PSYC 573 Bayesian Data Analysis (2024 Fall): Course Notes

Hok Chio (Mark) Lai

2024-09-16

Table of contents

Preface

There will be some math in this notes. Don't worry if you feel the math is challenging; for applied focused students, it is much more important to understand the concepts of Bayesian methods than to understand the mathematical symbols, as they usually can be handled by the software.

Part I

Week 1

1 Introduction

1.1 History of Bayesian Statistics

Here is a nice brief video that covers some of the 250+ years of history of Bayesian statistics:

https://www.youtube.com/watch?v=BcvLAw-JRss

If you are interested in learning more about the story, check out the popular science book, "The theory that would not die," by McGrayne (2011)

1.1.1 Thomas Bayes (1701–1762)

You may find a biography of Bayes from https://www.britannica.com/biography/Thomas-Bayes. There is also a nice story in the book by Lambert (2018). He was an English Presbyterian minister. The important work he wrote that founded Bayesian statistics was "An Essay Towards Solving a Problem in the Doctrine of Chances," which he did not publish and was later discovered and edited by his friend, Richard Price, after Bayes's death [1]

1.1.2 Pierre-Simon Laplace (1749–1827)

Laplace, a French Mathematician, was an important figure in not just Bayesian statistics but other areas of mathematics, astronomy, and physics. We know much more about the work by Laplace than by Bayes, and Laplace has worked independently on the inverse probability problem (i.e., $P[\text{Parameter}|\text{Data}]$). Indeed, he was credited for largely formalizing the Bayesian interpretation of probability and most of the machinery for Bayesian statistics, and making it a useful technique for different problems, despite the discipline being called "Bayesian." His other contributions include the methods of least squares and the central limit theorem. See a short biography of him at https://www.britannica.com/biography/Pierre-Simon-marquis-de-Laplace.

[1]Price is another important figure in mathematics and philosophy, and had taken Bayes' theorem and applied it to insurance and moral philosophy.

1.1.3 20th Century

Until the early 1920s, the *inverse probability* method, which is based on what is now called Bayes's Theorem, was pretty much the predominant point of view of statistics. Then a point of view later known as *frequentist* statistics arrived, and quickly became the mainstream school of thinking for statistical inferences, and is still the primary framework for quantitative research. In the early 1920s, frequentist scholars, most notably R. A. Fisher and Jerzy Neyman, criticized Bayesian inference for using subjective elements in an objective discipline. In Fisher's words,

> The theory of inverse probability is founded upon an error, and must be wholly rejected—Fisher, 1925

Ironically, the term *Bayesian* was first used in one of Fisher's works. And interestingly, Fisher actually thought he "[had] been doing almost exactly what Bayes had done in the 18th century."[2]

Despite criticisms from frequentist scholars, Bayesian methods have been used by scholars in the Allies in World War II, such as Alan Turing, in an algorithm to break coded messages in the Enigma machine that the German Navy used to communicate. However, because of the more complex mathematics involved in Bayesian statistics, Bayesian statistics is limited to straight-forward problems and theoretical discussions until the early 1980s, when computing speed increased tremendously and made *Markov Chain Monte Carlo*—the primary algorithm for Bayesian estimation in modern Bayesian statistics—feasible. With the help of increased computing speed, Bayesian statistics has come back and been used as an alternative way of thinking, especially given the growing dissatisfaction towards the misuse of frequentist statistics by some scholars across disciplines. Bayesian estimation methods have also been applied to many new research questions where frequentist approaches work less well, as well as in big data analytics and machine learning.

1.2 Motivations for Using Bayesian Methods

Based on my personal experience, Bayesian methods are used quite often in statistics and related departments, as it is consistent and coherent, in contrast to frequentist where a new and probably ad hoc procedure needed to be developed to handle a new problem. For Bayesian, as long as you can formulate a model, you just run the analysis the same way as you would for simpler problems, or in Bayesian people's words "turning the Bayesian crank," and likely the difficulties would be more technical than theoretical, which is usually solved with better computational speed.

Social and behavioral scientists are relatively slow to adopt the Bayesian method, but things have been changing. In a recently accepted paper by Van De Schoot et al. (2017), the authors

[2]See the paper by John Aldrich on this.

reviewed papers in psychology between 1990 and 2015 and found that whereas less than 10% of the papers from 1990 to 1996 mentioned “Bayesian”, the proportion increased steadily and was found in close to 45% of the psychology papers in 2015. Among studies using Bayesian methods, more than 1/4 cited computational problems (e.g., nonconvergence) in frequentist methods as a reason, and about 13% cited the need to incorporate prior knowledge into the estimation process. The other reasons included the flexibility of Bayesian methods for complex and nonstandard problems, and the use of techniques traditionally attached to Bayesian such as missing data and model comparisons.

1.2.1 Problem with classical (frequentist) statistics

The rise of Bayesian methods is also related to the statistical reform movement in the past two decades. The problem is that applied researchers are obsessed with $p < .05$ and often misinterpreted a small p-value as something that it isn't (read Gigerenzer, 2004). Some scholars coined the term p-hacking to refer to the practice of obtaining statistical significance by choosing to test the data in a certain way, either consciously or subconsciously (e.g., dichotomizing using mean or median, trying the same hypothesis using different measures of the same variable, etc). This is closely related to the recent “replication crisis” in scientific research, with psychology being in the center under close scrutiny.

Bayesian is no panacea to the problem. Indeed, if misused, it can give rise to the same problems as statistical significance. My goal in this class is to help you appreciate the Bayesian tradition of embracing the uncertainty in your results, and adopt rigorous model checking and comprehensive reporting rather than relying merely on a p-value. I see this as the most important mission for someone teaching statistics.

1.3 Comparing Bayesian and Frequentist Statistics

Attributes	Frequentist	Bayesian
Interpretation of probability	Frequentist	Subjectivist
Uncertainty	How estimates vary in repeated sampling from the same population	How much prior beliefs about parameters change in light of data
What's relevant?	Current data set + all that might have been observed	Only the data set that is actually observed
How to proceed with analyses	MLE; ad hoc and depends on problems	“Turning the Bayesian crank”

1.4 Software for Bayesian Statistics

The following summarizes some of the most popular Bayesian software. Currently, JAGS and Stan are the most popular. General statistical programs like SPSS, SAS, and Stata also have some support for Bayesian analyses as well.

- WinBUGS
  - Bayesian inference Using Gibbs Sampling
  - Free, and most popular until late 2000s. Many Bayesian scholars still use WinBUGS
  - No further development
  - One can communicate from R to WinBUGS using the package `R2WinBUGS`
- JAGS
  - Just Another Gibbs Sampler
  - Very similar to WinBUGS, but written in C++, and supports user-defined functionality
  - Cross-platform compatibility
  - One can communicate from R to JAGS using the package `rjags` or `runjags`
- Stan
  - Named in honour of Stanislaw Ulam, who invented the Markov Chain Monte Carlo method
  - Uses new algorithms that are different from Gibbs sampling
  - Under very active development
  - Can interface with R through the package `rstan`, and the R packages `rstanarm` and `brms` automates the procedure for fitting models in Stan for many commonly used models

Part II

Week 2

2 Probability

2.1 History of Probability

2.1.1 Games of chance

Correspondence between French Mathematicians (Pierre de Fermat and Blaise Pascal) on gambling problem by Antoine Gombaud, Chevalier de Méré. The problem is roughly of the form[1]:

> Imagine two people playing a multi-round game. In each round, each person has an equal chance of winning. The first person who wins six rounds will get a huge cash prize. Now, consider a scenario in which A and B have played six rounds, where A has won five and B has won one. At that time, the game had to be stopped due to a thunderstorm. Since neither A nor B have reached six wins, instead of giving the prize to either one of them, they agree to divide up the prize. What would be a fair way to do so?

The discussion led to the formalization of using mathematics to solve the problem. Basically, one way is to say if A has a 97% chance of winning the prize eventually and B has a 3% chance, then A should get 97% of the prize.

2.2 Different Ways to Interpret Probability

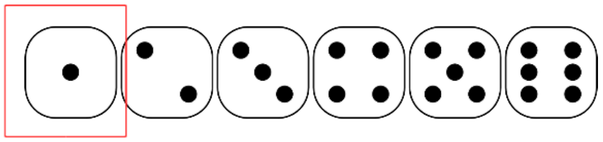
There are multiple perspectives for understanding probability.[2] What you've learned in your statistics training is likely based on the *frequentist* interpretation of probability (and thus frequentist statistics), whereas the foundation of what you will learn in this class is the *subjectivist* interpretation of probability. Understanding the different perspectives on probability is helpful for understanding the Bayesian framework.

> You don't need to commit to one interpretation of probability in order to conduct Bayesian data analysis.

[1]see the exact form at https://en.wikipedia.org/wiki/Problem_of_points
[2]See http://plato.stanford.edu/entries/probability-interpret/ for more information

2.2.1 Classical Interpretation

This is an earlier perspective and is based on counting rules. The idea is that probability is equally distributed among all "indifferent" outcomes. "Indifferent" outcomes are those where a person has no evidence to say that one outcome is more likely than another. For example, when one throws a die, one does not think that a certain number is more likely than another unless one knows that the die is biased. In this case, there are six equally likely outcomes, so the probability of each outcome is 1 / 6.



2.2.2 Frequentist Interpretation

The frequentist interpretation states that probability is essentially the long-run relative frequency of an outcome. For example, to find the probability of getting a "1" when throwing a die, one can repeat the experiment many times, as illustrated below:

Trial	Outcome
1	2
2	3
3	1
4	3
5	1
6	1
7	5
8	6
9	3
10	3

And we can plot the relative frequency of "1"s in the trials:

As you can see, with more trials, the relative frequency approaches 1 / 6. It's the reason why in introductory statistics, many of the concepts require you to think in terms of repeated sampling (e.g., sampling distribution, *p*-values, standard errors, confidence intervals), because

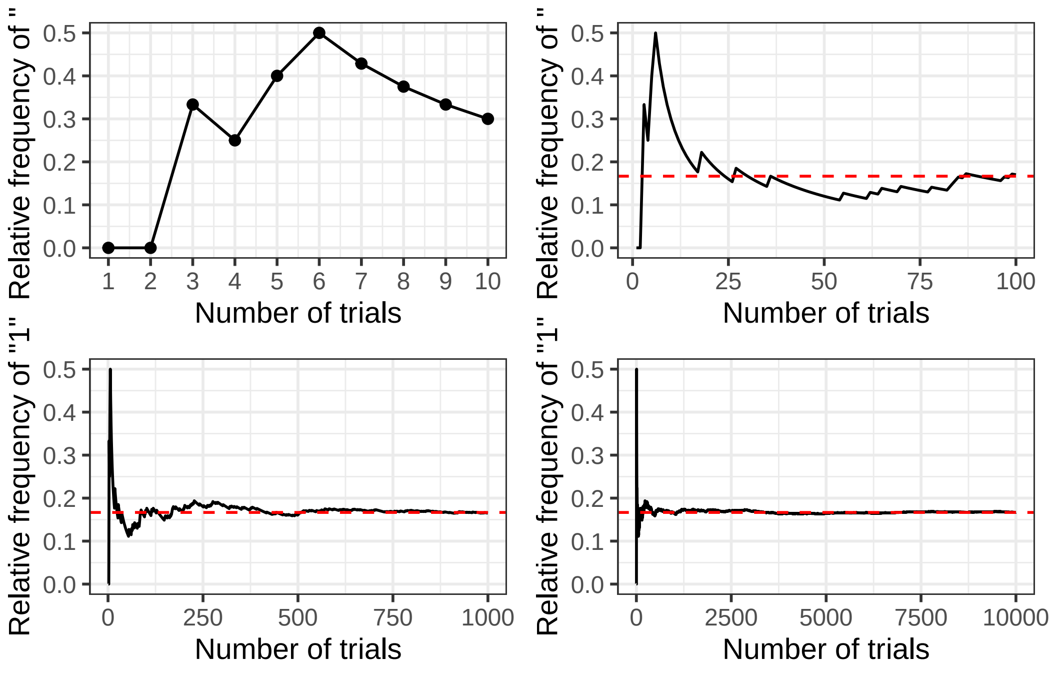


Figure 2.1: Relative frequency when repeatedly rolling a die.

probability in this framework is only possible when the outcome can be repeated. It's also the reason why we don't talk about something like:

- the probability of the null hypothesis being true, or
- the probability that the population mean is in the interval [75.5, 80.5],

because the population is fixed and cannot be repeated. Only the samples can be repeated, so probability in frequentist statistics is only about samples.

2.2.2.1 Problem of the single case

Because of the frequentist's reference to long-run relative frequency, it does not make sense to talk about the probability of an event that cannot be repeated under this framework. For example, it does not make sense to talk about

- the probability that the Democrats/Republicans will win the 2028 US Presidential Election, or
- the probability that the LA Chargers winning the 2024 Super Bowl, or
- the probability that it will rain on Christmas Day in LA in 2024,

because all these are specific events that cannot be repeated. However, it is common for laypeople to talk about probabilities or chances for these events.

2.2.3 Subjectivist Interpretation

The frequentist interpretation is sometimes called the "objectivist view," as the reference of probability is based on empirical evidence of long-run relative frequency (albeit hypothetical in many cases). In contrast, the *subjectivist* view of probability is based on one's belief. For example, when I say that the probability of getting a "1" from rolling a die is 1 / 6, it reflects the state of my mind about the die. My belief can arise from different sources: Maybe I make the die and know it is a fair one; maybe I saw someone throwing the die 1,000 times, and the number of "1"s was close to 1,000 / 6, or maybe someone I trust and with authority says that the die has a 1-in-6 chance of showing a "1".

The "subjective" component has been criticized a lot by frequentist scholars, sometimes unfairly. To be clear, what "subjective" here means is that probability reflects the state of one's mind instead of the state of the world, and so it is totally fine that two people can have different beliefs about the same event. However, it does not mean that probability is arbitrary, as the beliefs are subjected to the constraints of the axioms of probability as well as the condition that the person possessing such beliefs is *rational*.[3] Therefore, if two persons are exposed to the same information, they should form similar, though likely not identical, beliefs about the event.

The subjective interpretation works perfectly fine with single events, as one can have a belief about whether it rains on a particular day or a belief about a particular election result.

2.2.3.1 Calibrating a subjective belief

In order to represent one's belief by probability, one needs to assign a nonzero value to every plausible outcome of an event. This has been the job of odds-makers for a long time. Indeed, a lot of the development in the field of probability has to do with coming up with a fair bet. The process of assigning probabilities to outcomes of an event according to one's belief is called *calibration*. For example, consider three possible outcomes for tomorrow's weather. For simplicity, consider three mutually exclusive possible outcomes: sunny, cloudy, and rainy.

To calibrate my belief, consider first if you bet \$10, and the return is (a) \$30 for sunny, (b) \$30 for cloudy, and (c) \$30 for rainy. Which one will you bet? If you're like me in LA, I'm pretty sure I'll bet on (a), as I think that it is more likely to have a sunny day. This means that setting $P(\text{sunny}) = P(\text{cloudy}) = P(\text{rainy}) = 1 / 3$ is not a good reflection of my belief.

[3] In a purely subjectivist view of probability, assigning a probability P to an event does not require any justifications, as long as it follows the axioms of probability. For example, I can say that the probability of me winning the lottery and thus becoming the wealthiest person on earth tomorrow is 95%, which by definition would make the probability of me not winning the lottery 5%. Most Bayesian scholars, however, do not endorse this version of subjectivist probability and require justifications of one's beliefs (that have some correspondence to the world).

Now consider the bet with the returns (a) $20 for sunny, (b) $30 for cloudy, and (c) $60 for rainy. This would reflect the belief that there is a 50% chance of a sunny day, 33.33% chance of a cloudy day, and 16.67% chance of a rainy day. Will you take the bet? This is an improvement from the last one, but I would still say a sunny day is a good bet, which suggests that the probability of 50% is too low for a sunny day. The idea is to continue iterating until it is hard to consider (a), (b), or (c) as a clear betting favorite. For me, this would end up being something like (a) $16.7 for sunny, (b) $33.3 for cloudy, and (c) $100 for rainy, which would correspond to 60% sunny, 30% cloudy, and 10% rainy.

If it's hard for you to consider the gambling analogy, an alternative way is to consider how many times is a sunny day more likely than a non-sunny day, and how many times is a cloudy day more likely than a rainy day. For example, I may consider a sunny day to be twice as likely as a non-sunny day, which would give the probability of a sunny day to be 66.67%. Then, if I also think that a cloudy day is three times as likely as a rainy day, I would assign a probability of 33.33% $\times$ 3 / 4 = 25% for a cloudy day, and a probability of 33.33% $\times$ 1 / 4 = 8.33% for a rainy day.

The process of calibrating one's belief plays a key role in Bayesian data analysis, namely in the form of formulating a *prior* probability distribution.

2.3 Basics of Probability

Kolmogorov axioms

For an event A_i (e.g., getting a "1" from throwing a die)

- $P(A_i) \geq 0$ [All probabilities are non-negative]
- $P(A_1 \cup A_2 \cup \cdots) = 1$ [Union of all possibilities is 1]
- $P(A_1) + P(A_2) = P(A_1 \text{ or } A_2)$ [Addition rule]

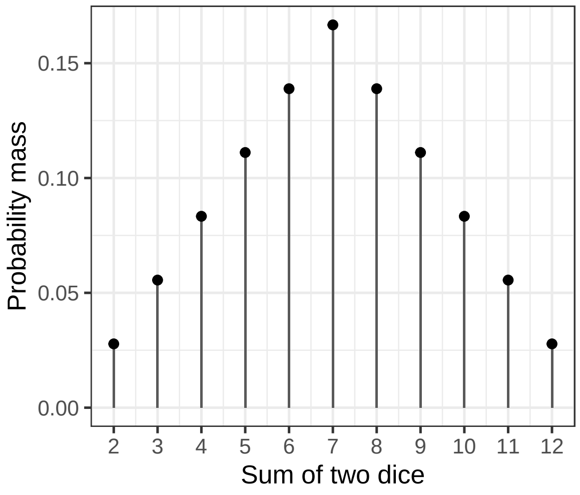
Consider two events, for example, on throwing a die,

- A: The number is odd
- B: The number is larger than or equal to 4

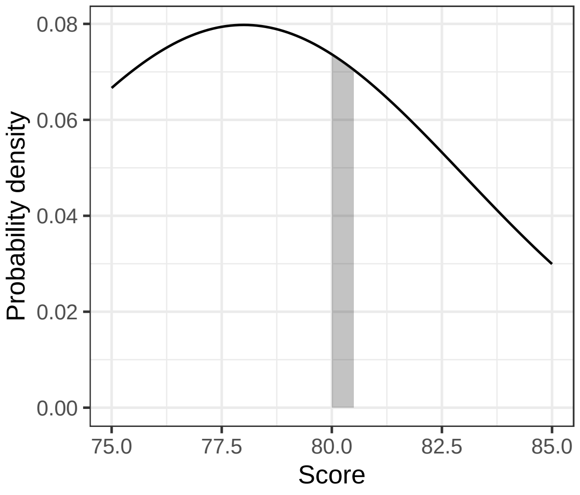
Assuming that die is (believed to be) fair, you can verify that the probability of A is $P(A) = 3 / 6 = 1 / 2$, and the probability of B is also $P(B) = 3 / 6 = 1 / 2$.

2.3.1 Probability Distributions

- Discrete event (e.g., the outcome of throwing a die or an election): probability *mass*. The probability is nonzero, at least for some outcomes. The graph below on the left shows the probability mass of the sum of the numbers from two dice.
- Continuous event (e.g., temperature): probability *density*.[4] The probability is basically zero for any outcome. Instead, the probability density is approximated by $P(A \leq a \leq A + h)/h$ for a very small h.
  - For example, to find the probability density that a person's well-being score is 80, we first find the probability that a person scores between 80 and 80.5 (or 80 and 80.0005), and divide that probability by 0.5 (or 0.0005). See the shaded area of the graph below on the right.



(a) Probability mass



(b) Probability density

Figure 2.2: Examples of probability distributions

For this course, as in the textbook, we use $P(x)$ to mean both the probability mass for an outcome x when the event is discrete, and the probability density at an outcome x when the event is continuous.

[4]For many problems in the social and behavioral sciences, the measured variables are not truly continuous, but we still use continuous distributions to approximate them.

2.3.1.1 Example: Normal Distribution

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left[\frac{x - \mu}{\sigma}\right]^2\right)$$

```r
# Write a function to compute the density of an outcome x
# for a normal distribution
my_normal_density <- function(x, mu, sigma) {
    exp(- ((x - mu) / sigma) ^2 / 2) / (sigma * sqrt(2 * pi))
}
# For example, density at x = 36 in a normal distribution
# with mu = 50 and sigma = 10
my_normal_density(36, mu = 50, sigma = 10)
```

```
#> [1] 0.01497275
```

2.3.2 Summarizing a Probability Distribution

While it is useful to know the probability mass/density of every possible outcome, in many situations, it is helpful to summarize a distribution by some numbers.

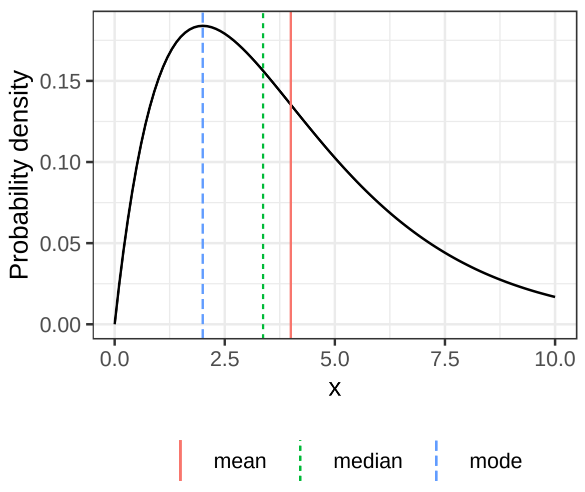
2.3.2.1 Central Tendency

- Mean: $E(X) = \int x \cdot P(x) dx$
- Median: 50th percentile; the median of X is Mdn_X such that $P(X \leq Mdn_X) = 1 \ / \ 2$
- Mode: A value with maximum probability mass/density

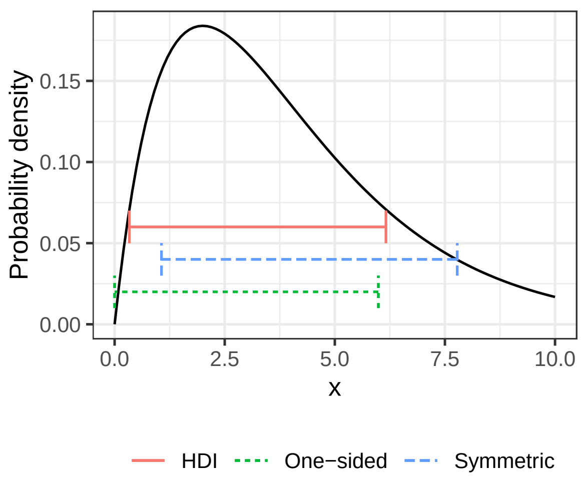
See Figure 2.3a for examples.

