Chapter 13
The past and future of statistics: Bayesian modeling

James Myers

2022/5/14

1. Introduction

Almost everything that we've done in this book, especially all those tests that compute all those *p* values, is built on an approach to probability called **frequentist**. This is just a fancy term for the idea that the probability of an outcome is the frequency the outcome occurs divided by the total number of all possible events. This approach, which seems to be the most "objective" way to define probability, has dominated statistics since at least the early twentieth century.

However, in recent years an alternative (though very old) approach has grown dramatically in popularity: **Bayesian** statistics (I'll explain the name shortly). This takes a "subjective" approach to probability, interpreting it as the strength of one's belief in a hypothesis. This sounds like a subtle philosophical distinction, but it gives rise to dramatic differences in the workings and conclusions of Bayesian statistics, as compared with the more familiar traditional frequentist statistics.

We'll get into the details below, but the core idea of Bayesian statistics is that you start with a **prior** belief in a hypothesis and a sense of the **likelihood** (似然) that the hypothesis will generate this versus that kind of observation. Then you observe some **evidence**, which leads you to update your belief in the hypothesis to a **posterior** state. If you collect more evidence, this posterior becomes your new prior, and the cycle starts over again.

This simple idea differs from the frequentist approach in two major ways: (1) Bayesian statistics uses the evidence to learn about the abstract hypothesis (bottom-up reasoning); (2) prior belief can affect the influence of the evidence (e.g., if you really love your initial hypothesis, you will need a lot of contrary evidence to change your mind). By contrast: (1) in the frequentist approach we start with the null hypothesis, imagine all the possible samples that could be generated by it, and check if our actual sample fits this pattern (top-down reasoning); (2) our prior belief in the null hypothesis versus alternative hypothesis plays no role at all.

The major advantage of Bayesian statistics is that its core logic closely matches how people intuitively think about probabilities. Many of the warnings I gave throughout the book when using frequentist statistics, relating to the null hypothesis, *p* values, confidence intervals, multiple comparisons, and other things, are necessary because the human brain naturally wants to think in Bayesian terms, so you have to actively suppress these intuitions when you do traditional statistics. Similarly, as I've already mentioned in the probability chapter, Bayesian logic provides an insightful description of how scientists test hypotheses and draw conclusions from their observations. No scientist can get truly passionate about falsifying the null hypothesis, which is what they're required to do if they use traditional statistics. Instead, they want to see how their observations affect their confidence in their research hypothesis, and that's just what Bayesian statistics calculates for you.

The major disadvantage of Bayesian statistics is that it is difficult to compute. Some would argue that it's also problematic to start the calculations with subjective intuitions about prior probabilities, but as we'll see, subjectivity sneaks into traditional frequentist statistics too. In fact, Bayesians would say that since subjectivity is unavoidable in any human activity, including science, it's best not to be sneaky about it.

So the real main difficulty of Bayesian statistics is the computation. Even though the core ideas go back centuries, running a Bayesian analysis, even on very simple data, requires a powerful, fast computer, since like many of the methods we've discussed in the past few chapters (logistic regression, generalized additive modeling, mixed-effects modeling), only the very simplest Bayesian analyses can be solved using fixed equations. The rest require looping through estimation algorithms, and that takes powerful, fast computers.

Another difficulty with Bayesian statistics is that its scope is potentially just as vast as all of traditional statistics, so there's no way I can give you a completely satisfying introduction to it in just one chapter. It would be quite possible to write an entire statistics textbook entirely on Bayesian methods, and in fact there are many such books (e.g., Dienes, 2008; Jaynes, 2003; Gelman & Hill, 2007; Gill, 2014; Kruschke, 2015; Lee, 2004; Lynch, 2007). Of all of these books, the easiest book for beginners is probably Kruschke (2015), though even it is quite a bit more "mathy" than the usual introductory statistics book. For less mathematically heavy overviews of the core ideas, you can also read about the history of Bayesian statistics in Salsburg (2001) and McGrayne (2011), see how Bayesian logic may apply to the philosophy of science, as argued by Howson & Urbach (2006), or ponder the idea that maybe even the study of history would benefit from Bayesian logic, as argued by Carrier (2012).

As for this chapter, my goals are more modest: explain why Bayesian statistics is getting so popular now, describe how to use it to do simple things that traditional statistics can't do at all, and give you a small sense of how it could be applied to handle more complicated (i.e., realistic) situations.

2. The Bayesian way

I claimed above that Bayesian statistics is fundamentally more intuitive than traditional statistics. I'll start this section by trying to convince you that this is true. Then I'll explain the simple mathematical core of Bayesian statistics, called **Bayes's rule**. Finally I'll explain a very simple method that uses Bayes's rule to do something impossible in traditional statistics: argue *for* the null hypothesis.

2.1 Why Bayes?

Bayesian statistics is named after the British amateur mathematician Thomas Bayes (1701-1761), who never published his work while alive and left it to other mathematicians, and future generations, to work out the details. I'll get into these details later, but for this section all we need is the informal description I gave in the introduction: it describes how evidence affects a researcher's belief in a hypothesis.

This description makes Bayesian statistics sound very subjective, and this is its most controversial aspect. Yet in all other ways, Bayesian logic is clearly superior to frequentist logic, and even its supposed subjectivity can be justified.

As we've seen throughout this book, frequentist statistics is heavily focused on the *p* value, yet as we've also seen, the *p* value is easy to misunderstand. As Cohen (1994, p. 997) puts it: "What we want to know is 'Given these data, what is the probability that [the null hypothesis] H_0 is true?' But ... what [the *p* value actually] tells us is 'Given that H_0 is true, what is the probability of these (or more extreme) data?'"

By contrast, Bayesian statistics doesn't use *p* values at all, and instead tells us more directly what we *really* want to know, which is this: "Given these data, what is the probability that that our *research* hypothesis H_1 is true?"

One big problem with the *p* value is that it represents the probability of the null hypothesis, and null hypotheses are often quite implausible, making them so easy to reject that it's not clear what if we're learning anything from them at all. For example, as we've seen, almost any correlation with linguistic variables will turn out to be statistically significant, if you have enough data. But that doesn't mean the correlation is big enough to mean anything in real life. By contrast, Bayesian statistics doesn't give special status to the null hypothesis, but treats it as just one of many possible hypotheses, and careful consideration is given to choosing plausible alternatives.

On the other hand, sometimes our research hypothesis *is* the null hypothesis, and then traditional statistics really cannot help us at all. This is a pretty common situation in linguistics, since grammar is based on **invariances**. In syntax, *dog* and *linguistics* are both nouns, so at some level they should be treated exactly the same way; in phonology, [p] and [p^h] are both allophones of the English phoneme /p/, so again, there should be a level where they're exactly the same. So if you hypothesize such invariances, you predict that the right sort of study on them will show no difference.

But as I've repeated endlessly throughout this book, traditional statistics doesn't let you "accept" the null hypothesis. If you can't reject it, all you can say is that you failed to reject it: absence of evidence (for a difference) is not the same as evidence of absence (of a difference). This is counterintuitive: if you have lots of well-collected evidence and none of it shows a difference, why doesn't that count as evidence *for* the lack of a difference? Indeed, that's exactly what Bayesian statistics lets you do, as we will see.

Another big problem with frequentist statistics is that despite calling itself "objective", it actually depends a lot on the researcher's subjective beliefs, just in indirect and tricky ways. I'll illustrate this with an example based on one in Kruschke (2015, pp. 300ff), modified in linguistic terms. Suppose a linguist hypothesizes that a certain sentence is ungrammatical. She asks 25 people to give binary good/bad judgments on this sentence, and only 8 of them accept it, while the remaining 17 reject it. Does this imply that the sentence is truly unacceptable? (To keep things simple, we'll ignore the experimental design flaws: the linguist really should have a control sentence to compare with, and using just one sentence raises the "language-as-fixed-effect" fallacy.)

In traditional statistics, we know how to test this research hypothesis against the null hypothesis that the random probabilities for acceptability are equal: we use a binomial test. Since the linguist's research hypothesis is directional, let's use a one-tailed test (since we're talking about math here, not publishing a real paper, and we want to keep the probability logic as simple as possible). But the results are not significant: the one-tailed $p = .05387607 > .05$ (try it!).

```
pbinom(8, 25, 0.5)
```

But note that this analysis assumes that the linguist decided ahead of time that she would test exactly 25 sentences. What if she actually was trying to collect eight "yes" responses, and kept asking people until she got her eighth "yes", and that just happened to be the twenty-fifth person? Frequentist statisticians say that you should never do this because it undermines the assumptions of the standard test. But it happens a lot in real life, when researchers decide to stop their research based on the results they've collected so far, rather than fixing on a specific sample size ahead of time.

What is the probability that the eighth "yes" will happen to come from the twenty-fifth person? We can't use the binomial distribution, but we can use a related one called the **negative binomial** distribution (a generalization of the Poisson distribution). Of course R has functions for this, including one for computing probability: **pnbinom()** (if you want to see what this distribution looks like you can use **plot()** and **dnbinom()**, as we did for the binomial, normal, *t*, *F*, and Poisson distributions in earlier chapters).

The negative binomial analysis gives us a totally different *p* value from before. The one-tailed $p = .002275692 < .05$ is significant (try it)!

```
pnbinom(8, 25, 0.5)
```

Kruschke (2015) notes that the *p* values here are not as objective as they seem: exactly the same situation can arise in the real world, but whether we use the binomial or the negative binomial distribution depends on the linguist's intentions, which is ultimately a private part of her psychology, not something that critical outsiders can check for themselves.

Kruschke also points out that the null hypothesis that "yes" and "no" have equal probabilities was unrealistic in the first place, and as linguists, we know this is true for this particular situation too. Of course the linguist must have already observed some asymmetry, or else why did she want to run the experiment at all? Thus either way she computes the *p* value, it doesn't actually test her research hypothesis. Kruschke concludes that it's better for researchers to be honest about their prior assumptions, instead of pretending to be "objective", and plug those prior values into a Bayesian analysis so everybody can critically examine it.

Another example of hidden subjectivity in traditional statistics concerns multiple comparisons. Remember that in the first ANOVA chapter we proved that if you have a three-level factor that's significant by a one-way ANOVA, it's not legitimate to compare each level against each other with *t* tests, since the three repeated comparisons on the same data would raise the Type I error rate higher than the alpha level that we set ahead of time. That's why folks like Tukey worked so hard to invent **post hoc** multiple-comparison tests that don't raise the Type I error rate too much.

But what if these post-ANOVA level-versus-level comparisons were **planned comparisons**? Then your decision to do any one analysis is unrelated to your decision to run the others, so maybe we can treat the three analyses as independent. Or maybe not: Gelman and Loken (2013) argue that it's much harder to apply this logic in a valid way than researchers tend to think. So like the binomial versus negative binomial analyses, you can see that the *p* values are strongly dependent on the researcher's intentions. Is this really an "objective" kind of statistics?

Yet another problem with traditional statistics concerns confidence intervals. You might remember that I already explained this in the *t* test chapter: we cannot say that the population mean has a 95% chance of being somewhere within a traditional 95% confidence interval, even though that's what we really wish we could say. This is because in traditional statistics, your observed sample is treated as just one of many possible samples from the same population, each with its own sample mean. This means that if we took the 95% confidence interval calculated from *our* sample, and repeatedly put this fixed interval around the mean of *every* same-sized sample from the population, then 95% of *those repeated intervals* would contain the population mean. As pointed out by Lee (2004), a Bayesian statistics textbook, everybody finds this very confusing, since the logic is exactly backwards from what we want to say.

This confusion is an automatic side-effect of the frequentist approach. In order to understand our data set, we're supposed to imagine an infinity of other data sets generated by the null hypothesis. This paradoxically makes the abstract population more "real" than the observations we see with our own eyes!

By contrast, in Bayesian statistics, it is the actual data set that is considered more "real": you start with your data set and then compute what it implies about your research hypothesis, instead of starting with the null hypothesis and null population and computing what it implies about your real data set, as in traditional statistics. So instead of the confusing confidence intervals of traditional statistics, Bayesian statistics has something called the **highest density interval**, explained later in this chapter, which does exactly what we wish the traditional confidence interval did: it reflects the degree of confidence we learn from our data about the hypothesized parameter (e.g., the mean).

In general, then, Bayesian statistics is more intuitive than traditional statistics. In yet another Bayesian textbook, Jaynes (2003, pp. 3ff) illustrates this with a story. Once night a cop is walking down an empty city street and hears a burglar alarm ringing in a nearby jewelry store. When he runs over he sees that the big glass storefront window has been smashed open, and a man in a mask is crawling out with a bag of jewels. What should the cop do?

Apply Bayesian reasoning, of course. The cop first considers the prior probability of a theft in general. Store thefts don't happen all the time to every store, so this prior probability is relatively low. But this low prior probability is outweighed by the high likelihood that a theft would generate all of the things that he observed: the alarm, the broken window, the guy with a mask and a bag of jewels. This gives a relatively high posterior probability that a theft is going on. Finally, he compares this with an alternative scenario where some innocent series of events would generate the same observations. For example, Jaynes suggests that maybe the man is the store owner himself, and he's wearing a mask because he just returned from a costume party, but he didn't bring his store key so he was locked out, but just then a truck drove by and accidently knocked a rock into the window, breaking the glass and setting off the alarm, and so the store owner had no choice but to crawl in to save his jewelry. Obviously the likelihood of this other scenario is much lower than the robbery hypothesis, which means that its posterior probability is also much lower. Therefore, Bayesian logic gives the cop pretty good justification to arrest the man.

What could frequentist statisticians say in this situation? Maybe we could collect data on thefts and non-thefts, and use logistic regression to predict this binary variable from variables like the presence/absence of burglar alarms, broken windows, and men wearing masks crawling out with bags of jewels. Even if such an analysis could be performed, it would certainly be completely different from the immediate, intuitive, and completely convincing logic of the cop!

Bayesians thus argue that scientific reasoning is actually Bayesian (e.g., Jaynes, 2003; Howson & Urbach, 2006; Dienes, 2008), and there is some experimental evidence that even ordinary people are natural Bayesian reasoners (e.g., Oaksford & Chater, 2009).

If Bayesian statistics is so great, why isn't everybody using it already? One reason is historical (see Salsburg, 2001; McGrayne, 2011). As noted above, Mr. Bayes himself didn't push his basic insight very far. Its early development was mostly due to independent work by the French mathematician Pierre-Simon Laplace (1749-1827). This guy was a real genius, and his work lives on today in many areas of statistics, including the Laplace approximation that **glmer()** uses to compute generalized mixed-effects models. But after playing around with Bayesian logic for a while, Laplace discovered the Central Limit Theorem, which convinced him that as long as you have enough data, Bayesian and frequentist analyses will always come out the same. Since frequentist mathematics is easier to work with, that's the way he went, and so did most statisticians, starting with the likewise brilliant German mathematician Johann Carl Friedrich Gauss (1777-1855), who is the reason why the normal distribution is also called the Gaussian distribution.

Moreover, when modern frequentist statistics matured in the early 1900s, the Bayesian approach was attacked for being too subjective. One of the biggest critics was our friend Fisher; his hatred of Bayes and his own obvious genius together caused Bayes almost to become taboo. Ironically, during World War II and the Cold War, Bayesian methods were found to be very good at inferring hidden things from observations, so they were used a lot in code-breaking, finding enemy submarines, and things like that. But this also meant that Bayesian methods became Top Secret and nobody else could use them!

