

Bayesian Inference 2019

Ville Hyvönen & Topias Tolonen

2019-03-13

Contents

1	Introduction	5
1.1	Motivating example : thumbtack tossing	5
1.2	Components of Bayesian inference	11
1.3	Prediction	13
2	Conjugate distributions	19
2.1	One-parameter conjugate models	19
2.2	Prior distributions	28
3	Summarizing the posterior distribution	33
3.1	Credible intervals	33
3.2	Posterior mean as a convex combination of means	43
4	Approximate inference	45
4.1	Simulation methods	45
4.2	Monte Carlo integration	51
4.3	Monte Carlo markov chain (MCMC) methods	54
4.4	Probabilistic programming	58
4.5	Sampling from posterior predictive distribution	65
5	Multiparameter models	69
5.1	Marginal posterior distribution	69
5.2	Inference for the normal distribution with known variance	70
5.3	Inference for the normal distribution with noninformative prior	72
6	Hierarchical models	81
6.1	Two-level hierarchical model	81
6.2	Conditional conjugacy	84
6.3	Hierarchical model example	84
7	Linear model	99
7.1	Classical linear model	99

Chapter 1

Introduction

1.1 Motivating example : thumbtack tossing

A classical toy example of the random experiment in probability calculus is coin tossing. But this is a little bit boring example, since we know (at least if the coin is fair) *a priori* that the probability of both heads and tails is very close to 0.5.

Instead, let's consider a slightly more interesting toy example: thumbtack tossing. If we define the success as a thumbtack landing with its point up, we can only have a vague guess about the success probability before conducting the experiment.

Let's toss a thumbtack n times, and count the number of times it lands with its point up; denote this quantity as y . We are interested in deducing the true success probability θ .

Probably our first intuition is just to use the proportion of successes y/n as an estimate of the true success probability θ . But consider an outcome where you tossed the thumbtack $n = 3$ times, and each time the thumbtack landed point down; this means that your observed value is $y = 0$. Would it be sensible to conclude that the true success probability in this is $\theta = y/n = 0/3 = 0$? It clearly makes no sense to conclude that the *true* underlying success probability θ is equal to the *observed* proportion y/n .

Also if we toss the thumbtack $n = 3000$ times and observe the zero successes, the proportion of successes is also $y/n = 0$, but now it would make much more sense conclude that thumbtack landing point up is actually impossible, or at least a very rare event.

So in addition to the *most probable* value of θ we also need to measure the **uncertainty** of our estimates. Finding the most likely parameter values, and quantifying our uncertainty about them is called **statistical inference**.

1.1.1 Modelling thumbtack tossing

To generate some real world data I threw a thumbtack $N = 30$ times. It landed point up 16 times, and point down 14 times; this means we observed a data set $y = 16$.

Let's define a proper statistical model to quantify our uncertainty of the true probability of the thumbtack landing point up. We can consider an observed proportion of the successes y as a realization of random variable Y . As we remember from the probability calculus course, a repeated random experiment with constant success probability, binary outcome and independent repetitions is modelled with *binomial distribution*:

$$Y \sim \text{Bin}(n, \theta), \quad 0 < \theta < 1.$$

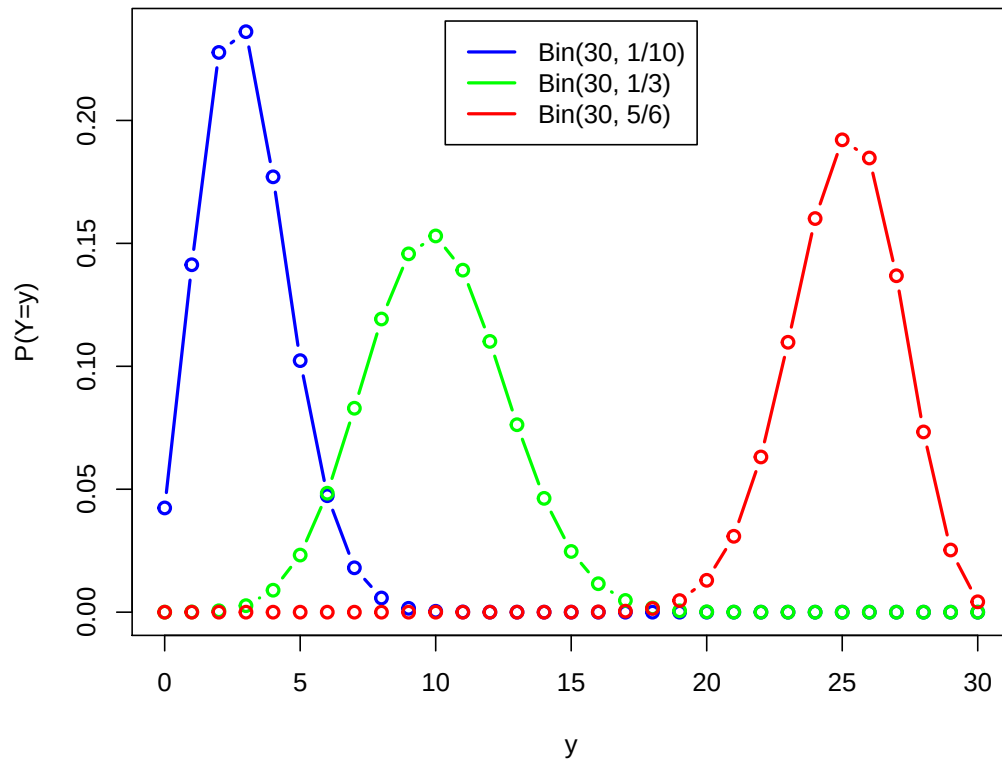
This means that random variable Y follows a binomial distribution with a (fixed) sample size n and a success probability θ . Unknown quantities in the model, such as θ here, are called **parameters** of the model.

The functional form of the probability mass function (pmf) of Y :

$$f(y; n, \theta) = \binom{n}{y} \theta^y (1 - \theta)^{n-y}$$

is fixed, and the value the parameter θ determines what it looks like. Let's draw some pmf:s of Y with a fixed sample size $N = 30$, and different parameter values:

```
par(mar = c(4, 4, .1, .1))
n <- 30
y <- 0:30
theta <- c(3, 10, 25) / n
plot(y, dbinom(y, size = n, prob = theta[1]), lwd = 2, col = 'blue', type = 'b',
     ylab = 'P(Y=y)')
lines(y, dbinom(y, size = n, prob = theta[2]), lwd = 2, col = 'green', type = 'b')
lines(y, dbinom(y, size = n, prob = theta[3]), lwd = 2, col = 'red', type = 'b')
legend('top', inset = .02, legend = c('Bin(30, 1/10)', 'Bin(30, 1/3)', 'Bin(30, 5/6)'),
     col = c('blue', 'green', 'red'), lwd = 2)
```



1.1.2 Frequentist thumbtack tossing

In classical (sometimes called frequentist) statistics we consider the likelihood function $L(\theta; y)$; this is just a pmf/pdf of the observations considered as a function of parameter θ :

$$\theta \mapsto f(y; \theta).$$

Then we can find the most likely value of the parameter by maximizing the likelihood function (normally we actually maximize the natural logarithm of the likelihood function often called the **log-likelihood**, $l(\theta; y) = \log L(\theta; y)$, which is computationally more convenient) w.r.t. parameter θ . This means that we find the parameter value, which has a highest probability of producing this particular data set. This parameter value

$\hat{\theta}$, which maximizes the likelihood function is called a **maximum likelihood estimate**:

$$\hat{\theta}(y) = \underset{\theta}{\operatorname{argmax}} L(\theta; y).$$

The maximum likelihood estimate is *the most likely value of the parameter given the data*.

Let's derive the maximum likelihood estimate for our binomial model. Because logarithm is a monotonously increasing function, the global maximum point of the log-likelihood maximizes also the likelihood function. Log-likelihood for this model is:

$$l(\theta; y) = \log f(y; \theta) \propto \log(\theta^y (1 - \theta)^{n-y}) = y \log \theta + (n - y) \log(1 - \theta)$$

We dropped the normalizing constant $\binom{n}{y}$ from the likelihood function because it is a constant w.r.t. parameter θ , and thus has no effect on the maximum point. Next we will find the critical points of the log-likelihood by derivating it w.r.t. θ , and solving the points where the derivative is zero:

$$\begin{aligned} l'(\theta; y) &= \frac{y}{\theta} - \frac{n-y}{1-\theta} = 0 \\ \theta &= \frac{y}{n}. \end{aligned}$$

We can see that this indeed is a maximum point by examining the value of the derivative on the both sides of this point (it changes from positive to negative), or if we are too lazy to think, by just computing the second derivative of the log-likelihood:

$$l''(\theta; y) = -\frac{y}{\theta^2} - \frac{n-y}{(1-\theta)^2}.$$

Because $0 \leq y \leq n$, this is always negative; thus, log-likelihood is a concave function and so its only critical point must be its global maximum point. This means that the maximum likelihood estimate of our model is

$$\hat{\theta}(y) = \frac{y}{n} = \frac{16}{30},$$

which also matches our intuitive solution. But the most likely value is not enough for us: we also want to know on the other hand how confident we are in our estimate, and on the other hand how likely are other parameter values (besides of the maximum likelihood estimate). We could for example ask what is the probability that the true value of the parameter lies between 0.4 and 0.6? Or what is the probability that the true value of the parameter is higher than 0.5? Or how much more probable it is that the true value of the parameter is higher than 0.5 than it is smaller than 0.5?

Somewhat surprisingly, it turns out that in the framework of classical statistics we cannot directly answer these questions: they are not considered well-defined! This is because in classical statistics the parameter θ is considered as a fixed, but unknown constant. There is nothing random about the parameter; hence we cannot make any probability statements about it.

In classical statistics the way to get around this restriction is to examine the values of the maximum likelihood estimate over all possible data sets that could have been observed. For instance, we can examine a maximum likelihood estimate as the function of the random variable Y instead of the observed data y . The resulting random variable is called a **maximum likelihood estimator (MLE)**:

$$\hat{\theta}(Y) = \frac{Y}{n}.$$

We can for example estimate the standard deviation of the maximum likelihood estimator (called standard error). It is also possible to construct *confidence intervals* for the parameter values: for example 95% confidence interval is an interval $(a(Y), b(Y))$, which has at least 95% probability of containing the true parameter value. Notice that here the randomness is over the observations, not the parameter value.

In the frequentist framework we can also test a so called *null hypothesis* concerning the parameter value, such as $H_0 : \theta = 0.5$ against an alternative hypothesis $H_1 : \theta \neq 0.5$. Again, we do not make any probability

statements about the parameter value, but we *assume that true value of the parameter is 0.5*, and examine how probable it would be to observe our current data set y with that parameter value.

If all this sounds quite complicated, don't worry: this is not what we are going to do in this course. Instead, the topic of this course is **Bayesian statistical inference**. Bayesian framework is conceptually simpler than the classical framework, because we actually *can* make probability statements about the parameter values. In Bayesian inference we consider the parameter to be a random variable instead of the fixed constant. Let's make this explicit by denoting the parameter by capital letter Θ instead of θ .

1.1.3 Fully Bayesian model

After this short digression into the frequentist statistics let's move back to our thumbtack tossing example. What is our probability estimate for the thumbtack landing point up before we have made any throws? Unlike in coin tossing or the dice throwing, we do not have a clear prior opinion about the possibility of the outcomes. So let's make an assumption that all values are equally likely for the probability Θ (the probability of thumbtack landing point up). Because Θ is a probability it resides in the interval $[0, 1]$. Thus, we can quantify our uncertainty about the true parameter value before conducting the experiment by saying that it has a uniform distribution over the interval $[0, 1]$:

$$\Theta \sim U(0, 1).$$

This is called the **prior distribution**, and it is a second of the two components required to fully define a Bayesian statistical model.

The first component of the Bayesian model, which we have already defined, is the distribution of the data given the parameter; this is usually called a **sampling distribution** or a **likelihood**. Because in Bayesian inference the parameter is thought as a random variable, let's change the notation for the sampling distribution a little bit:

$$f_{Y|\Theta}(y|\theta).$$

From this notation it is clear that the sampling distribution is a *conditional* probability distribution.

To recap, our full Bayesian model for the thumbtack tossing is:

$$\begin{aligned} Y|\Theta &\sim \text{Bin}(n, \Theta) \\ \Theta &\sim U(0, 1), \end{aligned}$$

and we observed a data set $y = 16$.

The next step of the Bayesian inference is to update our beliefs about the probability of the parameter values *after* observing the data. This is quantified by computing the **posterior** distribution of the parameter Θ . This is simply a conditional distribution of Θ given the data $Y = y$.

Thus, our task is to find out a conditional distribution $f_{\Theta|Y}(\theta|y)$ given the model and the observed data. From the probability calculus we remember the **chain rule**:

$$f_{X,Y} = f_X f_{Y|X},$$

which we can use to factorize the joint distribution of the parameter and the data:

$$f_{\Theta,Y}(\theta, y) = f_Y(y) f_{\Theta|Y}(\theta|y).$$

Using this factorization we can write the posterior distribution as a quotient of the joint distribution and the marginal distribution of the data:

$$f_{\Theta|Y}(\theta|y) = \frac{f_{\Theta,Y}(\theta, y)}{f_Y(y)}$$

We can utilize the chain rule again to write the joint distribution as the product of the prior distribution and the likelihood; hence we can write the posterior distribution as:

$$f_{\Theta|Y}(\theta|y) = \frac{f_{\Theta}(\theta) f_{Y|\Theta}(y|\theta)}{f_Y(y)}$$

We have just deduced **Bayes's theorem**, which is the cornerstone of Bayesian inference! Our model defines the numerator, so the only unknown component left is the denominator, which is the marginal distribution of the data (usually called a **marginal likelihood**). But luckily we can observe that the posterior distribution is a function of the parameter θ , and there is no θ in the denominator. This means that the denominator is a constant w.r.t. θ ; because we know that the posterior distribution is a probability distribution we can solve it up to the constant term, and deduce the normalizing constant later. Let's write a posterior distribution as proportional (The proportionality notation $f(x) \propto h(x)$ means simply that there exists a constant $c \in \mathbb{R}$, s.t. $f(x) = ch(x)$) to the joint distribution:

$$f_{\Theta|Y}(\theta|y) \propto f_{\Theta}(\theta)f_{Y|\Theta}(y|\theta) = 1 \cdot \binom{n}{y} \theta^y (1 - \theta)^{n-y}.$$

By dropping again drop all the constant terms from this expression, we can simply write:

$$f_{\Theta|Y}(\theta|y) \propto \theta^y (1 - \theta)^{n-y}.$$

Is there any probability distribution whose density has this kind of functional form over the interval $(0, 1)$? Luckily (or later we find out that this was not such a coincidence after all) it turns out that there indeed is: a beta distribution. Random variable X , which follows a beta distribution with parameters α and β , has a probability density function

$$f(x) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1},$$

over interval $(0, 1)$. The integral

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha + \beta)} = \int_0^1 x^{\alpha-1} (1-x)^{\beta-1} dx \quad (1.1)$$

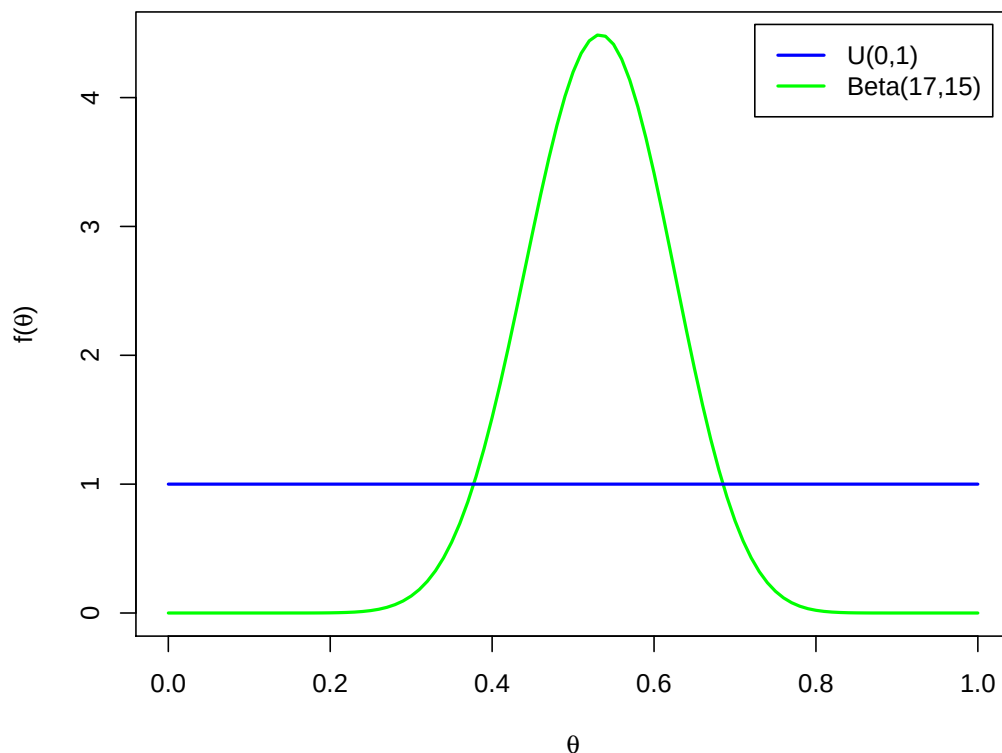
is called a *beta function* or *Euler's beta function*.

We can recognize that the unnormalized posterior distribution is a probability density function of the beta distribution with parameters $y + 1$ and $n - y + 1$ up to a normalizing constant. Hence, our posterior distribution must be a beta distribution

$$\Theta|Y \sim \text{Beta}(y + 1, n - y + 1).$$

Instead of the point estimate we actually have now a whole probability distribution for all the possible parameter values! Let's see what it looks like:

```
par(mar = c(4, 4, .1, .1))
y <- 16
n <- 30
theta <- seq(0,1, by = .01) # create tight grid for plotting
alpha <- y + 1
beta <- n - y + 1
plot(theta, dbeta(theta, alpha, beta), lwd = 2, col = 'green',
     type = 'l', xlab = expression(theta), ylab = expression(paste('f(', theta, ')')))
lines(theta, dunif(theta), lwd = 2, col = 'blue', type = 'l')
legend('topright', inset = .02,
     legend = c('U(0,1)', paste0('Beta(', alpha, ', ', beta, ')')),
     col = c('blue', 'green'), lwd = 2)
```



While the density of the prior distribution is flat, the density of posterior distribution is clearly concentrated near the value $\theta = 0.5$. Now that have the full posterior distribution, we can easily compute the probabilities we were interested in:

```
1 - pbeta(0.5, alpha, beta) # P(theta > 0.5)
```

```
## [1] 0.6399499
```

```
pbeta(0.6, alpha, beta) - pbeta(0.4, alpha, beta) # P(0.4 < theta < 0.6)
```

```
## [1] 0.7128906
```

From the picture we can observe that almost all of the probability mass of the posterior distribution is between 0.2 and 0.8. Indeed, it is very likely that the true probability of the thumbtack landing point up really resides on this interval:

```
pbeta(0.8, alpha, beta) - pbeta(0.2, alpha, beta) # P(0.2 < theta < 0.8)
```

```
## [1] 0.9996158
```

We can also summarize the posterior distributions with a point estimate. In Bayesian statistics **posterior mean**, which is the mean of the posterior distribution is a widely used point estimate because of its optimality in the sense of mean squared error. A posterior mean in our thumbtacking example is a mean of the beta distribution:

$$E(\Theta|Y = y) = \frac{\alpha}{\alpha + \beta} = \frac{y + 1}{(n - y + 1) + (y + 1)} = \frac{y + 1}{n + 2} = \frac{17}{32}.$$

This is very close to the maximum likelihood estimate of this model, but both the numbers of failures and successes are inflated by one “pseudo-observation”. We will examine this phenomenon more closely in the next week when we discuss the choice of prior distributions.

1.2 Components of Bayesian inference

Let's briefly recap and define more rigorously the main concepts of the Bayesian belief updating process, which we just demonstrated.

Consider a slightly more general situation than our thumbtack tossing example: we have observed a data set $\mathbf{y} = (y_1, \dots, y_n)$ of n observations, and we want to examine the mechanism which has generated these observations. To this end, we model the observed data set as an observed value of the random vector $\mathbf{Y} = (Y_1, \dots, Y_n)$.

In this course we limit ourselves to the **parametric inference**. Parametric inference is a special case of the statistical inference where it is assumed that the functional form of the joint distribution of the random vector $\mathbf{Y}$ is fixed up to the value of the parameter vector $\theta = (\theta_1, \dots, \theta_d) \in \Omega$ living in some **parameter space** Ω . The distribution of the data is written as the conditional distribution of the data given the parameter (because, as we remember, in Bayesian inference the parameter is considered as a random variable): $f_{\mathbf{Y}|\Theta}(\mathbf{y}|\theta)$. This means that inference about the distribution of the data is reduced to finding out the distribution of the unknown parameter Θ . This simplifies the inference process significantly, because we can limit ourselves to the vector spaces instead of the function spaces.

Sampling distribution / likelihood function

Conditional distribution of the data set given the parameter, $f_{\mathbf{Y}|\Theta}(\mathbf{y}|\theta)$, is called a *sampling distribution*, or the often simply a *likelihood function*.

More rigorously the sampling distribution means $f_{\mathbf{Y}|\Theta}(\mathbf{y}|\theta)$ as a function of the observed data:

$$y \mapsto f_{\mathbf{Y}|\Theta}(\mathbf{y}|\theta),$$

and likelihood function as a function of the parameter:

$$\theta \mapsto f_{\mathbf{Y}|\Theta}(\mathbf{y}|\theta),$$

but often these terms are used interchangeably in practice (and also on this course).

Because our data set is a vector, in the general case a structure of the sampling distribution can be quite complicated. However, if we assume that our observations are independent (given the value of the parameter Θ), denoted as

$$Y_1, \dots, Y_n \perp\!\!\!\perp \Theta,$$

the joint sampling distribution of random vector $\mathbf{Y}$ can be factorized into a product of the sampling distributions of its components:

$$f_{\mathbf{Y}|\Theta}(\mathbf{y}|\theta) = \prod_{i=1}^n f_{Y_i|\Theta}(y_i|\theta).$$

The situation is further simplified if our observations follow a same distribution. This situation is encountered quite often in this course, at least in the simplest examples. We say that random variables are **independent and identically distributed (i.i.d.)**. In this case each of n components of the random vector $\mathbf{Y}$ has a common sampling distribution $f(y|\theta)$, and the joint sampling distribution can be further simplified to

$$f_{\mathbf{Y}|\Theta}(\mathbf{y}|\theta) = \prod_{i=1}^n f(y_i|\theta).$$

In some cases, such as in our thumbtack tossing example the form of the sampling distribution (binomial distribution in this case) follows quite naturally from the structure of the experimental situation. Other distributions that often follow naturally from the symmetry arguments or physical aspects of the examined phenomenon are multinomial distribution (extension of binomial experiment into the experiments with more than two possible outcomes, such as throwing a dice), normal distribution (sums or means of the independent

random variables), Poisson distribution (occurrences of the independent events) and exponential distribution (waiting times or lifespans). In the more complex situations we cannot usually use any of these simple models directly, but we can try to build so called *hierarchical models* out of these basic distributions. Ultimately the choice of the sampling distribution is subjective, and up to our domain knowledge of the modelled phenomenon / and or computational convenience.

Prior distribution

A marginal distribution $f_{\Theta}(\theta)$ of the parameter is called a *prior distribution*. Priori is latin for *before*: the prior distribution describes our beliefs about the likely values of the parameter Θ before observing any data.

If we do not have any strong beliefs about the possible values of the parameter *or* we do not want let our beliefs to influence our results, we should choose as a vague priori distribution as possible, such as the uniform distribution in our thumbtack tossing example. This kind of the priori distribution is called an **uninformative prior**. But what we mean by “vague” here? It turns out that it is not possible to find a prior distribution that would be universally uninformative. For example uniform priors lead quickly to problems, if the parameter space is not restricted: how can you even define an uniform distribution over an interval of infinite length?

On the other hand, when we want to let our prior knowledge influence our posterior distribution, we set a stronger prior distribution. This kind of the prior distribution is called an **informative prior**. Informative prior distribution may be for example used to enforce sparsity into the model; this means we have a strong prior belief that some parameters of the model should be zero.

We will soon revisit uninformative and informative priors with a simple example.

The prior distribution for the parameter vector Θ is also a parametric distribution; its parameters $\phi = (\phi_1, \dots, \phi_k)$ are called **hyperparameters**. We can denote prior distribution also as $f_{\Theta|\Phi}(\theta|\phi)$, but often the notation is simplified by leaving out the hyperparameters.

Bayesian model

To specify the fully Bayesian probability model, besides of the sampling distribution, we also need to specify the prior distribution of the parameter.

Together they determine the **joint distribution** of the observed data and the parameter:

$$f_{\Theta, \mathbf{Y}}(\theta, \mathbf{y}) = f_{\Theta}(\theta) f_{\mathbf{Y}|\Theta}(\mathbf{y}|\theta).$$

This full joint distribution is rarely computed or handled explicitly. Instead, the Bayesian inference is based on computing conditional and marginal densities from it.

Posterior distribution

The conditional distribution of the parameter given the data is called a *posterior distribution*. Posteriori is latin for *after*: posterior distribution describes our beliefs about the probable values of the parameter after we have observed the data.

In principle, the posterior distribution is computed from the prior and the sampling distributions using the **Bayes’ theorem**:

$$f_{\Theta|\mathbf{Y}}(\theta|\mathbf{y}) = \frac{f_{\Theta, \mathbf{Y}}(\theta, \mathbf{y})}{f_{\mathbf{Y}}(\mathbf{y})} = \frac{f_{\Theta}(\theta) f_{\mathbf{Y}|\Theta}(\mathbf{y}|\theta)}{f_{\mathbf{Y}}(\mathbf{y})}.$$

In practice, we usually utilize the fact that the normalizing constant $f_{\mathbf{Y}}(\mathbf{y})$ contains no θ ; thus, it is a constant w.r.t. parameter θ . This means that we can compute the unnormalized density of the posterior distribution simply as a product of the sampling and prior distributions:

$$f_{\Theta|\mathbf{Y}}(\theta|\mathbf{y}) \propto f_{\Theta}(\theta) f_{\mathbf{Y}|\Theta}(\mathbf{y}|\theta),$$

and then deduce the missing normalizing constant. In the first examples of this course this is often done by recognizing the functional form of the familiar probability density.

Marginal likelihood

The normalizing constant $f_{\mathbf{Y}}(\mathbf{y})$ of the Bayes' theorem is called a *marginal likelihood* (sometimes also an *evidence*). It is computed by marginalizing out the parameter from the full joint probability distribution. For the continuous parameter this is done by integrating the joint probability distribution over the parameter space:

$$f_{\mathbf{Y}}(\mathbf{y}) = \int_{\Omega} f_{\Theta}(\theta) f_{\mathbf{Y}|\Theta}(\mathbf{y}|\theta) d\theta,$$

and for the discrete parameter by summing the joint probability distribution over the parameter space:

$$f_{\mathbf{Y}}(\mathbf{y}) = \sum_{\theta \in \Omega} f_{\Theta}(\theta) f_{\mathbf{Y}|\Theta}(\mathbf{y}|\theta).$$

If this averaging over all the possible parameter values seems a strange idea, it is probably easier to understand it by first considering the discrete case. You can for example take a look at the how the denominator of the Bayes' theorem is computed in the classical drug testing example: Bayes' theorem - Wikipedia.

In *Bayesian data analysis* (Gelman et al., 2013) the marginal likelihood is called a *prior predictive distribution*. This is because it presents our beliefs about the probabilities of the data *before* any observations are made. It is a distribution of the data computed *as a weighted average over all the possible parameter values*, and the weights are determined by the prior distribution.

If we denote

$$g(\mathbf{y}, \theta) := f_{\mathbf{Y}|\Theta}(\mathbf{y}|\theta),$$

we can write the marginal likelihood as:

$$f_{\mathbf{Y}}(\mathbf{y}) = \int_{\Omega} g(\mathbf{y}, \theta) f_{\Theta}(\theta) d\theta = E[g(\mathbf{y}, \Theta)], \quad (1.2)$$

So the marginal likelihood can be written as an expectation of the sampling distribution, where the expectation is taken over the prior distribution of the parameter Θ ! Again, it may be easier to consider first a case of a discrete parameter, where the expectation is actually computed as an weighted average.

1.3 Prediction

1.3.1 Motivating example, part II

Let's revisit the thumbtack tossing example: assume we have tossed a thumbtack $n = 30$ times, and observed that it has landed point up $y = 16$ times. But oftentimes instead of making inference about the parameters of the model, we are actually more interested in predicting the new observations. So what is our predictive distribution for the number of successes, if we throw the same thumbtack $m = 10$ more times?

Because the thumbtack stays the same, it makes sense to model the new throws as a sample from the same binomial distribution with the same successes probability as the original observations:

$$\tilde{Y} \sim \text{Bin}(m, \Theta)$$

Further, it makes sense to model the old and the new observations independent given the parameter:

$$\tilde{Y}, Y \perp\!\!\!\perp \Theta.$$

A naive way to obtain a probability mass function of $\tilde{Y}$ would be just to plug the point estimate, such as a maximum likelihood estimate $\hat{\theta}_{\text{MLE}}(y)$, as the parameter value of the probability mass function of the new observations: $f_{\tilde{Y}|\Theta}(\tilde{y}|\hat{\theta}_{\text{MLE}}(y))$. However, by identifying the success probability the observed proportion of the successes, we run into the same problems as in the case of the parameter estimation: what if we had again observed a data $y = 0$ with $n = 3$? Then the predictive distribution would assign a probability 1 to the value $\tilde{Y} = n$, and probability 0 to all the other values. Surely we would have not needed any statistics to arrive at the conclusion that the thumbtack will land point down every time!

Instead, we will derive the proper Bayesian predictive distribution by actually computing the probability of the new observations given the observed data! This is denoted by $f_{\tilde{Y}|Y}(\tilde{y}|y)$. We can immediately observe that the parameter θ does not exist at all in this formula. However, to derive the predictive distribution, we include the parameter as an auxiliary variable that is then integrated out. We first specify the joint distribution of the new observation $\tilde{y}$ and the parameter θ given the observed data $\tilde{y}$, and then get the predictive distribution by integrating over the parameter space:

$$\begin{aligned} f_{\tilde{Y}|Y}(\tilde{y}|y) &= \int_{\Omega} f_{\tilde{Y},\Theta|Y}(\tilde{y}|y) d\theta \\ &= \int_{\Omega} f_{\tilde{Y}|\Theta,Y}(\tilde{y}|\theta,y) f_{\Theta|Y}(\theta|y) d\theta \\ &= \int_{\Omega} f_{\tilde{Y}|\Theta}(\tilde{y}|\theta) f_{\Theta|Y}(\theta|y) d\theta. \end{aligned} \tag{1.3}$$

In the second equality we used a chain rule for the conditional probability densities:

$$f_{X,Y|Z} = f_{X|Y,Z} f_{Y|Z},$$

and in the final equality used a fact that the new observations are independent of the observed data given the parameter to simplify the expression. This predictive distribution $f_{\tilde{Y}|Y}(\tilde{y}|y)$ of the new observations given the data we just derived is known as a **posterior predictive distribution**.

Now that we derived a general form of the posterior predictive distribution, we can plug the sampling distribution of the new observations $f_{\tilde{Y}|\Theta}(\tilde{y}|\theta)$ and the posterior distribution $f_{\Theta|Y}(\theta|y)$ we derived in the part one of this example, into this formula:

$$\begin{aligned} f_{\tilde{Y}|Y}(\tilde{y}|y) &= \int_{\Omega} f_{\tilde{Y}|\Theta}(\tilde{y}|\theta) f_{\Theta|Y}(\theta|y) d\theta \\ &= \int_0^1 \binom{m}{\tilde{y}} \theta^{\tilde{y}} (1-\theta)^{m-\tilde{y}} \frac{1}{B(\alpha_1, \beta_1)} \theta^{\alpha_1-1} (1-\theta)^{\beta_1-1} d\theta \\ &= \binom{m}{\tilde{y}} \frac{1}{B(\alpha_1, \beta_1)} \int_0^1 \theta^{\tilde{y}+\alpha_1-1} (1-\theta)^{m+\beta_1-\tilde{y}-1} d\theta. \end{aligned}$$

To simplify the notation, we have denoted the parameters of the posterior distribution as $\alpha_1 = y + 1$, and $\beta_1 = n - y + 1$.

Next we are going to integrate in “a statistician way”: this means that we are not going to really integrate the expression, but we get rid of it by recognizing it as the integral whose value we know. We can do this by using one of the following tricks:

1. Explicitly recognize a familiar integral : We can immediately observe that the integral is a beta function (see eq. (1.1)), so we can write it more concisely as:

$$\int_0^1 \theta^{\tilde{y}+\alpha_1-1} (1-\theta)^{m+\beta_1-\tilde{y}-1} d\theta = B(\tilde{y} + \alpha_1, m + \beta_1 - \tilde{y}).$$

2. Recognize an unnormalized probability density function of the familiar distribution : We can also immediately observe that the integrand is a probability density function of the beta distribution

Beta($\tilde{y} + \alpha_1, m + \beta_1 - \tilde{y}$) up to a normalizing constant, and it is integrated over the support of the distribution. This means that if we add the missing normalizing constant, the integral is an integral of the probability density over its support:

$$\begin{aligned}
 & \int_0^1 \theta^{\tilde{y} + \alpha_1 - 1} (1 - \theta)^{m + \beta_1 - \tilde{y} - 1} d\theta \\
 &= B(\tilde{y} + \alpha_1, m + \beta_1 - \tilde{y}) \int_0^1 \frac{1}{B(\tilde{y} + \alpha_1, m + \beta_1 - \tilde{y})} \theta^{\tilde{y} + \alpha_1 - 1} (1 - \theta)^{m + \beta_1 - \tilde{y} - 1} d\theta \\
 &= B(\tilde{y} + \alpha_1, m + \beta_1 - \tilde{y}) \cdot 1 \\
 &= B(\tilde{y} + \alpha_1, m + \beta_1 - \tilde{y}).
 \end{aligned}$$

In this case the first trick was more straight-forward, but I also introduced the second one because in some cases recognizing the familiar integral requires performing a change of variables, and an unnormalized density function of the familiar distribution may be easier to recognize.

Whichever of these tricks you use, the posterior predictive distribution is simplified to

$$f_{\tilde{Y}|Y}(\tilde{y}|y) = \binom{m}{\tilde{y}} \frac{B(\tilde{y} + \alpha_1, m + \beta_1 - \tilde{y})}{B(\alpha_1, \beta_1)}.$$

This is a probability distribution of the so called beta-binomial distribution, so we can denote our posterior predictive distribution as

$$\tilde{Y}|Y \sim \text{Beta-bin}(m, \alpha_1, \beta_1),$$

where $\alpha_1 = y + 1$, and $\beta_1 = n - y + 1$ are the parameters of the posterior distribution for the parameter Θ .

1.3.2 Posterior predictive distribution

Let's consider a general case: assume we have observations $\mathbf{Y} = (Y_1, \dots, Y_n)$ with a sampling distribution $f_{\mathbf{Y}|\Theta(\mathbf{y}|\theta)}$ conditional on the unknown parameter vector $\Theta \in \Omega$. Now we want to predict the distribution for the m new observations $\tilde{\mathbf{Y}} = (\tilde{Y}_1, \dots, \tilde{Y}_m)$ from the same process. Distribution

$$f_{\tilde{\mathbf{Y}}|\mathbf{Y}}(\tilde{\mathbf{y}}|\mathbf{y})$$

of the new observations given the observed data is called a *posterior predictive distribution*. If we further make a simplifying assumption that the new observations are independent of the observed data given the parameter, written as:

$$\tilde{\mathbf{Y}}, \mathbf{Y} | \Theta,$$

we can write the posterior predictive distribution as an integral

$$f_{\tilde{\mathbf{Y}}|\mathbf{Y}}(\mathbf{y}|\mathbf{y}) = \int_{\Omega} f_{\tilde{\mathbf{Y}}|\Theta}(\tilde{\mathbf{y}}|\theta) f_{\Theta|\mathbf{Y}}(\theta|\mathbf{y}) d\theta,$$

which we derived in Equation (1.3). This formula may seem a little bit intimidating at first, but let's try to find the intuition behind it.

The diagram shows the formula for the posterior predictive distribution: $f_{\tilde{\mathbf{Y}}|\mathbf{Y}}(\mathbf{y}|\mathbf{y}) = \int_{\Omega} f_{\tilde{\mathbf{Y}}|\Theta}(\tilde{\mathbf{y}}|\theta) f_{\Theta|\mathbf{Y}}(\theta|\mathbf{y}) d\theta$. Annotations include:

- A green arrow pointing to $f_{\tilde{\mathbf{Y}}|\Theta}(\tilde{\mathbf{y}}|\theta)$ with the text "Sampling distribution for the new observations".
- A red arrow pointing to $f_{\Theta|\mathbf{Y}}(\theta|\mathbf{y})$ with the text "Posterior distribution".
- A blue arrow pointing to the $d\theta$ term with the text "Integrated over the parameter space".

