda)") +
  xlab("lambda") +
  geom_hline(yintercept = 0)

print(plot1)
```



This is equivalent to estimating the log-likelihood with a second-order Taylor polynomial:

$$\ell^*(\lambda) = \ell(\hat{\lambda}^*) + (\lambda - \hat{\lambda}^*)\ell'(\hat{\lambda}^*) + \frac{1}{2}\ell''(\hat{\lambda}^*)(\lambda - \hat{\lambda}^*)^2$$

```r
approx_loglik <- function(lambda, lhat, ...) {
  loglik(lambda = lhat, ...) +
    score(lambda = lhat, ...) * (lambda - lhat) +
    1 / 2 * hessian(lambda = lhat, ...) * (lambda - lhat)^2
}

plot_loglik <- ggplot() +
  geom_function(
    fun = loglik,
    aes(col = "true log-likelihood"),
    n = 1001
  ) +
  geom_function(
    fun = approx_loglik,
    aes(col = "approximate log-likelihood"),
    n = 1001,
    args = list(lhat = cur_lambda_est)
  ) +
  geom_point(
    size = point_size,
    aes(
      x = cur_lambda_est, y = loglik(lambda = cur_lambda_est),
      col = "current estimate"
    )
  ) +
  geom_point(
    size = point_size,
    aes(
      x = xbar,
      y = loglik(xbar),
      col = "true MLE"
    )
  ) +
  xlim(min(cyclones$number) - 1, max(cyclones$number)) +
  ylab("l'(lambda)") +
  xlab("lambda")

print(plot_loglik)
```

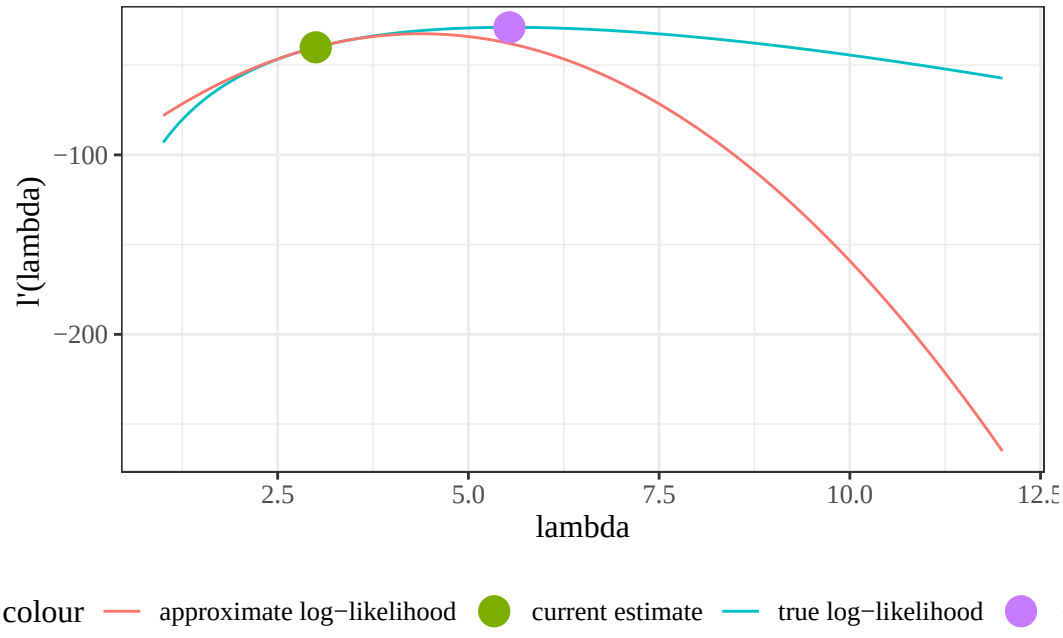


Figure 10: log-likelihood of Dobson cyclone data and approximate log-likelihood function

---

The approximate score function, $\ell'^*(\lambda)$, is a linear function of λ, so it is easy to solve the corresponding approximate score equation, $\ell'^*(\lambda) = 0$, for λ:

$$\begin{aligned}\lambda &= \hat{\lambda}^* - \ell'(\hat{\lambda}^*) \cdot \left(\ell''(\hat{\lambda}^*)\right)^{-1} \\ &= 4.375\end{aligned}$$

```
new_lambda_est <-
  cur_lambda_est -
  score(cur_lambda_est) * hessian(cur_lambda_est)^-1
```

```r
plot2 <- plot1 +
  geom_point(
    size = point_size,
    aes(
      x = new_lambda_est,
      y = 0,
      col = "new estimate"
    )
  ) +
  geom_segment(
    arrow = grid::arrow(),
    linewidth = 2,
    alpha = .7,
    aes(
      x = cur_lambda_est,
      y = approx_score(
        lhat = cur_lambda_est,
        lambda = cur_lambda_est
      ),
      xend = new_lambda_est,
      yend = 0,
      col = "update"
    )
  )
print(plot2)
```

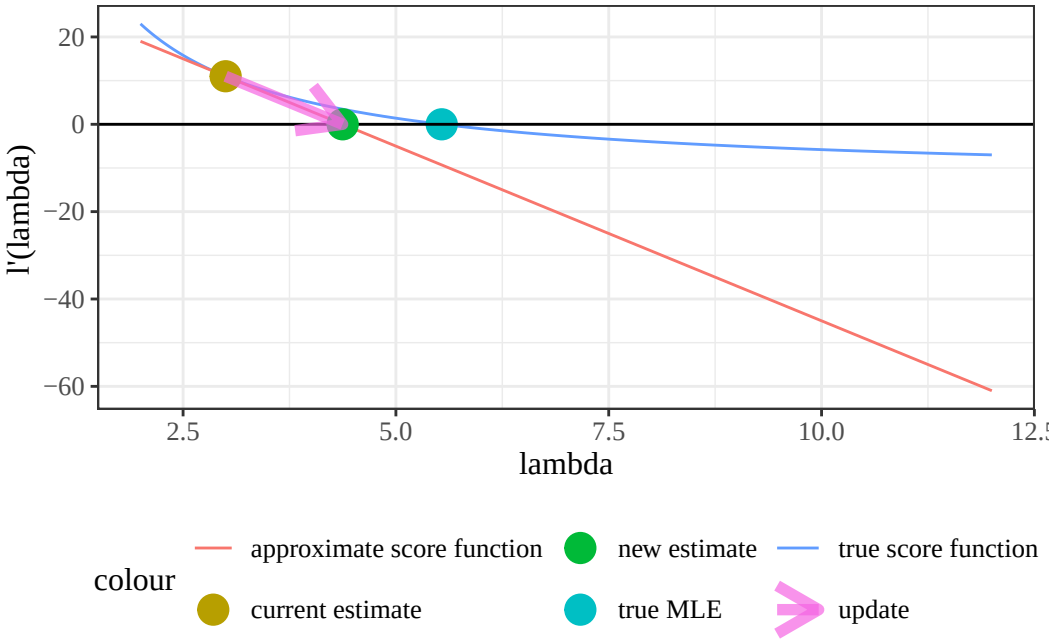


Figure 11: score function of Dobson cyclone data and approximate score function

---

So we update $\hat{\lambda}^* \leftarrow 4.375$ and repeat our estimation process:

```
plot2 +
  geom_function(
    fun = approx_score,
    aes(col = "new approximate score function"),
    n = 1001,
    args = list(lhat = new_lambda_est)
  ) +
  geom_point(
    size = point_size,
    aes(
      x = new_lambda_est, y = score(lambda = new_lambda_est),
      col = "new estimate"
    )
  )
```

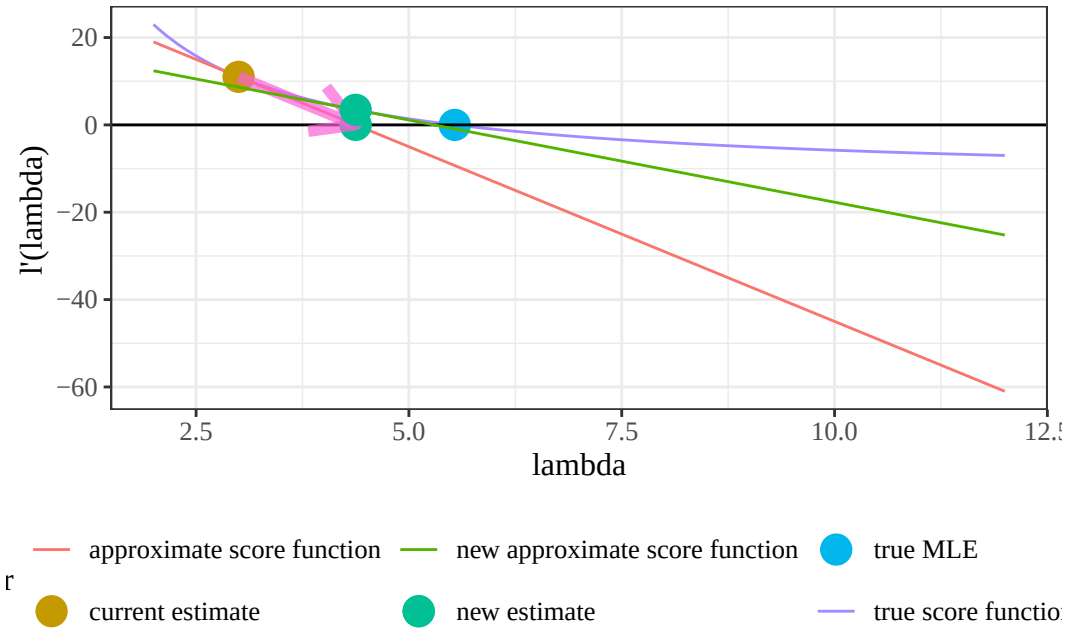


Figure 12: score function of Dobson cyclone data and approximate score function

---

We repeat this process until the likelihood converges:

Compare with Exercise 1.23

Table 3: Convergence of Newton-Raphson Algorithm for finding MLE of `cyclone` data