Another reason why Bayesian statistics was neglected has to do with the computations. Although the basic math is trivial, applying it to realistic cases makes it much more complex. In particular, we need to combine the prior, the likelihood and the evidence in a way that generates the same type of probability distribution for the posterior, so that when the next set of evidence comes in, our former posterior becomes the new prior for the next round. But this turns out to be extremely hard to do. The area under probability distributions is calculated using **integrals** (積分) in **calculus** (微積分學), but the integrals needed for realistic Bayesian analyses often have no formulas: you can only estimate them. This wasn't really practical until the 1980s, when a computer algorithm called **Gibbs sampling** finally became routine, even though it's named after an American scientist, Josiah Willard Gibbs (1839-1903), who died decades earlier (just as Fisher dreamed up mixed-effects modeling long before it was practical). The computer implementation of Gibbs sampling ultimately led to influential programs like **BUGS** (for "**B**ayesian inference **U**sing **G**ibbs **S**ampling"; Lunn et al., 2009). As computers got even faster and more powerful, other types of computational methods become possible as well, leading to alternative programs like **Stan** (Stan Development Team, 2019), named in honor of Polish-American mathematician Stanisław Marcin Ulam (1909-1984). He pioneered the **Monte Carlo** method that is also used elsewhere in statistics (e.g., Monte Carlo methods are used to estimate the *p* value for **fisher.test(simulate.p.value = TRUE)**); it's named after the famous gambling city because it's essentially a random resampling method.

2.2 Bayes's rule

The basic idea of Bayesian statistics can be expressed in a probability formula known as **Bayes's rule** (貝氏定理、貝斯定理), also known as **Bayes's law** or **Bayes's theorem** (and as with many *s*-final words in this dumb language of English, it's also sometimes spelled **Bayes' rule**: see https://en.wikipedia.org/wiki/Apostrophe).

2.2.1 The math of Bayes's rule

Formally, Bayes's rule is concerned with that most confusing part of probability theory, **conditional probability** (條件機率). Maybe you still remember its formal definition from the probability chapter. The equation for conditional probability, shown below, says that the probability of event A, given event B, is the probability of A & B happening together (called their **conjoint probability**) given B, that is, relative to the probability of B in general. From now on, we'll write P(A & B) as P(A,B) to save a bit of space and to follow the more formal mathematical style:

Conditional probability: $P(A|B) = P(A,B)/P(B)$

Let's test this equation with a case where A and B are independent. Say A = A ("ace") and B = ♠ ("spades"). Imagine you pull out a card randomly, and it's a spade (the "given" part). What's the probability that it's also an ace? Obviously it's 1/13, since there are 13 spades. That's the left side of the above equation: P(A|♠). Now, what's P(A,♠)? That's the probability that the card is an ace and a spade at the same time, i.e., A♠. Again obviously, that's 1/52, since there are 52 cards in the whole deck. Finally, what's P(♠)? That's the probability that the card is a spade in general, and that is 1/4, since there are four "suits" (花色). Now we apply the above equation, and yes, it works!

$P(A|♠) = P(A,♠)/P(♠) = (1/52)/(1/4) = 4/52 = 1/13$

But conditional probability also works if A and B are not independent, as in the contingency table in Table 1 (which we discussed in the chapter on chi-squared tests). We want to know whether the vowel (/a/ vs. /i/) influences the place of the nasal consonant ([n] vs. [ŋ]). As we saw, the frequentist chi-squared test shows that the vowels and nasals aren't independent ($\chi^2(1) = 150.2$, $p < .0001$).

Table 1. The frequencies of various consonant phonemes and allophones

rime	/aŋ/	/iŋ/	totals
[n] tokens	57	112	169
[ŋ] tokens	190	11	201
totals	247	123	**370**

In terms of probability theory, what we're looking at is conditional probabilities for the nasal forms, with the vowels as given. If A = [n] and B = /a/, the conditional probability of [n] given /a/ is like so:

P([n]|/a/) = P(/a/,[n])/P(/a/)

P(/a/,[n]) is the count of that combination (57) divided by the grand total (370). P(/a/) is the **marginal** at the bottom of the /a/ row in Table 1 (247), also divided by the grand total (370):

P([n]|/a/) = (57/370)/(247/370) = 57/247

But there's an easier way to calculate this: P([n]|/a/) just counts the frequency of [n] within the /a/ column (just as P(A|♠) is just the proportion of aces in the set of spades):

P([n]|/a/) = P([n] in /a/ column) = 57/247

There's an apparently silly implication of the above discussion that will actually become crucial later. Namely, since the conditional probability P(A|B) is related to the conjoint probability P(A,B), we can also use algebra to derive the latter from the former:

Conditional probability:	P(A|B) = P(A,B)/P(B)
Conjoint probability:	P(A,B) = P(A|B)·P(B)

This lets us rewrite P(A) in terms of P(A|B) and P(B) for all possible Bs. This could be B vs. ¬B (not-B), or B = ♠ vs. B = ♣ vs. B = ♥ vs. B = ♦. Then P(A) is the sum of all the conjoint probabilities. For example, with B = black card (♠ or ♣) and ¬B = red (♥ or ♦). As shown below, the proportions all come out correct:

P(A,black) = P(A|black)·P(black) = (2/26)×(1/2) = 2/52
P(A,red) = P(A|red)·P(red) = (2/26)×(1/2) = 2/52
P(A) = P(A,black) + P(A,red) = 2/52 + 2/52 = 4/52 = 1/13

We can actually generalize this logic to any number of Bs with the sum operator Σ:

Probability as conjoint probability $P(A) = \sum_i P(A|B_i) \cdot P(B_i)$

Now for Bayes's rule, which flips conditional probability around: if you already know P(A|B), you can use Bayes's rule to compute P(B|A). Here's the proof.

We start by giving both conditional probabilities:

$P(A|B) = P(A,B)/P(B)$	{the conditional probability we played with above}
$P(B|A) = P(A,B)/P(A)$	{the flipped-around conditional probability}

This gives us two ways to express the conjoint probability, so we can equate them:

$P(A,B) = P(A|B)\cdot P(B)$	{from the first conditional probability}
$P(A,B) = P(B|A)\cdot P(A)$	{from the second conditional probability}
$P(A|B)\cdot P(B) = P(B|A)\cdot P(A)$	{equate the two sides}

Now we divide P(A) from both sides, and get the simplest version of Bayes's rule:

Bayes's rule (simplest version): $P(B|A) = \frac{P(A|B)\cdot P(B)}{P(A)}$

This formula makes perfect sense in terms of the contingency table in Table 1. As we saw earlier, P([n]|/a/) can be thought of as the probability of [n] in the /a/ column: 57/247. Similarly, the flipped-around conditional probability P(/a/|[n]) can be thought of as the probability of /a/ in the [n] column: 57/169.

Here's the confirmation that this follows Bayes's rule. We've already computed P([n]|/a/) before, using P([n],/a/) and P(/a/), and now we also need P([n]), which is the marginal row total for [n] (169) divided by the grand total (370):

P(/a/|[n]) = P([n]|/a/)·P(/a/)/P([n]) = (57/~~247~~)(~~247/370~~)/(169/~~370~~) = 57/169

Because the vowels and nasals aren't independent, P([n]|/a/) and P(/a/|[n]) are different. Phonologists would be more interested in P([n]|/a/) (since the vowels "cause" the nasal place features), but if somehow they only had access to information about P(/a/|[n]), they could use Bayes's rule to flip around the conditional probabilities.

2.2.2 Basic applications of Bayes's rule

This discussion may make Bayes's rule seem trivial and very dry. In particular, maybe it's not clear to you how Bayes's rule relates to the notion of probability as subjective. Well, you can think of science as collecting some data (*D*) and trying to figure out the hypothesis (θ)

that best explains it (Bayesian statisticians like to use the Greek letter theta for the unknown parameter, maybe from “theory”?). In order to be testable, our scientific theory must make some prediction about the likelihood that our hypothesis θ will generate our observed data *D*, and that’s just a kind of conditional probability, namely the probability that we will observe *D* given our hypothesis θ: P(D|θ). But what we really want to know is the other way around, namely the conditional probability P(θ|*D*), which represents the probability that our hypothesis θ is true, given our data *D*.

Bayes’s rule to the rescue! If we plug in our values, we see that all we need to know to find the **posterior** probability P(θ|*D*) is the **likelihood** P(*D*|θ), the **evidence** P(*D*) (that is, the probability of making these observations in general), and the **prior** probability P(θ) (that is, our belief in θ before we collect any evidence):

Bayes’s rule (simplest science-style version): $\mathrm{P}(\theta|D) = \frac{\mathrm{P}(D|\theta)\cdot\mathrm{P}(\theta)}{\mathrm{P}(D)}$

Since linguistics is also supposed to be a science, this logic is quite useful in linguistics too. For example, Hammond (2016) uses it to explain how learning constraints helped shaped the peculiar structure of the Welsh gender system. Even more dramatically, Perfors et al. (2010, 2011) apply Bayes’s rule to the classic innateness question: how do children learn a grammar G from language data L? In the formula below, the likelihood P(L|G) is the probability that the language is generated by the grammar, P(L) is the probability of the language magically arising by chance, and the posterior P(G|L) is the learnability of the grammar from language data. The prior P(G) then represents the *innate* expectations that a child has about possible human grammars (i.e., Universal Grammar).

A Bayesian analysis of innateness and language learning: $\mathrm{P}(\mathrm{G}|\mathrm{L}) = \frac{\mathrm{P}(\mathrm{L}|\mathrm{G})\cdot\mathrm{P}(\mathrm{G})}{\mathrm{P}(\mathrm{L})}$

As noted earlier, the most controversial part of Bayesian reasoning is the prior, since it’s not always obvious where it comes from. For example, what is the prior probability for the hypothesis that you tested in your most recent linguistic analysis? Maybe you personally had strong confidence in it, but your colleagues may have been a bit more skeptical. How can we possibly assign a specific number between 0 and 1 for that?

Fortunately, if we have a huge amount of evidence, the influence of the prior probability disappears (which is part of the reason that frequentist and Bayesian statistics can give very similar results with larger samples). Still, until we collect enough evidence, the prior can affect our conclusions, and thus how convincing they are to skeptics.

One strategy is to choose an **uninformative** prior, weighing all competing hypotheses equally. This strategy doesn't work in many real cases, since even if we really know what all the competing hypotheses are, it's not realistic to say that all are equally likely. This is one reason why Fisher was so skeptical: the prior is never truly "objective".

Bayesian statisticians respond to this criticism by suggesting that we should choose a prior that relates to our existing beliefs, but be totally public about those beliefs. Ideally, you should work with your worst enemy to choose a prior that you can both live with. Sometimes you may agree to choose an uninformative prior, but other times you may choose an informative prior. You can also try out different priors (yours vs. your enemy's) to see how much difference this makes. Often it doesn't change the practical conclusions at all.

To actually use Bayes's rule in many realistic situations, however, we need to complicate it a bit. One problem with the simplest version above is that it assumes that our hypothesis θ is a single value, but this is never really true. For example, if our study involves binary outcomes, our prior might involve the pair of null hypotheses that half of the observations will be "yes" ($P(\theta_{yes}) = .5$) and half will be "no" ($P(\theta_{no}) = .5$). So the prior is actually a probability *distribution*, not a single point estimate. Another problem is that the simple Bayes's rule treats the evidence $P(D)$ as unrelated to the hypothesis, but actually, data only become evidence in relation to some hypothesis.

Most seriously, the simple formula doesn't give you enough room to specify your own prior in detail. For example, with binary data, the simple formula would look like this:

Simple version of Bayes's rule for binary data: $P(\theta_{yes}|D) = \frac{P(D|\theta_{yes})\cdot P(\theta_{yes})}{P(D)}$

So there's a place where you could enter your own prior for $P(\theta_{yes})$, but there's no place to enter the prior value for $P(\theta_{no})$. You might think this doesn't matter, since the two values must add up to 1 anyway, but there's no place that this crucial information is mentioned in the simple version. This makes it easy to get absurd results. For example, if $P(D|\theta_{yes}) = .6$ and $P(D) = .5$, setting $P(\theta_{yes}) = 1$ will make $P(\theta_{yes} \mid D) = .6/.5 > 1$: but no probability can go over 1!

We can solve all of these problems at the same time, by exploiting that apparently silly mathematical trick we saw earlier, whereby $P(D)$ can be written as the sum of $P(D|\theta)$ for all possible θs. For binary data where θ relates the outcome "yes" versus "no", $P(D|\theta)$ can be written like so (note the use of the addition rule of probability theory, since responses can't be "yes" and "no" at the same time):

$$\begin{aligned} P(D) &= P(D,\theta_{yes}) + P(D,\theta_{no}) \\ &= P(D|\theta_{yes})\cdot P(\theta_{yes}) + P(D|\theta_{no})\cdot P(\theta_{no}) \end{aligned}$$

Plugging this into the simplest version of Bayes's rule gives us the following slightly more complex version for analyzing binary data:

Bayes's rule (for binary data): $$\mathrm{P}(\theta_{yes}|\mathrm{D}) = \frac{\mathrm{P}(\mathrm{D}|\theta_{yes})\cdot\mathrm{P}(\theta_{yes})}{\mathrm{P}(\mathrm{D}|\theta_{yes})\cdot\mathrm{P}(\theta_{yes})+\mathrm{P}(\mathrm{D}|\theta_{no})\cdot\mathrm{P}(\theta_{no})}$$

Note that Bayes's rule now looks more clearly like a proportion: P(θ_{yes}|D) is the ratio of the situation where the hypothesis is true, P(D|θ_{yes})·P(θ_{yes}), out of all possible P(D) = P(D|θ_{yes})·P(θ_{yes}) + P(D|θ_{no})·P(θ_{no}) (formally that's *something*$_{yes}$ / (*something*$_{yes}$ + *something*$_{no}$)).

For example, consider a common scientific and practical situation, where a doctor is trying to diagnose an illness (θ) from observed symptoms (D); see Operskalski & Barbey (2016) for more discussion of this kind of problem, and some visual aids that may help make it more intuitive. Since this is a linguistics book, we'll imagine that the illness relates to language.

Suppose there's a Martian child who still hasn't begun to talk by the age of two. The scariest possible cause of this observation is Bloopy Syndrome (θ_{Bloopy}), which is guaranteed to prevent the child from ever developing language: P(D_{NoTalk}|θ_{Bloopy}) = 1. It's also known that most Martian two-year-olds without Bloopy Syndrome can talk already: P(D_{NoTalk}|$\theta_{NoBloopy}$) = .1. However, Bloopy Syndrome is very rare: P(θ_{Bloopy}) = .0001, which means that P($\theta_{NoBloopy}$) = 1-P(θ_{Bloopy}) = .9999. What is the probability P(θ_{Bloopy}|D_{NoTalk}) that the child really has Bloopy Syndrome?

