Introduction to Bayesian Modeling using Stan

FGME2017, Tuebingen, Germany

Bruno Nicenboim and Shravan Vasishth

2017-09-16

Contents

1 Introduction to Bayesian modeling 4
- 1.1 Suggestions on how to read this section . . . 4
- 1.2 Introduction . . . 4
  - 1.2.1 Steps in Bayesian analysis . . . 4
  - 1.2.2 Comparison of Bayesian with frequentist methodology . . . 4
  - 1.2.3 Statistics is the inverse of probability . . . 6
- 1.3 Brief review of probability theory and random variables . . . 6
  - 1.3.1 Kolmogorov Axioms of Probability . . . 6
    - 1.3.1.1 Three important propositions . . . 7
  - 1.3.2 Conditional Probability . . . 8
  - 1.3.3 Independence of events . . . 9
  - 1.3.4 Bayes' rule . . . 10
  - 1.3.5 Random variables . . . 10
    - 1.3.5.1 Some basic results concerning random variables . . . 15
  - 1.3.6 What you can do with a pdf . . . 16
  - 1.3.7 Some basic facts about expectation and variance . . . 17
  - 1.3.8 Three common and important distributions . . . 19
    - 1.3.8.1 Binomial distribution . . . 19
    - 1.3.8.2 Uniform random variable . . . 19
    - 1.3.8.3 Normal random variable . . . 20
- 1.4 Important probability distributions . . . 23
  - 1.4.1 Multinomial coefficients and multinomial distributions . . . 23
  - 1.4.2 The Poisson distribution . . . 24
  - 1.4.3 Geometric distribution . . . 25
  - 1.4.4 Exponential random variables . . . 25
  - 1.4.5 Gamma distribution . . . 27
  - 1.4.6 Beta distribution . . . 29
  - 1.4.7 *t* distribution . . . 30
- 1.5 Jointly distributed random variables . . . 30
  - 1.5.1 Discrete case . . . 30
  - 1.5.2 Continuous case . . . 32
  - 1.5.3 Marginal probability distribution functions . . . 34
  - 1.5.4 Independent random variables . . . 34
  - 1.5.5 Sums of independent random variables . . . 35

1.5.6 Conditional distributions . . . 36
1.5.6.1 Discrete case . . . 36
1.5.6.2 Continuous case . . . 37
1.5.7 Joint and marginal expectation . . . 38
1.5.8 Covariance and correlation . . . 38
1.5.9 Conditional expectation . . . 39
1.5.10 Multivariate normal distributions . . . 40
1.6 Maximum likelihood estimation . . . 43
1.6.1 Discrete case . . . 43
1.6.2 Continuous case . . . 44
1.6.3 Finding maximum likelihood estimates for different distributions . . . 44
1.6.4 Visualizing likelihood and maximum log likelihood for normal . . . 47
1.6.5 Obtaining standard errors . . . 47
1.6.6 MLE using R . . . 53
1.6.6.1 One-parameter case . . . 53
1.6.6.2 Using optim in the one-parameter case . . . 56
1.7 Introduction to Bayesian data analysis . . . 57
1.7.1 Example 1: Binomial Likelihood, Beta prior, Beta posterior . . . 59
1.7.2 Example 2: Poisson Likelihood, Gamma prior, Gamma posterior . . . 62
1.7.2.1 Concrete example given data . . . 66
1.7.2.2 The posterior is a weighted mean of prior and likelihood . . . 67
1.8 Introduction to Gibbs sampling . . . 68
1.8.1 Monte Carlo integration . . . 68
1.8.2 Markov chain sampling . . . 69
1.8.3 Monte Carlo sampling . . . 70
1.8.4 The inversion method . . . 72
1.8.5 The rejection method . . . 72
1.8.6 Monte Carlo Markov Chain: The Gibbs sampling algorithm . . . 75
1.8.7 Gibbs sampling using rejection sampling . . . 77
1.8.8 More realistic example: Sampling from a bivariate density . . . 79
1.8.9 Sampling the parameters given the data . . . 83
1.8.9.1 Example of first approach . . . 84
1.8.9.2 Example of second approach . . . 84
1.8.10 Summary of the story so far . . . 87
1.9 Exercises . . . 88
1.10 Further readings . . . 89

2 Introduction to Stan probabilistic language 90
2.1 A Stan program . . . 90
2.2 A first example . . . 91
2.3 MCMC diagnostics: Convergence problems and Stan warnings . . . 93
2.4 A small experiment . . . 94
2.4.1 Preprocessing of the data . . . 94
2.4.2 Probability model . . . 95
2.4.3 Running the Stan model . . . 96

2.4.4 Summarizing the posterior . . . 98
2.4.5 Influence of priors and sensitivity analysis . . . 100
2.5 A slightly more interesting analysis of the small experiment . . . 103
2.5.1 Preprocessing the data . . . 103
2.5.2 Probability model . . . 103
2.5.3 Summarizing the posterior and inference . . . 105
2.5.4 Posterior predictive checks . . . 106
2.5.5 A better probability model using the log-normal distribution . . . 110
2.5.6 The slightly more realistic probability model . . . 111
2.5.7 Summarizing the posterior and inference . . . 115
2.5.8 Posterior predictive checks and distribution of summary statistics . . . 115
2.5.9 General workflow . . . 117
2.6 Key concepts . . . 118
2.7 Exercises . . . 118
2.8 Appendix - Troubleshooting problems of convergence . . . 120

3 Bayesian hierarchical models (also known as multilevel or mixed-effects models) 121
3.1 Advantages (and disadvantages) . . . 121
3.2 Applied example: Attachment preference . . . 121
3.2.1 Complete pooling or fixed effects model (M_{cp}) . . . 125
3.2.2 No pooling model (M_{np}) . . . 130
3.2.3 Varying intercept model (M_{vi}) . . . 137
3.2.4 (Uncorrelated) varying intercept varying slopes model (M_{uvivs}) . . . 141
3.2.5 Correlated varying intercept varying slopes model (M_{cvivs}) . . . 156
3.2.6 Recovery of the parameters . . . 167
3.3 Why should we take the trouble of fitting a Bayesian hierarchical model? . . . 172
3.4 Key concepts . . . 173
3.5 Exercises . . . 174

References 175

1 Introduction to Bayesian modeling

1.1 Suggestions on how to read this section

For a first pass, it may be best to start with sections 1.2, 1.7, and 1.8, and then return to sections 1.3 to 1.6 as needed. But some may prefer to read this section in the order provided.

1.2 Introduction

The basic idea we will explore in this course is Bayes' theorem, which, as Lunn et al 2012 put it, allows us to learn from experience and turn a prior distribution into a posterior distribution given data.

A central departure from frequentist methodology is that the unknown population parameter is expressed as a random variable. For example, we can talk about how sure we are that the population parameter has a particular value, and this uncertainty is subjective. In the frequentist case, the unknown population parameter is a point value; in the frequentist world, you cannot talk about how sure you are that a parameter has value θ.

Notational conventions: A binomial distribution, n trials each with probability θ of occurring, is written $Bi(\theta,n)$. Given a random variable with this distribution, we can write $R \mid \theta,n \sim Bi(\theta,n)$ or $p(r \mid \theta,n) = Bi(\theta,n)$, where r is the realization of R. We can drop the conditioning in $R \mid \theta,n$, so that we can write: given $R \sim Bi(\theta,n)$, what is $Pr(\theta_1 < \theta < \theta_2 \mid r,n)$.

1.2.1 Steps in Bayesian analysis

1. Given data, specify a likelihood function or sampling density.
2. Specify prior distribution for model parameters.
3. Derive posterior distribution for parameters given likelihood function and prior density.
4. Simulate parameters to get samples from posterior distribution of parameters.
5. Summarize parameter samples.

1.2.2 Comparison of Bayesian with frequentist methodology

1. In Bayesian data analysis (BDA), the unknown parameter is treated as if it's a random variable, but in reality we are just using a probability density function (pdf) to charac-

terize our uncertainty about parameter values. In frequentist methods, the unknown parameter is a point value.

2. Frequentists want (and get) $P(data \mid parameters)$, whereas Bayesians want (and get) $P(parameters \mid data)$.

   What this means is that frequentists will set up a null hypothesis and then compute the probability of the data given the null (the p-value). The Bayesian obtains the probability of hypothesis of interest (or the null hypothesis) given the data.

   Frequentists will be willing to be wrong about rejecting the null 5% of the time—under repeated sampling. A problem with that is that a specific data set is a unique thing; it's hard to interpret literally the idea of repeated sampling.

   In this context, Here's a telling criticism of the frequentist approach by Jeffreys:

   > "What the use of [the p-value] implies, therefore, is that a hypothesis that may be true may be rejected because it has not predicted observable results that have not occurred."

   Jeffreys' quote is taken from Lee's book.

   Another interesting quote is from the entsophy blog entry of 23 Sept 2013: "The question of how often a given situation would arise is utterly irrelevant to the question of how we should reason when it does arise." (Attributed to Jaynes.)

3. For frequentists, the probability of an event is defined in terms of (hypothetical) long run relative frequencies; so, one cannot talk about the probability of a single event. For Bayesians, a single event is associated with a probability and this probability is subjective: it depends on the observer's beliefs. An example is the familiar one: the probability of a heads in a coin toss. You believe that any coin you pull out of your pocket is fair; that's a subjective belief. From the frequentist viewpoint, the 0.5 probability of a heads is a consequence of long-term relative frequencies under repeated coin tosses.

One argument one could make in favor of using Bayesian tools always is that most people who use frequentist methods just don't understand them. Indeed, hardly anyone really understands what a p-value is (many people treat $P(data \mid parameter)$ as if it is identical to $P(parameter \mid data)$), or what a 95% confidence interval is (many people think a 95% CI tells you the range over which the parameter is likely to lie with probability 0.95), etc. People abuse frequentist tools (such as publishing null results with low power experiments, not checking model assumptions, chasing after p-values, etc., etc.), and part of the reason

for this abuse is that the concepts (e.g., confidence intervals) are very convoluted, or don't even address the research question. Regarding this last point, consider p-values. They tell you the probability of observing a statistic (t-value) or something more extreme assuming that a null hypothesis is true; p-values don't tell you anything about the actual research hypothesis.

1.2.3 Statistics is the inverse of probability

In probability, we are given a probability density or mass function f(x) (see below), and parameters, and we can deduce the probability.

In statistics, we are given a collection of events, and we want to discover the parameters that produced them (assuming f(x) is the pdf that generated the data). The classical approach is:

1. Estimate parameters assuming $X \sim f(x)$.
2. Do inference.

For example, for reading times, we assume a random variable X_i that comes from a normal distribution $N(0, \sigma^2)$ (the null hypothesis). We compute the most likely parameter value that generated the data, the mean of the random variable,[1] and compute the probability of observing a value like the mean or something more extreme given the null hypothesis. I write this statement in short-hand as $P(data \mid parameter)$.

1.3 Brief review of probability theory and random variables

We begin with a review of basic probability theory. The best book out there on probability theory that I know is the freely available book by Kerns.%[?] This chapter very closely based on this book.

As Christensen et al. %[?] nicely put it, for most of our data analysis goals in this course, probability is simply the area under the curve in a probability distribution function.

1.3.1 Kolmogorov Axioms of Probability

I assume basic knowledge of set theory. Let S be a set of events. For example, for a single coin toss, $S = \{A_1, A_2\}$, where A_1 is the event that we get a heads, and A_2 the event that we get a tails.

[1] This is the maximum likelihood estimate.

1. Axiom 1 ($\mathbb{P}(A) \geq 0$) for any event ($A \subset S$).
2. Axiom 2 ($\mathbb{P}(S) = 1$).
3. Axiom 3 If the events $(A_1), (A_2), (A_3) \ldots$ are disjoint then

$$\mathbb{P}\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} \mathbb{P}(A_i) \text{ for every } n, \tag{1}$$

and furthermore,

$$\mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mathbb{P}(A_i). \tag{2}$$

1.3.1.1 Three important propositions

%We'll be using these later on a lot.

Proposition 1

Let $E \cup E^c = S$. Then,

$$1 = P(S) = P(E \cup E^c) = P(E) + P(E^c) \tag{3}$$

or:

$$P(E^c) = 1 - P(E) \tag{4}$$

** Proposition 2**

If $E \subset F$ then $P(E) \leq P(F)$.

Proposition 3

$$P(E \cup F) = P(E) + P(F) - P(EF) \tag{5}$$

%This result will be needed for a (to me) totally non-obvious outcome in multivariate distributions.

1.3.2 Conditional Probability

This is a central concept in this course. The conditional probability of event B given event A, denoted $\mathbb{P}(B \mid A)$, is defined by

$$\mathbb{P}(B \mid A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)}, \quad \text{if } \mathbb{P}(A) > 0. \tag{6}$$

** Theorem **

For any fixed event A with $\mathbb{P}(A) > 0$,

1. $\mathbb{P}(B|A) \geq 0$, for all events $B \subset S$,
2. $\mathbb{P}(S|A) = 1$, and
3. If B_1, B_2, B_3,... are disjoint events,

then:

$$\mathbb{P}\left(\bigcup_{k=1}^{\infty} B_k \,\middle|\, A \right) = \sum_{k=1}^{\infty} \mathbb{P}(B_k|A). \tag{7}$$

In other words, $\mathbb{P}(\cdot|A)$ is a legitimate probability function. With this fact in mind, the following properties are immediate:

For any events A, B, and C with $\mathbb{P}(A) > 0$,

1. $\mathbb{P}(B^c|A) = 1 - \mathbb{P}(B|A)$.
2. If $B \subset C$ then $\mathbb{P}(B|A) \leq \mathbb{P}(C|A)$.
3. $\mathbb{P}[(B \cup C)|A] = \mathbb{P}(B|A) + \mathbb{P}(C|A) - \mathbb{P}[(B \cap C|A)]$.
4. The Multiplication Rule. For any two events A and B,

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B|A). \tag{8}$$

And more generally, for events A_1, A_2, A_3,..., A_n,

$$\mathbb{P}(A_1 \cap A_2 \cap \cdots \cap A_n) = \mathbb{P}(A_1)\mathbb{P}(A_2|A_1) \cdots \mathbb{P}(A_n|A_1 \cap A_2 \cap \cdots \cap A_{n-1}). \tag{9}$$

1.3.3 Independence of events

[Taken nearly verbatim from Kerns.]

Definition

Events A and B are said to be independent if

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B). \tag{10}$$

Otherwise, the events are said to be dependent.

From the above definition of conditional probability, we know that when $\mathbb{P}(B) > 0$ we may write

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}. \tag{11}$$

In the case that A and B are independent, the numerator of the fraction factors so that $\mathbb{P}(B)$ cancels, with the result:

$$\mathbb{P}(A|B) = \mathbb{P}(A) \text{ when } A, B \text{ are independent.} \tag{12}$$

Proposition

If E and F are independent events, then so are E and F^c, E^c and F, and E^c and F^c.

Proof:

Assume E and F are independent. Since $E = EF \cup EF^c$ and EF and EF^c are mutually exclusive,

$$\begin{aligned} P(E) =& P(EF) + P(EF^c) \\ =& P(E)P(F) + P(EF^c) \end{aligned} \tag{13}$$

Equivalently:

$$\begin{aligned} P(EF^c) =& P(E)[1 - P(F)] \\ =& P(E)P(F^c) \end{aligned} \tag{14}$$

1.3.4 Bayes' rule

[Quoted nearly verbatim from Kerns.]

Theorem Bayes' Rule. Let B_1, B_2, ..., B_n be mutually exclusive and exhaustive and let A be an event with $\mathbb{P}(A) > 0$. Then

$$\mathbb{P}(B_k|A) = \frac{\mathbb{P}(B_k)\mathbb{P}(A|B_k)}{\sum_{i=1}^{n}\mathbb{P}(B_i)\mathbb{P}(A|B_i)}, \quad k = 1,2,\ldots,n. \tag{15}$$

The proof follows from looking at $\mathbb{P}(B_k \cap A)$ in two different ways. For simplicity, suppose that $P(B_k) > 0$ for all k. Then

$$\mathbb{P}(A)\mathbb{P}(B_k|A) = \mathbb{P}(B_k \cap A) = \mathbb{P}(B_k)\mathbb{P}(A|B_k). \tag{16}$$

Since $\mathbb{P}(A) > 0$ we may divide through to obtain

$$\mathbb{P}(B_k|A) = \frac{\mathbb{P}(B_k)\mathbb{P}(A|B_k)}{\mathbb{P}(A)}. \tag{17}$$

Now remembering that $\{B_k\}$ is a partition (i.e., mutually exclusive and exhaustive), the denominator of the last expression is

$$\mathbb{P}(A) = \sum_{k=1}^{n}\mathbb{P}(B_k \cap A) = \sum_{k=1}^{n}\mathbb{P}(B_k)\mathbb{P}(A|B_k). \tag{18}$$

1.3.5 Random variables

A random variable X is a function $X : S \to \mathbb{R}$ that associates to each outcome $\omega \in S$ exactly one number $X(\omega) = x$.

S_X is all the x's (all the possible values of X, the support of X). I.e., $x \in S_X$. It seems we can also sloppily write $X \in S_X$ (not sure about this).

Good example: number of coin tosses till H

- $X : \omega \to x$
- ω: H, TH, TTH,...(infinite)
- $x = 0,1,2,\ldots; x \in S_X$

Every discrete (continuous) random variable X has associated with it a probability mass (distribution) function (pmf, pdf). I.e., PMF is used for discrete distributions and PDF for continuous. (I will sometimes use lower case for pdf and sometimes upper case. Some books, like Christensen et al., use pdf for both discrete and continuous distributions.)

$$p_X : S_X \to [0,1] \tag{19}$$

defined by

$$p_X(x) = P(X(\omega) = x), x \in S_X \tag{20}$$

[Note: Books sometimes abuse notation by overloading the meaning of X. They usually have: $p_X(x) = P(X = x), x \in S_X$]

Probability density functions (continuous case) or probability mass functions (discrete case) are functions that assign probabilities or relative frequencies to all events in a sample space.

The expression

$$X \sim g(\cdot) \tag{21}$$

means that the random variable X has pdf/pmf $g(\cdot)$. For example, if we say that $X \sim N(\mu, \sigma^2)$, we are assuming that the pdf is

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp[-\frac{(x-\mu)^2}{2\sigma^2}] \tag{22}$$

We also need a cumulative distribution function or cdf because, in the continuous case, P(X=some point value) is zero and we therefore need a way to talk about P(X in a specific range). cdfs serve that purpose.