The integrand in the formula is a product of the sampling distribution for the new observations given the parameter, and the posterior distribution of the parameter given the old observations. When we denote the sampling distribution for the new observations as

$$g(\tilde{\mathbf{y}}, \theta) := f_{\tilde{\mathbf{Y}}|\Theta}(\tilde{\mathbf{y}}|\theta),$$

we can write the posterior predictive distribution as

$$f_{\tilde{\mathbf{Y}}|\mathbf{Y}}(\mathbf{y}|\mathbf{y}) = \int_{\Omega} g(\tilde{\mathbf{y}}, \theta) f_{\Theta|\mathbf{Y}}(\theta|\mathbf{y}) d\theta = E[g(\tilde{\mathbf{y}}, \theta) | \mathbf{Y} = \mathbf{y}].$$

where the expectation is taken over the posterior distribution $f_{\mathbf{Y}|\Theta}$. Like marginal likelihood (see Equation (1.2)), posterior predictive distribution is also a weighted average of the sampling distribution over the parameter values. However, the marginal likelihood was an unconditional expectation and the weights of the parameter values came from the prior distribution, whereas the posterior predictive distribution is a conditional expectation (conditioned on the observed data $\mathbf{Y} = \mathbf{y}$) and weights for the parameter values come from the posterior distribution.

The posterior predictive distribution takes into account also the uncertainty of our parameter estimates, which is quantified by the posterior distribution. Thus, the variance of the posterior predictive distribution is in general higher than the variance of the sampling distribution into which a point estimate for the parameter θ , for example the maximum likelihood estimate or the posterior mean, is plugged.

1.3.3 Short note about the notation

In this introduction chapter we used quite a verbose notation: we explicitly wrote the random variables whose density functions we were handling as subscripts: for example we denoted the conditional density of random variable $\mathbf{Y}$ given $\Theta = \theta$ as:

$$f_{\mathbf{Y}|\Theta}(\mathbf{y}|\theta).$$

This makes it immediately clear which densities we are handling, but when the formulas get longer, using this heavy notation may become quite cumbersome. This is why in statistics and machine learning literature a more concise notation is generally used. In this slight abuse of notation all the density and probability mass functions are denoted with the same letter (usually p) without any subscripts. The random variables whose density functions they are can be recognized by the *arguments* of the densities. For example the conditional density $f_{\mathbf{Y}|\Theta}(\mathbf{y}|\theta)$ is written concisely as $p(\mathbf{y}|\theta)$, and the Bayes' theorem can be written as

$$p(\theta|\mathbf{y}) = \frac{p(\theta)p(\mathbf{y}|\theta)}{p(\mathbf{y})}.$$

This shorthand notation makes formulas shorter and more clear to read assuming that you know in the first place for which it is shorthand for. In the following chapters we will use this notation.

Often also the random variables and their realizations are denoted with the same lowercase letter if there is no risk of confusion. This is particularly the case with the parameters, in part because there exist no useful uppercase versions of many greek alphabets. So when we talk about “the parameter θ ” in the following chapters, you have to remember that usually a random variable is meant.

Chapter 2

Conjugate distributions

Conjugate distribution or **conjugate pair** means a pair of a sampling distribution and a prior distribution for which the resulting posterior distribution belongs into the same parametric family of distributions than the prior distribution. We also say that the prior distribution is a **conjugate prior** for this sampling distribution.

A parametric family of distributions

$$\{f_{Y|\Theta}(y|\theta) : \theta \in \Omega\}$$

means simply a set of distributions which have a same functional form, and differ only by the value of the finite-dimensional parameter $\theta \in \Omega$. For instance, all beta distributions or all normal distributions form a parametric families of distributions.

We have already seen one example of the conjugate pair in the thumbtack tossing example: the binomial and the beta distribution. You may now be wondering: “But Ville, in our example the prior distribution was an uniform distribution, not a beta distribution??” It turns out that the prior was indeed a beta distribution, because the uniform distribution $U(0, 1)$ is actually a same distribution than the beta distribution $\text{Beta}(1, 1)$ (check that this holds!).

Using conjugate pairs of distributions makes a life of the statistician more convenient, because the marginal likelihood, and thus also the posterior distribution and the posterior predictive distribution can be solved in a closed form. Actually, it turns out that this is the second of the only two special cases in which this is possible:

1. The parameter space is discrete and finite: $\Omega = (\theta_1, \dots, \theta_p)$; in this case the marginal likelihood can be computed as a finite sum:

$$f_Y(\mathbf{y}) = \sum_{i=1}^p f_{Y|\Theta}(\mathbf{y}_i|\theta_i) f_{\Theta}(\theta_i).$$

2. The prior distribution is a conjugate prior for the sampling distribution.

In all the other cases we have to approximate the posterior distributions and the posterior predictive distributions. Usually this is done by simulating values from them; we will return to this topic soon.

2.1 One-parameter conjugate models

When parameter $\Theta \in \Omega$ is a scalar, the inference is particularly simple. We have already seen one example of the one-parameter conjugate model (the thumbtacking example), but let’s examine another simple model.

2.1.1 Example: Poisson-gamma model

A Poisson distribution is a discrete distribution which can get any non-negative integer values. It is a natural distribution for modelling counts, such as goals in a football game, or a number of bicycles passing a certain point of the road in one day. Both the expected value and the variance of a Poisson distributed random variable are equal to the parameter of the distribution: if $Y \sim \text{Poisson}(\lambda)$,

$$E[Y] = \lambda, \quad \text{Var}[Y] = \lambda.$$

Let's cheat a little bit this time: we will first generate observations from the distribution with a known parameter, and then try estimate the posterior distribution of the parameter from this data:

```
n <- 5
lambda_true <- 3

# set seed for the random number generator, so that we get replicable results
set.seed(111111)
y <- rpois(n, lambda_true)
y

## [1] 4 3 11 3 6
```

Now we actually know that the true generating distribution of our observations $y = (4, 3, 11, 3, 6)$ is $\text{Poisson}(3)$; but let's forget this for a moment, and proceed with the inference.

Assume that the observed variables are counts, which means that they can in principle take any non-negative integer value. Thus, it is natural to model them as independent Poisson-distributed random variables:

$$Y_1, \dots, Y_n \sim \text{Poisson}(\lambda) \perp\!\!\!\perp \lambda$$

Because the parameter of the Poisson distribution can in principle be any positive real number, we want use a prior whose support is $(0, \infty)$. If we used for example an uniform prior $U(0, 100)$, posterior density would also be zero outside of this interval, even if all the observations were greater than 100. So usually we want a prior that assigns a non-zero density for all the possible parameter values.

It is not possible to set a uniform distribution over the infinite interval $(0, \infty)$, so we have to come up with something else. A gamma distribution is a convenient choice. It is a distribution with a peak close to zero, and a tail that goes to infinity. It also turns out that the gamma distribution is a conjugate prior for the Poisson distribution: this means that we can actually solve the posterior distribution in a closed form.

We can set the parameters of the prior distribution for example to $\alpha = 1$ and $\beta = 1$; we will examine the choice of both the prior distribution and its parameters (called hyperparameters) later. For now on, let's just solve the posterior with the conjugate gamma prior:

$$\lambda \sim \text{Gamma}(\alpha, \beta).$$

Because the observations are independent given the parameter, a likelihood function for all the observations $\mathbf{Y} = (Y_1, \dots, Y_n)$ can be written as a product of the Poisson distributions:

$$p(\mathbf{y}|\lambda) = \prod_{i=1}^n p(y_i|\lambda) = \prod_{i=1}^n \lambda^{y_i} \frac{e^{-\lambda}}{y_i!} \propto \lambda^{\sum_{i=1}^n y_i} e^{-n\lambda} = \lambda^{n\bar{y}} e^{-n\lambda},$$

where

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

is a mean of the observations. Again we dropped the constant terms which do not depend on the parameter from the expression of the likelihood.

The unnormalized posterior distribution for the parameter λ can now be written as

$$\begin{aligned} p(\lambda|\mathbf{y}) &\propto p(\mathbf{y}|\lambda)p(\lambda) \\ &\propto \lambda^{n\bar{y}} e^{-n\lambda} \lambda^{\alpha-1} e^{-\beta\lambda} \\ &= \lambda^{\alpha+n\bar{y}-1} e^{-(\beta+n)\lambda}. \end{aligned} \quad (2.1)$$

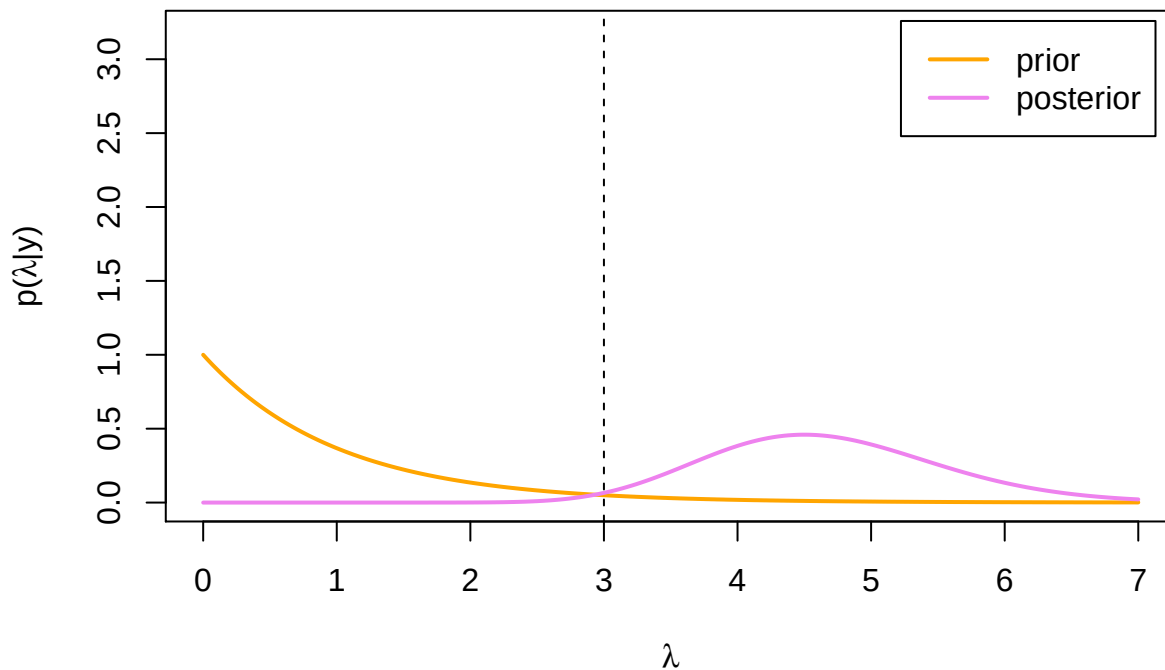
The gamma prior was chosen because a gamma distribution is a conjugate prior for the Poisson distribution, and indeed we can recognize the unnormalized posterior distribution as the kernel of the gamma distribution. Thus, the posterior distribution is

$$\lambda | \mathbf{Y} \sim \text{Gamma}(\alpha + n\bar{y}, \beta + n).$$

We can now plot the prior and the posterior distributions:

```
alpha <- 1
beta <- 1

lambda <- seq(0,7, by = 0.01) # set up grid for plotting
plot(lambda, dgamma(lambda, alpha, beta), type = 'l', lwd = 2, col = 'orange',
      ylim = c(0, 3.2), xlab = expression(lambda),
      ylab = expression(paste('p(', lambda, '|y)'))
lines(lambda, dgamma(lambda, alpha + sum(y), beta + n),
      type = 'l', lwd = 2, col = 'violet')
abline(v = lambda_true, lty = 2)
legend('topright', inset = .02, legend = c('prior', 'posterior'),
      col = c('orange', 'violet'), lwd = 2)
```



We can see that the posterior distribution is concentrated quite a bit higher than the true parameter value. This is because our third observation happened to be a bit of an outlier: the probability of drawing a value of 11 or higher from $\text{Poisson}(3)$ -distribution (if we draw only one value), is only:

```
ppois(10,3, lower.tail = FALSE)
```

```
## [1] 0.000292337
```

But because we are anyway using simulated data, let's draw some more observations from the same Poisson(3)-distribution:

```
n_total <- 200
set.seed(111111) # use same seed, so first 5 obs. stay same
y_vec <- rpois(n_total, lambda_true)
head(y_vec)
```

```
## [1] 4 3 11 3 6 3
```

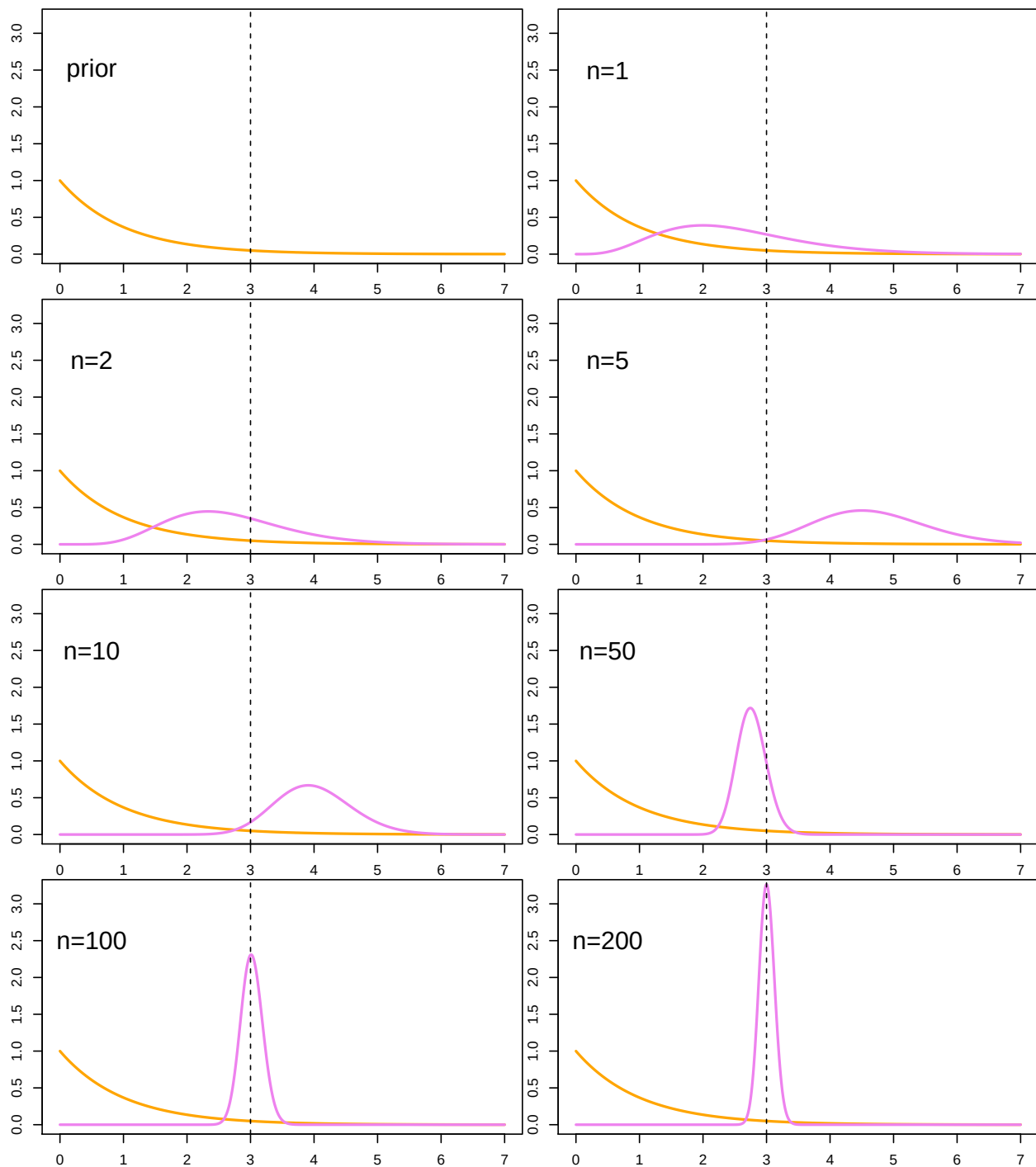
and plot the posterior distributions with different sample sizes to see if things even out:

```
n_vec <- c(1, 2, 5, 10, 50, 100, 200)

par(mfrow = c(4,2), mar = c(2, 2, .1, .1))

plot(lambda, dgamma(lambda, alpha, beta), type = 'l', lwd = 2, col = 'orange',
      ylim = c(0, 3.2), xlab = '', ylab = '')
abline(v = lambda_true, lty = 2)
text(x = 0.5, y = 2.5, 'prior', cex = 1.75)

for(n_crnt in n_vec) {
  y_sum <- sum(y_vec[1:n_crnt])
  plot(lambda, dgamma(lambda, alpha, beta), type = 'l', lwd = 2, col = 'orange',
        ylim = c(0, 3.2), xlab = '', ylab = '')
  lines(lambda, dgamma(lambda, alpha + y_sum, beta + n_crnt),
        type = 'l', lwd = 2, col = 'violet')
  abline(v = lambda_true, lty = 2)
  text(x = 0.5, y = 2.5, paste0('n=', n_crnt), cex = 1.75)
}
```



After the first two observations the posterior is still quite close to the prior distribution, but the third observation, which was an outlier, shifts the peak of the posterior from the left side of the mean heavily to the right. But when more observations are drawn, we can observe that the posterior starts to concentrate more heavily on the neighborhood of the true parameter value.

2.1.2 Example: prediction in Poisson-gamma model

Let's denote the parameters of the posterior distribution computed in the previous example as

$$\alpha_1 := \alpha + n\bar{y}$$

and

$$\beta_1 := \beta + n,$$

and solve the posterior predictive distribution for one new observation $\tilde{Y}_1$ from the same Poisson distribution as the observed data:

$$\tilde{Y}_1, Y_1, \dots, Y_n \sim \text{Poisson}(\lambda) \perp\!\!\!\perp \lambda.$$

The posterior predictive distribution for $\tilde{Y}_1$ can be written as:

$$\begin{aligned} p(\tilde{y}_1 | \mathbf{y}) &= \int_{\Omega} p(\tilde{y}_1 | \lambda) p(\lambda | \mathbf{y}) d\lambda \\ &= \int_0^{\infty} \lambda^{\tilde{y}_1} \frac{e^{-\lambda}}{\tilde{y}_1!} \frac{\beta_1^{\alpha_1}}{\Gamma(\alpha_1)} \lambda^{\alpha_1-1} e^{-\beta_1 \lambda} d\lambda \\ &= \frac{\beta_1^{\alpha_1}}{\Gamma(\alpha_1) \tilde{y}_1!} \int_0^{\infty} \lambda^{\tilde{y}_1 + \alpha_1 - 1} e^{-(\beta_1 + 1)\lambda} d\lambda. \end{aligned}$$

Now it would be probably easiest to use the first of the tricks introduced in Example 1.3.1, and complete the integral into an integral of a gamma density over its support. But just to make things more interesting, let's use the second trick by completing it into a gamma function by the following change of variables:

$$t = (\beta_1 + 1)\lambda.$$

Now

$$\lambda = g(t) := \frac{t}{\beta_1 + 1},$$

and

$$d\lambda = g'(t) dt = \frac{1}{\beta_1 + 1} dt.$$

This change of variables is only a multiplication by a positive constant, so it has no effect on the limits of the integral. After performing the change of variables we can recognize the gamma integral:

$$\begin{aligned} \int_0^{\infty} \lambda^{\tilde{y}_1 + \alpha_1 - 1} e^{-(\beta_1 + 1)\lambda} d\lambda &= \int_0^{\infty} \left(\frac{t}{\beta_1 + 1} \right)^{\tilde{y}_1 + \alpha_1 - 1} e^{-t} \frac{1}{\beta_1 + 1} dt \\ &= \left(\frac{1}{\beta_1 + 1} \right)^{\tilde{y}_1 + \alpha_1} \int_0^{\infty} t^{\tilde{y}_1 + \alpha_1 - 1} e^{-t} dt \\ &= \left(\frac{1}{\beta_1 + 1} \right)^{\tilde{y}_1 + \alpha_1} \Gamma(\tilde{y}_1 + \alpha_1). \end{aligned}$$

Thus, we can write the posterior predictive density as

$$\begin{aligned} p(\tilde{y}_1 | \mathbf{y}) &= \frac{\beta_1^{\alpha_1}}{\Gamma(\alpha_1) \tilde{y}_1!} \cdot \left(\frac{1}{\beta_1 + 1} \right)^{\tilde{y}_1 + \alpha_1} \Gamma(\tilde{y}_1 + \alpha_1) \\ &= \frac{\Gamma(\tilde{y}_1 + \alpha_1)}{\Gamma(\alpha_1) \tilde{y}_1!} \left(\frac{1}{\beta_1 + 1} \right)^{\tilde{y}_1} \left(\frac{\beta_1}{\beta_1 + 1} \right)^{\alpha_1} \\ &= \frac{\Gamma(\tilde{y}_1 + \alpha_1)}{\Gamma(\alpha_1) \tilde{y}_1!} \left(1 - \frac{\beta_1}{\beta_1 + 1} \right)^{\tilde{y}_1} \left(\frac{\beta_1}{\beta_1 + 1} \right)^{\alpha_1}. \end{aligned}$$

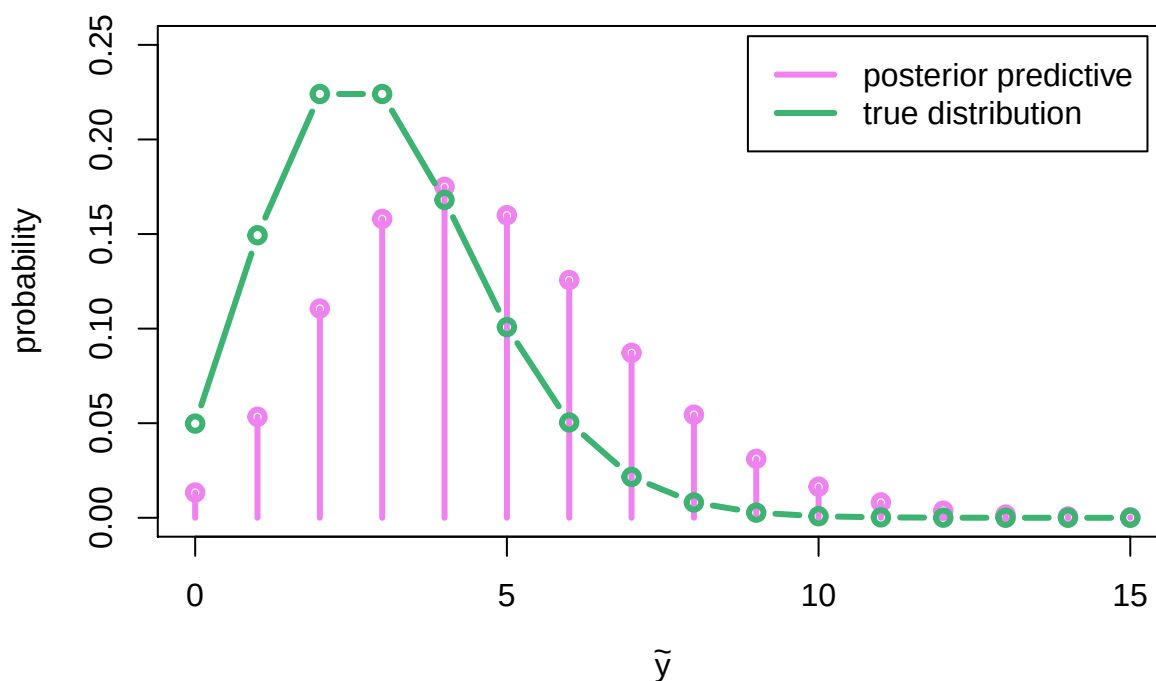
This is a density function of the following negative binomial distribution:

$$\tilde{Y}_1 | \mathbf{Y} \sim \text{Neg-Bin} \left(\alpha_1, \frac{\beta_1}{\beta_1 + 1} \right).$$

Still assuming that our prior was Gamma(1,1)-distribution, we can compare this posterior predictive distribution to the true generative distribution of the data:

```
y_grid <- 0:15
alpha_1 <- alpha + sum(y)
beta_1 <- beta + n

plot(y_grid, dnbinom(y_grid, size = alpha_1, prob = beta_1 / (1 + beta_1)),
     type = 'h', lwd = 3, col = 'violet', xlab = expression(tilde(y)),
     ylab = 'probability', ylim = c(0, 0.25))
lines(y_grid, dnbinom(y_grid, size = alpha_1, prob = beta_1 / (1 + beta_1)),
      type = 'p', lwd = 3, col = 'violet')
lines(y_grid, dpois(y_grid, lambda_true), type = 'b', lwd = 3, col = 'mediumseagreen')
legend('topright', inset = .02,
      legend = c('posterior predictive', 'true distribution'),
      col = c('violet', 'mediumseagreen'), lwd = 3)
```



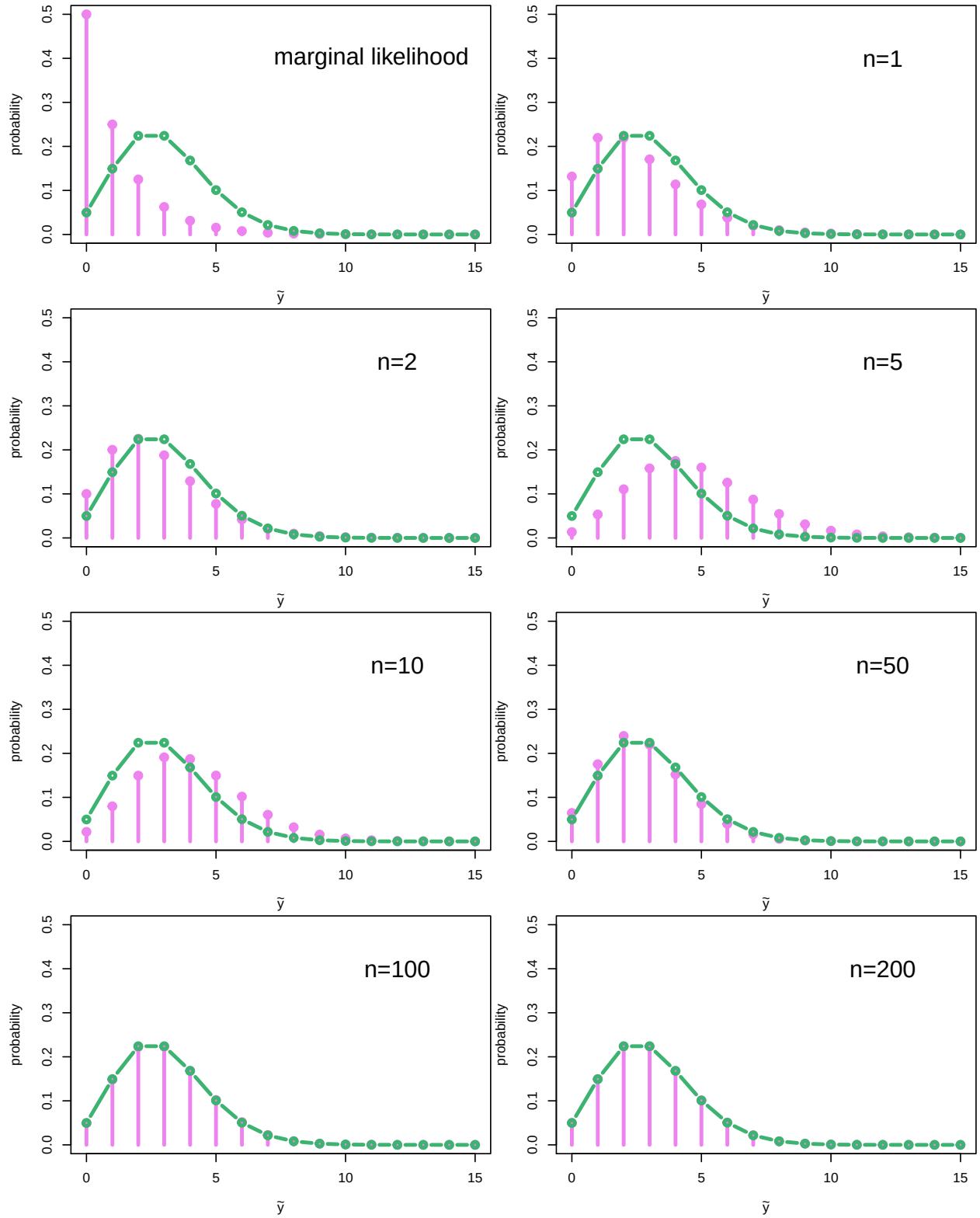
As could be expected based on the posterior distribution for parameter λ , which was concentrated on the larger values than the true value $\lambda = 3$, also the posterior predictive distribution is concentrated (remember that the expected value of Poisson distribution is its parameter) on the higher values compared to the generating distribution Poisson(3).

Let's see what the posterior predictive distribution looks like for the different sample sizes (using the data we generated earlier):

```
par(mfrow = c(4,2), mar = c(4, 4, .1, .1))

plot(y_grid, dnbinom(y_grid, size = alpha, prob = beta / (1 + beta)),
     type = 'h', lwd = 3, col = 'violet', xlab = expression(tilde(y)),
     ylab = 'probability', ylim = c(0, 0.5))
lines(y_grid, dnbinom(y_grid, size = alpha, prob = beta / (1 + beta)),
      type = 'p', lwd = 3, col = 'violet')
lines(y_grid, dpois(y_grid, lambda_true), type = 'b', lwd = 3, col = 'mediumseagreen')
text(x = 11, y = 0.4, 'marginal likelihood', cex = 1.75)
```

```
for(n_crnt in n_vec) {  
  y_sum <- sum(y_vec[1:n_crnt])  
  alpha_1 <- alpha + y_sum  
  beta_1 <- beta + n_crnt  
  plot(y_grid, dnbinom(y_grid, size = alpha_1, prob = beta_1 / (1 + beta_1)),  
        type = 'h', lwd = 3, col = 'violet', xlab = expression(tilde(y)),  
        ylab = 'probability', ylim = c(0, 0.5))  
  lines(y_grid, dnbinom(y_grid, size = alpha_1, prob = beta_1 / (1 + beta_1)),  
        type = 'p', lwd = 3, col = 'violet')  
  lines(y_grid, dpois(y_grid, lambda_true), type = 'b', lwd = 3, col = 'mediumseagreen')  
  text(x = 12, y = 0.4, paste0('n=', n_crnt), cex = 1.75)  
}
```



The first plot contains actually the marginal likelihood for one observation Y_1 :

$$p(y_1) = \int_{\Omega} p(y_1|\lambda)p(\lambda) d\lambda$$

This marginal likelihood is $\text{Neg-bin}\left(\alpha, \frac{\beta}{\beta+1}\right)$ -distribution. We already basically derived this when we computed the posterior predictive distribution; the only difference was in the parameters of the gamma distribution. This also holds in a more general case: the derivation for the marginal likelihood and the posterior predictive distribution is the same; the only difference is in the value of the parameters of the conjugate prior distribution. This means that every time we can solve the posterior distribution in a closed form, we can also solve the posterior predictive distribution!

But I digress... Let's look at the plots again: when we have only one or two observations, the posterior predictive distribution is closer to the marginal likelihood. Again, the third observation, which was the outlier, tilts the posterior predictive distribution immediately towards the higher values, until the it starts to resemble more or less the true generating distribution when more data is generated.

This is recurring theme in a Bayesian inference: when the sample size is small, the prior has more influence on the posterior, but when the sample size grows, the data starts to influence our posterior distribution more and more, until at the limit the posterior is determined purely by the data (at least when the certain conditions hold). Examining the case $n \rightarrow \infty$ is called asymptotics, and it is a cornerstone of the statistical inference, but we do not have time go very deep into this topic on this course.

Now you may be thinking: “But if have enough data, then we do not have to care about the priors, don't we?” Well, in this case you are lucky, but before you can forget about the priors, you have to ask yourself (at least) two things:

1. How complex model you want to fit? In general, more complex the model, more data you need. For example modern deep learning models may have millions of parameters, so probably a sample size of $n = 50$ is not “high enough”, although this was the case in our toy example.
2. In what resolution level you want examine your data? You may have enough data to fit your model at the level of the country, but what if you want to model the differences between the towns? Or the neighborhoods? We will actually have a concrete example of this exact situation on the exercises later.

2.2 Prior distributions

The most often criticized aspect of the Bayesian approach to statistical inference is the requirement to choose a prior distribution, and especially the subjectivity of this prior selection procedure. The Bayesian answer to this criticism is to point out that the whole modeling procedure is inherently subjective: it is never possible for the data to fully “speak for itself” because we have to always make some assumptions about its sampling distribution.

Even in the most trivial coin-flipping example the choice of the binomial distribution for the outcome of the coinflip can be questioned: if we were truly ignorant about the outcome of the coinflip, would it make sense to model the outcome with a trinomial distribution, where the outcomes were head, tails and the coin landing on its side? So even the choice of the restricting the parameter space to $\Omega = \{\text{heads, tails}\}$ is based on the our prior knowledge about the previous coinflips and the common sense knowledge that the coin landing on its side is almost impossible. It can be argued that we always use somehow our prior knowledge in the modelling process, but the Bayesian framework just makes utilizing prior knowledge more transparent and easier to quantify.

A less philosophical and more practical example of the inherent subjectivity of the modelling process is any situation in which our observations are continuous instead of the discrete. For instance, let's consider a classical statistical problem of estimating the true population distribution of some quantity, say the average height of adult females, on the basis of the subsample from some human population. Assume that we have measured the following heights of the five people from this population, say some tribe in South America (in metres):

$$\mathbf{y} = (1.563, 1.735, 1.642, 1.662, 1.528).$$

Now we could of course “let the data speak for itself”, and assume that the true distribution of the height of the females of this tribe is the empirical distribution of our observations:

$$P(Y = y) = \begin{cases} 1/5 & \text{if } y = 1.563, \\ 1/5 & \text{if } y = 1.735, \\ 1/5 & \text{if } y = 1.642, \\ 1/5 & \text{if } y = 1.662, \\ 1/5 & \text{if } y = 1.528, \\ 0 & \text{otherwise.} \end{cases}$$

But this would of course be an absurd conclusion. In practice, we have to impose some kind of the sampling distribution, for example the normal distribution, for the observations for our inferences to be sensible. Even if we do not want to impose any parametric distribution on the data, we have to choose some nonparametric method to smooth a height distribution.

So this is the Bayesian counter-argument: the choice of the *sampling distribution* is as subjective as the choice of the prior distribution. Take for instance a classical linear regression. It makes huge simplifying assumptions: that the true that the error terms are normally distributed given the predictors, and that the parameters of this normal distribution do not depend on the values of the predictors. Also the choices of the predictors inject very strong subjective beliefs into the model: if we exclude some predictors from the model, this means that we assume that this predictor has no effect at all on the output variable. If we do not include any second or higher order terms, this means that we make a rather dire assumption that the all the relationships between the predictors and the output variables are linear, and so on.

Of course the models with different predictors and model structures can be tested (for example by predicting on the test set or by cross-validation), and then the best model can be chosen, but the same thing can be also done for the prior distributions. So we do not have to choose the first prior distribution or hyperparameters that we happen to test, but like the different sampling distributions, we can also test different prior distributions and hyperparameter values to see which of them make sense. This kind of the comparing the effects of the choice of prior distribution is called **sensitivity analysis**.

Besides being the most criticized aspect of the Bayesian inference, the choice of the prior distribution is also one of the hardest. Often there are not any “right” priors, but the usual choices are often based on the computational convenience or desired statistical properties.

2.2.1 Informative priors

If we have prior knowledge about the possible parameter values, it often makes sense to limit the sampling to these parameter values. The prior distribution which is designed to encode our prior knowledge of the likely parameter values and to affect the posterior distribution with small sample sizes is called an **informative prior**. Using informative prior often makes the solution more **stable** with the smaller sample sizes, and on the other hand the sampling from the posterior is often more efficient when informative prior is used, because then we do not waste too much energy sampling the highly improbable regions of the parameter space.

However, when using an informative prior distribution, it is better to use soft instead of the hard restrictions on the possible parameter values. Let’s illustrate this by returning to the problem of estimating the distribution of the mean height of the females of some population, and assume that we model the height by the normal distribution $N(\mu, \sigma^2)$. Because the estimated parameter μ is a mean of the height of adult females, it would make sense to limit the possible parameter values to the interval $(0.5, 2.5)$ because clearly it is impossible for the mean height of the adults be outside of this interval; this can be done by using as a prior the uniform distribution

$$\mu \sim U(0.5, 2.5).$$

This prior has the probability mass of zero outside of this interval; thus also the value of the posterior distribution for μ is zero outside of this interval. In this example it actually makes sense to use this kind

of the prior because it is based on the natural constraints of the human height. However, in general this approach has two weaknesses:

1. If the posterior mean falls near one of the limits of this interval, the interval “cuts” the posterior distribution. Also the sampling works worse near the limit.
2. Often this kind of the uniform prior on the interval gives undue influences to the extreme values which are near the limits.