Let's plug the numbers we have into Bayes's rule. This is easy to compute in either R or Excel (aside from making sure you don't mix up the numbers):

$$\mathrm{P}(\theta_{Bloopy}|D_{NoTalk}) = \frac{\mathrm{P}(D_{NoTalk}|\theta_{Bloopy})\cdot\mathrm{P}(\theta_{Bloopy})}{\mathrm{P}(D_{NoTalk}|\theta_{Bloopy})\cdot\mathrm{P}(\theta_{Bloopy})+\mathrm{P}(D_{NoTalk}\mathrm{D}|\theta_{NoBloopy})\cdot\mathrm{P}(\theta_{NoBloopy})}$$

```
NoTalk.Bloopy = 1
NoTalk.NoBloopy = 0.1
Bloopy = 0.0001
NoBloopy = 0.9999
Bloopy.NoTalk = (1*Bloopy)/(NoTalk.Bloopy*Bloopy+NoTalk.NoBloopy*NoBloopy)
Bloopy.NoTalk
```

```
[1] 0.0009991008
```

Thus despite the scariness of Bloopy Syndrome (it always prevents talking) and the rarity of this symptom without Bloopy Syndrome, the low prior probability of this syndrome results in a low posterior probability P(θ_{Bloopy}|D_{NoTalk}) too: only around .001. This is a higher probability than our prior probability of P(θ) = .0001, since the lack of talking does provide some evidence to support it, but our evidence isn't dramatic enough, and could be explained other ways as well. Thus we probably don't have to worry about Bloopy Syndrome, unless we get more evidence in support of this hypothesis, just as the cop did when he observed not just the alarm, but also the broken window, the man with the mask, and the bag of jewels.

Poldrack (2006) illustrates the Bayesian formula for binary data with an example from neurolinguistics, where he counts the number of studies in the literature showing activation versus non-activation in a certain brain region in language versus non-language studies:

	Language study	Not language study
Activated	166	199
Not activated	703	2154

Of course we could use ordinary frequentist statistics to run a two-way chi-squared test on this contingency table, but this would not let us start from a default prior assumption. In his case, he starts from the assumption that the probability of language being activated in an unknown task is 50/50, so P(language) = P(not-language) = .5. Based on this, and the complex Bayes' rule, he derives the posterior probability of .69. Can you get the same result? (Kruschke, 2011, p. 74, assigns the same exercise, but for some reason it's dropped in the revised edition of Kruschke, 2015).

2.2.3 Naive Bayes classifiers

The math of probability means that we can incorporate new information in a later cycle of the analysis, when we treat the posterior probability as the prior probability as new evidence comes in, or we can combine all of the information together in one step; we get exactly the same results either way. If we also make the naive assumption that each piece of the information is independent of all of the others, then we get something called a **naive Bayes classifier**, which, as the name suggests, can be very useful for classifying things.

Recent examples of this method, as applied to linguistics, can be seen in Lewis (1998) and Jurafsky & Martin (2009). But the method is quite old: one of the first linguistic applications of Bayesian logic was Mosteller & Wallace (1963) (see nontechnical summary in Salsburg, 2001, pp. 130-135). These researchers wanted to determine the authorship of twelve disputed papers from early American history: both Madison and Hamilton claimed to have written them. Both authors wrote a lot, so the prior probabilities for both were essentially equal. But it turns out that writers naturally differ in the use of function morphemes like *upon*, and in other writings, Madison almost never used *upon* while Hamilton used it a lot: P(*upon*|Madison) = Low and P(*upon*|Hamilton) = High. In the disputed papers, *upon* actually appears quite rarely, so Bayes's rule suggests that Madison was the more likely author.

To get a concrete feel for how a naive Bayes classifier works, here's a simplified version of the Martian (but Chinese-like) classifier example from the logistic regression chapter. What is the probability that the Martian word for an oblong animal should take the classifier *tiao*? This question involves two observed properties (oblong, animal), so what we want to compute is P(*tiao*|oblong,animal). Applying Bayes's rule to this situation, we get this equation:

$$P(tiao|\text{oblong, animal}) = \frac{\mathbf{P(oblong, animal}|\boldsymbol{tiao}\mathbf{)}\cdot P(tiao)}{\mathbf{P(oblong,animal}|\boldsymbol{tiao}\mathbf{)}\cdot P(tiao) + \mathbf{P(oblong, animal}|\neg\boldsymbol{tiao}\mathbf{)}\cdot P(\neg tiao)}$$

Now we apply the conditional probability rule to unpack the likelihoods bolded above, taking advantage of the independence assumption so we can use the following equalities:

P(animal|*tiao*,oblong) = P(animal|*tiao*)
P(oblong|*tiao*,animal) = P(oblong|*tiao*)
P(oblong,animal|*tiao*) = P(oblong|*tiao*)·P(animal|*tiao*)
P(oblong,animal|¬*tiao*) = P(oblong|¬*tiao*)·P(animal|¬*tiao*)

This ends up giving us the following formula:

$$P(tiao|\text{oblong,animal}) = \frac{P(\text{oblong}|tiao)\cdot P(\text{animal}|tiao)\cdot P(tiao)}{P(\text{oblong}|tiao)\cdot P(\text{animal}|tiao)\cdot P(tiao) + P(\text{oblong}|\neg tiao)\cdot P(\text{animal}|\neg tiao)\cdot P(\neg tiao)}$$

We can try this with a Martian data set obeying the independence assumption:

```
tiao = c(1,1,0,0); oblong = c(1,1,0,0); animal = c(1,0,1,0)
classifiers = data.frame(tiao,oblong,animal)
rownames(classifiers) = c("snake","rope","bird","apple")
classifiers
      tiao oblong animal
snake    1      1      1
rope     1      1      0
bird     0      0      1
apple    0      0      0
```

We can compute the six probabilities we need to apply Bayes's rule from the data:

```
Ptiao = sum(tiao)/length(tiao) # Counts each tiao, divides by the total
Pnottiao = sum(1-tiao)/length(tiao) # Counts each non-tiao, divides by the total
Poblong.tiao = sum(oblong*tiao)/sum(tiao) # oblong*tiao = 1 only if both are 1
Panimal.tiao = sum(animal*tiao)/sum(tiao)
Poblong.nottiao = sum(oblong*(1-tiao))/sum(1-tiao)
Panimal.nottiao = sum(animal*(1-tiao))/sum(1-tiao)
```

If we plug all of these values into Bayes's rule to compute P(*tiao*|oblong,animal), what we are actually computing is the probability that a thing that is both oblong and an animal will take the *tiao* classifier. Since in our data set, there is just one thing like this (snake) and it does take *tiao*, the probability we get is one:

```
Poblong.tiao*Panimal.tiao*Ptiao /
 (Poblong.tiao*Panimal.tiao*Ptiao + Poblong.nottiao*Panimal.nottiao*Pnottiao)

[1] 1
```

But we can also use this model to make predictions when we only have partial information. For example, we may know that something is an animal, but we don't know whether it's oblong. In this case, we might leave out this feature:

```
Panimal.tiao*Ptiao / (Panimal.tiao*Ptiao + Panimal.nottiao*Pnottiao)

[1] 0.5
```

That is, an unknown animal has a 50% chance of taking *tiao*. That's because in our pre-classified data, we have exactly two animals, and one takes *tiao* (snake) and the other doesn't (bird).

2.2.4 Bayes's rule for continuous data

Yet another complexity that arises in applying Bayes's rule for real-life problems is that P(θ) often relates to continuous values. For example, we might believe $\theta = 23$ to be the most plausible value, but admit that even if θ were a little higher or lower it would still be mostly plausible, though if it were much higher or lower its plausibility would drop dramatically. In this case, a bell-shaped curve may be a good choice (though not necessarily a normal distribution).

Since we're now dealing continuous values, though, we can't split up all the possibilities into discrete parts and add them up, as we did for θ_{yes} and θ_{no}. Instead, we have to imagine an infinite number of infinitesimal probabilities that all get summed up using the magic of calculus, specifically using integrals (積分). This turns the sum operator Σ for adding up discrete values into the ∫ operator for integrals, where $\int f(x)\, dx$ stands for the sum of all values of the $f(x)$ function across all continuous values of x:

Bayes's rule for continuous-valued hypotheses: $\mathrm{P}(\theta|D) = \frac{\mathrm{P}(D|\theta)\cdot\mathrm{P}(\theta)}{\int \mathrm{P}(D|\theta)\cdot\mathrm{P}(\theta)d\theta}$

But even this rather scary-looking formula isn't quite realistic enough. The hypothesized parameter θ is not floating around by itself; it's actually part of some larger model *M*. For example, *M* might be a regression model, and θ is just a regression coefficient in this model. So θ is conditional on M, and the prior is not simply P(θ) but P(θ|M), and the likelihood is not P(D|θ) but P(D|θ,M):

$$\mathrm{P}(\theta|\mathrm{D},\mathrm{M}) = \frac{\mathrm{P}(\mathrm{D}|\theta,\mathrm{M})\cdot\mathrm{P}(\theta|\mathrm{M})}{\int \mathrm{d}\theta\ \mathrm{P}(\mathrm{D}|\theta,\mathrm{M})\cdot\mathrm{P}(\theta|\mathrm{M})}$$

This is the kind of equation that we'll be using implicitly in the applications below, though we'll never need to actually compute this directly ourselves. But maybe now you can see why Bayesian statistics is heavily dependent on computers.

2.3 Testing evidence *for* the null hypothesis

Though we've just started looking at Bayesian statistics, you already know enough to make one very useful application of it: the **Bayes factor**. This is used to compare two competing models of the same data, to see which is more likely. If one of these models happens to be the null hypothesis, the Bayes factor actually allows you to decide if the null hypothesis is *better* than the alternative: we're not restricted to merely *not rejecting* the null hypothesis, as in traditional statistics.

The concept of the Bayes factor is very simple. We want to know which of two competing models has the higher posterior probability given our data, assuming that both have the same prior probabilities. Bayes's rule computes these posterior probabilities for us, so if we divide one by the other, with the one we prefer on top, we will get 1 if the two models give the same posterior probability, some big number of our preferred model is more probable, and some tiny number if the other model is more probable. Since the data are the same in both cases, and we're assuming that the prior probabilities are the same, everything will cancel out in this division except for the one thing that differs between the two models, namely their likelihoods. Remember that the likelihood is the P(*D*|θ) part of Bayes's rule, i.e., the likelihood that we'd get our observed data *D* given the hypothesis θ.

So suppose we want to compare the null hypothesis model M_0 against the alternative model M_1, to see if the evidence supports M_0 over M_1 (again, this is impossible to do in traditional statistics). The equation below shows Bayes's rule for M_0 divided by Bayes's rule for M_1. The bolded part below is the ratio of the two likelihoods:

Dividing Bayes's rule for two models: $$\frac{\mathrm{P}(\mathrm{M}_0|\mathrm{D})}{\mathrm{P}(\mathrm{M}_1|\mathrm{D})} = \frac{\mathbf{P(D|M_0)}}{\mathbf{P(D|M_1)}}\cdot\frac{\mathrm{P}(\mathrm{M}_0)}{\mathrm{P}(\mathrm{M}_1)}$$

We're comparing M_0 and M_1 because we can't decide ahead of time which is better, so we assume that the prior probabilities are the same for both, so $P(M_0)/P(M_1) = 1$. That is, we assume that **prior odds** for the two competing models is 1:1. This assumption allows us to compute the Bayes factor in favor of the null hypothesis:

Bayes factor in favor of the null hypothesis: $$B_{01} = \frac{P(D|\theta,M_0)}{P(D|\theta,M_1)}$$

If you are a good statistics student, there should be something familiar about this. The Bayes factor is a likelihood ratio, but haven't we already been doing a lot of **likelihood ratio tests** for traditional regression models? Exactly: the English mathematician Harold Jeffreys (1891-1989), the inventor/discoverer of the Bayes factor, somehow managed to be friends with Fisher, and thought that Fisher's **maximum likelihood** method (later crucial for logistic regression and mixed-effects modeling) was fundamentally a Bayesian method (see McGrayne, 2011, for the story).

Let's try a trivial example (adapted from Dienes, 2008) before we go on to more realistic cases. Suppose you see a strange woman at an airport somewhere and wonder if she knows English. If she does, it is quite likely that if you say "hello" to her, she will reply with "hello". Maybe she won't (deaf, tired, rude), but let's say the likelihood P("hello"|English) = 9/10. By contrast, if she knows no English, she might only reply "hello" if she likes repeating things, which suggests a low likelihood, say P("hello"|no-English) = 1/10. So you go up to her, say "hello", and she replies "hello" back. What can we conclude?

Well, the Bayes factor in favor of her knowing English is quite large:

P("hello"|English) / P("hello"|no-English)

(9/10) / (1/10)

```
[1] 9
```

Jeffreys (1961) suggested that a Bayes factor greater than 3 represents "some evidence" for the hypothesis on the top (numerator) of the ratio, greater than 10 gives "strong evidence", and greater than 30 "very strong evidence". So in this case, we have "some evidence" that the woman knows English. A Bayes factor less than 1 tends to favors the hypothesis in the denominator (bottom of the ratio), so you should flip the ratio around and calculate it again to apply Jeffreys's suggestion. The specific numbers aren't crucial, except that for a large enough sample, a Bayes factor of 3 *against* the null hypothesis roughly corresponds to $p < .05$ in traditional null hypothesis testing (as implied by Johnson, 2005, p. 695, Figure 1).

Calculating Bayes factors in more realistic cases is hard to do by hand, but quite easy with R. The conceptually simplest method is to exploit the link with likelihood ratio tests, and take another look at the **Bayesian information criterion** (BIC) that I briefly mentioned in the previous chapter. This is the value that R reports for likelihood ratio tests along with the related AIC (Akaike information criterion) that we've used more often.

The BIC is calculated from the maximum likelihood that's used to fit a model. The equation looks like this, where, L = likelihood, k = parameters, N = sample size:

Bayesian information criterion: $\mathrm{BIC} = -2 \cdot \ln(\mathrm{L}) + \mathrm{k} \cdot \ln(\mathrm{N})$

If the sample is large enough, and the model is not too complex (basically, anything less complex than a mixed ANOVA), we can easily use BIC to estimate a Bayes factor for the null hypothesis model M_0 against the alternative M_1 (Raftery, 1995; Rouder et al., 2009). We do this basically by inverting the above equation (we divide by -2, and then the exponential function *exp*(*x*) = e^x cancels the natural logarithm function *ln*):

Estimated Bayes factor in favor of the null hypothesis: $\mathrm{B}_{01} \approx \exp\left(\frac{\mathrm{BIC}_1 - \mathrm{BIC}_0}{2}\right)$

In R, this formula looks like this, where the **BIC()** function extracts the BIC values from models M0 and M1, created using **lm()** or similar functions:

```
B01 = exp((BIC(M1) - BIC(M0))/2) # Just an abstract example for you to look at
```

2.3.1 Bayesian *t* tests

Let's try this out on some data, specifically the kind of data that would be analyzed with a *t* test in frequentist statistics. Maybe you remember **voweldurationsR.txt**, which contains (fake) vowel lengths from a single speaker for /i/ and /u/ in different words:

```
vd = read.delim("voweldurationsR.txt")
head(vd)
```

```
     Vowel  Duration
  1      i       283
  2      i       325
  3      i       426
  4      i       340
  5      i       352
  6      i       278
```

When we do an unpaired (homoscedastic) *t* test on this, it's not significant: $t(46) = 1.328$, $p = .1907$ (try it!):

```
t.test(Duration~Vowel, data=vd, var.equal=T)
```

In traditional statistics, all we can say is that we are *not justified in rejecting* the null hypothesis; $p > .05$ does not mean that we can be confident that the vowel durations are truly equal.