2.3.2.2 Dispersion

- Variance: $V(X) = E[X - E(X)]^2$
  - Standard deviation: $\sigma(X) = \sqrt{V(X)}$
- Median absolute deviation (MAD): $1.4826 \times Mdn(|X - Mdn_X|)$



(a) Central Tendency



(b) Interval

Figure 2.3: Measures of central tendency and interval

2.3.2.3 Interval

Use $C(X)$ to denote an interval. A $W\%$ interval means $P(X \in C[X]) \approx W\%$

- One-sided interval: $C(X)$ is half-bounded
- Symmetric $W\%$ interval: $C(X) = [L(X), U(X)]$ is bounded, with $P(X < L[X]) = P(X > U[X]) \approx W\%/2$
  - Also called *equal-tailed* interval
- Highest density $W\%$ interval (HDI): $P(x_c) \geq P(x_o)$ for every x_c in $C(X)$ and every x_o outside $C(X)$. In general, the HDI is the shortest $W\%$ interval.

The plot in Figure 2.3b shows several 80% intervals.

2.3.2.4 Computing Summaries of Sample Distributions Using R

```r
# Simulate data from a half-Student's t distribution with
# df = 4, and call it sim_s
sim_s <- rt(10000, df = 4) # can be both positive and negative
sim_s <- abs(sim_s) # take the absolute values
ggplot(data.frame(x = sim_s), aes(x = x)) +
    geom_histogram(binwidth = 0.1)
```

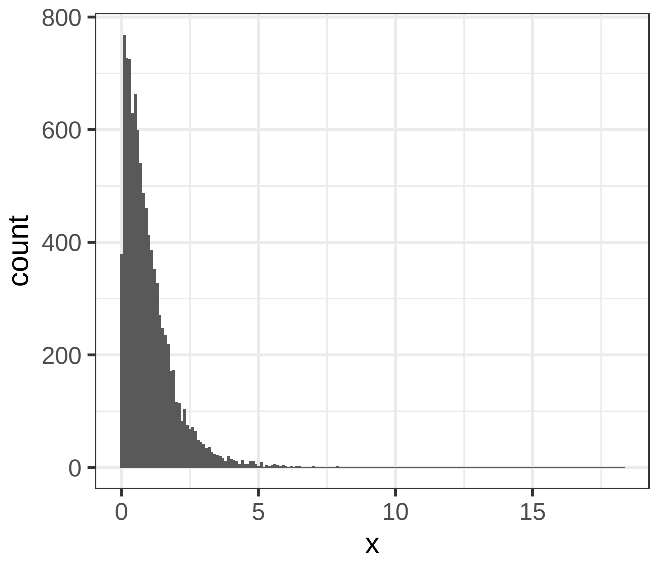


Figure 2.4: Simulated density of a half-Student's t distribution

```r
# Central tendency
# (note: the mode is difficult to compute for continuous
# variables, and rarely used in this course.)
c(mean = mean(sim_s),
  median = median(sim_s),
  mode = density(sim_s, bw = "SJ")$x[
      which.max(density(sim_s, bw = "SJ")$y)
  ])
```

```
#>      mean    median      mode 
#> 1.0082926 0.7426326 0.1021776
```

```r
# Dispersion
c(
    variance = var(sim_s),
    sd = sd(sim_s),
    mad = mad(sim_s)
)
```

```
#>  variance        sd       mad 
#> 1.0231058 1.0114869 0.6996741
```

```r
# 80% Interval
c(`0%` = 0, quantile(sim_s, probs = .8)) # right-sided
```

```
#>     0%    80%
#> 0.0000 1.5598
```

```r
c(quantile(sim_s, probs = .2), `100%` = Inf) # left-sided
```

```
#>       20%      100%
#> 0.2680906       Inf
```

```r
quantile(sim_s, probs = c(.1, .9)) # equal-tailed/symmetric
```

```
#>       10%       90%
#> 0.1299027 2.1371636
```

```r
HDInterval::hdi(sim_s)
```

```
#>        lower        upper
#> 0.0003616342 2.7992620765
#> attr(,"credMass")
#> [1] 0.95
```

2.3.3 Multiple Variables

- Joint probability: $P(X, Y)$
- Marginal probability:

$$P(X) = \int P(X, y) dy$$

$$P(Y) = \int P(x, Y) dx$$

  - The probability that outcome X happens, regardless of what values Y take.

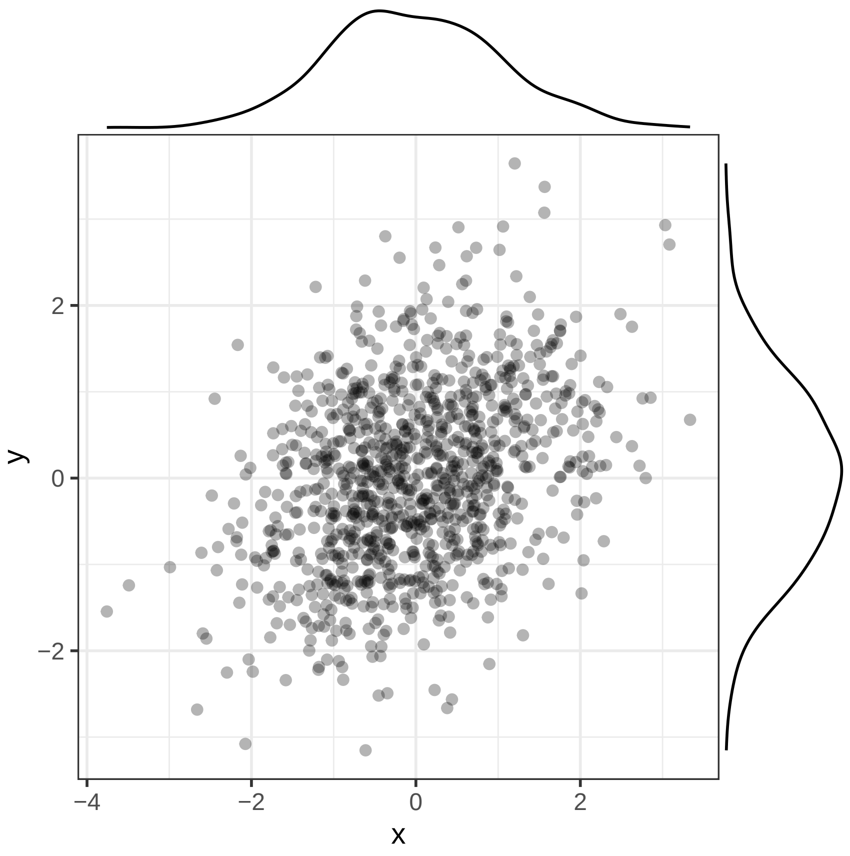


Figure 2.5: Joint and Marginal Distributions

2.3.3.1 Conditional Probability

Conditional probability is the probability of an event given some other information. In the real world, you can say that everything is conditional. For example, the probability of getting an odd number on throwing a die is 1/2 is conditional on the die being fair. We use $P(A \mid B)$ to represent the the conditional probability of event A given event B..

Continuing from the previous example, $P(A \mid B)$ is the conditional probability of getting an odd number, *knowing that the number is at least 4*. By definition, conditional probability is the probability that both A and B happen, divided by the probability that B happens.

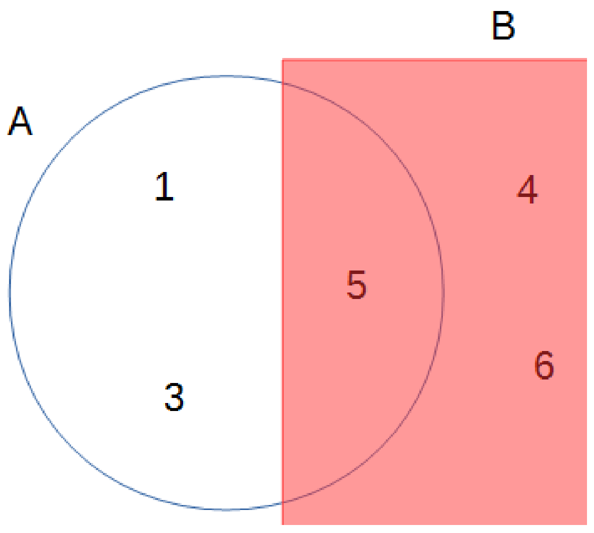
! **Conditional Probability**

$$P(A \mid B) = \frac{P(A,B)}{P(B)}$$

In the example, $P(A,B) = 1 \ / \ 6$, because 5 is the only even number ≥ 4 when throwing a die. Thus,

$$\begin{aligned} P(A \mid B) &= 1/3 \\ &= \frac{P(A,B)}{P(B)} \\ &= \frac{1/6}{1/2} \end{aligned}$$

This picture should make it clear:



❗ Please recognize that $P(A \mid B) \neq P(B \mid A)$. For example, when throwing a die, $P(\text{number is six} \mid \text{even number}) = 1/3$, but $P(\text{even number} \mid \text{number is six})$ is 1.

2.3.3.2 Independence

❗ Two events, A and B, are independent if $P(A \mid B) = P(A)$

This means that any knowledge of B does not (or should not) affect one's belief about A. Consider the example:

- A: A die shows five or more
- B: A die shows an odd number

Here is the joint probability

	>= 5	<= 4
odd	1/6	2/6
even	1/6	2/6

So the conditional probability of $P(>= 5 \mid \text{odd}) = (1/6)\ /\ (1/2) = 1/3$, which is the same as $P(>= 5 \mid \text{even}) = (1/6)\ /\ (1/2) = 1/3$. Similarly it can be verified that $P(<= 4 \mid \text{odd}) = P(<= 4 \mid \text{even}) = 2/3$. Therefore, A and B are independent.

On the other hand, for the example

- A: A die shows four or more
- B: A die shows an odd number

the joint probabilities are

	>= 4	<= 3
odd	1/6	2/6
even	2/6	1/6

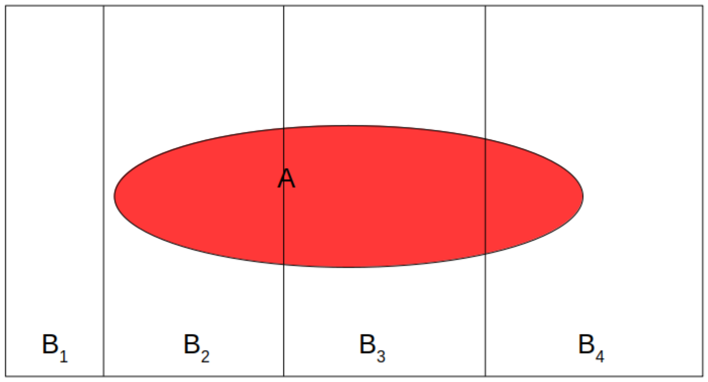
Obviously, A and B are not independent because once we know that the number is four or above, the probability of whether it is an odd number or not changes.

! Independence can also be expressed as

If A and B are independent, $P(A, B) = P(A)P(B)$

2.3.4 Law of Total Probability

When we talk about conditional probability, like $B_1 = 4$ or above and $B_2 = 3$ or below, we can get $P(A \mid B_1)$ and $P(A \mid B_2)$ (see the figure below), we refer $P(A)$ as the *marginal probability*, meaning that the probability of A without knowledge of B.



If $B_1, B_2, \cdots, B_n$ are all mutually exclusive possibilities for an event (so they add up to a probability of 1), then

! Law of Total Probability

$$\begin{aligned} P(A) &= P(A, B_1) + P(A, B_2) + \cdots + P(A, B_n) \\ &= P(A \mid B_1)P(B_1) + P(A \mid B_2)P(B_2) + \cdots + P(A \mid B_n)P(B_n) \\ &= \sum_{k=1}^{n} P(A \mid B_k)P(B_k) \end{aligned}$$

3 Bayes's Theorem

The Bayes's theorem is, surprisingly (or unsurprisingly), very simple:

$$P(B \mid A) = \frac{P(A \mid B)P(B)}{P(A)}$$

More generally, we can expand it to incorporate the law of total probability to make it more applicable to data analysis. Consider B_i as one of the n many possible mutually exclusive events, then

$$\begin{aligned}
P(B_i \mid A) &= \frac{P(A \mid B_i)P(B_i)}{P(A)} \\
&= \frac{P(A \mid B_i)P(B_i)}{P(A \mid B_1)P(B_1) + P(A \mid B_2)P(B_2) + \cdots + P(A \mid B_n)P(B_n)} \\
&= \frac{P(A \mid B_i)P(B_i)}{\sum_{k=1}^{n} P(A \mid B_k)P(B_k)}
\end{aligned}$$

If B_i is a continuous variable, we will replace the sum by an integral,

$$P(B_i \mid A) = \frac{P(A \mid B_i)P(B_i)}{\int_k P(A \mid B_k)P(B_k)}$$

The denominator is not important for practical Bayesian analysis, therefore, it is sufficient to write the above equality as

$$P(B_i \mid A) \propto P(A \mid B_i)P(B_i)$$

3.1 Example 1: Base rate fallacy (From Wikipedia)

3.1.1 Question

A police officer stops a driver *at random* and does a breathalyzer test for the driver. The breathalyzer is known to detect true drunkenness 100% of the time, but in 1% of the cases, it gives a false positive when the driver is sober. We also know that, in general, for every 1,000 drivers passing through that spot, one is driving drunk. Suppose that the breathalyzer shows positive for the driver. What is the probability that the driver is truly drunk?

3.1.2 Solution

$P(\text{positive}|\text{drunk}) = 1$
$P(\text{positive}|\text{sober}) = 0.01$
$P(\text{drunk}) = 1/1000$
$P(\text{sober}) = 999/1000$

Using Bayes' Theorem,

$$\begin{aligned} P(\text{drunk}|\text{positive}) &= \frac{P(\text{positive}|\text{drunk})P(\text{drunk})}{P(\text{positive}|\text{drunk})P(\text{drunk}) + P(\text{positive}|\text{sober})P(\text{sober})} \\ &= \frac{1 \times 0.001}{1 \times 0.001 + 0.01 \times 0.999} \\ &= 100/1099 \approx 0.091 \end{aligned}$$

So there is less than a 10% chance that the driver is drunk even when the breathalyzer shows positive.

You can verify that with a simulation using R:

```r
set.seed(4)
truly_drunk <- c(rep("drunk", 100), rep("sober", 100 * 999))
table(truly_drunk)
```

```
#> truly_drunk
#> drunk sober
#>   100 99900
```

```r
breathalyzer_test <- ifelse(truly_drunk == "drunk",
    # If drunk, 100% chance of showing positive
    "positive",
    # If not drunk, 1% chance of showing positive
    sample(c("positive", "negative"), 999000,
        replace = TRUE, prob = c(.01, .99)
    )
)
# Check the probability p(positive | sober)
table(breathalyzer_test[truly_drunk == "sober"])
```

```
#>
#> negative positive
#>    98903      997
```

```
# 997 / 99900 = 0.00997998, so the error rate is less than 1%
# Now, Check the probability p(drunk | positive)
table(truly_drunk[breathalyzer_test == "positive"])
```

```
#>
#> drunk sober
#>   100   997
```

```
# 100 / (100 + 997) = 0.0911577, which is only 9.1%!
```

3.2 Bayesian Statistics

`Bayesian statistics` is a way to estimate some parameter θ (i.e., some quantities of interest, such as the population mean, regression coefficient, etc) by applying Bayes' Theorem.

$$P(\theta|D) \propto P(D|\theta)P(\theta)$$

There are three components in the above equality:

- $P(D|\theta)$, the probability that you observe data D, given the parameter θ; this is called the `likelihood` (Note: It is the likelihood of θ, but probability about y)
- $P(\theta)$, the probability distribution θ, without referring to the data D. This usually requires appeals to one's degree of belief, and so is called the *prior*
- $P(\theta|y)$, the updated probability distribution of θ, after observing the data D; this is called the *posterior*

On the other hand, classical/frequentist statistics focuses solely on the likelihood function.[1] In Bayesian statistics, the goal is to update one's belief about θ based on the observed data D.

[1]The likelihood function in classical/frequentist statistics is usually written as $P(y;\theta)$. You will notice that here, I write the likelihood for classical/frequentist statistics to be different from the one used in Bayesian statistics. This is intentional: In frequentist conceptualization, θ is fixed, and it does not make sense to talk about the probability of θ. This implies that we cannot condition on θ, because conditional probability is defined only when $P(\theta)$ is defined.

3.3 Example 2: Locating a Plane

Consider a highly simplified scenario of locating a missing plane in the sea. Assume that we know the plane, before missing, happened to be flying at the same latitude, heading west across the Pacific, so we only need to find its longitude. We want to go out to collect debris (data) so that we can narrow the location (θ) of the plane down.

3.3.1 Prior

We start with our prior. Assume that we have some rough idea where the plane should be, so we express our belief in a probability distribution like the following:

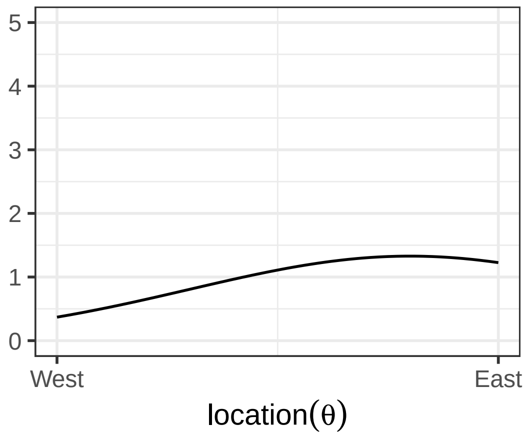


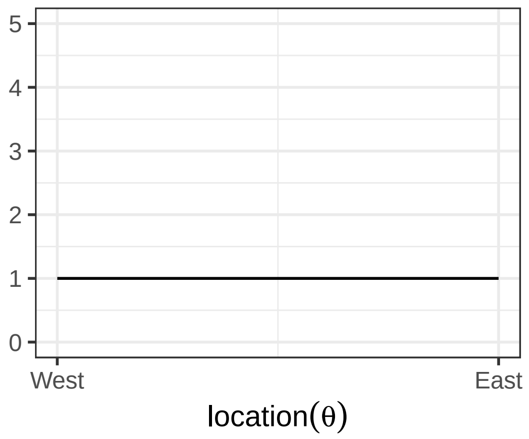
Figure 3.1: Prior distribution.

which says that our belief is that the plane is about twice more likely to be towards the east than towards the west. Below are two other options for priors (out of infinitely many), one providing virtually no information and the other encoding stronger information:

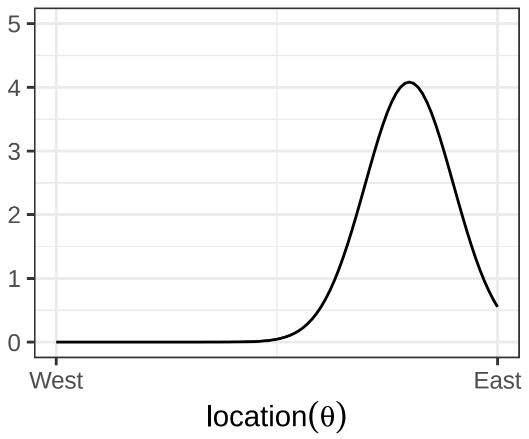
The prior is chosen to reflect the researcher's belief, so different researchers will likely formulate a different prior for the same problem, and that's okay as long as the prior is reasonable and justified. Later, we will learn that in regular Bayesian analyses, with a moderate sample size, different priors generally only make negligible differences.

3.3.2 Likelihood

Now, assume that we have collected debris in the locations shown in the graph,



(a) Noninformative prior.



(b) Informative prior.

Figure 3.2: More options for prior distribution.

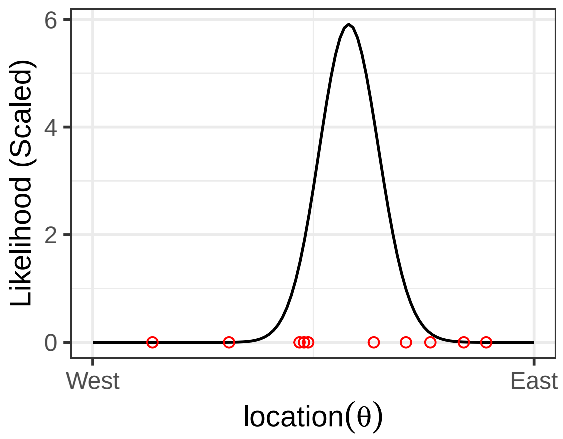


Figure 3.3

3.3.3 Posterior

Now, from Bayes's Theorem,

$$\text{Posterior Probability} \propto \text{Prior Probability} \times \text{Likelihood}$$

So we can simply multiply the prior probabilities and the likelihood to get the posterior probability for every location. A rescaling step is needed to ensure that the area under the curve will be 1, which is usually performed by the software.

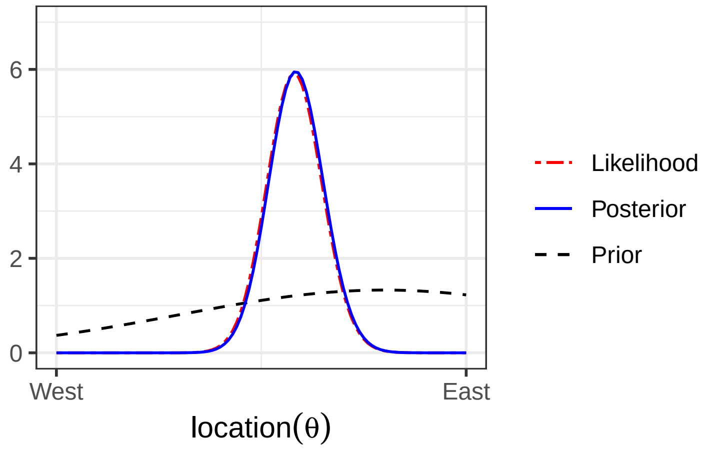


Figure 3.4

As illustrated below, the posterior distribution is a synthesis of (a) the prior and (b) the data (likelihood).