In the continuous case, the cdf or distribution function is defined as:

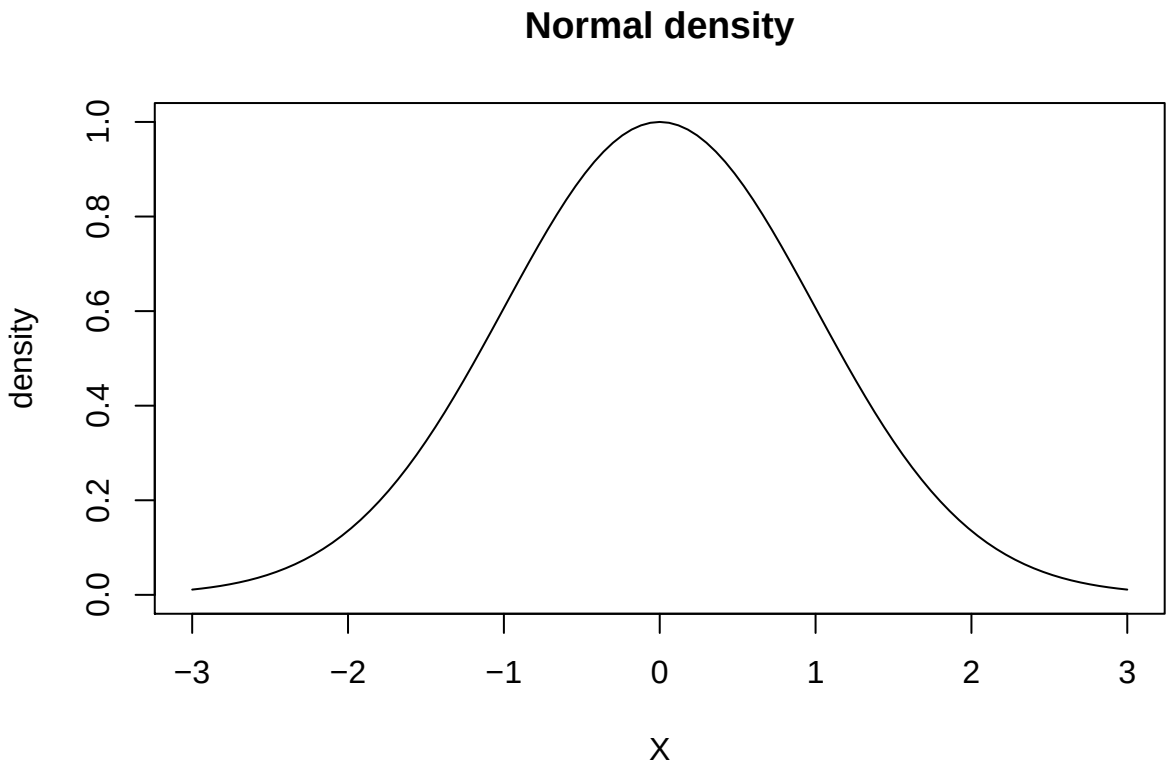
$$P(x < X) = F(x < X) = \int_{-\infty}^{X} f(x)\, dx \tag{23}$$

Note: Almost any function can be a pdf as long as it sums to 1 over the sample space. Here is an example of a function that doesn't sum to 1:

$$f(x) = \exp[-\frac{(x-\mu)^2}{2\sigma^2}] \tag{24}$$

This is (I think this is what it's called) the "kernel" of the normal pdf, and it doesn't sum to 1:

```
normkernel <- function(x, mu = 0, sigma = 1) {
    exp((-(x - mu)^2/(2 * (sigma^2))))
}

x <- seq(-10, 10, by = 0.01)

plot(function(x) normkernel(x), -3, 3, main = "Normal density",
    ylim = c(0, 1), ylab = "density", xlab = "X")
```



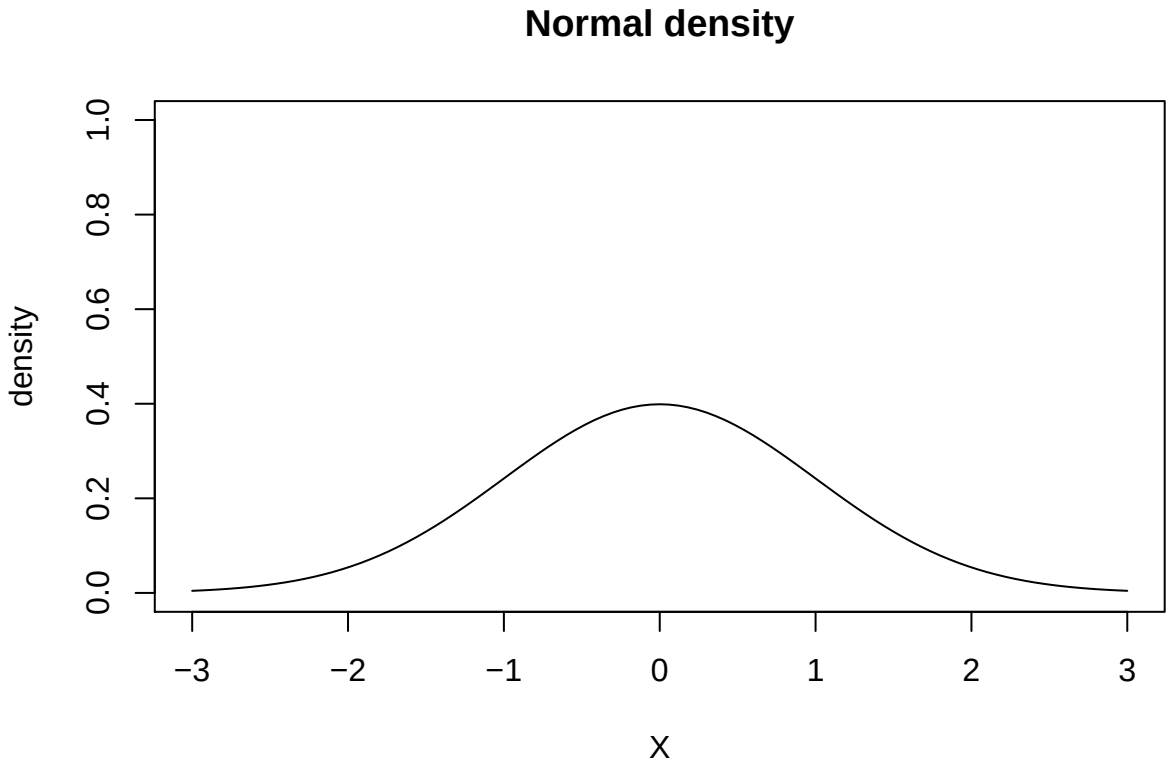

```
### area under the curve:
integrate(normkernel, lower = -Inf, upper = Inf)
```

```
## 2.51 with absolute error < 0.00023
```

Adding a normalizing constant makes the above kernel density a pdf.

```
norm <- function(x, mu = 0, sigma = 1) {
    (1/sqrt(2 * pi * (sigma^2))) * exp((-(x - mu)^2/(2 * (sigma^2))))
}
```

```r
x <- seq(-10, 10, by = 0.01)

plot(function(x) norm(x), -3, 3, main = "Normal density", ylim = c(0,
    1), ylab = "density", xlab = "X")
```




```r
### area under the curve:
integrate(norm, lower = -Inf, upper = Inf)
```

```
## 1 with absolute error < 0.000094
```

Recall that a random variable X is a function $X : S \to \mathbb{R}$ that associates to each outcome $\omega \in S$ exactly one number $X(\omega) = x$. S_X is all the x's (all the possible values of X, the support of X). I.e., $x \in S_X$.

X is a continuous random variable if there is a non-negative function f defined for all real $x \in (-\infty, \infty)$ having the property that for any set B of real numbers,

$$P\{X \in B\} = \int_B f(x)\,dx \tag{25}$$

Kerns has the following to add about the above:

> Continuous random variables have supports that look like

$$S_X = [a,b] \text{ or } (a,b), \tag{26}$$

or unions of intervals of the above form. Examples of random variables that are often taken to be continuous are:

- the height or weight of an individual,
- other physical measurements such as the length or size of an object, and
- durations of time (usually).

E.g., in linguistics we take as continous:

1. reading time: Here the random variable X has possible values ω ranging from 0 ms to some upper bound b ms, and the RV X maps each possible value ω to the corresponding number (0 to 0 ms, 1 to 1 ms, etc.).
2. acceptability ratings (technically not true; but people generally treat ratings as continuous, at least in psycholinguistics)
3. EEG signals

Every continuous random variable X has a probability density function (PDF) denoted f_X associated with it that satisfies three basic properties:

1. $f_X(x) > 0$ for $x \in S_X$,
2. $\int_{x \in S_X} f_X(x)\,\mathrm{d}x = 1$, and
3. $\mathbb{P}(X \in A) = \int_{x \in A} f_X(x)\,\mathrm{d}x$, for an event $A \subset S_X$.

We can say the following about continuous random variables:

- Usually, the set A in condition 3 above takes the form of an interval, for example, $A = [c,d]$, in which case

$$\mathbb{P}(X \in A) = \int_c^d f_X(x)\,\mathrm{d}x. \tag{27}$$

- It follows that the probability that X falls in a given interval is simply the area under the curve of f_X over the interval.
- Since the area of a line $x = c$ in the plane is zero, $\mathbb{P}(X = c) = 0$ for any value c. In other words, the chance that X equals a particular value c is zero, and this is true for any number c. Moreover, when $a < b$ all of the following probabilities are the same:

$$\mathbb{P}(a \leq X \leq b) = \mathbb{P}(a < X \leq b) = \mathbb{P}(a \leq X < b) = \mathbb{P}(a < X < b). \tag{28}$$

- The PDF f_X can sometimes be greater than 1. This is in contrast to the discrete case; every nonzero value of a PMF is a probability which is restricted to lie in the interval $[0,1]$.

$f(x)$ is the probability density function of the random variable X.

Since X must assume some value, f must satisfy

$$1 = P\{X \in (-\infty, \infty)\} = \int_{-\infty}^{\infty} f(x)\,dx \tag{29}$$

If $B = [a,b]$, then

$$P\{a \leq X \leq b\} = \int_{a}^{b} f(x)\,dx \tag{30}$$

If $a = b$, we get

$$P\{X = a\} = \int_{a}^{a} f(x)\,dx = 0 \tag{31}$$

Hence, for any continuous random variable,

$$P\{X < a\} = P\{X \leq a\} = F(a) = \int_{-\infty}^{a} f(x)\,dx \tag{32}$$

F is the cumulative distribution function. Differentiating both sides in the above equation:

$$\frac{dF(a)}{da} = f(a) \tag{33}$$

The density (PDF) is the derivative of the CDF.

1.3.5.1 Some basic results concerning random variables

1.
$$E[X] = \int_{-\infty}^{\infty} x f(x)\,dx \tag{34}$$

2.
$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f(x)\,dx \tag{35}$$

3.

$$E[aX+b] = aE[X]+b \tag{36}$$

4.

$$Var[X] = E[(X-\mu)^2] = E[X^2] - (E[X])^2 \tag{37}$$

5.

$$Var(aX+b) = a^2 Var(X) \tag{38}$$

So, so far, we know what a random variable is, and we know that by definition it has a pdf and a cdf associated with it.

1.3.6 What you can do with a pdf

You can:

1. Calculate the mean:

   Discrete case:

$$E[X] = \sum_{i=1}^{n} x_i p(x_i) \tag{39}$$

   Continuous case:

$$E[X] = \int_{-\infty}^{\infty} x f(x)\, dx \tag{40}$$

2. Calculate the variance:

$$Var(X) = E[X^2] - (E[X])^2 \tag{41}$$

3. Compute quartiles: e.g., for some pdf f(x):

$$\int_{-\infty}^{Q} f(x)\, dx \tag{42}$$

   For example, take $f(x)$ to be the normal distribution with mean 0 and sd 1. Suppose we want to know:

$$\int_{0}^{1} f(x)\, dx \tag{43}$$

We can do this in R as follows:[2]

“‘r pnorm(1) - pnorm(0) “‘

1.3.7 Some basic facts about expectation and variance

1. Computing expectation:

$$E[X] = \sum_{i=1}^{n} x_i p(x_i) \tag{44}$$

$$E[X] = \int_{-\infty}^{\infty} x f(x)\, dx \tag{45}$$

2. Computing expectation of a function of a random variable:

$$E[g(X)] = \sum_{i=1}^{n} g(x_i) p(x_i) \tag{46}$$

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f(x)\, dx \tag{47}$$

3. Computing variance:

$$Var(X) = E[(X-\mu)^2] \tag{48}$$

$$Var(X) = E[X^2] - (E[X])^2 \tag{49}$$

4. Computing variance of a linear transformation of an RV:

$$Var(aX+b) = a^2 Var(X) \tag{50}$$

[Notice that later on in matrix form we will get: $Var(AX+b) = AVar(X)A'$ for linear transformations like $y = AX+b$.]

5.

$$SD(X) = \sqrt{Var(X)} \tag{51}$$

[2]This is a very important piece of R code here. Make sure you understand the relationship between the integral and the R functions used here.

6. For two independent random variables X and Y,

$$E[XY] = E[X]E[Y] \tag{52}$$

7. Covariance of two random variables:

$$Cov(X,Y) = E[(X - E[X])(Y - E[Y])] \tag{53}$$

8. Note that Cov(X,Y)=0 if X and Y are independent.

   Corollary in 4.1 of Ross:

$$E[aX + b] = aE[X] + b \tag{54}$$

9. A related result is about linear combinations of RVs:
   Theorem 1. Given two not necessarily independent random variables X and Y:

$$E[aX + bY] = aE[X] + bE[Y] \tag{55}$$

10. If X and Y are independent,

$$Var(X + Y) = Var[X] + Var[Y] \tag{56}$$

    and

$$Var(aX + bY) = a^2 Var(X) + b^2 Var(Y) \tag{57}$$

    If $a = 1, b = -1$, then

$$Var(X - Y) = Var(X) + Var(Y) \tag{58}$$

11. If X and Y are not independent, then

$$Var(X - Y) = Var(X) + Var(Y) - 2Cov(X,Y) \tag{59}$$

1.3.8 Three common and important distributions

1.3.8.1 Binomial distribution

If we have x successes in n trials, given a success probability p for each trial. If $x \sim Bin(n,p)$.

$$P(x \mid n,p) = \binom{n}{k} p^k (1-p)^{n-k} \tag{60}$$

[Recall that: $\binom{n}{k} = \frac{n!}{(n-r)!r!}$]

The mean is np and the variance $np(1-p)$.

When $n = 1$ we have the Bernoulli distribution.

```
###pmf:
dbinom(x, size, prob, log = FALSE)
### cdf:
pbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
### quantiles:
qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
### pseudo-random generation of samples:
rbinom(n, size, prob)
```

1.3.8.2 Uniform random variable

A random variable (X) with the continuous uniform distribution on the interval (α,β) has PDF

$$f_X(x) = \begin{cases} \frac{1}{\beta-\alpha}, & \alpha < x < \beta, \\ 0, & \text{otherwise} \end{cases} \tag{61}$$

The associated R function is $\mathsf{dunif}(\mathtt{min} = a, \mathtt{max} = b)$. We write $X \sim \mathsf{unif}(\mathtt{min} = a, \mathtt{max} = b)$. Due to the particularly simple form of this PDF we can also write down explicitly a formula for the CDF F_X:

$$F_X(a) = \begin{cases} 0, & a < 0, \\ \frac{a-\alpha}{\beta-\alpha}, & \alpha \le t < \beta, \\ 1, & a \ge \beta. \end{cases} \tag{62}$$

$$E[X] = \frac{\beta + \alpha}{2} \tag{63}$$

$$Var(X) = \frac{(\beta - \alpha)^2}{12} \tag{64}$$

```
dunif(x, min = 0, max = 1, log = FALSE)
punif(q, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
qunif(p, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
runif(n, min = 0, max = 1)
```

1.3.8.3 Normal random variable

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \quad -\infty < x < \infty. \tag{65}$$

We write $X \sim \mathsf{norm}(\mathtt{mean} = \mu, \mathtt{sd} = \sigma)$, and the associated R function is `dnorm(x, mean = 0, sd = 1)`.

```
plot(function(x) dnorm(x), -3, 3, main = "Normal density", ylim = c(0,
    0.4), ylab = "density", xlab = "X")
```

If X is normally distributed with parameters μ and σ^2, then $Y = aX + b$ is normally distributed with parameters $a\mu + b$ and $a^2\sigma^2$.

Standard or unit normal random variable:

If X is normally distributed with parameters μ and σ^2, then $Z = (X - \mu)/\sigma$ is normally distributed with parameters $0, 1$.

We conventionally write $\Phi(x)$ for the CDF:

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{-y^2}{2}} dy \quad \text{where } y = (x - \mu)/\sigma \tag{66}$$

Old-style (pre-computer era) printed tables give the values for positive x; for negative x we do:

$$\Phi(-x) = 1 - \Phi(x), \quad -\infty < x < \infty \tag{67}$$

If Z is a standard normal random variable (SNRV) then

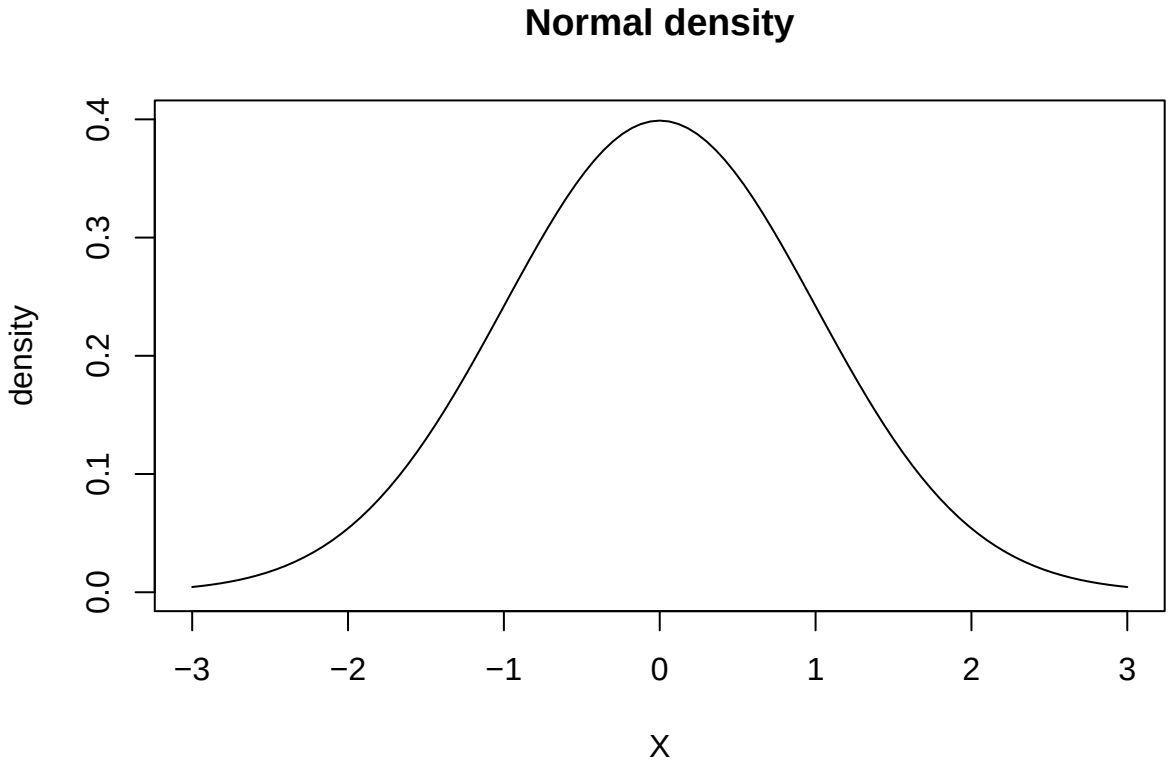


Figure 1: Normal distribution.

$$p\{Z \leq -x\} = P\{Z > x\}, \quad -\infty < x < \infty \tag{68}$$

Since $Z = ((X-\mu)/\sigma)$ is an SNRV whenever X is normally distributed with parameters μ and σ^2, then the CDF of X can be expressed as:

$$F_X(a) = P\{X \leq a\} = P\left(\frac{X-\mu}{\sigma} \leq \frac{a-\mu}{\sigma}\right) = \Phi\left(\frac{a-\mu}{\sigma}\right) \tag{69}$$

The standardized version of a normal random variable X is used to compute specific probabilities relating to X (it is also easier to compute probabilities from different CDFs so that the two computations are comparable).

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

The expectation of the standard normal random variable:

Here is how we can calculate the expectation of an SNRV.

$$E[Z] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x e^{-x^2/2}\, dx$$

Let $u = -x^2/2$.

Then, $du/dx = -2x/2 = -x$. I.e., $du = -x\,dx$ or $-du = x\,dx$.