```
library(tibble)
cur_lambda_est <- 3 # restarting
diff_loglik <- Inf
tolerance <- 10^-4
max_iter <- 100
NR_info <- tibble( # nolint: object_name_linter
  iteration = 0,
  lambda = cur_lambda_est |> num(digits = 4),
  likelihood = lik(cur_lambda_est),
  `log(likelihood)` = loglik(cur_lambda_est) |> num(digits = 4),
  score = score(cur_lambda_est),
  hessian = hessian(cur_lambda_est)
)

for (cur_iter in 1:max_iter) {
  new_lambda_est <-
    cur_lambda_est - score(cur_lambda_est) * hessian(cur_lambda_est)^-1

  diff_loglik <- loglik(new_lambda_est) - loglik(cur_lambda_est)

  new_NR_info <- tibble( # nolint: object_name_linter
    iteration = cur_iter,
    lambda = new_lambda_est,
    likelihood = lik(new_lambda_est),
    `log(likelihood)` = loglik(new_lambda_est),
    score = score(new_lambda_est),
    hessian = hessian(new_lambda_est),
    `diff(loglik)` = diff_loglik
  )

  NR_info <- NR_info |> bind_rows(new_NR_info) # nolint: object_name_linter

  cur_lambda_est <- new_lambda_est

  if (abs(diff_loglik) < tolerance) {
    break
  }
}

NR_info
#> # A tibble: 6 x 7
#>   iteration  lambda likelihood `log(likelihood)`    score hessian `diff(loglik)`
#>       <dbl> <num:.>      <dbl>         <num:.4!>    <dbl>   <dbl>          <dbl>
#> 1         0  3.0000   4.00e-18          -40.0610 1.1 e+ 1   -8         NA
#> 2         1  4.3750   4.33e-14          -30.7708 3.46e+ 0   -3.76       9.29e+ 0
#> 3         2  5.2941   2.57e-13          -28.9897 6.00e- 1   -2.57       1.78e+ 0
#> 4         3  5.5277   2.76e-13          -28.9176 2.54e- 2   -2.36       7.21e- 2
#> 5         4  5.5384   2.76e-13          -28.9175 4.93e- 5   -2.35       1.37e- 4
#> 6         5  5.5385   2.76e-13          -28.9175 1.87e-10   -2.35       5.18e-10
```