3.3.4 Influence of Prior

Figure 3.5 shows what happen with a stronger prior:

3.3.5 Influence of More Data

Figure 3.6 shows what happen with 20 more data points:

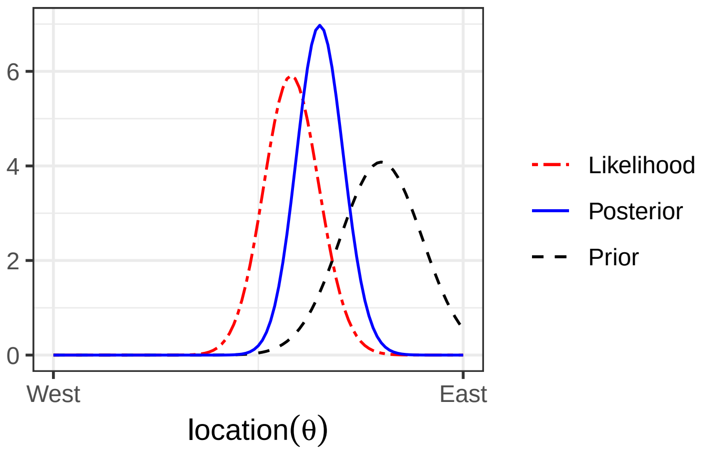


Figure 3.5

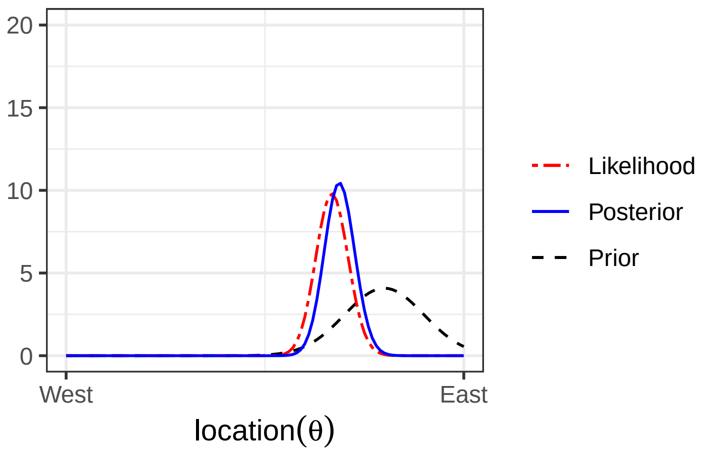


Figure 3.6

3.4 Data-Order Invariance

In many data analysis applications, researchers collect some data D_1, and then collect some more data D_2. An example would be researchers conducting two separate experiments to study the same research question. In Bayesian statistics, one can consider three ways to obtain the posterior:

1. Update the belief with D_1, and then with D_2
2. Update the belief with D_2, and then with D_1
3. Update the belief with both D_1 and D_2 simultaneously

Whether these three ways give the same posterior depends on whether **data-order invariance** holds. If the inference of D_1 does not depend on D_2, or vice versa, then all three ways lead to the same posterior. Specifically, if we have **conditional independence** such that

$$P(D_1, D_2 \mid \theta) = P(D_1 \mid \theta)P(D_2 \mid \theta),$$

then one can show all three ways give the same posterior (see section 4.4 and 4.5 of Johnson et al., 2022).

Exchangeability*

Exchangeability is an important concept in Bayesian statistics. Data are exchangeable when the joint distribution, $P(D_1, \dots, D_N)$, does not depend on the ordering of the data. A simple way to think about it is if you scramble the order of your outcome variable in your data set and still can obtain the same statistical results, then the data are exchangeable. An example situation where data are not exchangeable is

- D_1 is from year 1990, D_2 is from year 2020, and the parameter θ changes from 1990 to 2020

When data are exchangeable, conditional independence would generally hold.[2]

3.5 Bernoulli Likelihood

For binary data y (e.g., coin flip, pass/fail, diagnosed/not), an intuitive way to analyze is to use a Bernoulli model:

$$P(y = 1 \mid \theta) = \theta$$
$$P(y = 0 \mid \theta) = 1 - \theta\text{'}$$

[2]The de Finetti's theorem shows that when the data are exchangeable and can be considered an infinite sequence (i.e., not from a tiny finite population), then the data are conditionally independent given some θ.

which is more compactly written as

$$P(y \mid \theta) = \theta^y (1 - \theta)^{(1-y)},$$

where $\theta \in [0, 1]$ is the probability of a "1". You can verify that the compact form is the same as the longer form.

3.5.1 Multiple Observations

When there are more than one y, say $y_1, \dots, y_N$, that are conditionally independent, we have

$$\begin{aligned} P(y_1, \dots, y_N \mid \theta) &= \prod_{i=1}^{N} P(y_i \mid \theta) \\ &= \theta^{\sum_{i=1}^N y_i} (1 - \theta)^{\sum_{i=1}^N (1 - y_i)}, \\ &= \theta^z (1 - \theta)^{N - z} \end{aligned}$$

where z is the number of "1"s (e.g., the number of heads in coin flips). Note that the likelihood only depends on z, not the individual ys. In other words, the likelihood is the same as long as there are z heads, regardless of when those heads occur.

Let's say $N = 4$ and $z = 1$. We can plot the likelihood in R:

```r
# Write the likelihood as a function of theta
lik <- function(th, num_flips = 4, num_heads = 1) {
    th ^ num_heads * (1 - th) ^ (num_flips - num_heads)
}
# Likelihood of theta = 0.5
lik(0.5)
```

```
#> [1] 0.0625
```

```r
# Plot the likelihood
ggplot(data.frame(th = c(0, 1)), aes(x = th)) +
    # `stat_function` for plotting a function
    stat_function(fun = lik) +
    # use `expression()` to get greek letters
    labs(x = expression(theta),
    y = "Likelihood with N = 4 and z = 1")
```

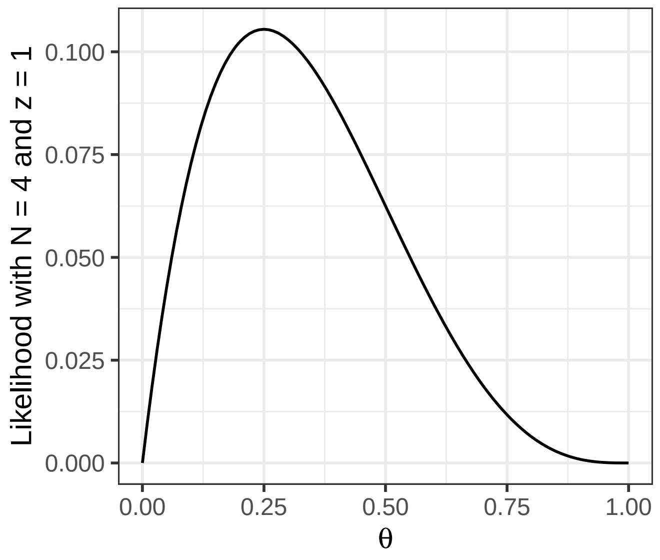


Figure 3.7: Binomial likelihood function with $N = 4$ and $z = 1$

3.5.2 Setting Priors

Remember again the relationship between the prior and the posterior:

$$P(\theta|y) \propto P(y|\theta)P(\theta)$$

The posterior distributions are mathematically determined once the priors and the likelihood are set. However, the mathematical form of the posterior is sometimes very difficult to deal with.

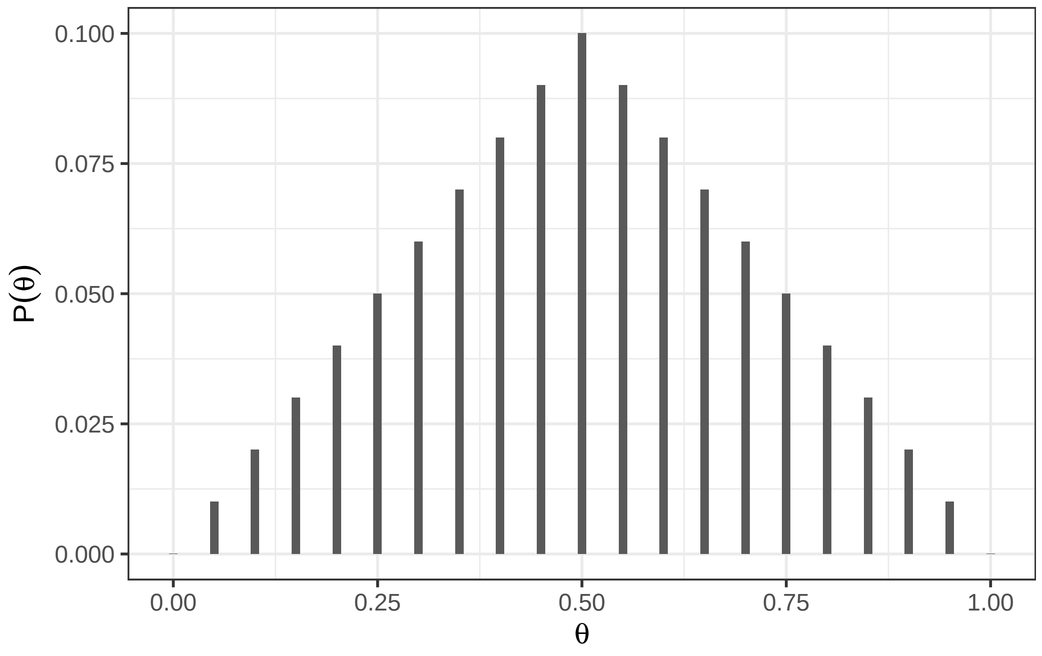
One straightforward, brute-force method is to discretize the parameter space into a number of points. For example, by taking $\theta = 0$, 0.05, 0.10, . . . , 0.90, 0.95, 1.00, one can evaluate the posterior at these 21 **grid points**.

Let's use a prior that peaks at 0.5 and linearly decreases to both sides. I assume that $\theta = 0.5$ is twice as likely as $\theta = 0.25$ or $\theta = 0.75$ to reflect my belief that the coin is more likely to be fair.

```r
# Define a grid for the parameter
grid_df <- data.frame(th = seq(0, 1, by = 0.05))
# Set the prior mass for each value on the grid
grid_df$pth <- c(0:10, 9:0)  # linearly increasing, then decreasing
# Convert pth to a proper distribution such that the value
# sum to one
```

```r
grid_df$pth <- grid_df$pth / sum(grid_df$pth)                    (1)
# Plot the prior
ggplot(grid_df, aes(x = th, y = pth)) +
    geom_col(aes(x = th, y = pth),
        width = 0.01,
    ) +
    labs(y = expression(P(theta)), x = expression(theta))
```

(1) This line ensures that the probability values sum to one. This is a trick we will use to obtain the posterior probability.



Prior Predictive Distribution

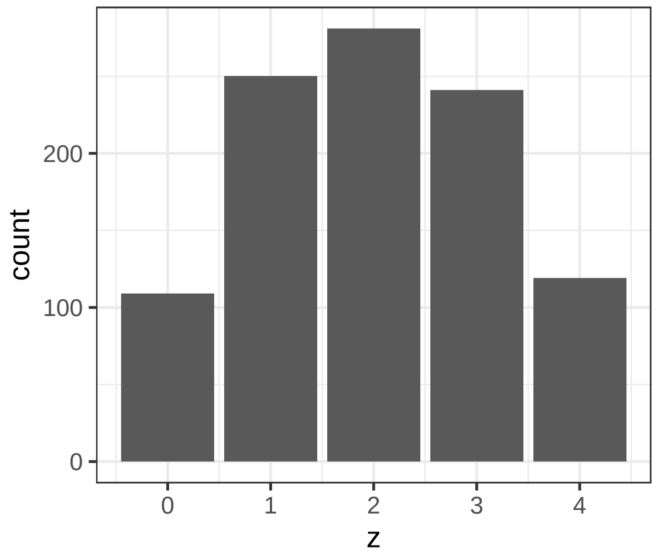
One way to check whether the prior is appropriate is to use the **prior predictive distribution**. Bayesian models are **generative** because they can be used to simulate data. The prior predictive distribution can be obtained by first simulating some θ values from the prior distribution and then simulating a data set for each θ.

```r
# Draw one theta
num_trials <- 4  # number of draws
sim_th1 <- sample(grid_df$th, size = 1,
                  # based on prior probability
                  prob = grid_df$pth)
# Simulate new data of four flips based on model
sim_y1 <- rbinom(num_trials, size = 1, prob = sim_th1)

# Repeat many times
# Set number of simulation draws
num_draws <- 1000
sim_th <- sample(grid_df$th, size = num_draws, replace = TRUE,
                 # based on prior probability
                 prob = grid_df$pth)
# Use a for loop
# Initialize output
sim_y <- matrix(NA, nrow = num_trials, ncol = num_draws)
for (s in seq_len(num_draws)) {
    # Store simulated data in the sth column
    sim_y[, s] <- rbinom(num_trials, size = 1, prob = sim_th[s])
}
# Show the first 10 simulated data sets based on prior:
sim_y[, 1:10]
```

```
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#> [1,]    1    0    0    1    0    1    0    0    0     1
#> [2,]    0    0    0    1    1    0    1    0    1     1
#> [3,]    1    0    0    1    0    0    0    1    1     1
#> [4,]    1    0    1    1    1    0    0    1    1     0
```

```r
# Show the distribution of number of heads
sim_heads <- colSums(sim_y)
ggplot(data.frame(z = sim_heads), aes(x = z)) +
    geom_bar()
```



The outcome seems to fit our intuition that it's more likely to be half heads and half tails, but there is a lot of uncertainty.

3.5.3 Summarizing the Posterior

```
grid_df <- grid_df |>
    mutate(
        # Use our previously defined lik() function
        py_th = lik(th, num_flips = 4, num_heads = 1),
        # Product of prior and likelihood
        `prior x lik` = pth * py_th,
        # Scaled the posterior
        pth_y = `prior x lik` / sum(`prior x lik`)
    )
# Print a table
knitr::kable(grid_df)
```

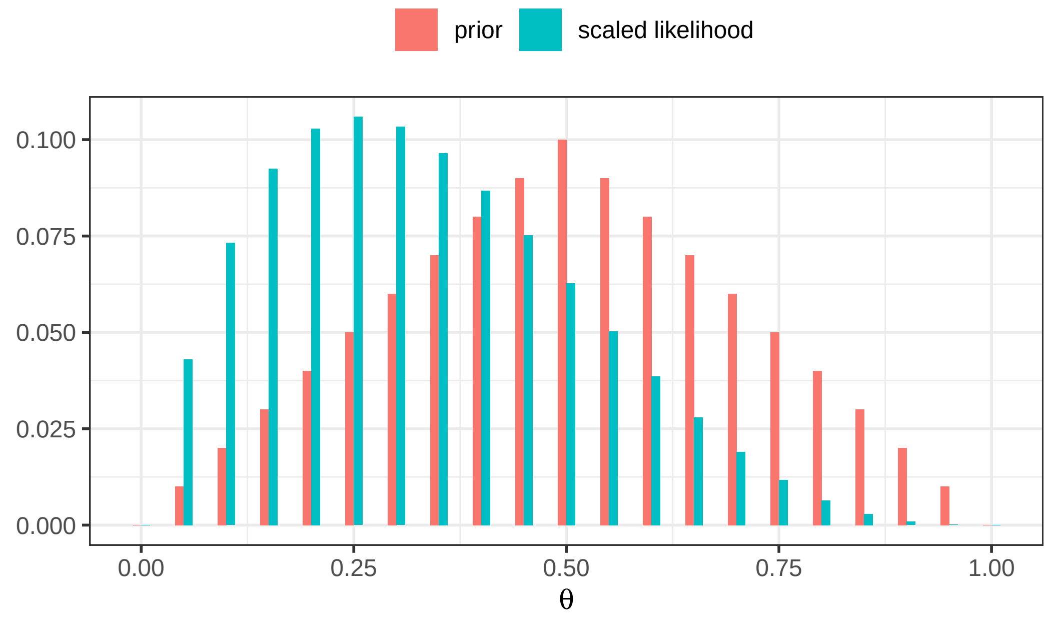
th	pth	py_th	prior x lik	pth_y
0.00	0.00	0.0000000	0.0000000	0.0000000
0.05	0.01	0.0428687	0.0004287	0.0073359
0.10	0.02	0.0729000	0.0014580	0.0249500
0.15	0.03	0.0921188	0.0027636	0.0472914

th	pth	py_th	prior x lik	pth_y
0.20	0.04	0.1024000	0.0040960	0.0700927
0.25	0.05	0.1054688	0.0052734	0.0902416
0.30	0.06	0.1029000	0.0061740	0.1056525
0.35	0.07	0.0961187	0.0067283	0.1151381
0.40	0.08	0.0864000	0.0069120	0.1182815
0.45	0.09	0.0748688	0.0067382	0.1153071
0.50	0.10	0.0625000	0.0062500	0.1069530
0.55	0.09	0.0501187	0.0045107	0.0771891
0.60	0.08	0.0384000	0.0030720	0.0525695
0.65	0.07	0.0278687	0.0019508	0.0333832
0.70	0.06	0.0189000	0.0011340	0.0194056
0.75	0.05	0.0117188	0.0005859	0.0100268
0.80	0.04	0.0064000	0.0002560	0.0043808
0.85	0.03	0.0028687	0.0000861	0.0014727
0.90	0.02	0.0009000	0.0000180	0.0003080
0.95	0.01	0.0001187	0.0000012	0.0000203
1.00	0.00	0.0000000	0.0000000	0.0000000

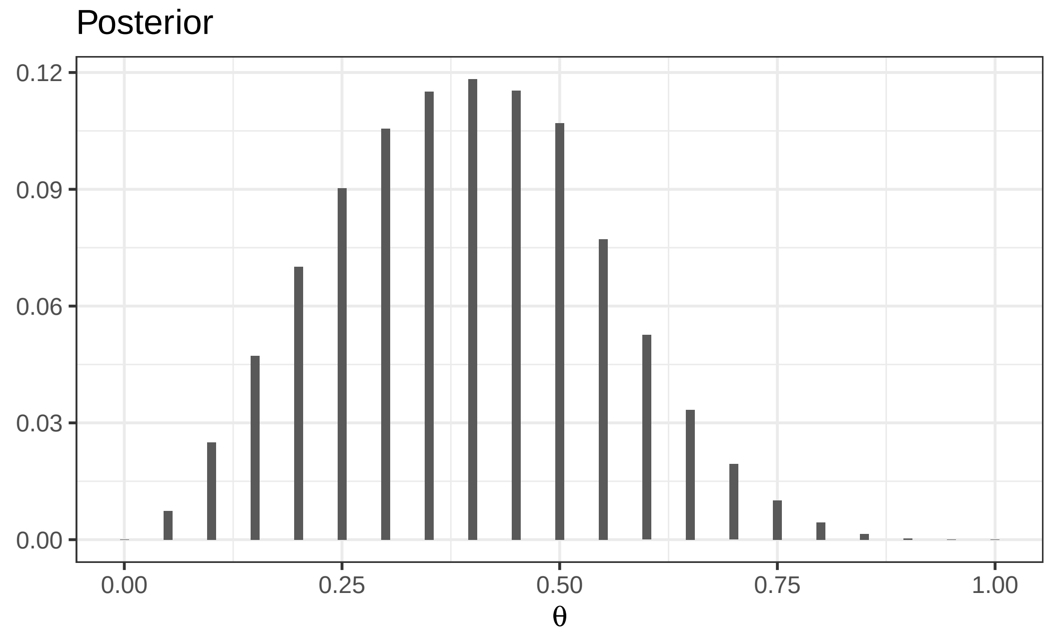
```r
# Plot the prior/likelihood and the posterior
ggplot(data = grid_df, aes(x = th)) +
    geom_col(aes(x = th - 0.005, y = pth, fill = "prior"),
        width = 0.01,
    ) +
    geom_col(aes(x = th + 0.005, y = py_th / sum(py_th),
        fill = "scaled likelihood"), width = 0.01,
    ) +
    labs(fill = NULL, y = NULL, x = expression(theta)) +
    theme(legend.position = "top")
ggplot(data = grid_df, aes(x = th)) +
    geom_col(aes(x = th, y = pth_y), width = 0.01) +
    labs(
        fill = NULL, y = NULL, title = "Posterior",
        x = expression(theta)
    )
```

Figure 3.8b shows the posterior distribution, which represents our updated belief about θ. We can summarize it by simulating θ values from it and compute summary statistics:

```r
# Define a function for computing posterior summary
summ_draw <- function(x) {
```



(a) Prior and likelihood



(b) Posterior

Figure 3.8: Bernoulli posterior distribution

```r
    c(
        mean = mean(x),
        median = median(x),
        sd = sd(x),
        mad = mad(x),
        `ci.1` = quantile(x, prob = .1, names = FALSE),
        `ci.9` = quantile(x, prob = .9, names = FALSE)
    )
}
# Sample from the posterior
post_samples <- sample(
    grid_df$th,
    size = 1000, replace = TRUE,
    prob = grid_df$pth_y
)
summ_draw(post_samples)
```

```
#>      mean    median        sd       mad      ci.1      ci.9
#> 0.3848000 0.4000000 0.1538429 0.1482600 0.2000000 0.6000000
```

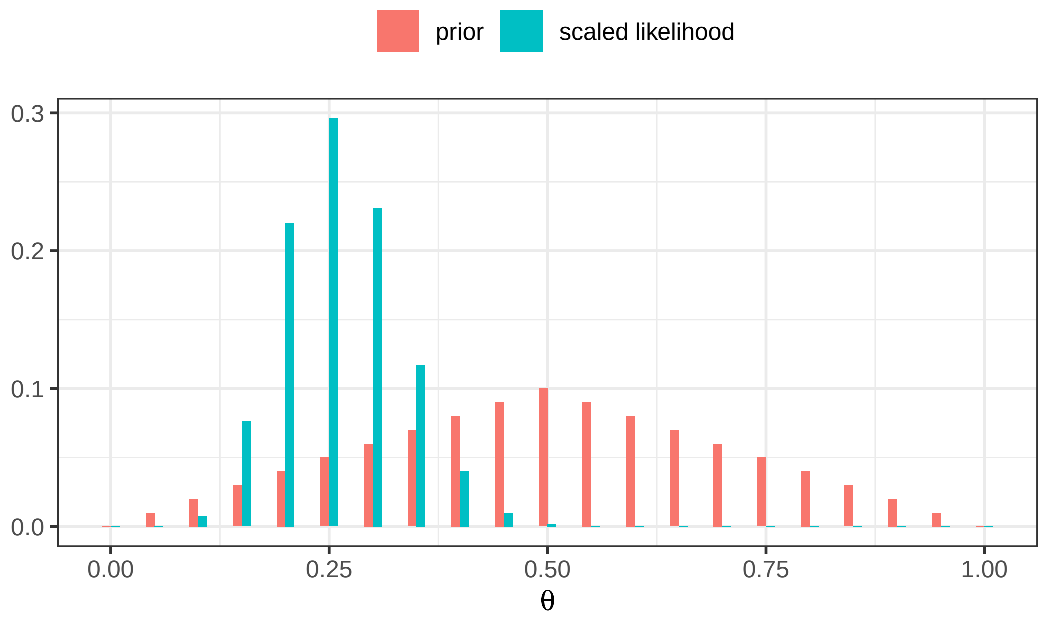
```r
# Alternatively, use the `posterior` package
data.frame(theta = post_samples) |>
    posterior::summarize_draws()
```

```
#> # A tibble: 1 x 10
#>   variable  mean median    sd   mad    q5   q95  rhat ess_bulk ess_tail
#>   <chr>    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
#> 1 theta    0.385    0.4 0.154 0.148  0.15  0.65  1.00    1030.     721.
```