We can rewrite the integral as:

$$E[Z] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{u} x\, dx$$

Replacing $x\,dx$ with $-du$ we get:

$$-\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{u}\, du$$

which yields:

$$-\frac{1}{\sqrt{2\pi}} [e^{u}]_{-\infty}^{\infty}$$

Replacing u with $-x^2/2$ we get:

$$-\frac{1}{\sqrt{2\pi}} [e^{-x^2/2}]_{-\infty}^{\infty} = 0$$

The variance of the standard normal distribution:

We know that

$$\mathrm{Var}(Z) = E[Z^2] - (E[Z])^2$$

Since $(E[Z])^2 = 0$ (see immediately above), we have

$$\mathrm{Var}(Z) = E[Z^2] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x^2 e^{-x^2/2}\, dx$$

Write x^2 as $x \times x$ and use integration by parts:

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x x e^{-x^2/2}\, dx = \frac{1}{\sqrt{2\pi}} x - e^{-x^2/2} - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} -e^{-x^2/2} 1\, dx = 1$$

$x \to \pm\infty$, $e^{-x^2/2}$ gets small more quickly than x gets large. The second summand is just the standard normal density integrated over its domain, so the value of this summand is 1. Therefore, the variance of the standard normal density equals 1."

Example:
Given $X \sim N(10, 16)$, write distribution of $\bar{X}$, where $n = 4$. Since $SE = sd/sqrt(n)$, the distribution of $\bar{X}$ is $N(10, 16/4)$.

1.4 Important probability distributions

1.4.1 Multinomial coefficients and multinomial distributions

[Taken almost verbatim from Kerns, with some additional stuff from Ross.]

We sample n times, with replacement, from an urn that contains balls of k different types. Let X_1 denote the number of balls in our sample of type 1, let X_2 denote the number of balls of type 2, ..., and let X_k denote the number of balls of type k. Suppose the urn has proportion p_1 of balls of type 1, proportion p_2 of balls of type 2, ..., and proportion p_k of balls of type k. Then the joint PMF of $(X_1, \ldots, X_k)$ is

$$f_{X_1,\ldots,X_k}(x_1, \ldots, x_k) = \binom{n}{x_1\, x_2 \cdots x_k} p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k}, \tag{70}$$

for $(x_1, \ldots, x_k)$ in the joint support $S_{X_1,\ldots X_K}$. We write

$$(X_1, \ldots, X_k) \sim \mathsf{multinom}(\mathtt{size} = n, \mathtt{prob} = \mathbf{p}_{\mathrm{k}\times 1}). \tag{71}$$

Note:

First, the joint support set $S_{X_1,\ldots X_K}$ contains all nonnegative integer k-tuples $(x_1, \ldots, x_k)$ such that $x_1 + x_2 + \cdots + x_k = n$. A support set like this is called a simplex. Second, the proportions p_1, p_2, ..., p_k satisfy $p_i \geq 0$ for all i and $p_1 + p_2 + \cdots + p_k = 1$. Finally, the symbol

$$\binom{n}{x_1\, x_2 \cdots x_k} = \frac{n!}{x_1!\, x_2! \cdots x_k!} \tag{72}$$

is called a multinomial coefficient which generalizes the notion of a binomial coefficient.

Example from Ross

Suppose a fair die is rolled nine times. The probability that 1 appears three times, 2 and 3 each appear twice, 4 and 5 each appear once, and 6 not at all, can be computed using the multinomial distribution formula.

Here, for $i = 1, \ldots, 6$, it is clear that $p_i == \frac{1}{6}$. And it is clear that $n = 9$, and $x_1 = 3$, $x_2 = 2$, $x_3 = 2$, $x_4 = 1$, $x_5 = 1$, and $x_6 = 0$. We plug in the values into the formula:

$$f_{X_1,\ldots,X_k}(x_1,\ldots,x_k) = \binom{n}{x_1\, x_2 \cdots x_k} p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k} \tag{73}$$

Plugging in the values:

$$f_{X_1,\ldots,X_k}(x_1,\ldots,x_k) = \binom{9}{3\,2\,2\,1\,1\,0} \frac{1}{6}^3 \frac{1}{6}^2 \frac{1}{6}^2 \frac{1}{6}^1 \frac{1}{6}^1 \frac{1}{6}^0 \tag{74}$$

Answer: $\frac{9!}{3!2!2!}\left(\frac{1}{6}\right)^9$

1.4.2 The Poisson distribution

As Kerns puts it (I quote him nearly exactly, up to the definition):

> This is a distribution associated with "rare events", for reasons which will become clear in a moment. The events might be:
>
> - traffic accidents,
> - typing errors, or
> - customers arriving in a bank.
>
> For us, I suppose one application might be in eye tracking: modeling number of fixations. Psychologists often treat these as continuous values, which doesn't seem to make much sense to me (what kind of continuous random variable would generate a distribution of 0,1, 2, 3 fixations?).
>
> Let λ be the average number of events in the time interval $[0,1]$. Let the random variable X count the number of events occurring in the interval. Then under certain reasonable conditions it can be shown that

$$f_X(x) = \mathbb{P}(X = x) = \mathrm{e}^{-\lambda} \frac{\lambda^x}{x!}, \quad x = 0, 1, 2, \ldots \tag{75}$$

1.4.3 Geometric distribution

From Ross (page 155): %[?]:

> Let independent trials, each with probability p, $0 < p < 1$ of success, be performed until a success occurs. If X is the number of trials required till success occurs, then
>
> $$P(X = n) = (1-p)^{n-1}p \quad n = 1, 2, \ldots$$
>
> I.e., for X to equal n, it is necessary and sufficient that the first $n-1$ are failures, and the nth trial is a success. The above equation comes about because the successive trials are independent.

X is a geometric random variable with parameter p.

Note that a success will occur, with probability 1:

$$\sum_{i=1}^{\infty} P(X = n) = p\sum_{i=1}^{\infty}(1-p)^{n-1} = \frac{p}{1-(1-p)} = 1$$

Mean and variance of the geometric distribution

$$E[X] = \frac{1}{p}$$

$$Var(X) = \frac{1-p}{p^2}$$

For proofs, see Ross [?] (pages 156-157).

1.4.4 Exponential random variables

For some $\lambda > 0$,

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0. \end{cases}$$

A continuous random variable with the above PDF is an exponential random variable (or is said to be exponentially distributed).

The CDF:

$$\begin{aligned} F(a) =& P(X \leq a) \\ =& \int_0^a \lambda e^{-\lambda x} dx \\ =& \left[-e^{-\lambda x} \right]_0^a \\ =& 1 - e^{-\lambda a} \quad a \geq 0 \end{aligned}$$

[Note: the integration requires the u-substitution: $u = -\lambda x$, and then $du/dx = -\lambda$, and then use $-du = \lambda dx$ to solve.]

Expectation and variance of an exponential random variable

For some $\lambda > 0$ (called the rate), if we are given the PDF of a random variable X:

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0. \end{cases}$$

Find E[X].

[This proof seems very strange and arbitrary—one starts really generally and then scales down, so to speak. The standard method can equally well be used, but this is more general, it allows for easy calculation of the second moment, for example. Also, it's an example of how reduction formulae are used in integration.]

$$E[X^n] = \int_0^\infty x^n \lambda e^{-\lambda x} dx$$

Use integration by parts:

Let $u = x^n$, which gives $du/dx = nx^{n-1}$. Let $dv/dx = \lambda e^{-\lambda x}$, which gives $v = -e^{-\lambda x}$. Therefore:

$$\begin{aligned} E[X^n] =& \int_0^\infty x^n \lambda e^{-\lambda x} dx \\ =& \left[-x^n e^{-\lambda x} \right]_0^\infty + \int_0^\infty e^{\lambda x} n x^{n-1} dx \\ =& 0 + \frac{n}{\lambda} \int_0^\infty \lambda e^{-\lambda x} n^{n-1} dx \end{aligned}$$

Thus,

$$E[X^n] = \frac{n}{\lambda} E[X^{n-1}]$$

If we let $n = 1$, we get $E[X]$:

$$E[X] = \frac{1}{\lambda}$$

Note that when $n = 2$, we have

$$E[X^2] = \frac{2}{\lambda} E[X] = \frac{2}{\lambda^2}$$

Variance is, as usual,

$$var(X) = E[X^2] - (E[X])^2 = \frac{2}{\lambda^2} - (\frac{1}{\lambda})^2 = \frac{1}{\lambda^2}$$

1.4.5 Gamma distribution

[The text is an amalgam of Kerns and Ross %[?] (page 215). I don't put it in double-quotes as a citation because it would look ugly.]

This is a generalization of the exponential distribution. We say that X has a gamma distribution and write $X \sim \texttt{gamma}(\texttt{shape} = \alpha, \texttt{rate} = \lambda)$, where $\alpha > 0$ (called shape) and $\lambda > 0$ (called rate). It has PDF

$$f(x) = \begin{cases} \frac{\lambda e^{-\lambda x} (\lambda x)^{\alpha - 1}}{\Gamma(\alpha)} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0. \end{cases}$$

$\Gamma(\alpha)$ is called the gamma function:

$$\Gamma(\alpha) = \int_0^\infty e^{-y} y^{\alpha - 1} \, dy = (\alpha - 1)\Gamma(\alpha - 1)$$

Note that for integral values of n, $\Gamma(n) = (n-1)!$ (follows from above equation).

The associated R functions are `gamma(x, shape, rate = 1)`, `pgamma`, `qgamma`, and `rgamma`, which give the PDF, CDF, quantile function, and simulate random variates, respectively. If $\alpha = 1$ then $X \sim \texttt{exp}(\texttt{rate} = \lambda)$. The mean is $\mu = \alpha/\lambda$ and the variance is $\sigma^2 = \alpha/\lambda^2$.

To motivate the gamma distribution recall that if X measures the length of time until the first event occurs in a Poisson process with rate λ then $X \sim \texttt{exp}(\texttt{rate} = \lambda)$. If we let Y measure the length of time until the α^{th} event occurs then $Y \sim \texttt{gamma}(\texttt{shape} = \alpha, \texttt{rate} = \lambda)$. When α is an integer this distribution is also known as the Erlang distribution.

```
x <- seq(0, 4, by = 0.01)
plot(x, gamma.fn(x), type = "l")
```

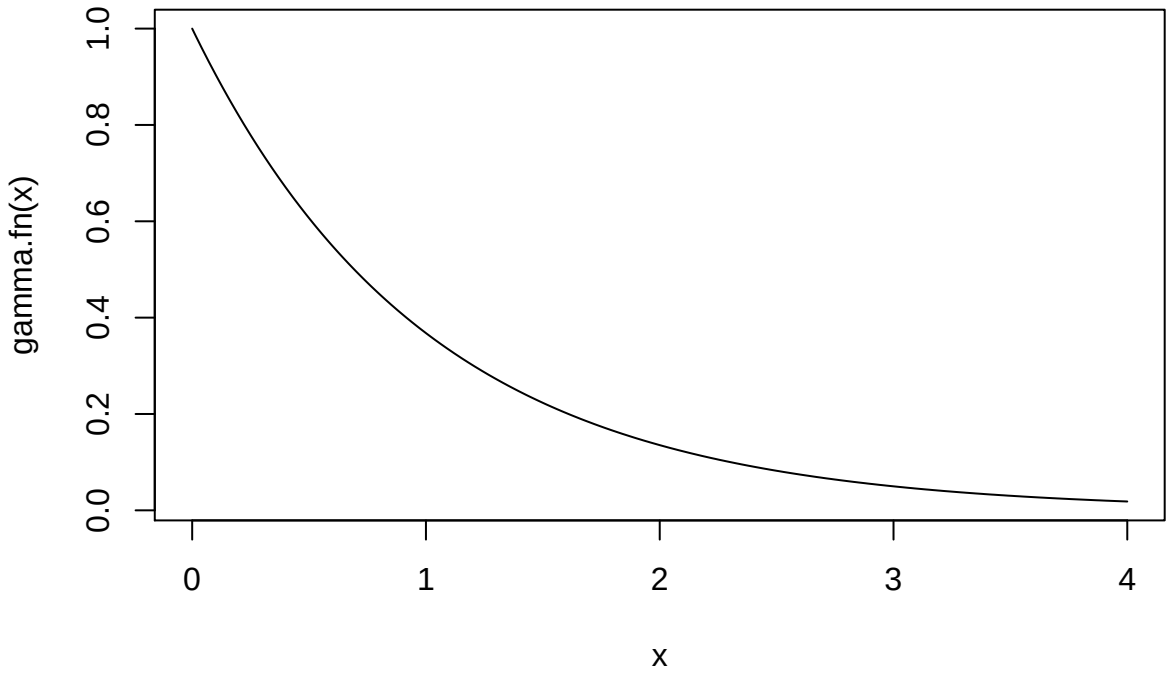


Figure 2: The gamma distribution.

The Chi-squared distribution is the gamma distribution with $\lambda = 1/2$ and $\alpha = n/2$, where n is an integer:

```
x <- seq(0, 100, by = 0.01)
plot(x, gamma.fn(x), type = "l")
```

Mean and variance of Gamma distribution

Let X be a gamma random variable with parameters α and λ.

$$\begin{aligned} E[X] &= \frac{1}{\Gamma(\alpha)} \int_0^\infty x\lambda e^{-\lambda x} (\lambda x)^{\alpha-1} \, dx \\ &= \frac{1}{\lambda\Gamma(\alpha)} \int_0^\infty e^{-\lambda x} (\lambda x)^{\alpha} \, dx \\ &= \frac{\Gamma(\alpha+1)}{\lambda\Gamma(\alpha)} \\ &= \frac{\alpha}{\lambda} \quad \text{see derivation of } \Gamma(\alpha), p.\ 215 \text{ of Ross} \end{aligned}$$

It is easy to show (exercise) that

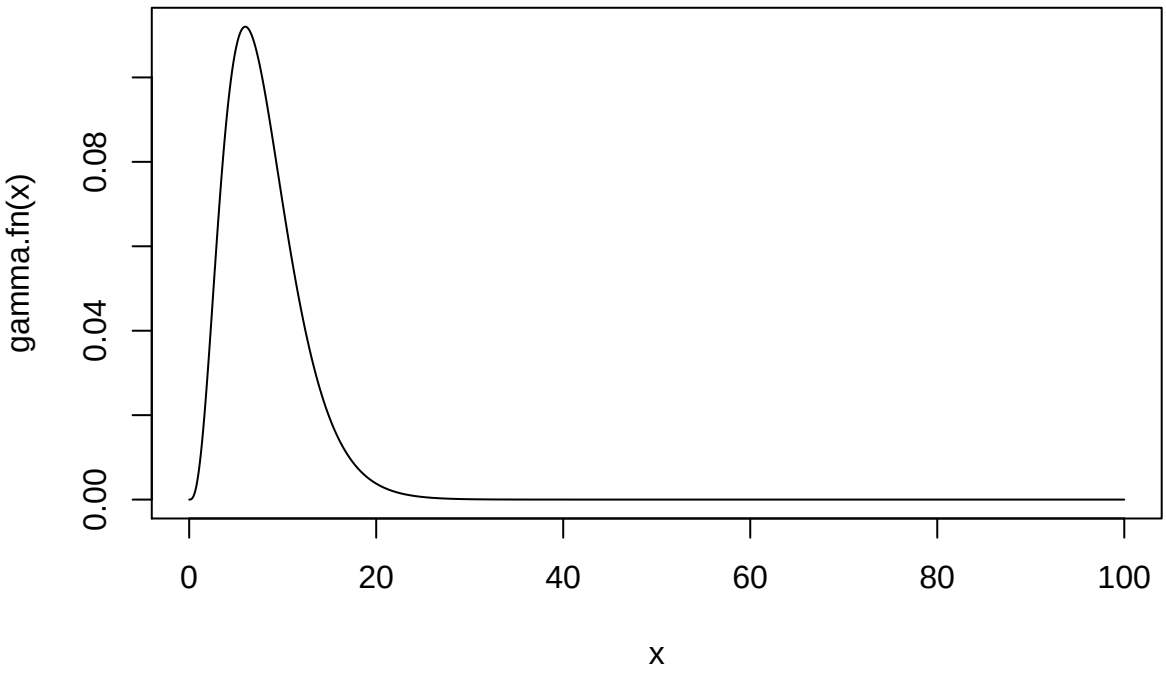


Figure 3: The chi-squared distribution.

$$Var(X) = \frac{\alpha}{\lambda^2}$$

1.4.6 Beta distribution

This is a generalization of the continuous uniform distribution.

$$f(x) = \begin{cases} \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1} & \text{if } 0 < x < 1 \\ 0 & \text{otherwise} \end{cases}$$

where

$$B(a,b) = \int_0^1 x^{a-1}(1-x)^{b-1}\,dx$$

There is a connection between the beta and the gamma:

$$B(a,b) = \int_0^1 x^{a-1}(1-x)^{b-1}\,dx = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$

which allows us to rewrite the beta PDF as

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}, \quad 0 < x < 1. \tag{76}$$

We write $X \sim \texttt{beta}(\texttt{shape1} = \alpha, \texttt{shape2} = \beta)$. The associated R function is =dbeta(x, shape1, shape2)=.

The mean and variance are

$$E[X] = \frac{a}{a+b} \text{ and } Var(X) = \frac{ab}{(a+b)^2(a+b+1)}. \tag{77}$$

This distribution is going to turn up a lot in Bayesian data analysis.

1.4.7 t distribution

A random variable X with PDF

$$f_X(x) = \frac{\Gamma[(r+1)/2]}{\sqrt{r\pi}\,\Gamma(r/2)} \left(1 + \frac{x^2}{r}\right)^{-(r+1)/2}, \quad -\infty < x < \infty \tag{78}$$

is said to have Student's t distribution with r degrees of freedom, and we write $X \sim \texttt{t}(\texttt{df} = r)$. The associated R functions are dt, pt, qt, and rt, which give the PDF, CDF, quantile function, and simulate random variates, respectively.

We will just write:

$X \sim t(\mu, \sigma^2, r)$, where r is the degrees of freedom $(n-1)$, where n is sample size.

1.5 Jointly distributed random variables

1.5.1 Discrete case

[This section is an extract from Kerns.]

Consider two discrete random variables X and Y with PMFs f_X and f_Y that are supported on the sample spaces S_X and S_Y, respectively. Let $S_{X,Y}$ denote the set of all possible observed pairs (x,y), called the joint support set of X and Y. Then the joint probability mass function of X and Y is the function $f_{X,Y}$ defined by

$$f_{X,Y}(x,y) = \mathbb{P}(X = x, Y = y), \quad \text{for } (x,y) \in S_{X,Y}. \tag{79}$$

Every joint PMF satisfies

$$f_{X,Y}(x,y) > 0 \text{ for all } (x,y) \in S_{X,Y}, \tag{80}$$

and

$$\sum_{(x,y)\in S_{X,Y}} f_{X,Y}(x,y) = 1. \tag{81}$$

It is customary to extend the function $f_{X,Y}$ to be defined on all of $\mathbb{R}^2$ by setting $f_{X,Y}(x,y) = 0$ for $(x,y) \notin S_{X,Y}$.

In the context of this chapter, the PMFs f_X and f_Y are called the marginal PMFs of X and Y, respectively. If we are given only the joint PMF then we may recover each of the marginal PMFs by using the Theorem of Total Probability: observe

$$\begin{aligned} f_X(x) &= \mathbb{P}(X = x), & (82)\\ &= \sum_{y\in S_Y} \mathbb{P}(X = x, Y = y), & (83)\\ &= \sum_{y\in S_Y} f_{X,Y}(x,y). & (84) \end{aligned}$$

By interchanging the roles of X and Y it is clear that

$$f_Y(y) = \sum_{x\in S_X} f_{X,Y}(x,y). \tag{85}$$

Given the joint PMF we may recover the marginal PMFs, but the converse is not true. Even if we have both marginal distributions they are not sufficient to determine the joint PMF; more information is needed. %[3]

Associated with the joint PMF is the joint cumulative distribution function $F_{X,Y}$ defined by

$$F_{X,Y}(x,y) = \mathbb{P}(X \le x, Y \le y), \quad \text{for } (x,y) \in \mathbb{R}^2.$$

The bivariate joint CDF is not quite as tractable as the univariate CDFs, but in principle we could calculate it by adding up quantities of the form in Equation~79. The joint CDF is typically not used in practice due to its inconvenient form; one can usually get by with the joint PMF alone.