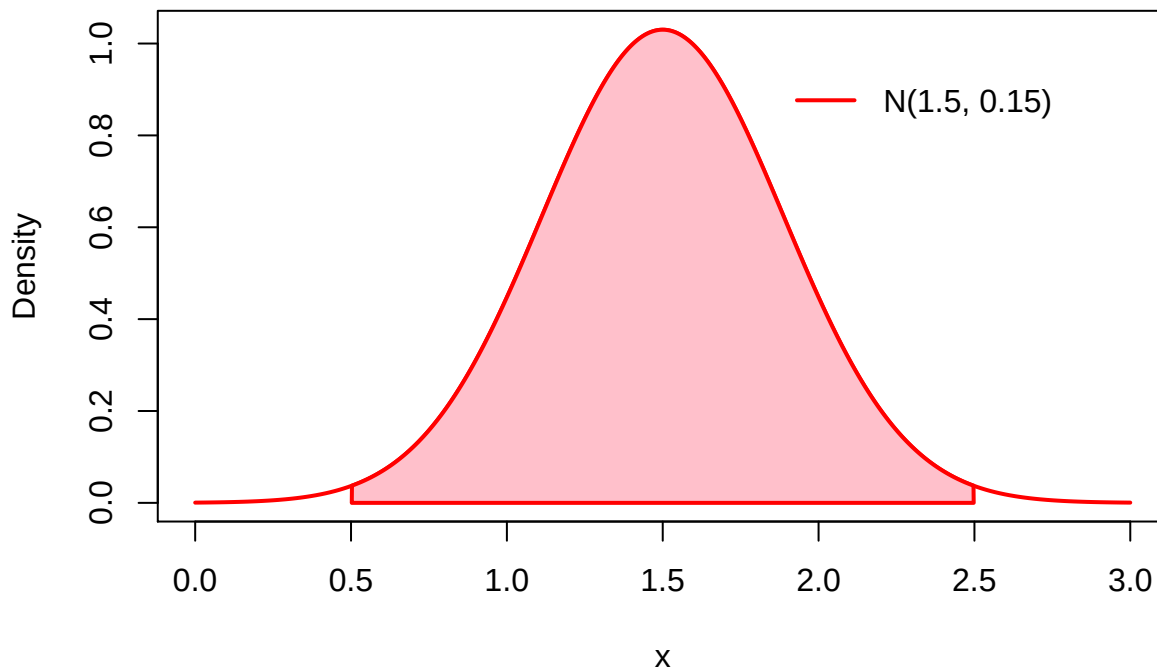
Both of these problems can be circumvented by using a prior which has most of its probability mass on the interval where the true parameter value is assumed to surely lie, but that does not limit it to this interval. For this example this kind of the prior which sets “soft” limits to the parameter values would be for example the normal distribution with mean 1.5 and variance 0.15:

$$\mu \sim N(1.5, 0.15).$$

This normal distribution has approximately 99% of its probability mass (pink area under the curve) on the interval (0.5, 2.5), but does not limit the parameter values to this interval¹:

```
x <- seq(0,3, by = .001)
mu <- 1.5
sigma <- sqrt(.15)
plot(x, dnorm(x, mu, sigma), type = 'l', col = 'red', lwd = 2, ylab = 'Density')

q_lower <- qnorm(.005, mu, sigma)
q_upper <- qnorm(.995, mu, sigma)
y_val <- dnorm(x, mu, sigma)
x_coord <- c(q_lower, x[x >= q_lower & x <= q_upper], q_upper)
y_coord <- c(0, y_val[x >= q_lower & x <= q_upper], 0)
polygon(x_coord, y_coord, col = 'pink', lwd = 2, border = 'red')
legend('topright', legend='N(1.5, 0.15)', col='red', inset=.1, lwd=2, bty='n')
```



This distribution has also a pleasant property that it pulls the posterior distribution towards the center of the distribution. Informative priors can be based on our prior knowledge of the examined phenomenon. For

¹Of course the height cannot be negative... maybe it could be better to choose a gamma or some other distribution whose support is positive real axis for our prior. But the normal distribution is a very convenient choice for this example because its parameters have direct interpretations as the mean and the variance of the distribution.

instance, this prior distribution may be an observed distribution of the means of the heights of the females of the all South-American tribes measured. We will return to the topic of combining inferences from the several subpopulations in the chapter about hierarchical models. If there is no this kind of the prior knowledge, it is better to use a non-informative prior, or at least to set a variance of the prior quite high.

2.2.2 Non-informative priors

A **non-informative** or **uninformative** prior is a prior distribution which is designed to influence the posterior distribution as little as possible. It makes sense to use a non-informative prior in situations in which we do not have any clear prior beliefs about the possible parameter values, or we do not want these prior beliefs to influence the inference process.

Non-informative and informative prior are not formally defined terms. They are better be thought as a continuum: some prior distributions are more informative than others. However, often some prior distribution are clearly non-informative and some are informative, but it is important to remember that this distinction is just a heuristic, not any definition.

But what kind of the prior distribution is non-informative? An intuitive answer would be an uniform distribution. This was also a suggestion of the pioneers of the Bayesian inference, Bayes and Laplace. But as we observed in the beta-binomial example 1.1.3, in the binomial model with beta prior the uniform prior $\text{Beta}(1,1)$ actually corresponds to having two pseudo-observations: one failure and one success. So it is not completely uninformative. Another problem with the uniform priors are that they are not invariant with respect to parametrization: if we change to parametrization of the likelihood, the prior is not uniform anymore. We will explore this phenomenon for the beta-binomial model in the exercises.

2.2.3 Improper priors

Often the distributions are most non-informative near the limits of their parameter space. For instance, the parameters of the beta prior $\text{Beta}(\alpha, \beta)$ can be thought as the (possibly non-integer) pseudo-observations: α represents pseudo-successes, and β represents pseudo-failures. With this logic the most non-informative prior would be $\text{Beta}(0,0)$. But the problem with this prior is that it is not a probability distribution, because the Beta function approaches infinity when the parameters $\alpha, \beta \rightarrow 0$.

However, it turns out that we can plug this kind of the function that cannot be normalized into the proper probability distribution into the place of the prior in the Bayes' theorem, as long the resulting posterior distribution is a proper probability distribution. We call this kind of the priors that are not densities of any probability distribution as **improper priors**.

In the beta-binomial example we can denote the aforementioned improper prior (known as *Haldane's prior*) as:

$$p(\theta) \propto \theta^{-1}(1 - \theta)^{-1}.$$

It can be easily shown that the resulting posterior is proper as long as we have observed at least one success and one failure.

Improper priors are often obtained as the limits of the proper priors, and they are often used because they are non-informative. We can demonstrate both of these properties with our height estimation example: the noninformative prior for the average height μ would be an uniform distribution over the whole real axis:

$$p(\mu) \propto 1.$$

But of course this cannot be normalized into the probability distribution by dividing it by its integral over the real axis, because this integral is infinite. However, the resulting posterior is a normal distribution if we have at least one observation (assuming known variance). This improper prior can also be interpreted as a normal distribution with infinite variance.

When using improper priors, it is important to check that the resulting posterior is a proper probability distribution.

Chapter 3

Summarizing the posterior distribution

In principle, the posterior distribution contains all the information about the possible parameter values. In practice, we must also present the posterior distribution somehow. If the examined parameter θ is one- or two dimensional, we can simply plot the posterior distribution. Or when we use simulation to obtain values from the posterior, we can draw a histogram or scatterplot of the simulated values from the posterior distribution. If the parameter vector has more than two dimensions, we can plot the marginal posterior distributions of the parameters of interest.

However, we often also want to summarize the posterior distribution numerically. The usual summary statistics, such as the mean, median, mode, variance, standard deviation and different quantiles, that are used to summarize probability distributions, can be used. These summary statistics are often also easier to present and interpret than the full posterior distribution.

3.1 Credible intervals

Credible interval is a “Bayesian confidence interval”. But unlike frequentist confidence intervals, credible intervals have a very intuitive interpretation: it turns out that we can actually say 95% credible interval actually contains a true parameter value with 95% probability! Let’s first define as credible interval more rigorously, and then examine the most common ways to choose the credible intervals.

3.1.1 Credible interval definition

For one-dimensional parameter $\Theta \in \Omega$ (in this section we will also assume that the parameter is continuous, because it makes no sense to talk about the credible *intervals* for the discrete parameter), and confidence level $\alpha \in (0, 1)$, an interval $I_\alpha \subseteq \Omega$ which contains a proportion $1 - \alpha$ of the probability mass of the posterior distribution:

$$P(\Theta \in I_\alpha | \mathbf{Y} = \mathbf{y}) = 1 - \alpha, \tag{3.1}$$

is called a **credible interval**¹. Usually we talk about a $(1 - \alpha) \cdot 100\%$ credible interval; for example, if the confidence level is $\alpha = 0.05$, we talk about the 95% credible interval.

¹Remember that we assumed the parameter having a continuous distribution. This means that we can always choose an interval I_α for which the condition (3.1) holds; we can choose the interval for which the probability is *exactly* $1 - \alpha$, so we do not have to define the credible interval of having the probability of *at least* $1 - \alpha$.

For the vector-valued $\Theta \in \Omega \subseteq \mathbb{R}^d$, a (contiguous) region $I_\alpha \subseteq \Omega$ containing a proportion $1 - \alpha$ of the probability mass of the posterior distribution:

$$P(\Theta \in I_\alpha | \mathbf{Y} = \mathbf{y}) = 1 - \alpha,$$

is called a **credible region**.

On the definition we conditioned on the observed data, but we can also talk about a credible interval before observing any data. In this case a credible interval means an interval I_α containing a proportion $1 - \alpha$ of the probability mass of the prior distribution:

$$P(\Theta \in I_\alpha) = 1 - \alpha.$$

This may actually be useful if we want to calibrate an informative prior distribution. We may for example have an ad hoc estimate of the region of the parameter space where the true parameter value lies with 95% certainty. Then we just have to find a prior distribution whose 95% credible interval agrees with this estimate. But usually credible intervals are examined after observing the data.

The condition (3.1) does not determine a unique $(1 - \alpha) \cdot 100\%$ credible interval: actually there is an infinite number of such intervals. This means that we have to define some additional condition for choosing the credible interval. Let's examine two of the most common extra conditions.

3.1.2 Equal-tailed interval

An **equal-tailed interval** (also called a **central interval**) of confidence level α is an interval

$$I_\alpha = [q_{\alpha/2}, q_{1-\alpha/2}],$$

where q_z is a z -quantile (remember that we assumed the parameter to be have a continous distribution; this means that the quantiles are always defined) of the posterior distribution.

For instance, 95% equal-tailed interval is an interval

$$I_{0.05} = [q_{0.025}, q_{0.975}],$$

where $q_{0.025}$ and $q_{0.975}$ are the quantiles of the posterior distribution. This is an interval on whose both right and left side lies 2.5% of the probability mass of the posterior distribution; hence the name *equal-tailed* interval.

If we can solve the posterior distribution in a closed form, quantiles can be obtained via the quantile function of the posterior distribution:

$$\begin{aligned} P(\Theta \leq q_z | Y = y) &= z \\ F_{\Theta|\mathbf{Y}}(q_z | y) &= z \\ q_z &= F_{\Theta|\mathbf{Y}}^{-1}(z | y), \end{aligned}$$

This quantile function $F_{\Theta|\mathbf{Y}}^{-1}$ is an inverse of the cumulative density function (cdf) $F_{\Theta|\mathbf{Y}}$ of the posterior distribution.

Usually, when a credible interval is mentioned without specifying which type of the credible interval it is, an equal-tailed interval is meant.

However, unless the posterior distribution is unimodal and symmetric, there are point outsed of the equal-tailed credible interval having a higher posterior density than some points of the interval. If we want to choose the credible interval so that this not happen, we can do it by using the highest posterior density criterion for choosing it. We will examine this criterion more closely after an example of equal-tailed credible intervals.

3.1.3 Example of credible intervals

Let's revisit Example 2.1.1: we have observed a data set $y = (4, 3, 11, 3, 6)$, and model it as a Poisson-distributed random vector $\mathbf{Y}$ using a gamma prior with hyperparameters $\alpha = 1$, $\beta = 1$ for the parameter λ . Now we want to compute 95% confidence interval for the parameter λ .

Let's first set up our data, hyperparameters and a confidence level:

```
y <- c(4, 3, 11, 3, 6)
n <- length(y)
alpha <- 1
beta <- 1

alpha_conf <- 0.05
```

A posterior distribution for the parameter λ is $\text{Gamma}(n\bar{y} + \alpha, n + \beta)$. Let's set up also the parameters of the posterior distribution:

```
alpha_1 <- sum(y) + alpha
beta_1 <- n + beta
```

Now we can compute 0.025- and 0.975-quantiles using the quantile function $F_{\Lambda|\mathbf{Y}}^{-1}$ of the posterior distribution:

$$q_{0.025} = F_{\Lambda|\mathbf{Y}}^{-1}(0.025|y)$$

$$q_{0.975} = F_{\Lambda|\mathbf{Y}}^{-1}(0.975|y).$$

Luckily R contains a quantile function of the gamma distribution, so we get the 95% credible interval simply as:

```
q_lower <- qgamma(alpha_conf / 2, alpha_1, beta_1)
q_upper <- qgamma(1 - alpha_conf / 2, alpha_1, beta_1)
c(q_lower, q_upper)
```

```
## [1] 3.100966 6.547264
```

Let's examine this credible interval visually:

```
lambda <- seq(0,7, by = 0.001) # set up grid for plotting
lambda_true <- 3

plot(lambda, dgamma(lambda, alpha_1, beta_1), type = 'l', lwd = 2, col = 'violet',
     ylim = c(0, 1.5), xlab = expression(lambda),
     ylab = expression(paste('p(', lambda, '|y)'))))

y_val <- dgamma(lambda, alpha_1, beta_1)
x_coord <- c(q_lower, lambda[lambda >= q_lower & lambda <= q_upper], q_upper)
y_coord <- c(0, y_val[lambda >= q_lower & lambda <= q_upper], 0)
polygon(x_coord, y_coord, col = 'pink', lwd = 2, border = 'violet')
abline(v = lambda_true, lty = 2)

lines(lambda, dgamma(lambda, alpha, beta),
     type = 'l', lwd = 2, col = 'orange')
legend('topright', inset = .02, legend = c('prior', 'posterior'),
     col = c('orange', 'violet'), lwd = 2)
```

Even though the 95 % credible interval is quite wide because of the low sample size, this time it actually does not contain the true parameter value $\lambda = 3$ (which we know, because we generated the data from Poisson(3)-distribution!). But let's see what happens when we increase the sample size:

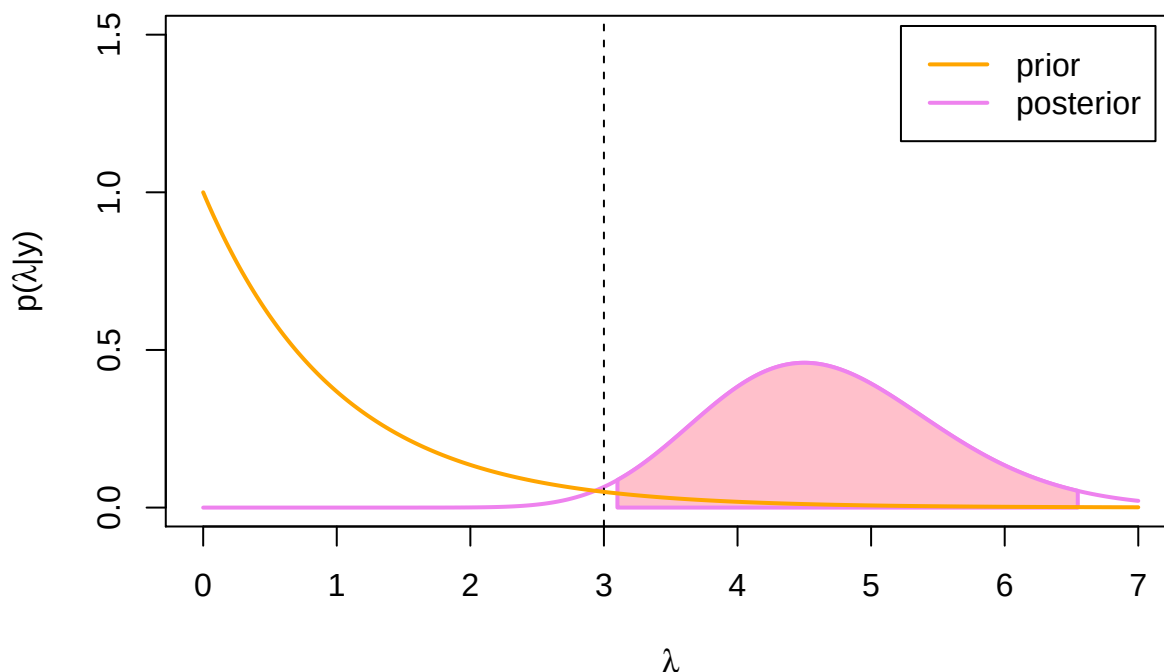


Figure 3.1: 95% equal-tailed CI for Poisson-gamma model

```

n_total <- 200
set.seed(111111) # use same seed, so first 5 obs. stay same
y_vec <- rpois(n_total, lambda_true)
head(y_vec)

## [1] 4 3 11 3 6 3

n_vec <- c(1, 2, 5, 10, 50, 100, 200)
par(mfrow = c(4,2), mar = c(2, 2, .1, .1))

plot_CI <- function(alpha, beta, y_vec, n_vec, alpha_conf, lambda_true) {
  lambda <- seq(0,7, by = 0.01) # set up grid for plotting
  plot(lambda, dgamma(lambda, alpha, beta), type = 'l', lwd = 2, col = 'orange',
        ylim = c(0, 3.2), xlab = '', ylab = '')
  q_lower <- qgamma(alpha_conf / 2, alpha, beta)
  q_upper <- qgamma(1 - alpha_conf / 2, alpha, beta)
  y_val <- dgamma(lambda, alpha, beta)
  polygon(c(q_lower, lambda[lambda >= q_lower & lambda <= q_upper], q_upper),
          c(0, y_val[lambda >= q_lower & lambda <= q_upper], 0),
          col = 'goldenrod1', lwd = 2, border = 'orange')

  abline(v = lambda_true, lty = 2)
  text(x = 0.5, y = 2.5, 'prior', cex = 1.75)

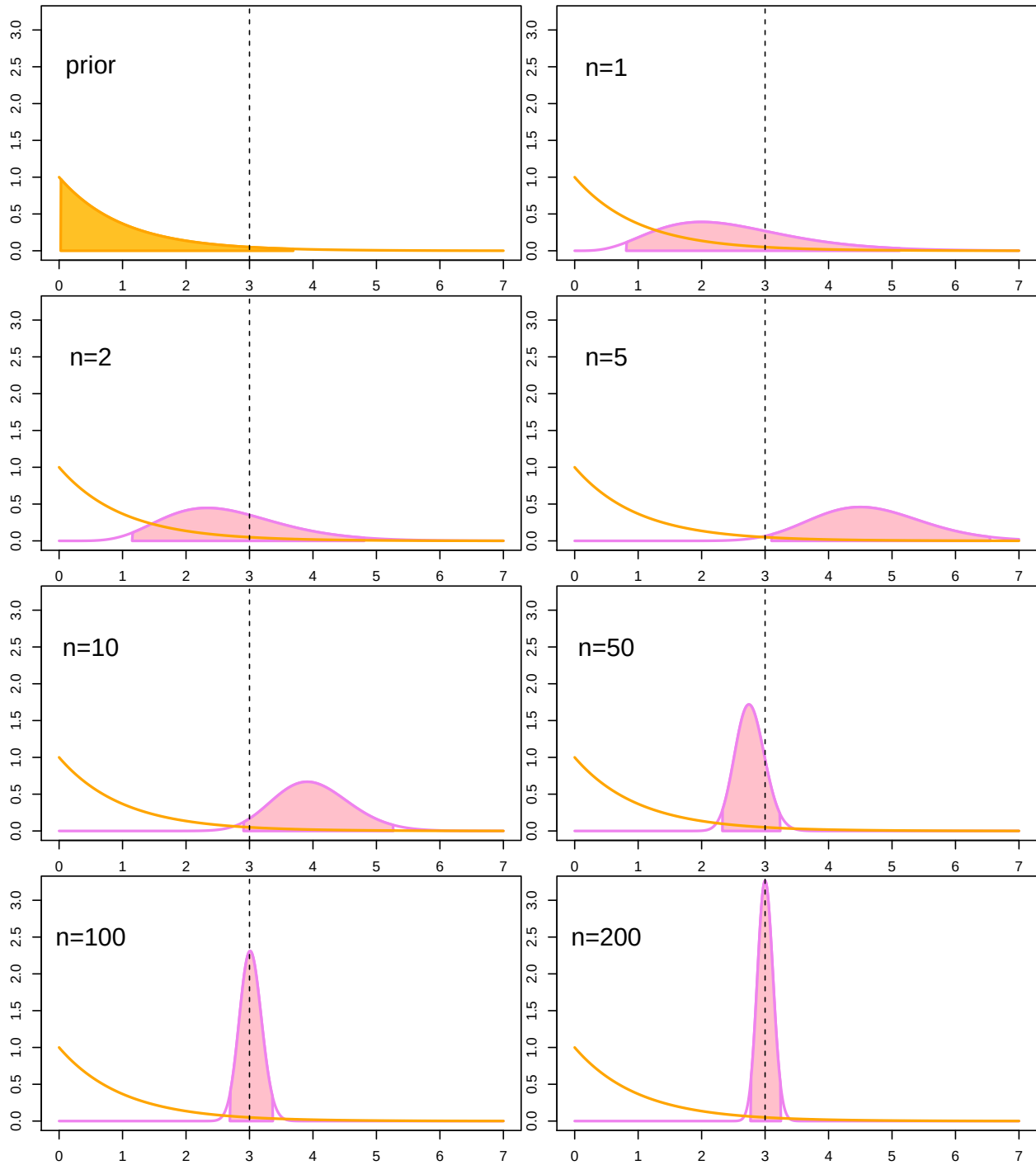
  for(n_crnt in n_vec) {
    y_sum <- sum(y_vec[1:n_crnt])
    alpha_1 <- alpha + y_sum
    beta_1 <- beta + n_crnt

    plot(lambda, dgamma(lambda, alpha_1, beta_1), type = 'l', lwd = 2, col = 'violet',

```

```
    ylim = c(0, 3.2), xlab = '', ylab = '')
    q_lower <- qgamma(alpha_conf / 2, alpha_1, beta_1)
    q_upper <- qgamma(1 - alpha_conf / 2, alpha_1, beta_1)
    y_val <- dgamma(lambda, alpha_1, beta_1)
    x_coord <- c(q_lower, lambda[lambda >= q_lower & lambda <= q_upper], q_upper)
    y_coord <- c(0, y_val[lambda >= q_lower & lambda <= q_upper], 0)
    polygon(x_coord, y_coord, col = 'pink', lwd = 2, border = 'violet')
    lines(lambda, dgamma(lambda, alpha, beta),
          type = 'l', lwd = 2, col = 'orange')
    abline(v = lambda_true, lty = 2)
    text(x = 0.5, y = 2.5, paste0('n=', n_crnt), cex = 1.75)
  }
}

plot_CI(alpha, beta, y_vec, n_vec, alpha_conf, lambda_true)
```



When we observe more data, the credible interval get narrower. This reflects our growing certainty about the range where the true parameter value lies. Turns out that this time the credible interval contains the true parameter value with all the other tested sample sizes expect $n = 5$.

But unlike the frequentist confidence interval, the credible interval does not depend only on the data: the prior distribution also influences the credible intervals. That orange area in the first of the figures is a credible interval that is computed using the prior distribution. It describes our belief where 95% of the probability mass of the distribution *should* lie before we observe any data.

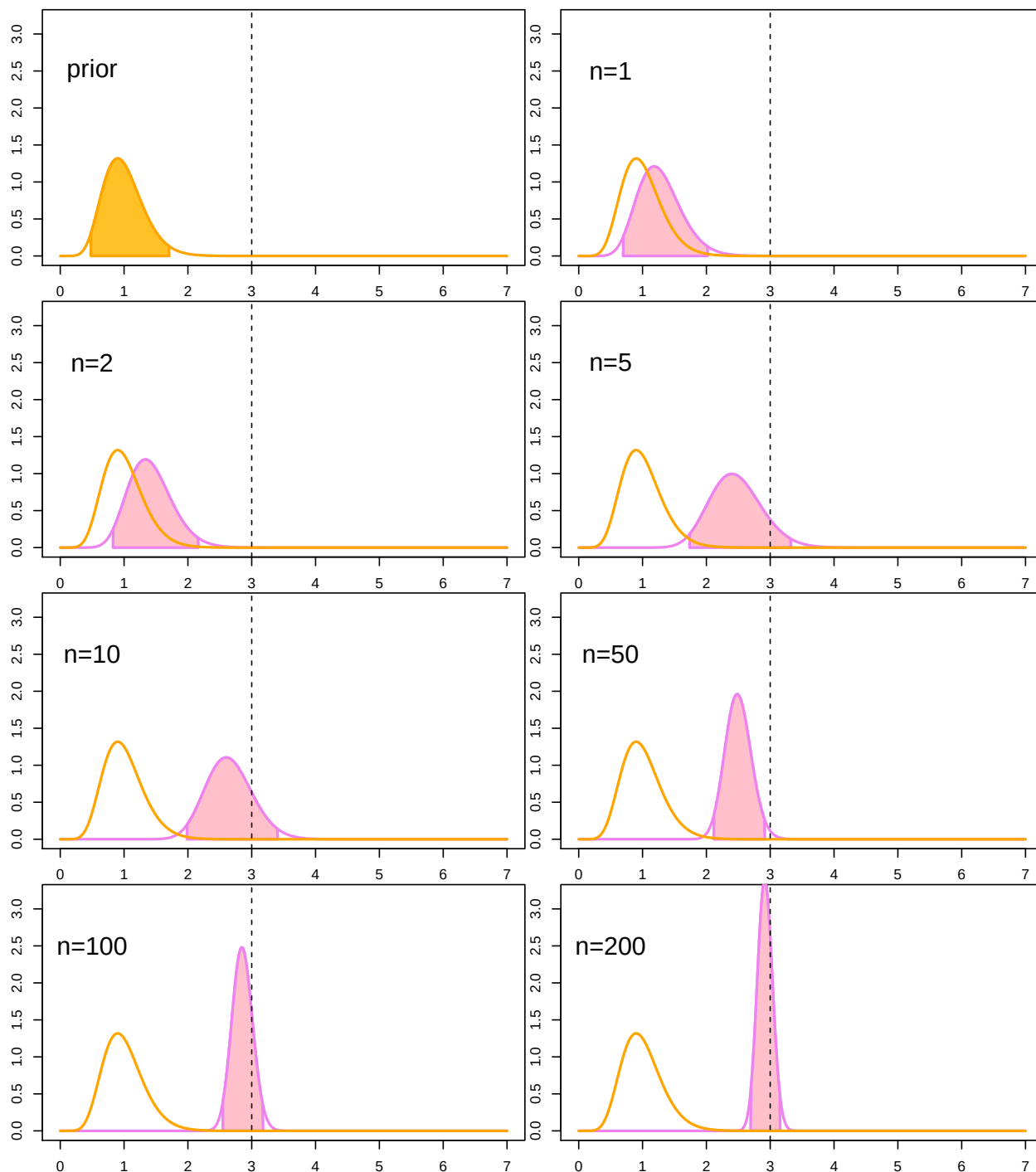
When we get more observations, credible intervals are influenced more by the the data, and less by the prior distribution. This can be more clearly seen if we use a more strongly peaked prior $\text{Gamma}(10, 10)$. The

expected value of the gamma distributed random variable X is

$$EX = \frac{\alpha}{\beta},$$

so this prior has a same expected value $E\lambda = 1$ than the prior $\text{Gamma}(1,1)$. But its probability mass is concentrated on much smaller area compared to the relatively flat $\text{Gamma}(1,1)$ -prior, so it has a much stronger effect on the posterior inferences:

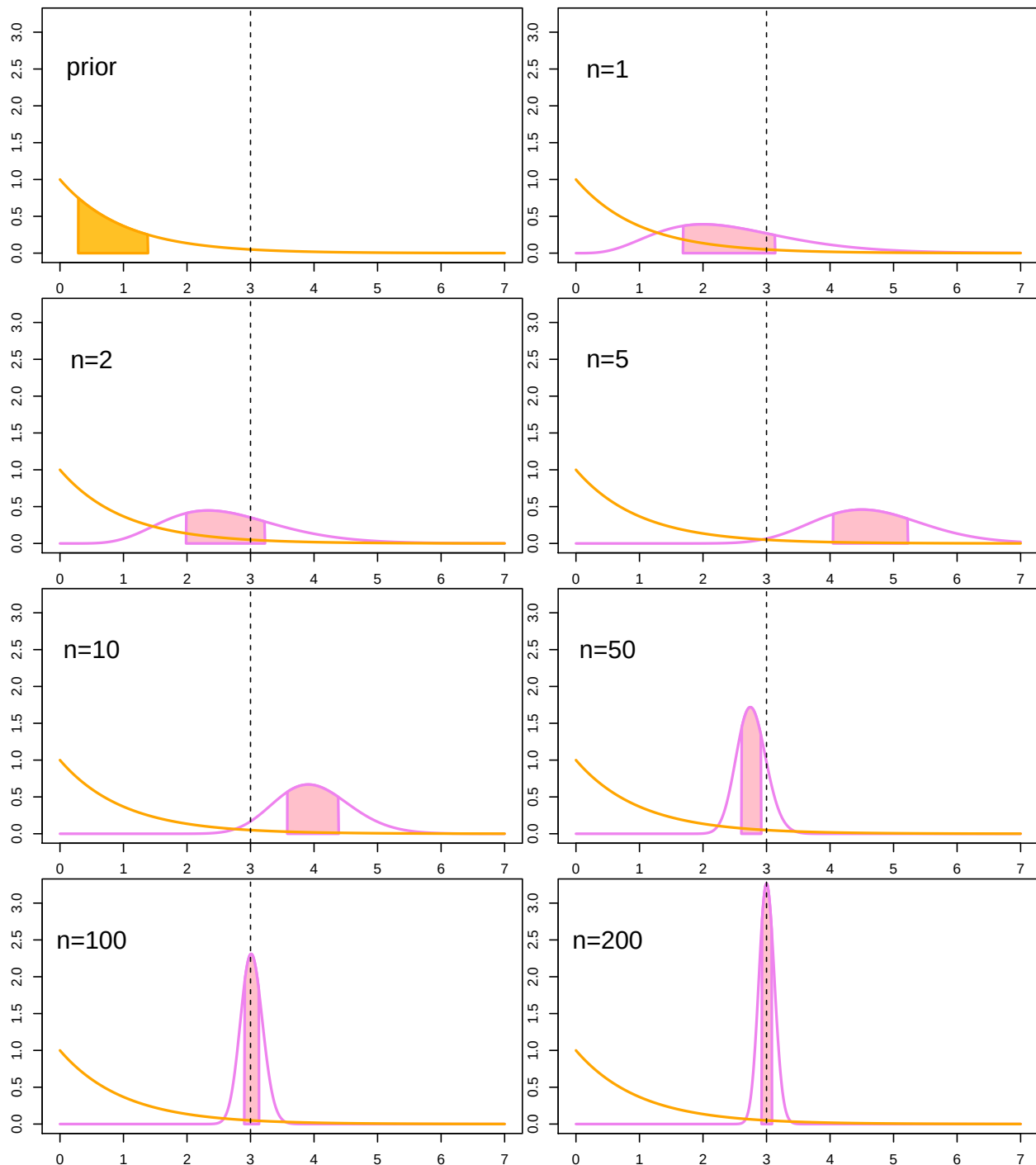
```
par(mfrow = c(4,2), mar = c(2, 2, .1, .1))
plot_CI(alpha = 10, beta = 10, y_vec, n_vec, alpha_conf, lambda_true)
```



With small sample size the posterior distribution, and thus also the credible intervals, are almost fully determined by the prior; only with the higher sample sizes the data starts to override the effect of the prior distribution on the posterior.

Of course the credible intervals do not have to always be 95% credible intervals. Another widely used credible interval is a 50% credible interval, which contains half of the probability mass of the posterior distribution:

```
par(mfrow = c(4,2), mar = c(2, 2, .1, .1))
plot_CI(alpha, beta, y_vec, n_vec, alpha_conf = 0.5, lambda_true)
```



3.1.4 Highest posterior density region

A **highest posterior density (HPD)** region of confidence level α is a $(1 - \alpha)$ -confidence region I_α for which holds that the posterior density for every point in this set is higher than the posterior density for any point outside of this set:

$$f_{\Theta|Y}(\theta|y) \geq f_{\Theta|Y}(\theta'|y)$$

for all $\theta \in I_\alpha$, $\theta' \notin I_\alpha$. This means that a $(1 - \alpha)$ -highest density posterior region is a smallest possible $(1 - \alpha)$ -credible region.

An observant reader may notice that the HPD region is not necessarily an interval (or a contiguous region in a higher-dimensional case): if the posterior distribution is multimodal, the HPD region of this distribution may be an union of distinct intervals (or distinct contiguous regions in a higher-dimensional case). This means that HPD regions are not necessarily always strictly credible intervals or regions according to Definition (3.1). However, in Bayesian statistics we often talk simply about HPD intervals, even though may not always be intervals.