Now, remember that the unpaired homoscedastic *t* test is really a simple linear regression, so the following gives exactly the same *t*, *df*, and *p* values (try it!):

```
summary(lm(Duration~Vowel, data=vd))
```

Linear models like this have BIC values, so let's get it with the **BIC()** function (all caps):

```
BIC1 = BIC(lm(Duration~Vowel, data=vd))
```

Now we need the BIC for the null hypothesis that Vowel has no effect on Duration. So we remove Vowel from the model to get the intercept-only model:

```
BIC0 = BIC(lm(Duration~1, data=vd))
```

Now we estimate the Bayes factor favoring the null hypothesis over the alternative hypothesis. The sample size is pretty small, so the estimate will be somewhat off:

```
B01 = exp((BIC1-BIC0)/2)
B01
```

```
[1] 2.808711
```

The value $B_{01} = 2.8$ is right below Jeffreys's arbitrary value of 3 for "some evidence", but assuming the true value is not much smaller than this, we are at least justified in saying that we (almost) have some evidence in favor of the null hypothesis here. With traditional statistics, we're not even allowed to say that!

Computing more accurate Bayes factors to test evidence for the null hypothesis requires calculus and integrals, but fortunately R has packages that have all of this built in.

To get a sense of how the math works, let's look at the method described by Rouder et al. (2009) for computing good estimates of the Bayes factor for one-sample *t* tests. The first part of their method is to choose a plausible alternative hypothesis. If we choose an alternative with an effect size too far from the observations, the Bayes factor will automatically favor the null hypothesis, so we wouldn't learn anything from new data. For example, if our likelihood says that we should treat the durations of /i/ and /u/ as "the same" as long as the difference is less than 1000 ms, that would be pretty useless.

So they suggest that we should assume a bell-shaped prior distribution for the alternative effect size, so that most of the area close to the null hypothesis. The prior implicitly assumed by the above BIC estimate is such a distribution, since it's based on the residuals for the intercept-only model, which are assumed to conform to the standard normal distribution. But we also don't want the alternative prior to be too narrow, or else it looks like we're cheating the opposite way, by making the prior too informative (i.e., building in a bias for the alternative).

Rouder et al. (2009) suggest a compromise: the prior should be the $t(1)$ distribution (i.e., the t distribution for $df = 1$, also known as the **Cauchy distribution**). This is less informative than the standard normal distribution, since it has "fat" tails. In case you've forgotten what fat t tails look like, take a look at Figure 1, plotted with the following code:

```
curve(dnorm(x), -3,3, ylab = "Density") # Plot normal distribution (solid line)
curve(dt(x,1), -3, 3, add=T, lty=2) # Add Cauchy distribution (dashed line)
legend("topright",lty=c(1,2),legend=c("Normal","Cauchy")) # Explain stuff
```

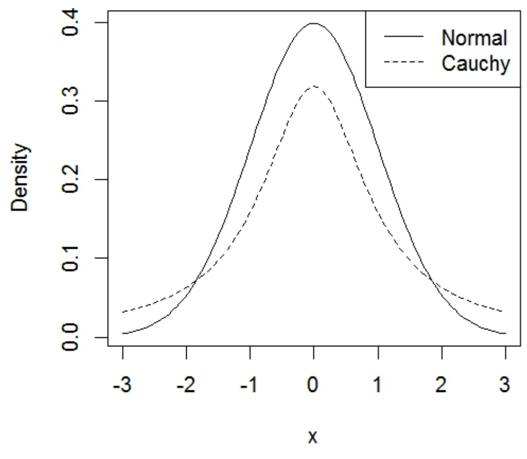

Figure 1. The fat tails of the Cauchy distribution ($t(1)$).

Assuming the Cauchy prior for the effect size, and priors for data variance suggested by Jeffreys (1961) and for effect size variance suggested by Zellner & Siow (1980), which together Rouder et al. (2009) call the JZS prior, they calculate the Bayes factor for the one-sample t test by taking the sample size N and the t and df values from the traditional analysis (e.g., $t(46) = 1.328$ for the vowel durations), where df is symbolized as v (to make it just one letter long), and then put these three values into the hilariously complicated equation below (π and e are the famous constants, and g is just used to define the integral). This formula can also be scaled up or down a bit with a factor called r (which we'll see shortly).

Please memorize for the exam: $$B_{01} = \frac{\left(1+\frac{t^2}{v}\right)^{-(v+t)/2}}{\int_0^{\infty}(1+Ng)^{-\frac{1}{2}}\left(1+\frac{t^2}{(1+Ng)v}\right)^{-(v+1)/2}(2\pi)^{-1/2}g^{-3/2}e^{-1/(2g)}dg}$$

Fortunately, we don't have to apply this formula by hand, since the authors built an R package: **BayesFactor** (Morey & Rouder, 2014). This estimates integrals with iterative algorithms like resampling. Let's try it on the vowel durations!

After you install **BayesFactor** and load it up, the first thing it does is point you to an HTML manual, which unlike most R manuals, is actually written for ordinary human beings:

```
library(BayesFactor) # It says: "Type BFManual() to open the manual."
BFManual() # There it is!
```

To compute the Bayes factor for an unpaired *t* test of the vowel durations, we do the following. It doesn't let us use the formula notation, so we have to extract the /i/ and /u/ values first:

```
i.Dur = vd$Duration[vd$Vowel=="i"]
u.Dur = vd$Duration[vd$Vowel=="u"]
```

Now we run the **ttestBF()** function of the **BayesFactor** package:

```
ttestBF(i.Dur,u.Dur)

Bayes factor analysis
--------------
[1] Alt., r=0.707 : 0.5871358 ±0.01%

Against denominator:
  Null, mu1-mu2 = 0
---
Bayes factor type: BFindepSample, JZS
```

Checking **?ttestBF** clarifies that the prior here is actually scaled to be a bit narrower than the Cauchy distribution, by using that *r* factor; *r* = 0.707 reflects the default scaling factor for two-sample tests (0.707 = **sqrt(2)/2**). The number following this is the Bayes factor in favor of the alternative hypothesis ("Alt."), so to get the Bayes factor in favor of the null hypothesis we divide it into one: 1/0.5871358 = 1.703183. We can test the evidence in *favor* of the null directly by using **1/ttestBF()**:

```
1/ttestBF(i.Dur,u.Dur)
```

```
Bayes factor analysis
--------------
[1] Null, mu1-mu2=0 : 1.703183 ±0.01%

Against denominator:
  Alternative, r = 0.707106781186548, mu =/= 0
---
Bayes factor type: BFindepSample, JZS
```

This analysis is more accurate than the one we did with BIC, since our sample is so small. Sadly, it seems that our sample may be too small for us to be very confident about the null hypothesis here, since the Bayes factor for it (1.7) is now far below Jeffreys's "some evidence" value of 3.

2.3.2 Bayesian chi-squared tests

The **BayesFactor** package can do far more than just one-sample or unpaired *t* tests. Explore the manual for instructions on how to do this for much more complex cases, including chi-squared tests, ANOVA, and multiple linear regression.

To illustrate, let's look at the Bayesian version of a chi-squared test. We'll try this out on something like our Mandarin sociolinguistic data from earlier, but now in a data set where a traditional frequentist analysis gives us a null result. So in Table 2, it seems that there is no clear correlation between the vowel and coda.

Table 2. The frequencies of various consonant phonemes and allophones

rime	/aŋ/	/iŋ/
[n] tokens	190	112
[ŋ] tokens	57	27

Here's the data in R:

```
socdata = matrix(c(190,57, 112, 27), ncol = 2) # Entering by columns to match table
rownames(socdata) = c("n","N") # "N" for velar nasal
colnames(socdata) = c("a","i")
socdata
```

```
     a   i
n  190 112
N   57  27
```

If there is any effect, it must be that the vowel affects the coda, and not the other way around, so the proportions we want to compare are these (or similarly for [ŋ]):

P([n]|a) = 190/(190+57) = .769
P([n]|i) = 112/(112+27) = .806

Those values seem pretty close, and indeed, a frequentist two-way chi-squared test gives us a null result:

chisq.test(socdata)

```
        Pearson's Chi-squared test with Yates' continuity correction

data:   socdata
X-squared = 0.49891, df = 1, p-value = 0.48
```

Can a Bayesian analysis help clarify what this result really means? The **BayesFactor** package has a function called **contingencyTableBF()**, and it turns out to have some nice features even beyond the null result issue.

Here's one way we can apply it to our data set. The argument **sampleType = "hypergeom"** makes the null hypothesis the same as that assumed in the traditional chi-squared test, namely that both the column and row marginals are fixed, and the values in the cells are conditional on those fixed values (the fancy-sounding **hypergeometric distribution** is just like the binomial distribution, except that the binomial one assumes values are randomly taken *with* replacement, so earlier-chosen values don't affect later-chosen ones, whereas the hypergeometric distribution assumes values are random chosen *without* replacement, so that earlier-chosen values, as are fixed in the rows and columns, do affect later choices).

socdata.bf = contingencyTableBF(socdata, sampleType = "hypergeom")

As with **ttestBF()**, by default **contingencyTableBF()** gives you the Bayes factor for comparing the alternative hypothesis against the null hypothesis, but we already know the alternative hypothesis fails here, so let's use the inverse for the Bayes factor in favor of the null hypothesis (i.e., against the alternative hypothesis of non-independence of vowels and codas):

1/socdata.bf

```
Bayes factor analysis
--------------
[1] Indep. (a=1) : 6.231246 ±0%

Against denominator:
  Alternative, non-independence, a = 1
---
Bayes factor type: BFcontingencyTable, hypergeometric
```

Well now, that Bayes factor of 6.23 is far above the minimum of 3 for "some evidence" in favor of the null hypothesis, though it's not up to the level of 10 for "strong evidence".

But is this really the null hypothesis we want to test? While the standard chi-squared test assumes that we want to keep both row and column marginals fixed, in this case we actually don't, since in the real world, the vowels are determined first (in the mental lexicon), and the codas are only determined later (in the phonetics). So it would actually make more sense only to fix the columns (the vowels), as our conditional factor. Using **contingencyTableBF()**, we can do just that:

```
socdata.bf2 = contingencyTableBF(socdata, sampleType = "indepMulti",
    fixedMargin="cols")
1/socdata.bf2
```

```
Bayes factor analysis
--------------
[1] Indep. (a=1) : 6.576953 ±0%

Against denominator:
  Alternative, non-independence, a = 1
---
Bayes factor type: BFcontingencyTable, independent multinomial
```

The Bayes factor in favor of the null hypothesis is about the same as before, but not exactly, since the null hypothesis now allows the row values to vary freely. We can even run an analysis where both row and columns can vary freely, and all that is fixed is the total number of data points (386). This time the Bayes factor is noticeably lower than before (though still over 3), since with so much freedom to randomly vary the cell values, it's easier for our data set to be consistent with the null hypothesis by chance alone.

```
socdata.bf3 = contingencyTableBF(socdata, sampleType = "jointMulti")
1/socdata.bf3
```

```
Bayes factor analysis
--------------
[1] Indep. (a=1) : 4.765134 ±0%

Against denominator:
  Alternative, non-independence, a = 1
---
Bayes factor type: BFcontingencyTable, joint multinomial
```

So this Bayes factor stuff seems pretty useful. You can use it to analyze the same types of data we've been analyzing with *t* tests, chi-squared tests, and so on, and even if we don't get a null result, a Bayes factor over 3 (or 10 or 30) in favor of the alternative hypothesis gives us at least as much confidence that the pattern is “real” as traditional frequentist *p* value below .05 would give us, without the confusion over what the p value actually represents, and if we do get a null result, we can check out the inverse of the Bayes factor to see if that goes over 3 (or 10 or 30), and if we want to run special kinds of two-way chi-squared tests that make different assumptions about the marginals, we can do that too.

Still, you should know that not all Bayesian statisticians like Bayes factors; in the 759 pages of Kruschke (2015), he only mentions Bayes factors briefly, and mainly just to dismiss them. This is because Bayes factors suffer from one of the same logical flaw as *p* values. Namely they assume that we can treat the prior probabilities for the two competing hypotheses as if they are identical, but in real life, does anybody *really* believe that the null and alternative hypotheses are equally plausible? Thus many Bayesian statistics say that you should use *all* of Bayes’s rule, not just the likelihood parts (Lavine & Schervish, 1999).

3. A taste of Bayesian modeling

Bayesian models more complex than the simple ones above can be run in R, but so far, there is no “native” R package. That is, to run fancy models, all R can do is interface with non-R special-purpose Bayesian programs, including the aforementioned OpenBUGS and Stan, as well as other programs like JAGS (“Just Another Gibbs Sampler”; Johnson & Kuhn, 2013). Like R, these Bayesian programs are also free (OpenBUGS only works in Windows, but the others work on all platforms), and as mentioned, there are R packages designed to interface with them (**BRugs** for OpenBUGS, **rjags** for JAGS, **rstan** for Stan). Still, you have to download and install them separately from R, they use different syntax from R, and unlike the dominant position of R among quantitative linguists (aside from a few SPSS holdouts), it’s not clear which of these systems will the most useful to learn in the long run. For example, the first edition of Kutsche (2011) focused on R and OpenBUGS, but in the new edition of Kutsche (2015), the focus has shifted to JAGS and Stan (both with R interfaces). The last time I checked, R mostly has partial Bayesian packages; in addition to the **BayesFactor** package we used above, you can see more at http://cran.r-project.org/web/views/Bayesian.html).

So in the remainder of this chapter I’ll give an overview of three other useful things we can do with Bayesian statistics entirely within R: performing simple hypothesis testing on binary data, computing the Bayesian version of confidence intervals, and incorporating random and fixed variables (which we first saw in the paired *t* test, and in a much fancier form in mixed-effects modeling).

3.1 A Bayesian binomial test

Let's look at the simplest type of data that linguists might face: binary responses that are **independent and identically distributed** (**i.i.d.**, as statisticians abbreviate it), where we just want to know the overall proportion of "yes" vs. "no". In other words, we're going to look at a Bayesian version of the binomial test.

In this kind of test, the hypothesized parameter θ is seen as the built-in probability of getting "yes" vs. "no" (e.g., in people judging the acceptability of a sentence). Before we collect any data, we might assume that θ is around 0.5 (equal chances of "yes" or "no"): this is our prior. So in order to compute the posterior distribution, which reflects what we learn from the data about θ, we need to learn about three things in applying Bayes's rule: how to represent this prior formally, how to represent the likelihood, and how to compute the posterior from the prior and likelihood.

The most fundamental step is to formalize the likelihood, since this encodes the relationship between the data and the hypothesized parameter: $P(D|\theta)$. In our case, what we need is called the **Bernoulli likelihood function**. The weird name comes from the fact that random binary responses are called **Bernoulli trials**, after Swiss mathematician Jacob Bernoulli (1655-1705), who was one of the founders of probability theory. For example, if you run a simple syntax experiment where you ask a series of people if they accept or reject a sentence, that's a series of Bernoulli trials ("trial" as in an experiment).

Since each response is i.i.d., the probability of getting any particular series of responses is just the product of the probability of getting each response in the series. This situation is subtly different from a binomial distribution, since we're not assuming a fixed total *N* ahead of time. As we saw earlier in this chapter, we can't be sure that an experimenter using the traditional binomial test has really fixed *N* ahead of time, but for this Bayesian analysis we aren't fixing *N* ahead of time, so this worry is gone.