[3]We are not at a total loss, however. There are Frechet bounds which pose limits on how large (and small) the joint distribution must be at each point.

Examples from Kerns:

Example 1:

Roll a fair die twice. Let X be the face shown on the first roll, and let Y be the face shown on the second roll. For this example, it suffices to define

$$f_{X,Y}(x,y) = \frac{1}{36}, \quad x = 1,\ldots,6,\ y = 1,\ldots,6.$$

The marginal PMFs are given by $f_X(x) = 1/6$, $x = 1,2,\ldots,6$, and $f_Y(y) = 1/6$, $y = 1,2,\ldots,6$, since

$$f_X(x) = \sum_{y=1}^{6} \frac{1}{36} = \frac{1}{6}, \quad x = 1,\ldots,6,$$

and the same computation with the letters switched works for Y.

Here, and in many other ones, the joint support can be written as a product set of the support of X "times" the support of Y, that is, it may be represented as a cartesian product set, or rectangle, $S_{X,Y} = S_X \times S_Y$ where $S_X \times S_Y = \{(x,y) :\ x \in S_X, y \in S_Y\}$. This form is a necessary condition for X and Y to be independent (or alternatively exchangeable when $S_X = S_Y$). But please note that in general it is not required for $S_{X,Y}$ to be of rectangle form.

Example 2: see very involved example 7.2 in Kerns, worth study.

1.5.2 Continuous case

For random variables X and y, the joint cumulative pdf is

$$F(a,b) = P(X \le a, Y \le b) \quad -\infty < a,b < \infty \tag{86}$$

The marginal distributions of F_X and F_Y are the CDFs of each of the associated RVs:

1. The CDF of X:

$$F_X(a) = P(X \le a) = F_X(a,\infty) \tag{87}$$

2. The CDF of Y:

$$F_Y(a) = P(Y \le b) = F_Y(\infty,b) \tag{88}$$

Definition 1. Jointly continuous: Two RVs X and Y are jointly continuous if there exists a function $f(x,y)$ defined for all real x and y, such that for every set C:

$$P((X,Y) \in C) = \iint\limits_{(x,y)\in C} f(x,y)\,dx\,dy \tag{89}$$

$f(x,y)$ is the joint PDF of X and Y.

Every joint PDF satisfies

$$f(x,y) \geq 0 \text{ for all } (x,y) \in S_{X,Y}, \tag{90}$$

and

$$\iint\limits_{S_{X,Y}} f(x,y)\,\mathrm{d}x\,\mathrm{d}y = 1. \tag{91}$$

For any sets of real numbers A and B, and if $C = \{(x,y) : x \in A, y \in B\}$, it follows from equation~89 that

$$P((X \in A, Y \in B) \in C) = \int_B \int_A f(x,y)\,dx\,dy \tag{92}$$

Note that

$$F(a,b) = P(X \in (-\infty,a]), Y \in (-\infty,b])) = \int_{-\infty}^{b}\int_{-\infty}^{a} f(x,y)\,dx\,dy \tag{93}$$

Differentiating, we get the joint pdf:

$$f(a,b) = \frac{\partial^2}{\partial a \partial b} F(a,b) \tag{94}$$

One way to understand the joint PDF:

$$P(a < X < a+da, b < Y < b+db) = \int_b^{d+db}\int_a^{a+da} f(x,y)\,dx\,dy \approx f(a,b)dadb \tag{95}$$

Hence, $f(x,y)$ is a measure of how probable it is that the random vector (X,Y) will be near (a,b).

1.5.3 Marginal probability distribution functions

If X and Y are jointly continuous, they are individually continuous, and their PDFs are:

$$\begin{aligned} P(X \in A) =& P(X \in A, Y \in (-\infty, \infty)) \\ =& \int_A \int_{-\infty}^{\infty} f(x,y)\, dy\, dx \\ =& \int_A f_X(x)\, dx \end{aligned} \tag{96}$$

where

$$f_X(x) = \int_{-\infty}^{\infty} f(x,y)\, dy \tag{97}$$

Similarly:

$$f_Y(y) = \int_{-\infty}^{\infty} f(x,y)\, dx \tag{98}$$

1.5.4 Independent random variables

Random variables X and Y are independent iff, for any two sets of real numbers A and B:

$$P(X \in A, Y \in B) = P(X \in A) P(Y \in B) \tag{99}$$

In the jointly continuous case:

$$f(x,y) = f_X(x) f_Y(y) \quad \text{for all } x, y \tag{100}$$

A necessary and sufficient condition for the random variables X and Y to be independent is for their joint probability density function (or joint probability mass function in the discrete case) $f(x,y)$ to factor into two terms, one depending only on x and the other depending only on y. %This can be stated as a proposition:

Easy-to-understand example from Kerns: Let the joint PDF of (X,Y) be given by

$$f_{X,Y}(x,y) = \frac{6}{5}\left(x + y^2\right), \quad 0 < x < 1,\ 0 < y < 1.$$

The marginal PDF of X is

$$\begin{aligned} f_X(x) &= \int_0^1 \frac{6}{5}\left(x+y^2\right)\,\mathrm{d}y, \\ &= \frac{6}{5}\left(xy+\frac{y^3}{3}\right)\Big|_{y=0}^{1}, \\ &= \frac{6}{5}\left(x+\frac{1}{3}\right), \end{aligned}$$

for $0 < x < 1$, and the marginal PDF of Y is

$$\begin{aligned} f_Y(y) &= \int_0^1 \frac{6}{5}\left(x+y^2\right)\,\mathrm{d}x, \\ &= \frac{6}{5}\left(\frac{x^2}{2}+xy^2\right)\Big|_{x=0}^{1}, \\ &= \frac{6}{5}\left(\frac{1}{2}+y^2\right), \end{aligned}$$

for $0 < y < 1$.

In this example the joint support set was a rectangle $[0,1] \times [0,1]$, but it turns out that X and Y are not independent. This is because $\frac{6}{5}\left(x+y^2\right)$ cannot be stated as a product of two terms $(f_X(x)f_Y(y))$.

1.5.5 Sums of independent random variables

[Taken nearly verbatim from Ross.]

Suppose that X and Y are independent, continuous random variables having probability density functions f_X and f_Y. The cumulative distribution function of $X+Y$ is obtained as follows:

$$
\begin{aligned}
F_{X+Y}(a) =& P(X+Y \leq a) \\
&= \iint\limits_{x+y\leq a} f_{XY}(x,y)\,dx\,dy \\
&= \iint\limits_{x+y\leq a} f_X(x) f_Y(y)\,dx\,dy \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{a-y} f_X(x) f_Y(y)\,dx\,dy \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{a-y} f_X(x)\,dx f_Y(y)\,dy \\
&= \int_{-\infty}^{\infty} F_X(a-y) f_Y(y)\,dy
\end{aligned}
\tag{101}
$$

The CDF F_{X+Y} is the convolution of the distributions F_X and F_Y.

If we differentiate the above equation, we get the pdf f_{X+Y}:

$$
\begin{aligned}
f_{X+Y} =& \frac{d}{dx}\int_{-\infty}^{\infty} F_X(a-y) f_Y(y)\,dy \\
&= \int_{-\infty}^{\infty} \frac{d}{dx} F_X(a-y) f_Y(y)\,dy \\
&= \int_{-\infty}^{\infty} f_X(a-y) f_Y(y)\,dy
\end{aligned}
\tag{102}
$$

1.5.6 Conditional distributions

1.5.6.1 Discrete case

Recall that the conditional probability of B given A, denoted $\mathbb{P}(B \mid A)$, is defined by

$$
\mathbb{P}(B \mid A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)}, \quad \text{if } \mathbb{P}(A) > 0. \tag{103}
$$

If X and Y are discrete random variables, then we can define the conditional PMF of X given that $Y = y$ as follows:

$$\begin{aligned} p_{X|Y}(x \mid y) =& P(X = x \mid Y = y) \\ =& \frac{P(X = x, Y = y)}{P(Y = y)} \\ =& \frac{p(x,y)}{p_Y(y)} \end{aligned} \tag{104}$$

for all values of y where $p_Y(y) = P(Y = y) > 0$.

The conditional cumulative distribution function of X given $Y = y$ is defined, for all y such that $p_Y(y) > 0$, as follows:

$$\begin{aligned} F_{X|Y} =& P(X \leq x \mid Y = y) \\ =& \sum_{a \leq x} p_{X|Y}(a \mid y) \end{aligned} \tag{105}$$

If X and Y are independent then

$$p_{X|Y}(x \mid y) = P(X = x) = p_X(x) \tag{106}$$

See the examples starting p. 264 of Ross.

1.5.6.2 Continuous case

[Taken almost verbatim from Ross.]

If X and Y have a joint probability density function $f(x,y)$, then the conditional probability density function of X given that $Y = y$ is defined, for all values of y such that $f_Y(y) > 0$,by

$$f_{X|Y}(x \mid y) = \frac{f(x,y)}{f_Y(y)} \tag{107}$$

We can understand this definition by considering what $f_{X|Y}(x \mid y)\,dx$ amounts to:

$$\begin{aligned} f_{X|Y}(x \mid y)\,dx =& \frac{f(x,y)}{f_Y(y)} \frac{dxdy}{dy} \\ =& \frac{f(x,y)dxdy}{f_Y(y)dy} \\ =& \frac{P(x < X < d + dx, y < Y < y + dy)}{y < P < y + dy} \end{aligned} \tag{108}$$

1.5.7 Joint and marginal expectation

[Taken nearly verbatim from Kerns.]

Given a function g with arguments (x,y) we would like to know the long-run average behavior of $g(X,Y)$ and how to mathematically calculate it. Expectation in this context is computed by integrating (summing) with respect to the joint probability density (mass) function.

Discrete case:

$$\mathbb{E}\,g(X,Y) = \sum_{(x,y)\in S_{X,Y}}\sum g(x,y)\,f_{X,Y}(x,y). \tag{109}$$

Continuous case:

$$\mathbb{E}\,g(X,Y) = \iint_{S_{X,Y}} g(x,y)\,f_{X,Y}(x,y)\,\mathrm{d}x\,\mathrm{d}y, \tag{110}$$

1.5.8 Covariance and correlation

There are two very special cases of joint expectation: the covariance and the correlation. These are measures which help us quantify the dependence between X and Y.
Definition 2. The covariance of X and Y is

$$\mathrm{Cov}(X,Y) = \mathbb{E}(X-\mathbb{E}X)(Y-\mathbb{E}Y). \tag{111}$$

Shortcut formula for covariance:

$$\mathrm{Cov}(X,Y) = \mathbb{E}(XY) - (\mathbb{E}X)(\mathbb{E}Y). \tag{112}$$

The Pearson product moment correlation between X and Y is the covariance between X and Y rescaled to fall in the interval $[-1,1]$. It is formally defined by

$$\mathrm{Corr}(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sigma_X\,\sigma_Y}. \tag{113}$$

The correlation is usually denoted by $\rho_{X,Y}$ or simply ρ if the random variables are clear from context. There are some important facts about the correlation coefficient:

1. The range of correlation is $-1 \le \rho_{X,Y} \le 1$.

2. Equality holds above ($\rho_{X,Y} = \pm 1$) if and only if Y is a linear function of X with probability one.

Continuous example from Kerns: Let us find the covariance of the variables (X,Y) from an example numbered 7.2 in Kerns. The expected value of X is

$$\mathbb{E}X = \int_0^1 x \cdot \frac{6}{5}\left(x + \frac{1}{3}\right) \mathrm{d}x = \frac{2}{5}x^3 + \frac{1}{5}x^2\Big|_{x=0}^{1} = \frac{3}{5},$$

and the expected value of Y is

$$\mathbb{E}Y = \int_0^1 y \cdot \frac{6}{5}\left(\frac{1}{2} + y^2\right) \mathrm{d}x = \frac{3}{10}y^2 + \frac{3}{20}y^4\Big|_{y=0}^{1} = \frac{9}{20}.$$

Finally, the expected value of XY is

$$\begin{aligned}
\mathbb{E}XY &= \int_0^1 \int_0^1 xy\frac{6}{5}\left(x + y^2\right) \mathrm{d}x\,\mathrm{d}y, \\
&= \int_0^1 \left(\frac{2}{5}x^3 y + \frac{3}{10}xy^4\right)\Big|_{x=0}^{1} \mathrm{d}y, \\
&= \int_0^1 \left(\frac{2}{5}y + \frac{3}{10}y^4\right) \mathrm{d}y, \\
&= \frac{1}{5} + \frac{3}{50},
\end{aligned}$$

which is 13/50. Therefore the covariance of (X,Y) is

$$\mathrm{Cov}(X,Y) = \frac{13}{50} - \left(\frac{3}{5}\right)\left(\frac{9}{20}\right) = -\frac{1}{100}.$$

1.5.9 Conditional expectation

Recall that

$$f_{X|Y}(x \mid y) = P(X = x \mid Y = y) = \frac{p_{X,Y}(x,y)}{p_Y(y)} \tag{114}$$

for all y such that $P(Y = y) > 0$.

It follows that

$$\begin{aligned}
E[X \mid Y = y] =& \sum_x xP(X = x \mid Y = y) \\
=& \sum_x x p_{X|Y}(x \mid y)
\end{aligned} \tag{115}$$

$E[X \mid Y]$ is that function of the random variable Y whose value at $Y = y$ is $E[X \mid Y = y]$. $E[X \mid Y]$ is a random variable.

Relationship to 'regular' expectation

Conditional expectation given that $Y = y$ can be thought of as being an ordinary expectation on a reduced sample space consisting only of outcomes for which $Y = y$. All properties of expectations hold. Two examples (to-do: spell out the other equations):

Example 1: to-do: develop some specific examples.

$$E[g(X) \mid Y = y] = \begin{cases} \sum_x g(x) p_{X|Y}(x,y) & \text{in the discrete case} \\ \int_{-\infty}^{\infty} g(x) f_{X|Y}(x \mid y)\, dx & \text{in the continuous case} \end{cases}$$

Example 2:

$$E\left[\sum_{i=1}^{n} X_i \mid Y = y\right] = \sum_{i=1}^{n} E[X_i \mid Y = y] \tag{116}$$

Proposition 1. Expectation of the conditional expectation

$$E[X] = E[E[X \mid Y]] \tag{117}$$

If Y is a discrete random variable, then the above proposition states that

$$E[X] = \sum_y E[X \mid Y = y] P(Y = y) \tag{118}$$

1.5.10 Multivariate normal distributions

Recall that in the univariate case:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e\{-\frac{(\frac{(x-\mu)}{\sigma})^2}{2}\} \quad -\infty < x < \infty \tag{119}$$

We can write the power of the exponential as:

$$(\frac{(x-\mu)}{\sigma})^2 = (x-\mu)(x-\mu)(\sigma^2)^{-1} = (x-\mu)(\sigma^2)^{-1}(x-\mu) = Q \tag{120}$$

Generalizing this to the multivariate case:

$$Q = (x-\mu)'\Sigma^{-1}(x-\mu) \tag{121}$$

So, for multivariate case:

$$f(x) = \frac{1}{\sqrt{2\pi det\Sigma}} e\{-Q/2\} \quad -\infty < x_i < \infty, i = 1, \ldots, n \tag{122}$$

Properties of the multivariate normal (MVN) X:

- Linear combinations of X are normal distributions.
- All subset's of X's components have a normal distribution.
- Zero covariance implies independent distributions.
- Conditional distributions are normal.

Visualizing bivariate distributions

First, a visual of two uncorrelated RVs:

```
library(MASS)

bivn <- mvrnorm(1000, mu = c(0, 1), Sigma = matrix(c(1, 0, 0,
    2), 2))
bivn.kde <- kde2d(bivn[, 1], bivn[, 2], n = 50)
persp(bivn.kde, phi = 10, theta = 0, shade = 0.2, border = NA,
    main = "Simulated bivariate normal density")
```

And here is an example of a positively correlated case:

```
bivn <- mvrnorm(1000, mu = c(0, 1), Sigma = matrix(c(1, 0.9,
    0.9, 2), 2))
bivn.kde <- kde2d(bivn[, 1], bivn[, 2], n = 50)
persp(bivn.kde, phi = 10, theta = 0, shade = 0.2, border = NA,
    main = "Simulated bivariate normal density")
```

And here is an example with a negative correlation:

```
bivn <- mvrnorm(1000, mu = c(0, 1), Sigma = matrix(c(1, -0.9,
    -0.9, 2), 2))
bivn.kde <- kde2d(bivn[, 1], bivn[, 2], n = 50)
persp(bivn.kde, phi = 10, theta = 0, shade = 0.2, border = NA,
    main = "Simulated bivariate normal density")
```

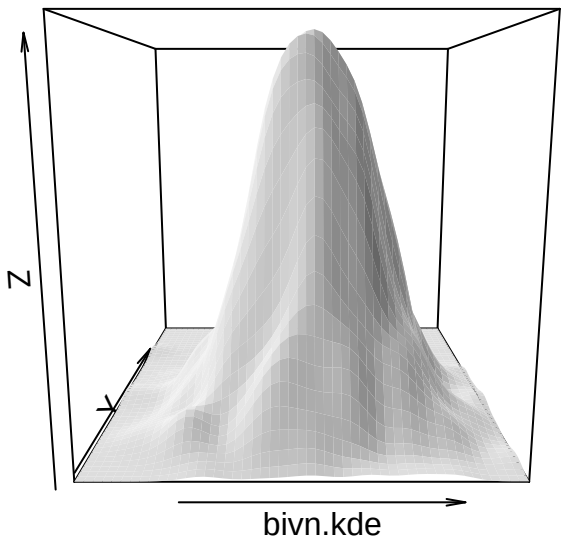


Figure 4: Visualization of two uncorrelated random variables.

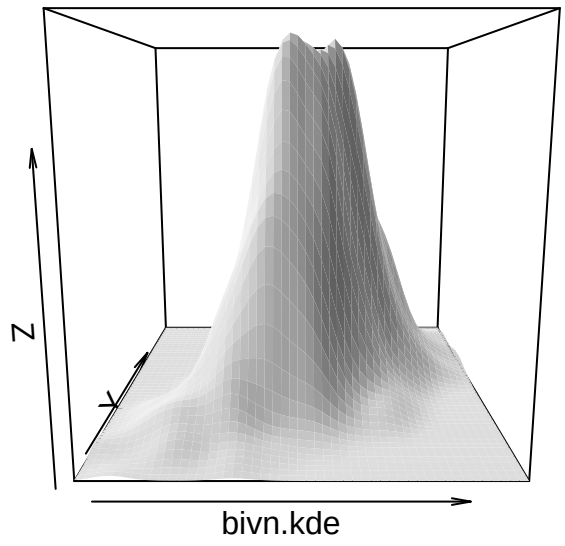


Figure 5: Visualization of two correlated random variables.

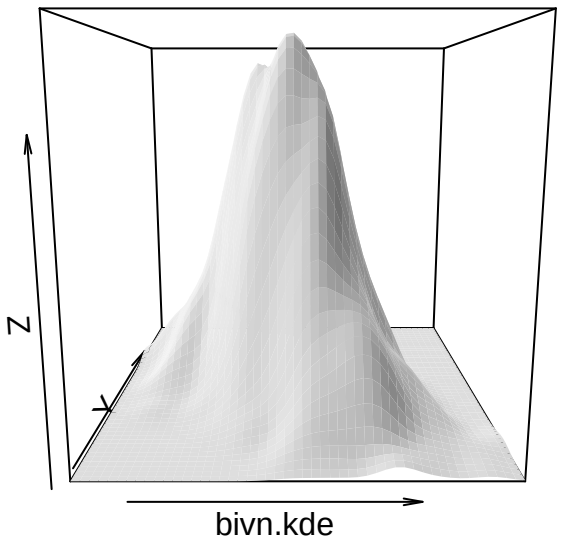


Figure 6: Visualization of two negatively correlated random variables.