3.5.4 Influence of Sample Size

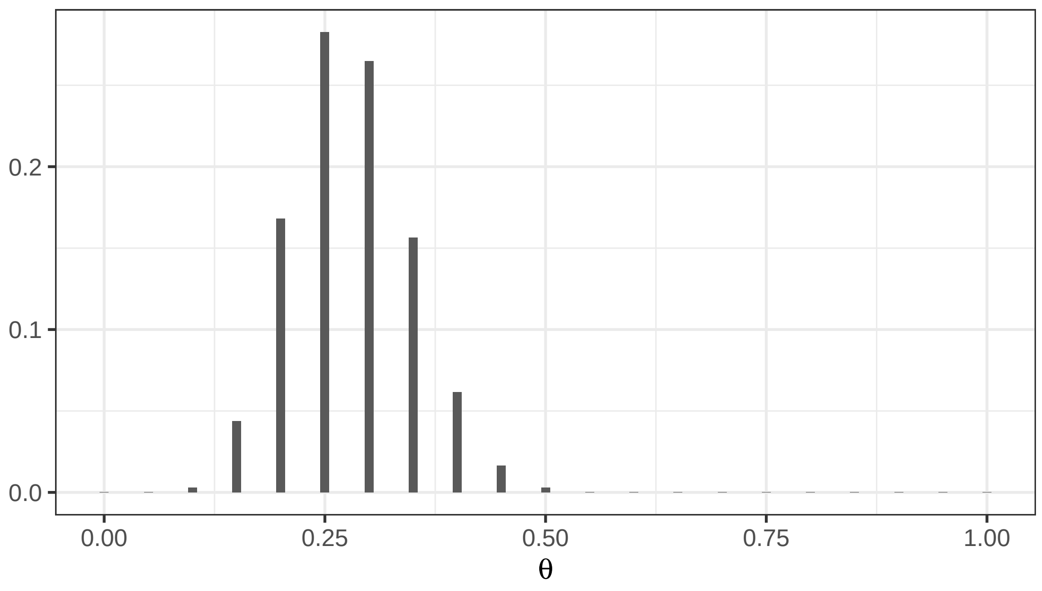
If, instead, we have $N = 40$ and $z = 10$, the posterior will be more similar to the likelihood.

```r
grid_df2 <- grid_df |>
    mutate(
        # Use our previously defined lik() function
        py_th = lik(th, num_flips = 40, num_heads = 10),
        # Product of prior and likelihood
        `prior x lik` = pth * py_th,
        # Scaled the posterior
        pth_y = `prior x lik` / sum(`prior x lik`)
    )
# Plot the prior/likelihood and the posterior
ggplot(data = grid_df2, aes(x = th)) +
    geom_col(aes(x = th - 0.005, y = pth, fill = "prior"),
        width = 0.01,
    ) +
    geom_col(aes(x = th + 0.005, y = py_th / sum(py_th),
        fill = "scaled likelihood"), width = 0.01,
    ) +
    labs(fill = NULL, y = NULL, x = expression(theta)) +
    theme(legend.position = "top")
```



```
ggplot(data = grid_df2, aes(x = th)) +
    geom_col(aes(x = th, y = pth_y), width = 0.01) +
    labs(
        fill = NULL, y = NULL, title = "Posterior",
        x = expression(theta)
    )
```

Posterior



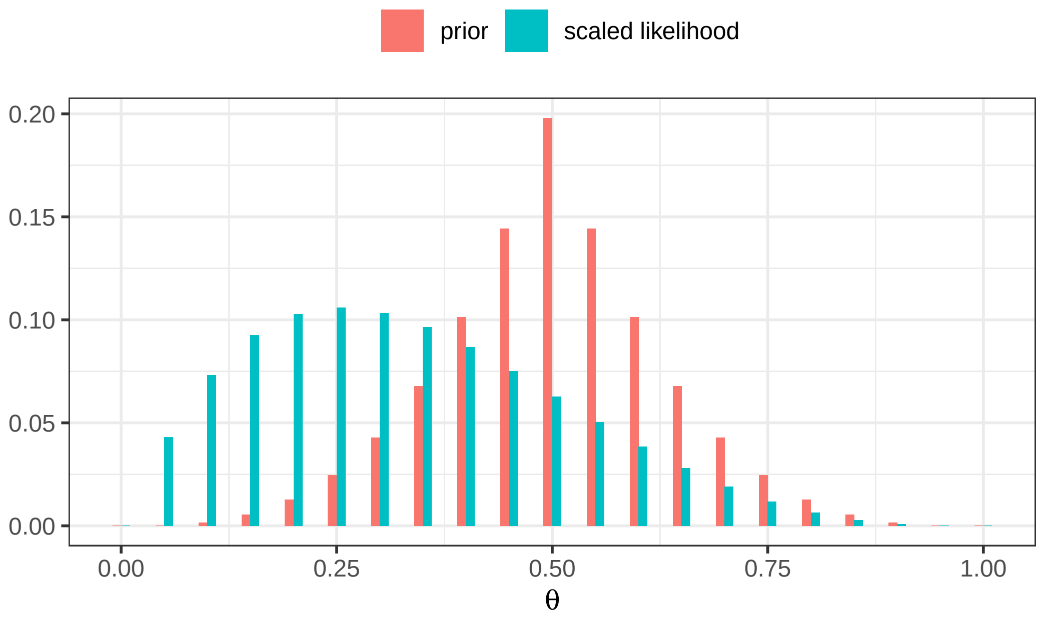
```
# Sample from the posterior
post_samples <- sample(
    grid_df2$th,
    size = 1000, replace = TRUE,
    prob = grid_df2$pth_y
)
summ_draw(post_samples)
```

```
#>       mean     median         sd        mad       ci.1       ci.9
#> 0.28085000 0.30000000 0.06542215 0.07413000 0.20000000 0.35000000
```

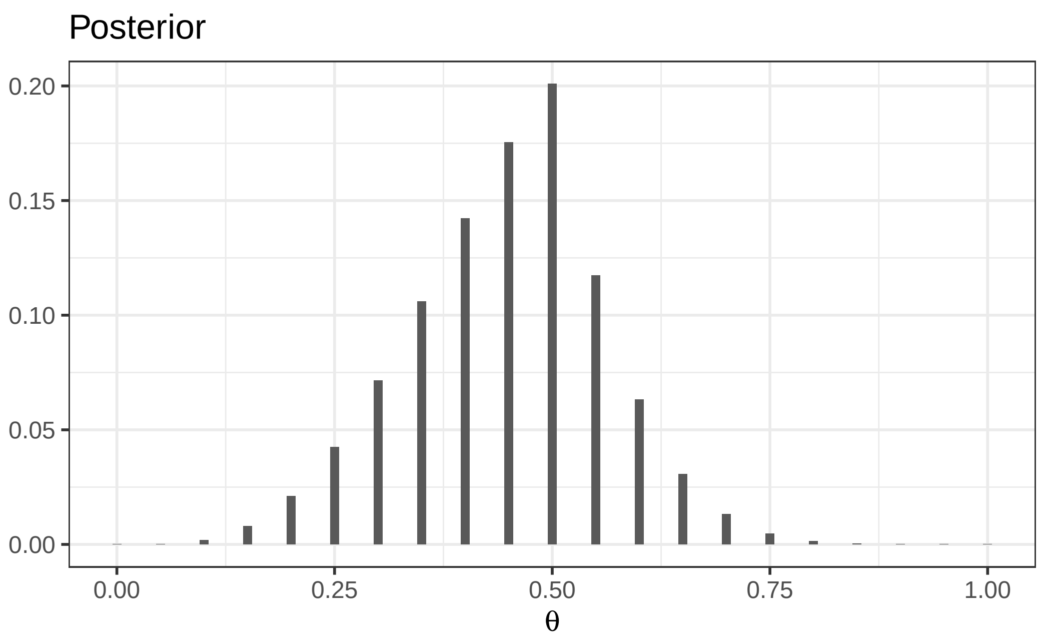
3.5.5 Influence of Prior

If we have a very strong prior concentrated at $\theta = .5$, but still with $N = 40$ and $z = 10$, the posterior will be more similar to the prior.

```
grid_df3 <- grid_df |>
    mutate(
        # stronger prior
        pth = pth ^ 3,
        # scale the prior to sume to 1
        pth = pth / sum(pth),
        # Use our previously defined lik() function
        py_th = lik(th, num_flips = 4, num_heads = 1),
        # Product of prior and likelihood
        `prior x lik` = pth * py_th,
        # Scaled the posterior
        pth_y = `prior x lik` / sum(`prior x lik`)
    )
# Plot the prior/likelihood and the posterior
ggplot(data = grid_df3, aes(x = th)) +
    geom_col(aes(x = th - 0.005, y = pth, fill = "prior"),
        width = 0.01,
    ) +
    geom_col(aes(x = th + 0.005, y = py_th / sum(py_th),
        fill = "scaled likelihood"), width = 0.01,
    ) +
    labs(fill = NULL, y = NULL, x = expression(theta)) +
    theme(legend.position = "top")
```



```
ggplot(data = grid_df3, aes(x = th)) +
    geom_col(aes(x = th, y = pth_y), width = 0.01) +
    labs(
        fill = NULL, y = NULL, title = "Posterior",
        x = expression(theta)
    )
```



```
# Sample from the posterior
post_samples <- sample(
    grid_df3$th,
    size = 1000, replace = TRUE,
    prob = grid_df3$pth_y
)
summ_draw(post_samples)
```

```
#>      mean    median        sd       mad      ci.1      ci.9
#> 0.4493000 0.4500000 0.1096656 0.0741300 0.3000000 0.6000000
```

```
# Alternatively, use the `posterior` package
data.frame(theta = post_samples) |>
    posterior::summarize_draws()
```

```
#> # A tibble: 1 x 10
#>   variable  mean median    sd    mad    q5   q95  rhat ess_bulk ess_tail
#>   <chr>    <dbl>  <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
#> 1 theta    0.449   0.45 0.110 0.0741  0.25   0.6  1.00    1001.     899.
```

3.5.6 Remark on Grid Approximation

In this note, we discretized θ into a finite number of grid points to compute the posterior, mainly for pedagogical purposes. A big limitation is that our posterior will have no density for values other than the chosen grid points. While increasing the number of grid points (e.g., 1,000) can give more precision, the result is still not truly continuous. A bigger issue is that the computation breaks down when there is more than one parameter; if there are p parameters, with 1,000 grid points per parameter, one needs to evaluate the posterior probability for $1,000^p$ grid points, which is not feasible even with modern computers. So more efficient algorithms, namely Markov chain Monte Carlo (MCMC) methods, will be introduced as we progress in the course.

Part III

Week 3

4 Beta-Bernoulli Model

4.1 Steps of Bayesian Data Analysis

Some authors described the process as "turning the Bayesian Crank," as the same workflow applies to a variety of research scenarios.

Adapted from Gelman et al. (2020), I conceptualize Bayesian data analysis as the following steps:

1. **Identify/Collect the data** required to answer the research questions.
    - As a general recommendation, it is helpful to **visualize** the data to get a sense of how they look, as well as to inspect for potential anomalies in the data collection.
2. **Choose an initial statistical model** for the data in relation to the research questions. The model should have some theoretical justification and have parameters that are meaningful for the research questions. However, it is unlikely that any chosen model will capture everything important in the data, and this initial model will be modified and expanded in later steps.
3. **Specify prior distributions** for the model parameters. Although this is a subjective endeavor, the priors chosen should be sensible to a skeptical audience.
4. **Check the prior distributions**. It is recommended you conduct a **prior predictive check**, by simulating fake data based on the chosen model and prior distributions. This is especially important for complex models as the parameters are more difficult to interpret.
5. **Obtain the posterior distributions** for the model parameters. As described below and later in the course, this can be obtained by analytical or various mathematical approximations.
    - For mathematical approximations, one should check the algorithms for **convergence** to make sure the results closely mimic the target posterior distributions.
6. Conduct a **posterior predictive check** to examine the fit between the model and the data, i.e., whether the chosen model with the estimated parameters generates predictions that deviate from the data being analyzed on important features.
7. It is unlikely that your initial model fully describes the major aspects of the data as pertaining to your research questions. Therefore, one should repeat steps 2 to 6 to **specify and compare different models**.

8. If the fit between the model and the data is deemed satisfactory, one can proceed to **interpret the results** in the context of the research questions. It is also important to **visualize the results** in ways that are meaningful for the analysis.

4.2 Beta-Bernoulli Example

We will be using a built-in data set in R about patients diagnosed with AIDS in Australia before July 1, 1991. Here is a description of the variables (from the R documentation):

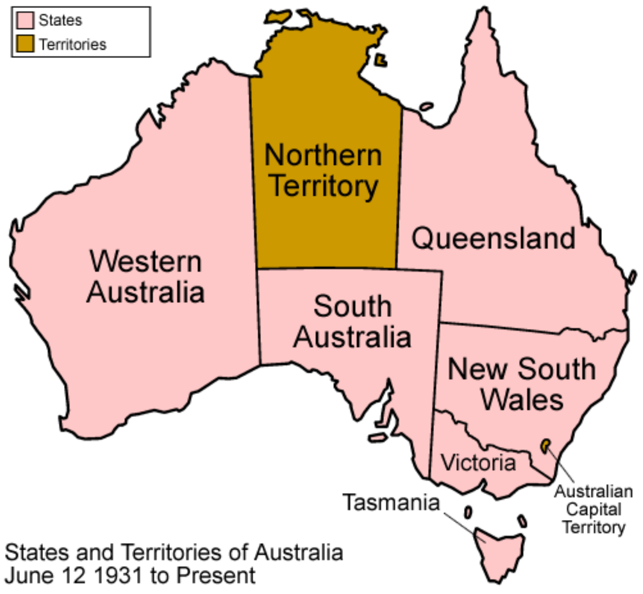


Figure 4.1: Figure from https://commons.wikimedia.org/wiki/File:Australia_states_1931-present.png

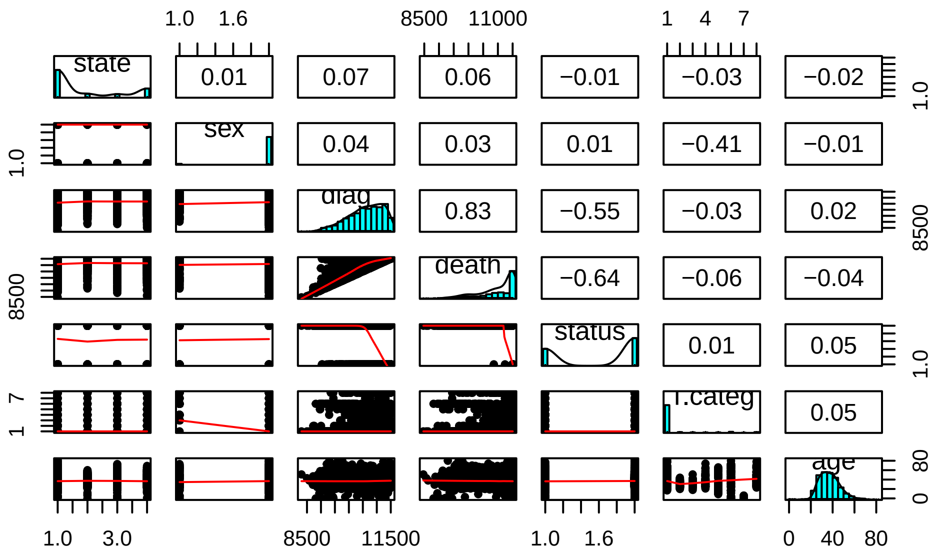
- `state`: Grouped state of origin: "NSW"includes ACT and "other" is WA, SA, NT, and TAS.
- `sex`: Sex of patient.
- `diag`:(Julian) date of diagnosis.
- `death`: (Julian) date of death or end of observation.
- `status`: "A" (alive) or "D" (dead) at end of observation.
- `T.categ`: Reported transmission category.
- `age`: Age (years) at diagnosis.

You should always first plot your data and get some summary statistics:

```r
data("Aids2", package = "MASS")
head(Aids2)
```

```
  state sex  diag death status T.categ age
1   NSW   M 10905 11081      D      hs  35
2   NSW   M 11029 11096      D      hs  53
3   NSW   M  9551  9983      D      hs  42
4   NSW   M  9577  9654      D    haem  44
5   NSW   M 10015 10290      D      hs  39
6   NSW   M  9971 10344      D      hs  36
```

```r
pairs.panels(Aids2, ellipses = FALSE)
```



We will be using the `status` variable. Our simple research question is:

> What was the death rate of AIDS in Australia when the data were collected?

4.2.1 Bernoulli Model

If we assume that the outcomes of the observations are exchangeable, meaning that the observations can be reordered in any way and still give the same inference, then one can choose a model:

$$y_i \sim \text{Bern}(\theta) \text{ for } i = 1, 2, \dots, N$$

- y_i = status of observation i (0 = "A", 1 = "D")
- N = number of patients in the data set
- θ = probability of "D"[1]

  The model states that: the sample data y follows a Bernoulli distribution with n with a parameter θ

When the data consist of binary observations, the variable is called a Bernoulli variable. It is conventional to denote one outcome as success and code it as 1, and the other as failure and code it as 0 (poor terminology, maybe, but that's by convention). Therefore, in the AIDS example, each observation is considered a "Bernoulli" outcome (Alive vs. Dead).

4.2.2 Exchangeability

To illustrate exchangeability in an example, say we take 6 rows in our data set:

```
     state sex  diag death status T.categ age
1      NSW   M 10905 11081      D      hs  35
31     NSW   F 10961 11504      A      id  30
1802   QLD   F  9495  9753      D   blood  66
1811   QLD   M 10770 11504      A    hsid  29
2129   VIC   M  8499  8568      D      hs  43
2137   VIC   M  9055  9394      D      hs  29
```

Now, when we reorder the column `status` to something like:

```
     state sex  diag death status T.categ age
1      NSW   M 10905 11081      D      hs  35
31     NSW   F 10961 11504      D      id  30
1802   QLD   F  9495  9753      D   blood  66
1811   QLD   M 10770 11504      A    hsid  29
2129   VIC   M  8499  8568      D      hs  43
2137   VIC   M  9055  9394      A      hs  29
```

[1]An additional thing to note for the Bernoulli/binomial model is that, instead of setting the prior on θ, sometimes we are more interested in setting the prior for a transformed parameter that has values between $-\infty$ and ∞, such as one on the *logit* scale (as related to logistic regression).

If the results are expected to be the same, then we say that the observations are assumed exchangeable. It happens when we assume that all observations have one common mean. However, if we think that there is a mean for females and a different mean for males, we cannot reorder the outcome randomly because they are no longer exchangeable (i.e., you cannot exchange a female score for a male score and expect to get the same results).

Exchangeability

A set of observations is said to be exchangeable if their joint probability distribution stays the same under all permutations. Roughly speaking, it means that the observations can be reordered and still provide the same inferences.

4.2.3 Check the Support

It is important to identify the *support* of the parameter, θ. Because θ is a probability, its support is $[0, 1]$, meaning it is continuous and can take any value from 0 to 1. For a continuous parameter, there are infinitely many possible values, and it is impossible to specify our beliefs for each value. So, more commonly, we choose a probability density function with the same support as the parameter to express our prior belief.

4.2.4 Conjugate Prior: Beta Distribution

A commonly used family of prior distributions for a Bernoulli/binomial model is the *Beta distribution*, which has two parameters. We can write the prior as

$$P(\theta) \sim \text{Beta}(a, b)$$

a and b are the two hyperparameters. Here are a few examples:

You will notice that when $a > b$, there is more density closer to the right region (i.e., larger θ), and vice versa. Also, the variance decreases when a and b become larger.[2]

A nice interpretation of a and b in a Beta prior distribution is to consider

- $a - 1$ = number of prior 'successes' (e.g., "D")
- $b - 1$ = number of prior 'failures' (e.g., "A")

[2]The $\text{Beta}(1/2, 1/2)$ distribution is called a *Jeffreys prior* (https://en.wikipedia.org/wiki/Jeffreys_prior), which is derived according to some statistical principles for different models. One big advantage of a Jeffreys prior is that it is **invariant**, meaning that the prior will stay the same even under reparameterization. However, like conjugate priors, Jeffreys prior limits the choice of prior even when true prior information is available.

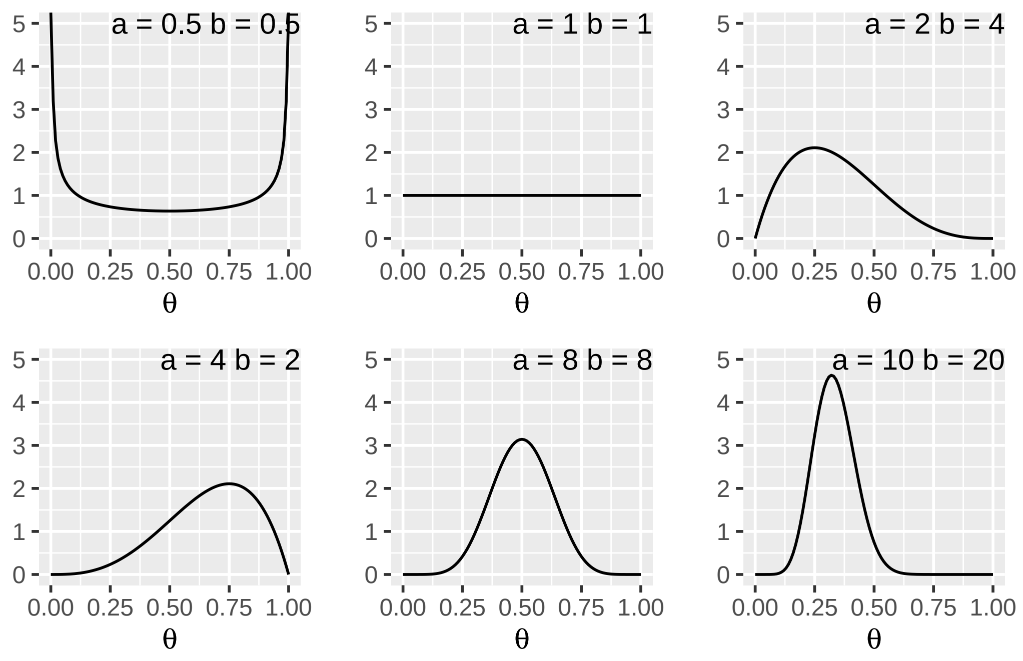


Figure 4.2: Beta distributions with different a and b values

Therefore, with $\text{Beta}(1, 1)$, one has seen 0 prior success and 0 failure, meaning that there is no prior information (i.e., *noninformative*). Therefore, it makes sense that all θ values are equally likely. On the other hand, if one chooses $\text{Beta}(10, 20)$, one has seen 9 prior successes and 19 prior failures, so one has quite a lot of prior information (indeed more than the data with only 10 observations), so this is a *strong* prior.

> The smaller the variance of the prior distribution, the stronger one's belief before looking at the data, the more prior information

So by manipulating the values of a and b, which are sometimes called *hyperparameters*, you can control the shape of the prior distribution as well as its strength, so it is quite flexible. Another advantage of using a beta prior is that it is a *conjugate prior* of the Bernoulli model, which means that the posterior distribution $P(\theta \mid y)$ is also a beta distribution, the same as the prior distribution, although with different parameter values.