```r
ll_plot +
  geom_segment(
    data = NR_info,
    arrow = grid::arrow(),
    alpha = .7,
    aes(
      x = lambda,
      xend = lead(lambda),
      y = `log(likelihood)`,
      yend = lead(`log(likelihood)`),
      col = factor(iteration)
    )
  )
```

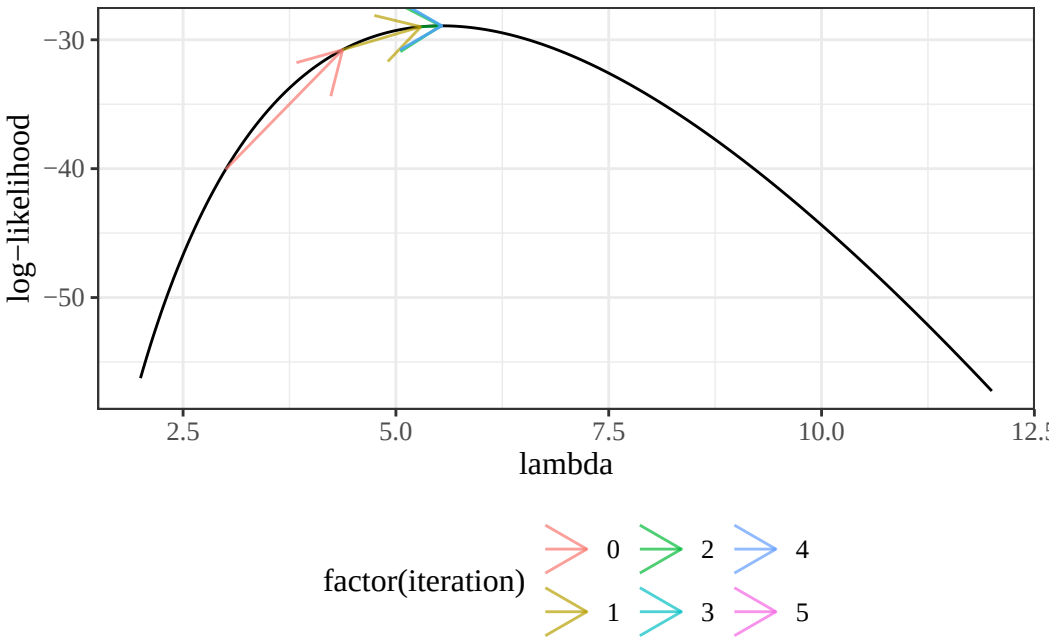


Figure 13: Newton-Raphson algorithm for finding MLE of model 13 for `cyclone` data

1.4 Maximum likelihood inference for univariate Gaussian models

Suppose $X_1, ..., X_n \sim_{\text{iid}} N(\mu, \sigma^2)$. Let $X = (X_1, ..., X_n)^\top$ be these random variables in vector format. Let x_i and x denote the corresponding observed data. Then $\theta = (\mu, \sigma^2)$ is the vector of true parameters, and $\Theta = (\mathrm{M}, \Sigma^2)$ is the vector of parameters as a random vector.

$$\mathcal{L} = \prod_{i=1}^{n} (2\sigma^2\pi)^{-1/2} \exp\left\{ -\frac{1}{2} \frac{(x_i - \mu)^2}{\sigma^2} \right\}$$

Then the log-likelihood is:

$$\begin{aligned} \ell &\propto -\frac{n}{2} \log \sigma^2 - \frac{1}{2} \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{\sigma^2} \\ &= -\frac{n}{2} \log \sigma^2 - \frac{1}{2\sigma^2} \sum_{i=1}^{n} x_i^2 - 2x_i\mu + \mu^2 \end{aligned}$$

1.4.1 The score function

$$\ell'(x, \theta) \overset{\text{def}}{=} \frac{\partial}{\partial \theta} \ell(x, \theta) = \begin{pmatrix} \frac{\partial}{\partial \mu} \ell(\theta; x) \\ \frac{\partial}{\partial \sigma^2} \ell(\theta; x) \end{pmatrix} = \begin{pmatrix} \ell'_\mu(\theta; x) \\ \ell'_{\sigma^2}(\theta; x) \end{pmatrix}$$

.

$\ell'(x, \theta)$ is the function we set equal to 0 and solve to find the MLE:

$$\hat{\theta}_{ML} = \{\theta : \ell'(x, \theta) = 0\}$$

1.4.2 MLE of μ

$$\begin{aligned}\frac{d\ell}{d\mu} &= -\frac{1}{2}\sum_{i=1}^{n}\frac{-2(x_i - \mu)}{\sigma^2} \\ &= \frac{1}{\sigma^2}\left[\left(\sum_{i=1}^{n} x_i\right) - n\mu\right]\end{aligned}$$

If $\frac{d\ell}{d\mu} = 0$, then $\mu = \overline{x} \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} x_i$.

$$\frac{d^2\ell}{(d\mu)^2} = \frac{-n}{\sigma^2} < 0$$

So $\hat{\mu}_{ML} = \overline{x}$.

1.4.3 MLE of σ^2

> **Reparametrizing the Gaussian distribution**
>
> When solving for $\hat{\sigma}_{ML}$, you can treat σ^2 as an atomic variable (don't differentiate with respect to σ or things get messy). In fact, you can replace σ^2 with $1/\tau$ and differentiate with respect to τ instead, and the process might be even easier.

$$\begin{aligned}\frac{d\ell}{d\sigma^2} &= \frac{\partial}{\partial\sigma^2}\left(-\frac{n}{2}\log\sigma^2 - \frac{1}{2}\sum_{i=1}^{n}\frac{(x_i - \mu)^2}{\sigma^2}\right) \\ &= -\frac{n}{2}\left(\sigma^2\right)^{-1} + \frac{1}{2}\left(\sigma^2\right)^{-2}\sum_{i=1}^{n}(x_i - \mu)^2\end{aligned}$$

If $\frac{d\ell}{d\sigma^2} = 0$, then:

$$\frac{n}{2}\left(\sigma^2\right)^{-1} = \frac{1}{2}\left(\sigma^2\right)^{-2}\sum_{i=1}^{n}(x_i - \mu)^2$$

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

We plug in $\hat{\mu}_{ML} = \overline{x}$ to maximize globally (a technique called profiling):

$$\hat{\sigma}^2_{ML} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2$$

Now:

$$\begin{aligned}
\frac{d^2\ell}{(d\sigma^2)^2} &= \frac{\partial}{\partial\sigma^2}\left\{-\frac{n}{2}\left(\sigma^2\right)^{-1}+\frac{1}{2}\left(\sigma^2\right)^{-2}\sum_{i=1}^{n}\left(x_i-\mu\right)^2\right\} \\
&= \left\{-\frac{n}{2}\frac{\partial}{\partial\sigma^2}\left(\sigma^2\right)^{-1}+\frac{1}{2}\frac{\partial}{\partial\sigma^2}\left(\sigma^2\right)^{-2}\sum_{i=1}^{n}\left(x_i-\mu\right)^2\right\} \\
&= \left\{\frac{n}{2}\left(\sigma^2\right)^{-2}-\left(\sigma^2\right)^{-3}\sum_{i=1}^{n}\left(x_i-\mu\right)^2\right\} \\
&= \left(\sigma^2\right)^{-2}\left\{\frac{n}{2}-\left(\sigma^2\right)^{-1}\sum_{i=1}^{n}\left(x_i-\mu\right)^2\right\}
\end{aligned}$$

Evaluated at $\mu=\overline{x},\sigma^2=\frac{1}{n}\sum_{i=1}^{n}\left(x_i-\overline{x}\right)^2$, we have:

$$\begin{aligned}
\frac{d^2\ell}{(d\sigma^2)^2} &= \left(\hat{\sigma}^2\right)^{-2}\left\{\frac{n}{2}-\left(\hat{\sigma}^2\right)^{-1}\sum_{i=1}^{n}\left(x_i-\overline{x}\right)^2\right\} \\
&= \left(\hat{\sigma}^2\right)^{-2}\left\{\frac{n}{2}-\left(\hat{\sigma}^2\right)^{-1}n\hat{\sigma}^2\right\} \\
&= \left(\hat{\sigma}^2\right)^{-2}\left\{\frac{n}{2}-n\right\} \\
&= \left(\hat{\sigma}^2\right)^{-2}n\left\{\frac{1}{2}-1\right\} \\
&= \left(\hat{\sigma}^2\right)^{-2}n\left(-\frac{1}{2}\right)<0
\end{aligned}$$

Finally, we have:

$$\begin{aligned}
\frac{d^2\ell}{d\mu\ d\sigma^2} &= \frac{\partial}{\partial\mu}\left\{-\frac{n}{2}\left(\sigma^2\right)^{-1}+\frac{1}{2}\left(\sigma^2\right)^{-2}\sum_{i=1}^{n}\left(x_i-\mu\right)^2\right\} \\
&= \frac{1}{2}\left(\sigma^2\right)^{-2}\frac{\partial}{\partial\mu}\sum_{i=1}^{n}\left(x_i-\mu\right)^2 \\
&= \frac{1}{2}\left(\sigma^2\right)^{-2}\sum_{i=1}^{n}-2(x_i-\mu) \\
&= -\left(\sigma^2\right)^{-2}\sum_{i=1}^{n}\left(x_i-\mu\right)
\end{aligned}$$

Evaluated at $\mu=\hat{\mu}=\overline{x},\sigma^2=\hat{\sigma}^2=\frac{1}{n}\sum_{i=1}^{n}\left(x_i-\overline{x}\right)^2$, we have:

$$\frac{d^2\ell}{d\mu\ d\sigma^2}=-\left(\hat{\sigma}^2\right)^{-2}\left(n\overline{x}-n\overline{x}\right)=0$$

1.4.4 Covariance matrix

$$I=\begin{bmatrix}\frac{n}{\widehat{\sigma}^2} & 0\\ 0 & \left(\hat{\sigma}^2\right)^{-2}n\left(\frac{1}{2}\right)\end{bmatrix}=\begin{bmatrix}a & 0\\ 0 & d\end{bmatrix}$$

So:

$$I^{-1}=\frac{1}{ad}\begin{bmatrix}d & 0\\ 0 & a\end{bmatrix}=\begin{bmatrix}\frac{1}{a} & 0\\ 0 & \frac{1}{d}\end{bmatrix}$$

$$I^{-1}=\begin{bmatrix}\frac{\widehat{\sigma}^2}{n} & 0\\ 0 & \frac{2\left(\widehat{\sigma}^2\right)^2}{n}\end{bmatrix}$$

Table 4: HERS dataset

```
hers |> head()
#> # A tibble: 6 x 37
#>   HT              age raceth  nonwhite smoking drinkany exercise physact globrat
#>   <dbl+lbl>     <dbl> <dbl+l> <dbl+lb> <dbl+l> <dbl+lb> <dbl+lb> <dbl+l> <dbl+l>
#> 1 0 [placebo]      70 2 [Afr~ 1 [yes]  0 [no]  0 [no]   0 [no]   5 [muc~ 3 [goo~
#> 2 0 [placebo]      62 2 [Afr~ 1 [yes]  0 [no]  0 [no]   0 [no]   1 [muc~ 3 [goo~
#> 3 1 [hormone t~    69 1 [Whi~ 0 [no]   0 [no]  0 [no]   0 [no]   3 [abo~ 3 [goo~
#> 4 0 [placebo]      64 1 [Whi~ 0 [no]   1 [yes] 1 [yes]  0 [no]   1 [muc~ 3 [goo~
#> 5 0 [placebo]      65 1 [Whi~ 0 [no]   0 [no]  0 [no]   0 [no]   2 [som~ 3 [goo~
#> 6 1 [hormone t~    68 2 [Afr~ 1 [yes]  0 [no]  1 [yes]  0 [no]   3 [abo~ 3 [goo~
#> # i 28 more variables: poorfair <dbl+lbl>, medcond <dbl>, htnmeds <dbl+lbl>,
#> #   statins <dbl+lbl>, diabetes <dbl+lbl>, dmpills <dbl+lbl>,
#> #   insulin <dbl+lbl>, weight <dbl>, BMI <dbl>, waist <dbl>, WHR <dbl>,
#> #   glucose <dbl>, weight1 <dbl>, BMI1 <dbl>, waist1 <dbl>, WHR1 <dbl>,
#> #   glucose1 <dbl>, tchol <dbl>, LDL <dbl>, HDL <dbl>, TG <dbl>, tchol1 <dbl>,
#> #   LDL1 <dbl>, HDL1 <dbl>, TG1 <dbl>, SBP <dbl>, DBP <dbl>, age10 <dbl>
```

See Casella and Berger (2002) p322, example 7.2.12.

To prove it's a maximum, we need:

- $\ell' = 0$
- At least one diagonal element of ℓ'' is negative.
- Determinant of ℓ'' is positive.

1.5 Example: hormone therapy study

Now, we're going to analyze some real-world data using a Gaussian model, and then we're going to do a simulation to examine the properties of maximum likelihood estimation for that Gaussian model.

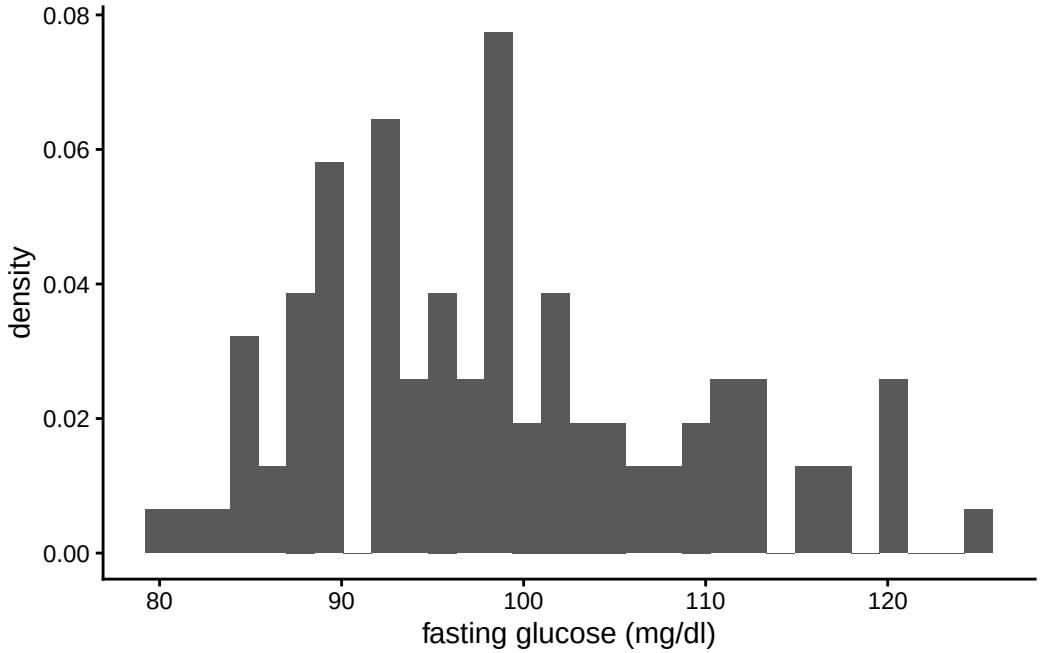
The "heart and estrogen/progestin study" (HERS) was a clinical trial of hormone therapy for prevention of recurrent heart attacks and death among 2,763 post-menopausal women with existing coronary heart disease (CHD) (Hulley et al. 1998).

We are going to model the distribution of fasting glucose among non-diabetics who don't exercise.