Visualizing conditional distributions

You can run the following code to get a visualization of what a conditional distribution looks like when we take "slices" from the conditioning random variable:

```
for (i in 1:50) {
    plot(bivn.kde$z[i, 1:50], type = "l", ylim = c(0, 0.1))
    Sys.sleep(0.5)
}
```

1.6 Maximum likelihood estimation

Here, we look at the sample values and then choose as our estimates of the unknown parameters the values for which the probability or probability density of getting the sample values is a maximum.

1.6.1 Discrete case

Suppose the observed sample values are $x_1, x_2, \ldots, x_n$. The probability of getting them is

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = f(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n; \theta) \tag{123}$$

i.e., the function f is the value of the joint probability distribution of the random variables $X_1, \ldots, X_n$ at $X_1 = x_1, \ldots, X_n = x_n$. Since the sample values have been observed and are fixed, $f(x_1, \ldots, x_n; \theta)$ is a function of θ. The function f is called a likelihood function.

1.6.2 Continuous case

Here, f is the joint probability density, the rest is the same as above.
Definition 3. If $x_1, x_2, \ldots, x_n$ are the values of a random sample from a population with parameter θ, the likelihood function of the sample is given by

$$L(\theta) = f(x_1, x_2, \ldots, x_n; \theta) \tag{124}$$

for values of θ within a given domain. Here, $f(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n; \theta)$ is the joint probability distribution or density of the random variables $X_1, \ldots, X_n$ at $X_1 = x_1, \ldots, X_n = x_n$.

So, the method of maximum likelihood consists of maximizing the likelihood function with respect to θ. The value of θ that maximizes the likelihood function is the MLE (maximum likelihood estimate) of θ.

1.6.3 Finding maximum likelihood estimates for different distributions

Example 1

Let X_i, $i = 1, \ldots, n$ be a random variable with PDF $f(x; \sigma) = \frac{1}{2\sigma} exp(-\frac{|x|}{\sigma})$. Find $\hat{\sigma}$, the MLE of σ.

$$L(\sigma) = \prod f(x_i; \sigma) = \frac{1}{(2\sigma)^n} exp(-\sum \frac{|\, x_i \,|}{\sigma}) \tag{125}$$

Let ℓ be log likelihood. Then:

$$\ell(x; \sigma) = \sum \left[-\log 2 - \log \sigma - \frac{|\, x_i \,|}{\sigma} \right] \tag{126}$$

Differentiating and equating to zero to find maximum:

$$\ell'(\sigma) = \sum \left[-\frac{1}{\sigma} + \frac{|\, x_i \,|}{\sigma^2} \right] = -\frac{n}{\sigma} + \frac{|\, x_i \,|}{\sigma^2} = 0 \tag{127}$$

Rearranging the above, the MLE for σ is:

$$\hat{\sigma} = \frac{\sum |\, x_i \,|}{n} \tag{128}$$

Example 2: Exponential

$$f(x;\lambda) = \lambda exp(-\lambda x) \tag{129}$$

Log likelihood:

$$\ell = n \log \lambda - \sum \lambda x_i \tag{130}$$

Differentiating:

$$\ell'(\lambda) = \frac{n}{\lambda} - \sum x_i = 0 \tag{131}$$

$$\frac{n}{\lambda} = \sum x_i \tag{132}$$

I.e.,

$$\frac{1}{\hat{\lambda}} = \frac{\sum x_i}{n} \tag{133}$$

Example 3: Poisson

$$L(\mu;x) = \prod \frac{\exp^{-\mu} \mu^{x_i}}{x_i!} \tag{134}$$

$$= \exp^{-\mu} \mu^{\sum x_i} \frac{1}{\prod x_i!} \tag{135}$$

Log lik:

$$\ell(\mu;x) = -n\mu + \sum x_i \log \mu - \sum \log y! \tag{136}$$

Differentiating:

$$\ell'(\mu) = -n + \frac{\sum x_i}{\mu} = 0 \tag{137}$$

Therefore:

$$\hat{\lambda} = \frac{\sum x_i}{n} \tag{138}$$

Example 4: Binomial

$$L(\theta) = \binom{n}{x} \theta^x (1-\theta)^{n-x} \tag{139}$$

Log lik:

$$\ell(\theta) = \log \binom{n}{x} + x \log \theta + (n-x) \log(1-\theta) \tag{140}$$

Differentiating:

$$\ell'(\theta) = \frac{x}{\theta} - \frac{n-x}{1-\theta} = 0 \tag{141}$$

Thus:

$$\hat{\theta} = \frac{x}{n} \tag{142}$$

Example 5: Normal

Let $X_1, \ldots, X_n$ constitute a random variable of size n from a normal population with mean μ and variance σ^2, find joint maximum likelihood estimates of these two parameters.

$$L(\mu; \sigma^2) = \prod N(x_i; \mu, \sigma) \tag{143}$$

$$= (\tfrac{1}{2\pi\sigma^2})^{n/2} \exp(-\tfrac{1}{2\sigma^2} \sum (x_i - \mu)^2) \tag{144}$$

$$\tag{145}$$

Taking logs and differentiating with respect to μ and σ^2, we get:

$$\hat{\mu} = \frac{1}{n} \sum x_i = \bar{x} \tag{146}$$

and

$$\hat{\sigma}^2 = \frac{1}{n} \sum (x_i - \bar{x})^2 \tag{147}$$

Example 6: Geometric

$$f(x;p) = (1-p)^{x-1}p \tag{148}$$

$$L(p) = p^n(1-p)^{\Sigma x - n} \tag{149}$$

Log lik:

$$\ell(p) = n\log p + (\sum x - n)\log(1-p) \tag{150}$$

Differentiating:

$$\ell'(p)\frac{n}{p} - \frac{\sum x - n}{1-p} = 0 \tag{151}$$

$$\hat{p} = \frac{1}{\bar{x}} \tag{152}$$

1.6.4 Visualizing likelihood and maximum log likelihood for normal

For simplicity consider the case where $N(\mu = 0, \sigma^2 = 1)$.

```
op<-par(mfrow=c(1,2),pty="s")
plot(function(x) dnorm(x,log=F), -3, 3,
      main = "Normal density",#ylim=c(0,.4),
              ylab="density",xlab="X")
abline(h=0.4)
plot(function(x) dnorm(x,log=T), -3, 3,
      main = "Normal density (log)",#ylim=c(0,.4),
              ylab="density",xlab="X")
abline(h=log(0.4))
```

1.6.5 Obtaining standard errors

Once we have found a maximum likelihood estimate, say of p in the binomial distribution, we need a way to a quantify our uncertainty about how good $\hat{p}$ is at estimating the population parameter p.

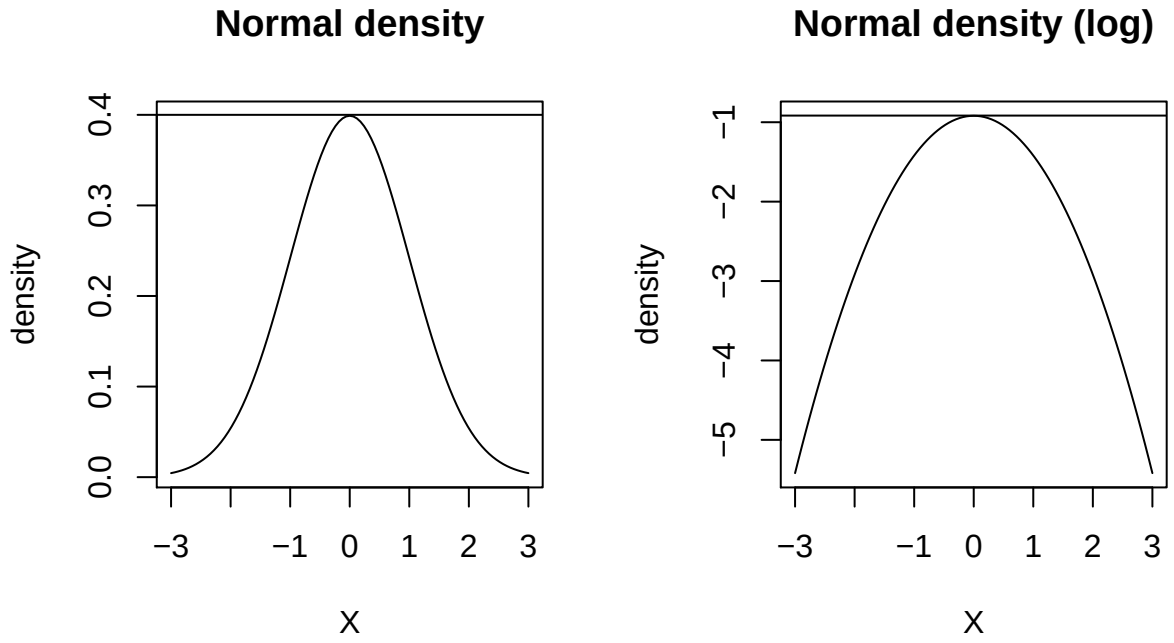


Figure 7: Maximum likelihood and log likelihood.

The second derivative of the log likelihood gives you an estimate of the variance of the sampling distribution of the sample mean (SDSM).

Here is a sort-of explanation for why the second derivative does this. The second derivative is telling us the rate at which the rate of change is happening in the slope, i.e., the rate of curvature of the curve. When the variance of the SDSM is low, then we have a sharp rate of change in slope (high value for second derivative), and so if we take the inverse of the second derivative, we get a small value, an estimate of the low variance. And when the variance is high, we have a slow rate of change in slope (slow value for second derivative), so if we invert it, we get a large value.

```
op<-par(mfrow=c(1,2),pty="s")

plot(function(x) dnorm(x,log=F,sd=0.001), -3, 3,
      main = "Normal density",#ylim=c(0,.4),
              ylab="density",xlab="X")
plot(function(x) dnorm(x,log=F,sd=10), -3, 3,
      main = "Normal density",#ylim=c(0,.4),
              ylab="density",xlab="X")
```

Notice that all these second derivatives would be negative, because we are approaching a maximum as we reach the peak of the curve. So when we take an inverse to estimate the

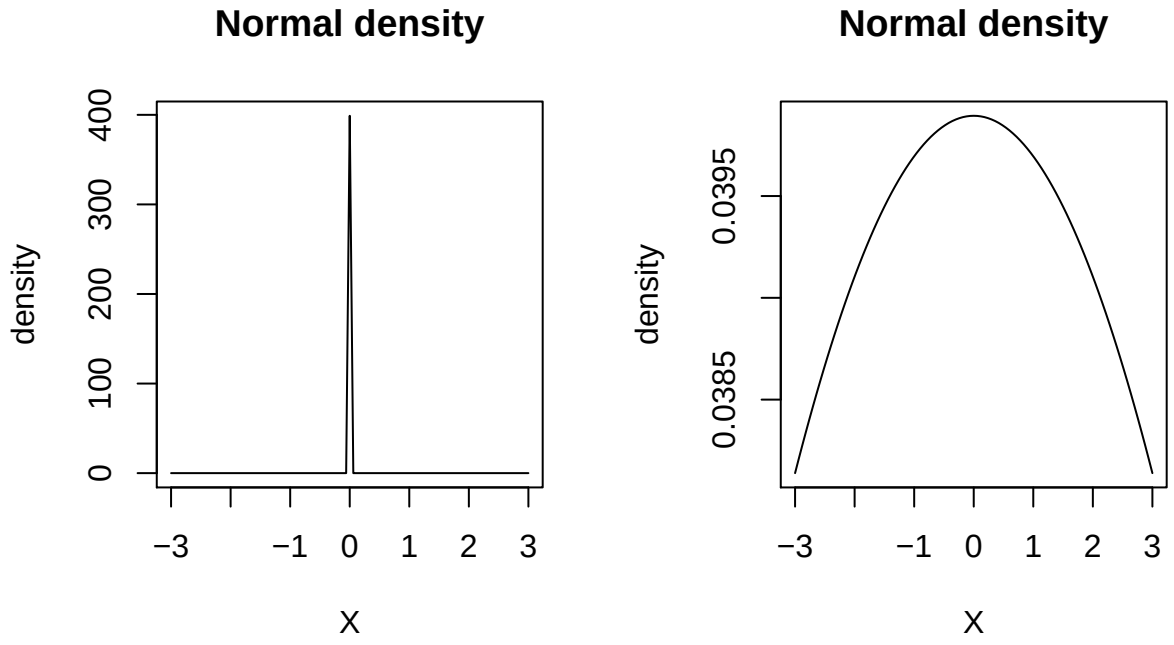


Figure 8: How variance relates to the second derivative.

variances, we get negative values. It follows that if we were to take a negative of the inverse, we'd get a positive value.

This is the reasoning that leads to the following steps for computing the variance of the SDSM:

1. Take the second partial derivative of the log-likelihood.
2. Compute the negative of the expectation of the second partial derivative. This is called the Information Matrix $I(\theta)$.
3. Invert this matrix to obtain estimates of the variances and covariances. To get standard errors take the square root of the diagonal elements in the matrix.

It's better to see this through an example:

Example 1: Binomial

Instead of calling the parameter θ I will call it p.

$$L(p) = \binom{n}{x} p^x (1-p)^{n-x} \tag{153}$$

Log lik:

$$\ell(p) = \log \binom{n}{x} + x \log p + (n-x) \log(1-p) \tag{154}$$

Differentiating:

$$\ell'(p) = \frac{x}{p} - \frac{n-x}{1-p} = 0 \tag{155}$$

Taking the second partial derivative with respect to p:

$$\ell''(p) = -\frac{x}{p^2} - \frac{n-x}{(1-p)^2} \tag{156}$$

The quantity $-\ell''(p)$ is called observed Fisher information.

Taking expectations:

$$E(\ell''(p)) = E(-\frac{x}{p^2} - \frac{n-x}{(1-p)^2}) \tag{157}$$

Exploiting that fact the $E(x/n) = p$ and so $E(x) = E(n \times x/n) = np$, we get

$$E(\ell''(p)) = E(-\frac{x}{p^2} - \frac{n-x}{(1-p)^2}) = -\frac{np}{p^2} - \frac{n-np}{(1-p)^2} = -\frac{n}{p(1-p)} \tag{158}$$

Next, we negate and invert the expectation:

$$-\frac{1}{E(\ell''(\theta))} = \frac{p(1-p)}{n} \tag{159}$$

Evaluating this at $\hat{p}$, the estimated value of the parameter, we get:

$$-\frac{1}{E(\ell''(\theta))} = \frac{\hat{p}(1-\hat{p})}{n} = \frac{1}{I(p)} \tag{160}$$

[Here, $I(p)$ is called expected Fisher Information.]

If we take the square root of the inverse Information Matrix

$$\sqrt{\frac{1}{I(p)}} = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \tag{161}$$

we have the estimated standard error.

Another example using the normal distribution:

Example 2: Normal distribution

This example is based on Khuri[?] (p. 309). Let $X_1, \ldots, X_n$ be a sample of size n from $N(\mu, \sigma^2)$, both parameters of the normal unknown.

$$L(x \mid \mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp[-\frac{1}{2\sigma^2} \sum (x-\mu)^2] \tag{162}$$

Taking log likelihood:

$$\ell = -\frac{n}{2} \log \frac{1}{(2\pi\sigma^2)} - \frac{1}{2\sigma^2} \sum (x-\mu)^2 \tag{163}$$

Taking partial derivatives with respect to μ and σ^2 we have:

$$\frac{\partial \ell}{\partial \mu} = \frac{1}{\sigma^2} \sum (x-\mu) = 0 \Rightarrow n(\bar{x}-\mu) = 0 \tag{164}$$

$$\frac{\partial \ell}{\partial \sigma^2} = \frac{1}{2\sigma^4} \sum (x-\mu)^2 - \frac{n}{2\sigma^2} = 0 \Rightarrow \sum (x-\mu)^2 - n\sigma^2 = 0 \tag{165}$$

Simplifying we get the maximum likelihood estimates of μ and σ^2: $\hat{\mu} = \bar{x}$ and $\hat{\sigma}^2 = \frac{1}{n} \sum (x-\bar{x})^2$. Note that these are unique values.

We can verify that $\hat{\mu}$ and $\hat{\sigma}^2$ are the values of μ and σ^2 that maximize $L(x \mid \mu, \sigma^2)$. This can be done by taking the second order partial derivatives and finding out whether we are at a maxima or not. It is convenient to write the four partial derivatives in the above example as a matrix, and this matrix is called a Hessian matrix. If this matrix is positive definite (i.e., if the determinant[4] of the matrix is greater than 0), we are at a maximum.

The Hessian is also going to
lead us to the information matrix as in the previous example: we just take the negative of the expectation of the Hessian, and invert it to get the variance covariance matrix. (This is just like in the binomial example above, except that we have two parameters to worry about rather than one.)[5]

[4] Suppose a a matrix represents a system of linear equations, as happens in linear modeling. A determinant of a matrix tells us whether there is a unique solution to this system of equations; when the determinant is non-zero, there is a unique solution. Given a matrix

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

the determinant is $ad - bc$. In this course, we don't need to know much about the determinant. This is the only place in this course that this term turns up.

[5] The inverse of a matrix:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

Consider the Hessian matrix H of the second partial derivatives of the log likelihood ℓ.

$$H = \begin{pmatrix} \frac{\partial^2 \ell}{\partial \mu^2} & \frac{\partial^2 \ell}{\partial \mu \partial \sigma^2} \\ \frac{\partial^2 \ell}{\partial \mu \partial \sigma^2} & \frac{\partial^2 \ell}{\partial \sigma^4} \end{pmatrix} \tag{166}$$

Now, if we compute the second-order partial derivatives replacing μ with $\hat{\mu}$ and σ^2 with $\hat{\sigma}^2$ (i.e., the values that we claim are the MLEs of the respective parameters), we will get:

$$\frac{\partial^2 \ell}{\partial \mu^2} = -\frac{n}{\hat{\sigma}^2} \tag{167}$$

$$\frac{\partial^2 \ell}{\partial \mu \partial \sigma^2} = -\frac{1}{\hat{\sigma}^2} \sum (x - \hat{\mu}) = 0 \tag{168}$$

$$\frac{\partial^2 \ell}{\partial \sigma^4} = -\frac{n}{2\hat{\sigma}^2} \tag{169}$$

The determinant of the Hessian is $\frac{n^2}{2\hat{\sigma}^6} > 0$. Hence, $(\hat{\mu}, \hat{\sigma}^2)$ is a point of local maximum of ℓ. Since it's the only maximum (we established that when we took the first derivative), it must also be the absolute maximum.

As mentioned above, if we take the negation of the expectation of the Hessian, we get the Information Matrix, and if we invert the Information Matrix, we get the variance-covariance matrix.

Once we take the negation of the expectation, we get ($\theta = (\mu, \sigma^2)$):

$$I(\theta) = \begin{pmatrix} \frac{n}{\sigma^2} & 0 \\ 0 & \frac{n}{2\sigma^4} \end{pmatrix} \tag{170}$$

Finally, if we take the inverse and evaluate it at the MLEs, we will get:

$$\frac{1}{I(\theta)} = \begin{pmatrix} \frac{\hat{\sigma}^2}{n} & 0 \\ 0 & \frac{2\hat{\sigma}^4}{n} \end{pmatrix} \tag{171}$$

And finally, if we take the square root of each element in the matrix, we get the estimated standard error of $\hat{\mu}$ to be $\frac{\hat{\sigma}}{\sqrt{n}}$, and the standard error of the $\hat{\sigma}^2$ to be $\hat{\sigma}^2\sqrt{2/n}$. The estimated standard error of the sample mean should look familiar!