Conjugate Prior

For a specific model, conjugate priors yield posterior distributions in the same distribution family as the priors

Conjugacy greatly simplifies the computational burden for Bayesian analyses, so conjugate priors are almost the only ones used in earlier literature. However, this limited the applications of Bayesian methods, as for many problems, no conjugate priors can provide a realistic representation of one's belief. Modern Bayesian analysis instead relies on *simulation-based* methods to approximate the posterior distribution, which can accommodate almost any kind of prior distribution. Aside from a few examples in this note, mainly for pedagogical purposes, we will be using simulation-based methods in the coming weeks.

Proof of Conjugacy*

To derive the form of the posterior, first recognize that the Beta distribution has the form:

$$\begin{aligned} P(\theta) &= \mathrm{B}^{-1}(a,b)\theta^{a-1}(1-\theta)^{b-1} \\ &\propto \theta^{a-1}(1-\theta)^{b-1} \end{aligned}$$

Where $\mathrm{B}(\cdot)$ is the beta function which is not very important for the class. As the density function is a function of θ, it suffices to write only the terms that involve θ. Similarly,

$$P(\mathbf{y} \mid \theta) \propto \theta^z(1-\theta)^{N-z}.$$

Therefore,

$$\begin{aligned} P(\theta \mid \mathbf{y}) &\propto P(y \mid \theta)P(\theta) \\ &\propto \theta^z(1-\theta)^{N-z}\theta^{a-1}(1-\theta)^{b-1} \\ &= \theta^{a+z-1}(1-\theta)^{b+N-z-1}. \end{aligned}$$

If we let $a^* = a + z$, $b^* = b + N - z$, we can see that $P(\theta \mid \mathbf{y})$ is in the same form as the prior with a and b replaced by a^* and b^*. Therefore, the posterior is also a beta distribution. So the beta distribution is a conjugate prior for the Bernoulli model.

In this example, we will choose a *weakly informative* Beta(2, 2) prior, which represents a weak belief as below:

```r
ggplot(data.frame(th = c(0, 1)), aes(x = th)) +
    stat_function(fun = dbeta, args = list(shape1 = 2, shape2 = 2)) +
    ylim(0, 3) +
    labs(y = "", x = expression(theta), title = "Beta(2, 2)")
```

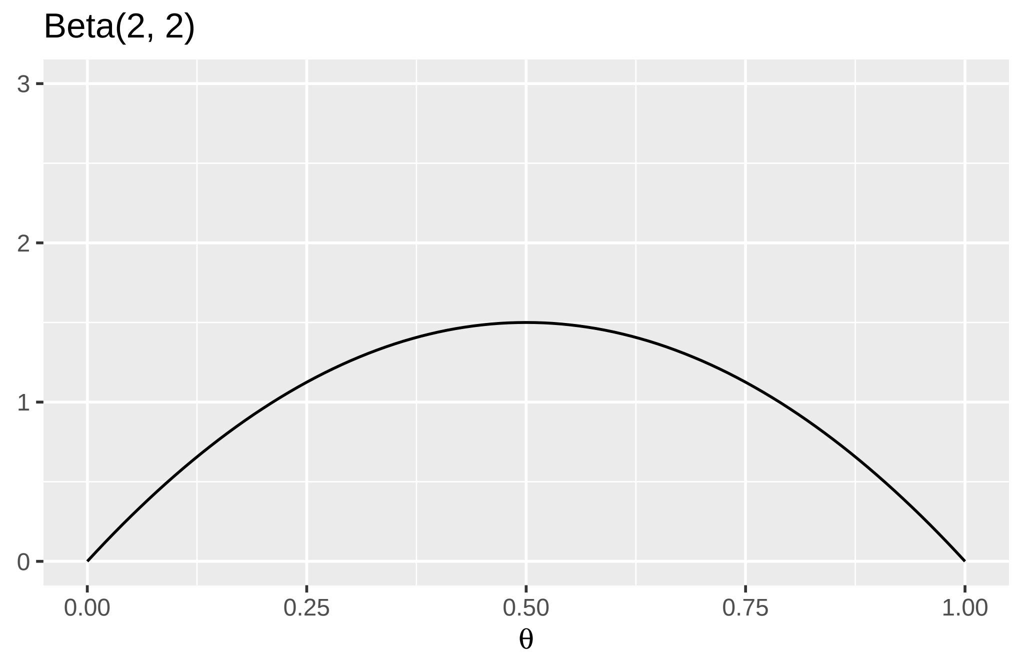


Figure 4.3: A weakly informative Beta(2, 2) prior

! Don't Be Stubborn

> A good prior should give a non-zero probability/density for all possible values of a parameter

Otherwise, if the prior density for some parameter values is zero, the posterior density will be zero, regardless of how much the data support those parameter values

4.2.5 Data

```
count(Aids2, status)
```

```
  status    n
1      A 1082
2      D 1761
```

The likelihood function is highly concentrated. I ran into some numerical issues as the computation gave zero, so I plotted the log-likelihood instead.

```r
loglik <- function(th, N = 1082 + 1761, z = 1761) {
    z * log(th) + (N - z) * log(1 - th)
}
ggplot(data.frame(th = c(0.61, 0.63)), aes(x = th)) +
    stat_function(fun = loglik, n = 501) +
    labs(x = expression(theta), y = "Log-likelihood")
```

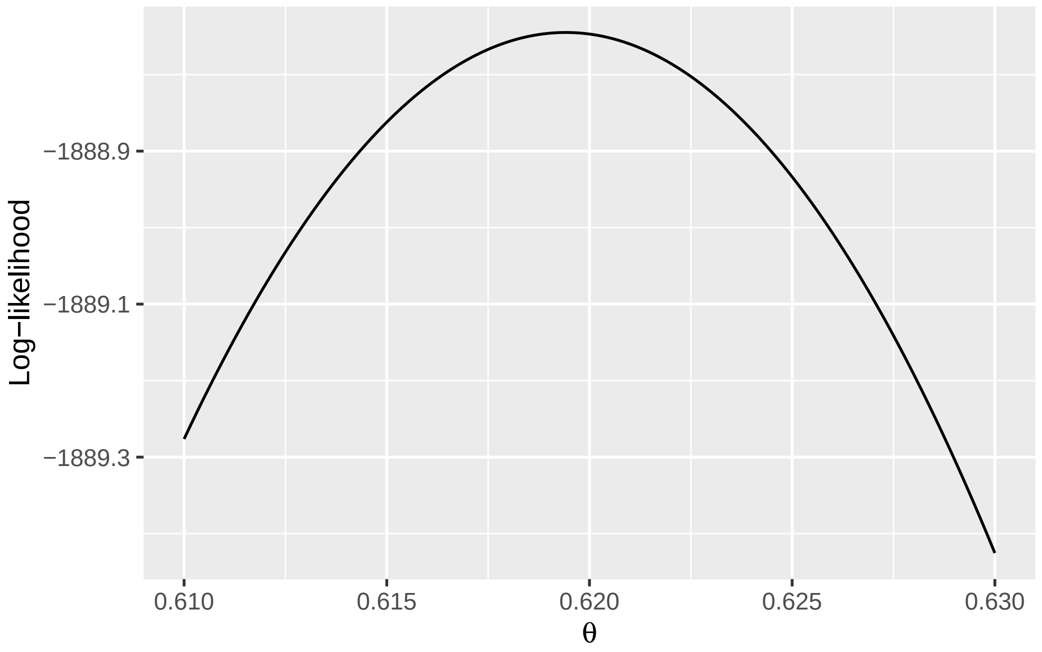


Figure 4.4: Log-likelihood function of the theta parameter

Note I only show a range of [0.610, 0.630] for the x-axis, which contains where the likelihood (thus also the log-likelihood) peaked.

4.2.6 Posterior

Based on the conjugacy, the posterior of θ is Beta(1,807, 1,116). As we are using a conjugate prior, the posterior is also a Beta distribution:

$$P(\theta \mid y) \sim \text{Beta}(a + z, b + N - z),$$

which is a distribution for $a + z - 1$ successes and $b + N - z$ failures. This makes perfect sense as our prior information has $a - 1$ successes and $b - 1$ failures, and from our data, we have y successes and $n - y$ failures, so our updated belief is based on adding up those successes and failures.

4.2.7 Summarize the posterior

```r
set.seed(2119)
num_draws <- 1000
sim_theta <- rbeta(num_draws, shape1 = 1807, shape2 = 1116)
c(`Bayes estimate` = mean(sim_theta),
  `Posterior median` = median(sim_theta),
  `Posterior SD` = sd(sim_theta),
  `MAD` = mad(sim_theta),
  `90% Credible interval (equal-tailed)` = quantile(sim_theta, probs = c(.1, .9)),
  `90% HDI` = HDInterval::hdi(sim_theta, credMass = .9))
```

```
                             Bayes estimate
                                0.618209694
                           Posterior median
                                0.618382945
                               Posterior SD
                                0.008829290
                                        MAD
                                0.009254166
90% Credible interval (equal-tailed).10%
                                0.606862338
90% Credible interval (equal-tailed).90%
                                0.628990588
                               90% HDI.lower
                                0.604051851
                               90% HDI.upper
                                0.632766654
```

4.2.8 Posterior Predictive Check

Now, we need to know whether the model fits the data well. We do not have much to check for a Bernoulli model if we only have the `status` variable. However, as there is information for other variables, we can use them to check the exchangeability assumption. For example, we can ask whether the data from different state categories are exchangeable. The death rate across the 4 state categories are

```
       status
state      A    D
  NSW    664 1116
  Other   107  142
  QLD      78  148
  VIC     233  355

       status
state           A         D
  NSW   0.3730337 0.6269663
  Other 0.4297189 0.5702811
  QLD   0.3451327 0.6548673
  VIC   0.3962585 0.6037415
```

We can now generate predictions from our posterior distribution and model.

```r
plist <- vector("list", 12L)
plist[[1]] <- ggplot(
    Aids2,
    aes(x = state, y = mean(status == "D"), fill = state)
) +
    geom_bar(stat = "identity") +
    guides(fill = "none") +
    labs(x = "Observed data", y = "Number of Deaths") +
    theme(axis.title.x = element_text(color = "red")) +
    ylim(0, 1200)
for (i in 1:11) {
    # Get the a value from posterior samples
    theta_post <- rbeta(1, 1763, 1084)
    # For each plausible theta value, generate a status variable
    status_new <- sample(c("D", "A"), nrow(Aids2),
        replace = TRUE,
        prob = c(theta_post, 1 - theta_post)
    )
    df_new <- Aids2 |>
        mutate(status = factor(status_new))
    plist[[i + 1]] <- plist[[1]] %+% df_new +
        labs(x = paste("Simulated data", i)) +
        theme(axis.title.x = element_text(color = "black"))
}
gridExtra::grid.arrange(grobs = plist, nrow = 3)
```

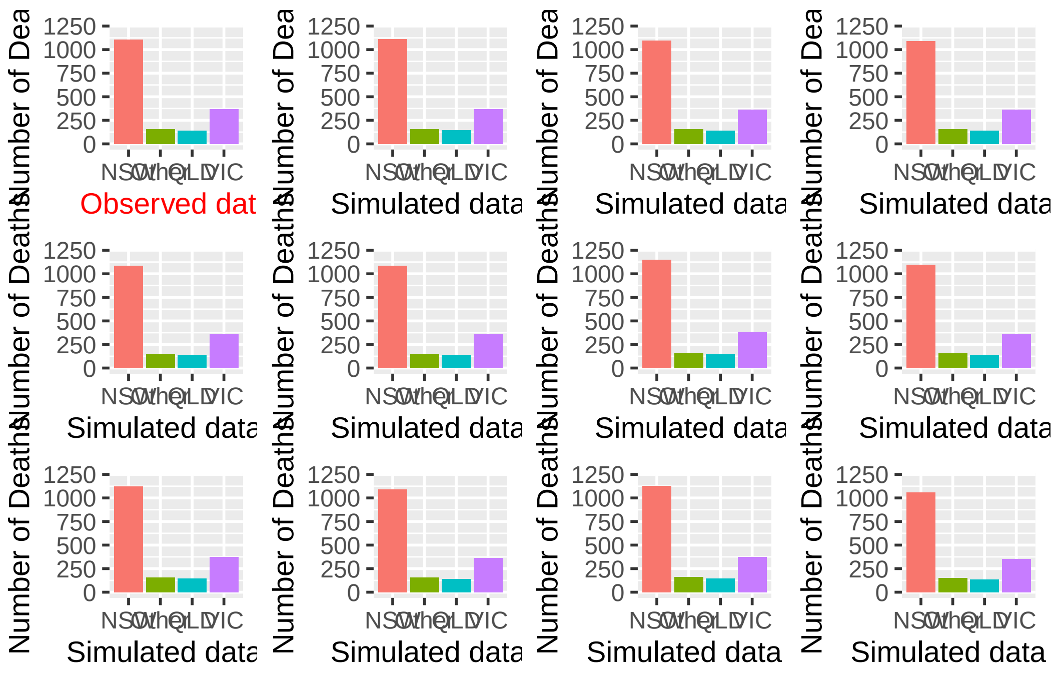


Figure 4.5: Posterior predictive check by comparing the observed data with 11 simulated data sets based on the model

So the observed data (the first subplot) look similar to the simulated data. We can also conduct a posterior predictive check by a test statistic for subgroups. Here, we will use the `bayesplot` package and look at fit across groups:

```
# Draw posterior samples of theta
post_sample <- rbeta(1e4, 1807, 1116)
# Initialize a S by N matrix to store the simulated data
y_tilde <- matrix(NA,
                  nrow = length(post_sample),
                  ncol = length(Aids2$status))
for (s in seq_along(post_sample)) {
    theta_s <- post_sample[s]
    status_new <- sample(c("D", "A"), nrow(Aids2),
        replace = TRUE,
        prob = c(theta_s, 1 - theta_s)
    )
    y_tilde[s,] <- as.numeric(status_new == "D")
}
bayesplot::ppc_stat_grouped(
    as.numeric(Aids2$status == "D"),
    yrep = y_tilde,
    group = Aids2$state
)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

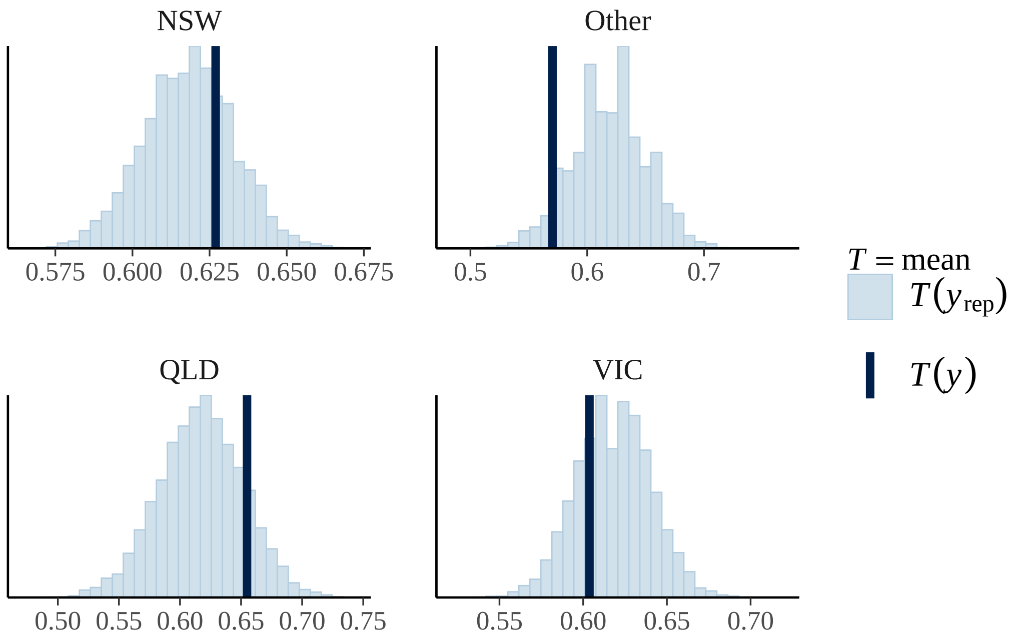


Figure 4.6: Posterior predictive check by states

If the fit is good, the mean, indicated by the darker line, should be within the simulated distribution based on the model. So the model that assumes observations are exchangeable across states is not too off, although it seems fitting less well for `Other` states.

4.2.8.1 Another check on `age`

```
# Create an age group indicator
age50 <- factor(Aids2$age > 50, labels = c("<= 50", "> 50"))
# Draw posterior samples of theta
post_sample <- rbeta(1e4, 1807, 1116)
# Initialize a S by N matrix to store the simulated data
```

```r
y_tilde <- matrix(NA,
                  nrow = length(post_sample),
                  ncol = length(Aids2$status))
for (s in seq_along(post_sample)) {
    theta_s <- post_sample[s]
    status_new <- sample(c("D", "A"), nrow(Aids2),
        replace = TRUE,
        prob = c(theta_s, 1 - theta_s)
    )
    y_tilde[s,] <- as.numeric(status_new == "D")
}
bayesplot::ppc_stat_grouped(
    as.numeric(Aids2$status == "D"),
    yrep = y_tilde,
    group = age50
)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

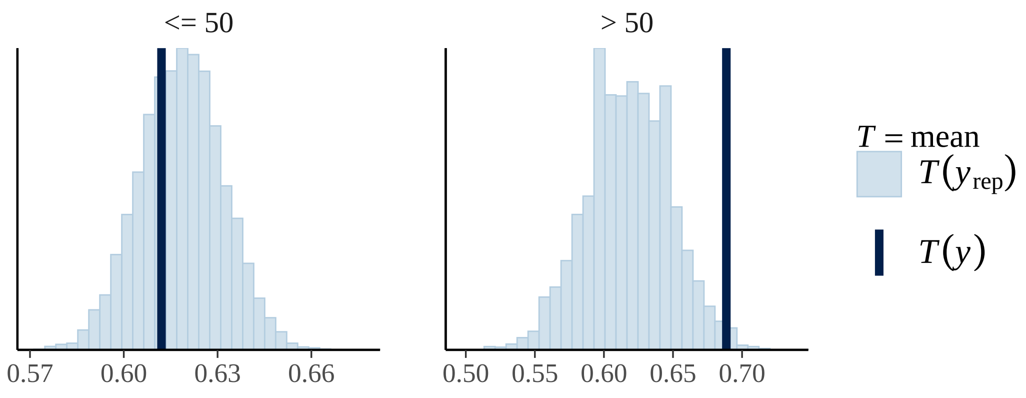


Figure 4.7: Posterior predictive check by age groups (<= 50 vs. > 50)

As can be seen, the model seems off for those aged 50+.

4.2.9 Comparison to frequentist results

Using maximum likelihood, the estimated death rate would be $\hat{\theta} = 1761/2843 = 0.62$, with a standard error (SE) of $\sqrt{0.62(1-0.62)/n} = 0.0091$, with a 90% confidence interval of $[0.6, 0.63]$, which is similar to the interval with Bayesian inference.

4.2.10 Sensitivity to different priors

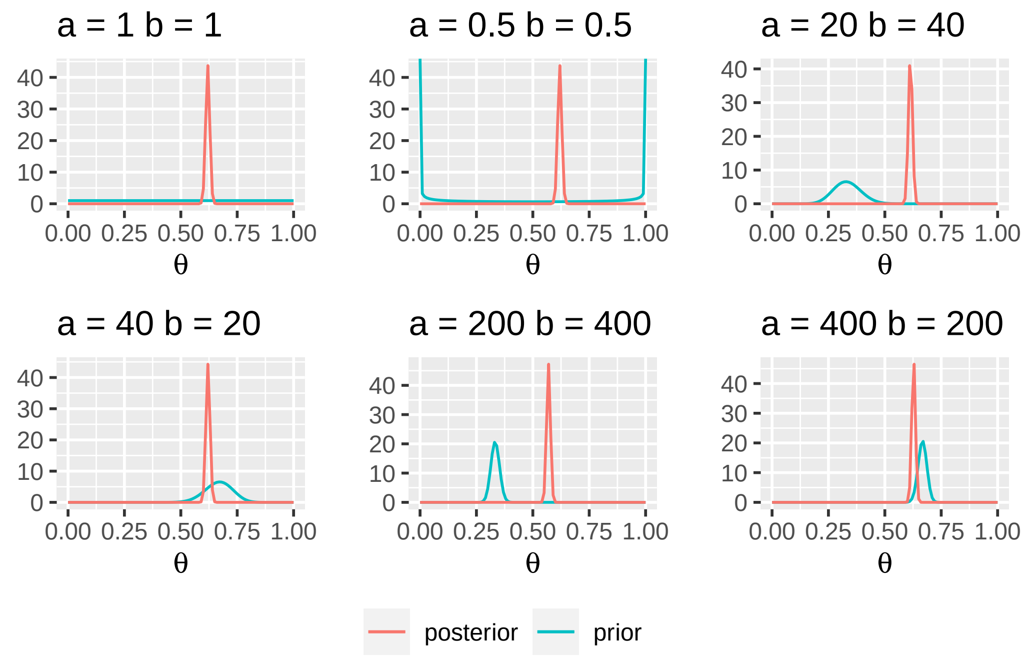


Figure 4.8: Sensitivity of posterior to different priors

You can see one needs (a) a very strong prior (equivalent to 600 data points) and (b) the prior and the data not agreeing to get a substantially different conclusion.

5 Beta-Bernoulli Model With Stan

In the previous lecture, we fitted a Beta-Bernoulli model using Gibbs sampling with our own R code. While this is doable in this relatively simple model (it only has one parameter), for more complex models, Gibbs sampling and other MCMC methods (to be introduced in a later class) require quite a lot of programming. Fortunately for us, we have some readily available software for doing MCMC. While it is probably overkill for the Beta-Bernoulli model, we will learn working with Stan by using it to analyze the Beta-Bernoulli model. Specifically, we will learn to:

1. Install Stan and the R package `cmdstanr` for communicating between R and Stan;
2. Write Stan code for the Beta-Bernoulli model;
3. Draw posterior samples using Stan;
4. Summarize and plot the posterior samples.

We will talk about convergence of MCMC, which is an extremely important topic, in a later class.

5.1 Installing Stan

1. Follow the steps in https://mc-stan.org/cmdstanr/ to install the `cmdstanr` package.
2. Load the `cmdstanr` package, and run `install_cmdstan()`

```
library(cmdstanr)
install_cmdstan()
```

3. If you run into an error in the previous step related to C++ toolchain, follow the directions here: https://mc-stan.org/cmdstanr/articles/cmdstanr.html

5.2 Fitting a Beta-Bernoulli Model in Stan

5.2.1 Data Import

From last class:

```r
data("Aids2", package = "MASS")
head(Aids2)
```

```
  state sex  diag death status T.categ age
1   NSW   M 10905 11081      D      hs  35
2   NSW   M 11029 11096      D      hs  53
3   NSW   M  9551  9983      D      hs  42
4   NSW   M  9577  9654      D    haem  44
5   NSW   M 10015 10290      D      hs  39
6   NSW   M  9971 10344      D      hs  36
```

5.2.2 Writing Stan syntax

Stan has its own syntax and is different from R. For example, we want to fit the following Beta-Bernoulli model:

$$\begin{aligned} y_i &\sim \text{Bern}(\theta) \text{ for } i = 1, 2, \dots, N \\ P(\theta) &\sim \text{Beta}(2, 2) \end{aligned}$$

and the model can be written in Stan as follows:

```stan
data {
  int<lower=0> N;  // number of observations
  array[N] int<lower=0,upper=1> y;  // y
}
parameters {
  real<lower=0,upper=1> theta;  // theta parameter
}
model {
  theta ~ beta(2,2);  // prior: Beta(2, 2)
  y ~ bernoulli(theta);  // model: Bernoulli
}
```

Save the above model syntax in a separate file ending in `.stan`. For example, I saved the syntax in the file `beta-bernoulli.stan` in a folder named `stan_code`.