Formally speaking, the likelihood function is as follows, where the *y*s represent the *N* binary data points in the data set *D* (z = number of "yes") and the big pi Π means "product" (積), which is an operator like the sigma Σ (for "sum"), except that it involves multiplying instead of adding.

$$P(y_1, \ldots, y_N |\, \theta) = \prod_i P(y_i|\theta) = \prod_i \theta^{y_i} (1-\theta)^{(1-y_i)} = \theta^z (1-\theta)^{(N-z)}$$

The formula only looks complex because it's meant to be as general as possible, but it's not actually saying anything you don't already know. For example, suppose $D = \{1,1,0\}$ (e.g., our syntax experiment gives us the responses "yes", "yes", "no" for the test sentence), and suppose $\theta = ½$, which is a possible hypothesis about the overall probability of getting 1 (note that I'm not calling it a "null hypothesis", since Bayesian statistics doesn't need that). In that case:

$$P(D|\theta) = P(1,1,0|½) = P(1|½)\times P(1|½)\times P(0|½) = [(½)^1\cdot(½)^0]\times[(½)^1\cdot(½)^0]\times[(½)^0\cdot(½)^1] = (½)^3$$

Now for the prior. Remember that this is not just a **point estimate** like $\theta = .5$, but a whole distribution. That's why the above equation is so complex looking: it allows us to compute probabilities for any possible θ between 0 and 1, and thus compute an entire distribution.

In our case, a single point value doesn't really make sense anyway. After all, we might not be very confident that $\theta = .5$, so we would want a wider distribution, where $\theta = .5$ may be the most believable, but $\theta = .7$ wouldn't be out of the question either. Or we may want to make our prior completely uninformative, where θ has an equal chance to be any value from 0 (never "yes") to 1 (always "yes"). We might even believe that the prior is bimodal, with peaks for 0 and 1 and a valley in between. Our prior distribution should also relate to our situation in some natural way, namely a series of Bernoulli trials, where each trial can only take two values. So the distribution should only be defined between 0 and 1, and should be generated by values relating to the numbers of "yes" and "no" responses.

Finally, remember that we'll need to combine the prior with our likelihood function to get a posterior distribution, and if this process is hard to compute, we'll be in trouble. In fact, the prior and posterior should ideally come from the same distribution family, so we could run through the cycle again with new data, using our old posterior as the new prior on each cycle. This means that we need a family of distributions that is flexible enough to represent a narrow bell shape (if we're very confident in our prior $\theta = .5$), or a uniform distribution (for a completely uninformative prior), or anything in between. With fast computational algorithms, we don't really have to follow these constraints, but in this simple case, it *is* possible follow them.

The trick is to use yet another distribution family called the **beta distribution**, which was first described informally by Bayes himself, developed further by Pearson, but used today mainly by Bayesian statisticians precisely to deal with the above technical problems.

No matter what its shape is, the beta distribution only varies θ from 0 to 1, since it relates to the proportion of "yes" responses in a series of Bernoulli trials. It's built using two parameters with the unimaginative names of a and b, which have to be positive, since they relate to the numbers of "yes" (a) and "no" responses (b). Kruschke (2015, p. 128) plots beta distributions for various values of a and b, and we can do so too using the following R code. This uses the function **dbeta()** (remember that "d" = density), in which the two parameters **shape1** and **shape2** (just as unimaginative as a and b). The result is shown in Figure 2.

```
par(mfrow=c(5,5), # Prepares 5 x 5 mini-plots
 mai=c(0,0,0,0)) # Gets rid of the four margins around each mini-plot
for (b in c(0.5,1:4)) {
 for (a in c(0.5,1:4)) {
  curve(dbeta(x,a,b),0,1, # plots beta curve with a & b values from x=0 to 1
   xaxt="n",yaxt="n",ylim=c(0,3)) # hides axes and keeps plots same size
  text(0.5,2.5,labels=paste("a=",a,",b=",b,sep="")) # shows a & b for each plot
 }
}
```

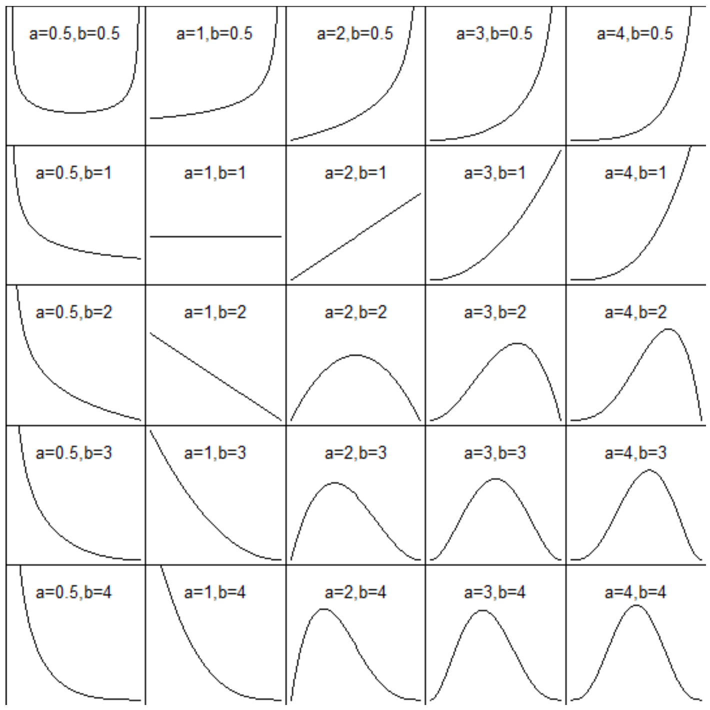

Figure 2. Various beta distributions

The shapes you get do make some sort of sense. As Kruschke (2015) points out, when they are used to generate a prior distribution, *a* and *b* reflect how much prior evidence you have (or believe in), with *a* reflecting "yes" (1) and *b* reflecting "no" (0), and the beta distribution itself reflects your prior belief about the proportion *a*/(*a*+*b*).

So if *a* and *b* are both 1, this is like only having two prior data points, with one "yes" and one "no": too little information to posit anything but an uninformative uniform prior (a horizontal line). If *a* and *b* are both 4 (more data), we get a bell shaped distribution centered around 0.5. If *a* is 4 but *b* is 2, this gives a skewed distribution, with the peak closer to 1 (the *a* end) than to 0 (the *b* end). We can even get a bimodal distribution with *a* and *b* both set to 0.5 (indicating that each has a 50/50 chance, without much in between). The beta family is a very flexible distribution family!

The formula for the beta distribution is shown below. It looks scary, but unfortunately we'll need the details again shortly. The numerator (top) is just the Bernoulli function, applied to the parameters *a* (for the number of "yes" responses) and *b* (for the number of "no" responses). The denominator (bottom) is actually just a **normalizing constant** (the θ in the integral just indicates that the total area goes from $\theta = 0$ to $\theta = 1$), cleverly designed so that the area of the ratio adds up to 1, which has to be the case, since *beta(θ|a,b)* is a probability distribution.

The beta function: $beta(\theta|a,b) = \frac{\theta^{(a-1)}(1-\theta)^{(b-1)}}{B(a,b)} = \frac{\theta^{(a-1)}(1-\theta)^{(b-1)}}{\int_0^1 \theta^{(a-1)}(1-\theta)^{(b-1)}d\theta}$

Now we need a way to compute the posterior from the prior, after we've applied the likelihood function to our actual data set. As Kutsche (2015, p. 132) shows, the flexibility of the beta family and the simplicity of the Bernoulli function together imply that there is a fixed formula for the posterior, which can be found just by plugging everything into Bayes's rule. Namely, given the prior parameters *a* and *b*, the total number of data points *N*, the total number of "yes" (1) responses *z*, and the normalizing constant *B*, the posterior distribution is computed as follows. Note the bolded prior at the left: P(*θ*). Using algebra, this gets unpacked as the bolded portion of the ratio in the middle (see the *B(a,b)* denominator?).

$$P(\theta|z,N) =$$

$$\frac{P(z,N|\theta)\cdot \boldsymbol{P(\theta)}}{P(z,N)} = \frac{(\theta^z(1-\theta)^{N-z})\cdot \left(\boldsymbol{\theta^{(a-1)}(1-\theta)^{(b-1)}}\right)}{\boldsymbol{B(a,b)}\cdot P(z,N)} = \frac{\theta^{(z+a-1)}(1-\theta)^{(N+b-z-1)}}{B(z+a,N+b-z)}$$

$$= beta(\theta|z+a,N-z+b)$$

Putting this more informally, in the ratios, the top (numerator) is the same as the Bernoulli likelihood function, except with *z+a* instead of *z*, and *N+b* instead of *N*. Why? Because *a* is our prior information about "yes" (we observe *z* "yes"es, but already had *a* prior "yes"es), and *b* is our prior information about "no" (so we observe *N* total responses, but already had *b* prior "no"s in there too). On the bottom (denominator) we divide by the normalizing constant *B* defined for the current evidence to make sure the area of the distribution adds up to 1, as a probability distribution must. And *voilà*: we get another beta distribution!

This ugly-looking formula is actually quite simple to use, and it gives us the power to describe the posterior distribution, given the prior and the evidence. After all, R has functions for the beta distribution family, so all we have to do is plug in *a* and *b* for our prior and *z* and *N* for our data.

To see how this works, let's return to the case we discussed earlier in this chapter. Namely, a linguist asks 25 people to give binary good/bad judgments on a sentence, and only 8 of them accept it. Does this imply that the sentence is unacceptable? Remember that in traditional statistics, there are two different answers, depending on whether we fix the total *N* ahead of time and compute the probability of getting the *z* "yes" responses by chance (significant by a binomial test), or fix *z* and compute the probability of getting the total *N* (not significant by a negative binomial test). The reason for the two answers is that there are two different things that secretly might be going on the linguist's head: this is the subjective part of supposedly objective traditional statistics.

In Bayesian statistics, however, the subjective part is explicit: the prior probability. So it's quite easy to try out the prior you prefer, and then try out the prior your worst enemy would prefer, using exactly the same computations, in the hope of coming to some agreement about the best way to analyze the data.

Maybe you and your worst enemy agree that the best prior is the completely uninformative uniform prior, namely that horizontal line expressed with a beta distribution with *a* and *b* both 1. Then, just as in the original story, we observe that $N = 25$ and $z = 8$. This gives us the following posterior distribution:

$P(\theta|8,25) = beta(\theta|8+1,25\text{-}8+1) = beta(\theta|9,18)$

Or maybe you and your worst enemy agree on a prior where θ is very close to .5, so we set both *a* and *b* to a high number like 100 to make the beta distribution peak sharply at that point. This very informative or biased prior yields the following posterior distribution:

$P(\theta|8,25) = beta(\theta|8+100,25\text{-}8+100) = beta(\theta|108, 117)$

Using R's beta distribution functions, we can now plot the analyses associated with each of the two possible priors described above, resulting in Figure 3 (cf. Kutsche, 2011, p. 86):

```
likely.8.25 = function(theta) { theta^8 * (1-theta)^(25-8)}     # P(8,25|theta)
par(mfrow=c(3,2), mai=rep(0.5,4)) # Two columns of three plots, small margins
curve(dbeta(x,1,1),0,1,main="Uniform prior") # (x = theta: curve requires "x")
curve(dbeta(x,100,100),0,1,main="Informative prior") # Sharp peak near theta = 0.5
curve(likely.8.25(x),0,1,main="Likelihood") # Likelihood based on observations
curve(likely.8.25(x),0,1,main="Likelihood") # Likelihood (same!)
curve(dbeta(x,8+1,25-8+1),0,1,main="Posterior") # Posterior for uniform model
curve(dbeta(x,8+100,25-8+100),0,1,main="Posterior") # Posterior for other model
```

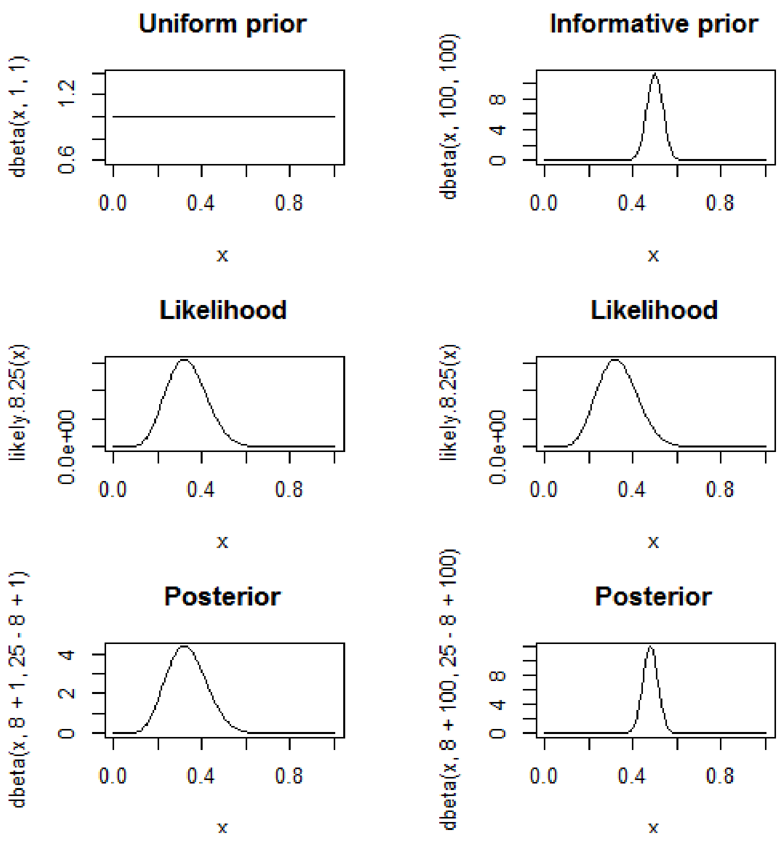

Figure 3. Left: Uniform prior, likelihood, posterior; right: biased prior, likelihood, posterior

This little exercise shows that if the prior is uninformative, as in the left side of Figure 3, the shape of the posterior is identical to that of the likelihood: everything we know is what we learned from the data. If it's very informative, as in the right side of Figure 3, the influence of our tiny data set is almost invisible, and the posterior remains almost identical to the prior: our prior beliefs are too strong, and the evidence too weak, to change our minds.