Let's examine a (hypothetical) bimodal posterior density (a mixture of two beta distributions) for which the HPD region is not an interval. An equal-tailed 95% CI is always an interval, even though in this case density values are very low near the saddle point of the density function:

```
alpha_conf <- .05
alpha_1 <- 11
beta_1 <- 30
alpha_2 <- 25
beta_2 <- 8

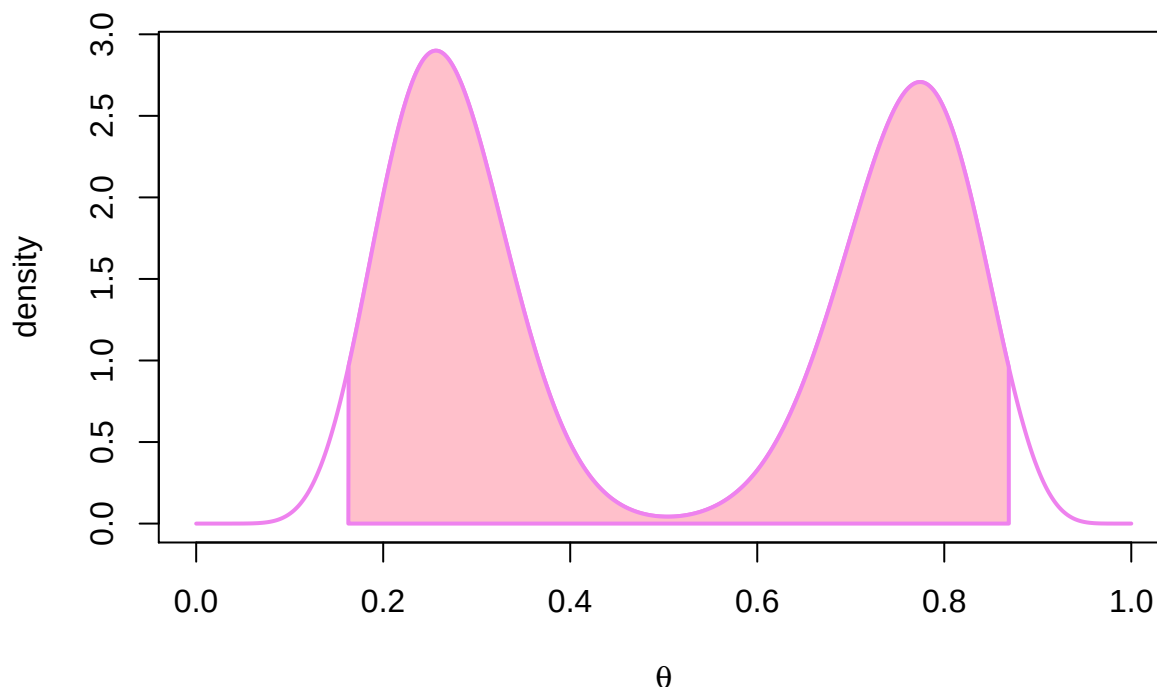
mixture_density <- function(x, alpha_1, alpha_2, beta_1, beta_2) {
  .5 * dbeta(x, alpha_1, beta_1) + .5 * dbeta(x, alpha_2, beta_2)
}

# generate data to compute empirical quantiles
n_sim <- 1000000
theta_1 <- rbeta(n_sim / 2, alpha_1, beta_1)
theta_2 <- rbeta(n_sim / 2, alpha_2, beta_2)
theta <- sort(c(theta_1, theta_2))

lower_idx <- round((alpha_conf / 2) * n_sim)
upper_idx <- round((1 - alpha_conf / 2) * n_sim)
q_lower <- theta[lower_idx]
q_upper <- theta[upper_idx]

x <- seq(0,1, by = 0.001)
y_val <- mixture_density(x, alpha_1, alpha_2, beta_1, beta_2)
x_coord <- c(q_lower, x[x >= q_lower & x <= q_upper], q_upper)
y_coord <- c(0, y_val[x >= q_lower & x <= q_upper], 0)

plot(x, mixture_density(x, alpha_1, alpha_2, beta_1, beta_2),
     type='l', col = 'violet', lwd = 2,
     xlab = expression(theta), ylab = 'density')
polygon(x_coord, y_coord, col = 'pink', lwd = 2, border = 'violet')
```



On the other hand a 95% HPD region for this bimodal distribution consists of two distinct intervals:

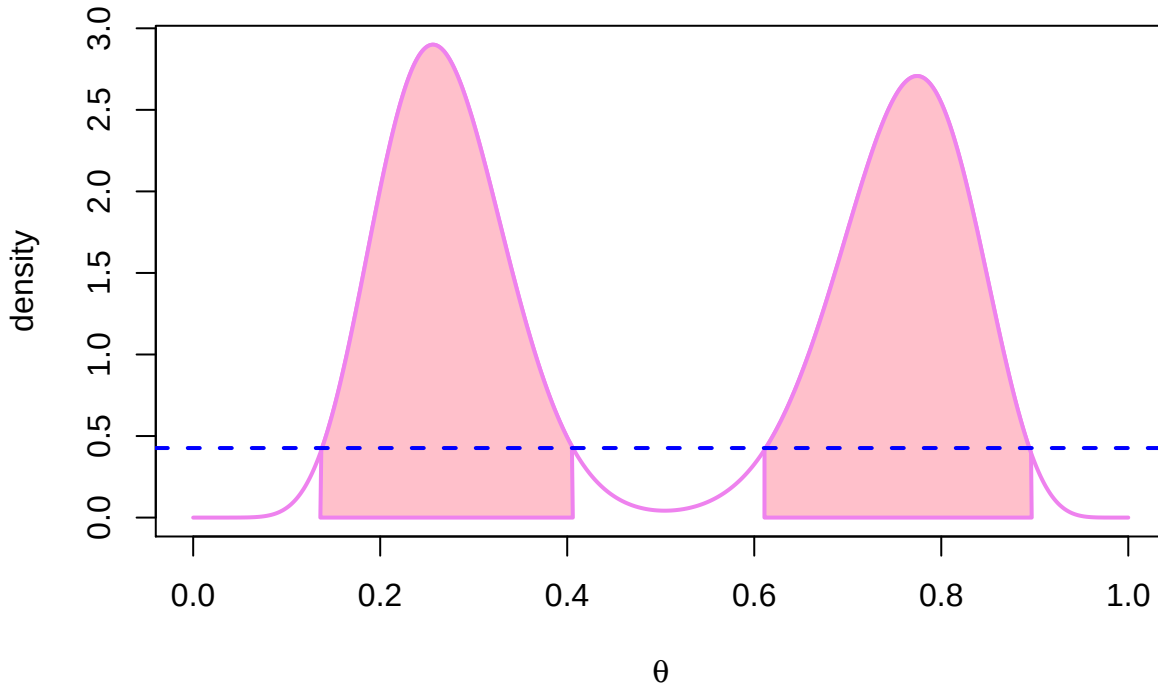
```
# install.packages('HDInterval')
dens <- density(theta)
HPD_region <- HDInterval::hdi(dens, allowSplit = TRUE)
height <- attr(HPD_region, 'height')
lower <- HPD_region[1,1]
upper <- HPD_region[1,2]

x_coord <- c(lower, x[x >= lower & x <= upper], upper)
y_coord <- c(0, y_val[x >= lower & x <= upper], 0)

plot(x, mixture_density(x, alpha_1, alpha_2, beta_1, beta_2),
     type='l', col = 'violet', lwd = 2,
     xlab = expression(theta), ylab = 'density')
polygon(x_coord, y_coord, col = 'pink', lwd = 2, border = 'violet')

lower <- HPD_region[2,1]
upper <- HPD_region[2,2]
x_coord <- c(lower, x[x >= lower & x <= upper], upper)
y_coord <- c(0, y_val[x >= lower & x <= upper], 0)
polygon(x_coord, y_coord, col = 'pink', lwd = 2, border = 'violet')

abline(h = height, col = 'blue', lty = 2, lwd = 2)
```



In this case it seems that a highest posterior density region is a better summary of the distribution than the equal-tailed confidence interval. This (imagined) example also demonstrates why it is dangerous to try to reduce the posterior distribution to single summary statistics, such as the mean or the mode of the posterior distribution.

3.2 Posterior mean as a convex combination of means

A mean of the posterior distribution is often also called a Bayes estimator, denoted as

$$\hat{\theta}_{\text{Bayes}}(Y) := E[\lambda | \mathbf{Y}].$$

A mean of the gamma distribution $\text{Gamma}(\alpha, \beta)$ is $\frac{\alpha}{\beta}$, so a posterior mean for the model Poisson-gamma model of Example 2.1.1 is

$$E[\lambda | \mathbf{Y} = \mathbf{y}] = \frac{\alpha + n\bar{y}}{\beta + n}. \quad (3.2)$$

A posterior mean can also be written as a convex combination of the mean of the prior distribution, and the mean of the observations:

$$E[\lambda | \mathbf{Y} = \mathbf{y}] = \frac{\alpha + n\bar{y}}{\beta + n} = \kappa \frac{\alpha}{\beta} + (1 - \kappa)\bar{y},$$

where the mixing proportion is

$$\kappa = \frac{\beta}{\beta + n}.$$

The higher the sample size, the higher is the contribution of the data to the posterior mean (compared to the contribution of the prior mean). And at the limit when $n \rightarrow \infty$, $\kappa \rightarrow 0$. This means that for this model the posterior mean is asymptotically equivalent to the maximum likelihood estimator, which for this model is just the mean of the observations:

$$\hat{\theta}_{\text{MLE}}(\mathbf{Y}) = \bar{Y}.$$

The formula for the posterior mean of the Poisson-gamma model given in Equation (3.2) also gives us a hint why increasing the rate parameter β of the prior gamma distribution increased the effect of the prior of the

posterior distribution: The location parameter α is added to the sum of the observations, and β is added to the sample size. So the prior could be interpreted as “pseudo-observations” that are added to the actual observations: parameter α could be interpreted as the “pseudo-events”, and β as the “pseudo-sample size” (although they are not necessarily integers). So using prior $\alpha = 15$, $\beta = 10$ could be interpreted as having a prior data set of 10 observations, and having total 15 events in this data set.

Chapter 4

Approximate inference

In the preceding chapters we have examined conjugate models for which it is possible to solve the marginal likelihood, and thus also the posterior and the posterior predictive distributions in a closed form. However, in more realistic scenarios in which more complex models are required, the marginal likelihood is usually intractable, and because of this the posterior cannot be solved analytically.

This means that usually we have to approximate the posterior distribution $p(\boldsymbol{\theta}|\mathbf{y})$ somehow, and then use this approximation to compute the quantities of interest, such as posterior mean or credible intervals.

In general, there are two ways to approximate the posterior distribution:

1. Simulation: generate a random sample from the posterior distribution, and use its empirical distribution function as an approximation of the posterior.
2. Distributional approximation: approximate the posterior directly by some simpler parametric distribution, such as the normal distribution.

A simple form of the distributional approximation is a normal approximation, where the central limit theorem is invoked to justify the use of normal distribution to approximate the posterior distribution. This is analogous to the normal approximation used in frequentist statistics to approximate the distribution of the estimator of the parameter of interest with high sample sizes. More generally, approximating the posterior density by some tractable density $q(\boldsymbol{\theta})$ is called **variational inference**.

However, on the rest of this chapter we will focus to the approximating the posterior distribution by generating a random sample from it.

4.1 Simulation methods

The first step is to generate a random sample $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_S$ from a posterior distribution $p(\boldsymbol{\theta}|\mathbf{y})$. If the posterior distribution is a known distribution, whose simulation method has been implemented in R or Python, then this is of course easy. Of course, in this case you do not need the sample the posterior to distribution to approximate it, because you already know the *exact* posterior distribution. However, the simulating the may still be the easiest way to evaluate some integrals over the posterior distributions, such as the probability of some set. We will return to this later in this section.

But let's consider the more interesting case, where the posterior distribution cannot be solved in a closed form. Now you may be wondering how on earth is it possible to generate sample from the unknown distribution? Turns out that this is actually super easy: even though the normalizing constant $p(\mathbf{y})$ is unknown, we can utilize the same trick that we used to compute the posterior analytically for the conjugate models. Instead of the posterior density, it is sufficient to generate a random sample from an unnormalized posterior density, that is, any function $\boldsymbol{\theta} \rightarrow q(\boldsymbol{\theta}; \mathbf{y})$, which is proportional to the posterior density:

$$p(\boldsymbol{\theta}|\mathbf{y}) \propto q(\boldsymbol{\theta}; \mathbf{y}).$$

In particular, we can utilize the unnormalized version of the Bayes' theorem:

$$p(\boldsymbol{\theta}|\mathbf{y}) \propto p(\boldsymbol{\theta})p(\mathbf{y}|\boldsymbol{\theta}),$$

and simulate the posterior by generating a random sample from the unnormalized posterior distribution $q(\boldsymbol{\theta}; \mathbf{y}) \propto p(\boldsymbol{\theta})p(\mathbf{y}|\boldsymbol{\theta})$.

Now the only problem is how to generate this random sample? This can be done for example by **rejection sampling** or **importance sampling** for the simple models. On this course we will not concentrate on these sampling methods. For those more interested on the sampling methods, there is a course called **Computational statistics**, which is dedicated solely on the computational aspects of Bayesian inference. It will be possible to do the course as self-study next spring, and it will be lectured with a high probability next autumn.

Fortunately, there are nowadays automated **probabilistic programming** tools that do these simulations automatically for us, so that we do not have to write a sampler manually each time we want to simulate from a new posterior distribution. So our plan is to demonstrate simulation from the posterior distribution manually with a simple example, and after this to introduce these automated tools that make a life of the statistician easier.

4.1.1 Grid approximation

For our example we will use a straightforward simulation recipe called **grid approximation** or **direct discrete approximation**:

1. Create an even-spaced grid $g_1 = a + i/2, \dots, g_m = b - i/2$, where a is the lower, and b is the upper limit of the interval on which we want to evaluate the posterior, i is the increment of the grid, and m is the number of grid points.
2. Evaluate values of the unnormalized posterior density in the grid points $q(g_1; \mathbf{y}), \dots, q(g_m; \mathbf{y})$, and normalized them to obtain the estimated values of the posterior distribution at the grid points:

$$\hat{p}_1 := \frac{q(g_1; \mathbf{y})}{\sum_{i=1}^m q(g_i; \mathbf{y})}, \dots, \hat{p}_m := \frac{q(g_m; \mathbf{y})}{\sum_{i=1}^m q(g_i; \mathbf{y})}$$

3. For every $s = 1, \dots, S$:
 - Generate λ_s from a categorical distribution with outcomes $g_1, \dots, g_m$ which have the probabilities $\hat{p}_1, \dots, \hat{p}_m$
 - Add jitter which is uniformly distributed around zero, and whose interval length is equal to the grid spacing, to the generated values: $\lambda_s = \lambda_s + X$, where $X \sim U(-i/2, i/2)$ (to push generated values out of the grid points).

You may have observed that this basically amounts to performing a numerical integration by sampling. Grid approximation also has the downsides of numerical integration: we can only simulate from the finite interval, and if we keep the grid spacing constant, the size of the grid grows exponentially w.r.t. dimension of the parameter. However, this crude method will do for our introductory example.

4.1.2 Example: grid approximation

Let's demonstrate a simulation from the posterior distribution with the Poisson-gamma conjugate model of Example 2.1.1. Of course we know that the true posterior distribution for this model is

$$\text{Gamma}(\alpha + n\bar{y}, \beta + n),$$

and thus we wouldn't have to simulate at all to find out the posterior of this model. However, the point of doing simulation first with a known distribution is to verify that our simulation method works by confirming that the simulated posterior density is very close to the analytically solved posterior density.

Let's start by setting the same parameter values and generating the same observations used in Example 2.1.1:

```
lambda_true <- 3
alpha <- beta <- 1
n <- 5
set.seed(111111)
y <- rpois(n, lambda_true)
y
```

```
## [1] 4 3 11 3 6
```

The unnormalized posterior for this model can be written (cf. Equation (2.1)) as:

$$q(\lambda; \mathbf{y}) = \lambda^{\sum_{i=1}^n y_i + \alpha - 1} e^{-(n + \beta)\lambda}$$

Let's define this as a function:

```
q <- function(lambda, y, n, alpha, beta) {
  lambda^(alpha + sum(y) - 1) * exp(-(n + beta) * lambda)
}
```

The parameter space $\Omega = (0, \infty)$ is a whole positive real axis. But this crude simulation method we use has a limitation that an interval on which we simulate the posterior distribution must be finite. How do we then choose this interval? In a real scenario, we would compute some initial point estimates such as maximum likelihood estimates for the mean and the variance of the parameter, and then use these to choose an interval which should contain almost all of the probability mass of the posterior distribution. However, in this introductory example we have already seen the true posterior, so we can be sure that for example the interval $(0, 20)$ contains almost all of the probability mass of the distribution. So let's use set a grid on the interval $(0, 20)$ by an increment $i = 0.01$, evaluate the unnormalized density at the points of this grid, and normalize the values by dividing them by the sum of all values:

```
lower_lim <- 0
upper_lim <- 20
i <- 0.01
grid <- seq(lower_lim + i/2, upper_lim - i/2, by = i)

n_sim <- 1e4
n_grid <- length(grid)
grid_values <- q(grid, y, n, alpha, beta)
normalized_values <- grid_values / sum(grid_values)
```

Now the probabilities $\hat{p}_1, \dots, \hat{p}_m$ sum to one, and thus define a proper categorical probability distribution (with grid points $g_1, \dots, g_m$ being the values into which these probabilities correspond to). Let's generate the sample $\lambda_1, \dots, \lambda_S$ from this distribution, and then add some uniform jitter to them:

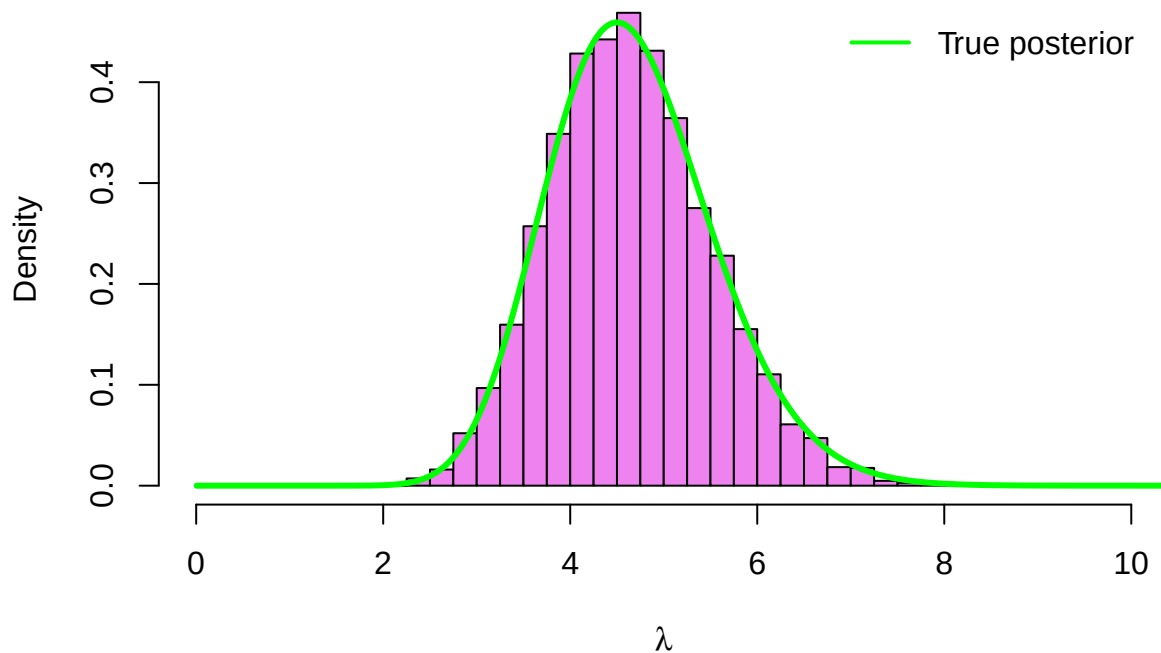
```
idx_sim <- sample(1:n_grid, n_sim, prob = normalized_values, replace = TRUE)
lambda_sim <- grid[idx_sim]

X <- runif(n_sim, -i/2, i/2)
lambda_sim <- lambda_sim + X
```

Now we should have simulated a sample from the posterior distribution. Let's draw a histogram of our sample, and overlay it with the analytically solved posterior distribution to see if they match:

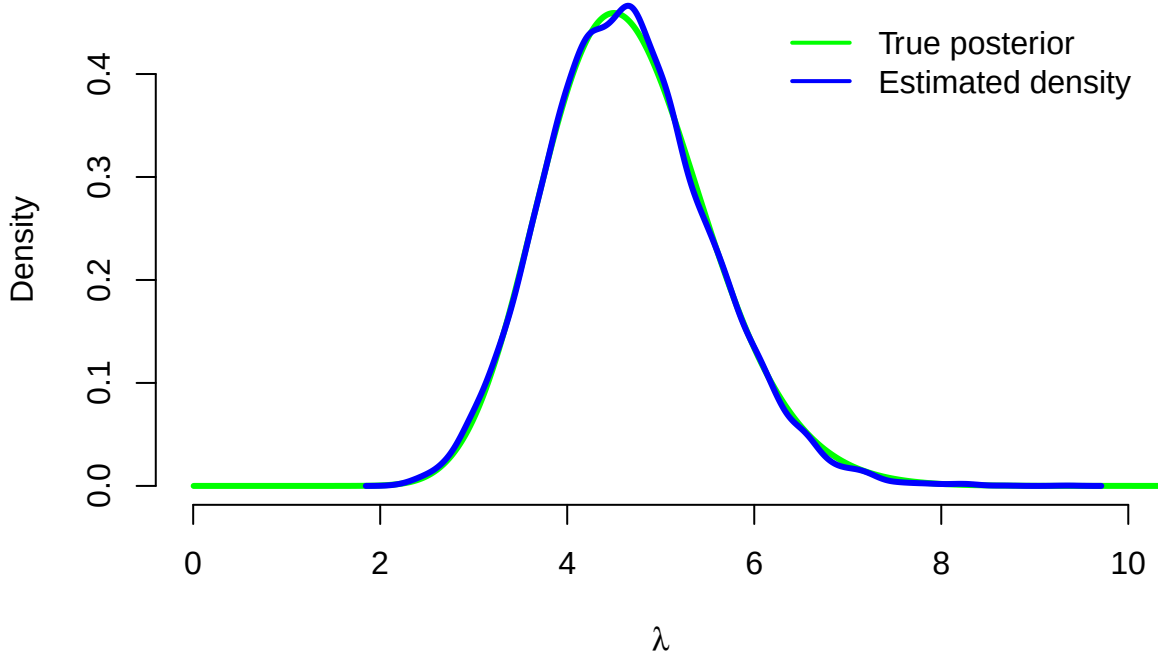
```
hist(lambda_sim, col = 'violet', breaks = seq(0,10, by = .25), probability = TRUE,
     main = '', xlab = expression(lambda), xlim = c(0,10))
lines(grid, dgamma(grid, alpha + sum(y), beta + n), type = 'l', col = 'green', lwd=3)
```

```
legend('topright', legend = 'True posterior', bty = 'n',
      col = 'green', lwd = 2, inset = .02)
```



Our simulation seems to have worked correctly! Instead of the histogram we can also compute a smoothed density estimation (with some R magic in the form of `density()`-function) based on our sample, and verify that it is very close to the true posterior density:

```
density_sim <- density(lambda_sim)
plot(grid, dgamma(grid, alpha + sum(y), beta + n), type = 'l', col = 'green',
      lwd=3, xlim = c(0,10), bty = 'n', xlab = expression(lambda), ylab = 'Density')
lines(density_sim, type = 'l', col = 'blue', lwd=3 )
legend('topright', legend = c('True posterior', 'Estimated density'),
      col = c('green', 'blue'), lwd = 2, inset = .02, bty = 'n')
```



Of course this was not a super interesting example because we already knew a posterior density which we had solved analytically. But now that we are simulating anyway, we don't actually have to limit our choice of the prior distribution to conjugate priors. So now when we have verified that our simulation algorithm works, let's try a different prior.

4.1.3 Example : non-conjugate prior for Poisson model

Another popular prior for the Poisson likelihood is a log-normal distribution. If a random variable X follows a normal distribution $N(\mu, \sigma^2)$, then $Y = e^X$ has a log-normal distribution $\text{Log-normal}(\mu, \sigma^2)$. And correspondingly, if $Y \sim \text{Log-normal}(\mu, \sigma^2)$ and $X = \log Y$, then $Y \sim N(\mu, \sigma^2)$; hence the name of the distribution. Parameters μ and σ^2 are not the location and scale parameter of the log-normal distribution, but the location and the scale parameter of the normal distribution you get, when you take a logarithm of the log-normally distributed random variable.

Using a log-normal prior, our model is now:

$$\begin{aligned} Y_i &\sim \text{Poisson}(\lambda) \quad \text{for all } i = 1, \dots, n \\ \lambda &\sim \text{Log-normal}(\mu, \sigma^2). \end{aligned}$$

A density function of the log-normal distribution is

$$p(\lambda) = \frac{1}{\lambda \sqrt{2\pi\sigma^2}} e^{-\frac{(\log \lambda - \mu)^2}{2\sigma^2}},$$

and thus we can write the unnormalized posterior density as

$$\begin{aligned} p(\lambda|\mathbf{y}) &\propto p(\lambda)p(\mathbf{y}|\lambda) \\ &\propto \lambda^{-1} e^{-\frac{(\log \lambda - \mu)^2}{2\sigma^2}} \lambda^{\sum_{i=1}^n y_i} e^{-n\lambda} \\ &\propto \lambda^{\sum_{i=1}^n y_i - 1} e^{-n\lambda - \frac{(\log \lambda - \mu)^2}{2\sigma^2}}. \end{aligned}$$

This cannot be normalized into any known probability distribution: the normalizing constant

$$p(\mathbf{y}) = \int p(\lambda)p(\mathbf{y}|\lambda) d\lambda$$

is intractable! But this is not a problem, because we know how to simulate from an unnormalized posterior distribution. Let's first define a function¹ for the unnormalized posterior:

```
q <- function(lambda, y, n, mu, sigma_squared) {
  lambda^(sum(y) - 1) * exp(-n * lambda - (log(lambda) - mu)^2 / (2 * sigma_squared))
}
```

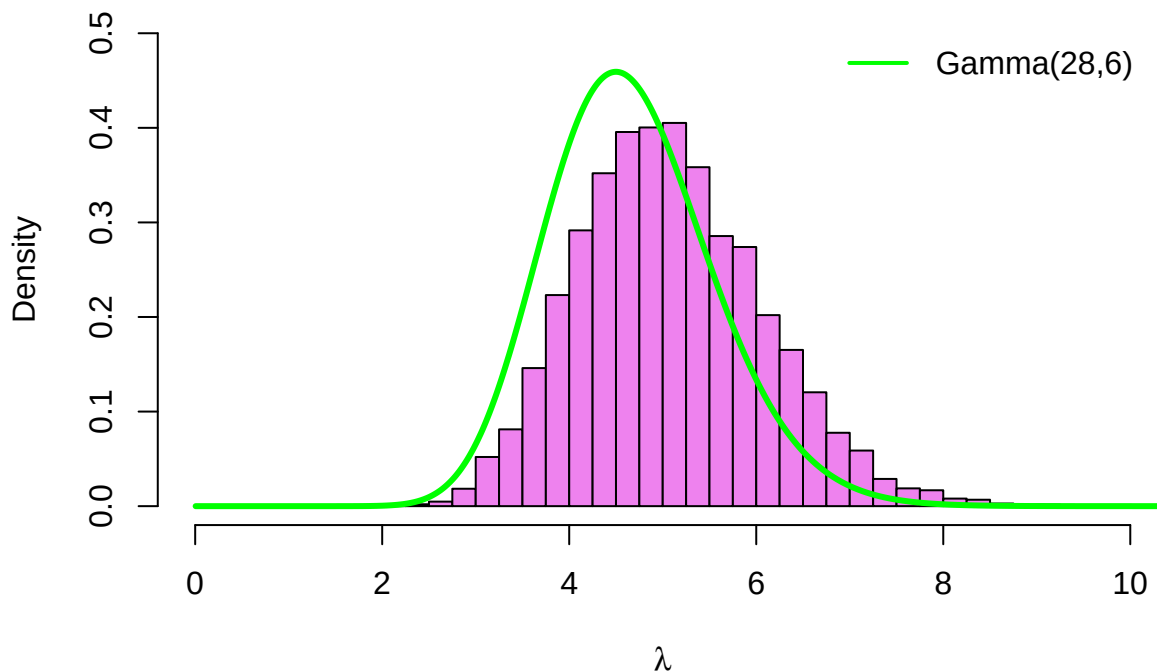
Let's also set parameters $\mu = 0$, $\sigma^2 = 1$ of the prior:

```
mu <- 0
sigma_squared <- 1
```

Now we are ready to use our simulation recipe again, and visualize the results:

```
grid_values <- q(grid, y, n, mu, sigma_squared)
normalized_values <- grid_values / sum(grid_values)
idx_sim <- sample(1:n_grid, n_sim, prob = normalized_values, replace = TRUE)
lambda_sim2 <- grid[idx_sim] + runif(n_sim, -i/2, i/2)

hist(lambda_sim2, col = 'violet', breaks = seq(0,10, by = .25), probability = TRUE,
     main = '', xlab = expression(lambda), xlim = c(0,10), ylim = c(0, 0.5))
lines(grid, dgamma(grid, alpha + sum(y), beta + n), type='l', col='green', lwd=3)
legend('topright', legend = paste0('Gamma(', sum(y) + alpha, ',', n + beta, ')'),
     col = 'green', lwd = 2, inset = .02, bty = 'n')
```



The green line is a density of the posterior with Gamma(1,1)-prior. This time our posterior is concentrated on the slightly higher values. This is because Log-normal(0,1)-distribution has a higher mean ($E\lambda = 1.65$) and a heavier right tail than the Gamma(1,1)-distribution.

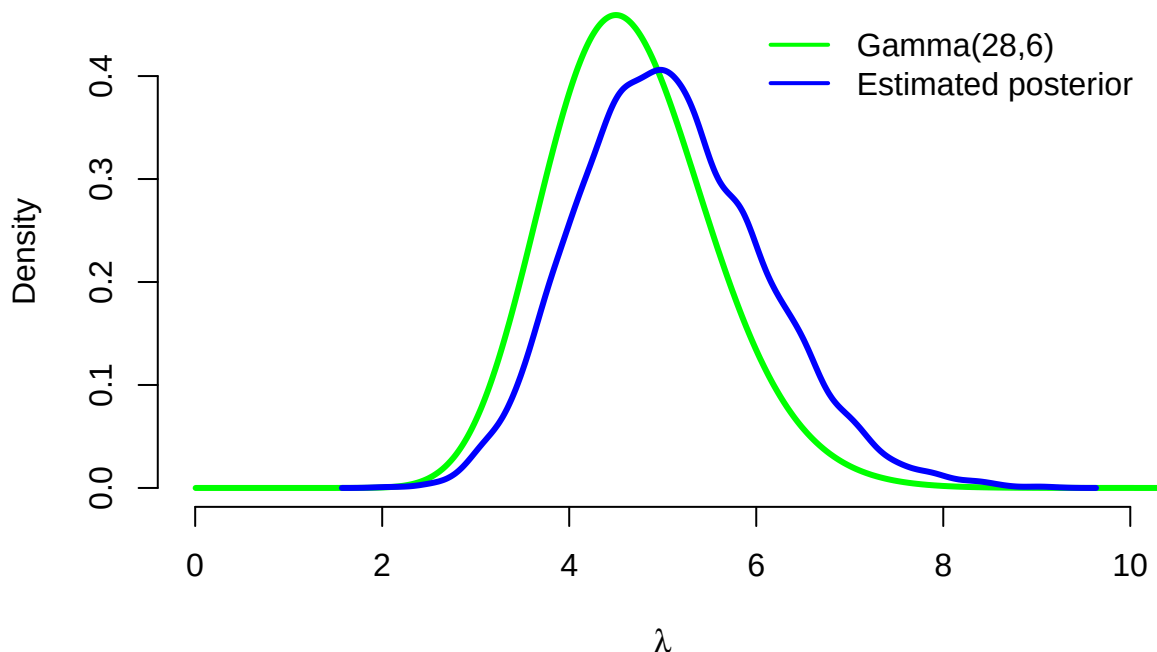
We can also plot estimated posterior density with the log-normal prior, and compare it to the posterior density with the gamma prior:

¹Normally we would compute with the logarithms, which means using values of the function $\log q(\lambda; \mathbf{y})$ instead of $q(\lambda; \mathbf{y})$, and exponentiate as late as possible to avoid over- and underflows and other numerical problems. However, let's not complicate things unnecessarily in this introductory example.

```

density_sim <- density(lambda_sim2)
plot(grid, dgamma(grid, alpha + sum(y), beta + n), type = 'l', col = 'green',
      lwd=3, xlim = c(0,10), bty = 'n', xlab = expression(lambda), ylab = 'Density')
lines(density_sim, type = 'l', col = 'blue', lwd=3 )
legend('topright', legend = c(paste0('Gamma(', sum(y) + alpha,
  ',', n + beta, ')'), 'Estimated posterior'), col = c('green', 'blue'),
      lwd = 2, inset = .02, bty = 'n')

```



4.2 Monte Carlo integration

In Example 4.1.2 we observed that the empirical posterior density obtained by simulation started to resemble very closely the true posterior density obtained analytically with a high simulation size. This phenomenon can also be utilized to compute summary statistics, such as posterior mean, posterior variance, and credible intervals from the simulated sample.

More generally computing integrals by simulation is known as **Monte Carlo integration** or **Monte Carlo method**. It turns on the classical result on a probability theory called a strong law of law numbers².

4.2.1 Strong law of large numbers (SLL)

Let $Y_1, Y_2, \dots$ be i.i.d. random variables with an expected value $\mu := EY_1$ that is finite: $E|Y_1| < \infty$. Now

$$\frac{1}{n} \sum_{i=1}^n Y_i \rightarrow \mu$$

almost surely (a.s.), as $n \rightarrow \infty$.

²There are several versions of law of large numbers with different assumptions; the version introduced here was proved by Kolmogorov in 1930s.

Almost sure convergence means that the sequence converges with a probability one: another way to state the result is

$$P\left(\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n Y_i = \mu\right) = 1.$$

4.2.2 Example of SLL : coinflips

The strong law of law number simply states that the sample mean of i.i.d. random variables converges to an expected value of the distribution with probability one. We intuitively use this result all the time, but the strong law of large numbers states it formally.

Denote by $Y_1, Y_2, \dots$ a series of coinflips, where $Y_1 = 1$ means heads and $Y_1 = 0$ means tails. Assuming a fair coin, $P(Y_1 = 1) = 1/2$, and thus $\mu = EY_1 = 1/2$. By a strong law of large numbers the proportion of heads converges to the probability of heads:

$$\frac{1}{n} \sum_{i=1}^n Y_i \xrightarrow{\text{a.s.}} \frac{1}{2}$$

with probability one. Although there exists an infinite number of sequences which do not converge to $1/2$, such as a sequence of only heads $(1, 1, \dots)$, the probability of the set of these sequences is zero.

4.2.3 Example of Monte carlo integration

Let's revisit Example 4.1.1. Because our simulated values $\lambda_1, \dots, \lambda_S$ are an i.i.d. sample of the posterior distribution, which has a finite expected value, by the strong law of large numbers the posterior mean converges almost surely to this expected value:

$$\frac{1}{S} \sum_{i=1}^S \lambda_i \xrightarrow{\text{a.s.}} E[\Lambda | \mathbf{Y} = \mathbf{y}].$$

This means that we can approximate the posterior expectation with the posterior mean:

$$E[\Lambda | \mathbf{Y} = \mathbf{y}] \approx \frac{1}{S} \sum_{i=1}^S \lambda_i.$$

Because we know the posterior expectation

$$E[\Lambda | \mathbf{Y} = \mathbf{y}] = \frac{\alpha_1}{\beta_1} = \frac{\sum_{i=1}^n Y_i + \alpha}{n + \beta}$$

for this example, we can verify that the posterior mean is very close to the true expected value:

```
alpha_1 <- alpha + sum(y)
beta_1 <- beta + n
alpha_1 / beta_1
```

```
## [1] 4.666667
```

```
mean(lambda_sim)
```

```
## [1] 4.648235
```

The second moment $E\lambda^2$ of the posterior distribution also exists, so we can invoke again the strong law of large numbers for the sequence of random variables $\Lambda_1^2, \Lambda_2^2, \dots$ to approximate the posterior variance:

$$\begin{aligned}\text{Var}[\Lambda \mid \mathbf{Y} = \mathbf{y}] &= E[\Lambda^2 \mid \mathbf{Y} = \mathbf{y}] - E[\Lambda \mid \mathbf{Y} = \mathbf{y}]^2 \\ &\approx \frac{1}{S} \sum_{i=1}^S \lambda_i^2 - \left(\frac{1}{S} \sum_{i=1}^S \lambda_i \right)^2 \\ &= \frac{1}{S-1} \sum_{i=1}^S (\lambda_i - \bar{\lambda})^2.\end{aligned}$$

Again the empirical variance is very close to the true variance of the posterior distribution:

```
alpha_1 / beta_1^2
```

```
## [1] 0.7777778
```

```
var(lambda_sim)
```

```
## [1] 0.7517682
```

We can also use SLL for the sequence of transformations $I_{(a,b)}(\Lambda_1), I_{(a,b)}(\Lambda_2), \dots$ of the parameter Λ , where $I_{(a,b)}$ is an indicator function:

$$I_{(a,b)}(x) = \begin{cases} 1 & \text{if } x \in (a, b), \\ 0 & \text{otherwise.} \end{cases}$$

This means that we can approximate the posterior probabilities by the empirical proportions:

$$\begin{aligned}P(a < \Lambda < b \mid \mathbf{Y} = \mathbf{y}) &= E[I_{(a,b)}(\Lambda) \mid \mathbf{Y} = \mathbf{y}] \\ &\approx \frac{1}{S} \sum_{i=1}^S I_{(a,b)}(\lambda_i) \\ &= \frac{1}{S} \#\{a < \lambda_i < b\}.\end{aligned}$$

Here $\#$ marks the number of elements of the set. Let's demonstrate this by approximating the posterior probabilities $P(\Lambda > 3 \mid \mathbf{Y} = \mathbf{y})$:

```
pgamma(3, alpha_1, beta_1, lower.tail = FALSE)
```

```
## [1] 0.9826824
```

```
mean(lambda_sim > 3)
```

```
## [1] 0.9811
```

and $P(4 < \Lambda < 6 \mid \mathbf{Y} = \mathbf{y})$:

```
pgamma(6, alpha_1, beta_1) - pgamma(4, alpha_1, beta_1)
```

```
## [1] 0.694159
```

```
mean(lambda_sim > 4 & lambda_sim < 6)
```

```
## [1] 0.6984
```

Because the empirical distribution function can be used to approximate the cumulative density function $F_{\Lambda \mid \mathbf{Y}}$ of the posterior distribution, we can also use the empirical quantiles to estimate the quantiles of the posterior distribution, and thus to approximate equal-tailed credible intervals:

```
alpha_conf <- 0.05
```

```
qgamma(alpha_conf / 2, alpha_1, beta_1) # 0.025 - quantile
```

```
## [1] 3.100966
quantile(lambda_sim, alpha_conf / 2)

##      2.5%
## 3.081615
qgamma(1 - alpha_conf / 2, alpha_1, beta_1) # 0.975 - quantiles

## [1] 6.547264
quantile(lambda_sim, 1 - alpha_conf / 2)

##      97.5%
## 6.484451
```

Normally strong law of law numbers is not mentioned explicitly when the empirical quantities are used to approximate expected values, but anyway it is a theoretical result behind these approximations. Also the finiteness of the expected value of the posterior is rarely checked explicitly. However, in the exercises we will have an example of the distribution for which the expected value is infinite.

4.3 Monte Carlo markov chain (MCMC) methods

Our simple grid approximation method worked smoothly, but what would happen if the dimension of the parameter were higher? In our example we set a grid on the interval $(0, 10)$ with a grid increment $i = 0.01$, so the grid had 1000 points. If the parameter were two-dimensional, the grid with the same increment over the two-dimensional interval $(0, 10) \times (0, 10)$ would have million points. And to approximate 3-dimensional parameter with the same grid increment we would need milliard grid points!