In Stan, anything after `//` denotes comments (like `#` in R) and will be ignored by the program. In each block (e.g., `data {}`), a statement should end with a semicolon (`;`). There are several blocks in the above Stan code:

- `data`: The data input for Stan is usually not only an R data frame, but a list that includes other information, such as sample size, number of predictors, and prior scales. Each type of data has an input type, such as
  - `int` = integer,
  - `real` = numbers with decimal places,
  - `matrix` = 2-dimensional data of real numbers,
  - `vector` = 1-dimensional data of real numbers, and
  - `array` = 1- to many-dimensional data. For example, we use `array[N]` for the data type of `y`, because it is a vector, but each element is an integer (and cannot take decimals).

  We can set the lower and upper bounds so that Stan can check the input data. In the above, we used `<lower=0,upper=1>`.
- `parameters`: The parameters to be estimated
- `transformed parameters`: optional variables that are transformations of the model parameters. It is usually used for more advanced models to allow more efficient MCMC sampling.
- `model`: It includes expressions of prior distributions for each parameter and the likelihood for the data. There are many possible distributions that can be used in Stan.
- `generated quantities`: Any quantities that are not part of the model but can be computed from the parameters for every iteration. Examples include posterior generated samples, effect sizes, and log-likelihood (for fit computation). We will see an example later.

5.2.3 Compiling the Stan model from R

To compile the model, we can call the `cmdstan_model` function in `cmdstanr`:

```r
bern_mod <- cmdstan_model("stan_code/beta-bernoulli.stan")
```

5.2.4 Posterior Sampling

We need to first prepare the data for input to Stan. In the Stan code, we have two objects in the data block: N and y, so we need to have a list of two elements in R:

```
Aids2_standata <- list(
    N = nrow(Aids2),
    y = as.integer(Aids2$status == "D")  # integer
)
```

Now we can draw posterior samples:

```
fit <- bern_mod$sample(Aids2_standata)
```

```
Running MCMC with 4 sequential chains...

Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1 finished in 0.0 seconds.
Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2 finished in 0.0 seconds.
Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
Chain 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3 Iteration: 1900 / 2000 [ 95%]  (Sampling)
Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3 finished in 0.0 seconds.
Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4 finished in 0.0 seconds.

All 4 chains finished successfully.
Mean chain execution time: 0.0 seconds.
Total execution time: 0.7 seconds.
```

For this simple model, this takes less than a second.

5.2.5 Summarizing and plotting the posterior samples

```r
# Actual posterior samples
fit$draws("theta", format = "draws_df")
```

```
# A draws_df: 1000 iterations, 4 chains, and 1 variables
   theta
1   0.63
2   0.62
3   0.63
```

```
4   0.63
5   0.62
6   0.62
7   0.62
8   0.63
9   0.62
10  0.62
# ... with 3990 more draws
# ... hidden reserved variables {'.chain', '.iteration', '.draw'}
```

```r
# Summary table
fit$summary()
```

```
# A tibble: 2 x 10
  variable      mean    median      sd     mad       q5      q95  rhat ess_bulk
  <chr>        <dbl>     <dbl>   <dbl>   <dbl>    <dbl>    <dbl> <dbl>    <dbl>
1 lp__     -1892.    -1892.    0.668   0.282   -1894.    -1.89e+3  1.00    2161.
2 theta        0.619     0.619 0.00886 0.00868     0.604  6.34e-1  1.00    1294.
# i 1 more variable: ess_tail <dbl>
```

```r
# Histogram
fit$draws("theta") |>
    mcmc_hist()
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

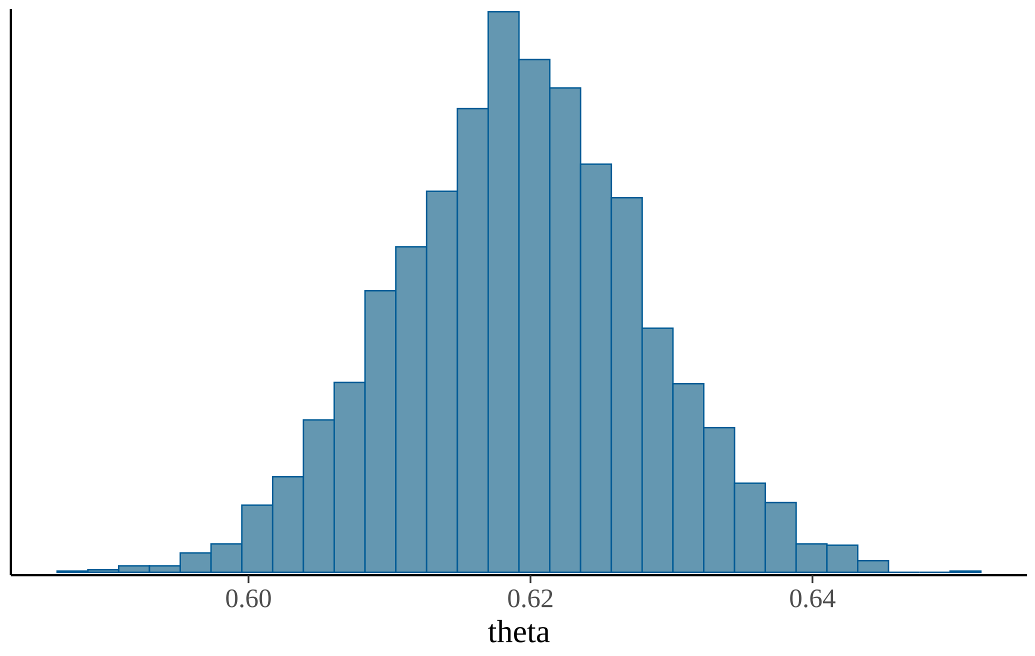


Figure 5.1: Posterior distribution of the parameter.

The results are similar to those in last class.

5.3 Prior Predictive Check

In Bayesian analyses, it is recommended to check both the prior and the model. This can be done by

1. Prior predictive check: Simulating data from the prior distribution, and see if the simulated data fit our prior belief.
2. Posterior predictive check: Simulating data from the posterior distribution, and see if the simulated data are comparable to the observed data.

We will use the following Stan code to do prior and posterior predictive checks, which has an additional generated quantity block to obtain

a. `prior_theta`: simulated values of θ based on the prior distribution
b. `prior_ytilde`: simulated data based on the prior distribution of θ
c. `ytilde`: simulated data based on the posterior distribution of θ

```
data {
  int<lower=0> N;  // number of observations
  array[N] int<lower=0,upper=1> y;  // y
}
parameters {
  real<lower=0,upper=1> theta;  // theta parameter
}
model {
  theta ~ beta(2, 2);  // prior: Beta(2, 2)
  y ~ bernoulli(theta);  // model: Bernoulli
}
generated quantities {
  real prior_theta = beta_rng(2, 2);
  array[N] int prior_ytilde;
  array[N] int ytilde;
  for (i in 1:N) {
    ytilde[i] = bernoulli_rng(theta);
    prior_ytilde[i] = bernoulli_rng(prior_theta);
  }
}
```

```
bern_pp_mod <- cmdstan_model("stan_code/beta-bernoulli-pp.stan")
bern_pp_fit <- bern_pp_mod$sample(
    Aids2_standata,
    refresh = 500  # show progress every 500 iterations
)
```

```
Running MCMC with 4 sequential chains...

Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1 finished in 0.8 seconds.
Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2 finished in 0.8 seconds.
Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3 finished in 0.8 seconds.
Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4 finished in 0.9 seconds.

All 4 chains finished successfully.
Mean chain execution time: 0.8 seconds.
Total execution time: 3.7 seconds.
```

With Stan, because we obtained 4,000 prior/posterior draws (the software default) of θ, we also obtained 4,000 simulated data sets. We can see the first one, based on only the prior distribution (i.e., $\theta \sim \text{Beta}(2, 2)$):

```r
bern_pp_fit$draws("prior_ytilde", format = "draws_df")[1, ] |>
    as.numeric() |>
    table()
```

```

   0    1 
2171  675 
```

> Note that the data set has more 1's than 0's. Our prior is weak, which means that it allows for a lot of variation in how the data would look.

The distribution of simulated *data* based on the prior distribution of the parameters is called the *prior predictive distribution*. Mathematically, we write it as

$$P(\tilde{y}) = \int P(\tilde{y}|\theta)P(\theta)\ \mathrm{d}\theta$$

Because we have 4,000 data sets, it is not easy to visualize all individual data points. Instead, we can visualize some summary statistics of the simulated data. Here, we will choose the proportion of deaths (i.e., the mean of the variable) in each simulated data set:

```r
bern_pp_fit$draws("prior_ytilde", format = "draws_matrix") |>
    ppd_stat(stat = "mean")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

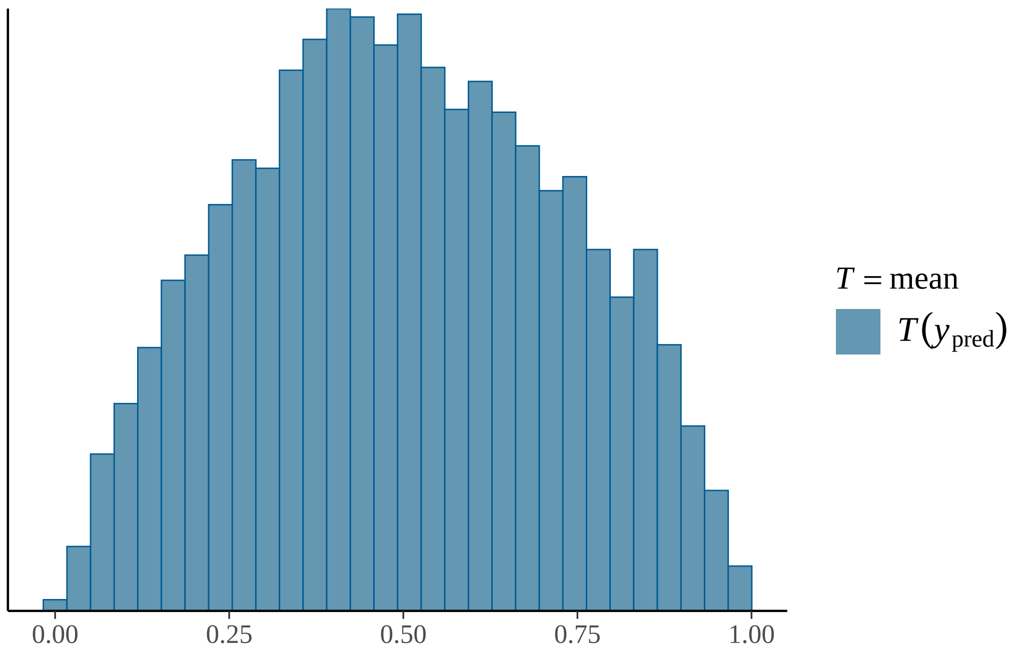


Figure 5.2: Prior predictive distribution of the sample mean.

This represents our prior belief about the proportion of deaths without looking at the actual data. If this doesn't seem to match our belief, we may want to modify our prior distribution and do the prior predictive check again, until the simulated data matches our actual prior belief.

5.4 Posterior Predictive Check

5.4.1 Check 1: Sample Mean

$$P(\tilde{y}|y) = \int P(\tilde{y}|\theta, y)P(\theta|y)\,\mathrm{d}\theta$$

The difference here is that we use the posterior distribution $P(\theta|y)$ instead of the prior distribution $P(\theta)$. We can obtain the posterior predictive distribution of the death rate, and indicate the actual data in the plot:

```r
bern_pp_fit$draws("ytilde", format = "draws_matrix") |>
    ppc_stat(y = Aids2_standata$y, stat = "mean")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

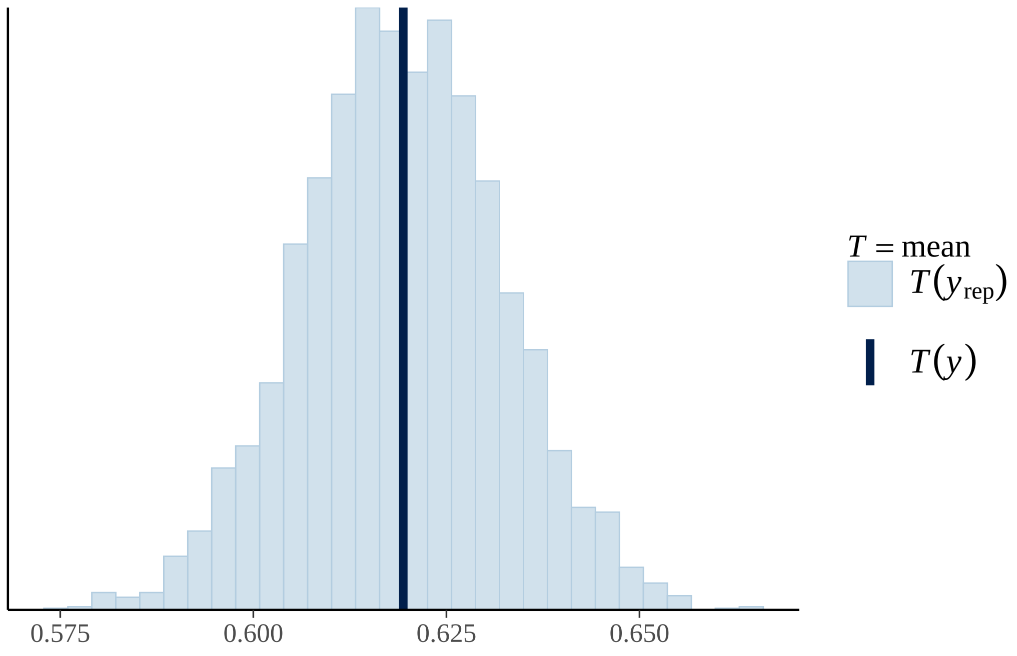


Figure 5.3: Posterior predictive distribution of the sample mean.

5.4.2 Check 2: Sample Mean by Age Group

```r
# Create binary indicator of two age groups
age50 <- factor(Aids2$age > 50, labels = c("<= 50", "> 50"))
bern_pp_fit$draws("ytilde", format = "draws_matrix") |>
    ppc_stat_grouped(y = Aids2_standata$y, group = age50,stat = "mean")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

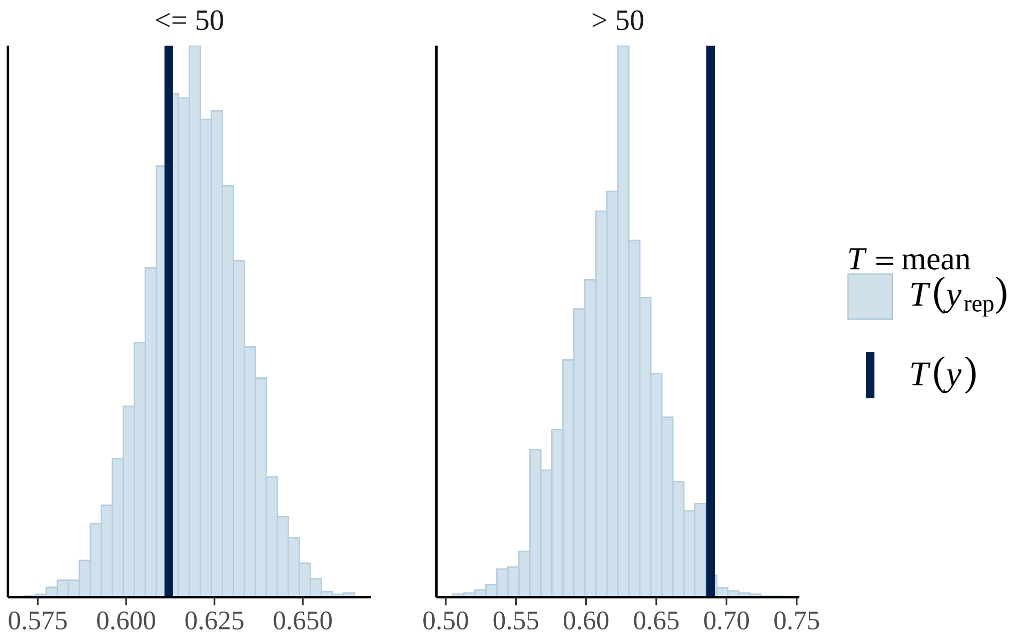


Figure 5.4: Posterior predictive distribution of the sample mean by age group.

6 Poisson Model

Please note: This document uses count data on fatal police shootings.

I came across this data set from https://andrewpwheeler.com/2021/01/11/checking-a-poisson-distribution-fit-an-example-with-officer-involved-shooting-deaths-wapo-data-r-functions/

As explained here, the data are by the Washington Post in an effort to record every fatal shooting in the United States by a police officer since January 1, 2015.

6.1 Research Question

> What's the rate of fatal police shootings in the United States per year?

6.2 Data Import and Pre-Processing

```
# Import data
fps_dat <- read_csv(
    "https://github.com/washingtonpost/data-police-shootings/raw/master/v2/fatal-police-shoo
)
```

We first count the data by year

```
# Create a year column
fps_dat <- fps_dat |>
    mutate(year = format(date, format = "%Y"))
# Filter out the latest year
fps_1523 <- filter(fps_dat, year != max(year))
count(fps_1523, year)
```

```
# A tibble: 9 x 2
  year      n
  <chr> <int>
1 2015    995
2 2016    959
3 2017    984
4 2018    991
5 2019    994
6 2020   1020
7 2021   1050
8 2022   1095
9 2023   1162
```

Our interest is the rate of occurrence of fatal police shootings per year. Denote this as θ. The *support* of θ is $[0, \infty)$.

A *Poisson model* is usually a starting point for analyzing count data in a fixed amount of time. It assumes that the data follow a Poisson distribution with a fixed rate parameter:

$$P(y \mid \theta) \propto \theta^y \exp(-\theta),$$

where the data can be any non-negative integers (no decimals).

6.3 Choosing a Prior

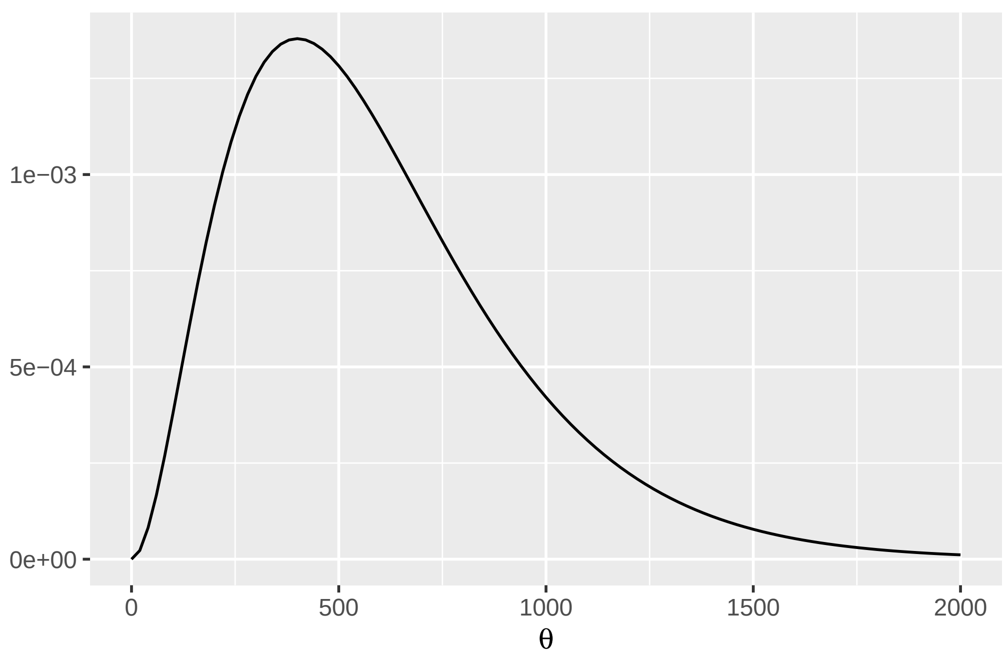
The Gamma distribution has support: $[0, \infty)$, and is a conjugate family to the Poisson model. The Gamma distribution has the form

$$P(\theta) \propto \theta^{a-1} \exp(-b\theta),$$

where a is the prior incidence rate, and b is the number of prior data points to control for the prior strength. Here, without much prior knowledge, I would simply guess there is one fatal shooting per state per month, so 600 shootings per year, but my belief is pretty weak, so I will assume a prior b of 1 / 200 (one observation is one year). The a will be 600 * $b = 3$.