I know that this is heavy going; luckily for us, for our limited purposes we can always let R do the work, as I show below.

1.6.6 MLE using R

1.6.6.1 One-parameter case

Example 1: Estimating λ in the Box-Cox transform

Let's assume that there exists a λ such that

$$f_\lambda(y_i) = x_i^T\beta + \varepsilon_i \quad \varepsilon_i \sim N(0,\sigma^2) \tag{172}$$

We use maximum likelihood estimation to estimate λ. Note that

$L(\beta_\lambda, \sigma_\lambda^2, \lambda; y) \propto$

$$(\frac{1}{\sigma})^n \exp[-\frac{1}{2\sigma^2}\sum[f_\lambda(y_i) - x_i^T\beta]^2][\prod f'_\lambda(y_i)] \tag{173}$$

For fixed λ, we estimate $\hat{\beta}$ and $\hat{\sigma}^2$ in the usual MLE way, and then we turn our attention to λ:

$$L(\hat{\beta}_\lambda, \hat{\sigma}_\lambda^2, \lambda; y) = S_\lambda^{-n/2}\prod f'_\lambda(y_i) \tag{174}$$

Taking logs:

$$\ell = c - \frac{n}{2}\log S_\lambda + \sum \log f'_\lambda(y_i) \tag{175}$$

One interesting case is the Box-Cox family.The relevant paper is by Box and Cox.[?] (An interesting side-note is that one reason that Box and Cox co-wrote that paper is that they thought it would be cool to have a paper written by two people called Box and Cox. At least that's what Box wrote in his autobiography, which is actually quite interesting to read.[?])

$$f_\lambda(y) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \lambda \neq 0 \\ \log y & \lambda = 0 \end{cases} \tag{176}$$

We assume that $f_\lambda(y) \sim N(x_i^T\beta, \sigma^2)$. So we have to just estimate λ by MLE, along with β. Here is how to do it by hand:

Since $f_\lambda = \frac{y^\lambda - 1}{\lambda}$, it follows that $f'_\lambda(y) = y^{\lambda-1}$.

Now, for different λ you can figure out the log likelihoods by hand by solving this equation:

$$\ell = c - \frac{n}{2}\log S_\lambda + (\lambda - 1)\sum \log(y_i) \tag{177}$$

Next, we do this maximum likelihood estimation using R.

```
data <- rnorm(100, mean = 500, sd = 50)

bc <- function(lambda, x = data) {
    if (lambda == 0) {
        sum(log(x))
    }
    if (lambda != 0) {
        sum(log((x^lambda - 1)/lambda))
    }
}
```

```
opt.vals.default <- optimize(bc, interval = c(-2, 2))
```

Example 2: Estimating p for the binomial distribution

A second example is to try doing MLE for the binomial. Let's assume we have the result of 10 coin tosses. We know that the MLE is the number of successes divided by the sample size:

```
x <- rbinom(10, 1, prob = 0.5)
sum(x)/length(x)
```

```
## [1] 0.6
```

We will now get this number using MLE. We do it numerically to illustrate the principle. First, we define a negative log likelihood function for the binomial. Negative because the function we will use to optimize does minimization by default, so we just flip the sign on the log likelihood.

```
negllbinom <- function(p, x) {
    -sum(dbinom(x, size = 1, prob = p, log = T))
}
```

Then we run the optimization function:

```
optimize(negllbinom, interval = c(0, 1), x = x)
```

```
## $minimum
```

```
## [1] 0.6
##
## $objective
## [1] 6.73
```

Two-parameter case

Here is an example of MLE using R. Note that in this example, we could have analytically figured out the MLEs. Instead, we are doing this numerically. The advantage of the numerical approach becomes obvious when the analytical way is closed to us.

Assume that you have some data that was generated from a normal distribution, with mean 500, and standard deviation 50. Let's say you have 100 data points.

```
data <- rnorm(100, mean = 500, sd = 50)
```

Let's assume we don't know what the mean and standard deviation are. Now, of course you know how to estimate these using the standard formulas. But right now we are going to estimate them using MLE.

We first write down the negation of the log likelihood function. We take the negation because the optimization function we will be using (see below) does minimization by default, so to get the maximum with the default setting, we just change the sign.

The function `nllh.normal` takes a vector `theta` of parameter values, and a data frame `data`.

```
nllh.normal <- function(theta, data) {
    ## decompose the parameter vector to its two parameters:
    m <- theta[1]
    s <- theta[2]
    ## read in data
    x <- data
    n <- length(x)
    ## log likelihood:
    logl <- sum(dnorm(x, mean = m, sd = s, log = TRUE))
    ## return negative log likelihood:
    -logl
}
```

Here is the negative log lik for mean = 40, sd 4, and for mean = 800 and sd 4:

```
nllh.normal(theta = c(40, 4), data)
```

```
## [1] 664900
```

```
nllh.normal(theta = c(800, 4), data)
```

```
## [1] 293684
```

As we would expect, the negative log lik for mean 500 and sd 50 is much smaller (due to the sign change) than the two log liks above:

```
nllh.normal(theta = c(500, 50), data)
```

```
## [1] 539
```

Basically, you could sit here forever, playing with combinations of values for mean and sd to find the combination that gives the optimal log likelihood. R has an optimization function that does this for you. We have to specify some sensible starting values:

```
opt.vals.default <- optim(theta <- c(700, 40), nllh.normal,
    data = data, hessian = TRUE)
```

Finally, we print out the estimated parameter values that maximize the likelihood:

```
(estimates.default <- opt.vals.default$par)
```

```
## [1] 498.2  52.8
```

And we compute standard errors by taking the square root of the diagonals of the Hessian computed by the optim function:

```
(SEs.default <- sqrt(diag(solve(opt.vals.default$hessian))))
```

```
## [1] 5.28 3.73
```

These values match our standard calculations:

```
### compute estimated standard error from data:
sd(data)/sqrt(100)
```

```
## [1] 5.31
```

1.6.6.2 Using optim in the one-parameter case

Note that we could not get the Hessian in the one-parameter case using optimize (for the binomial). We can get the 1×1 Hessian by using optim in the binomial case, but we have

to make sure that our syntax is correct.

Note that optim takes a vector of parameter names. That's the key insight. One other minor detail (for us) is that the optimization method has to be specified when using optim for one parameter optimization.

```
negllbinom2 <- function(theta) {
    p <- theta[1]
    -sum(dbinom(x, size = 1, prob = p, log = T))
}

### testing:
negllbinom2(c(0.1))
```

```
## [1] 14.2
```

```
opt.vals.default <- optim(theta <- c(0.5), negllbinom2, method = "Brent",
    hessian = TRUE, lower = 0, upper = 1)

### SE:
sqrt(solve(opt.vals.default$hessian))
```

```
##       [,1]
## [1,] 0.155
```

The estimated SE matches our analytical result earlier:

$$\sqrt{\frac{1}{I(p)}} = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \tag{178}$$

This calculation and the number that the optimization procedure delivers coincide, up to rounding error:

```
sqrt((0.4 * (1 - 0.4))/10)
```

```
## [1] 0.155
```

1.7 Introduction to Bayesian data analysis

Recall Bayes' rule:

Theorem 2. Bayes' Rule. Let B_1, B_2, ..., B_n be mutually exclusive and exhaustive and let A be an event with $\mathbb{P}(A) > 0$. Then

$$\mathbb{P}(B_k|A) = \frac{\mathbb{P}(B_k)\mathbb{P}(A|B_k)}{\sum_{i=1}^{n}\mathbb{P}(B_i)\mathbb{P}(A|B_i)}, \quad k = 1,2,\ldots,n. \tag{179}$$

When A and B are observable events, we can state the rule as follows:

$$p(A \mid B) = \frac{p(B \mid A)p(A)}{p(B)} \tag{180}$$

Note that $p(\cdot)$ is the probability of an event.

When looking at probability distributions, we will encounter the rule in the following form.

$$f(\theta \mid \text{data}) = \frac{f(\text{data} \mid \theta)f(\theta)}{f(y)} \tag{181}$$

Here, $f(\cdot)$ is a probability density, not the probability of a single event. $f(y)$ is called a "normalizing constant", which makes the left-hand side a probability distribution.

$$f(y) = \int f(x,\theta)\,d\theta = \int f(y \mid \theta)f(\theta)\,d\theta \tag{182}$$

If θ is a discrete random variable taking one value from the set $\{\theta_1,\ldots,\theta_n\}$, then

$$f(y) = \sum_{i=1}^{n} f(y \mid \theta_i)P(\theta = \theta_i) \tag{183}$$

Without the normalizing constant, we have the relationship:

$$f(\theta \mid \text{data}) \propto f(\text{data} \mid \theta)f(\theta) \tag{184}$$

Note that the likelihood $L(\theta;\text{data})$ (our data is fixed) is proportional to $f(\text{data} \mid \theta)$, and that's why we can refer to $f(\text{data} \mid \theta)$ as the likelihood in the following Bayesian incantation:

$$\text{Posterior} \propto \text{Likelihood} \times \text{Prior} \tag{185}$$

Our central goal is going to be to derive the posterior distribution and then summarize its properties (mean, median, 95% credible interval, etc.). As Christiansen et al say on p. 31 of their book, "To a Bayesian, the best information one can ever have about θ is to know the posterior density."

Usually, we don't need the normalizing constant to understand the properties of the posterior distribution. That's why Bayes' theorem is often stated in terms of the proportionality shown above.

Incidentally, this is supposed to be the moment of great divide between frequentists and Bayesians: the latter assign a probability distribution to the parameter, the former treat the parameter as a point value.

Two examples will clarify how we can use Bayes' rule to obtain the posterior. Both examples involve so-called conjugate priors, which are defined as follows:

> Given the likelihood $f(x \mid \theta)$, if the prior $f(\theta)$ results in a posterior $f(\theta \mid x)$ that has the same form as $f(\theta)$, then we call $f(\theta)$ a conjugate prior.

1.7.1 Example 1: Binomial Likelihood, Beta prior, Beta posterior

This is a contrived example, just meant to provide us with an entry point into Bayesian data analysis. Suppose that an individual with aphasia answered 46 out of 100 questions correctly in a particular sentence comprehension task. The research question is, what is the probability that their average response is greater than 0.5, i.e., above chance.

The likelihood function will tell us $P(\text{data} \mid \theta)$:

```
dbinom(46, 100, 0.5)
```

```
## [1] 0.058
```

Note that

$$P(\text{data} \mid \theta) \propto \theta^{46}(1-\theta)^{54} \tag{186}$$

So, to get the posterior, we just need to work out a prior distribution $f(\theta)$. Note that we use θ and p interchangeably when talking about the probability parameter for the binomial. We will correct this in a future edition of these notes. <–! to-do–>

$$f(\theta \mid \text{data}) \propto f(\text{data} \mid \theta) f(\theta) \tag{187}$$

For the prior, we need a distribution that can represent our uncertainty about the probabiliy p of success. The Beta distribution (a generalization of the continuous uniform distribution) is commonly used as prior for proportions. We say that the Beta distribution is conjugate to the binomial density; i.e., the two densities have similar functional forms.

The pdf is[6]

$$f(x) = \begin{cases} \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1} & \text{if } 0 < x < 1 \\ 0 & \text{otherwise} \end{cases}$$

where

$$B(a,b) = \int_0^1 x^{a-1}(1-x)^{b-1}\,dx$$

In R, we write $X \sim \texttt{beta}(\texttt{shape1} = \alpha, \texttt{shape2} = \beta)$. The associated R function is `dbeta(x, shape1, shape2)`.

The mean and variance are

$$E[X] = \frac{a}{a+b} \text{ and } Var(X) = \frac{ab}{(a+b)^2(a+b+1)}. \tag{189}$$

The Beta distribution's parameters a and b can be interpreted as (our beliefs about) prior successes and failures, and are called hyperparameters. Once we choose values for a and b, we can plot the Beta pdf. Here, I show the Beta pdf for three sets of values of a,b:

```
plot(function(x) dbeta(x, shape1 = 1, shape2 = 1), 0, 1, main = "Beta density",
    ylab = "density", xlab = "X", ylim = c(0, 3))

text(0.5, 1.1, "a=1,b=1")

plot(function(x) dbeta(x, shape1 = 3, shape2 = 3), 0, 1, add = T)
text(0.5, 1.6, "a=3,b=3")


plot(function(x) dbeta(x, shape1 = 6, shape2 = 6), 0, 1, add = T)
text(0.5, 2.6, "a=6,b=6")
```

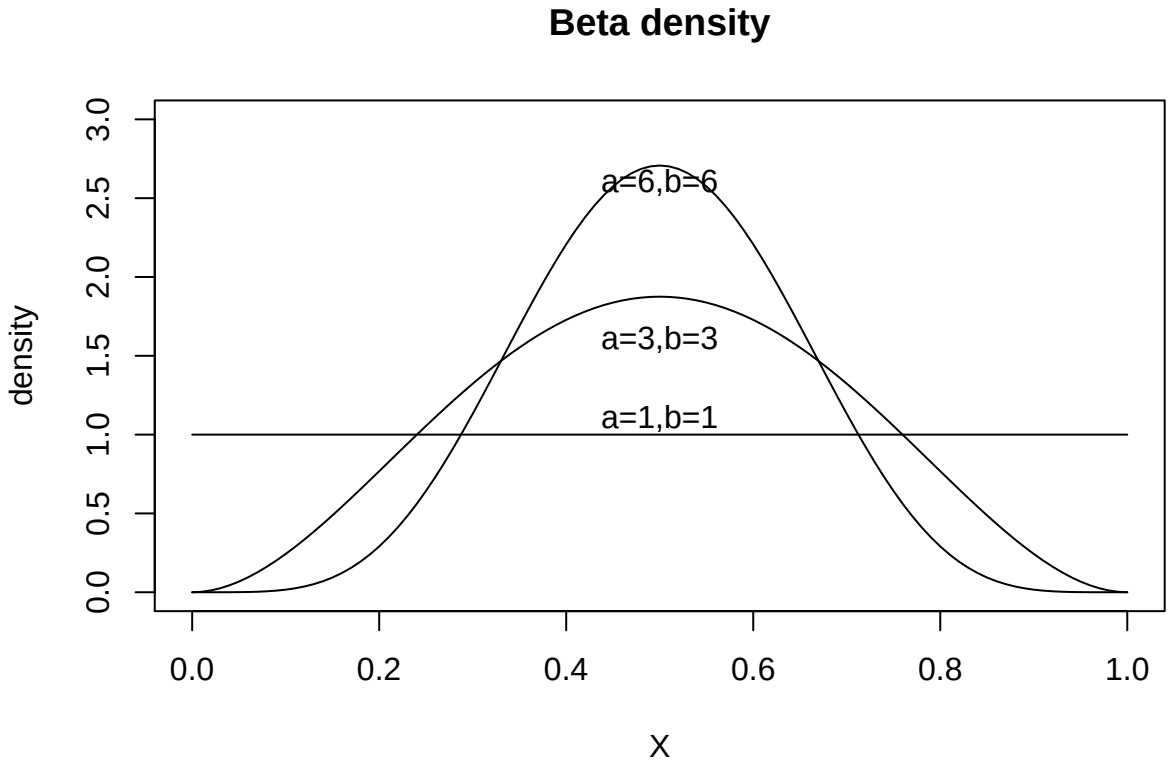
[6] Incidentally, there is a connection between the beta and the gamma:

$$B(a,b) = \int_0^1 x^{a-1}(1-x)^{b-1}\,dx = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$

which allows us to rewrite the beta PDF as

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}, \quad 0 < x < 1. \tag{188}$$

Here, x refers to the probability p.



As the figure shows, as the a,b values are increased, the spread decreases.

If we don't have much prior information, we could use a=b=1; this gives us a uniform prior; this is called an uninformative prior or non-informative prior (although having no prior knowledge is, strictly speaking, not uninformative). If we have a lot of prior knowledge and/or a strong belief that p has a particular value, we can use a larger a,b to reflect our greater certainty about the parameter. Notice that the larger our parameters a and b, the narrower the spread of the distribution; this makes sense because a larger sample size (a greater number of successes a, and a greater number of failures b) will lead to more precise estimates.

The central point is that the Beta distribution can be used to define the prior distribution of p.

Just for the sake of argument, let's take four different beta priors, each reflecting increasing certainty.

1. Beta(a=2,b=2)
2. Beta(a=3,b=3)
3. Beta(a=6,b=6)
4. Beta(a=21,b=21)

Each reflects a belief that p=0.5, with varying degrees of (un)certainty. Now we just need to plug in the likelihood and the prior:

$$f(\theta \mid \text{data}) \propto f(\text{data} \mid \theta) f(\theta) \tag{190}$$

The four corresponding posterior distributiosn would be (I hope I got the sums right):

$$f(\theta \mid \text{data}) \propto [p^{46}(1-p)^{54}][p^{2-1}(1-p)^{2-1}] = p^{47}(1-p)^{55} \tag{191}$$

$$f(\theta \mid \text{data}) \propto [p^{46}(1-p)^{54}][p^{3-1}(1-p)^{3-1}] = p^{48}(1-p)^{56} \tag{192}$$

$$f(\theta \mid \text{data}) \propto [p^{46}(1-p)^{54}][p^{6-1}(1-p)^{6-1}] = p^{51}(1-p)^{59} \tag{193}$$

$$f(\theta \mid \text{data}) \propto [p^{46}(1-p)^{54}][p^{21-1}(1-p)^{21-1}] = p^{66}(1-p)^{74} \tag{194}$$

We can now visualize each of these triplets of priors, likelihoods and posteriors. Note that I use the beta to model the likelihood because this allows me to visualize all three (prior, lik., posterior) in the same plot. The likelihood function is actually:

```
theta = seq(0, 1, by = 0.01)

plot(theta, dbinom(x = 46, size = 100, prob = theta), type = "l",
    main = "Likelihood")
```

We can represent the likelihood in terms of the beta as well:

```
plot(function(x) dbeta(x, shape1 = 46, shape2 = 54), 0, 1, ylab = "",
    xlab = "X")
```

1.7.2 Example 2: Poisson Likelihood, Gamma prior, Gamma posterior

This is also a contrived example. Suppose we are modeling the number of times that a speaker says the word "the" per day.

The number of times x that the word is uttered in one day can be modeled by a Poisson distribution:

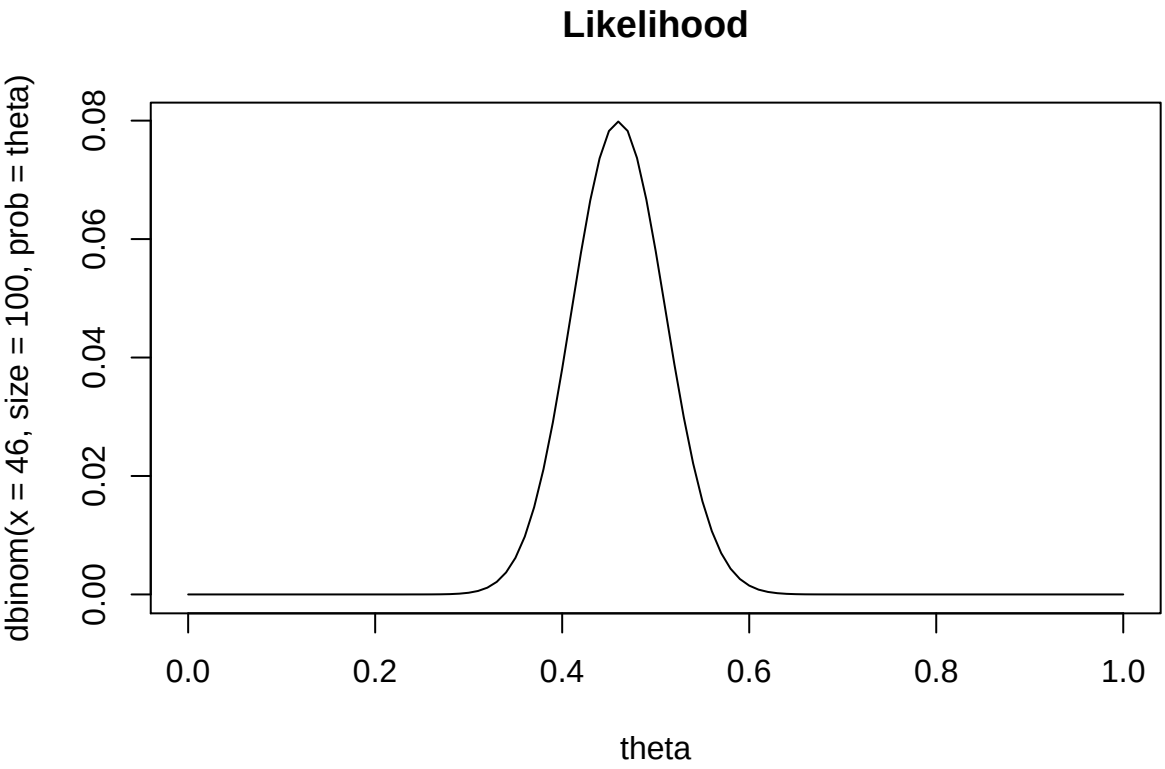


Figure 9: Binomial likelihood function.