Hence, grid approximation quickly becomes infeasible as the dimension of the parameter grows. Rejection and importance sampling have similar problems. This is why for the more complex models sampling is usually done by using **Monte Carlo markov chain (MCMC)** methods. They are based by iteratively sampling from a Markov chain whose stationary distribution is the target distribution, which in the case of Bayesian computation is most often the posterior distribution $p(\theta|\mathbf{y})$.

4.3.1 Markov chain

A discrete time **Markov chain** is a sequence of random variables $\mathbf{X}_1, \mathbf{X}_2, \dots$, which has a Markov property:

$$P(X_{i+1} = x_{i+1} | X_i = x_i, \dots, X_0 = x_0) = P(X_{i+1} = x_{i+1} | X_i = x_i)$$

for all $i = 1, 2, \dots$. This means that any given time the future state X_{i+1} of the state depends only on the present state X_i of the chain, and not on the rest of the history.

A **state space** $\mathcal{S}$ of the Markov chain is the set of all possible values for these random variables X_i .

4.3.2 MCMC sampling

Simple simulation methods, such as rejection sampling, importance sampling, and grid approximation, which we just demonstrated, generate an i.i.d. sample from the target distribution. However, the components of the sample $\theta_1, \dots, \theta_S$ generated by the Monte Carlo markov chain methods has a very high autocorrelation: this means that next value θ_{i+1} is likely to be somewhere near the current value θ_i of the chain. But how does this even work? The trick is that because we generate a large sample, and then use the whole sample to approximate our posterior distribution, the autocorrelation of the single values does not matter.

We already mentioned that the Markov chains used in MCMC methods are designed so that their **stationary distribution** is the target posterior distribution. But what does the stationary distribution mean? It is

simply a distribution $\pi(x)$ with a following property: if you start the chain from the stationary distribution so that $P(X_0 = k) = \pi(k)$ for all $k \in \mathcal{S}$, then also $P(X_i = k) = \pi(k)$ for all $i = 1, 2, \dots$

This means that once the chain hits its stationary distribution it stays there, and thus the value $\pi(k)$ is also a long run proportion of the time the chain stays in a state k . And because we defined the chain so that the stationary distribution π is the posterior distribution $p(\theta|\mathbf{y})$, if the chain moves in it stationary distribution long enough, we get a sample from the posterior!

First iterations of MCMC sampling are usually discarded because the values of the chain before it has converged to the stationary distribution are not representative of the posterior distribution. Exactly how many sampled points are discarded is matter of choice: a very conservative and safe approach is to discard the first half of the iterations. These discarded iterations are called a **burn-in period** or a **warm-up period**. Stan discards the warm-up period automatically, so you don't have to worry about this.

But how do we then know that the chain has converged to its stationary distribution? Actually, in principle this cannot be never known for sure! So we just have to check the model diagnostics (we will examine these more closely later), and check if our results make any sense. Luckily Stan has quite advanced model diagnostics, so it should indicate somehow about the non-convergent chains. An efficient strategy for monitoring the convergence is to run several chains starting from the different initial values in parallel: if they all converge into a similar distribution, it is quite likely that this is the stationary distribution. Stan runs four parallel chains as default.

Markov chains designed so that their stationary distribution is the target posterior distribution, or more generally the implementations of these chains, are called **MCMC samplers**. The most popular ones are the **Gibbs sampler**, and the **Metropolis-Hastings sampler** (actually the Gibbs sampler can also be seen as a special case of the Metropolis-Hasting sampler).

Next we will demonstrate Gibbs sampling with a simple example, so you will get some intuition about how this MCMC sampling business works. However, in this course we will not go into the details about how these samplers work. After this introductory example we will introduce some probabilistic programming tools that have them already implemented, so we don't have to worry about the technical details, and can concentrate on the statistical inference which this course is all about.

4.3.3 Example of MCMC: Gibbs sampler

The Gibbs sampler is an efficient and popular MCMC sampler which updates components of the parameter vector one at a time. Assume that the parameter vector is multi-dimensional $\theta = (\theta_1, \dots, \theta_d)$. For each component θ_j the Gibbs sampler generates a value from the conditional posterior distribution of this component given all the other components:

$$p(\theta_j | \theta_{-j}, \mathbf{y}),$$

where $\theta_{-j} = (\theta_1, \dots, \theta_{j-1}, \theta_{j+1}, \dots, \theta_d)$.

Let's demonstrate this with a 2-dimensional example. Assume that we have one observation $(y_1, y_2) = (0, 0)$ from the two-dimensional normal distribution $N(\mu, \Sigma_0)$, where the parameter of interest is a mean vector $\mu = (\mu_1, \mu_2)$ and the covariance matrix

$$\Sigma_0 = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

is assumed as a known constant matrix. Assume that the covariance is $\rho = -0.7$. Further assume that we are using an improper uniform prior $p(\mu) \propto 1$ for parameter μ . Now the posterior is (do not care about the inference of the posterior right now; we will consider posterior inference for the multi-dimensional parameter on next week) a 2-dimensional normal distribution $N(\mu, \Sigma_0)$.

Of course we could generate a sample from this normal distribution using a library implementation of the multinormal distribution, but let's write a Gibbs sampler to demonstrate MCMC methods in practice.

From the properties of the multinormal distribution we get the conditional posterior distributions of μ_1 given μ_2 , and μ_2 given μ_1 :

$$\begin{aligned}\mu_1 | \mu_2, \mathbf{Y} &\sim N(y_1 + \rho(\mu_2 - y_2), 1 - \rho^2) \\ \mu_2 | \mu_1, \mathbf{Y} &\sim N(y_2 + \rho(\mu_1 - y_1), 1 - \rho^2).\end{aligned}$$

To implement a Gibbs sampler, let's set the parameter and observation values and define these conditional posterior distributions:

```
y <- c(0,0)
rho <- -0.7

mu1_update <- function(y, rho, mu2) rnorm(1, y[1] + rho * (mu2-y[2]), sqrt(1-rho^2))
mu2_update <- function(y, rho, mu1) rnorm(1, y[2] + rho * (mu1-y[1]), sqrt(1-rho^2))
```

Note that in R the normal distribution is parametrized with standard deviation, not variance, so that the parameter is (μ, σ) instead of the usual parameter (μ, σ^2) . A classical R mistake is to give for `dnorm` or `rnorm` the variance instead of the standard deviation, and then wonder why the results look strange... I have done this many times. Anyway, this is why we take the square root of the variance when we plug it into the formula.

Then we will set an initial value (2, 2) for μ , and start sampling:

```
n_sim <- 1000
mu1 <- mu2 <- numeric(n_sim)
mu1[1] <- 2
mu2[1] <- 2

for(i in 2:n_sim) {
  mu1[i] <- mu1_update(y, rho, mu2[i-1])
  mu2[i] <- mu2_update(y, rho, mu1[i])
}
```

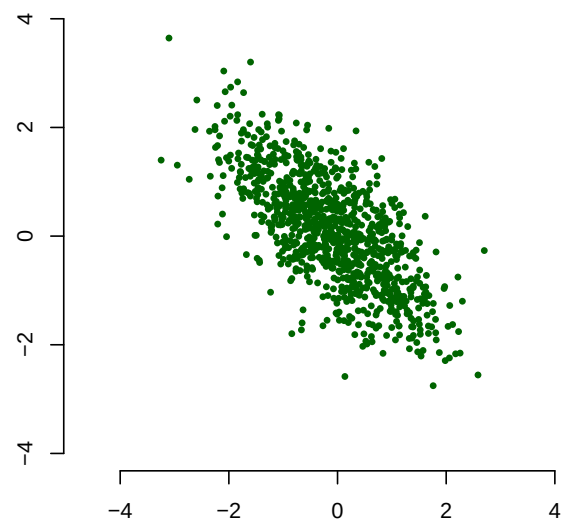
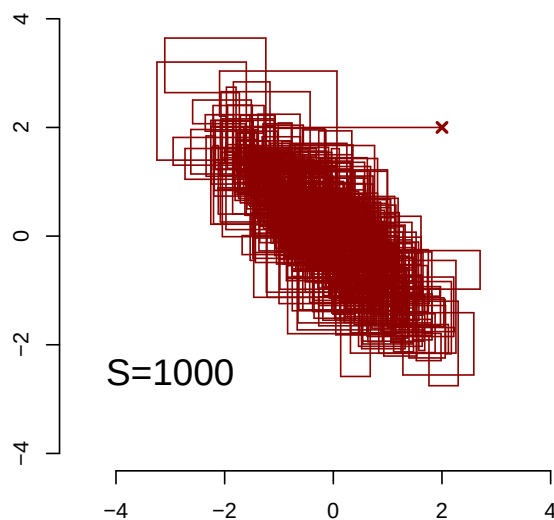
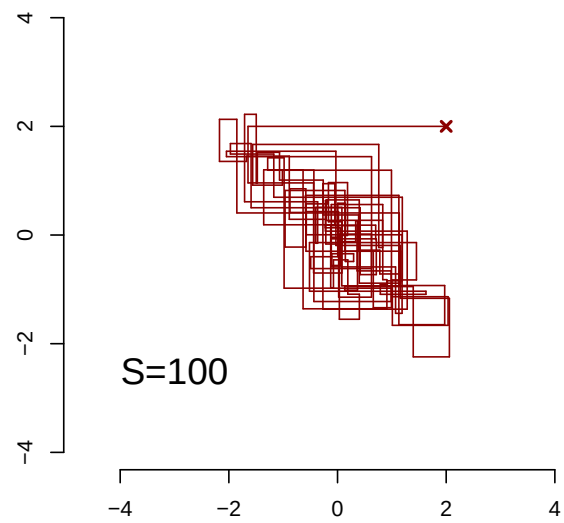
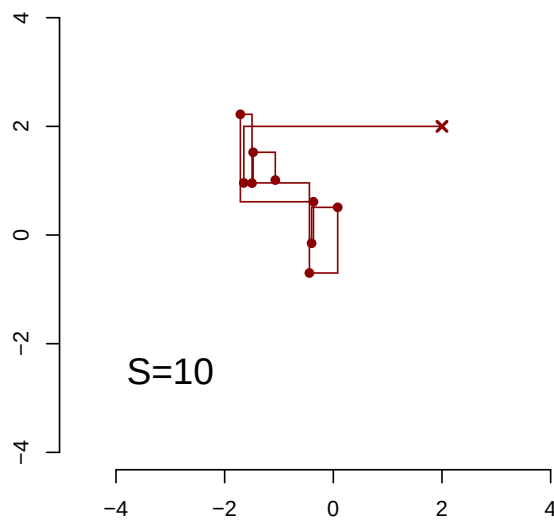
This was all that was required to implement a Gibbs sampler! Let's examine the trace of the sampler after 10, 100, and 1000 simulation rounds:

```
draw_gibbs <- function(mu1, mu2, S, points = FALSE) {
  plot(mu1[1], mu2[1], pch = 4, lwd = 2, xlim = c(-4,4), ylim = c(-4,4), asp = 1,
       xlab = expression(mu[1]), ylab = expression(mu[2]), bty = 'n', col = 'darkred')
  for(j in 2:S) {
    lines(c(mu1[j-1], mu1[j]), c(mu2[j-1], mu2[j]), type = 'l', col = 'darkred')
    lines(c(mu1[j], mu1[j]), c(mu2[j-1], mu2[j]), type = 'l', col = 'darkred')
    if(points) points(mu1[j], mu2[j], pch = 16, col = 'darkred')
  }
  text(x = -3, y = -2.5, paste0('S=', S), cex = 1.75)
}

draw_sample <- function(mu1, mu2, ...) {
  plot(mu1, mu2, pch = 16, col = 'darkgreen',
       xlim = c(-4,4), ylim = c(-4,4), asp = 1, xlab = expression(mu[1]),
       ylab = expression(mu[2]), bty = 'n', ...)
}

par(mfrow = c(2,2), mar = c(2,2,4,4))
draw_gibbs(mu1, mu2, 10, points = TRUE)
draw_gibbs(mu1, mu2, 100)
draw_gibbs(mu1, mu2, n_sim)
```

```
draw_sample(mu1[10:length(mu1)], mu2[10:length(mu2)], cex = 0.7)
```



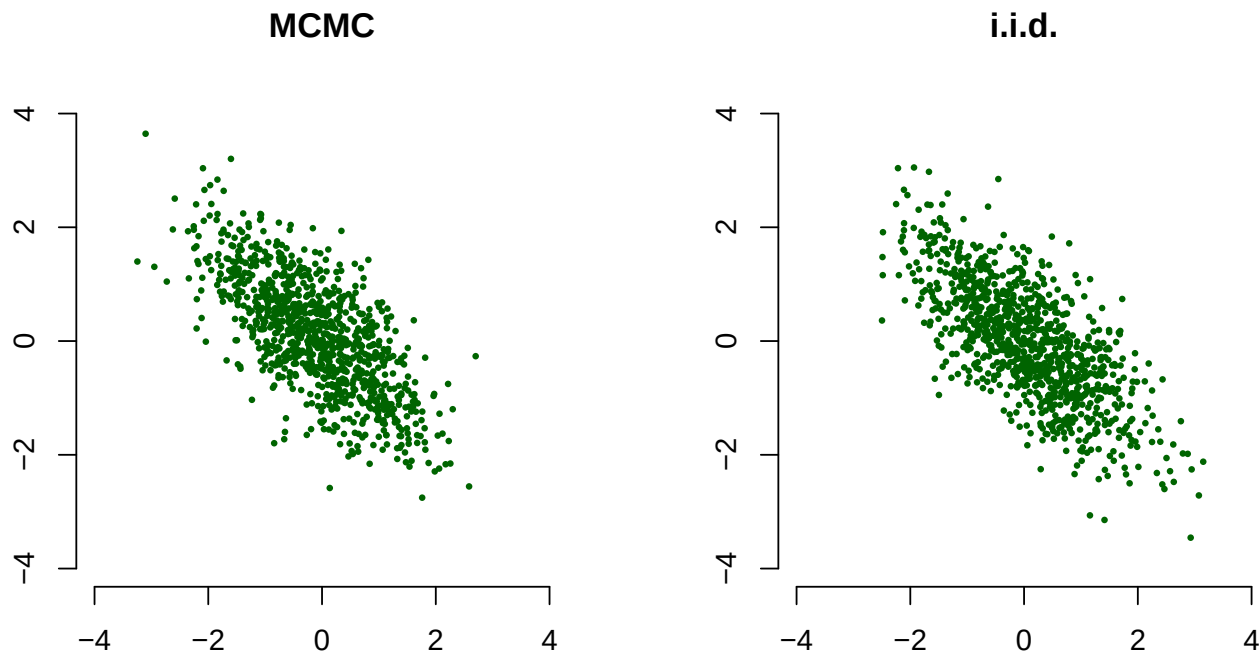
Although the initial value was away from the center of the probability mass of the distribution, the sampler moved quickly to the dense area of the distribution, and after this seemed to explore it efficiently. These trace plots also illustrate the autocorrelation of the sample: subsequent samples (marked explicitly into the first plot with $S = 10$) tend to be close to another.

The last plot contains the sampled points (with a burn-in period of 10 points discarded): although the sample is autocorrelated, this does not matter for the final results. In fact, our MCMC sample is indistinguishable from the i.i.d. sample from the true posterior distribution:

```
Sigma <- matrix(c(1, rho, rho, 1), ncol = 2)
X <- MASS::mvrnorm(n_sim, y, Sigma)

par(mfrow = c(1,2), mar = c(2,2,4,4))
draw_sample(mu1[10:length(mu1)], mu2[10:length(mu2)], cex = 0.5, main = 'MCMC')
draw_sample(X[,1], X[,2], cex = 0.5, main = 'i.i.d.')

```



4.4 Probabilistic programming

Although easy in our introductory example, deriving and testing the samplers quickly becomes very time-consuming when models become more complicated. It may take several weeks worth of effort from a statistician to derive an efficient sampler for the new model. This has been one of the main reasons why it has took so long to adapt Bayesian methods into the mainstream statistical practice, although the main principles of Bayesian statistics are even older than the ones of frequentist statistics, which originated in the beginning of the last century. Another, and in the past of course more restricting, reason has been a lack of computational power required to do efficient sampling.

But nowadays computers are fast enough, and luckily also the human effort required has diminished significantly : **probabilistic programming** systems, which have multi-purpose samplers that can be used to generate a sample of the posterior of the very large array of models, so that we don't have to write a specific sampler for each different model.

Probabilistic programming means basically automatic inference of (often, but not necessarily, Bayesian) statistical models. In principle, the only thing the user has to do is to specify the statistical model in a high-level modelling language, and the probabilistic programming system takes care of the sampling. Using these systems has an advantage that they abstract most of the computational details from us (at least when the sampling works...), so that we can concentrate on building the statistical model instead of implementing the sampler.

One of the pioneers of probabilistic programming tools³ was BUGS (**B**ayesian inference **U**sing **G**ibbs **S**ampling). As the abbreviation hints, it used Gibbs samplers to approximate posterior, and was widely used on the fields requiring applied statistics (or at least by those who used Bayesian methodology on those fields).

However, in the recent years much more powerful probabilistic programming tools have emerged. In part this is because of the development on the **H**amiltonian **M**onte **C**arlo (**HMC**) methods, which allows sampling from a much more general class of models than the Gibbs samplers. The most well-known of these new tools are Stan, PyMC3 and Edward.

³Although BUGS was an early example of probabilistic programming, the nomenclature *probabilistic programming* is quite recent. BUGS project was originated in 1989, so it is much older than this term.

Next we are going to get familiar with probabilistic programming by using **Stan**, and more specifically RStan, which is its R interface. The Stan library itself is written in C++, and in addition to R, it has an interface also for Python (PyStan) and some other high-level languages.

Installing RStan requires little more tuning than installing a normal R package. Detailed instructions for installing RStan for your operating systems can be found from: RStan-Getting-Started. That being said, installing RStan for Linux or MacOS may also work by just running the following line in R:

```
install.packages("rstan", repos = "https://cloud.r-project.org/", dependencies=TRUE)
```

However, your mileage may vary; and following the official instructions is anyway recommended to optimize the compiling and running speed of Stan models.

4.4.1 Minimal Stan-example : model declaration

Now that you have installed Stan, all the hard work is done: fortunately using it fun and easy! When trying new software, I like to run a minimal “Hello World!”-example just to check that everything is set up and working correctly. So as a “Stan - Hello world!” - example, let’s revisit Example 2.1.1 (Poisson sampling distribution with gamma prior) again, and this time use Stan to simulate from the posterior.

Stan models are specified using a high-level modeling language whose syntax resembles R syntax. Models are written into their own `.stan`-files, which Stan first translates into C++ code and then compiles. Let’s start writing our model into a new file, which we can name for example as `poisson.stan`.

A stan model consists of named blocks which are written inside the curly brackets. In principle all the blocks are optional, but three necessary blocks to specify a non-trivial probability model are `data`, `parameters`, and `model`.

First we need to declare the variables for the input data of our model into the `data`-block:

```
data {
  int<lower=0> n;
  int<lower=0> y[n];
}
```

We declared a sample size `n` as a non-negative integer, and `y` as a vector of non-negative integers having `n` components. Note that unlike in R syntax, we had to specify data types of the variables we are declaring; and in addition to specifying our variables as integers, we also constrained them to be *non-negative* integers with the specifier `lower=0`. We could have also constrained our variable into a certain interval: for example we could declare the observation `y` from the binomial distribution $\text{Bin}(n, \theta)$, which is constrained into the interval $(0, n)$, as follows:

```
int<lower=0,upper=n> y;
```

Constraining the variables correctly (so that they are constrained to the support of their distribution⁴) is especially important when declaring the parameters, because Stan uses these constraints when sampling.

Notice also that unlike in R or Python, but like in C++ or Java, each line ends with a semicolon. Omitting it is a syntax error.

Next we declare the parameters of the model in the `parameters`-block:

```
parameters {
  real<lower=0> lambda;
}
```

Parameter of the $\text{Poisson}(\lambda)$ distribution is a real number, so we declare its type as `real`. Note that we do not declare the hyperparameters of the prior $\text{Gamma}(\alpha, \beta)$ -distribution in the `parameters`-block, because we consider them as fixed constants (here $\alpha = 1$, $\beta = 1$), not as random variables like λ .

⁴Support of the continuous probability distribution is a set where its density is positive.

Finally, we specify our probability model in the `model`-block:

```
model {
  lambda ~ gamma(1,1);
  y ~ poisson(lambda);
}
```

Compare this to our usual model declaration:

$$Y_i \sim \text{Poisson}(\lambda) \quad \text{for all } i = 1, \dots, n$$

$$\lambda \sim \text{Gamma}(1, 1)$$

Look pretty similar, right? Stan declaration is even a bit simpler, because Stan supports *vectorization*: a statement

```
y ~ poisson(lambda);
```

for the vector `y` means that each component of this vector follows $\text{Poisson}(\lambda)$ -distribution. We could have also used a more explicit and verbose form:

```
for(i in 1:n)
  y[i] ~ poisson(lambda);
```

A syntax of the for loop is similar to R. The body of the loop is enclosed in the curly brackets; if it consists only of one line, as above, these curly brackets can be omitted.

Our first two blocks consist of only **variable declarations**. The `model`-block is different: it contains **statements**. The statements of the form

```
y ~ poisson(lambda);
```

are called **sampling statements**. They simply tell Stan which probability distribution our variables follow; these sampling statements are used to implement the sampler for the model.

Stan supports most of the well-known distributions, and it is also possible to define own probability distributions by supplying its log-density function. A full list of the available distributions (and tons of other information) can be found from Stan reference manual.

So our full stan model, which we save into the file `poisson.stan`, is:

```
data {
  int<lower=0> n;
  int<lower=0> y[n];
}

parameters {
  real<lower=0> lambda;
}

model {
  lambda ~ gamma(1,1);
  y ~ poisson(lambda);
}
```

4.4.2 Minimal Stan-example : sampling

We have now specified our model and are ready to generate a sample from the posterior. But let's first generate our old data set `y`:

```
lambda_true <- 3
n_sample <- 5
set.seed(111111)
(y <- rpois(n, lambda_true))
```

```
## [1] 4 3 11 3 6
```

Then we wrap our observations and sample size into a list, which has components with the names corresponding to the variables declared in `data`-block of the Stan model:

```
poisson_dat <- list(y = y, n = n_sample)
```

We have not yet loaded a package RStan, so let's do it now:

```
library(rstan)
```

Hmm, it recommends to run some code, so let's do it:

```
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
```

The first line allows saving the compiled model to the hard disk, so it saves time because the model does not have to be recompiled every time it is used. The second line allows Stan to run several Markov chains in parallel, which also saves time.

Now we are finally ready for the actual sampling. The sampling is done via `stan`-function. The following code works if the `poisson.stan`-file that contains the model is in your working directory:

```
fit <- stan(file = 'poisson.stan', data = poisson_dat)
```

```
## recompiling to avoid crashing R session
```

```
# I cut the compiler and sampler messages from here to make this look more clean
```

Function `stan` first compiles the model, then draws a sample from the posterior, and finally returns the sampled values as `stanfit` object. Let's print the summary of the returned `stanfit`-object:

```
fit
```

```
## Inference for Stan model: poisson.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##           mean se_mean   sd  2.5%   25%   50%   75%  97.5% n_eff Rhat
## lambda    4.67     0.02 0.89  3.10  4.04  4.62  5.23  6.56 1301   1
## lp__     14.62     0.02 0.70 12.61 14.46 14.90 15.08 15.13 1842   1
##
## Samples were drawn using NUTS(diag_e) at Wed Mar 13 10:41:23 2019.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Stan runs as default 4 chains for 2000 iterations each, and it discards first half of the iterations as the warm-up period. So the default sample size is 4000, as shown above. Stan reports mean, median and 50% and 95% equal-tailed credible interval for our parameters of interest, in this case λ .

You can also run function `stan` without specifying the argument `data`. In case you omit this argument, Stan tries to find the input data (variables `y` and `n`) from the global R environment. With our model this would probably fail, because we have defined a sample size using the variable `n_sample`, not the variable `n`. Or then it would be pick some `n` we have defined earlier in our code, which may or may not be correct. So it is much more clear and less error-prone to specify the input data explicitly as a list.

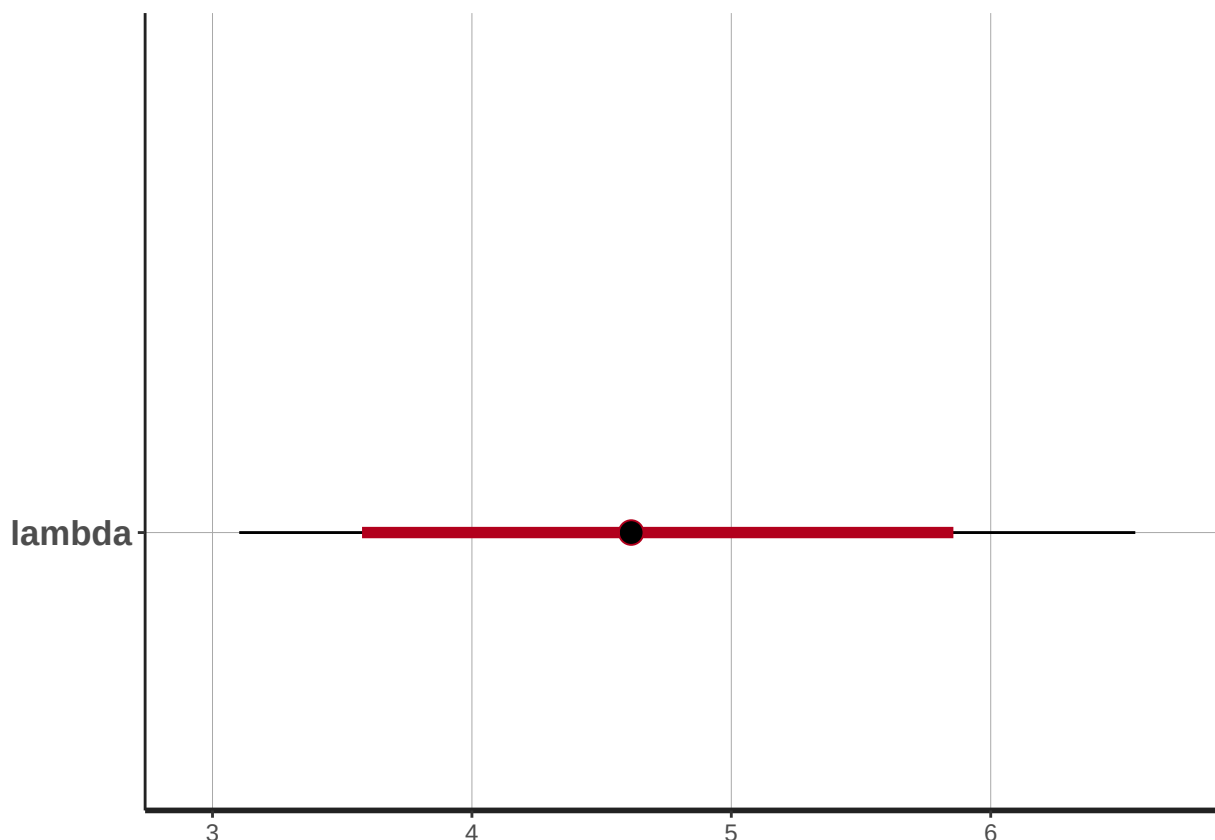
4.4.3 Minimal Stan example : illustrating the results

We can draw a boxplot of the simulated posterior distribution of the parameter λ simply as:

```
plot(fit)
```

```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```



Compare this to Figure 3.1: 95% CI estimated from the posterior lies slightly above the true parameter value ($\lambda = 3$) of the generating distribution, as does the 95% CI computed based on the exact posterior distribution.

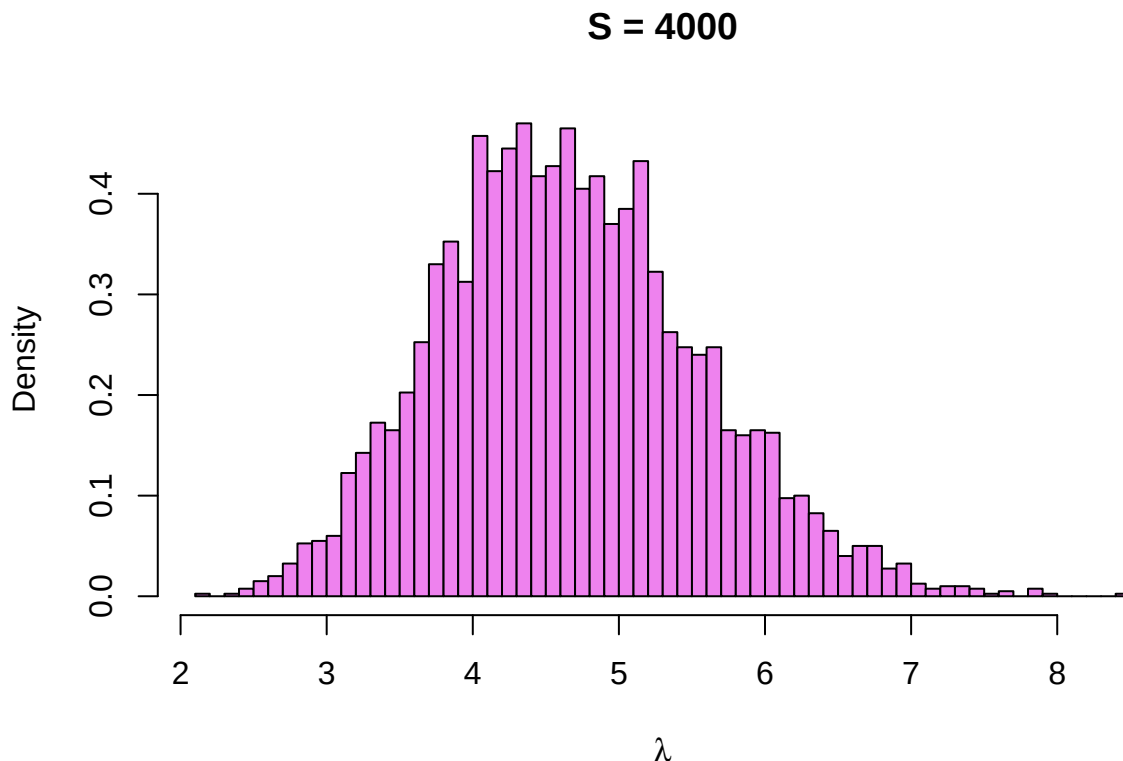
The simulated values can be extracted from the `stanfit`-object with `extract`-function:

```
sim <- extract(fit, permuted = TRUE)
str(sim)
```

```
## List of 2
## $ lambda: num [1:4000(1d)] 4.13 4.28 6.05 5.68 4.5 ...
##   .. attr(*, "dimnames")=List of 1
##   .. ..$ iterations: NULL
## $ lp__ : num [1:4000(1d)] 14.9 15 14.1 14.5 15.1 ...
##   .. attr(*, "dimnames")=List of 1
##   .. ..$ iterations: NULL
```

These simulated values can be used like any sample from the posterior distribution. We can for example draw a histogram of the sample:

```
hist(sim$lambda, breaks = 50, col = 'violet', probability = TRUE,
     main = paste0('S = ', length(sim$lambda)), xlab = expression(lambda))
```



Hmm, it looks a little bit jagged, so maybe we should increase the sample size. Function `stan` has arguments `chains` and `iter`, which can be used to specify the sample size. Let's set iterations to 20000, which means that we should get a sample of $4 \cdot 20000/2 = 40000$ points:

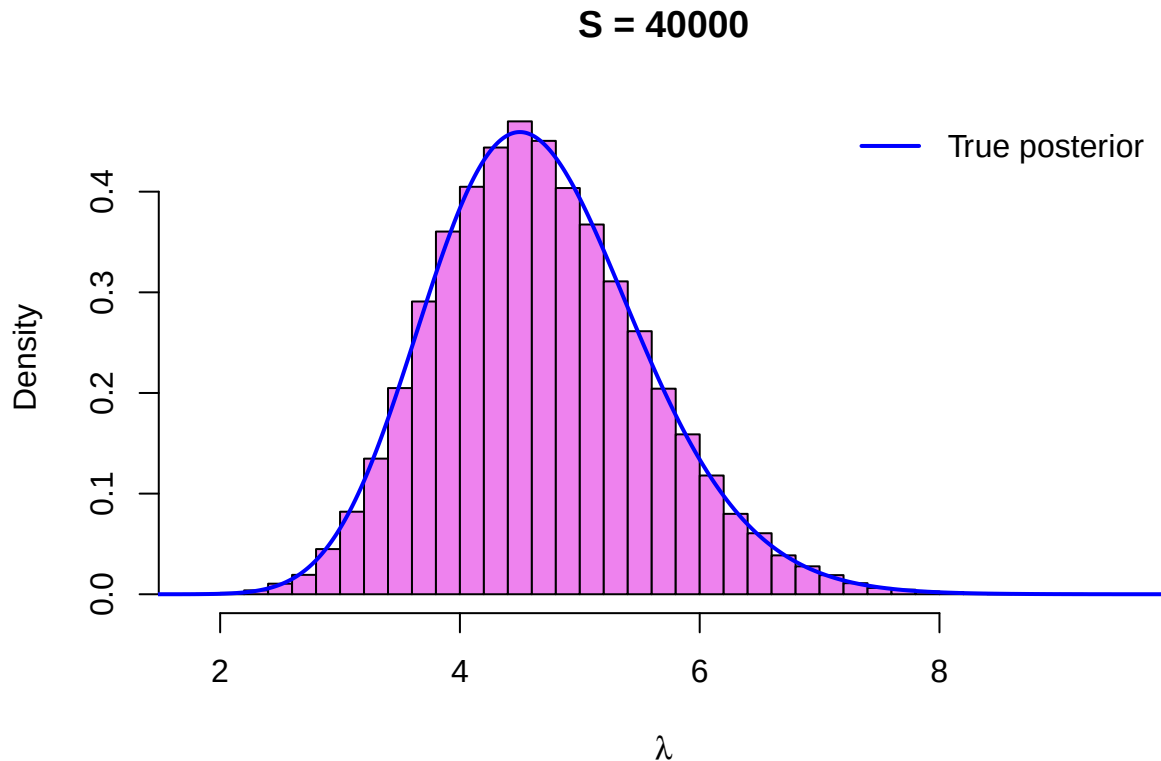
```
fit <- stan(file = 'poisson.stan', data = poisson_dat, iter = 20000, chains = 4)
sim <- extract(fit, permuted = TRUE)
str(sim$lambda)
```

```
## num [1:40000(1d)] 4.28 5.16 4.2 3.52 4.58 ...
## - attr(*, "dimnames")=List of 1
## ..$ iterations: NULL
```

Notice how everything worked much faster this time (at least if we have ran the line `rstan_options(auto_write = TRUE)`), even though the sample size of the simulation was 10 times higher? This is because Stan does not have to compile the model again; for this simple model compiling the model takes actually much longer than sampling from it (unless your simulation sample size is astronomic).

Let's draw a histogram of the sample with the density function of the true posterior Gamma ($\sum_{i=1}^n y_i + 1, n + 1$) on top of it:

```
x <- seq(0,10, by = .01)
hist(sim$lambda, breaks = 50, col = 'violet', probability = TRUE,
     main = paste('S =', length(sim$lambda)), xlab = expression(lambda))
lines(x, dgamma(x, sum(y) + 1, n_sample + 1), col = 'blue', type = 'l', lwd = 2)
legend('topright', legend = 'True posterior', lwd = 2, col = 'blue',
     inset = 0.01, bty = 'n')
```



The histogram looks now smoother as we expected, and it also seems to match the density of the true posterior very well, so everything seems to be working as it should.