However, even if we have a very informative prior, if we collect enough data, the influence of the prior will weaken. For example, say we continue running the experiment so that *N* becomes ten times larger, and so does *z*. The ratio of *z*/*N* will remain the same as for the small sample, but as shown in Figure 4, we'll learn more about *θ*: the posterior will move a little away from the prior and closer to the data.

```
likely.80.250 = function(theta) {theta^80*(1-theta)^(250-80)}    # P(80,250|theta)
par(mfrow=c(3,1), mai=rep(0.5,4)) # One column of three plots, small margins
curve(dbeta(x,100,100),0,1,main="Informative prior") # Sharp peak at theta = 0.5
curve(likely.80.250(x),0,1,main="Likelihood") # Likelihood for larger data set
curve(dbeta(x,8+100,25-8+100),0,1,main="Posterior") # Shifted towards data
```

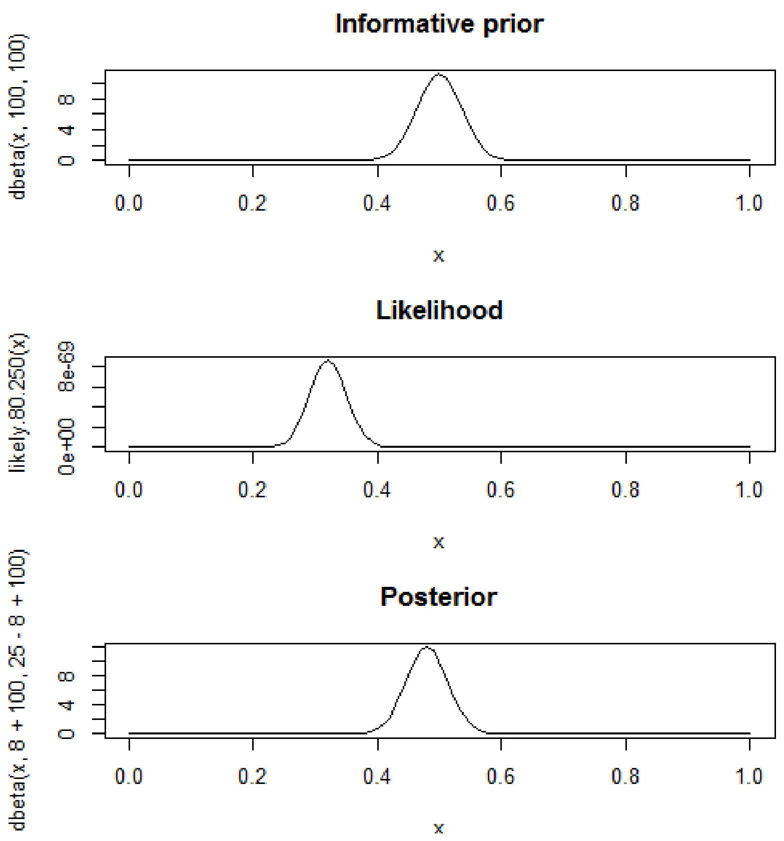

Figure 4. More data means a (slightly) greater shift from the prior to the posterior

In a real research situation, you should either choose an uninformative prior (to avoid fights with your critics), or choose an informative prior that you think your critics would also find believable. Besides the uniform prior, another type of uninformative prior was that recommended by Jeffreys (1961) (hence called the **Jeffreys prior**): set *a* and *b* to 0.5, to get a bimodal distribution, as shown in Figure 5 for our original small data set:

```
likely.8.25 = function(theta) { theta^8 * (1-theta)^(25-8)}      # P(8,25|theta)
par(mfrow=c(3,1), mai=rep(0.5,4)) # One column of three plots, small margins
curve(dbeta(x,0.5,0.5),0,1,main="Jeffreys prior") # Bimodal prior
curve(likely.8.25(x),0,1,main="Likelihood") # Likelihood for original small data set
curve(dbeta(x,8+0.5,25-8+0.5),0,1,main="Posterior") # Posterior for bimodal model
```

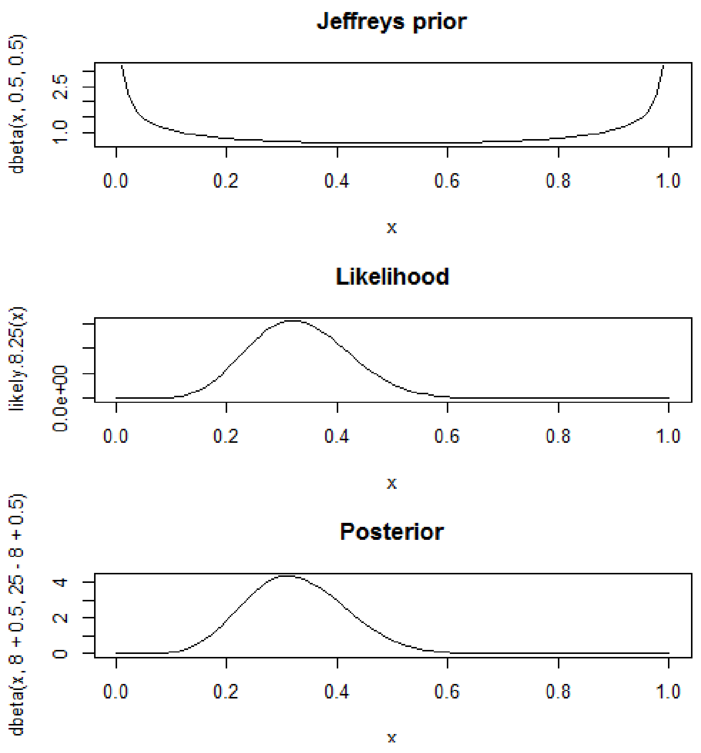

Figure 5. The effect of the Jeffreys prior

Now that we've computed a posterior distribution based on our data (as constrained by our prior distribution), what can we actually do with it? Well, the posterior tells us where the most probable value for θ is, given the data (and our prior). For example, with the uniform prior ($a = b = 1$) and $N = 25$ and $z = 8$, we can conclude that the most probable value for θ is the mean of the posterior. We can compute this by playing around with distributions:

```
set.seed(1) # So you and I get the same results
mean(rbeta(10000,8+1,25-8+1)) # Estimated using resampling
```

```
[1] 0.3336166
```

But our situation is so simple that there's also a fixed equation to compute it (Krutsche, 2015, p. 133):

Posterior for Bernoulli trials: $P(\theta|D) = (z+a)/(N+a+b)$

```
(8+1)/(25+1+1)
```

```
[1] 0.3333333
```

Still, we don't want to get stuck on point estimates again. The posterior is actually a distribution, and its spread means something too. In particular, a wider posterior also means less confidence in *θ*, and a narrower one means more confidence. This is the insight that gives rise to the Bayesian version of confidence intervals, which we'll discuss in the next section.

The upshot of all this is that if you and your worst enemy can agree on a prior, there's only one "best" answer to our question about the 8 acceptances from the 25 speakers. Of course, the prior is "subjective", but unlike the "mind-reading" we have to do in the frequentist analysis, the Bayesian priors are public, and anybody can calculate what happens if they try a different one.

Another nice consequence of the Bayesian approach is that unlike the case in traditional statistics, it's perfectly acceptable to stop our experiment part-way to see how it's going, and then decide to give up or continue based on the progress. Doing so doesn't change the final *N* or *z*, so the final result will be the same (assuming some particular prior). I mentioned this property earlier when I introduced the naive Bayes classifier, which works equally well if you input the features one at a time, or all at once.

To illustrate this with our syntactic judgment experiment, say we collect the same *N* and *z* responses, but in two separate phases (try it!):

```
data1 = c(rep(1,6),rep(0,10),1) # 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1: our first try
length(data1); sum(data1) # N = 17, z = 7
data2 = c(rep(0,7),1) # 0 0 0 0 0 0 0 1: we decide to get more data
length(data2); sum(data2) # N = 8, z = 1
length(c(data1,data2)); sum(c(data1,data2)) # total N = 25, total z = 8
```

Then we analyze the first data set (data1) and get a posterior distribution. To incorporate the second data set (data2) into the analysis, we simply turn this posterior distribution into our new prior. The final result ends up being exactly the same as when we did it in one step, as you can see if you compare last two bolded plots in Figure 5:

```
likely.7.17 = function(theta) {theta^7*(1-theta)^(17-7)} # data1 likelihood
likely.1.8 = function(theta) {theta^1*(1-theta)^(8-1)} # data2 likelihood
par(mfrow=c(2,4), mai=rep(0.5,4)) # Plots for data1 (top) & data2 (bottom)
curve(dbeta(x,1,1),0,1,main="data1: Uniform prior")
curve(likely.7.17(x),0,1,main="data1: Likelihood")
curve(dbeta(x,7+1,17-7+1),0,1,main="data1: Posterior")
frame() # Empty plot (so rows line up)
curve(dbeta(x,7+1,17-7+1),0,1,main="data2: Prior = data 1 posterior")
curve(likely.1.8(x),0,1,main="data2: Likelihood")
curve(dbeta(x,1+(7+1),8-1+(17-7+1)),0,1,lwd=2,main="data2: Posterior")
curve(dbeta(x,8+1,25-8+1),0,1,lwd=2,main="All: Posterior (= Figure 3, bottom left)")
# 1+(7+1) = 8+1 = 9,    8-1+(17-7+1) = 25-8+1 = 18
```

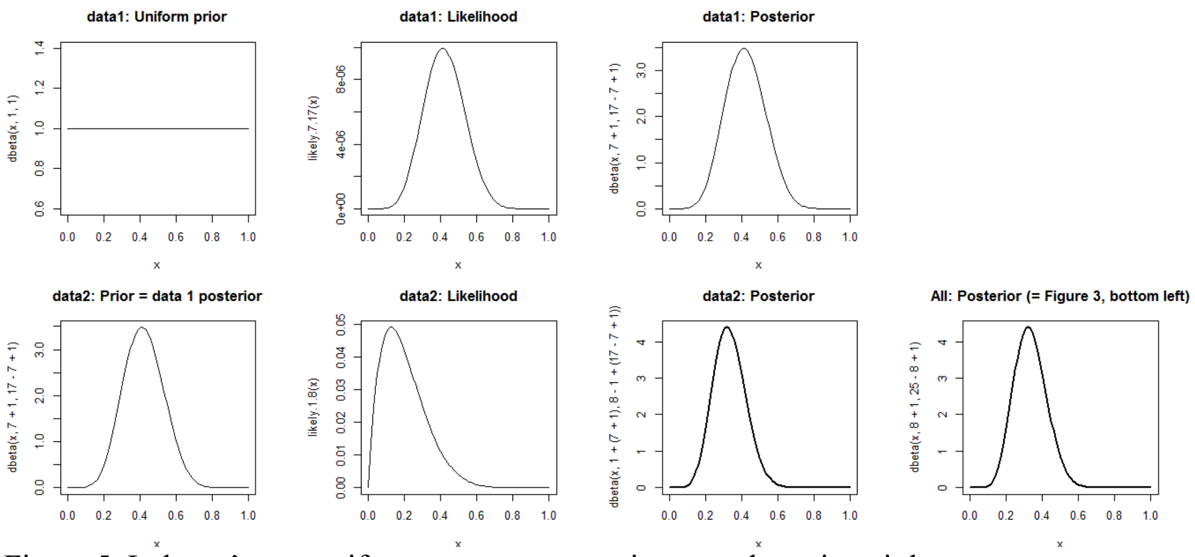

Figure 5. It doesn't matter if you stop your experiment and continue it later

This is dramatically different from what I've emphasized about using traditional statistics. If you run part of an experiment, aren't happy with the *p* values you're getting, and then use that information to collect more data until the *p* value goes below .05, then this doesn't count as truly statistically significant. This rule doesn't seem like it gives a realistic picture of real science, however. After all, if your worst enemy runs an experiment that finds no significant result for your favorite hypothesis, then isn't this going to inspire you to run a bigger experiment to see if you can increase the power and get a significant result? More generally, traditional statistics doesn't capture the valid reasoning behind scientific data gathering: of course scientists gather data to see if this changes their mind about hypotheses, and of course they will gather some data, study (and maybe publish) it, and then go collect some more. Despite its intimidating math, then, Bayesian statistics allows you to analyze your research results in a way that fits much better with how you actually collect and interpret them.

3.2 Bayesian confidence intervals

Remember how I mentioned in the *t* test chapter that the Bayesian statistician Lee (2004) complained that traditional confidence intervals don't mean what you think they mean? By contrast, a 95% **Bayesian confidence interval**, also called a **highest density region** (**HDI**) or **credible interval**, really does indicate that the hypothesized parameter (θ) has a 95% probability of lying within it.

The 95% HDI is based on the posterior distribution, but this won't necessarily be symmetrical (as it isn't if we compute is using the beta distribution, as with our Bernoulli experiments). We still want the two tails of the distribution to collectively contain exactly 5% of the distribution area. Moreover, we want the "body" of the HDI to be as close as possible to the hypothesized parameter *θ*, or else we might end up with a ridiculous situation where one tail has 5% of the area, so that there's no other tail at all.

Because the posterior distribution can have any beta-style shape, and this distribution family can have all sorts of shapes, we need to search for the HDI by using an iterative loop, gradually shifting our guesses until it fits the above criteria (95% of the area, but not all stuck on one end if we can help it).

As an example, let's compute the HDI for our Bernoulli experiment, where $N = 25$ and $z = 8$, analyzed with the uniform prior ($a = b = 1$). We search iteratively for the two 5% points closest to *θ* by slowly moving up the point for the left tail (making sure the right tail has the rest of the 5% area), gradually shrinking the distance to the right point, until moving the left point up any more will make the right point move away instead of moving closer. In the code below, the looping is done using **while()**, which, unlike **for()**, keeps looping as long as the expression inside the parentheses remains true, so we don't have to specify the total number of loops ahead of time (we don't even know many we need).

```
betaHDI.try = function(N=25, z=8, a=1, b=1, confidence=0.95) {
  tail.L = 0 # Initialize left tail theta of the HDI
  old.width = 1 # Initial width for HDI at maximum (we want to find minimum)
  not.done = T # A logical variable to tell us that we haven't yet found the HDI
  while(not.done) { # Keep looping until we're done
    tail.L = tail.L + 0.000001 # Increase size of left tail point a tiny bit
    p.tail.L = pbeta(tail.L,z+a,N-z+b) # Area in left tail now
    p.tail.R = (1-confidence) - p.tail.L # Area in right tail now (both add up to 5%)
    tail.R = qbeta(p.tail.R,z+a,N-z+b,lower.tail=F) # Right tail point now
    new.width = tail.R - tail.L # Distance between two tails now
    if (old.width < new.width) { # Moving tail.L more makes width bigger
      not.done = F # We're done! We can jump out of the while-loop!
    } else { # Too bad, we're not done yet....
      old.width = new.width # Reset old width and try again
    } # End of if-else
  } # End of while-loop
  return(c(tail.L,tail.R)) # This is the HDI
} # End of betaHDI function
```

Running the above function with the default arguments gives us the lower and upper endpoints of the HDI (though it runs slowly and, as we'll see, doesn't give ideal results):

```
betaHDI.try()
```

```
[1] 0.164620 0.508787
```

Of course this job can be done in a faster and more accurate way; Kruschke (2015, pp. 138-9) himself uses a function called **BernBeta.R** (Bernoulli beta), which uses another function called **HDIofICDF.R** (inverse cumulative density function). The **BernBeta.R** function makes use of R's **optimize()** function, which does the same thing my **while**-loop does, but faster and better. The **BernBeta.R** and **HDIofICDF.R** functions are available online at the following links (for the first edition of Kruschke, 2011), for you to download and copy/paste into R, or to load directly from the Web using the **source()** function (just copy each URL, put it inside "", inside **source()**):

https://jkkweb.sitehost.iu.edu/DoingBayesianDataAnalysis/Programs/HDIofICDF.R
https://jkkweb.sitehost.iu.edu/DoingBayesianDataAnalysis/Programs/BernBeta.R

Here's a simplified function that combines his two (without the plots and other fancy things that his code does):

```
betaHDI = function(N=25, z=8, a=1, b=1, confidence=0.95) {
  p.tails = 1 -confidence
  find.width = function(p.L, conf) { # This is the function to optimize
    L = qbeta(p.L, z+a, N-z+b) # I'm using simpler argument names here...
    R = qbeta(p.L + conf, z+a, N-z+b) # ... to avoid confusion with main function
    return(R - L) # We want to minimize this by varying p.L
  }
  p.tail.L = optimize(find.width, # Name of function to optimize
    conf = confidence, # The other argument for the find.width function (fixed)
    interval = c(0,p.tails), # The range of p.L to search for the minimum R-L
    tol=1e-8 # A very strict "tolerance", for accuracy
  )$minimum # one of the two outputs of optimize function
  tail.L = qbeta(p.tail.L, z+a, N-z+b)
  tail.R = qbeta(p.tail.L + confidence, z+a, N-z+b)
  return(c(tail.L, tail.R))
}
betaHDI()
```

```
[1] 0.1646193 0.5087863
```

The results are quite close to what I got with my totally home-made function, but we get them much faster.