Here's a plot:

```r
ggplot(data.frame(th = c(0, 2000)), aes(x = th)) +
    stat_function(fun = dgamma,
    args = list(shape = 3, rate = 1 / 200)) +
    labs(y = "", x = expression(theta))
```

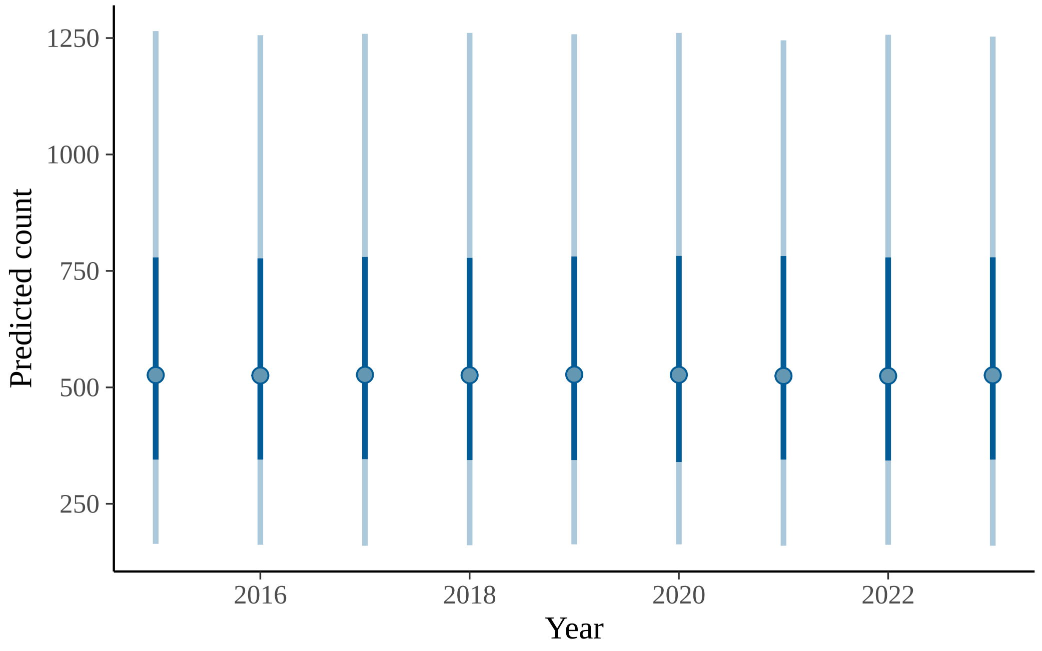


1e−03
5e−04
0e+00
0
500
1000
1500
2000
θ

6.3.4 Prior predictive check

Here I plot simulated trends from the prior distribution.

```r
# Extract predicted y from prior
fit$draws("prior_ytilde", format = "draws_matrix") |>
    ppd_intervals(x = 2015:2023) +
    labs(x = "Year", y = "Predicted count")
```

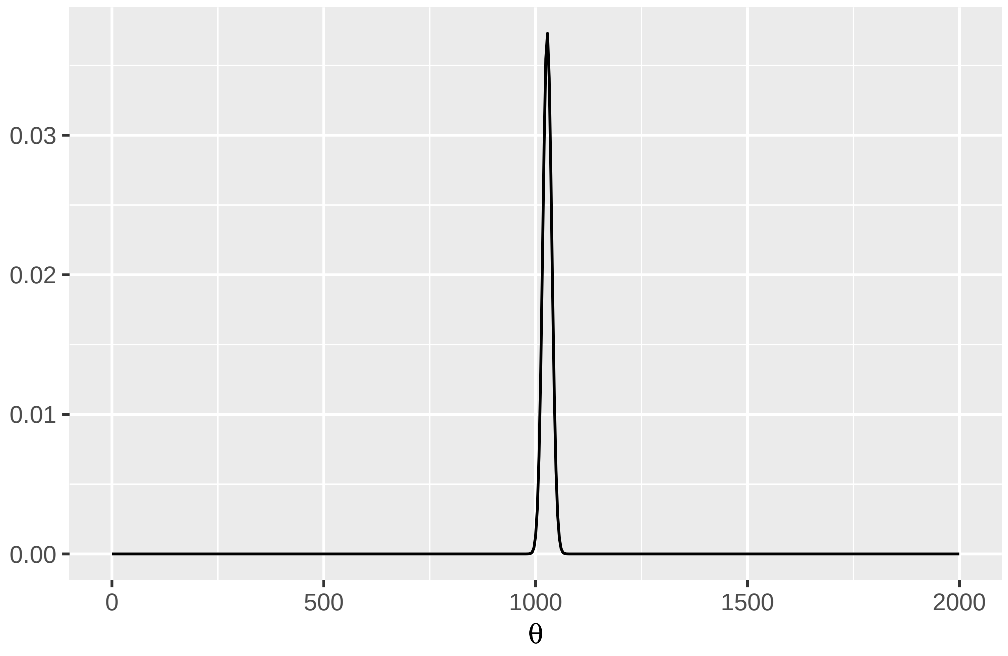


The check is on whether the numbers seem reasonably reflective of my knowledge.

6.4 Posterior

With a conjugate prior, the posterior distribution is Gamma$(a + \sum_{i=1}^{N} y_i, b + N)$.

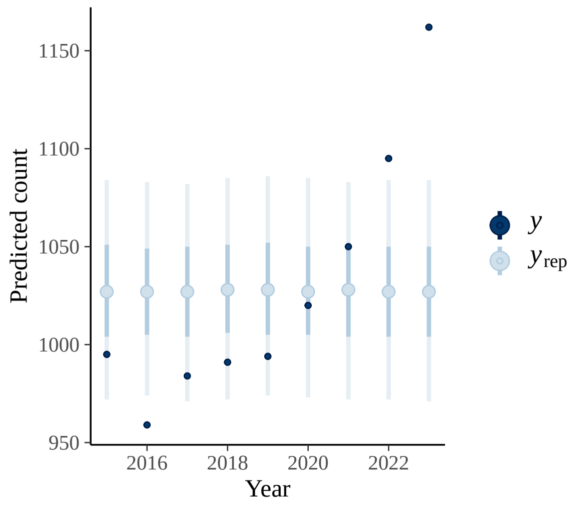
```r
ggplot(data.frame(th = c(0, 2000)), aes(x = th)) +
    stat_function(fun = dgamma,
    args = list(shape = 3 + nrow(fps_1523),
                rate = 1 / 200 + fps_standata$N), n = 501) +
    labs(y = "", x = expression(theta))
```



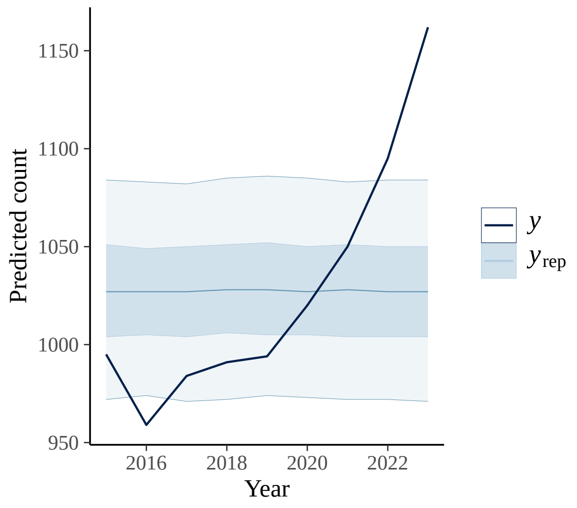
6.5 Posterior Predictive Check

Plot predicted data from the posterior against observed data

```r
# Extract predicted y from posterior
fit$draws("ytilde", format = "draws_matrix") |>
    ppc_intervals(
        y = fps_standata$y,
        x = 2015:2023
    ) +
    labs(x = "Year", y = "Predicted count")
# We can also use `bayesplot::ppc_ribbon()`
fit$draws("ytilde", format = "draws_matrix") |>
    ppc_ribbon(
        y = fps_standata$y,
        x = 2015:2023
    ) +
    labs(x = "Year", y = "Predicted count")
```



(a) Interval plots.



(b) Ribbon plots.

Figure 6.1: Posterior predictive check.

From Figure 6.1, one can see that the fit is not good, as there is a large gap between the model prediction and the observed data from recent years. This suggests a need to incorporate the time trend in the model.

6.6 Summary of Posterior

For now, we will proceed with interpreting the posterior distribution, despite the apparent misfit.

```
(summ_theta <- fit$summary("theta"))
```

```
# A tibble: 1 x 10
  variable  mean median    sd   mad    q5   q95  rhat ess_bulk ess_tail
  <chr>    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
1 theta    1028.  1028.  10.7  10.8 1010. 1045.  1.00    1630.    2119.
```

So the estimated rate is 1,028 per year, with a 90% CI [1,010, 1,045].

Part IV

Week 4

7 Hierarchical Models

Although many statistical models can be fitted using Bayesian or frequentist methods, some models are more naturally used in the Bayesian framework. One class of such models is the family of **hierarchical models.** Consider situations when the data contain *clusters*, such as multiple data points in each of many participants, or multiple participants in each of several treatment conditions. While it is possible to run J Bayesian analyses for the J subsets of the data, it is usually more efficient to pool the data together. In this approach, each cluster j has some parameters θ_j, and these J θ values themselves come from a common distribution. Figure 7.1 shows a graphical representation of the concept of pooling. This is the same idea as multilevel modeling, a topic we will discuss more later in the course.

> **!** A hierarchical model is one in which some higher-level distributions govern one or more parameters, and those higher-level distributions are characterized by *hyperparameters* and can be assigned *hyperpriors*.
> Hierarchical models are commonly used to study and account for individual differences in some model parameters.

In this note, you will see two examples, one from a textbook (Kruschke, 2015) with a hierarchical Bernoulli/binomial model, and another from a classic data set with eight schools, modelled by a hierarchical normal model.

7.1 Hierarchical Bernoulli/Binomial

We will first consider a therapeutic touch example (Kruschke, 2015, Chapter 9). Therapeutic touch is a technique in alternative medicine to relieve pain, but scientific evidence does not support its effectiveness. The data here are from an experiment where the experimenter randomly hovered their hand over either the participant's left or right hand, and the participant had to guess which hand was being hovered without seeing. This is repeated 10 times for each participant. There are a total of 28 participants in the dataset.

Previously, we have seen the Bernoulli model for N outcomes (i.e., whether the guess is correct):

$$y_i \sim \text{Bern}(\theta), \text{for } i = 1, \dots, N$$

We assumed exchangeability with θ being the participant's "ability" to guess correctly.

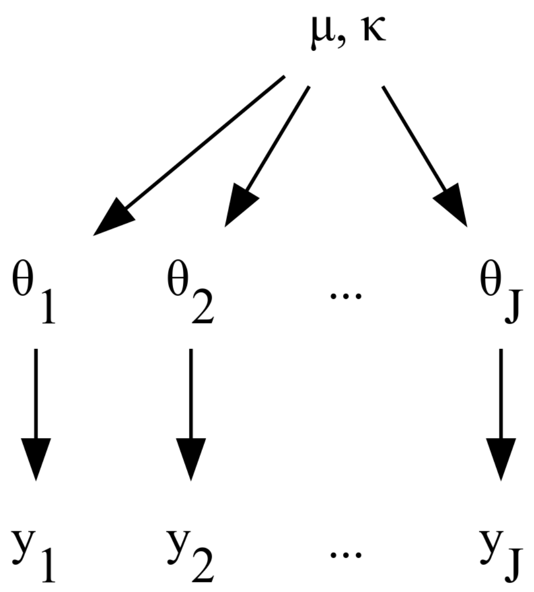


Figure 7.1: A graphical representation of a hierarchical binomial model.

Alternative Parameterization of Beta

In the last class, we have used the Beta(a, b) prior for a Bernoulli outcome, such that

$$P(\theta \mid a, b) \propto \theta^{a-1}(1-\theta)^{b-1}.$$

However, in hierarchical models to be discussed later, it is beneficial to consider another way to express the Beta distribution, in terms of the *prior mean*, $\mu = a/(a+b)$, $\mu \in [0, 1]$, and the concentration, $\kappa = a + b$, $\kappa \in [0, \infty)$. So, instead of the above formula, we can write

$$P(\theta \mid \mu, \kappa) \propto \theta^{\mu\kappa-1}(1-\theta)^{(1-\mu)\kappa-1}.$$

The two expressions represent exactly the same distribution, but just in terms of parameters of different meanings. Therefore, they are referred to as **different parameterization** of the Beta distribution.

7.1.1 Multiple Bernoulli = Binomial

With N exchangeable Bernoulli observations, an equivalent but more efficient way to code the model is to use the *binomial* distribution. Let $z = \sum_{i=1}^{N} y_i$, then

$$z \sim \text{Bin}(N, \theta)$$

7.1.2 Multiple Binomial Observations

Now, because we have multiple participants, we could study whether each participant had noticeable differences in their guessing ability. We can use a binomial model for each participant, from participant 1 to participant J:

$$\begin{aligned} z_1 &\sim \text{Bin}(N_1, \theta_1) \\ z_2 &\sim \text{Bin}(N_2, \theta_2) \\ &\vdots \\ z_J &\sim \text{Bin}(N_J, \theta_J) \end{aligned}$$

Or instead of writing J equations, we can use the subscript j to refer to each participant:

$$z_j \sim \text{Bin}(N_j, \theta_j)$$

If we believe that all participants have the same ability, then we could consider the model

$$z_j \sim \text{Bin}(N_j, \theta),$$

which still contains only one parameter, θ. However, if we have reason to believe that the coins have different biases, then we should have

$$z_j \sim \text{Bin}(N_j, \theta_j),$$

with parameters $\theta_1, \dots, \theta_j$.

We can assign priors to each θ_j. However, suppose our prior belief is that there's something common among the different participants (e.g., they're all human beings), so they come from a common distribution. In that case, we can have common parameters for the prior distributions of the θs:

$$\theta_j \sim \text{Beta\_Proportion}(\mu, \kappa),$$

note I use Beta_Proportion to denote the mean parameterization. Here, we express the prior belief that the **mean ability** of the different participants is μ, and how each participant differs from the mean depends on κ. Now, μ and κ are hyperparameters. We can assign some fixed values to μ and κ, as if we know what the average ability is. However, the power of the hierarchical model is that we can put priors (or hyper priors) on μ and κ, and obtain posterior distributions of them, based on what the data say.

What priors to use for μ and κ? μ is relatively easy because it is the mean ability; if we put a Beta prior for each participant's ability, we can again use a Beta prior for the mean ability. κ is more challenging. A larger κ means that the participants' abilities are more similar to each other. We can perform a prior predictive check to see what the data look like. As a starting point, some textbook (e.g., chapter 9 of Kruschke, 2015) suggested using Gamma(0.01, 0.01). So the full model in our case, with a weak Beta(1.5, 1.5) prior on μ, is

Model:

$$\begin{aligned} z_j &\sim \text{Bin}(N_j, \theta_j) \\ \theta_j &\sim \text{Beta2}(\mu, \kappa) \end{aligned}$$

Prior:

$$\begin{aligned} \mu &\sim \text{Beta}(1.5, 1.5) \\ \kappa &\sim \text{Gamma}(0.01, 0.01) \end{aligned}$$

We will import the data and fit the model

```r
# Data file from GitHub
tt_url <- paste0(
    "https://github.com/boboppie/kruschke-doing_bayesian_data_analysis/",
    "raw/master/2e/TherapeuticTouchData.csv"
)
tt_dat <- read.csv(tt_url)
# Get aggregated data by summing the counts
tt_agg <- tt_dat |>
    group_by(s) |>
    summarise(y = sum(y),  # total number of correct
              n = n())
# Plot proportion correct distribution
p1 <- ggplot(tt_agg, aes(x = y / n)) +
    geom_histogram(binwidth = .1) +
    labs(x = "Proportion Correct")
```

7.1.3 Stan Code

```stan
data {
  int<lower=0> J;  // number of clusters (e.g., studies, persons)
  array[J] int y;  // number of "1"s in each cluster
```

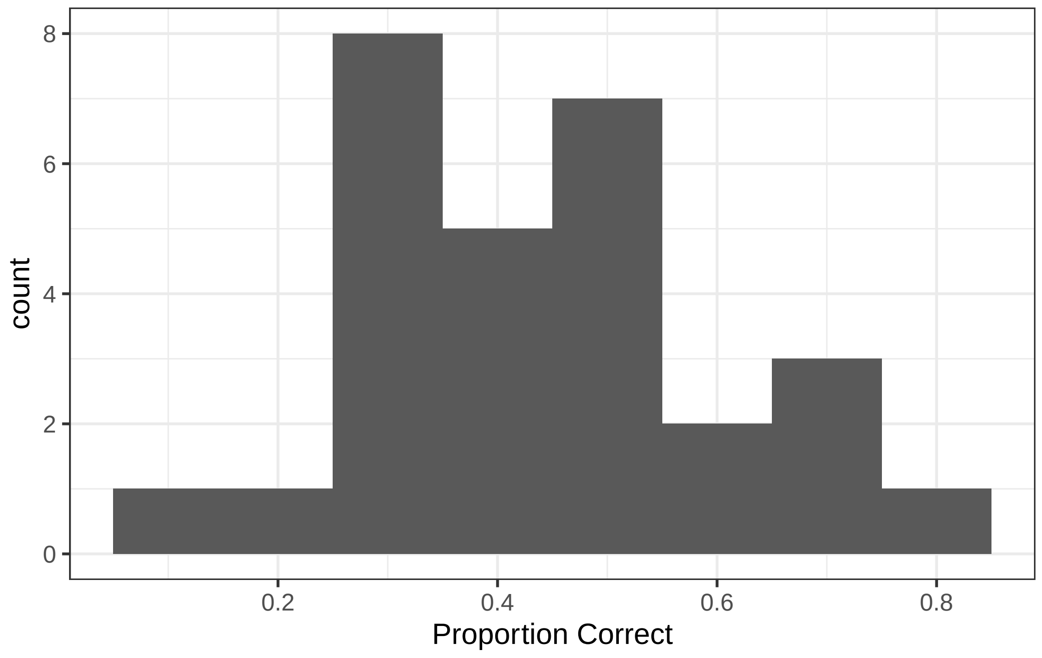


Figure 7.2: Distribution of proportions of correct responses across participants.

```
  array[J] int N;  // sample size for each cluster
}
parameters {
  // cluster-specific probabilities
  vector<lower=0, upper=1>[J] theta;
  real<lower=0, upper=1> mu;  // overall mean probability
  real<lower=0> kappa;        // overall concentration
}
model {
  y ~ binomial(N, theta);  // each observation is binomial
  // Priors
  theta ~ beta_proportion(mu, kappa);
  mu ~ beta(1.5, 1.5);      // weak prior
  kappa ~ gamma(.1, .1);  // prior recommended by Kruschke
}
generated quantities {
  // Prior and prior predictive
  real<lower=0, upper=1> prior_mu = beta_rng(1.5, 1.5);
  real<lower=0> prior_kappa = gamma_rng(.1, .1);
  vector<lower=0, upper=1>[J] prior_theta;
  for (j in 1:J) {
    prior_theta[j] = beta_proportion_rng(prior_mu, prior_kappa);
  }
  array[J] int prior_ytilde = binomial_rng(N, prior_theta);
  // Posterior predictive
  array[J] int ytilde = binomial_rng(N, theta);
}
```

```
hbin_mod <- cmdstan_model("stan_code/hierarchical-binomial.stan")
```

7.1.4 Prior predictive

You can use Stan to sample the prior and obtain the prior predictive distribution; here, I show how to do it in R, with a Gamma(.01, .01) prior on κ:

```
set.seed(1706)
plist <- vector("list", 12L)
plist[[1]] <- p1 +
    labs(x = "Observed data") +
    theme(axis.title.x = element_text(color = "red"))
num_subjects <- 28
for (s in 1:11) {
    # Get prior values of mu and kappa
    mu_s <- rbeta(1, shape1 = 1.5, shape2 = 1.5)
    kappa_s <- rgamma(1, shape = 0.01, rate = 0.01)
    # Generate theta
    theta <- rbeta(num_subjects,
                   shape1 = mu_s * kappa_s,
                   shape2 = (1 - mu_s) * kappa_s)
    # Generate data
    new_y <- rbinom(num_subjects, size = tt_agg$n, prob = theta)
    plist[[s + 1]] <-
        p1 %+% mutate(tt_agg, y = new_y) +
        labs(x = paste("Simulated data", s)) +
        theme(axis.title.x = element_text(color = "black"))
}
gridExtra::grid.arrange(grobs = plist, nrow = 3)
```

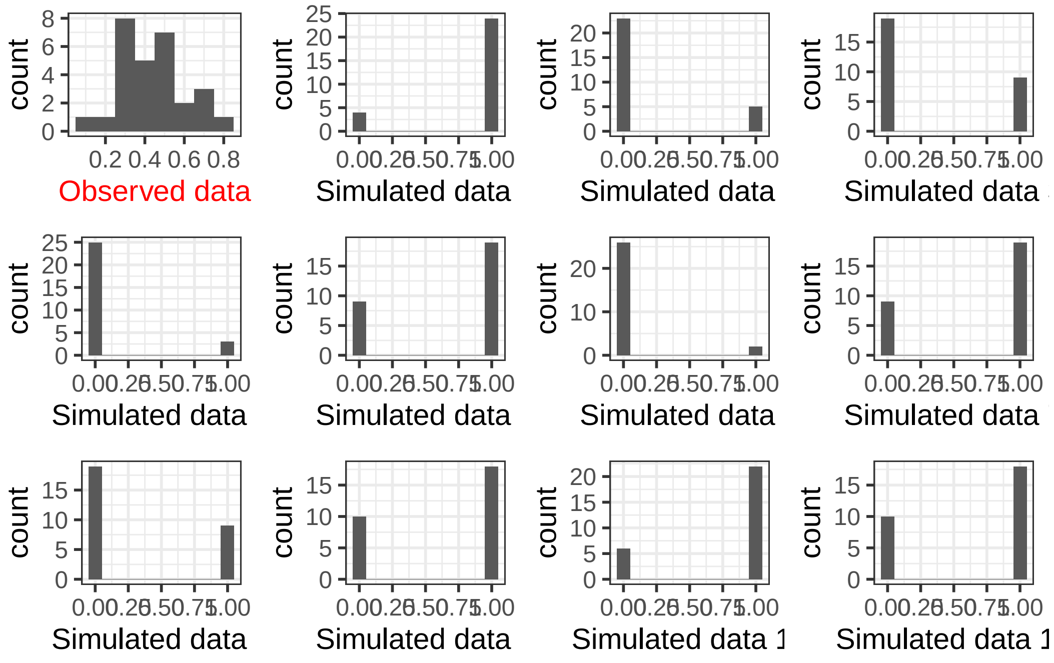


Figure 7.3: Prior predictive distribution.

The prior on κ is not very realistic because it pushes the bias to either 0 or 1. Using something like Gamma(0.1, 0.1) or Gamma(2, 0.01) may be more reasonable (you can try it out yourself).