4.4.4 Minimal Stan-example: changing the prior

To make our minimal Stan example not so minimal anymore, let's change the prior of our model to the Log-normal distribution, so that the new model is:

$$Y_i \sim \text{Poisson}(\lambda) \quad \text{for all } i = 1, \dots, n$$

$$\lambda \sim \text{Log-normal}(\mu, \sigma^2).$$

Let's also use hyperparameters $\mu = 0, \sigma^2 = 1$. To declare this model in Stan modelling language, the only thing we have to change in our previous declaration is to change the prior distribution for the parameter λ :

```
data {
  int<lower=0> n;
  int<lower=0> y[n];
}

parameters {
  real<lower=0> lambda;
}

model {
  lambda ~ lognormal(0,1);
  y ~ poisson(lambda);
}
```

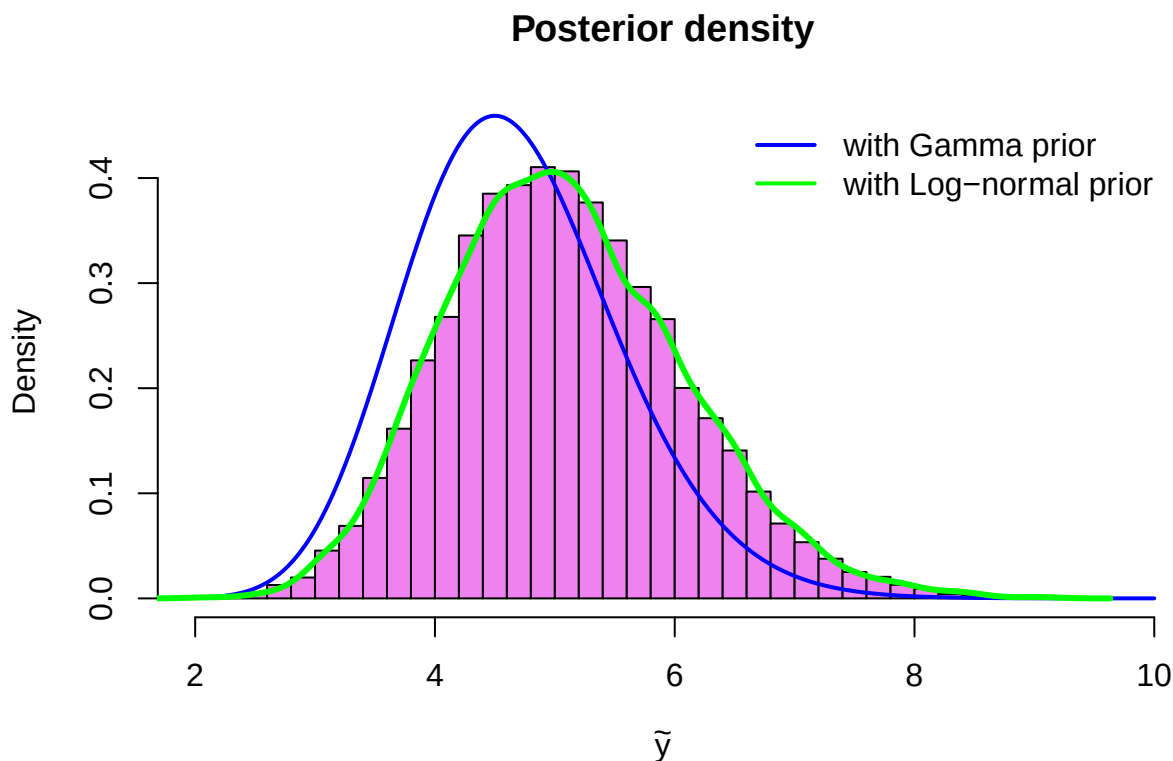
Let's save this model into the file `poisson_lognormal.stan`, and generate a sample from it:

```
fit2 <- stan('poisson_lognormal.stan', iter = 20000, chains = 4)
```

```
## recompiling to avoid crashing R session
```

Now we can draw a histogram of the sample, and compare it to the posterior with the Gamma(1,1)-prior and the estimated density of the posterior with the same Log-normal(0,1)-prior, which we simulated via grid approximation in Example 4.1.2:

```
sim2 <- extract(fit2, permuted = TRUE)
x <- seq(0,10, by = .01)
hist(sim2$lambda, breaks = 50, col = 'violet', probability = TRUE,
     xlab = expression(tilde(y)), ylim = c(0, 0.45),
     main = 'Posterior density')
lines(x, dgamma(x, sum(y) + alpha, n_sample + beta), col = 'blue', type = 'l', lwd = 2)
lines(density_sim, type = 'l', col = 'green', lwd=3)
legend('topright', legend = c('with Gamma prior', 'with Log-normal prior'),
     col = c('blue', 'green'), lwd = 2, bty = 'n')
```



With Stan changing the prior distribution is very convenient. This makes it easy to try different prior distributions to see how sensitive your posterior inference is to the choice of prior distribution. If your posterior inferences are robust with respect to the choice of prior, that is, they do not change very much if you change your prior (assuming of course that the priors are reasonably non-informative), this is a good thing. This is called **sensitivity analysis**.

4.5 Sampling from posterior predictive distribution

We have demonstrated sampling from the posterior distribution, but how about the posterior predictive distribution? Turns out that this is super easy once we have a sample from the posterior distribution!

Let's assume for simplicity that we want to predict probabilities for the new observation $\tilde{Y}$ from the same

process as the original observations $\mathbf{Y} = (Y_1, \dots, Y_n)$ (for many new observations the posterior predictive distribution is same for every observation if they are i.i.d.).

Assume that we have generated the sample $\theta_1, \dots, \theta_S$ from the posterior distribution $p(\mathbf{y}|\theta)$. Now the simulation recipe to generate the sample $\tilde{Y}_1, \dots, \tilde{Y}_S$ from the posterior distribution is simply:

1. For all $s = 1, \dots, S$:
 - Draw $\tilde{Y}_s \sim p(\tilde{y}|\theta_s)$

So for each value of the parameter we sampled from the posterior distribution, we draw a new observation $\tilde{Y}$ from its sampling distribution into which we have plucked the sampled parameter value.

The empirical distribution of this sample can be used to approximate the posterior predictive distribution, which is a sampling distribution averaged (with weights given by the posterior distribution) over the possible parameter values:

$$p(\tilde{y}|\mathbf{y}) = \int p(\tilde{y}|\theta)p(\theta|\mathbf{y}) d\theta$$

Notice how this is different from plugging a single point estimate $\hat{\theta}$, such as the posterior mean or the maximum likelihood estimate to the sampling distribution for the new observation, that is, using $p(\tilde{y}|\hat{\theta})$ to predict the probabilities for the new values.

In practice, we can take a kernel density estimate of our simulated sample $\tilde{y}_1, \dots, \tilde{y}_S$, and use it to approximate the density of the posterior predictive distribution ($\tilde{y}|\mathbf{y}$). Or if the sampling distribution of $\tilde{Y}$ is discrete, then we can simply just normalize the counts into a probability distribution, as we will do in the following example.

4.5.1 Example : sampling from the posterior predictive distribution

Let's revisit our first Stan example (Example 4.4.1). Assume that we want a predictive distribution $p(\tilde{y}|\mathbf{y})$ for the new observation $\tilde{Y} \sim \text{Poisson}(\lambda)$ given the old observations $Y_1, \dots, Y_n$.

Now that we have generated the sample $\lambda_1, \dots, \lambda_S$ from the posterior distribution, we can generate the sample $\tilde{y}_1, \dots, \tilde{y}_S$ from the posterior predictive distribution simply as:

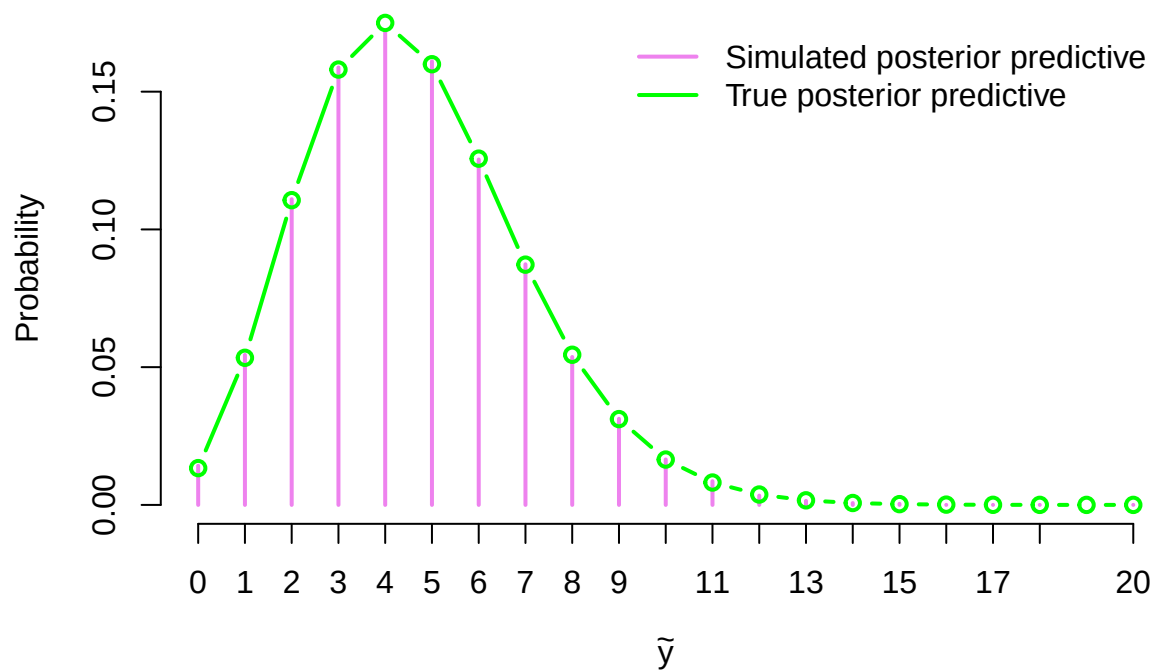
```
y_pred <- rpois(length(lambda_sim), lambda_sim)
```

Because the sampling distribution of $\tilde{Y}$ is discrete, we can approximate the posterior predictive distribution by normalising the counts of our simulated sample into a probability distribution. We have solved the true posterior predictive distribution

$$\tilde{Y} | \mathbf{Y} \sim \text{Neg-bin} \left(\sum_{i=1}^n y_i + \alpha, \frac{n + \beta}{n + \beta + 1} \right)$$

for this model in Example 2.1.2, so let's draw both our approximation and the true distribution to verify that they closely match each other:

```
y_pred <- rpois(length(sim$lambda), sim$lambda)
post_pred <- table(y_pred) / sum(table(y_pred))
plot(post_pred, col = 'violet', lwd = 2, ylab = 'Probability',
      xlab = expression(tilde(y)), bty = 'n')
x <- 0:20
lines(x, dnbinom(x, sum(y) + alpha, (n_sample + beta) / (n_sample + beta + 1)),
      col = 'green', type = 'b', lwd = 2)
legend('topright', legend = c('Simulated posterior predictive',
                              'True posterior predictive'), col = c('violet', 'green'),
      lwd = 2, bty = 'n', inset = 0.01)
```



Chapter 5

Multiparameter models

We have actually already examined computing the posterior distribution for the multiparameter model because we have made an assumption that the parameter $\boldsymbol{\theta} = (\theta_1, \dots, \theta_d)$ is a d -component vector, and examined one-dimensional parameter θ as a special case of this.

For instance, in the exercises we computed a posterior distribution for the parameter $\boldsymbol{\theta}$ of the multinomial distribution $\text{Multinom}(n, \boldsymbol{\theta})$. We were interested in the values of the whole parameter vector $\boldsymbol{\theta} = (\theta_1, \dots, \theta_d)$: this means that the full posterior distribution $p(\boldsymbol{\theta}|\mathbf{y})$ was the desired result. This situation did not in principle differ from the one-dimensional case.

However, often we are not interested in the full posterior $p(\boldsymbol{\theta}|\mathbf{y})$, but only in the marginal posterior distributions of some of the components of the parameter vector.

A classical example is a case in which we are interested in measuring some quantity, for example the speed of light, and model our measurements $Y_1, \dots, Y_n$ of the value of this quantity as an independent sample from the normal distribution:

$$Y_i \sim N(\mu, \sigma^2) \quad \text{for all } i = 1, \dots, n.$$

Now the parameter $\boldsymbol{\theta} = (\mu, \sigma^2)$ of the model is two-dimensional, but sometimes we are only interested in the true value of the quantity μ , and not so much on our measurement error σ^2 . The parameter σ^2 is called a **nuisance parameter** here.

More generally, we will consider a situation in which the parameter vector $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ is partitioned into two (possibly also vector-valued) components, $\boldsymbol{\theta}_1$ being the parameter of interest, and $\boldsymbol{\theta}_2$ being the nuisance parameter.

5.1 Marginal posterior distribution

Assume the partition of the parameter vector into two components: $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$. A distribution $p(\boldsymbol{\theta}_1|\mathbf{y})$ of the parameter of interest¹ given the data is called a **marginal posterior distribution**, and it can be computed by integrating the nuisance parameter out of the full posterior distribution:

$$p(\boldsymbol{\theta}_1|\mathbf{y}) = \int p(\boldsymbol{\theta}|\mathbf{y}) \, d\boldsymbol{\theta}_2$$

This integral can also be written as

$$\begin{aligned} p(\boldsymbol{\theta}_1|\mathbf{y}) &= \int p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2|\mathbf{y}) \, d\boldsymbol{\theta}_2 \\ &= \int p(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2, \mathbf{y}) p(\boldsymbol{\theta}_2|\mathbf{y}) \, d\boldsymbol{\theta}_2. \end{aligned}$$

¹Here we refer to $\boldsymbol{\theta}_1$ as the parameter of interest and to $\boldsymbol{\theta}_2$ as the nuisance parameter because of the clarity of presentation, but of course $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ can be any partition of the parameter vector.

A distribution $p(\theta_1|\theta_2, \mathbf{y})$ is called a **conditional posterior distribution** of the parameter θ_1 ; the above integral can be seen as an weighted average of the conditional posterior distribution, where the weights are given by the marginal posterior distribution of the nuisance parameter θ_2 .

5.2 Inference for the normal distribution with known variance

The normal distribution is ubiquitous in the statistics and machine learning models, and it is also a nice example of the multiparameter inference, because its parameter is two-dimensional $\theta = (\theta, \sigma^2)$, where often (but not always) an expected value θ is considered a parameter of interest, and a variance σ^2 is considered a nuisance parameter. Thus, we will go through the posterior inference for the normal model distribution here as an example of the multiparameter inference.

However, before going to the actual multiparameter inference, we will consider a simpler example where we assume the variance σ_0^2 of the normal distribution fixed. This is actually an example of the one-parameter conjugate model, because the only unknown parameter is the expected value θ of the distribution.

The posterior distribution for the inverse case in which the expected value is assumed to be known, but the variance is unknown, was derived in the exercises. These simple models in which one of the parameters is fixed are useful for deriving the conditional posterior distributions in the case where both the mean and variance are unknown.

5.2.1 One observation

Assume first that we have one observation Y from the normal distribution with an unknown mean θ and a fixed variance $\sigma_0^2 > 0$. A conjugate distribution for this model is a normal distribution, so that the full model is:

$$\begin{aligned} Y &\sim N(\theta, \sigma_0^2) \\ \theta &\sim N(\mu_0, \tau_0). \end{aligned}$$

The likelihood of this model can be written as

$$p(y|\theta) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{(y-\theta)^2}{2\sigma_0^2}\right) \propto \exp\left(-\frac{\theta^2 - 2y\theta}{2\sigma_0^2}\right),$$

and the prior distribution as

$$p(\theta) = \frac{1}{\sqrt{2\pi\tau_0^2}} \exp\left(-\frac{(\theta - \mu_0)^2}{2\tau_0^2}\right) \propto \exp\left(-\frac{\theta^2 - 2\mu_0\theta}{2\tau_0^2}\right).$$

In both the likelihood and the prior the term in the exponent is a quadratic function of the parameter θ , so this looks promising: we only have to recognize the same quadratic form of θ from the posterior to see that it is a normal distribution. Let's write the unnormalized posterior using the Bayes formula to find out the parameters of the posterior distribution:

$$\begin{aligned} p(\theta|y) &\propto p(y|\theta)p(\theta) \\ &\propto \exp\left(-\frac{\theta^2 - 2\mu_0\theta}{2\tau_0^2} - \frac{\theta^2 - 2y\theta}{2\sigma_0^2}\right) \\ &= \exp\left(-\frac{\sigma_0^2(\theta^2 - 2\mu_0\theta) + \tau_0^2(\theta^2 - 2y\theta)}{2\tau_0^2\sigma_0^2}\right) \\ &\propto \exp\left(-\frac{(\sigma_0^2 + \tau_0^2)\theta^2 - 2(\sigma_0^2\mu_0 + \tau_0^2y)\theta}{2\tau_0^2\sigma_0^2}\right) \\ &\propto \exp\left(-\frac{\theta^2 - 2\mu_1\theta}{2\tau_1^2}\right), \end{aligned}$$

where

$$\mu_1 = \frac{\sigma_0^2 \mu_0 + \tau_0^2 y}{\sigma_0^2 + \tau_0^2},$$

and

$$\tau_1^2 = \frac{\tau_0^2 \sigma_0^2}{\sigma_0^2 + \tau_0^2}.$$

This means that the posterior distribution of the parameter θ is the normal distribution

$$\theta | Y = y \sim N(\mu_1, \tau_1^2).$$

We can also write the parameters of the posterior distribution by using the **precision**, which is an inverse of the variance $1/\tau^2$. The posterior precision can be written as a sum of the prior precision and the sampling precision (which was assumed to be a known constant):

$$\frac{1}{\tau_1^2} = \frac{1}{\tau_0^2} + \frac{1}{\sigma_0^2},$$

and the posterior mean can be written as a convex combination of the prior mean and the value of the only observation:

$$\mu_1 = \frac{\frac{1}{\tau_0^2} \mu_0 + \frac{1}{\sigma_0^2} y}{\frac{1}{\tau_0^2} + \frac{1}{\sigma_0^2}},$$

where the weights are the prior and the sampling precision.

5.2.2 Many observations

In the previous example we derived the posterior distribution for the normal model with only one observation. But of course usually we have several observations, in which case the full model is:

$$\begin{aligned} Y_i &\sim N(\theta, \sigma^2) \quad \text{for all } i = 1, \dots, n, \\ \theta &\sim N(\mu_0, \tau^2). \end{aligned}$$

By repeating the above derivation, this time using the joint likelihood $p(\mathbf{y}|\theta) = \prod_{i=1}^n p(y_i|\theta)$ instead of the likelihood of the single observation, or by using the previous result and the fact that the mean of the normally distributed random variables has a normal distribution

$$\bar{Y} \sim N(\theta, \sigma^2/n),$$

(and that the sample mean $\bar{y}$ is a so called *sufficient statistic* for this model) we can see that the posterior is the normal distribution

$$\theta | \mathbf{Y} = \mathbf{y} \sim N(\mu_n, \tau_n^2),$$

where the expected value is

$$\mu_n = \frac{\frac{1}{\tau_0^2} \mu_0 + \frac{n}{\sigma_0^2} \bar{y}}{\frac{1}{\tau_0^2} + \frac{n}{\sigma_0^2}},$$

and the precision is

$$\frac{1}{\tau_n^2} = \frac{1}{\tau_0^2} + \frac{n}{\sigma_0^2}.$$

We can again see that the posterior mean is the convex combination of the prior mean and the mean of the observations, and that the weight of the data mean is proportional to the number of observations: the higher the sample size, the stronger the influence of the data on the posterior mean.

5.3 Inference for the normal distribution with noninformative prior

Next we will consider the general case in which we have again n observations from the normal distribution, but this time both the mean μ and variance of the distribution are assumed unknown. Using a noninformative improper prior $1/\sigma^2$ for the parameter (μ, σ^2) our full model is:

$$Y_i \sim N(\mu, \sigma^2) \quad \text{for all } i = 1, \dots, n,$$

$$p(\mu, \sigma^2) \propto \frac{1}{\sigma^2}.$$

First we will derive the full posterior distribution of this model, and using this full posterior derive the marginal posteriors for both the expected value μ and the variance σ^2 .

The general conjugate prior for this model is set hierarchically as:

$$\mu | \sigma^2 \sim N(\mu_0, \sigma^2 / \kappa_0),$$

$$\sigma^2 \sim \text{Inv-}\chi^2(\nu_0, \sigma_0^2),$$

so that the joint prior for the parameters is

$$p(\mu, \sigma^2) \propto (\sigma^2)^{-(\nu_0+3)/2} \exp \left\{ -\frac{\nu_0 \sigma_0^2 + \kappa_0 (\mu_0 - \mu)^2}{2\sigma^2} \right\}.$$

This distribution is called the **normal inverse chi-squared distribution (NIX)** and denoted as

$$(\mu, \sigma^2) \sim N\text{-Inv-}\chi^2(\mu_0, \sigma_0^2 / \kappa_0, \nu_0, \sigma_0^2).$$

We will show in the exercises that the full posterior distribution for the parameter (μ, σ^2) is also of this form, but let's first solve the joint posterior and the marginal posteriors in the special case of noninformative prior.

5.3.1 Full posterior

By using the following factorization (this can be easily proven by writing the left hand side out and rearranging terms):

$$\sum_{i=1}^n (y_i - \mu)^2 = \sum_{i=1}^n (y_i - \bar{y})^2 + n(\bar{y} - \mu)^2,$$

and the likelihood for n independent observations from the same normal distribution:

$$p(\mathbf{y} | \mu, \sigma^2) = \prod_{i=1}^n p(y_i | \mu, \sigma^2) \propto \sigma^{-n} \exp \left\{ -\frac{\sum_{i=1}^n (y_i - \mu)^2}{2\sigma^2} \right\}$$

we can write the unnormalized joint posterior distribution of both μ and σ^2 as:

$$\begin{aligned} p(\mu, \sigma^2 | \mathbf{y}) &\propto p(\mu, \sigma^2) p(\mathbf{y} | \mu, \sigma^2) \\ &\propto \sigma^{-2} \cdot \sigma^{-n} \exp \left\{ -\frac{\sum_{i=1}^n (y_i - \mu)^2}{2\sigma^2} \right\} \\ &\propto \sigma^{-n-2} \exp \left\{ -\frac{\sum_{i=1}^n (y_i - \bar{y})^2 + n(\bar{y} - \mu)^2}{2\sigma^2} \right\} \\ &\propto \sigma^{-n-2} \exp \left\{ -\frac{(n-1)s^2 + n(\bar{y} - \mu)^2}{2\sigma^2} \right\}, \end{aligned}$$

where the sample mean

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

and the sample variance

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (y_i - \bar{y})^2$$

form a two-dimensional sufficient statistics $(\bar{y}, s^2)$ for the parameter (μ, σ^2) .

This is a special case of the so-called normal inverse chi-squared distribution, which is a two-dimensional four-parameter distribution. To make this a little bit more concrete, we will generate a sample of 25 points from a standard normal distribution $N(0, 1)$, and plot (unnormalized) full posterior distributions for the first 2, 5, 10 and 25 points. Notice that because we use noninformative prior, the results are not very stable: the posterior for the first two observations is drastically different depending on the values of the observations. You can verify this by running the code without setting the random seed or using different values for the seed. However, with a sample size of $n = 25$ the posterior starts to concentrate on the neighborhood of the parameter value $(\mu, \sigma^2) = (0, 1)$ of the true generating distribution:

```
set.seed(0)

q <- function(mu, sigma_squared, m_0, kappa_0, nu_0, sigma_squared_0) {
  (1 / sigma_squared)^(nu_0 + 3 / 2) *
  exp(-(nu_0 * sigma_squared_0 + kappa_0 * (mu - m_0)^2) / (2 * sigma_squared))
}

persp_NI <- function(m_0, kappa_0, nu_0, sigma_squared_0,
  xlim = c(-1.5, 1.5), ylim = c(0, 2), grid_incr = .05, ...) {
  grid_1 <- seq(-1.5, 1.5, by = grid_incr)
  grid_2 <- seq(0.01, 2, by = grid_incr)
  grid_2d <- expand.grid(grid_1, grid_2)

  grid_density <- q(grid_2d[, 1], grid_2d[, 2], m_0, kappa_0, nu_0, sigma_squared_0)
  head(grid_density)
  grid_matrix1 <- matrix(grid_density / sum(grid_density), nrow = length(grid_1))

  persp(grid_1, grid_2, grid_matrix1, xlim = xlim, ylim = ylim, theta = -45, phi = 30,
    xlab = 'mean', ylab = 'variance', zlab = 'Density', ...)
}

persp_posterior <- function(y, mu_0, kappa_0, nu_0, sigma_squared_0) {
  print(y)
  n <- length(y)
  mu_n <- (kappa_0 * mu_0 + n * mean(y)) / (kappa_0 + n)
  kappa_n <- kappa_0 + n
  nu_n <- nu_0 + n
  sigma_squared_n <- (nu_0 * sigma_squared_0 + (n-1) * var(y) + (kappa_0 * n) /
    (kappa_0 + n) * (mean(y) - mu_0)^2) / nu_n
  persp_NI(mu_n, kappa_n, nu_n, sigma_squared_n)
}

S <- 100
y <- sample(rnorm(S))
par(mfrow = c(2, 2), mar = c(0, 0, 2, 2))
n_stops <- c(2, 5, 10, 25)
```

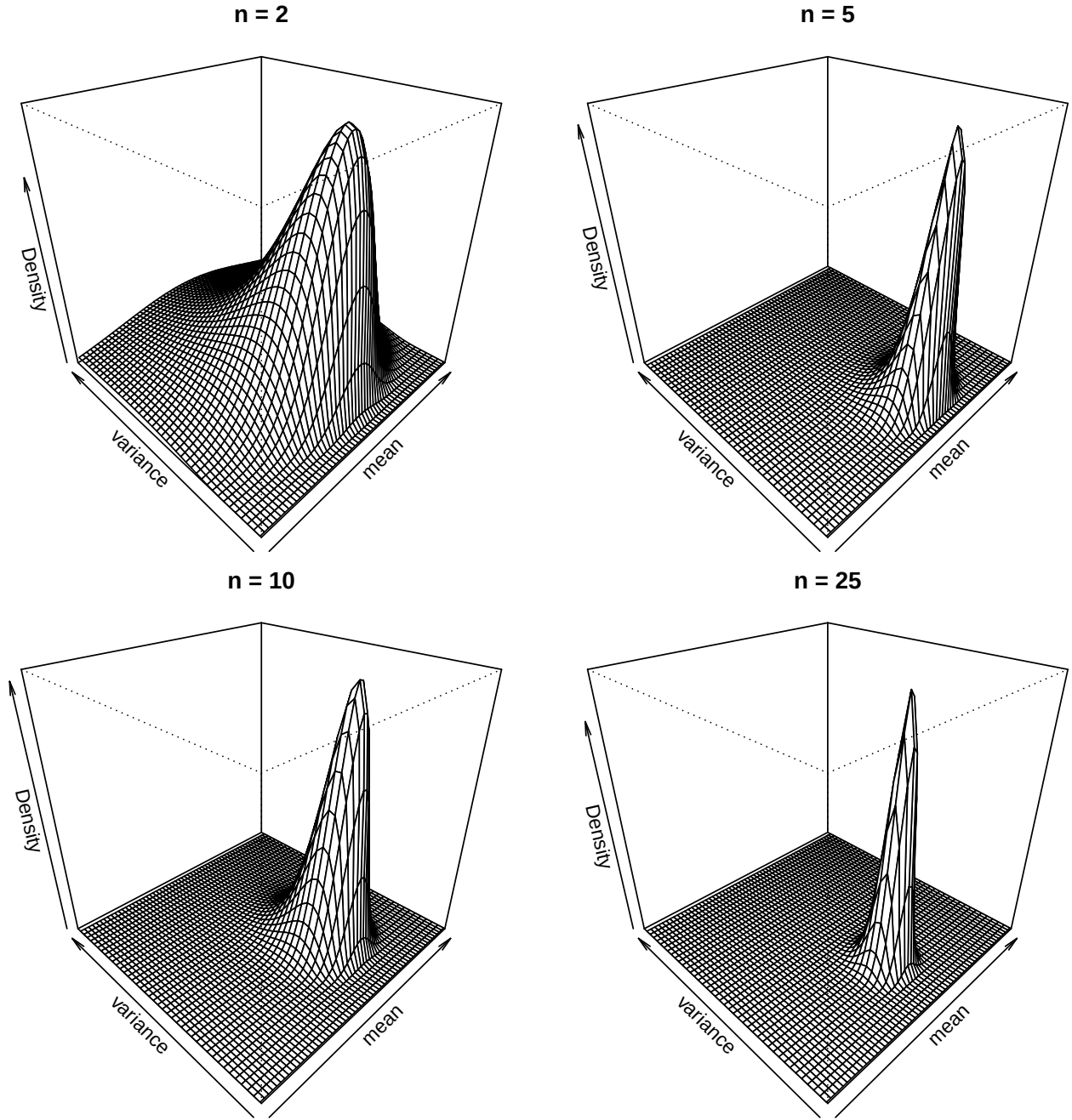
```
for(n in n_stops) {  
  y_crnt <- y[1:n]  
  cat('n =', n, ', mean =', round(mean(y_crnt), 2),  
      ', variance =', round(var(y_crnt), 2), '\n\n')  
  persp_NI(m_0 = mean(y_crnt), kappa_0 = n, nu_0 = n - 1, sigma_squared_0 = var(y_crnt),  
           main = paste('n =', n))  
}
```

```
## n = 2 , mean = 0.09 , variance = 2
```

```
## n = 5 , mean = 0.26 , variance = 0.53
```

```
## n = 10 , mean = 0.37 , variance = 1.09
```

```
## n = 25 , mean = 0.07 , variance = 0.86
```



5.3.2 Marginal posterior for the expected value

Assume that the expected value μ of the distribution is the parameter of interest and that the variance σ^2 is the nuisance parameter. Using the unnormalized joint posterior derived above, we get the marginal posterior of the expected value by integrating it over the variance.

The density of the inverted chi-squared distribution is

$$p(\sigma^2) = \frac{(\nu_0/2)^{\nu_0/2}}{\Gamma(\nu_0/2)} (\sigma_0^2)^{\nu_0/2} (\sigma^2)^{-(\nu_0/2+1)} \exp\left(-\frac{\nu_0 \sigma_0^2}{2\sigma^2}\right) \quad \text{when } \sigma^2 > 0,$$

and by adding the right constant term we can complete integral into the integral of the inverted chi-squared

distribution with parameters

$$\nu_0 := n$$

and

$$\sigma_0^2 := (n-1)s^2/n + (\bar{y} - \mu)^2$$

over its support:

$$\begin{aligned} p(\mu|\mathbf{y}) &= \int p(\mu, \sigma^2|\mathbf{y}) d\sigma^2 \\ &\propto \int_0^\infty \sigma^{-n-2} \exp \left\{ -\frac{(n-1)s^2 + n(\bar{y} - \mu)^2}{2\sigma^2} \right\} d\sigma^2 \\ &\propto (\sigma_0^2)^{-n/2} \int_0^\infty \frac{(n/2)^{-n/2}}{\Gamma(n/2)} (\sigma_0^2)^{n/2} (\sigma^2)^{-(\frac{n}{2}+1)} \exp \left\{ -\frac{n\sigma_0^2}{2\sigma^2} \right\} d\sigma^2 \\ &= ((n-1)s^2/n + (\bar{y} - \mu)^2)^{-\frac{n}{2}} \\ &= \left(1 + \frac{1}{(n-1)} \left(\frac{\mu - \bar{y}}{s/\sqrt{n}} \right)^2 \right)^{-\frac{(n-1)+1}{2}}. \end{aligned}$$

This can be recognized as the kernel of the non-standard t -distribution with a degree of freedom $n-1$:

$$\mu | \mathbf{Y} = \mathbf{y} \sim t_{n-1}(\bar{y}, s^2/n).$$

Thus, the scaled and shifted parameter μ follows a standard t distribution with a degree of freedom $n-1$:

$$\frac{\mu - \bar{y}}{s/\sqrt{n}} \Big| \mathbf{Y} = \mathbf{y} \sim t_{n-1}.$$

This is an interesting parallel to the result from the classical statistics stating that the so-called t -statistic, which is a normalized sample mean, has the same distribution² given the expected value and the variance of the sampling distribution:

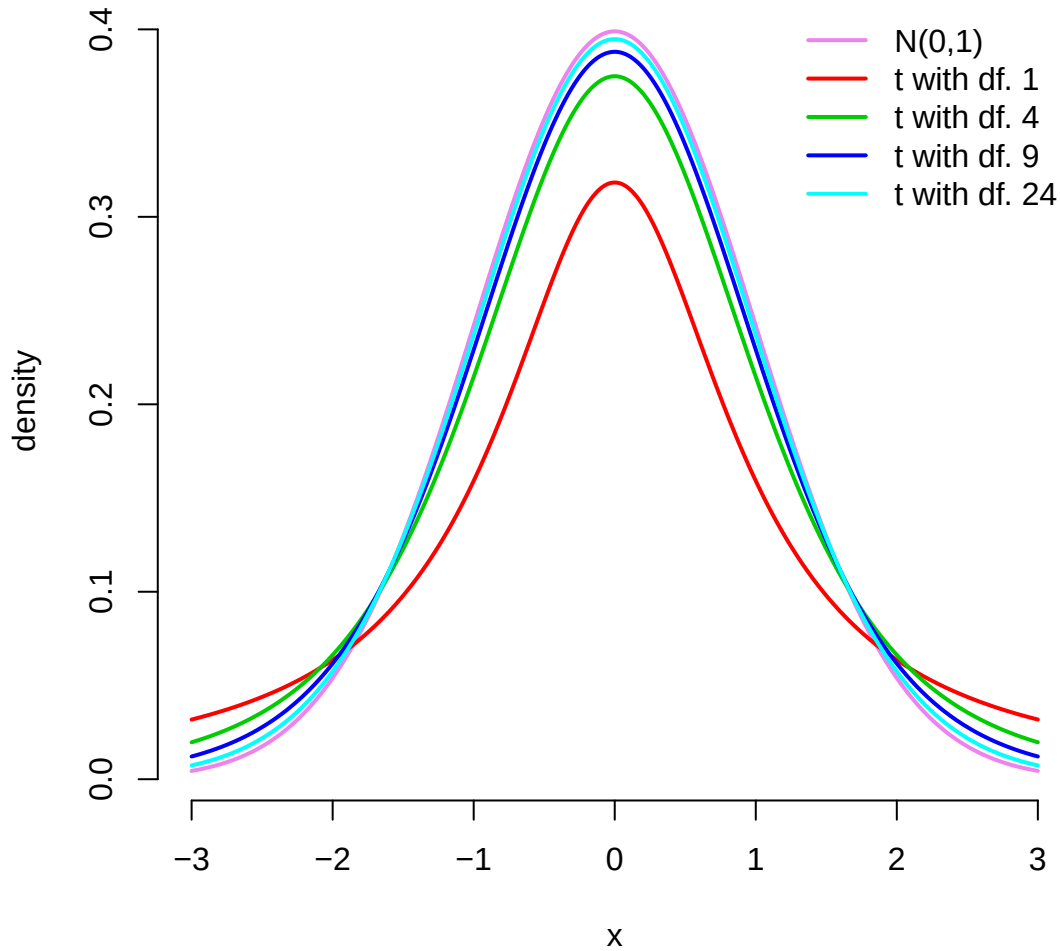
$$\frac{\bar{y} - \mu}{s/\sqrt{n}} \Big| \mu, \sigma^2 \sim t_{n-1}.$$

A t -distribution has a similar shape than the normal distribution, but it has heavier tails. However, with higher degrees of freedom its shape comes closer to the normal distribution. This behaviour can be seen by standard plotting the densities of standard t -distributions with different degrees of freedom and comparing them to the density of the standard normal distribution $N(0,1)$:

```
x <- seq(-3, 3, by = .01)
n <- c(2,5,10,25)

plot(x, dnorm(x), col = 'violet', lwd = 2, bty = 'n', ylab = 'density', type = 'l')
for(i in seq_along(n))
  lines(x, dt(x, n[i]-1), col = i+1, lwd = 2)
legend('topright', legend = c('N(0,1)', paste('t with df.', n-1)),
      col = c('violet', 2:(length(n)+1)), lwd = 2, bty = 'n')
```

²This result holds exactly for the observations $Y_i \sim N(\mu, \sigma^2)$ from the normal distribution (the model examined here), and asymptotically otherwise.



5.3.3 Marginal posterior for the variance

We can also derive the marginal posterior for the variance of the distribution. This time we will utilize the first of the tricks introduced in Example 1.3.1. The **gaussian integral** (a.k.a. **Euler-Poisson integral**):

$$\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}$$

can be evaluated by a transform into the polar coordinates. Also by the change of variables we can see that the gaussian integral of the affine transformation is:

$$\int_{-\infty}^{\infty} e^{-a(x+b)^2} dx = \sqrt{\frac{\pi}{a}}.$$

This is how the normalizing constant of the normal distribution is computed, so we see now that we could have as well used the second of the integrating tricks (completing the integral to the integral of the density function over its support by adding a normalizing constant)³.