```
# load the data directly from a UCSF website
hers <- haven::read_dta(
  paste0( # I'm breaking up the url into two chunks for readability
    "https://regression.ucsf.edu/sites/g/files",
    "/tkssra6706/f/wysiwyg/home/data/hersdata.dta"
  )
)
```

```
n_obs <- 100 # we're going to take a small subset of the data to look at;
# if we took the whole data set, the likelihood function would be hard to
# graph nicely

library(dplyr)
data1 <-
  hers |>
  filter(
    diabetes == 0,
    exercise == 0
```

```
  ) |>
  head(n_obs)

glucose_data <-
  data1 |>
  pull(glucose)

library(ggplot2)
plot1 <-
  data1 |>
  ggplot(aes(x = glucose)) +
  geom_histogram(aes(x = glucose, after_stat(density))) +
  theme_classic()

print(plot1)
```



Looks somewhat plausibly Gaussian. Good enough for this example!

1.5.1 Find the MLEs

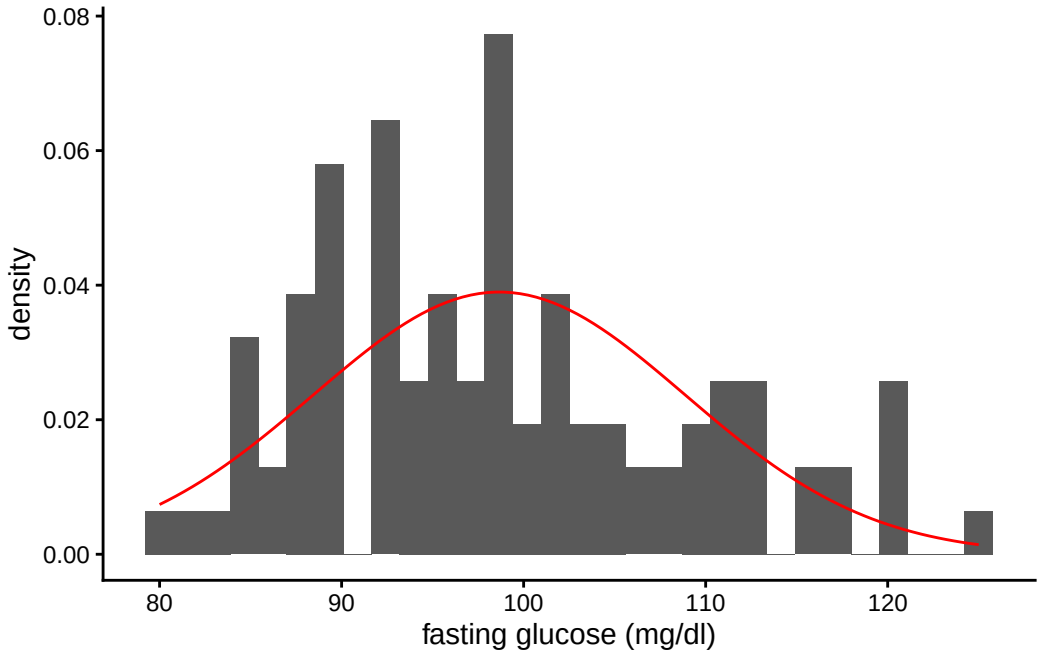
```
mu_hat <- mean(glucose_data)
sigma_sq_hat <- mean((glucose_data - mean(glucose_data))^2)
```

Our MLEs are:

- $\hat{\mu} = 98.66$
- $\hat{\sigma}^2 = 104.7444$

Here's the estimated distribution, superimposed on our histogram:

```
plot1 +
  geom_function(
    fun = function(x) dnorm(x, mean = mu_hat, sd = sqrt(sigma_sq_hat)),
    col = "red"
  )
```



Looks like a somewhat decent fit? We could probably do better, but that's for another time.

1.5.2 Construct the likelihood and log-likelihood functions

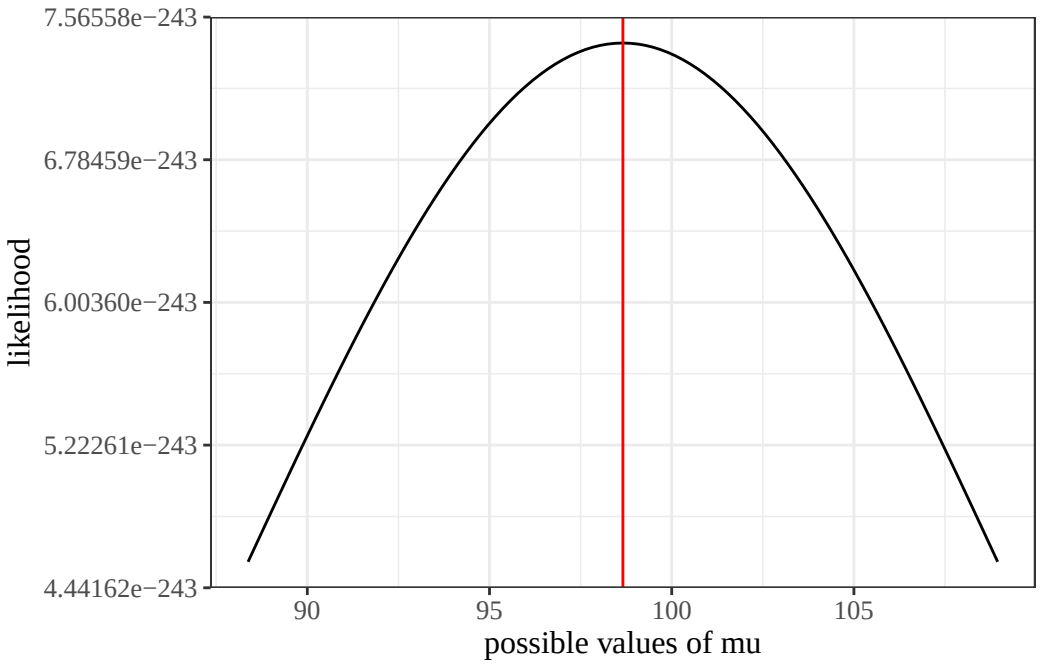
it's often computationally more effective to construct the log-likelihood first and then exponentiate it to get the likelihood