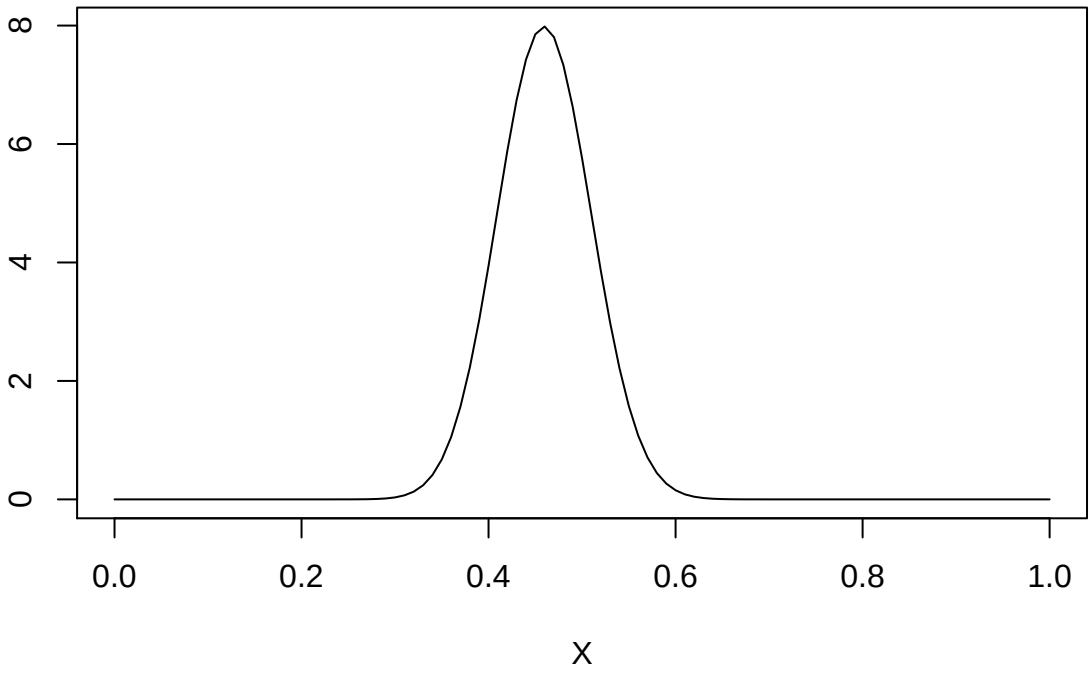


Figure 10: Using the beta distribution to represent a binomial.

$$f(x \mid \theta) = \frac{\exp(-\theta)\theta^x}{x!} \tag{195}$$

where the rate θ is unknown, and the numbers of utterances of the target word on each day are independent given θ.

We are told that the prior mean of θ is 100 and prior variance for θ is 225. This information could be based on the results of previous studies on the topic.

In order to visualize the prior, we first fit a Gamma density prior for θ based on the above information.

Note that we know that for a Gamma density with parameters a, b, the mean is $\frac{a}{b}$ and the variance is $\frac{a}{b^2}$. Since we are given values for the mean and variance, we can solve for a,b, which gives us the Gamma density.

If $\frac{a}{b} = 100$ and $\frac{a}{b^2} = 225$, it follows that $a = 100 \times b = 225 \times b^2$ or $100 = 225 \times b$, i.e., $b = \frac{100}{225}$. This means that $a = \frac{100 \times 100}{225} = \frac{10000}{225}$. Therefore, the Gamma distribution for the prior is as shown below (also see Fig~11):

$$\theta \sim Gamma(\frac{10000}{225}, \frac{100}{225}) \tag{196}$$

```
x <- 0:200
plot(x, dgamma(x, 10000/225, 100/225), type = "l", lty = 1,
    main = "Gamma prior", ylab = "density", cex.lab = 2, cex.main = 2,
    cex.axis = 2)
```

Recall that a distribution for a prior is conjugate if, multiplied by the likelihood, it yields a posterior that has the distribution of the same family as the prior.

It turns out that the Gamma distribution is a conjugate prior for the Poisson distribution. For the Gamma distribution to be a conjugate prior for the Poisson, the posterior needs to have the general form of a Gamma distribution. We derive this conjugacy result below. The proof just involves very simple algebra.

Given that

$$\text{Posterior} \propto \text{Prior Likelihood} \tag{197}$$

and given that the likelihood is:

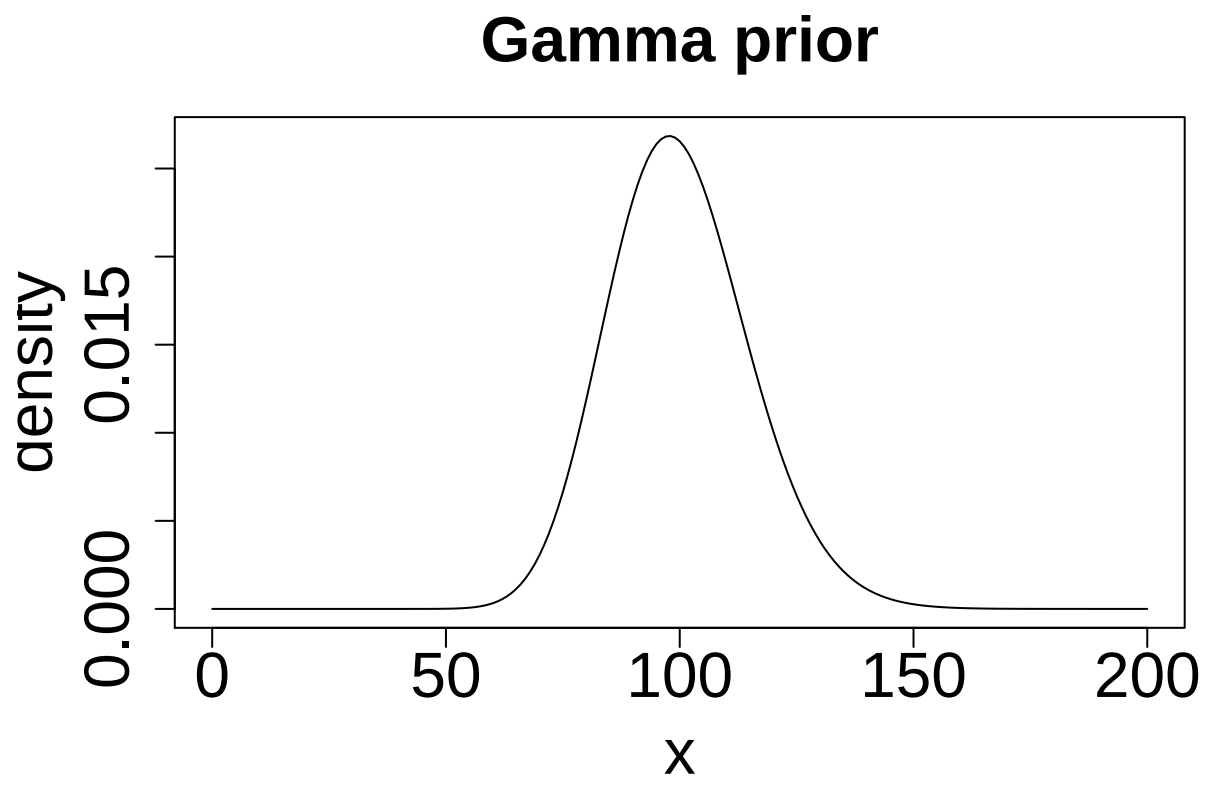


Figure 11: The Gamma prior for the parameter theta.

$$\begin{aligned} L(\mathbf{x} \mid \theta) &= \prod_{i=1}^{n} \frac{\exp(-\theta)\theta^{x_i}}{x_i!} \\ &= \frac{\exp(-n\theta)\theta^{\Sigma_i^n x_i}}{\prod_{i=1}^{n} x_i!} \end{aligned} \tag{198}$$

we can compute the posterior as follows:

$$\text{Posterior} = [\frac{\exp(-n\theta)\theta^{\Sigma_i^n x_i}}{\prod_{i=1}^{n} x_i!}][\frac{b^a \theta^{a-1}\exp(-b\theta)}{\Gamma(a)}] \tag{199}$$

Disregarding the terms $x!, \Gamma(a), b^a$, which do not involve θ, we have

$$\begin{aligned} \text{Posterior} &\propto \exp(-n\theta)\theta^{\Sigma_i^n x_i}\theta^{a-1}\exp(-b\theta) \\ &= \theta^{a-1+\Sigma_i^n x_i}\exp(-\theta(b+n)) \end{aligned} \tag{200}$$

We can now figure out the parameters of the posterior distribution, and show that it will be a Gamma distribution.

First, note that the Gamma distribution in general is $Gamma(a,b) \propto \theta^{a-1}\exp(-\theta b)$. So it's enough to state the above as a Gamma distribution with some parameters a, b.

If we equate $a^* - 1 = a - 1 + \sum_i^n x_i$ and $b^* = b + n$, we can rewrite the above as:

$$\theta^{a^*-1}\exp(-\theta b^*) \tag{201}$$

This means that $a^* = a + \sum_i^n x_i$ and $b^* = b + n$. We can find a constant k such that the above is a proper probability density function, i.e.:

$$\int_{-\infty}^{\infty} k\theta^{a^*-1}\exp(-\theta b^*) = 1 \tag{202}$$

Thus, the posterior has the form of a Gamma distribution with parameters $a^* = a + \sum_i^n x_i, b^* = b + n$. Hence the Gamma distribution is a conjugate prior for the Poisson.

1.7.2.1 Concrete example given data

Suppose we know that the number of "the" utterances is $115, 97, 79, 131$. We can derive the posterior distribution as follows.

The prior is Gamma(a=10000/225,b=100/225). The data are as given; this means that $\sum_i^n x_i = 422$ and sample size $n = 4$. It follows that the posterior is

$$\begin{aligned} Gamma(a^* = a + \sum_i^n x_i, b^* = b + n) =& Gamma(10000/225 + 422, 4 + 100/225) \\ =& Gamma(466.44, 4.44) \end{aligned} \tag{203}$$

The mean and variance of this distribution can be computed using the fact that the mean is $\frac{a*}{b*} = 466.44/4.44 = 104.95$ and the variance is $\frac{a*}{b*^2} = 466.44/4.44^2 = 23.66$.

```
### load data:
data <- c(115, 97, 79, 131)

a.star <- function(a, data) {
    return(a + sum(data))
}

b.star <- function(b, n) {
```

```
    return(b + n)
}

new.a <- a.star(10000/225, data)
new.b <- b.star(100/225, length(data))

### post. mean
post.mean <- new.a/new.b
### post. var:
post.var <- new.a/(new.b^2)

new.data <- c(200)

new.a.2 <- a.star(new.a, new.data)
new.b.2 <- b.star(new.b, length(new.data))

### new mean
new.post.mean <- new.a.2/new.b.2
### new var:
new.post.var <- new.a.2/(new.b.2^2)
```

1.7.2.2 The posterior is a weighted mean of prior and likelihood

We can express the posterior mean as a weighted sum of the prior mean and the maximum likelihood estimate of θ.

The posterior mean is:

$$\frac{a*}{b*} = \frac{a + \sum x_i}{n + b} \tag{204}$$

This can be rewritten as

$$\frac{a*}{b*} = \frac{a + n\bar{x}}{n + b} \tag{205}$$

Dividing both the numerator and denominator by b:

$$\frac{a*}{b*} = \frac{(a + n\bar{x})/b}{(n + b)/b} = \frac{a/b + n\bar{x}/b}{1 + n/b} \tag{206}$$

Since a/b is the mean m of the prior, we can rewrite this as:

$$\frac{a/b + n\bar{x}/b}{1 + n/b} = \frac{m + \frac{n}{b}\bar{x}}{1 + \frac{n}{b}} \tag{207}$$

We can rewrite this as:

$$\frac{m + \frac{n}{b}\bar{x}}{1 + \frac{n}{b}} = \frac{m \times 1}{1 + \frac{n}{b}} + \frac{\frac{n}{b}\bar{x}}{1 + \frac{n}{b}} \tag{208}$$

This is a weighted average: setting $w_1 = 1$ and $w_2 = \frac{n}{b}$, we can write the above as:

$$m\frac{w_1}{w_1 + w_2} + \bar{x}\frac{w_2}{w_1 + w_2} \tag{209}$$

A n approaches infinity, the weight on the prior mean m will tend towards 0, making the posterior mean approach the maximum likelihood estimate of the sample.

In general, in a Bayesian analysis, as sample size increases, the likelihood will dominate in determining the posterior mean.

Regarding variance, since the variance of the posterior is:

$$\frac{a*}{b*^2} = \frac{(a + n\bar{x})}{(n + b)^2} \tag{210}$$

as n approaches infinity, the posterior variance will approach zero: more data will reduce variance (uncertainty).

Summary

We saw two examples where we can do the computations to derive the posterior using simple algebra. There are several other such simple cases (see the Lynch textbook for more). However, in realistic data analysis settings, we cannot specify the posterior distribution as a particular density. For such cases, we need to use computer software so that we can sample from the posterior distribution.

1.8 Introduction to Gibbs sampling

1.8.1 Monte Carlo integration

It sounds fancy, but basically this amounts to sampling from a distribution, and computing summaries like the mean. Formally, we calculate E(f(X)) by drawing samples $\{X_1, \ldots, X_n\}$ and then approximating:

$$E(f(X)) \approx \frac{1}{n}\sum f(X) \tag{211}$$

For example:

```
x <- rnorm(1000, mean = 0, sd = 1)
mean(x)
```

```
## [1] 0.00346
```

We can increase sample size to as large as we want.

We can also compute quantities like $P(X < 1.96)$ by sampling:

```
mean((x < 1.96))
```

```
## [1] 0.97
```

```
### theoretical value:
pnorm(1.96)
```

```
## [1] 0.975
```

So, we can compute summary statistics using simulation. However, if we only know up to proportionality the form of the distribution to sample from, how do we get these samples to summarize from? Monte Carlo Markov Chain (MCMC) methods provide that capability: they allow you to sample from distributions you only know up to proportionality.

1.8.2 Markov chain sampling

We have been doing non-Markov chain sampling when we used rnorm:

```
indep.samp <- rnorm(500, mean = 0, sd = 1)
head(indep.samp)
```

```
## [1]  0.6718  2.0820  0.0191 -0.2365 -0.0699 -0.6944
```

The vector of values sampled here are statistically independent.

```
plot(1:500, indep.samp, type = "l")
```

If the current value influences the next one, we have a Markov chain. Here is a simple Markov chain: the i-th draw is dependent on the i-1 th draw:

```
nsim <- 500
x <- rep(NA, nsim)
```

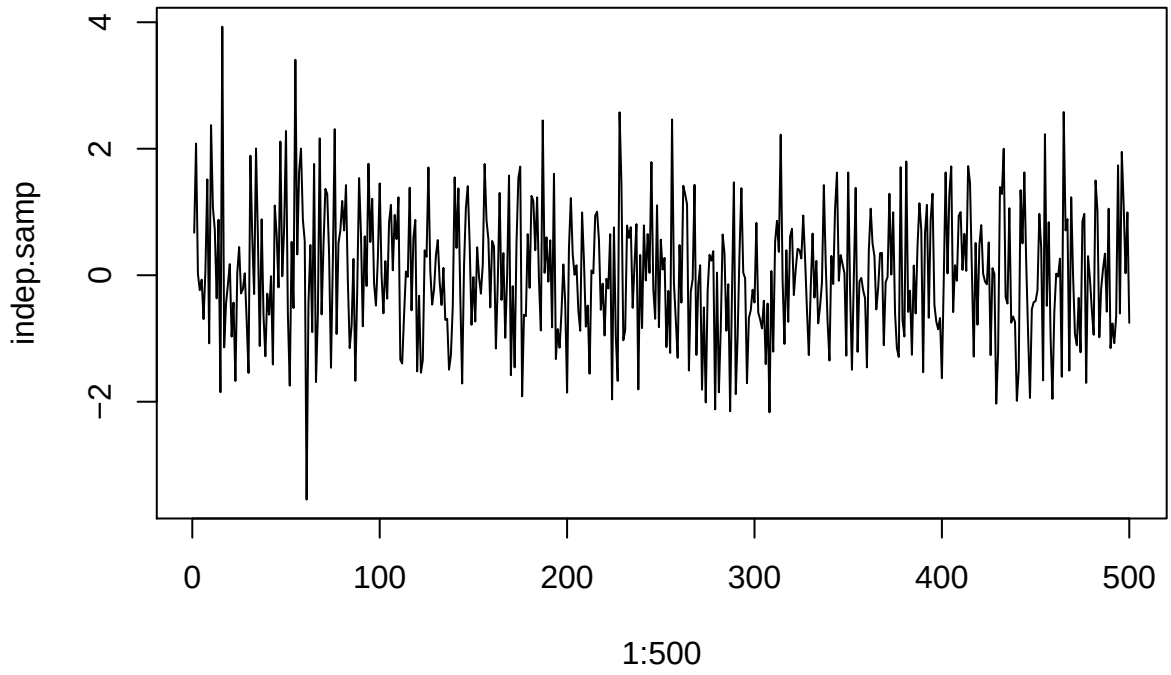


Figure 12: Example of independent samples.

```
y <- rep(NA, nsim)
x[1] <- rnorm(1)  ## initialize x
for (i in 2:nsim) {
    ### draw i-th value based on i-1-th value:
    y[i] <- rnorm(1, mean = x[i - 1], sd = 1)
    x[i] <- y[i]
}
plot(1:nsim, y, type = "l")
```

1.8.3 Monte Carlo sampling

In the example with the Beta distribution in the last section, we can sample from the posterior distribution easily using a built-in R function:

```
x <- rbeta(5000, 1498, 1519)
```

Once we have these samples, we can compute any kind of useful summary, e.g., the posterior probability (given the data) that p>0.5:

```
table(x > 0.5)[2]/sum(table(x > 0.5))
```

```
##  TRUE
```