So we get the marginal posterior of the variance σ^2 by integrating the expected value μ out of the joint

³And more generally, the second of integrating tricks always reduces into this first trick of doing a change of variables to recognize a familiar integral.

posterior distribution:

$$\begin{aligned}
 p(\sigma^2 | \mathbf{y}) &= \int_{-\infty}^{\infty} p(\mu, \sigma^2 | \mathbf{y}) d\mu \\
 &\propto \int_{-\infty}^{\infty} \sigma^{-n-2} \exp \left\{ -\frac{(n-1)s^2 + n(\bar{y} - \mu)^2}{2\sigma^2} \right\} d\mu \\
 &= (\sigma^2)^{-n/2+1} \exp \left\{ -\frac{(n-1)s^2}{2\sigma^2} \right\} \int_0^{\infty} \exp \left\{ -\frac{n}{2\sigma^2} (\bar{y} - \mu)^2 \right\} d\mu \\
 &= (\sigma^2)^{-n/2+1} \exp \left\{ -\frac{(n-1)s^2}{2\sigma^2} \right\} \sqrt{\frac{2\pi\sigma^2}{n}} \\
 &\propto (\sigma^2)^{-(\frac{n-1}{2}+1)} \exp \left\{ -\frac{(n-1)s^2}{2\sigma^2} \right\}.
 \end{aligned}$$

This can be recognized as the kernel of the inverted (scaled) chi-squared distribution with a degree of freedom $n - 1$ and the scale parameter s^2 :

$$\sigma^2 | \mathbf{Y} = \mathbf{y} \sim \chi^{-2}(n - 1, s^2).$$

We can also examine these marginal posteriors we just derived for the parameters μ and σ^2 visually. In the following are the joint posteriors with a simulated data from $N(0, 1)$, and the corresponding marginal posteriors for the parameters, first with 2, and then with 10 observations:

```

dnonstandard_t <- function(x, df, mu, sigma_squared) {
  gamma((df + 1) / 2) / (gamma(df / 2) * sqrt(df * pi * sigma_squared)) *
  (1 + 1 / df * (x - mu)^2 / sigma_squared)^(-(df + 1) / 2)
}

dinverted_chisq <- function(x, df, sigma_0_squared) {
  ifelse(x > 0, (df / 2)^(df / 2) / gamma(df / 2) * sigma_0_squared^(df / 2) *
  x^(-(df / 2 + 1)) * exp(- df * sigma_0_squared / (2 * x)), 0)
}

n_stops <- c(2, 10)
par(mfrow = c(3, 2), mar = c(4, 3, 3, 0), cex.lab = 1.5, cex.axis = 1.5,
    cex.sub = 1.5, cex.main = 1.5)

for(n in n_stops) {
  y_crnt <- y[1:n]
  persp_NI(m_0 = mean(y_crnt), kappa_0 = n, nu_0 = n - 1, sigma_squared_0 = var(y_crnt),
    main = paste('n =', n))
}

mu <- seq(-3, 3, by = .01)
for(n in n_stops) {
  y_crnt <- y[1:n]
  plot(x, dnonstandard_t(mu, n-1, mean(y_crnt), var(y_crnt) / n),
    type = 'l', bty = 'n', col = 'darkgreen', lwd = 2, xlab = 'mean', ylab = '')
  legend('topright', legend = paste0('t(', round(mean(y_crnt), 3),
    ', ', round(var(y_crnt) / n, 3), ')\nwith df ', n-1),
    col = 'darkgreen', lwd = 2, bty = 'n', cex = 1.3)
}

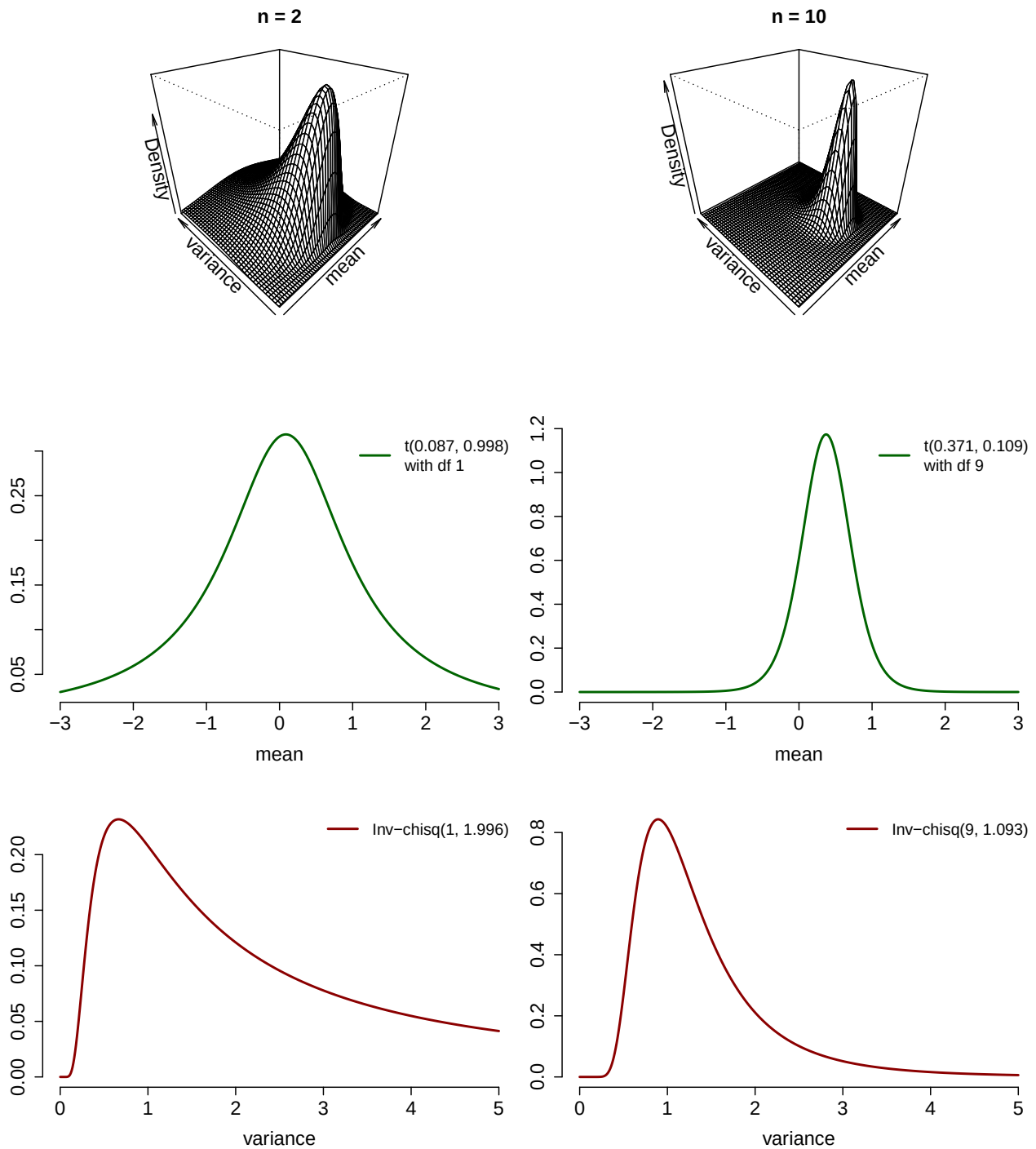
sigma_grid <- seq(0, 5, by = .01)
for(n in n_stops) {
  y_crnt <- y[1:n]

```

```

plot(sigma_grid, dinverted_chisq(sigma_grid, n-1, var(y_crnt)), ylab = '',
     type = 'l', bty = 'n', col = 'darkred', lwd = 2, xlab = 'variance')
legend('topright', legend = paste0('Inv-chisq(', n-1, ', ', ' ',
     round(var(y_crnt), 3), ')'), col = 'darkred', lwd = 2, bty = 'n', cex = 1.3)
}

```



Chapter 6

Hierarchical models

Often observations have some kind of a natural hierarchy, so that the single observations can be modelled belonging into different groups, which can also be modeled as being members of the common supergroup, and so on. For instance, the results of the survey may be grouped at the country, county, town or even neighborhood level. This kind of the spatial hierarchy is the most concrete example of the hierarchy structure, but for example different clinical experiments on the effect of the same drug can be also modeled hierarchically: the results of each test subject belong to the one of the experiments (=groups), and these groups can be modeled as a sample from the common population distribution. This kind of the combining of results of the different studies on the same topic is called **meta-analysis**.

Often the observations inside one group can be modeled as independent: for instance, the results of the test subjects of the randomized experiments, or responses of the survey participant chosen by the random sampling can be reasonably thought to be independent. On the other hand, the parameters of the groups, for example mean response of the test subjects to the same drug in the different clinical experiments, can hardly be thought as independent. However, because the experimental conditions, for example the age or other attributes of the test subjects, length of the experiment and so on, are likely to affect the results, it also does not feel right to assume there are no differences at all between the groups by pooling all the observations together.

The idea of the hierarchical modeling is to **use the data** to model the strength of the dependency between the groups. The groups are assumed to be a sample from the underlying **population distribution**, and the variance of this population distribution, which is estimated from the data, determines how much the parameters of the sampling distribution are shrunk towards the common mean.

First we will take a look at the general form of the two-level hierarchical model, and then make the discussion more concrete by carefully examining a classical example of the hierarchical model.

6.1 Two-level hierarchical model

The most basic two-level hierarchical model, where we have J groups, and $n_1, \dots, n_J$ observations from each of the groups, can be written as

$$\begin{aligned} Y_{ij} | \boldsymbol{\theta}_j &\sim p(y_{ij} | \boldsymbol{\theta}_j) \quad \text{for all } i = 1, \dots, n_j \\ \boldsymbol{\theta}_j | \boldsymbol{\phi} &\sim p(\boldsymbol{\theta}_j | \boldsymbol{\phi}) \quad \text{for all } j = 1, \dots, J. \end{aligned}$$

for each of the $j = 1, \dots, J$ groups.

We assume that the observations $Y_{1j}, \dots, Y_{n_j j}$ within each group are i.i.d., so that the joint **sampling distribution** can be written as a product of the sampling distributions of the single observations (which

were assumed to be the same):

$$p(\mathbf{y}_j|\boldsymbol{\theta}_j) = \prod_{i=1}^{n_j} p(y_{ij}|\boldsymbol{\theta}_j).$$

Group-level parameters $(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_J)$ are then modeled as an i.i.d. sample from the common **population distribution** $p(\boldsymbol{\theta}_j|\phi)$ so that their joint distribution can also be factorized as:

$$p(\boldsymbol{\theta}|\phi) = \prod_{j=1}^J p(\boldsymbol{\theta}_j|\phi).$$

The full model specification depends on how we handle the hyperparameters. We will introduce three options:

1. fix them to some constant values,
2. use a point estimates estimated from the data or
3. set a probability distribution over them.

When we speak about the Bayesian hierarchical models, we usually mean the third option, which means specifying the fully Bayesian model by setting the prior also for the hyperparameters.

6.1.1 No-pooling model

If we just fix the hyperparameters to some fixed value $\phi = \phi_0$, then the posterior distribution for the parameters $\boldsymbol{\theta}$ simply factorizes to J components:

$$p(\boldsymbol{\theta}|\mathbf{y}) \propto p(\boldsymbol{\theta}|\phi_0)p(\mathbf{y}|\boldsymbol{\theta}) = \prod_{j=1}^J p(\boldsymbol{\theta}_j|\phi_0)p(\mathbf{y}_j|\boldsymbol{\theta}_j),$$

because the prior distributions $p(\boldsymbol{\theta}_j|\phi_0)$ were assumed as independent (we could also have removed the conditioning on the ϕ_0 from the notation, because the hyperparameters are not assumed to be random variables in this model). Now all J components of the posterior distribution can be estimated separately; this means that we assume that we do not model any dependency between the group-level parameters $\boldsymbol{\theta}_j$ (except for the common fixed prior distribution).

This option means specifying the non-hierarchical model by assuming the group-level parameters independent. It is prone to overfitting, especially if there is only little data on some of the groups, because it does not allow us to “borrow statistical strength” for these groups with less data from the other more data-heavy groups.

6.1.2 Empirical Bayes

The no-pooling model fixes the hyperparameters so that no information flows through them. However, we can also avoid setting any distribution hyperparameters, while still letting the data dictate the strength of the dependency between the group-level parameters. This is done by approximating the hyperparameters by the point estimates, more specifically fixing them to their maximum likelihood estimates, which are estimated from the marginal likelihood of the data $p(\mathbf{y}|\phi)$:

$$\hat{\phi}_{\text{MLE}}(\mathbf{y}) = \underset{\phi}{\operatorname{argmax}} p(\mathbf{y}|\phi) = \underset{\phi}{\operatorname{argmax}} \int p(\mathbf{y}_j|\boldsymbol{\theta})p(\boldsymbol{\theta}|\phi) d\boldsymbol{\theta}.$$

This is why we computed the maximum likelihood estimate of the beta-binomial distribution in Problem 4 of Exercise set 3 (the problem of estimating the proportions of very liberals in each of the states): the marginal likelihood of the binomial distribution with beta prior is beta-binomial, and we wanted to find out maximum likelihood estimates of the hyperparameters to apply the empirical Bayes procedure.

When the hyperparameters are fixed, we can factorize the posterior as in the no-pooling model:

$$p(\boldsymbol{\theta}|\mathbf{y}) \propto p(\boldsymbol{\theta}|\boldsymbol{\phi}_{\text{MLE}})p(\mathbf{y}|\boldsymbol{\theta}) = \prod_{j=1}^J p(\boldsymbol{\theta}_j|\boldsymbol{\phi}_{\text{MLE}})p(\mathbf{y}_j|\boldsymbol{\theta}_j),$$

and compute the posterior for each of the J components separately. This is why we could compute the posteriors for the proportions of very liberals separately for each of the states in the exercises.

Note that despite of the name, the empirical Bayes is not a Bayesian procedure, because the maximum likelihood estimate is used. It is also a little bit of the “double counting”, because the data is first used to estimate the parameters of the prior distribution, and then this prior and the data are used to compute the posterior for the group-level parameters. However, the empirical Bayes approach can be seen as a computationally convenient approximation of the fully Bayesian model, because it avoids integrating over the hyperparameters. Also, often point estimates may be substituted for some of the parameters in the otherwise Bayesian model. We will actually do this for the within-group variances in our example of the hierarchical model.

6.1.3 Fully Bayesian model

To specify the fully Bayesian model, we set a prior distribution also for the hyperparameters, so that the full model becomes:

$$\begin{aligned} Y_{ij} | \boldsymbol{\theta}_j &\sim p(y_{ij}|\boldsymbol{\theta}_j) \quad \text{for all } i = 1, \dots, n_j \\ \boldsymbol{\theta}_j | \boldsymbol{\phi} &\sim p(\boldsymbol{\theta}_j|\boldsymbol{\phi}) \quad \text{for all } j = 1, \dots, J \\ \boldsymbol{\phi} &\sim p(\boldsymbol{\phi}). \end{aligned}$$

We have already explicitly made the following conditional independence assumptions:

$$\begin{aligned} Y_{11}, \dots, Y_{n_1 1}, \dots, Y_{1J}, \dots, Y_{n_J J} &\perp\!\!\!\perp \boldsymbol{\theta} \\ \boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_J &\perp\!\!\!\perp \boldsymbol{\phi}, \end{aligned}$$

but the crucial implicit conditional independence assumption of the hierarchical model is that the data depends on the hyperparameters only through the population level parameters:

$$\mathbf{Y} \perp\!\!\!\perp \boldsymbol{\phi} | \boldsymbol{\theta}$$

This means that the sampling distribution of the observations given the populations parameters simplifies to

$$p(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\phi}) = p(\mathbf{y}|\boldsymbol{\theta}),$$

and thus the **full posterior** over the parameters can be written using the Bayes formula:

$$\begin{aligned} p(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{y}) &\propto p(\boldsymbol{\theta}, \boldsymbol{\phi})p(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\phi}) \\ &= p(\boldsymbol{\phi})p(\boldsymbol{\theta}|\boldsymbol{\phi})p(\mathbf{y}|\boldsymbol{\theta}) \\ &= p(\boldsymbol{\phi}) \prod_{j=1}^J p(\boldsymbol{\theta}_j|\boldsymbol{\phi})p(\mathbf{y}_j|\boldsymbol{\theta}_j). \end{aligned}$$

Because now the full posterior does not factorize anymore, we cannot solve the marginal posteriors of the group-level parameters $p(\boldsymbol{\theta}_j|\mathbf{y})$ independently, and thus the whole model cannot be solved analytically. However, in the case of conditional conjugacy (which we will consider in the next section), we can mix simulation and techniques for multi-parameter inference from Chapter 5 to derive the marginal posteriors.

Because the empirical Bayes approximates the marginal posterior of the group-level parameters by plugging in the point estimates of the hyperparameters to the conditional posterior of the group-level parameters given the hyperparameters:

$$p(\boldsymbol{\theta}|\mathbf{y}) \approx p(\boldsymbol{\theta}|\hat{\boldsymbol{\phi}}_{\text{MLE}}, \mathbf{y}),$$

it underestimates the uncertainty coming from estimating the hyperparameters. In the fully Bayesian approach the marginal posterior of the group-level parameters is obtained by integrating the conditional posterior distribution of the group-level parameters over the whole marginal posterior distribution of the hyperparameters (i.e. by taking the expected value of the conditional posterior distribution of the group-level parameters over the marginal posterior distribution of the hyperparameters):

$$p(\boldsymbol{\theta}|\mathbf{y}) = \int p(\boldsymbol{\theta}, \boldsymbol{\phi}|\mathbf{y}) d\boldsymbol{\phi} = \int p(\boldsymbol{\theta}|\boldsymbol{\phi}, \mathbf{y})p(\boldsymbol{\phi}|\mathbf{y}) d\boldsymbol{\phi}.$$

This means that the fully Bayesian model properly takes into account the uncertainty about the hyperparameter values by averaging over their posterior.

In principle, this difference between the empirical Bayes and the full Bayes is the same as the difference between using the sampling distribution with a plug-in point estimate $p(\tilde{\mathbf{y}}|\hat{\boldsymbol{\theta}}_{\text{MLE}})$ and using the full proper posterior predictive distribution $p(\tilde{\mathbf{y}}|\mathbf{y})$, which is derived by integrating the sampling distribution over the posterior distribution of the parameter, for predicting the new observations. In Murphy's (Murphy, 2012) book there is a nice quote stating that "the more we integrate, the more Bayesian we are..."

6.2 Conditional conjugacy

If the population distribution $p(\boldsymbol{\theta}|\boldsymbol{\phi})$ is a conjugate distribution for the sampling distribution $p(\mathbf{y}|\boldsymbol{\theta})$, then we talk about the **conditional conjugacy**, because the conditional posterior distribution of the population parameters given the hyperparameters $p(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{\phi})$ can be solved analytically¹. Then simulating from the marginal posterior distribution of the hyperparameters $p(\boldsymbol{\phi}|\mathbf{y})$ is usually a simple matter.

In the following example we could have utilized the conditional conjugacy, because the sampling distribution is a normal distribution with a fixed variance, and the population distribution is also a normal distribution. However, we take a fully simulational approach by directly generating a sample $(\boldsymbol{\phi}^{(1)}, \boldsymbol{\theta}^{(1)}), \dots, (\boldsymbol{\phi}^{(S)}, \boldsymbol{\theta}^{(S)})$ from the full posterior $p(\boldsymbol{\theta}, \boldsymbol{\phi}|\mathbf{y})$. Then the components $\boldsymbol{\phi}^{(1)}, \dots, \boldsymbol{\phi}^{(S)}$ can be used as a sample from the marginal posterior $p(\boldsymbol{\phi}|\mathbf{y})$, and the components $\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(S)}$ can be used as a sample from the marginal posterior $p(\boldsymbol{\theta}|\mathbf{y})$.

The downside of this approach is that the amount of time to compile the model and to sample from it using Stan is orders of magnitudes greater than the time it would take to generate a sample from the posterior utilizing the conditional conjugacy. However, it takes only few minutes to write the model into Stan, whereas solving the part of the posterior analytically, and implementing a sampler for the rest would take a considerably longer time for us to do. So it is a trade-off between the human and the computing effort, and this time we decide to delegate the job to the computer.

6.3 Hierarchical model example

We will consider a classical example of a Bayesian hierarchical model taken from the red book (Gelman et al., 2013). The problem is to estimate the effectiveness of training programs different schools have for preparing their students for a SAT-V (scholastic aptitude test - verbal) test. SAT is designed to test the knowledge that students have accumulated during their years at school, and the test scores should not be affected by short term training programs. Nevertheless, each of the eight schools claim that their training program increases the SAT scores of the students, and we want to find out what are the real effects of these training programs. The data are not the raw scores of the students, but the training effects estimated on the basis of the preliminary SAT tests and SAT-M (scholastic aptitude test - mathematics) taken by the same students. You can read more about the experimental set-up from the section 5.5 of (Gelman et al., 2013).

¹This is why we chose the beta prior for the binomial likelihood in Problem 4 of Exercise set 3, in which we estimated the proportions of the very liberals in each of the states.

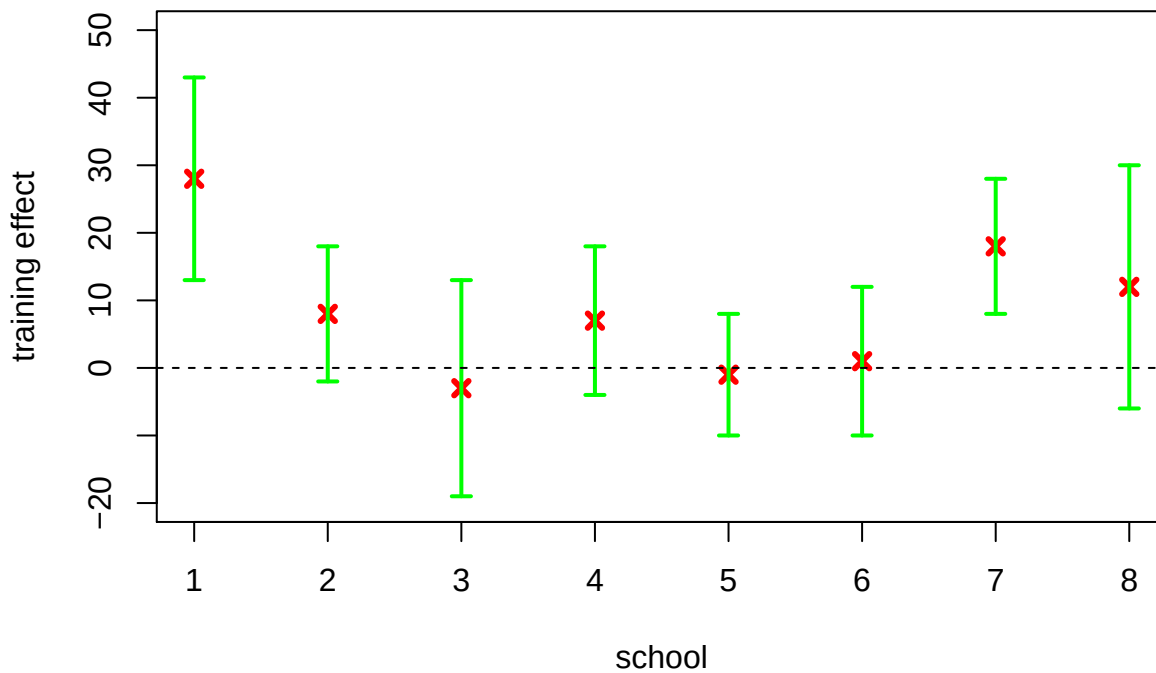
So there are in total $J = 8$ schools (=groups); in each of these schools we denote observed training effects of the students as $Y_{1j}, \dots, Y_{n_jj}$. We will use the point estimates for the standard deviations $\hat{\sigma}_j^2$ for each of the schools².

Let's first take a look at the raw data by plotting the observed training effects for each of the schools along with their standard errors, which we assume as known:

```
schools <- list(J = 8, y = c(28, 8, -3, 7, -1, 1, 18, 12),
               sigma = c(15, 10, 16, 11, 9, 11, 10, 18))

plot(schools$y, pch = 4, col = 'red', lwd = 3, ylim = c(-20, 50),
     ylab = 'training effect', xlab = 'school', main = 'Observed training effects')
arrows(1:8, schools$y-schools$sigma, 1:8, schools$y+schools$sigma,
      length=0.05, angle=90, code=3, col = 'green', lwd = 2)
abline(h = 0, lty = 2)
```

Observed training effects



There are clear differences between the schools: for one school the observed training effect is as high as 28 points (normally the test scores are between 200 and 800 with mean of roughly 500 and standard deviation about 100), while for two schools the observed effect is slightly negative. However, the standard errors are also high, and there is substantial overlap between the schools.

Because there are relatively many (> 30) test subjects in each of the schools, we can use the normal approximation for the distribution of the test scores within one school, so that the mean improvement in the training scores can modeled as:

$$\frac{1}{n_j} \sum_{i=1}^{n_j} Y_{ij} \sim N \left(\theta_j, \frac{\hat{\sigma}_j^2}{n_j} \right).$$

for each of the $j = 1, \dots, J$ schools.

²Actually this assumption was made to simplify the analytical computations. Since we are using probabilistic programming tools to fit the model, this assumption is no longer necessary. But because we do not have the original data, and it this simplifying assumption likely have very little effect on the results, we will stick to it anyway.

To simplify the notation, let's denote these group means as $Y_j := \frac{1}{n_j} \sum_{i=1}^{n_j} Y_{ij}$, and the group standard deviations as $\sigma_j^2 := \hat{\sigma}_j^2/n$. Because mean is a sufficient statistic for a normal distribution with a known variance, we can model the sampling distribution with only one observation from each of the schools:

$$Y_j | \theta_j \sim N(\theta_j, \sigma_j^2) \quad \text{for all } j = 1, \dots, J$$

using the notation defined above.

Furthermore, we assume that the true training effects $\theta_1, \dots, \theta_J$ for each school are a sample from the common normal distribution³:

$$\theta_j | \mu, \tau^2 \sim N(\mu, \tau^2) \quad \text{for all } j = 1, \dots, J.$$

However, before specifying the full hierarchical model, let's first examine two simpler ways to model the data.

6.3.1 No-pooling model

Probably the simplest thing to do would be to assume the true training effects θ_j as independent, and use a noninformative improper prior for them:

$$\begin{aligned} Y_j | \theta_j &\sim N(\theta_j, \sigma_j^2) \\ p(\theta_j) &\propto 1 \quad \text{for all } j = 1, \dots, J. \end{aligned}$$

Now the joint posterior factorizes:

$$p(\boldsymbol{\theta} | \mathbf{y}) \propto 1 \cdot \prod_{j=1}^J p(y_j | \theta_j),$$

which means that the posteriors for the true training effects can be estimated separately for each of the schools:

$$\theta_j | \mathbf{Y} = \mathbf{y} \sim N(y_j, \sigma_j^2) \quad \text{for all } j = 1, \dots, J.$$

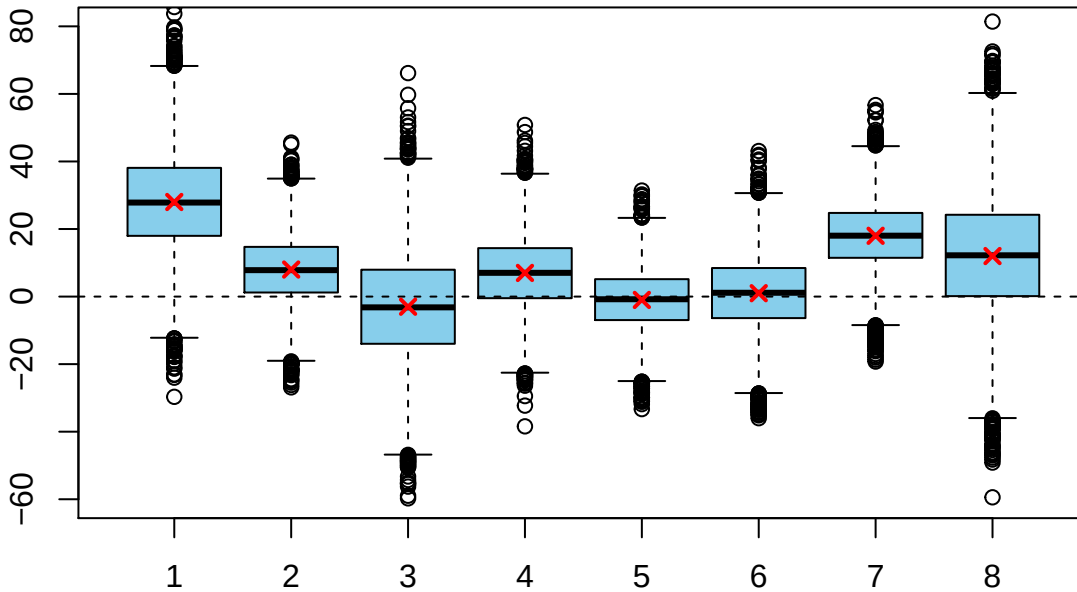
We have solved the posterior analytically, but let's also sample from it to draw a boxplot similar to the ones we will produce for the fully hierarchical model:

```
set.seed(123)
n_sim <- 1e4
theta <- matrix(numeric(n_sim * schools$J), ncol = schools$J)
for(j in 1:schools$J)
  theta[, j] <- rnorm(n_sim, schools$y[j], schools$sigma[j])

boxplot(theta, col = 'skyblue', ylim = c(-60, 80), main = 'No pooling model')
abline(h = 0, lty = 2)
points(schools$y, col = 'red', lwd=2, pch=4)
```

³By using the normal population distribution the model becomes conditionally conjugate. Now that we are using Stan to fit the model, also this assumption is no longer necessary.

No pooling model



The observed training effects are marked into the figure with red crosses. Because we use a non-informative prior, posterior modes are equal to the observed mean effects. It seems that by using the separate parameter for each of the schools without any smoothing we are most likely overfitting (we will actually see if this is the case at the next week!). Notice that if we used a noninformative prior, there actually would be some smoothing, but it would have been into the direction of the mean of the arbitrarily chosen prior distribution, not towards the common mean of the observations. Setting the arbitrary noninformative prior would make very little sense here, because we can actually use the values of the other groups to infer the parameters of this prior distribution (which is called a population distribution in the full hierarchical model).

6.3.2 Complete pooling model

But before we examine the full hierarchical distribution, let's try another simplified model. In the so-called complete pooling model we make an apriori assumption that there are no differences between the means of the schools (and probably the standard deviations are also the same; different observed standard deviations are due to different sample sizes and random variance), so that we need only single parameter θ , which presents the true training effect for all of the schools. Let's use a noninformative improper prior again:

$$Y_j | \theta \sim N(\theta, \sigma_j^2) \quad \text{for all } j = 1, \dots, J$$

$$p(\theta) \propto 1.$$

We have $J = 8$ observations from the normal distributions with the same mean and different, but known variances. We can derive the posterior for the common true training effect θ with a computation almost identical to one performed in Example 5.2.1, in which we derived a posterior for one observation from the normal distribution with known variance:

$$p(\theta | \mathbf{y}) = N \left(\frac{\sum_{j=1}^J \frac{1}{\sigma_j^2} y_j}{\sum_{j=1}^J \frac{1}{\sigma_j^2}}, \frac{1}{\sum_{j=1}^J \frac{1}{\sigma_j^2}} \right)$$

The posterior distribution is a normal distribution whose precision is the sum of the sampling precisions, and the mean is a weighted mean of the observations, where the weights are given by the sampling precisions.

Let's simulate also from this model, and then draw again a boxplot (which is little bit stupid, because exactly the same posterior is drawn eight times, but this is just for the illustration purposes):

```

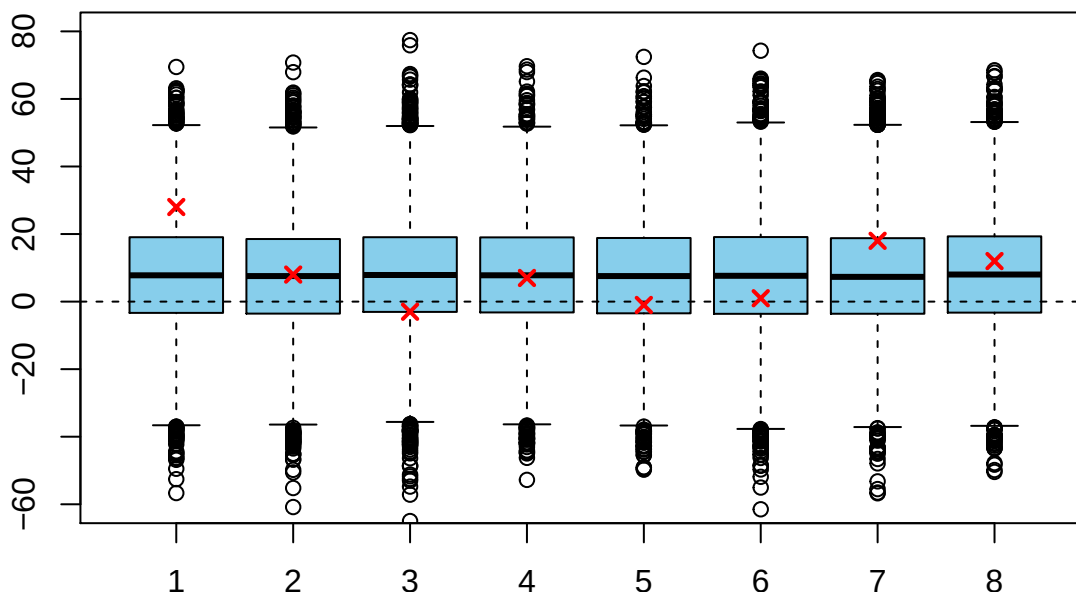
pooled_variance <- 1 / sum(1 / schools$sigma^2)
grand_mean <- pooled_variance * sum(schools$y / schools$sigma^2)

theta <- matrix(rnorm(n_sim * schools$J, grand_mean, pooled_variance),
               ncol = schools$J)

boxplot(theta, col = 'skyblue', ylim = c(-60, 80), main = 'Complete pooling')
abline(h = 0, lty = 2)
points(schools$y, col = 'red', lwd=2, pch=4)

```

Complete pooling



6.3.3 Bayesian hierarchical model

Because the simplifying assumptions of the previous two models do not feel very realistic, let's also fit a fully Bayesian hierarchical model. To do so we also have to specify a prior to the parameters μ and τ of the population distribution. It turns out that the improper noninformative prior

$$p(\mu, \tau^2) \propto (\tau^2)^{-1}, \tau > 0$$

that was used for the normal distribution in Section 5.3 does not actually lead to a proper posterior with this model: with this prior the integral of the unnormalized posterior diverges, so that it cannot be normalized into a probability distribution! However, it turns out that using a completely flat improper prior for the expected value and the *standard deviation*:

$$p(\mu, \tau) \propto 1, \tau > 0$$

leads to a proper posterior if the number of groups J is at least 3 (proof omitted), so we can specify the model as:

$$\begin{aligned}
 Y_j | \theta_j &\sim N(\theta_j, \sigma_j^2) \\
 \theta_j | \mu, \tau &\sim N(\mu, \tau^2) \quad \text{for all } j = 1, \dots, J \\
 p(\mu, \tau) &\propto 1, \tau > 0.
 \end{aligned}$$

We can translate this model directly into Stan modelling language:

```

data {
  int<lower=0> J;
  real y[J];
  real<lower=0> sigma[J];
}

parameters {
  real mu;
  real<lower=0> tau;
  real theta[J];
}

model {
  theta ~ normal(mu, tau);
  y ~ normal(theta, sigma);
}

```

Notice that we did not explicitly specify any prior for the hyperparameters μ and τ in Stan code: if we do not give any prior for some of the parameters, Stan automatically assign them uniform prior on the interval in which they are defined. In this case this uniform prior is improper, because these intervals are unbounded.

Now we can sample from this model:

```

library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())

```

```
fit3 <- stan('schools1.stan', data = schools, iter = 1e4, chains = 4)
```

```
## Warning: There were 415 divergent transitions after warmup. Increasing adapt_delta above 0.8 may help. See
```

```
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

Hmm... Stan warns that there are some divergent transitions: this indicates that there are some problems with the sampling. Stan suggests increasing the tuning parameter `adapt_delta` from its default value 0.8, so let's try it before looking at any sampling diagnostics. Values of the `adapt_delta` are between 0 and 1, and increasing it should decrease the number of divergent transitions while making the sampler slower. Sampling from this simple model is very fast anyway, so we can increase `adapt_delta` to 0.95. Tuning parameters are given as a named list to the argument `control`:

```
fit3 <- stan('schools1.stan', data = schools, iter = 1e4, chains = 4,
  control = list(adapt_delta = 0.95))
```

```
## Warning: There were 1015 divergent transitions after warmup. Increasing adapt_delta above 0.95 may help. See
```

```
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: There were 2 chains where the estimated Bayesian Fraction of Missing Information was low. See
```

```
## http://mc-stan.org/misc/warnings.html#bfmi-low
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

There are still some divergent transitions, but much less now. If there are lots of divergent transitions, it usually means that the model is specified so that HMC sampling from it is hard⁴, and that the results may be biased because the sampler did not explore the whole area of the posterior distribution efficiently. We will find out later why is it hard for Stan to sample from this model, and how to change the model structure to allow more efficient sampling from the model.

⁴Or it may mean that the model was specified completely wrong: for instance, some of the parameter constraints may be forgotten. This is a first thing that should be checked if there are lots of divergent transitions.