```
loglik <- function(
    mu, # I'm assigning default values, which the function will use
    # unless we tell it otherwise
    sigma = sd(x), # note that you can define some default inputs
    # based on other arguments
    x = glucose_data,
    n = length(x)) {
  normalizing_constants <- -n / 2 * log((sigma^2) * 2 * pi)

  likelihood_kernel <- -1 / (2 * sigma^2) * {
    # I have to do this part in a somewhat complicated way
    # so that we can pass in vectors of possible values of mu
    # and get the likelihood for each value;
    # for the binomial case it's easier
    sum(x^2) - 2 * sum(x) * mu + n * mu^2
  }

  answer <- normalizing_constants + likelihood_kernel

  return(answer)
}

# `...` means pass any inputs to lik() along to loglik()
lik <- function(...) exp(loglik(...))
```

1.5.3 Graph the Likelihood as a function of μ

(fixing σ^2 at $\hat{\sigma}^2 = 104.7444$)

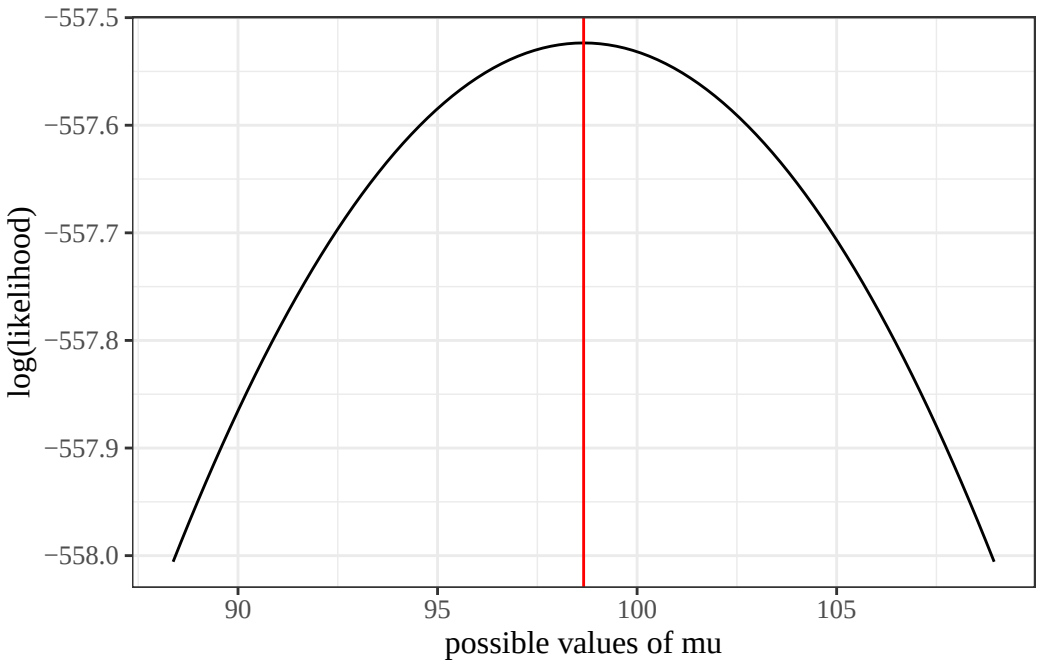
```
ggplot() +
  geom_function(fun = function(x) lik(mu = x, sigma = sigma_sq_hat)) +
  xlim(mean(glucose_data) + c(-1, 1) * sd(glucose_data)) +
  xlab("possible values of mu") +
  ylab("likelihood") +
  geom_vline(xintercept = mean(glucose_data), col = "red")
```



1.5.4 Graph the Log-likelihood as a function of μ

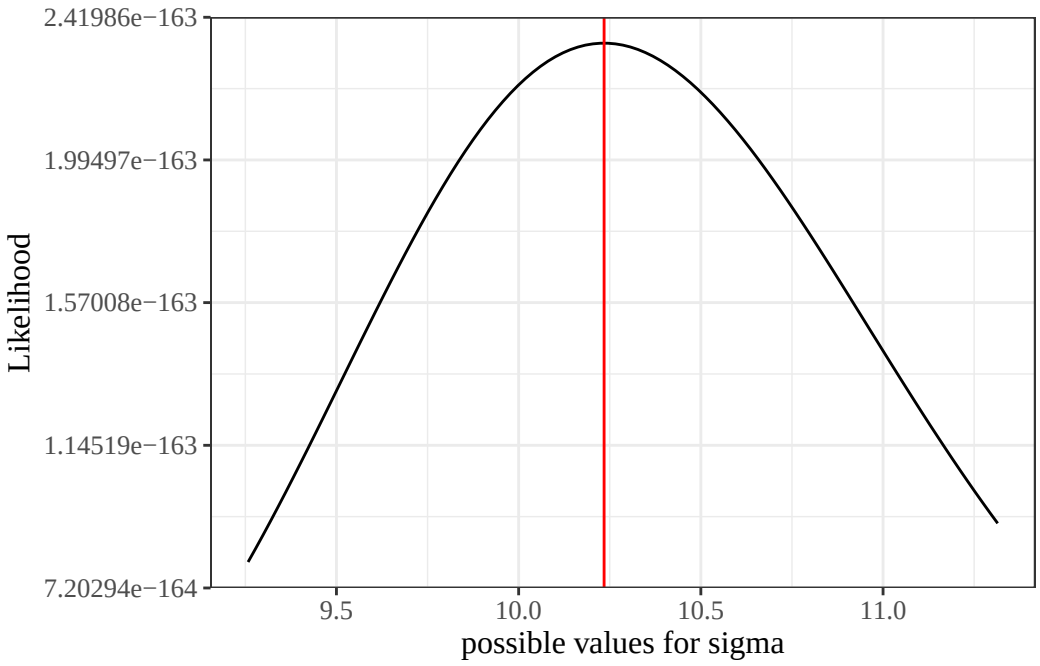
(fixing σ^2 at $\hat{\sigma}^2 = 104.7444$)

```
ggplot() +
  geom_function(fun = function(x) loglik(mu = x, sigma = sigma_sq_hat)) +
  xlim(mean(glucose_data) + c(-1, 1) * sd(glucose_data)) +
  xlab("possible values of mu") +
  ylab("log(likelihood)") +
  geom_vline(xintercept = mean(glucose_data), col = "red")
```



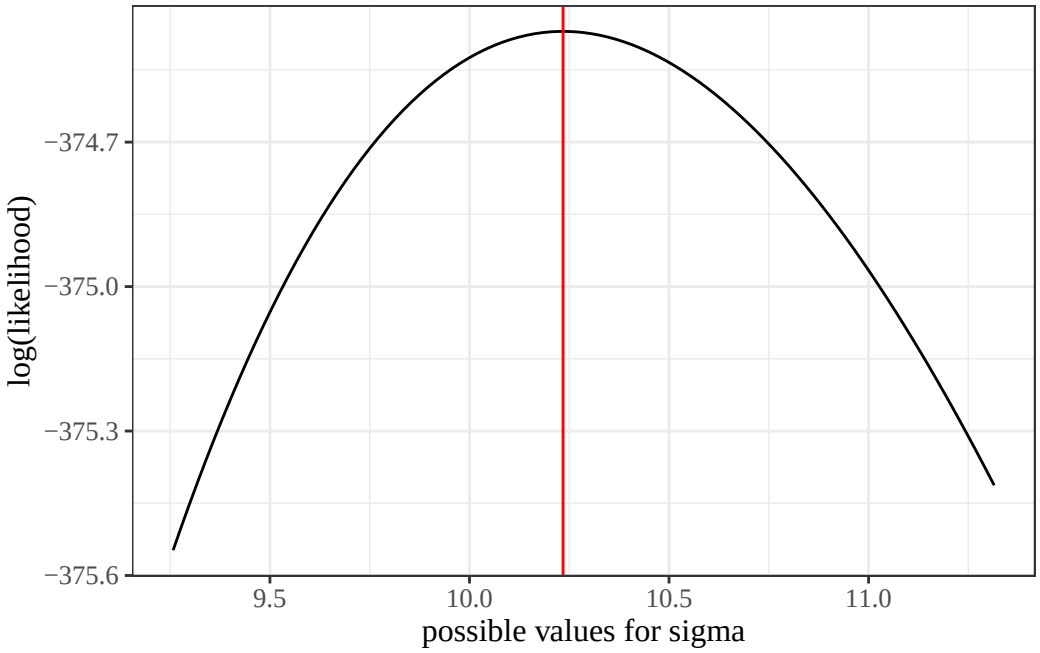
1.5.5 Likelihood and log-likelihood for σ, conditional on $\mu = \hat{\mu}$:

```
ggplot() +
  geom_function(fun = function(x) lik(sigma = x, mu = mean(glucose_data))) +
  xlim(sd(glucose_data) * c(.9, 1.1)) +
  geom_vline(
    xintercept = sd(glucose_data) * sqrt(n_obs - 1) / sqrt(n_obs),
    col = "red"
  ) +
  xlab("possible values for sigma") +
  ylab("Likelihood")
```