One thing we can do with the HDI is make decisions about hypotheses, similar to traditional *p*-value logic (but better, since *p* values are kind of evil, as we've seen). For example, if we want to know if getting 8 "yes" responses in 25 tries should count as "statistically different" from equal probabilities for "yes" vs. "no" ($\theta = .5$), we can look at the HDI we got: .165 to .509. Given our uniform prior and our data set, we should be 95% confident that the actual θ is between those values. And look, .5 is inside the HDI! Thus in this case, we can accept (not merely not reject) the "null" hypothesis that $\theta = .5$.

More realistically, our "null hypothesis" won't be just a single point value like $\theta = .5$, but will itself be a range, which Kutsche (2015) calls a **region of practical equivalence** (**ROPE**). So if we say that any value for θ from .45 to .55 would count as "the same", then again, in our case, the ROPE overlaps with the HDI, so we are justified in accepting the null hypothesis as consistent with the data (and prior).

Of course, skeptics may criticize our uniform prior, and this will affect what they think of our 95% HDI. If we instead use the highly informative, narrow bell-shaped prior from earlier ($a = b = 100$), our 95% HDI becomes much narrower, even though it still contains $\theta = .5$:

```
betaHDI(a=100, b=100) # 0.4149765 0.5451228: theta=0.5 firmly inside HDI
```

```
[1] 0.4149765 0.5451228
```

3.3 Bayesian hierarchical modeling

Since this is the very last section of the very last chapter, we may as well go out with a bang! So let's combine this weird Bayesian stuff with the other high-tech thing you learned recently, namely mixed-effects modeling (also known as multilevel modeling). In the Bayesian world, mixed-effects modeling is usually called **hierarchical modeling**, for two reasons. First, Bayesians don't strictly distinguish "fixed" from "random", since everything is a kind of probability in Bayes's rule. Second, even though hierarchical modeling seems like a crazy new idea in frequentist statistics, it turns out to be quite fundamental in Bayesian statistics, where lots of things are analyzed in terms of hierarchies. Indeed, the computational algorithms that made Bayesian statistics practical are closely related to the algorithms that made frequentist mixed-effects modeling possible.

The key insight is that a Bayesian hierarchical model simply implies a certain kind of conditional probability. Namely, just as the likelihood shows the data as conditional on the hypothesized parameter, the parameter itself is conditional on higher-level parameters.

For example, here's the simple version of Bayes's rule again, with D for data (e.g., the observed z and N in our binary analysis):

Bayes's rule (simplest version again): $$P(\theta|D) = \frac{P(D|\theta)\cdot P(\theta)}{P(D)}$$

As Lynch (2007) points out, you can think of Bayes's rule as already showing a kind of simple hierarchy: the posterior P($\theta|D$) is really just the prior P(θ), weighted by the likelihood and evidence P($D|\theta$)/P(D). The likelihood relates to the observed data, but to get the posterior we multiply it by the prior, which is not directly observed, and so the prior represents a higher, more abstract level.

The prior P(θ) itself need not be the highest level of abstraction. For example, maybe instead of asking for judgments about one sentence from a series of people, we instead ask each person to give judgments about a fixed set of sentences. Then the observations are no longer i.i.d., but instead come in grouping units. We're still interested in the ultimate parameter θ for the whole data set, but in between this highest level and our observations there is another layer of θ_i parameters, one for each grouping unit (person). θ is now the **hyperparameter** of the parameters θ_i, and P(θ) is the **hyperprior** for the priors P($\theta_i|\theta$). This implies the following extended version of Bayes's rule:

Bayes's rule (hyperparameter version): $$P(\theta_i,\theta|D)=\frac{P(D|\theta_i,\theta)\cdot P(\theta_i|\theta)\cdot P(\theta)}{P(D)}$$

If we want to avoid the "language-as-fixed-effects" fallacy, then we also have to add another layer of parameters θ_j for the sentences:

Bayes's rule (subjects × items version): $$P\left(\theta_i,\theta_j,\theta\middle|D\right)=\frac{P\left(D\middle|\theta_i,\theta_j,\theta\right)\cdot P(\theta_j|\theta_i,\theta)\cdot P(\theta_i|\theta)\cdot P(\theta)}{P(D)}$$

One clever use of the notion of hyperpriors is to build learning models that test hypotheses about Universal Grammar (UG), as is done by Perfors et al. (2010). As explained earlier, L is the language data received by the child and G is the language-specific grammar discovered by the child, but now we also have the hyperprior P(U) for some specific hypothesis that linguists may want to test about Universal Grammar:

UG as a Bayesian hyperprior: $$\mathrm{P(G,U|L)}=\frac{\mathrm{P(L|G,U)\cdot P(G|U)\cdot P(U)}}{\mathrm{P(L)}}$$

Unfortunately, it's hard to make Bayesian hierarchical models concrete. The problem comes from the definition of conditional probability: P(A|B) = P(A,B)/P(B). This means that a conditional probability like P($\theta_j|\theta_i,\theta$) is equal to P(θ_j,θ_i,θ)/P(θ_i,θ), or a ratio of two **joint** probabilities. These joint probabilities have two or three parameters, so they actually represent two- or three-dimensional distributions, not just the one-dimensional kind that we've been plotting all through this book (i.e., one dimension along the x-axis).

One consequence is trivial: we can no longer make our familiar plots for the prior, likelihood, or posterior. A more serious consequence is that joint probabilities don't conform to simple probability distribution families like the beta family. So we're forced to use those estimation procedures, like Gibbs sampling or Monte Carlo methods, mentioned earlier in the chapter.

Nevertheless, even using basic R, it's possible to try out the idea. The following example comes is based on one in Lynch (2007), and in addition to showing how hierarchical modeling works in R, it also shows two other things: how a Bayesian analysis works for continuous variables (not just the binary responses we've looked at so far in this book), and how Gibbs sampling works.

My linguistic version of Lynch's data is in **tests.txt**, which give the test scores for twenty Martian students in two different classes (actually heavily modified from a tiny bit of an example involving two years' salary; see Lynch, 2007, p. 241, fn. 1):

```
tdat = read.delim("tests.txt")
head(tdat[order(tdat$Student),]) # Reordered to show the grouping structure
```

```
   Student  Class  Score
1        1      1     56
21       1      2     70
2        2      1     54
22       2      2     83
3        3      1     61
23       3      2     88
```

Our somewhat odd goal is to model the overall mean Score values, even though each Student gives us two of them. So even though the data are paired, we're not doing a paired *t* test, but we're just practicing how we can separate out the influence of Student on Score. In other words, to express the goal in frequentist terms, rather than trying to predict Score from any "fixed" variable, we're actually modeling just the "random" effect. Anything more complex than this would require going beyond basic R (which is what Lynch, 2007, does, turning to WinBUGS, an older version of OpenBUGS).

The actual mean and standard deviation for Score are easy to compute, of course:

```
mean(tdat$Score) # 79.175
sd(tdat$Score) # 13.26048
```

Using the Central Limit Theorem, we can also compute a kind of idealized standard error for the raw data:

```
sd(tdat$Score)/sqrt(40) # 2.096666
```

It's also easy to compute the mean Score values for each Student (here shown rounded):

```
subjmeans = tapply(tdat$Score,tdat$Student,mean) # Actual means
round(subjmeans,0)
```

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
63 68 74 97 78 66 90 70 58 90 82 78 86 80 84 89 70 98 94 68
```

Now we can compute the variance for these Student means:

var(subjmeans) # 140.7441

Now for the modeling. To get you oriented with familiar stuff, let's first run an intercept-only model using traditional linear regression, ignoring the grouping by Student:

tdat.lm = lm(Score ~ 1, data = tdat)
summary(tdat.lm) # Just the coefficient table is shown below

```
Coefficients:
                  Estimate   Std. Error   t value   Pr(>|t|)
      (Intercept) 79.175     2.097        37.76     <2e-16     ***
Residual standard error: 13.26 on 39 degrees of freedom
```

As you might remember, the estimate for the intercept is just the overall mean of Score, and the standard error for the intercept is derived from the formula for one-sample *t*-tests (based on the Central Limit Theorem):

mean(tdat$Score) # 79.175
sd(tdat$Score)/sqrt(nrow(tdat)) # 2.096666 (remember how to calculate this?)

Now let's do LME, again using an intercept-only model, but grouping by Student. Again, the fixed intercept estimate is just the overall mean of Score, but the standard error is different now, since we're also taking the grouping by Student into account:

library(lme4)
tdat.lme = lmer(Score ~ 1 + (1|Student), data = tdat)
summary(tdat.lme) # Below only shows the tables for random and fixed effects

```
Random effects:
     Groups     Name          Variance    Std.Dev.
     Student    (Intercept)   103.01      10.149
     Residual                 75.47       8.688

Fixed effects:
                  Estimate   Std. Error   t value
      (Intercept) 79.175     2.653        29.85
```

The LME results also show estimates for the random variance for Student and the residuals. Notice that the LME residual standard deviation (8.688) is smaller than the residual standard error for the ordinary linear model (13.26, implying a variance s^2 of 175.8276), because the LME takes the grouping into account; the random standard deviation for Student

(10.149) is bigger than the residual standard deviation, showing that most of the variation in the data come from differences across students, not within students.

The LME model also makes predictions about the mean scores for each student, which we can compare with the actual means:

```
tdat.lme.pred = predict(tdat.lme)
tdat.lme.pred.student = (tdat.lme.pred[1:20]+ tdat.lme.pred[21:40])/2
subjmeans.lme = tdat.lme.pred.student
round(subjmeans.lme,0)
```

```
 1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  19  20
67  71  76  92  79  69  87  72  63  87  81  78  84  80  83  86  72  93  90  71
```

We can see how well (or poorly) this model captures Subject by making the scatter plot in Figure 6. The dots fall on a straight line because LME models the random variables in a linear way as well. The slope doesn't match the data exactly because this slope is what the LME algorithms was able to estimate from the data it was given.

```
plot(subjmeans, subjmeans.lme, xlim = c(50,100), ylim=c(50,100))
segments(x0=50,y0=50,x1=100,y1=100) # What a perfect fit would look like
abline(lm(subjmeans.lme ~ subjmeans), lty = 2) # Actual fit
legend("topleft",legend=c("Ideal fit","Actual fit"),lty =c(1,2))
```

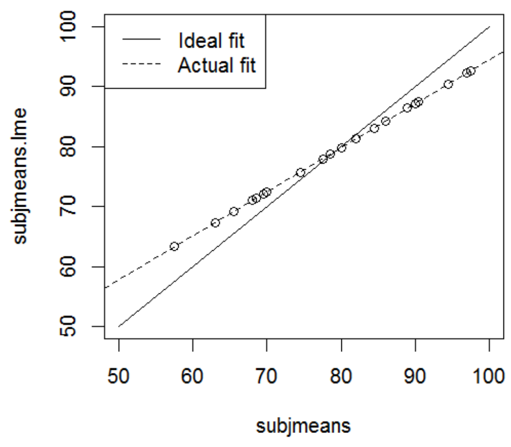

Figure 6. How LME models the random effect

Now let's do it in a Bayesian way, and see how the results compare with LME. The following R code is based on that given in Lynch (2007, pp. 245-6). I'll first show the code and the values that it outputs, and then explain how it works. Note that the code uses randomization, so every time you run it, you get somewhat different results, which is why I set a random seed, so you can follow along on your own computer (and lots of loops to let it converge well).

```
set.seed(2) # So you get the same results that I do
# Rearrange the scores into a matrix with two columns
y=as.matrix(cbind(tdat$Score[tdat$Class==1],tdat$Score[tdat$Class==2]))
# "Safe" initial values for the priors (explained below):
m=0; s2=100000; a=c=.00001; b=d=.00001; tau2=1; sigma2=1; alpha=0
n=nrow(y) # Number of grouping units (Students)
# Iterate through Gibbs sampling algorithm
for(B in 1:1000000) {
 alpha_i = rnorm(n, # Distribution for estimated by-Student means
   mean = (((tau2*(y[,1]+y[,2]))+sigma2*alpha)/(2*tau2+sigma2)),
   sd = sqrt((tau2*sigma2)/(2*tau2+sigma2)))
 alpha=rnorm(1, # Distribution for estimated overall mean
   mean=(tau2*m+s2*sum(alpha_i))/((tau2+n*s2)),
   sd=sqrt((tau2*s2)/(tau2+n*s2)))
 tau2=1/rgamma(1, # Distribution for estimated random Student variance
   shape=(n/2+a),
   rate=(sum((alpha_i-alpha)^2)+2*b)/2)
 sigma2=1/rgamma(1, # Distribution for estimated residual variance
   shape=n+c,
   rate=(sum((y-alpha_i)^2) +2*d)/2)
}
```

The estimated overall mean and its standard deviation are given, respectively, by the variable called **alpha** and a square root ratio computed using variables called **tau2** and **s2**:

```
alpha # 77.79978
sqrt((tau2*s2)/(tau2+n*s2)) # 2.647624
```

The Student and residual variance are represented by **tau2** and **sigma2**, respectively:

```
tau2 # 140.2081
sigma2 # 74.23824
```

In case you're having trouble keeping track of all these numbers, Table 3 compares the values (where available) for the raw data with the values (where available) estimated by the linear modeling ignoring grouping by Student, the LME with Student as grouping variable, and the hierarchical Bayesian model.

It's hard to say which model does the best. The two frequentist models capture the mean exactly, since that mean is built into their calculations; given that the Bayesian model had to figure it out using Gibbs sampling, it doesn't seem too far off. All three models give roughly similar values for the overall *SE*. LME and the Bayesian model give somewhat similar estimates for the student variance; both differ from the observed values, but it's not clear what the best way to compute a raw value for this should be anyway. They both also have lower variance for the residuals as compared with the simple linear model, showing that grouping by Student did improve model fit.

Table 3. Real values and their estimates by various types of models

	Overall mean	Overall SE	Student variance	Residual variance
Raw data	79.175	2.10	140.74	NA
Linear model	79.175	2.097	NA	175.83
LME	79.175	2.653	103.01	75.47
Bayesian	77.800	2.648	140.21	74.24

But how well did the Bayesian model do at estimating the individual Student means? These values are in the vector **alpha_i**:

```
subjmeans.bayes = alpha_i
names(subjmeans.bayes) = 1:20 # To compare with the real and LME-estimated means
round(subjmeans.bayes,0)
```

```
                  1 1 1 1 1 1 1 1 1 1 2
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
7 7 7 9 8 7 9 7 6 8 7 8 7 7 8 8 8 9 9 8
6 1 6 7 6 1 2 4 1 7 8 4 9 8 7 1 0 5 6 2
```

Let's see how these estimates compare with the real means, as shown in Figure 7:

```
plot(subjmeans, subjmeans.bayes, xlim = c(50,100), ylim=c(50,100))
segments(x0=50,y0=50,x1=100,y1=100) # What a perfect fit would look like
abline(lm(subjmeans.bayes ~ subjmeans), lty = 2) # Actual fit
legend("topleft",legend=c("Ideal fit","Actual fit"),lty =c(1,2))
```

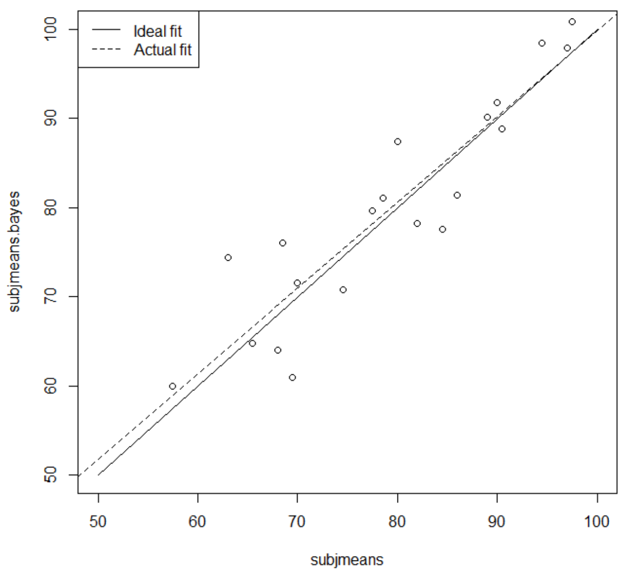

Figure 7. How Bayesian statistics models the "random" effect (to use a frequentist term)

The scatter of dots is clearly much broader than the neat line given by LME, but interestingly, the Bayesian analysis seems to be picking up on a similar trend in the raw data, tilting the Student means at a similar angle as the LME model, relative to the actual values.

But how does this strange R code work? Well, if you must know, the spelled out Greek letters (alpha, tau, and so on) represent parts of the following hierarchical model (using the formal notation of Lynch, 2007, p. 242). Our goal is to learn about the overall mean α, cross-Student variance τ^2, individual Student means α_i, and error variance σ^2 from the data Y. Note that the overall mean is represented with α, not our familiar population mean symbol μ, because this represents the hypothesized parameter that we actually want to learn about, not the mean of the boring old null hypothesis of traditional statistics. Also note how for each conditional probability, each of the parameters on the right of the "|" is itself conditioned by something at a higher level, until we reach the hyperparamers m, a, b, c, d for the hyperpriors, which aren't conditional on anything.

$$P(\alpha,\tau^2,\alpha_i,\sigma^2|Y) = \frac{P(Y|\alpha_i,\sigma^2)\cdot P(\alpha_i|\alpha,\tau^2)\cdot P(\alpha|m,\sigma^2)\cdot P(\tau^2|c,d)\cdot P(\sigma^2|a,b)}{P(Y)}$$

To turn the above Bayes's rule into the actual R code, Lynch (2007, pp. 242-245) has to go through a long series of algebraic and even calculus-based derivations to simplify it enough to actually run efficiently, but I'll spare you the details.

In computing the probabilities, we also assume that that the observed scores y for Student i in class c are normally distributed with mean α_i and variance σ^2, Student means α_i are normally distributed with overall mean α and overall Student variance τ^2, possible overall means α are normally distributed with “hypermean” *m* and variance s^2, possible overall Student variances τ^2 follow the **inverse gamma distribution** with parameters *a* and *b*, and possible error variances σ^2 follow the inverse gamma distribution with parameters *c* and *d*.

What is the inverse gamma distribution? It’s yet another special distribution used more commonly in Bayesian statistics than in traditional statistics, since it can model the unknown variance of a normal distribution (as it does here): it’s always positive (like variance) and positively skewed (favoring lower variances), and similar to the beta distribution, an inverse gamma prior will often (though unfortunately not always) generate a posterior from the same family, making the math easier to deal with.

This may sound exotic, but it’s related to things we’ve seen before. Namely, the inverse gamma distribution is the inverse of the **gamma distribution** (so if Γ is the gamma distribution, $1/\Gamma$ is the inverse gamma distribution). The gamma distribution is related to two familiar things: the chi-squared distribution (also positive and rightward skewed) and the **factorial** (階乘) function, generalized to continuous values. Naturally, R has a family of functions for it, including **rgamma()**, used in the above code, which takes the parameters that R calls **shape** and **rate**. So to select a random sample from the inverse gamma distribution in R, the code uses **1/rgamma()**.

Turning back to the above R code, note the random sampling (**rnorm()** and **1/rgamma()**) and the looping. These are two of the key features of Gibbs sampling, the algorithm that finally made Bayesian statistics feasible in the 1980s. If you look carefully at the code, you’ll see that each variable (**alpha_i**, **alpha**, **tau2**, **sigma2**) is defined in terms of all of the other variables, plus a little bit of noise due to the random sampling. Due to the looping, this causes each variable to be adjusted slightly each time the loop is repeated, gradually converging on the set of values that you get at the end.

Thus Gibbs sampling makes it possible to compute a posterior distribution for which there is no simple formula (unlike the simple equations for the binary response data we played with earlier). There are cases where even Gibbs sampling doesn’t work (e.g., where there is no known distribution to use), but for that, there is a more flexible method called the **Metropolis(-Hastings) algorithm**. Even though Metropolis sounds like it refers to a city, it’s actually named after the American physicist Nicholas Constantine Metropolis (1915-1999) (his family background was Greek). The Canadian statistician W. K. Hastings (1930-2016) got his name added because he came up with a generalization in the 1970s.

This randomization, however, means that we may get different results each time, especially if we don’t loop enough. Moreover, as is usual with Bayesian statistics, we also have to choose the priors carefully. At the start of the algorithm, the prior mean **alpha** and hyperprior mean **m** are both set to zero (like the null hypothesis). The hyperprior variance **s2** is set to a huge number, making the hyperprior distribution essentially a uniform distribution. The error variance **sigma2** and Student variance **tau2** are both set to 1 (which imply standard normal distributions). The hyperprior parameters **a**, **b**, **c**, **d** for the inverse gamma distributions of **tau2** and **sigma2** are made so tiny that they initially have no influence. (I actually had to make these priors even more extreme than Lynch had them, since my sample is much smaller than his.)

But what was the point of all this Bayesian work, if the end result is a bunch of values that essentially match what we can get with frequentist LME?

One reason is the same as we've seen throughout this chapter: despite the complex math, the ultimate product is more intuitive. For example, we can say that alpha represents our best guess about the population mean, and we can compute a 95% HDI around it, instead of being forced to say that we merely rejected the null hypothesis that the population mean is zero, and using the backwards logic of traditional 95% confidence intervals. Another benefit is reliability: if our sample is too small to allow the Central Limit Theorem to apply, Laplace's reasoning doesn't work and the Bayesian analysis is actually a better model of the data, as indeed we saw (compare Bayesan Figure 7 with LME Figure 6). Finally, because hierarchical modeling is directly implied by Bayes's rule, it seems that Bayesian modeling is capable of handling more complex situations than frequentist tools like LME (see Gelman & Hill, 2007).

Since this is the last section of the last chapter in the book, and I usually only get around to revising this part of the book towards the end of the semester when all those urgent end-of-the-semester things arrive to eat up my time, I am sadly unable to explain something really important: all of the stuff discussed in this section has become quite a bit easier with a package that talks to Stan in a more R-friendly way: **brms** (Bürkner, 2017), which stands for "Bayesian Regression Models using 'Stan'". Vasishth et al. (2018) give a detailed tutorial for analyzing phonetic data, which I hope to go through (along with other tutorials) so I can add my own even briefer, and hopefully even easier (while still accurate) tutorial to this chapter. But based on my current understanding, the main thing is that the package lets us stick with our old familiar R formula syntax, instead of having to use Stan directly. For example, to run a Bayesian mixed-effects regression, we might write something like this:

```
myBayes.brm = brm(RT ~ LogFreq + (LogFreq|Subjects), data = myData,
   prior = myPriors)
```

The new bit is the **prior** argument, which is a vector that describes what we assume by default about our model, for example:

```
myPriors = c(set_prior("normal(0, 200)", class = "Intercept"),
   set_prior("normal(0, 50)", class = "b", coef = "LogFreq"), ...)
```

Inside the **set_prior()** function, the first argument describes what are priors are shaped like. For example, the string **"normal(0, 200)"** describes a normal distribution with sd = 200, though 0 isn't the mean as we might expect, but rather the point where the distribution is "truncated", that is, the distribution starts at zero, since we expect the intercept for RT never to go below zero. The second argument, **class**, links each prior to different parts of the model (e.g., the intercept as **"Intercept"** and the ordinary predictor variable coefficient as **"b"**). Finally, the "**...**" represents all the other stuff that I don't have time to explain, namely prior assumptions about the variability in the model associated with the fixed and random variables (similar to **sigma2** in the model we did "manually" above).

So... while **brms** certainly seems to make Bayesian modeling easier, it still takes a bit of time to learn how to use it (as Vasishth et al., 2018, themselves admit). Stay tuned for future editions of this chapter!

4. Conclusions

Bayesian statistics has moved from an arcane mathematical game to a taboo in statistics to an increasingly popular approach to statistics that, in its extreme form, wants to get rid of most of traditional statistics, based on a frequency-based approach to probability, which is what you've been studying in the rest of this book. The problems with traditional statistics have been widely noted, even by traditional statisticians, including confusions over the meaning of *p* values, counterintuitive concepts like confidence intervals, and the hidden influences of subjectivity. Bayesian statistics aims to replace all this with a more intuitive notion of probability in which one modifies ones prior confidence in a hypothesis (expressed as a probability distribution) by taking into account the data, the likelihood that the data could result from the hypothesis, and the overall probability of the observed data themselves, as expressed in Bayes's rule. In its simplest forms, Bayes's rule allows us to use evidence to argue in favor of the null hypothesis, a common situation in linguistics and to classify things. In more complex forms, Bayes's rule allows us to compute probabilities for binary response variables in a way that makes our subjective assumptions explicit (as the priors), rather than hidden (as in the confusing contrast between the binomial and negative binomial tests), to compute a more intuitive form of a confidence interval and to accept or reject hypotheses, and even to compute models as sophisticated as mixed-effects (hierarchical) models. Unfortunately, Bayesian statistics has yet to receive a truly introductory treatment, aimed at total novices, but I hope this chapter at least makes some of its key ideas and methods understandable and practical for your own research... and hopefully future revisions will make this chapter even more useful!

References

Bürkner, P.-C. (2017). brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software, 80*(1), 1-28.

Carrier, R. C. (2012). *Proving history: Bayes's theorem and the quest for the historical Jesus*.

Prometheus Books.

Cohen, J. (1994). The Earth is round ($p < .05$). *American Psychologist, 49*(12), 997-1003.

Dienes, Z. (2008). *Understanding psychology as a science: An introduction to scientific and statistical inference*. Palgrave MacMillan.

Gelman, A., & Hill, J. (2007). *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press.

Gelman, A., & Loken, E. (2013). The garden of forking paths: Why multiple comparisons can be a problem, even when there is no "fishing expedition" or "p-hacking" and the research hypothesis was posited ahead of time. Columbia University and Penn State University ms. http://www.stat.columbia.edu/~gelman/research/unpublished/p_hacking.pdf

Gill, J. (2014). *Bayesian methods: A social and behavioral sciences approach* (third edition). New York: CRC Press.

Hammond, M. (2016). Predicting the gender of Welsh nouns. *Corpus Linguistics and Linguistic Theory, 12*(2), 221-261.

Howson, C., & Urbach, P. (2006). *Scientific reasoning: The Bayesian approach* (third edition). Chicago: Open Court.

Jaynes, E. T. (2003). *Probability theory: The logic of science*. Cambridge University Press.

Jeffreys, H. (1961). *Theory of probability* (third edition). Oxford: Oxford University Press, Clarendon Press.

Johnson, T. R., & Kuhn, K. M. (2013). Bayesian Thurstonian models for ranking data using JAGS. *Behavior Research Methods, 45* (3), 857-872.

Johnson, V. E. (2005). Bayes factors based on test statistics. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67*(5), 689-701.

Jurafsky, D., & Martin, J. H. (2009). *Speech and language processing* (second edition). Upper Saddle River, NJ: Pearson.

Kruschke, J. K. (2011). *Doing Bayesian data analysis: A tutorial with R and BUGS*. Academic Press.

Kruschke, J. K. (2015). *Doing Bayesian data analysis, Second edition: A Tutorial with R, JAGS, and Stan*. Academic Press.

Lavine, M., & Schervish, M. (1999). Bayes factors: What they are and what they are not. *The American Statistician, 53*(2), 119-122.

Lee, P. M. (2004). *Bayesian statistics: An introduction* (third edition). Arnold.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In Machine learning: ECML-98 (pp. 4-15). Springer Berlin Heidelberg.

Lunn, D., Spiegelhalter, D., Thomas, A., & Best, N. (2009). The BUGS project: Evolution, critique and future directions. *Statistics in Medicine, 28* (25), 3049-3067.

Lynch, S. M. (2007). *Introduction to applied Bayesian statistics and estimation for social scientists*. Springer.

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.

Morey, R. D., & Rouder, J. N. (2014). BayesFactor: Computation of Bayes factors for common designs. R package version 0.9.7.

Mosteller, F., & Wallace, D. L. (1963). Inference in an authorship problem. *Journal of the American Statistical Association, 58* (302), 275-309.

Oaksford, M., & Chater, N. (2009). Précis of *Bayesian rationality: The probabilistic approach to human reasoning*. *Behavioral and Brain Sciences, 32*, 69-120.

Operskalski, J. T., & Barbey, A. K. (2016). Risk literacy in medical decision-making. *Science, 352*(6284), 413-414.

Perfors, A., & Tenenbaum, J. B. (2010). Variability, negative evidence, and the acquisition of verb argument constructions. *Journal of Child Language, 37*, 607-642.

Perfors, A., Tenenbaum, J. B., & Regier, T. (2011). The learnability of abstract syntactic principles. *Cognition, 118*, 306-338.

Poldrack, R. A. (2006). Can cognitive processes be inferred from neuroimaging data? *Trends in Cognitive Sciences, 10*(2), 59-63.

Raftery, A. E. (1995). Bayesian model selection in social research. *Sociological Methodology, 25*, 111-163.

Rouder, J. N., Speckman, P. L., Sun, D., Morey, R. D., & Iverson, G. (2009). Bayesian *t* tests for accepting and rejecting the null hypothesis. *Psychonomic Bulletin & Review, 16*(2), 225-237.

Salsburg, D. (2001). *The lady tasting tea: How statistics revolutionized science in the twentieth century*. New York: W. H. Freeman & Company.

Stan Development Team. (2019). *Stan User's Guide*, Version 2.29. https://mc-stan.org/.

Vasishth, S., Nicenboim, B., Beckman, M. E., Li, F., & Kong, E. J. (2018). Bayesian data analysis in the phonetic sciences: A tutorial introduction. *Journal of Phonetics, 71*, 147-161.

Zellner, A., & Siow, A. (1980). Posterior odds ratios for selected regression hypotheses. In J. M. Bernardo, M. H. DeGroot, D. V. Lindley, & A. F. M. Smith (Eds.), *Bayesian statistics: Proceedings of the First International Meeting* (pp. 585-603). Valencia: University of Valencia Press.