7.1.5 Calling Stan

```
tt_fit <- hbin_mod$sample(
    data = list(J = nrow(tt_agg),
                y = tt_agg$y,
                N = tt_agg$n),
    seed = 1716,  # for reproducibility
    refresh = 1000
)
```

```
Running MCMC with 4 sequential chains...

Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1 finished in 0.3 seconds.
Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)

Chain 2 Informational Message: The current Metropolis proposal is about to be rejected becau

Chain 2 Exception: beta_proportion_lpdf: Location parameter is 1, but must be less than 1.00

Chain 2 If this warning occurs sporadically, such as for highly constrained variable types l

Chain 2 but if this warning occurs often then your model may be either severely ill-conditio

Chain 2

Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2 finished in 0.2 seconds.
Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3 finished in 0.3 seconds.
Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)

Chain 4 Informational Message: The current Metropolis proposal is about to be rejected becau

Chain 4 Exception: beta_proportion_lpdf: Location parameter is 0, but must be positive! (in

Chain 4 If this warning occurs sporadically, such as for highly constrained variable types l

Chain 4 but if this warning occurs often then your model may be either severely ill-conditio

Chain 4
Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4 finished in 0.2 seconds.

All 4 chains finished successfully.
Mean chain execution time: 0.3 seconds.
Total execution time: 1.6 seconds.
```

You can explore the convergence and posterior distributions using the `shinystan` package

```
shinystan::launch_shinystan(tt_fit)
```

7.1.6 Table of coefficients

```
tt_fit$summary(c("theta", "mu", "kappa")) |>
    # Use `knitr::kable()` for tabulation
    knitr::kable(digits = 2)
```

Table 7.1: Posterior summary of hierarchical binomial model for the therapeutic touch example.

variable	mean	median	sd	mad	q5	q95	rhat	ess_bulk	ess_tail
theta[1]	0.31	0.31	0.10	0.10	0.15	0.47	1	3268.39	2404.91
theta[2]	0.35	0.35	0.10	0.10	0.19	0.51	1	3741.46	2422.69
theta[3]	0.39	0.39	0.10	0.10	0.23	0.55	1	4744.99	2794.69
theta[4]	0.39	0.39	0.10	0.10	0.23	0.55	1	4859.60	3056.80
theta[5]	0.39	0.39	0.10	0.10	0.22	0.55	1	4817.40	2485.43
theta[6]	0.39	0.38	0.10	0.10	0.23	0.55	1	5260.92	2493.09
theta[7]	0.39	0.38	0.10	0.10	0.23	0.55	1	5586.84	2968.73
theta[8]	0.39	0.39	0.10	0.10	0.23	0.55	1	4926.08	3008.76
theta[9]	0.39	0.39	0.10	0.10	0.23	0.55	1	4802.85	2558.79
theta[10]	0.39	0.39	0.10	0.10	0.22	0.55	1	4743.85	2910.25
theta[11]	0.43	0.42	0.10	0.09	0.27	0.59	1	5041.77	3066.93
theta[12]	0.42	0.42	0.09	0.10	0.27	0.58	1	6055.56	2896.89
theta[13]	0.43	0.42	0.10	0.10	0.27	0.59	1	4898.48	3112.24
theta[14]	0.43	0.42	0.10	0.10	0.27	0.59	1	5909.48	3065.15
theta[15]	0.43	0.42	0.10	0.10	0.27	0.59	1	6304.56	3049.53
theta[16]	0.46	0.46	0.10	0.10	0.31	0.62	1	5642.01	3230.67
theta[17]	0.46	0.46	0.10	0.10	0.30	0.63	1	5517.58	3108.00
theta[18]	0.46	0.46	0.10	0.10	0.30	0.63	1	5622.46	2837.76

Table 7.1: Posterior summary of hierarchical binomial model for the therapeutic touch example.

variable	mean	median	sd	mad	q5	q95	rhat	ess_bulk	ess_tail
theta[19]	0.46	0.46	0.10	0.09	0.31	0.63	1	6228.82	3062.79
theta[20]	0.46	0.46	0.10	0.10	0.30	0.63	1	4758.62	2538.88
theta[21]	0.46	0.46	0.10	0.10	0.30	0.62	1	5922.64	2921.96
theta[22]	0.46	0.46	0.10	0.10	0.31	0.62	1	5218.69	3180.07
theta[23]	0.50	0.50	0.10	0.10	0.34	0.67	1	5424.13	2936.72
theta[24]	0.50	0.50	0.10	0.10	0.34	0.67	1	5321.61	2733.91
theta[25]	0.54	0.54	0.10	0.10	0.38	0.71	1	3861.44	2248.48
theta[26]	0.54	0.54	0.10	0.10	0.37	0.72	1	3808.59	2430.69
theta[27]	0.54	0.54	0.10	0.10	0.38	0.71	1	3679.86	2556.84
theta[28]	0.58	0.57	0.10	0.11	0.41	0.75	1	3005.40	2460.66
mu	0.44	0.44	0.04	0.04	0.38	0.50	1	2482.91	2932.61
kappa	18.65	16.37	9.63	7.92	7.49	36.92	1	992.72	1793.53

7.1.7 Posterior Predictive Check

With hierarchical models, there are two types of posterior predictive distributions:

1. *Same participants but new observations.* We can use the model to generate new observations, but the individual parameters (e.g., θ_j) remain the same. In our example, this would be the situation where we ask **the same 28 participants** to each do 10 more trials.
2. *New participants and new observations.* We can use the model to generate new observations with new parameters from the higher-order distribution (e.g., the Beta distribution for θ_j). In our example, this would be the situation where we ask **a new set of 28 participants** to each do 10 more trials.

In our Stan code, I used (1) to check the fit of the observed data. However, (2) can be helpful when one wants to use the model to make predictions of future data, as future data are unlikely to concern exactly the same participants.

```r
# The bayesplot::ppc_bars() unfortunately contains a bug
# (https://github.com/stan-dev/bayesplot/issues/266) at
# the time when I wrote this, so I'll use my own code.
# tt_fit$draws("ytilde", format = "draws_matrix") |>
#     ppc_bars(y = tt_agg$y)
yrep <- tt_fit$draws("ytilde", format = "draws_matrix")
yrep_intervals <- apply(
    yrep, MARGIN = 1, FUN = \(x) table(factor(x, levels = 0:10))
    ) |>
    apply(MARGIN = 1, FUN = \(x) {
        c(
            lo = quantile(x, .05)[[1]],
            me = median(x),
            hi = quantile(x, .95)[[1]]
        )
    })
data.frame(
    y = table(factor(tt_agg$y, levels = 0:10))
) |>
    setNames(c("x", "y")) |>
    cbind(t(yrep_intervals)) |>
    ggplot(aes(x = x, y = y)) +
    geom_col(alpha = 0.5) +
    geom_pointrange(aes(y = me, ymin = lo, ymax = hi)) +
    labs(y = "count", x = NULL)
```

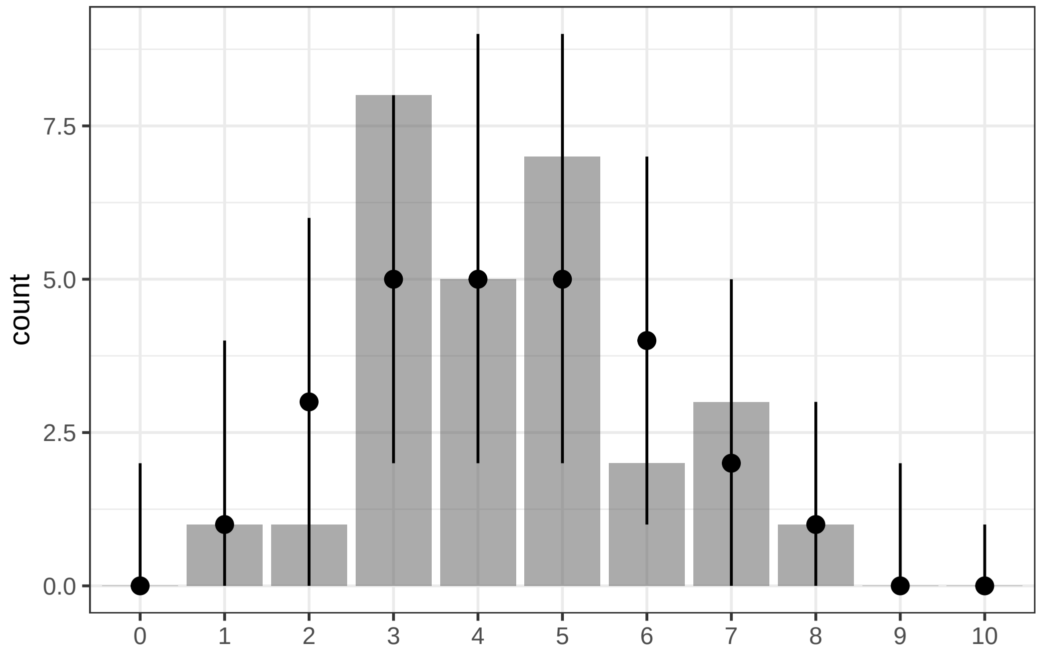


Figure 7.4: Posterior predictive check. The bar graph shows the observed data, and the error bars show the 90% posterior predictive interval in replicated data. The dots are the medians in the posterior predictive distribution.

7.1.8 Derived coefficients

One nice thing about MCMC is that it is straightforward to obtain posterior distributions that are functions of the parameters. For example, even though we only sampled from the posteriors of the θs, we can ask questions like whether there is evidence for a nonzero *difference* in θ between person 1 and person 28.

```r
tt_fit$draws() |>
    mutate_variables(
        theta1_minus14 = `theta[1]` - `theta[14]`,
        theta1_minus28 = `theta[1]` - `theta[28]`,
        theta14_minus28 = `theta[14]` - `theta[28]`
    ) |>
    subset(variable = c("theta1_minus14", "theta1_minus28",
                        "theta14_minus28")) |>
    summarise_draws() |>
    knitr::kable(digits = 2)
```

variable	mean	median	sd	mad	q5	q95	rhat	ess_bulk	ess_tail
theta1_minus14	-0.11	-0.11	0.13	0.13	-0.34	0.09	1	4950.80	2632.63
theta1_minus28	-0.27	-0.26	0.15	0.16	-0.53	-0.03	1	2312.44	3115.27
theta14_minus28	-0.15	-0.15	0.14	0.14	-0.38	0.06	1	4309.81	3013.97

7.1.9 Conclusion

As 0.5 is included in the 95% CI of θ for all participants, there is insufficient evidence that people can sense "therapeutic touch."

7.1.10 Shrinkage

```r
mcmc_intervals(tt_fit$draws(),
               # plot only parameters matching "theta"
               regex_pars = "^theta") +
    geom_point(
        data = tibble(
            parameter = paste0("theta[", 1:28, "]"),
            x = tt_agg$y / tt_agg$n
        ),
        aes(x = x, y = parameter),
        col = "red"
    ) +
    xlim(0, 1)
```

```
Scale for x is already present.
Adding another scale for x, which will replace the existing scale.
```

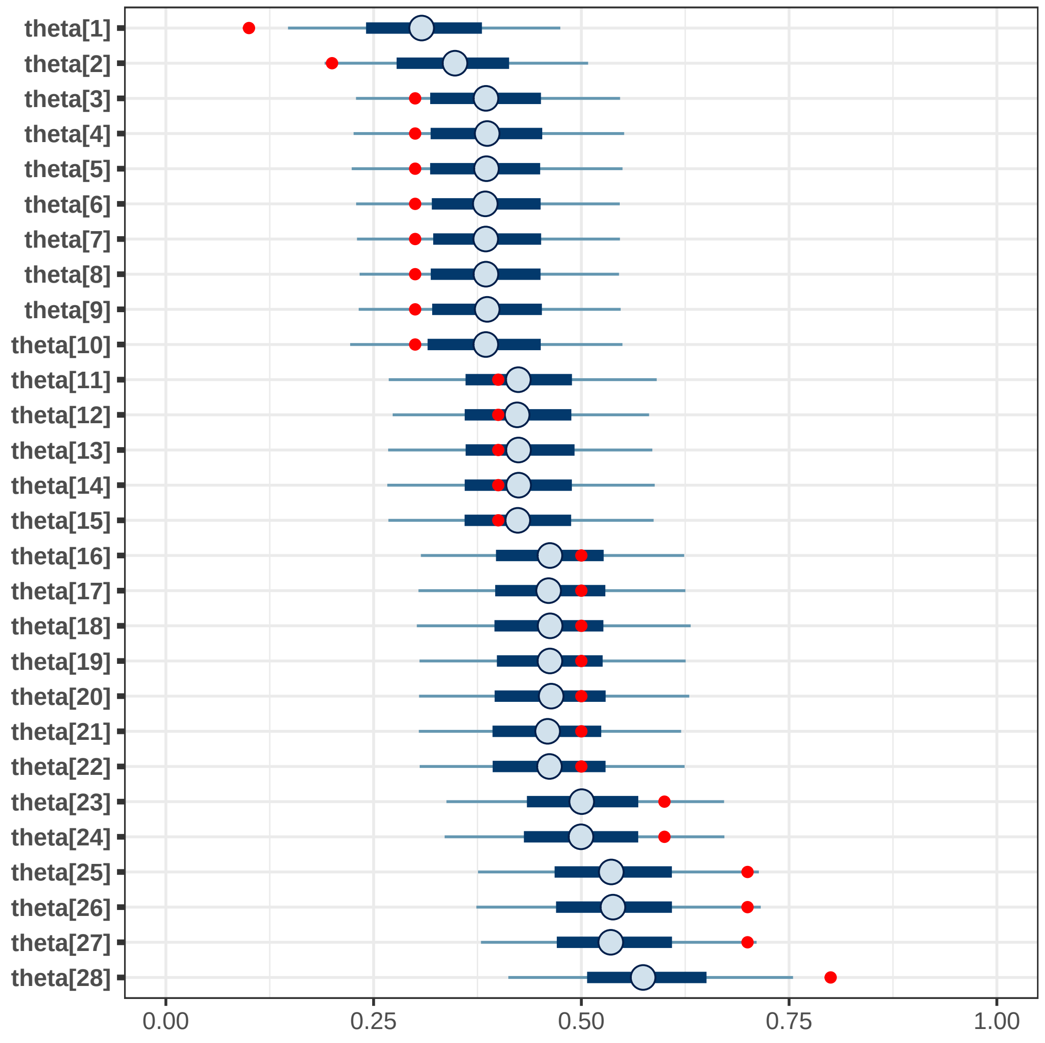


Figure 7.5: Shrinkage effect in a hierarchical model.

As can be seen, the posterior distributions are closer to the center than the data (in red). This

pooling results from the belief that the participants have something in common.

7.1.11 Multiple Comparisons?

Another benefit of a Bayesian hierarchical model is that *you don't need to worry about multiple comparisons*. There are multiple angles on why this is the case, but the basic answer is that the use of common prior distributions builds in the prior belief that the clusters/groups are likely to be equal. See discussion here and here.

7.2 Hierarchical Normal Model

7.2.1 Eight Schools Example

This is a classic data set first analyzed by Rubin (1981). It is also the example used in the RStan Getting Started page. The data contains the effect of coaching from randomized experiments in eight schools. The numbers shown (labelled as `y`) are the mean difference (i.e., effect size) in performance between the treatment and control groups on SAT-V scores.

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

In the above data, some numbers are positive, and some are negative. Because the sample sizes are different, the data also contained the standard errors (labelled as `sigma`) of the effect sizes. Generally speaking, a larger sample size corresponds to a smaller standard error. The research question is

1. What is the average treatment effect of coaching?
2. Are the treatment effects similar across schools?

7.2.2 Model

Model:

$$\begin{aligned} d_j &\sim N(\theta_j, s_j) \\ \theta_j &\sim N(\mu, \tau) \end{aligned}$$

Prior:

$$\begin{aligned} \mu &\sim N(0, 100) \\ \tau &\sim t_4^+(0, 100) \end{aligned}$$

Given the SAT score range, it is unlikely that a coaching program will improve scores by 100 or so, so we use a prior of $\mu \sim N(0, 100)$ and $\tau \sim t_4^+(0, 100)$.

The model above is the same as one used in a *random-effect meta-analysis*. See this paper for an introduction.

7.2.3 Non-Centered Parameterization

The hierarchical model is known to create issues in MCMC sampling, such that the posterior draws tend to be highly correlated even with more advanced techniques like HMC. One way to alleviate that is to reparameterize the model using what is called the **non-centered parameterization**. The basic idea is that, instead of treating the θs as parameters, one uses the **standardized deviation** from the mean to be parameters. You can think about it as converting the θs into z scores, and then sample the z scores instead of the original θs.

Model:

$$\begin{aligned} d_j &\sim N(\theta_j, s_j) \\ \theta_j &= \mu + \tau\eta_j \\ \eta_j &\sim N(0, 1) \end{aligned}$$

```
data {
  int<lower=0> J;             // number of schools
  vector[J] y;                // estimated treatment effects
  vector<lower=0>[J] sigma;   // s.e. of effect estimates
}
parameters {
  real mu;                    // overall mean
  real<lower=0> tau;          // between-school SD
  vector[J] eta;              // standardized deviation (z score)
}
transformed parameters {
  vector[J] theta;
  theta = mu + tau * eta;     // non-centered parameterization
}
model {
  eta ~ std_normal();         // same as eta ~ normal(0, 1);
  y ~ normal(theta, sigma);
  // priors
  mu ~ normal(0, 100);
  tau ~ student_t(4, 0, 100);
}
```

```r
hnorm_mod <- cmdstan_model("stan_code/hierarchical-normal.stan")
```

```r
fit <- hnorm_mod$sample(
    data = schools_dat,
    seed = 1804,  # for reproducibility
    refresh = 1000,
    adapt_delta = 0.9  # to improve convergence
)
```

```
Running MCMC with 4 sequential chains...

Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1 finished in 0.1 seconds.
Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2 finished in 0.1 seconds.
Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3 finished in 0.1 seconds.
Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4 finished in 0.2 seconds.

All 4 chains finished successfully.
Mean chain execution time: 0.1 seconds.
Total execution time: 0.7 seconds.
```

Treatment effect estimates of individual schools (θ), average treatment effect (μ), and treatment effect heterogeneity (τ).

```r
fit$summary(c("theta", "mu", "tau")) |>
    knitr::kable(digits = 2)
```

variable	mean	median	sd	mad	q5	q95	rhat	ess_bulk	ess_tail
theta[1]	11.70	10.60	8.69	7.38	-0.20	27.60	1.00	3475.69	3347.72
theta[2]	7.98	7.93	6.31	5.82	-2.18	18.34	1.00	5870.22	3353.39
theta[3]	6.13	6.77	7.97	6.50	-7.14	17.85	1.00	4318.80	2941.48
theta[4]	7.69	7.75	6.75	6.15	-3.43	18.53	1.00	5452.05	3268.65
theta[5]	5.22	5.53	6.22	5.82	-5.64	14.70	1.00	4118.93	3165.15
theta[6]	6.15	6.52	6.87	6.15	-5.19	16.86	1.00	4526.45	3352.39
theta[7]	10.80	10.14	6.78	6.32	0.88	23.09	1.00	4445.48	3533.95
theta[8]	8.55	8.33	7.74	6.42	-3.59	21.11	1.00	4978.08	3232.97
mu	7.95	8.00	5.26	4.97	-0.43	16.20	1.00	2917.45	1987.36
tau	6.75	5.31	5.91	4.84	0.55	17.53	1.01	1316.86	1673.79

On average, based on the 90% CI, coaching seemed to improve SAT-V by -0.43 to 16.2 points. There was substantial heterogeneity across schools.

We can also get the probability that the treatment effect was > 0:

```r
# Obtain draws
draws_mu <- fit$draws("mu", format = "draws_matrix")
mean(draws_mu > 0)
```

```
[1] 0.93975
```

Here are the individual-school treatment effects:

```
mcmc_areas_ridges(fit$draws(), regex_pars = "theta")
```

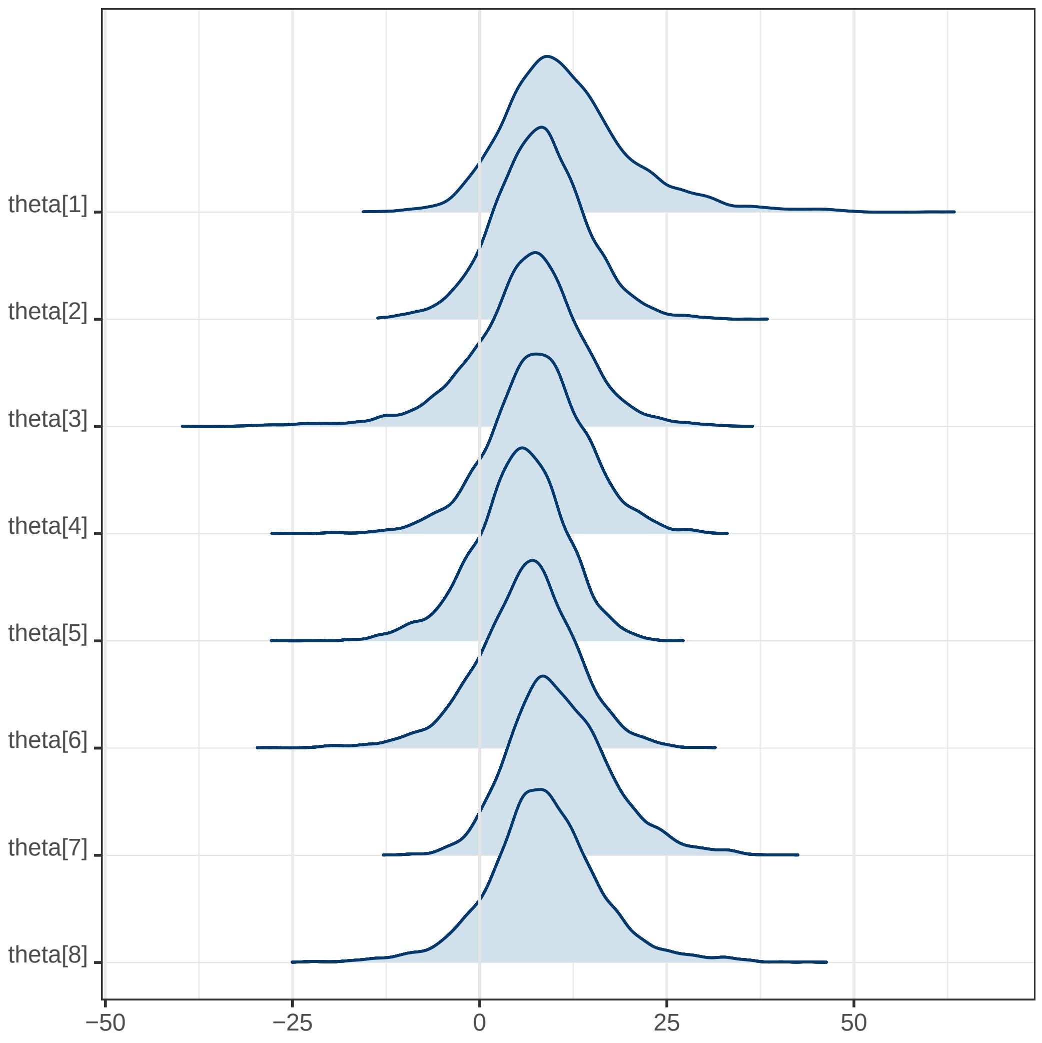


Figure 7.6: Posterior distribution of the true effect size in the eight schools example.

7.2.4 Prediction Interval

Posterior distribution of the true effect size of a new study, $\tilde{\theta}$

```
# Prediction Interval
# (can also be done in Stan, as in the previous example)
fit$draws(c("mu", "tau")) |>
```

```
    mutate_variables(
        theta_tilde = rnorm(4000, mean = mu, sd = tau)) |>
    summarise_draws() |>
    knitr::kable(digits = 2)
```

variable	mean	median	sd	mad	q5	q95	rhat	ess_bulk	ess_tail
mu	7.95	8.00	5.26	4.97	-0.43	16.20	1.00	2917.45	1987.36
tau	6.75	5.31	5.91	4.84	0.55	17.53	1.01	1316.86	1673.79
theta_tilde	8.03	7.87	10.19	7.00	-6.69	22.89	1.00	3947.08	3042.56

The posterior interval for $\tilde{\theta}$ indicates a range of the treatment effect for a new study.

References

Gigerenzer, G. (2004). Mindless statistics. *The Journal of Socio-Economics*, *33*(5), 587–606. https://doi.org/10.1016/j.socec.2004.09.033

Johnson, A. A., Ott, M. Q., & Dogucu, M. (2022). *Bayes rules! An introduction to Bayesian modeling with R*. CRC Press.

Lambert, B. (2018). *A student's guide to Bayesian statistics*. SAGE.

McGrayne, S. B. (2011). *The theory that would not die: How Bayes' rule cracked the enigma code, hunted down Russian submarines, & emerged triumphant from two centuries of controversy*. Yale university press.

Van De Schoot, R., Winter, S. D., Ryan, O., Zondervan-Zwijnenburg, M., & Depaoli, S. (2017). A systematic review of Bayesian articles in psychology: The last 25 years. *Psychological Methods*, *22*(2), 217–239. https://doi.org/10.1037/met0000100