```
ggplot() +
```

```
  geom_function(
    fun = function(x) loglik(sigma = x, mu = mean(glucose_data))
  ) +
  xlim(sd(glucose_data) * c(0.9, 1.1)) +
  geom_vline(
    xintercept =
      sd(glucose_data) * sqrt(n_obs - 1) / sqrt(n_obs),
    col = "red"
  ) +
  xlab("possible values for sigma") +
  ylab("log(likelihood)")
```

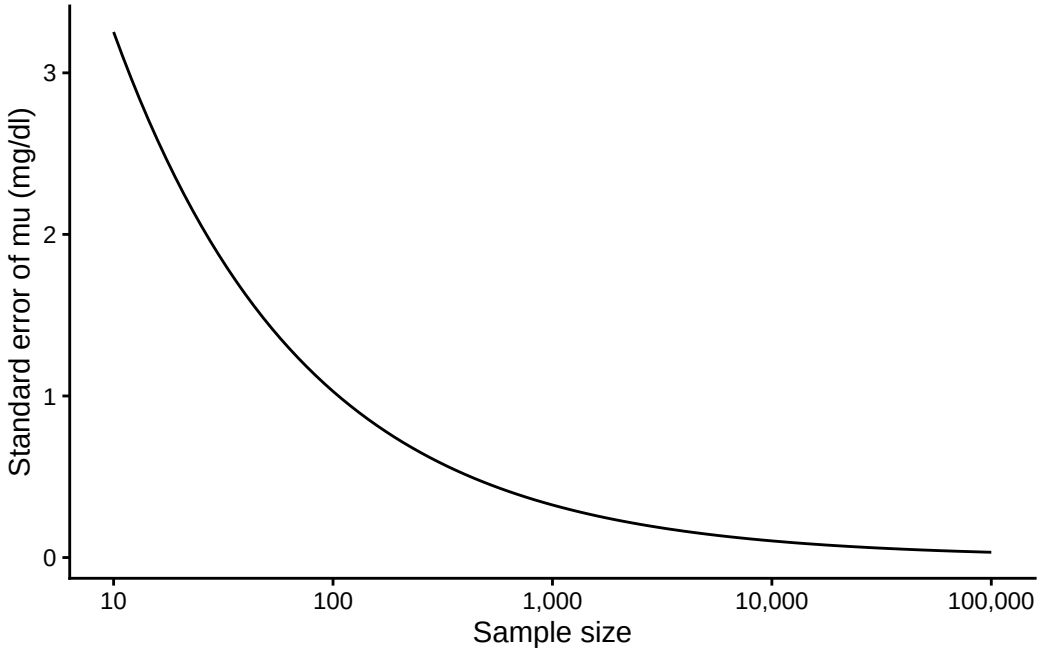


1.5.6 Standard errors by sample size:

Recall from Section 1.4.4 that the asymptotic standard error of $\hat{\mu}_{ML}$ is

$$\begin{aligned}\widehat{\mathrm{SE}}(\hat{\mu}) &= \sqrt{\left[\left(\hat{\mathcal{J}}(\hat{\mu}_{ML})\right)^{-1}\right]} \\ &= \frac{\hat{\sigma}}{\sqrt{n}}\end{aligned}$$

```
se_mu_hat <- function(n, sigma = sd(glucose_data)) sigma / sqrt(n)
ggplot() +
  geom_function(fun = se_mu_hat) +
  scale_x_log10(
    limits = c(10, 10^5), name = "Sample size",
    labels = scales::label_comma()
  ) +
  ylab("Standard error of mu (mg/dl)") +
  theme_classic()
```



1.5.7 Power

Rejection region

For example, suppose we wish to detect a difference from the hypothesized value $\mu_0 = 95$. We reject the null hypothesis for any mean value outside the "non-rejection interval"

$$\mu_0 \pm F^{-1}_{t(n-1)}(1-\alpha/2)\sqrt{\frac{\sigma^2}{n}}$$

```
mu_0 <- 95
n <- length(glucose_data)
se <- se_mu_hat(n = n)
margin <- qt(0.975, df = n - 1) * se
upperbound <- mu_0 + margin
lowerbound <- mu_0 - margin
```

In this case, the non-rejection interval is $[92.959028, 97.040972]$.

Calculate power under a simple alternative

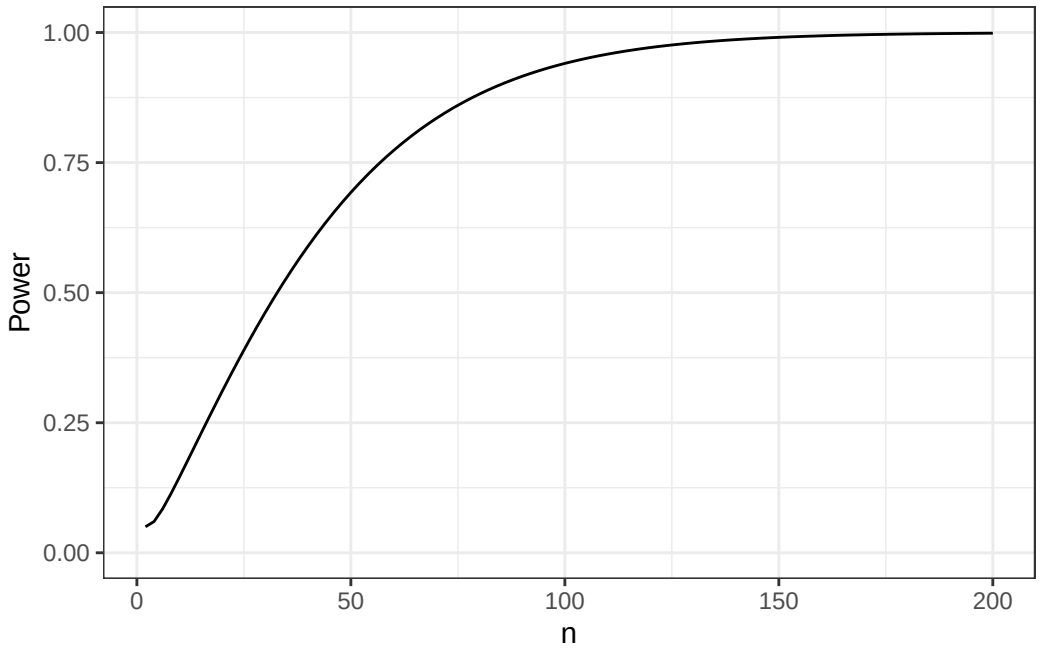
Consider the simple alternative that the true value is actually the estimated mean calculated from the data (i.e. 98.66). Let's also assume that the known standard deviation is what we estimated from the data.

```
prob_low <- pt(
  q = (lowerbound - mu_hat) / se,
  df = n - 1,
  lower.tail = TRUE
)

prob_high <- pt(
  q = (upperbound - mu_hat) / se,
  df = n - 1,
  lower.tail = FALSE
)

power <- prob_low + prob_high
```

```
print(power)
#> [1] 0.940662
```

Power as a function of sample size

```
power <- function(n = 100, null = 95, alt = 98.66) {
  # there's no such thing as fractional sample size:
  n <- floor(n)
  # using the function we wrote earlier:
  se <- se_mu_hat(n = n)
  reject_upper <- ((null + qt(0.975, df = n - 1) * se) - alt) / se
  reject_lower <- ((null - qt(0.975, df = n - 1) * se) - alt) / se
  p_reject_high <-
    pt(
      q = reject_lower,
      df = n - 1
    )
  p_reject_low <-
    pt(
      q = reject_upper,
      df = n - 1,
      lower = FALSE
    )
  p_reject <- p_reject_high + p_reject_low
  return(p_reject)
}
power_plot <-
  ggplot() +
  geom_function(fun = power, n = 100) +
  xlim(c(2, 200)) + # n = 1 is not allowed for t-distribution
  ylim(0, 1) +
  ylab("Power") +
  xlab("n") +
  theme_bw()
print(power_plot)
```



1.5.8 Simulations

Create simulation framework

Here's a function that performs a single simulation of a Gaussian modeling analysis:

```
do_one_sim <- function(
  n = 100,
  mu = mean(glucose_data),
  mu_0 = mean(glucose_data) * 0.9,
  sigma2 = var(glucose_data),
  return_data = FALSE # if this is set to true, we will create a list()
  # containing both the analytic results and the vector of simulated data
) {
  # generate data
  x <- rnorm(n = 100, mean = mu, sd = sqrt(sigma2))

  # analyze data
  mu_hat <- mean(x)
  sigmahat <- sd(x)
  se_hat <- sigmahat / sqrt(n)
  confint <- mu_hat + c(-1, 1) * se_hat * qt(.975, df = n - 1)
  tstat <- abs(mu_hat - mu_0) / se_hat
  pval <- pt(df = n - 1, q = tstat, lower = FALSE) * 2
  confint_covers <- between(mu, confint[1], confint[2])
  test_rejects <- pval < 0.05

  # if you want spaces, hyphens, or characters in your column names,
  # use "", '', or ``:
  to_return <- tibble(
    "mu-hat" = mu_hat,
    "sigma-hat" = sigmahat,
    "se_hat" = se_hat,
    "confint_left" = confint[1],
    "confint_right" = confint[2],
    "tstat" = tstat,
    "pval" = pval,
    "confint covers true mu" = confint_covers,
    "test rejects null hypothesis" = test_rejects
  )

  if (return_data) {
    return(
      list(
        data = x,
        results = to_return
      )
    )
  } else {
    return(to_return)
  }
}
```

Let's see what this function outputs for us:

```
do_one_sim()
#> # A tibble: 1 x 9
#>   `mu-hat` `sigma-hat` se_hat confint_left confint_right tstat     pval
#>      <dbl>       <dbl>  <dbl>        <dbl>         <dbl> <dbl>    <dbl>
#> 1     98.8        11.7   1.17         96.5          101.  8.58 1.32e-13
```

```
#> # i 2 more variables: `confint covers true mu` <lgl>,
#> #   `test rejects null hypothesis` <lgl>
```

Looks good!

Now let's check it against the `t.test()` function from the `stats` package:

```
set.seed(1)
mu <- mean(glucose_data)
mu_0 <- 80
sim_output <- do_one_sim(mu_0 = mu_0, return_data = TRUE)
our_results <-
  sim_output$results |>
  mutate(source = "`do_one_sim()`")

results_t_test <- t.test(sim_output$data, mu = mu_0)

results2 <-
  tibble(
    source = "`stats::t.test()`",
    "mu-hat" = results_t_test$estimate,
    "sigma-hat" = results_t_test$stderr * sqrt(length(sim_output$data)),
    "se_hat" = results_t_test$stderr,
    confint_left = results_t_test$conf.int[1],
    confint_right = results_t_test$conf.int[2],
    tstat = results_t_test$statistic,
    pval = results_t_test$p.value,
    "confint covers true mu" = between(mu, confint_left, confint_right),
    `test rejects null hypothesis` = pval < 0.05
  )

comparison <-
  bind_rows(
    our_results,
    results2
  ) |>
  relocate(
    "source",
    .before = everything()
  )

comparison
#> # A tibble: 2 x 10
#>   source   `mu-hat` `sigma-hat` se_hat confint_left confint_right tstat     pval
#>   <chr>       <dbl>       <dbl>  <dbl>        <dbl>         <dbl> <dbl>    <dbl>
#> 1 `do_one~     99.8        9.24  0.924         97.9          102.  21.4 6.23e-39
#> 2 `stats:~     99.8        9.24  0.924         97.9          102.  21.4 6.23e-39
#> # i 2 more variables: `confint covers true mu` <lgl>,
#> #   `test rejects null hypothesis` <lgl>
```

Looks like we got it right!

Run 1000 simulations

Here's a function that calls the previous function `n_sims` times and summarizes the results:

```
do_n_sims <- function(
  n_sims = 1000,
  ... # this symbol means "allow additional arguments to be passed on to the
```

```
  # `do_sim_once` function
) {
  sim_results <- NULL # we're going to create a "tibble" of results,
  # row by row (slightly different from the hint on the homework)

  for (i in 1:n_sims) {
    set.seed(i) # sets a different seed for each simulation iteration,
    # to get a different dataset each time

    current_results <-
      do_one_sim(...) |> # here's where the simulation actually gets run
      mutate(
        sim_number = i
      ) |>
      relocate("sim_number", .before = everything())

    sim_results <-
      sim_results |>
      bind_rows(current_results)
  }

  return(sim_results)
}
```

```
sim_results <- do_n_sims(
  n_sims = 1000,
  mu = mean(glucose_data),
  sigma2 = var(glucose_data),
  n = 100 # this is the number of samples per simulated data set
)

sim_results
#> # A tibble: 1,000 x 10
#>    sim_number `mu-hat` `sigma-hat` se_hat confint_left confint_right tstat
#>         <int>    <dbl>       <dbl>  <dbl>        <dbl>         <dbl> <dbl>
#>  1          1     99.8        9.24  0.924         97.9          102.  11.9
#>  2          2     98.3       11.9   1.19          96.0          101.   8.00
#>  3          3     98.8        8.81  0.881         97.0          101.  11.3
#>  4          4     99.7        9.40  0.940         97.8          102.  11.6
#>  5          5     99.0        9.72  0.972         97.1          101.  10.5
#>  6          6     98.6       10.6   1.06          96.4          101.   9.18
#>  7          7    100.         9.86  0.986         98.1          102.  11.5
#>  8          8     97.7       11.1   1.11          95.5           99.9  8.03
#>  9          9     98.1        9.86  0.986         96.2          100.   9.45
#> 10         10     97.3        9.68  0.968         95.3           99.2  8.74
#> # i 990 more rows
#> # i 3 more variables: pval <dbl>, `confint covers true mu` <lgl>,
#> #   `test rejects null hypothesis` <lgl>
```

The simulation results are in! Now we have to analyze them.

Analyze simulation results

To do that, we write another function:

```
summarize_sim <- function(
    sim_results,
    mu = mean(glucose_data),
    sigma2 = var(glucose_data),
```

```
    n = 100) {
  # calculate the true standard error based on the data-generating parameters:
  se_mu_hat <- sqrt(sigma2 / n)

  sim_results |>
    summarize(
      `bias[mu-hat]` = mean(.data$`mu-hat`) - mu,
      `SE(mu-hat)` = sd(.data$`mu-hat`),
      `bias[SE-hat]` = mean(.data$se_hat) - se_mu_hat,
      `SE(SE-hat)` = sd(.data$se_hat),
      coverage = mean(.data$`confint covers true mu`),
      power = mean(.data$`test rejects null hypothesis`)
    )
}
```

Let's try it out:

```
sim_summary <- summarize_sim(
  sim_results,
  mu = mean(glucose_data),
  # this function needs to know the true parameter values in order to assess
  # bias
  sigma2 = var(glucose_data),
  n = 100
)

sim_summary
#> # A tibble: 1 x 6
#>   `bias[mu-hat]` `SE(mu-hat)` `bias[SE-hat]` `SE(SE-hat)` coverage power
#>            <dbl>        <dbl>          <dbl>        <dbl>    <dbl> <dbl>
#> 1       -0.00501         1.00       -0.00113       0.0736    0.959     1
```

From this simulation, we observe that our estimate of μ, $\hat{\mu}$, has minimal bias, and so does our estimate of $SE(\hat{\mu})$, $\widehat{SE}(\hat{\mu})$.

The confidence intervals captured the true value even more often than they were supposed to, and the hypothesis test always rejected the null hypothesis.

I wonder what would happen with a different sample size, a different true μ value, or a different σ^2 value...

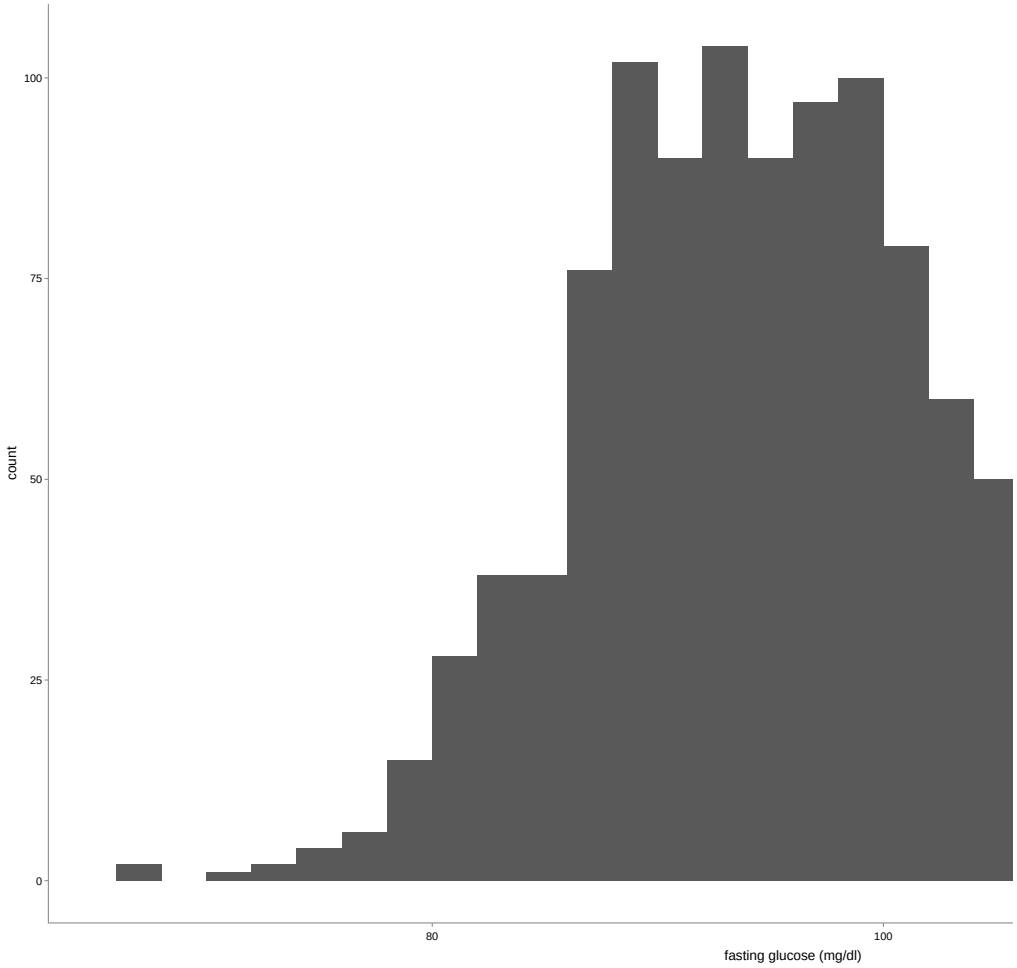
1.6 likelihood graphs