Nevertheless, the proportion of the divergent transitions was not so large when we increased the values of `adapt_delta`, so we are happy with the results for now. Let's look at the summary of the Stan fit:

```
fit3
```

```
## Inference for Stan model: schools1.
## 4 chains, each with iter=10000; warmup=5000; thin=1;
## post-warmup draws per chain=5000, total post-warmup draws=20000.
##
##               mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## mu            8.42    0.27 5.17  -1.87   5.08   8.39 11.97 17.89  368 1.01
## tau           6.23    0.29 5.41   0.51   2.21   4.89   8.67 19.95  358 1.01
## theta[1]     11.62    0.15 7.96  -1.94   6.54  10.90  15.58 30.77 2875 1.00
## theta[2]      8.40    0.27 6.28  -4.53   4.45   8.51  12.74 20.24  539 1.00
## theta[3]      6.82    0.36 7.69 -10.63   2.57   7.41  11.96 19.94  454 1.01
## theta[4]      8.20    0.29 6.53  -5.41   4.17   8.37  12.64 20.45  504 1.01
## theta[5]      5.83    0.40 6.50  -8.45   1.79   6.21  10.35 16.73  258 1.01
## theta[6]      6.76    0.35 6.79  -8.09   2.76   7.14  11.40 18.56  367 1.01
## theta[7]     10.98    0.16 6.65  -1.20   6.65  10.62  14.75 25.69 1765 1.00
## theta[8]      8.90    0.22 7.62  -6.22   4.45   8.81  13.43 25.03 1235 1.00
## lp__         -16.25    0.71 6.82 -27.37 -21.21 -17.39 -11.97 -1.33   91 1.03
##
## Samples were drawn using NUTS(diag_e) at Wed Mar 13 10:42:27 2019.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

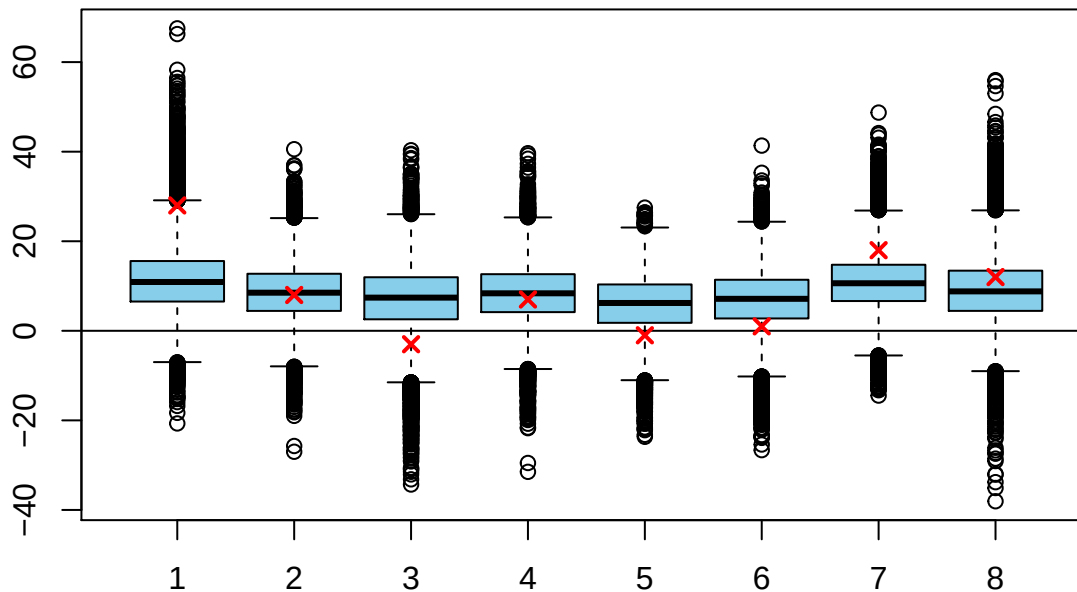
We have a posterior distribution for 10 parameters: expected value of the population distribution μ , standard deviation of the population distribution τ , and the true training effects $\theta_1, \dots, \theta_8$ for each of the schools.

Let's first examine the marginal posterior distributions $p(\theta_1|\mathbf{y}), \dots, p(\theta_8|\mathbf{y})$ of the training effects :

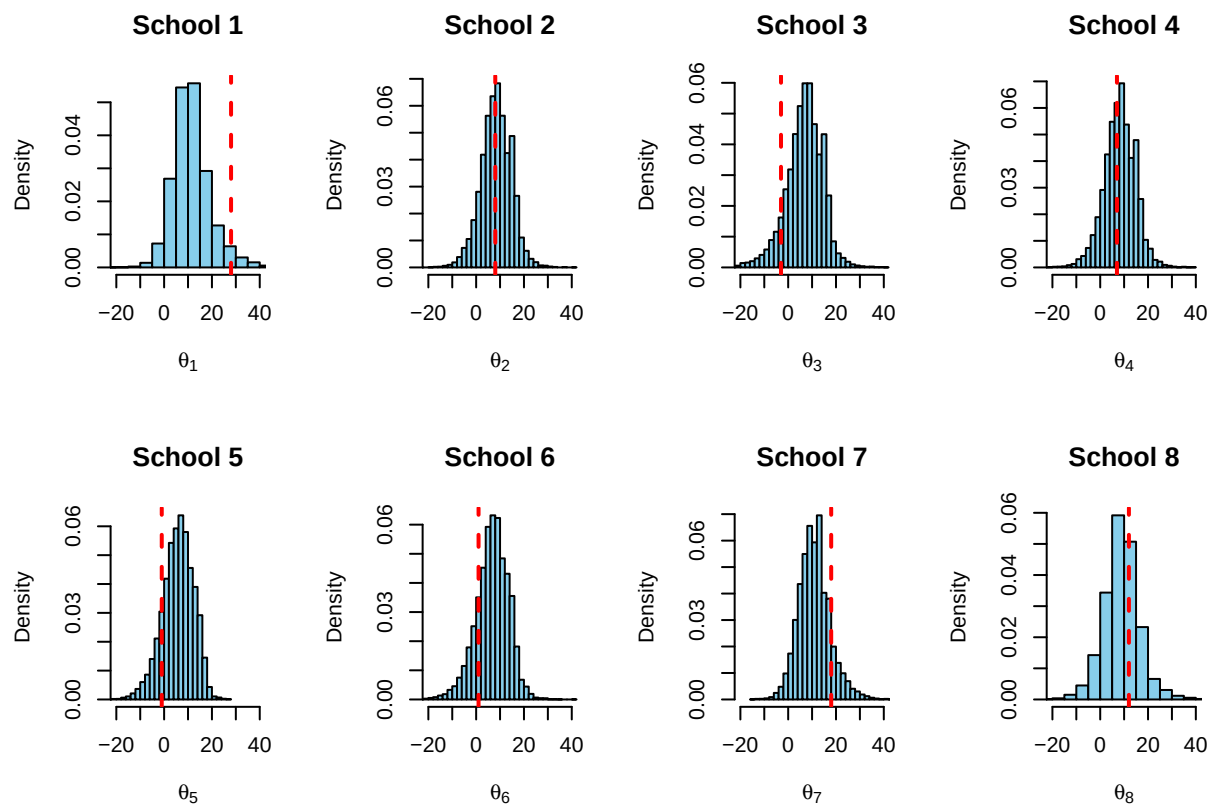
```
sim3 <- extract(fit3)

par(mfrow=c(1,1))
boxplot(sim3$theta, col = 'skyblue', main = 'Hierarchical model')
abline(h=0)
points(schools$y, col = 'red', lwd=2, pch=4)
```

Hierarchical model



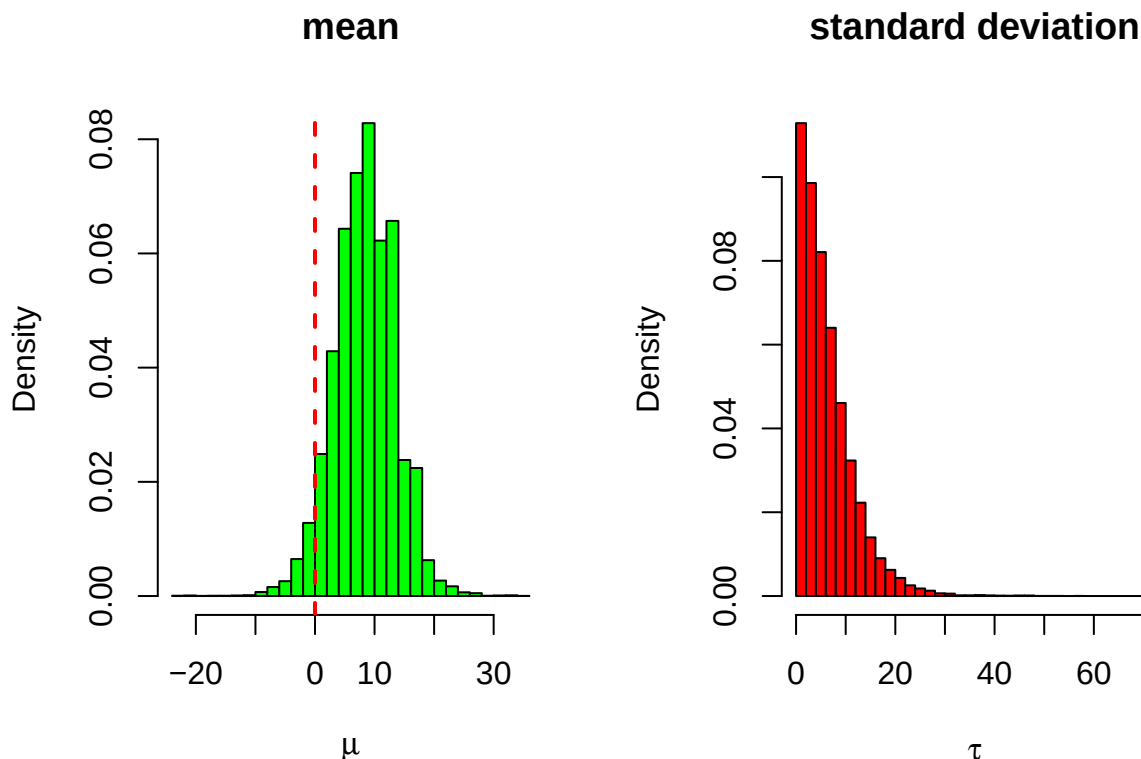
```
par(mfrow=c(2,4))
for(i in 1:8) {
  hist(sim3$theta[,i], col = 'skyblue', main = paste0('School ', i),
       breaks = 30, xlim = c(-20,40), probability = TRUE,
       xlab = bquote(theta[.(i)]))
  abline(v = schools$y[i], lty = 2, lwd = 2, col = 'red')
}
```



The observed training effects $y_1, \dots, y_8$ are marked into the boxplot by red crosses, and into the histograms by the red dashed lines. This time the posterior medians (the center lines of the boxplots) are shrunk towards the common mean.

Let's also take a look at the marginal posteriors of the parameters of the population distribution $p(\mu|\mathbf{y})$ and $p(\tau|\mathbf{y})$:

```
par(mfrow=c(1,2))
hist(sim3$mu, col = 'green', breaks = 30, probability = TRUE,
     main = 'mean', xlab = expression(mu))
abline(v = 0, lty = 2, lwd = 2, col = 'red')
hist(sim3$tau, col = 'red', breaks = 30, probability = TRUE,
     main = 'standard deviation', xlab = expression(tau))
```



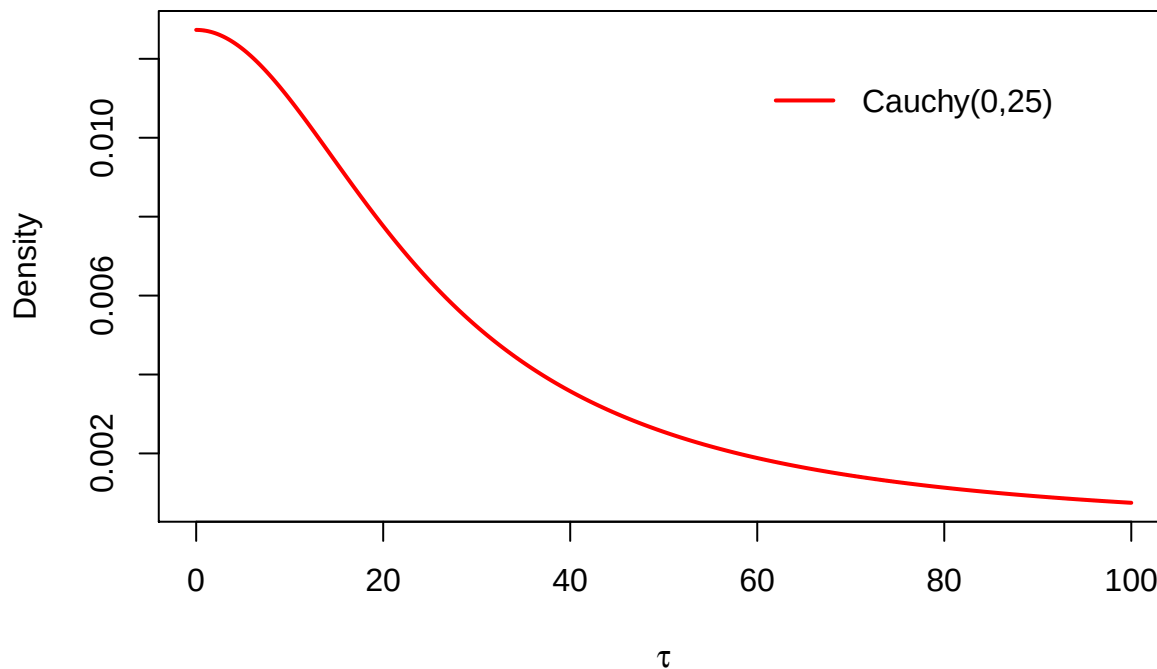
The marginal posterior of the standard deviation is peaked just above the zero. This means that utilizing the empirical Bayes approach here (substituting the posterior mode or the maximum likelihood estimate for the value of τ) in this model would actually lead to radically different results compared to the fully Bayesian approach: because the point estimate $\hat{\tau}$ for the between-groups variance would be zero or almost zero, the empirical Bayes would in principle reduce to the complete pooling model which assumes that there are no differences between the schools!

6.3.4 Hierarchical model with half-cauchy prior

The original improper prior for the standard deviation $p(\tau) \propto 1$ was chosen out of the computational convenience. Because we are using probabilistic programming tools to fit the model, we do not have to care about the conditional conjugacy anymore, and can use any prior we want. A good choice of prior for the group-level scale parameter in the hierarchical models is a distribution which is peaked at zero, but has a long right tail. Let's use the Cauchy distribution $\text{Cauchy}(0, 25)$. The standard deviation of the test scores of the students was around 100, and this could also be thought as an upper limit for the between-the-group variance, so that the realistic interval for τ is $(0, 100)$. Notice the scale of the y -axis: this distribution is super flat, but still almost all of its probability mass lies on the interval $(0, 100)$. This kind of a relatively flat

prior, which is concentrated on the range of the realistic values for the current problem is called a **weakly informative prior**:

```
x <- seq(0,100, by = .01)
plot(x, dcauchy(x,0,25), type = 'l', col = 'red', lwd = 2,
      xlab = expression(tau), ylab = 'Density')
legend('topright', 'Cauchy(0,25)', col = 'red', lwd = 2, inset = .1, bty = 'n')
```



Now the full model is:

$$\begin{aligned}
 Y_j | \theta_j &\sim N(\theta_j, \sigma_j^2) \\
 \theta_j | \mu, \tau &\sim N(\mu, \tau^2) \quad \text{for all } j = 1, \dots, J \\
 p(\mu | \tau) &\propto 1, \quad \tau \sim \text{half-Cauchy}(0, 25), \quad \tau > 0.
 \end{aligned}$$

The only thing we have to change in the Stan model is to add the half-cauchy prior for τ :

```
tau ~ cauchy(0,25);
```

Because τ is constrained into the positive real axis, Stan automatically uses half-cauchy distribution, so above sampling statement is sufficient. Now we can save the whole model into the file `schoolsc.stan`:

```
data {
  int<lower=0> J;
  real y[J];
  real<lower=0> sigma[J];
}

parameters {
  real mu;
  real<lower=0> tau;
  real theta[J];
}

model {
  tau ~ cauchy(0,25);
```

```

theta ~ normal(mu, tau);
y ~ normal(theta, sigma);
}

sim4 <- readRDS('sim7.rds')

```

Let's sample from the posterior of this model and examine the results:

```

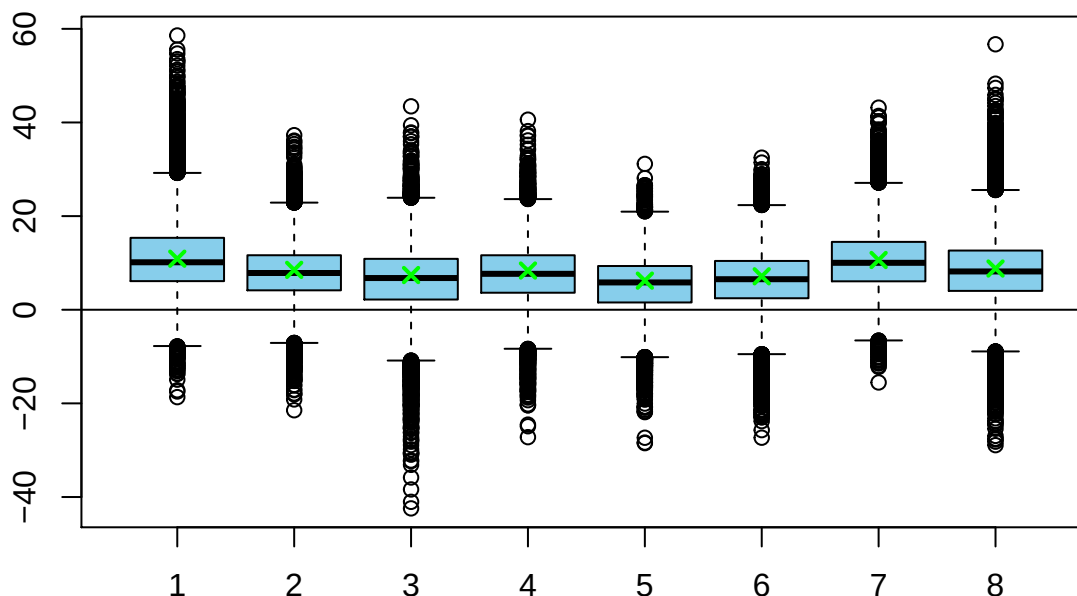
## fit4 <- stan('schoolsc.stan', data = schools, iter = 1e4, control = list(adapt_delta = .95))
## sim4 <- extract(fit4)

par(mfrow=c(1,1))
boxplot(sim4$theta, col = 'skyblue',
        main = 'Hierarchical model with Cauchy prior')
abline(h=0)

# compare to medians of model 3 with improper prior for variance
medians3 <- apply(sim3$theta, 2, median)
points(medians3, pch = 4, lwd=2, col = 'green')

```

Hierarchical model with Cauchy prior

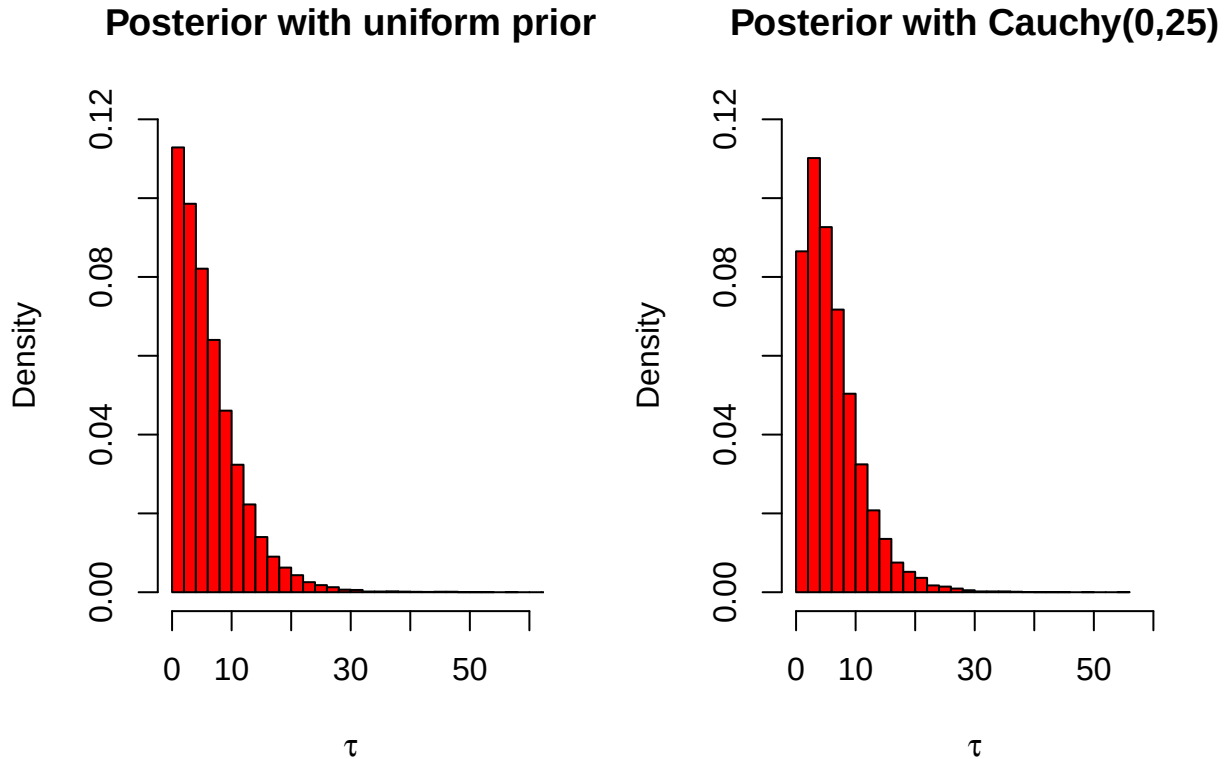


The posterior medians of the hierarchical model are denoted by the green crosses in the boxplot. They match almost exactly the posterior medians for this new model. Let's also compare the posterior distributions for the group-level variance τ :

```

par(mfrow=c(1,2))
hist(sim3$tau, col = 'red', breaks = 30, probability = TRUE,
     main = 'Posterior with uniform prior', xlab = expression(tau),
     ylim = c(0,.12), xlim = c(0,60))
hist(sim4$tau, col = 'red', breaks = 30, probability = TRUE,
     main = 'Posterior with Cauchy(0,25)', xlab = expression(tau),
     ylim = c(0,.12), xlim = c(0,60))

```



The posteriors for the standard deviation are also almost identical. This is a very good thing: if we want to use a relatively noninformative prior, it is useful to try different priors and prior parameters to see how they affect the posterior. If the posterior is relatively robust with respect to the choice prior, then it is likely that the priors tried really were noninformative. On the other hand, if there are substantial differences between the posterior inferences between the different priors, then at least some of the priors tried were not as noninformative as we believed. This kind of testing the effects of different priors on the posterior distribution is called **sensitivity analysis**.

6.3.5 Hierarchical model with inverse gamma prior

To perform little bit more ad-hoc sensitivity analysis, let's test one more prior. The inverse-gamma distribution is a conjugate prior for the variance of the normal distribution⁵, so it is a natural choice for a prior. A traditional noninformative, but proper, prior for used for nonhierarchical models is $\text{Inv-gamma}(\epsilon, \epsilon)$ with some small value of ϵ ; let's use a smallish value $\epsilon = 1$ for the illustration purposes. With this prior the full model is:

$$\begin{aligned} Y_j | \theta_j &\sim N(\theta_j, \sigma_j^2) \\ \theta_j | \mu, \tau &\sim N(\mu, \tau^2) \quad \text{for all } j = 1, \dots, J \\ p(\mu | \tau) &\propto 1, \quad \tau^2 \sim \text{Inv-gamma}(1, 1). \end{aligned}$$

Notice that we set a prior for the variance τ^2 of the population distribution instead of the standard deviation τ . Because of this we declare the variable `tau_squared` instead of `tau` in the `parameters`-block, and declare `tau` as a square root of `tau_squared` in the `transformed parameters`-block:

```
data {
  int<lower=0> J;
  real y[J];
```

⁵Remember that the inverse scaled chi squared distribution we used is just an inverse-gamma distribution with a convenient reparametrization.

```

    real<lower=0> sigma[J];
  }

  parameters {
    real theta[J];
    real mu;
    real<lower=0> tau_squared;
  }

  transformed parameters {
    real<lower=0> tau = sqrt(tau_squared);
  }

  model {
    tau_squared ~ inv_gamma(1,1);
    y ~ normal(theta, sigma);
    theta ~ normal(mu, tau);
  }

```

and then sample from this model:

```

fit7 <- stan('schoolsig.stan', data = schools, iter = 1e4,
            control = list(adapt_delta = .95))

```

```

## Warning: There were 49 divergent transitions after warmup. Increasing adapt_delta above 0.95 may help. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

```

```

## Warning: Examine the pairs() plot to diagnose sampling problems

```

```

sim7 <- extract(fit7)

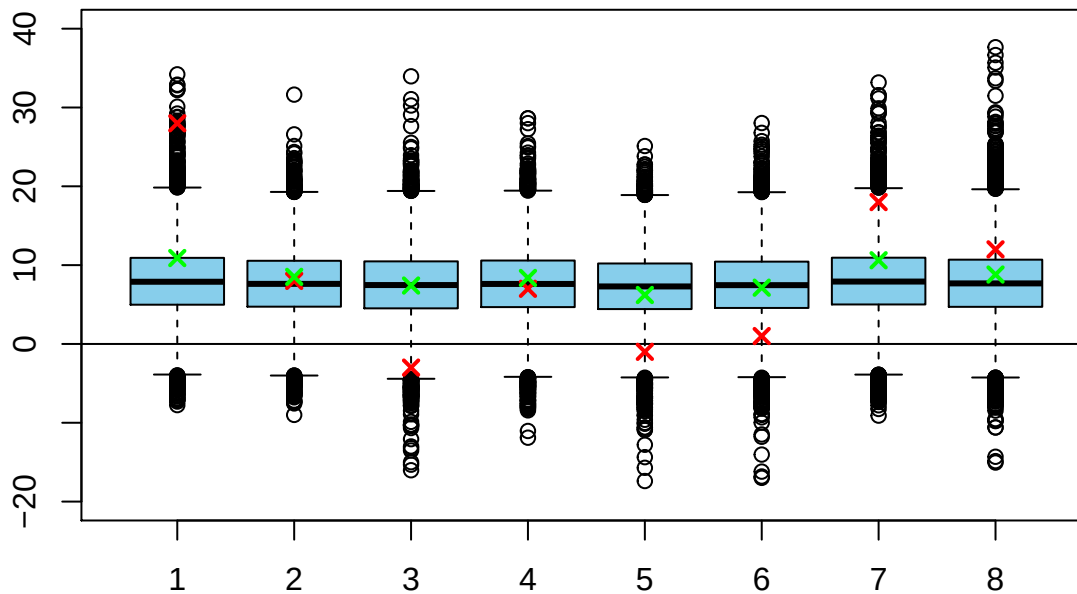
```

Let's compare the marginal posterior distributions for each of the schools to the posteriors computed from the hierarchical model with the uniform prior (posterior medians from the model with the uniform prior are marked by green crosses):

```

par(mfrow=c(1,1))
boxplot(sim7$theta, col = 'skyblue', ylim = c(-20, 40))
abline(h=0)
points(schools$y, col = 'red', lwd=2, pch=4)
points(medians3, pch = 4, lwd=2, col = 'green')

```

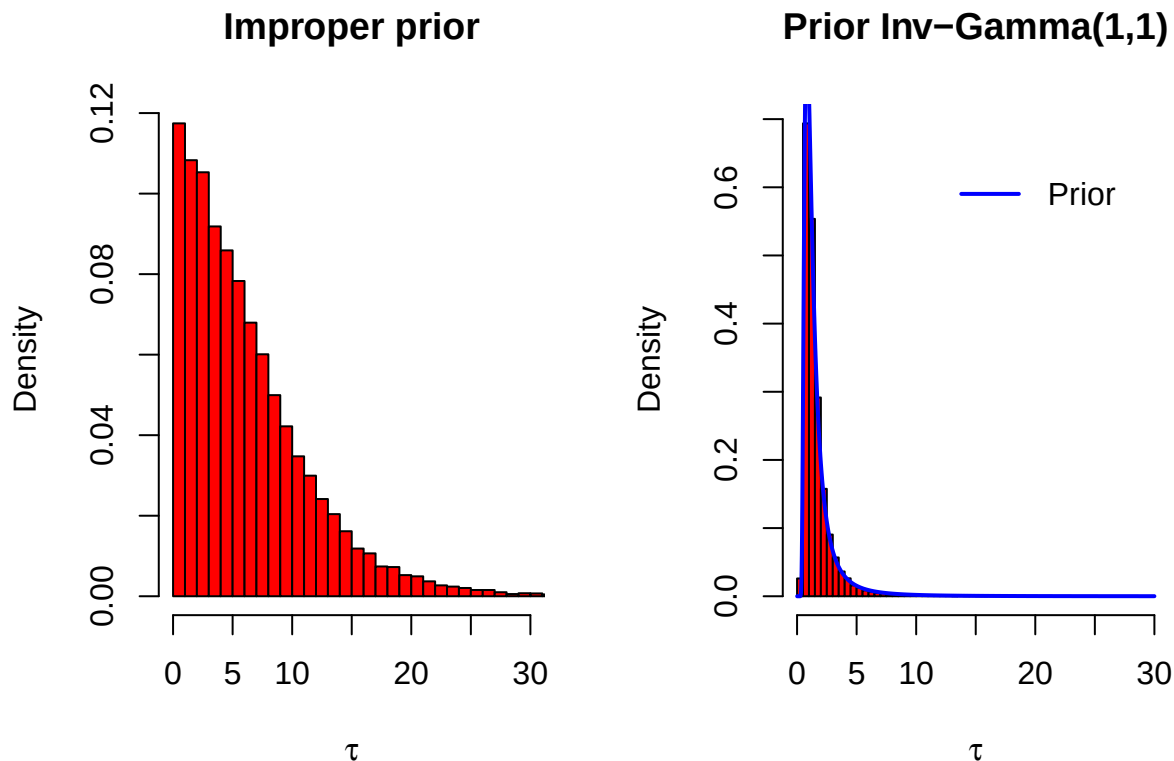


Now the model shrinks the training effects for each of the schools much more! It is almost identical to the complete pooling model. To see why, let's take a look at the posterior variances:

```
par(mfrow=c(1,2))
hist(sim3$tau, col = 'red', breaks = 50, probability = TRUE,
     main = 'Improper prior', xlim = c(0,30), xlab = expression(tau))
hist(sim7$tau, col = 'red', breaks = 50, probability = TRUE,
     main = 'Prior Inv-Gamma(1,1)', xlim = c(0,30), xlab = expression(tau))

# multiplied by the jacobian of the inverse transform
dinv_gamma <- function(x,alpha,beta){
  beta^alpha / gamma(alpha) * x^(-2 * (alpha + 1)) * exp(-beta / x^2) * 2 * x
}

x <- seq(0, 30, by=.01)
lines(x, dinv_gamma(x, 1, 1), type = 'l', col = 'blue', lwd = 2)
legend('topright', 'Prior', lwd = 2, col = 'blue', inset = .1, bty = 'n')
```



The prior distribution $\text{Inv-gamma}(1,1)$ (transformed for standard deviation) is drawn on the righthmost picture with a blue line: it seems that the data had almost no effect at all on the posterior of τ . So the prior which we thought would be reasonably noninformative, was actually very strong: it pulled the standard deviation of the population distribution to almost zero! This is why performing the sensitivity analysis is important.

Chapter 7

Linear model

So far on this course we have examined models with no predictors. However, usually the modeling situation is that have the observations $Y_1, \dots, Y_n$, often called **response variable** or **output variable**, and for each observation Y_i we have the vector of **predictors** $\mathbf{x}_i = (x_{i1}, \dots, x_{ik})$, which we use to predict its value.

We are interested in values of the response variable given the predictors, so they we can think the values of the predictors as constants, i.e. we do not have to set any prior for the them.

Liner models and generalized linear model are one of the most important tools of applied statistician. In principle the inference does not differ from the computations we have done earlier on this course. We have already examined the posterior inference for the normal distribution, on which the linear models are based on. However, usually on linear models we have multiple predictors: this means that the posterior for the regression coefficients is a multinormal distribution. This complicates the things a little bit, but the principle stays the same.

We can collect the values of the predicted variable $\mathbf{Y} = (Y_1, \dots, Y_n)$ into the $n \times 1$ -matrix

$$\mathbf{Y} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix},$$

and the values of the predictors into the $n \times k$ -matrix

$$\mathbf{X} = \begin{bmatrix} x_{11} & \dots & x_{1k} \\ \vdots & & \vdots \\ x_{n1} & \dots & x_{nk} \end{bmatrix},$$

so that we can use a convenient matrix notation for the linear model. Usually we also want to add a constant term into the model. This can be incorporated into the vector notation by setting the first column of the matrix of the predictors into the vector of ones: $(x_{11}, \dots, x_{n1}) = \mathbf{1}_n$. The regression coefficients can be written into the $k \times 1$ -matrix

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_k \end{bmatrix},$$

where β_1 is the intercept of the model (if the constant term is used).

7.1 Classical linear model

In the classical linear model, also known as **ordinary least squares regression**, it is assumed that the response variables are independent, and follow normal distributions given the values of the predictors, and

that the expected values of these normal distributions are linear combinations of the regression coefficients β :

$$E[Y_i | \beta, \mathbf{x}_i] = \mathbf{x}_i^T \beta = x_{i1}\beta_1 + \cdots + x_{ik}\beta_k,$$

and that these normal distributions have a same variance σ^2 . In the Bayesian setting the noninformative prior for the parameter vector is $p(\beta, \sigma^2) \propto (\sigma^2)^{-1}$. This means that the model can be written as

$$\begin{aligned} Y_i | \beta, \sigma^2 &\sim N(\mathbf{x}_i^T \beta, \sigma^2) \quad \text{for all } i = 1, \dots, n, \\ p(\beta, \sigma^2) &\propto \frac{1}{\sigma^2}, \end{aligned}$$

or more compactly using the matrix notation introduced above as:

$$\begin{aligned} \mathbf{Y} &\sim N(\mathbf{X}\beta, \sigma^2 \mathbf{I}) \\ p(\beta, \sigma^2) &\propto \frac{1}{\sigma^2}. \end{aligned}$$

7.1.1 Posterior for classical linear regression

With derivations similar to the ones done in Section 5.3 we can show that the conditional posterior distribution $p(\beta, |\sigma^2, \mathbf{y})$ of the regression coefficients given the variance is a k -dimensional multinormal distribution

$$\beta | \mathbf{y}, \sigma^2 \sim N(\hat{\beta}, V_\beta \sigma^2),$$

where

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y},$$

and

$$\mathbf{V}_\beta = (\mathbf{X}^T \mathbf{X})^{-1}.$$

The marginal posterior distribution for the variance σ^2 is an inverted chi-squared distribution with degrees of freedom $n - k$:

$$\sigma^2 | \mathbf{y} \sim \chi_{n-k}^{-2}(s^2),$$

where

$$s^2 = \frac{1}{n-k} (\mathbf{y} - \mathbf{X}\hat{\beta})^T (\mathbf{y} - \mathbf{X}\hat{\beta}).$$

We can observe that when the noninformative prior is used, the results are again quite close to the results of the classical statistical inference for the linear model.

Bibliography

- Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W., and Iannone, R. (2018). *rmarkdown: Dynamic Documents for R*. R package version 1.11.
- Bernardo, J. and Smith, A. (1994). *Bayesian Theory*. Wiley Series in Probability & Statistics. Wiley.
- Bernardo, J. M. (1996). The concept of exchangeability and its applications. *Far East Journal of Mathematical Sciences*, 4:111–122.
- Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A., and Rubin, D. (2013). *Bayesian Data Analysis, Third Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis.
- Goodrich, B., Gelman, A., Carpenter, B., Hoffman, M., Lee, D., Betancourt, M., Brubaker, M., Guo, J., Li, P., Riddell, A., Inacio, M., Morris, M., Arnold, J., Goedman, R., Lau, B., Trangucci, R., Gabry, J., Kucukelbir, A., Grant, R., Tran, D., Malecki, M., and Gao, Y. (2019). *StanHeaders: C++ Header Files for Stan*. R package version 2.18.1.
- Guo, J., Gabry, J., and Goodrich, B. (2018). *rstan: R Interface to Stan*. R package version 2.18.2.
- Koistinen, P. (2013). Todennakoisyyslaskenta. <http://wiki.helsinki.fi/pages/viewpage.action?pageId=196948970>.
- Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. Cambridge, MA.
- Nieminen, P. and Pentti, S. (2013). Tilastollinen paattely. <http://wiki.helsinki.fi/pages/viewpage.action?pageId=164335164>.
- R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
- Wickham, H., Chang, W., Henry, L., Pedersen, T. L., Takahashi, K., Wilke, C., and Woo, K. (2018). *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. R package version 3.1.0.
- Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.
- Xie, Y. (2018a). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.9.
- Xie, Y. (2018b). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.21.
- Young, G. and Smith, R. (2005). *Essentials of Statistical Inference*. Cambridge Series in Statistica. Cambridge University Press.