```
library(pander)
library(ggplot2)
library(plotly)
library(dplyr)
library(haven)
```

```
# load the data from package
hers = haven::read_dta(fs::path_package("rme", "extdata/hersdata.dta"))
# "https://regression.ucsf.edu/sites/g/files/tkssra16191/files/wysiwyg/home/data/hersdata.dta"
```

```
data1 =
  hers |>
  filter(
    diabetes == 0,
    exercise == 0)
```

```r
n.obs <- nrow(data1)


glucose_data =
  data1 |>
  pull(glucose)

plot1 =
  data1 |>
  ggplot() +
  geom_histogram(aes(x = glucose), bins = 30) +
  theme_classic()

plot1 |> ggplotly()
```



Looks somewhat plausibly Gaussian. Good enough for this example!

1.7 Construct the likelihood and log-likelihood functions

```
# it's computationally better to construct the log-likelihood first and then
# exponentiate it to get the likelihood

loglik = function(
    mu = mean(x), # I'm assigning default values, which the function will use
    # unless we tell it otherwise
```

```
    sigma = sd(x), # note that you can define some defaults based on other arguments
    x = glucose_data,
    n = length(x)
)
{

  normalizing_constants = -n/2 * log((sigma^2) * 2 * pi)

  likelihood_kernel = - 1/(2 * sigma^2) *
    {
      # I have to do this part in a somewhat complicated way
      # so that we can pass in vectors of possible values of mu
      # and get the likelihood for each value;
      # for the binomial case it's easier
      sum(x^2) - 2 * sum(x) * mu + n * mu^2
    }

  answer = normalizing_constants + likelihood_kernel

  return(answer)

}

# `...` means pass any inputs to lik() along to loglik()
lik = function(...) exp(loglik(...))
```

1.7.1 Graph the Likelihood

```
mu_likplot <-
  ggplot() +
  geom_function(fun = function(x) lik(mu = x)) +
  xlim(mean(glucose_data) + c(-1,1) * sd(glucose_data)) +
  ylab("likelihood") +
  xlab("mu") +
  geom_vline(xintercept = mean(glucose_data), col = "red")
```

Figure 14: Likelihood of `hers` data w.r.t. μ

1.7.2 Graph the Log-likelihood

```
ggplot() +
  geom_function(fun = function(x) loglik(mu = x)) +
  xlim(mean(glucose_data) + c(-1,1) * sd(glucose_data)) +
  ylab('log(likelihood)') +
  xlab("mu") +
  geom_vline(xintercept = mean(glucose_data), col = "red")
```

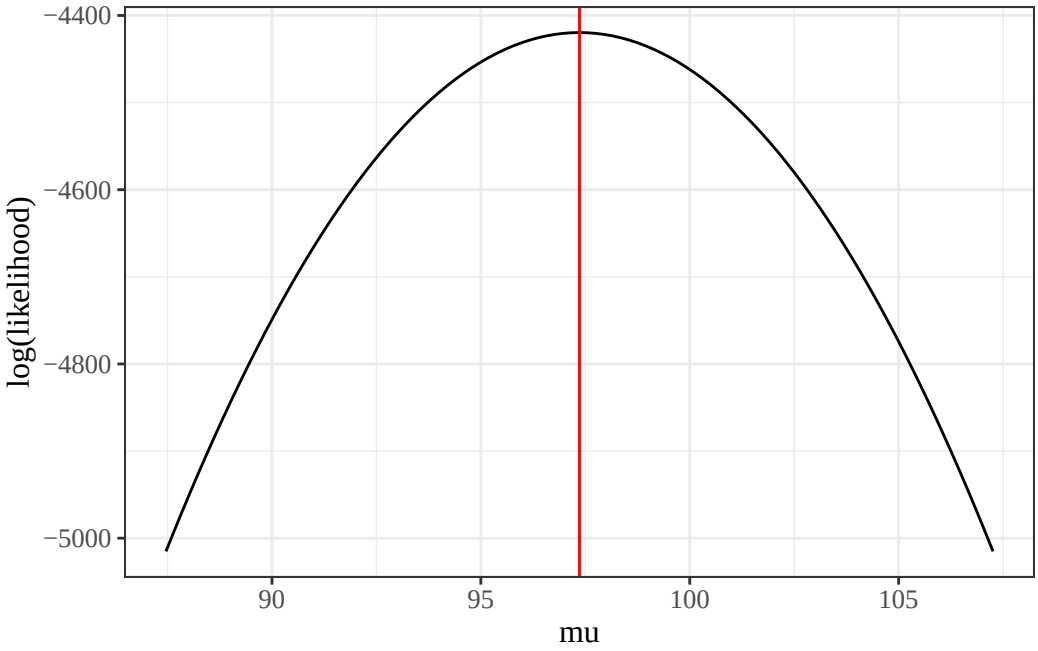


Figure 15: Log-likelihood of `hers` data w.r.t. μ

1.8 Likelihood and log-likelihood for σ^2, conditional on $\mu = \hat{\mu}$:

```
lik_plot = ggplot() +
  geom_function(fun = function(x) lik(sigma = x, mu = mean(glucose_data))) +
  xlim(sd(glucose_data) * c(.9,1.1)) +
  geom_vline(
    xintercept = sd(glucose_data) * sqrt(n.obs - 1)/sqrt(n.obs),
    col = "red") +
  ylab('Likelihood')
```

```
loglik_plot = ggplot() +
  geom_function(
    fun = function(x) loglik(sigma = x, mu = mean(glucose_data))
  ) +
  xlim(sd(glucose_data) * c(0.9, 1.1)) +
  geom_vline(
    xintercept =
      sd(glucose_data) * sqrt(n.obs - 1) / sqrt(n.obs),
    col = "red") +
  ylab("log(likelihood)")
```

```r
## Graph the log-likelihood ranging over both parameters at once:

library(plotly)

n_points = 25
mu = seq(90, 105, length.out = n_points)
sigma = seq(6, 20,
            length.out = n_points)
names(mu) = round(mu, 5)
names(sigma) = round(sigma, 5)
lliks = outer(mu, sigma, loglik)
liks = outer(mu, sigma, lik)

plotly::plot_ly(
  type = "surface",
  x = ~mu,
  y = ~sigma,
  z = ~t(lliks))
```

Casella, George, and Roger Berger. 2002. *Statistical Inference*. 2nd ed. Cengage Learning. https://www.cengage.com/c/statistical-inference-2e-casella-berger/9780534243128/.

Dobson, Annette J, and Adrian G Barnett. 2018. *An Introduction to Generalized Linear Models*. 4th ed. CRC press. https://doi.org/10.1201/9781315182780.

Efron, Bradley, and David V Hinkley. 1978. "Assessing the Accuracy of the Maximum Likelihood Estimator: Observed Versus Expected Fisher Information." *Biometrika* 65 (3): 457–83.

Hogg, Robert V., Elliot A. Tanis, and Dale L. Zimmerman. 2015. *Probability and Statistical Inference*. Ninth edition. Boston: Pearson.

Hulley, Stephen, Deborah Grady, Trudy Bush, Curt Furberg, David Herrington, Betty Riggs, Eric Vittinghoff, for the Heart, and Estrogen/progestin Replacement Study (HERS) Research Group. 1998. "Randomized Trial of Estrogen Plus Progestin for Secondary Prevention of Coronary Heart Disease in Postmenopausal Women." *JAMA : The Journal of the American Medical Association* 280 (7): 605–13.

Lehmann, E. L. 1999. *Elements of Large-Sample Theory*. Springer Texts in Statistics. New York: Springer. https://doi.org/10.1007/b98855.

McLachlan, Geoffrey J, and Thriyambakam Krishnan. 2007. *The EM Algorithm and Extensions*. 2nd ed. John Wiley & Sons. https://doi.org/10.1002/9780470191613.

Newey, Whitney K, and Daniel McFadden. 1994. "Large Sample Estimation and Hypothesis Testing." In *Handbook of Econometrics*, edited by Robert Engle and Dan McFadden, 4:2111–2245. Elsevier. https://doi.org/https://doi.org/10.1016/S1573-4412(05)80005-4.