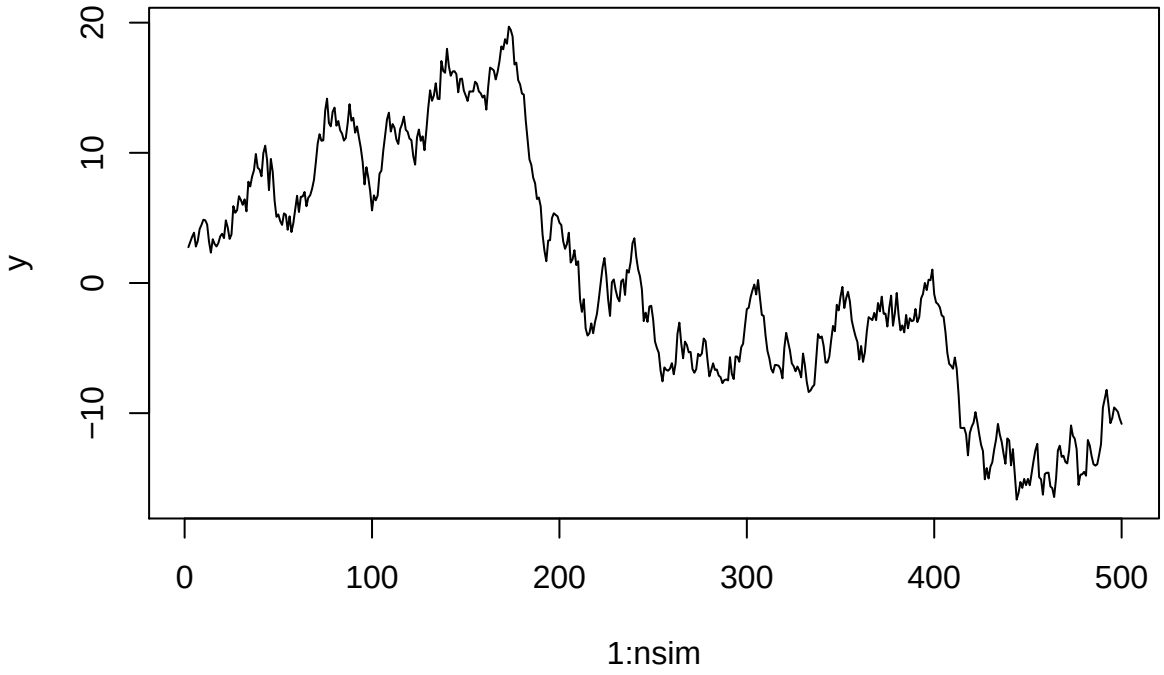


Figure 13: Example of markov chain.

```
## 0.348
```

Or we can compute a 95% interval within which we are 95% sure that the true parameter value lies (recall all the convoluted discussion about 95% CIs in the frequentist setting!):

```
### lower bound:
quantile(x, 0.025)
```

```
##  2.5%
## 0.479
```

```
### upper bound:
quantile(x, 0.975)
```

```
## 97.5%
## 0.515
```

Since we can integrate the Beta distribution analytically, we could have done the same thing with the `qbeta` function (or simply using calculus):

```
(lower <- qbeta(0.025, shape1 = 1498, shape2 = 1519))
```

```
## [1] 0.479
```

```
(upper <- qbeta(0.975, shape1 = 1498, shape2 = 1519))
```

```
## [1] 0.514
```

However—and here we finally get to a crucial point—integration of posterior densities is often impossible (e.g., because they may have many dimensions). In those situations we use sampling methods called Markov Chain Monte Carlo, or MCMC, methods.

As Lunn et al put it (p. 61), %[?] the basic idea is going to be that we sample from an approximate distribution, and then correct or adjust the values. The rest of this section elaborates on this statement.

First, let's look at two relatively simple methods of sampling.

1.8.4 The inversion method

This method works when we can know the closed form of the pdf we want to simulate from and can derive the inverse of that function.

Steps:

1. Sample one number u from $Unif(0,1)$. Let $u = F(z) = \int_L^z f(x)\,dx$ (here, L is the lower bound of the pdf f).
2. Then $z = F^{-1}(u)$ is a draw from $f(x)$.

Example: let $f(x) = \frac{1}{40}(2x+3)$, with $0 < x < 5$. We have to draw a number from the uniform distribution and then solve for z, which amounts to finding the inverse function:

$$u = \int_0^z \frac{1}{40}(2x+3) \tag{212}$$

```
u <- runif(1000, min = 0, max = 1)


z <- (1/2) * (-3 + sqrt(160 * u + 9))
```

This method can't be used if we can't find the inverse, and it can't be used with multivariate distributions.

1.8.5 The rejection method

If $F^{-1}(u)$ can't be computed, we sample from $f(x)$ as follows:

1. Sample a value z from a distribution $g(z)$ from which sampling is easy, and for which

$$mg(z) > f(z) \quad m \text{ a constant} \tag{213}$$

   $mg(z)$ is called an envelope function because it envelops $f(z)$.

2. Compute the ratio

$$R = \frac{f(z)}{mg(z)} \tag{214}$$

3. Sample $u \sim Unif(0,1)$.

4. If $R > u$, then z is treated as a draw from $f(x)$. Otherwise return to step 1.

For example, consider f(x) as above: $f(x) = \frac{1}{40}(2x+3)$, with $0 < x < 5$. The maximum height of $f(x)$ is 0.325 (why?). So we need an envelope function that exceeds 0.325. The uniform density $Unif(0,5)$ has maximum height 0.2, so if we multiply it by 2 we have maximum height 0.4, which is greater than 0.325.

In the first step, we sample a number x from a uniform distribution Unif(0,5). This serves to locate a point on the x-axis between 0 and 5 (the domain of x). The next step involves locating a point in the y direction once the x coordinate is fixed. If we draw a number u from Unif(0,1), then $mg(x)u = 2 * 0.2u$ is a number between 0 and $2 * 0.2$. If this number is less than f(x), that means that the y value falls within f(x), so we accept it, else reject. Checking whether $mg(x)u$ is less than $f(x)$ is the same as checking whether

$$R = f(x)/mg(z) > u \tag{215}$$

```
## R program for rejection method of sampling From Lynch
## book, adapted by SV.
count <- 0
k <- 1
accepted <- rep(NA, 1000)
rejected <- rep(NA, 1000)
while (k < 1001) {
    z <- runif(1, min = 0, max = 5)
    r <- ((1/40) * (2 * z + 3))/(2 * 0.2)
    if (r > runif(1, min = 0, max = 1)) {
        accepted[k] <- z
```

```
        k <- k + 1
    } else {
        rejected[k] <- z
    }
    count <- count + 1
}
```

```
hist(accepted, freq = F, main = "Example of rejection sampling")

fn <- function(x) {
    (1/40) * (2 * x + 3)
}

x <- seq(0, 5, by = 0.01)

lines(x, fn(x))
```

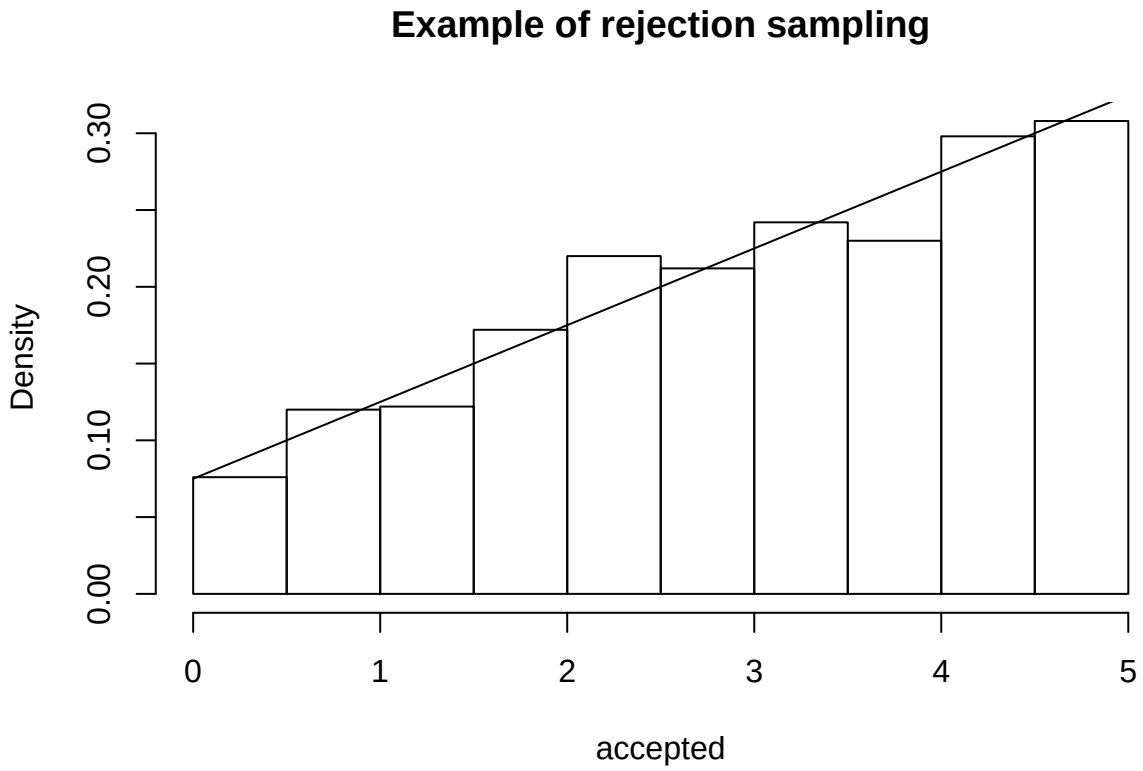


Figure 14: Example of rejection sampling.

```
### acceptance rate:
table(is.na(rejected))[2]/sum(table(is.na(rejected)))
```

```
## TRUE
## 0.48
```

Question: If you increase m, will acceptance rate increase or decrease? Stop here and come up with an answer before you read further.

Rejection sampling can be used with multivariate distributions.

Some limitations of rejection sampling: finding an envelope function may be difficult; the acceptance rate would be low if the constant m is set too high and/or if the envelope function is too high relative to f(x), making the algorithm inefficient.

1.8.6 Monte Carlo Markov Chain: The Gibbs sampling algorithm

Let Θ be a vector of parameter values, let length of Θ be k. Let j index the j-th iteration.

Algorithm:

1. Assign starting values to Θ:

   $\Theta^{j=0} \leftarrow S$

2. Set $j \leftarrow j+1$

3. 1. Sample $\theta_1^j \mid \theta_2^{j-1} \dots \theta_k^{j-1}$.
   2. Sample $\theta_2^j \mid \theta_1^j \theta_3^{j-1} \dots \theta_k^{j-1}$.

⋮

   k. Sample $\theta_k^j \mid \theta_1^j \dots \theta_{k-1}^j$.

4. Return to step 1.

Example: Consider the bivariate distribution:

$$f(x,y) = \frac{1}{28}(2x+3y+2) \tag{216}$$

We can analytically work out the conditional distributions:

$$f(x \mid y) = \frac{f(x,y)}{f(y)} = \frac{(2x+3y+2)}{6y+8} \tag{217}$$

$$f(y \mid x) = \frac{f(x,y)}{f(x)} = \frac{(2x+3y+2)}{4y+10} \tag{218}$$

The Gibbs sampler algorithm is:

1. Set starting values for the two parameters $x = -5, y = -5$. Set j=0.
2. Sample x^{j+1} from $f(x \mid y)$ using inversion sampling. You need to work out the inverse of $f(x \mid y)$ and $f(y \mid x)$ first. To do this, for $f(x \mid u)$, we have find z_1:

$$u = \int_0^{z_1} \frac{(2x+3y+2)}{6y+8}\, dx \tag{219}$$

   And for $f(y \mid x)$, we have to find z_2:

$$u = \int_0^{z_2} \frac{(2x+3y+2)}{4y+10}\, dy \tag{220}$$

   I leave that as an exercise; the solution is given in the code below.

```
## R program for Gibbs sampling using inversion method
## program by Scott Lynch, modified by SV:
x <- rep(NA, 2000)
y <- rep(NA, 2000)
x[1] <- -5
y[1] <- -5

for (i in 2:2000) {
    # sample from x | y
    u <- runif(1, min = 0, max = 1)
    x[i] <- sqrt(u * (6 * y[i - 1] + 8) + (1.5 * y[i - 1] +
        1) * (1.5 * y[i - 1] + 1)) - (1.5 * y[i - 1] + 1)
    # sample from y | x
    u <- runif(1, min = 0, max = 1)
    y[i] <- sqrt((2 * u * (4 * x[i] + 10))/3 + ((2 * x[i] +
        2)/3) * ((2 * x[i] + 2)/3)) - ((2 * x[i] + 2)/3)
}
```

You can run this code to visualize the simulated posterior distribution:

```
bivar.kde <- kde2d(x, y)
persp(bivar.kde, phi = 10, theta = 90, shade = 0, border = NA,
    main = "Simulated bivariate density using Gibbs sampling")
```

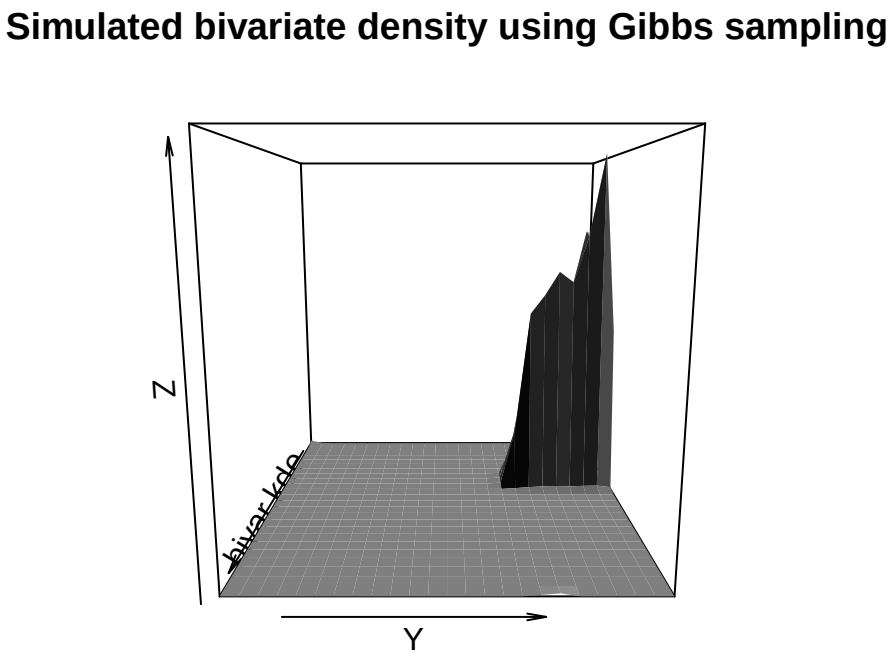


Figure 15: Example of posterior distribution of bivariate distribution.

A central insight here is that knowledge of the conditional distributions is enough to figure out (simulate from) the joint distribution, provided such a joint distribution exists.

1.8.7 Gibbs sampling using rejection sampling

Suppose the conditional distribution is not univariate–we might have conditional multivariate densities. Now we can't use inversion sampling. Another situation where we can't use inversion sampling is when $F^{-1}()$ can't be calculated even in one dimension (An example where the inversion sampling doesn't work is the bivariate normal; there is no way to compute the conditional CDF analytically).

```
## R program for Gibbs sampling using rejection sampling
## Program by Scott Lynch, modified by SV:
x <- rep(NA, 2000)
y <- rep(NA, 2000)
x[1] <- -1
y[1] <- -1
```

```
m <- 25

for (i in 2:2000) {
    ## sample from x | y using rejection sampling
    z <- 0
    while (z == 0) {
        u <- runif(1, min = 0, max = 2)
        if (((2 * u) + (3 * y[i - 1]) + 2) > (m * runif(1, min = 0,
            max = 1) * 0.5)) {
            x[i] <- u
            z <- 1
        }
    }
    # sample from y | x using z=0 while(z==0)
    z <- 0
    while (z == 0) {
        u <- runif(1, min = 0, max = 2)
        if (((2 * x[i]) + (3 * u) + 2) > (m * runif(1, min = 0,
            max = 1) * 0.5)) {
            {
                y[i] <- u
                z <- 1
            }
        }
    }
}
```

Note that we need to know the conditional densities only up to proportionality, we do not need to know the normalizing constant (see discussion in Lynch). Also, note that we need sensible starting values, otherwise the algorithm will never take off.

Visualization:

```
bivar.kde <- kde2d(x, y)
persp(bivar.kde, phi = 10, theta = 90, shade = 0, border = NA,
    main = "Simulated bivariate density using Gibbs sampling \n
```

```
      (rejection sampling)")
```

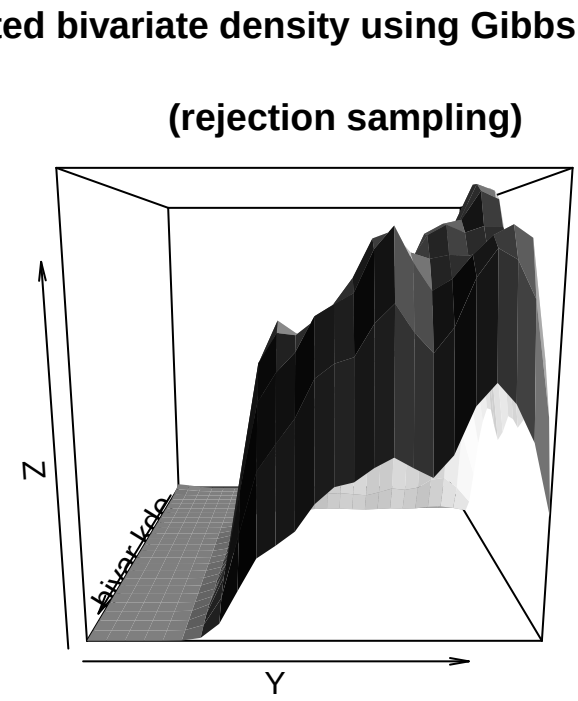


Figure 16: Sampling from posterior density, using rejection sampling.

1.8.8 More realistic example: Sampling from a bivariate density

Let's try to sample from a bivariate normal distribution using Gibbs sampling. We could have just done it with a built-in function from the `MASS` package for this (see earlier material on bivariate distribution sampling using `mvrnorm` in these notes). But it's instructive to do it "by hand".

Here, we can compute the conditional distributions but we can't compute $F^{-1}()$ analytically.

We can look up in a textbook (or derive this analytically) that the conditional distribuion $f(X \mid Y)$ in this case is:

$$f(X \mid Y) = N(\mu_x + \rho\sigma_x \frac{y - \mu_y}{\sigma_y}, \sigma_x \sqrt{(1 - \rho^2)}) \tag{221}$$

and similarly (mutatis mutandis) for $f(Y \mid X)$. Note that Lynch provides a derivation for conditional densities up to proportionality.

For simplicity we assume we have a bivariate normal, uncorrelated random variables X and Y each with mean 0 and sd 1. But you can play with all of the parameters below and see

how things change.

Here is my code:

```
### parameters:
mu.x <- 0
mu.y <- 0
sd.x <- 1
sd.y <- 1
rho <- 0

### Gibbs sampler:
x <- rep(NA, 2000)
y <- rep(NA, 2000)
x[1] <- -5
y[1] <- -5

for (i in 2:2000) {
    # sample from x | y, using (i-1)th value of y:
    u <- runif(1, min = 0, max = 1)
    x[i] <- rnorm(1, mu.x + rho * sd.x * ((y[i - 1] - mu.y)/sd.y),
        sd.x * sqrt(1 - rho^2))

    # sample from y | x, using i-th value of x
    u <- runif(1, min = 0, max = 1)
    y[i] <- rnorm(1, mu.y + rho * sd.y * ((x[i] - mu.x)/sd.x),
        sd.y * sqrt(1 - rho^2))
}
```

We can visualize this as well:

```
bivar.kde <- kde2d(x, y)
for (i in 1:100) {
    persp(bivar.kde, phi = 10, theta = i, shade = 0, border = NA,
        main = "Simulated bivariate normal density
      using Gibbs sampling")
    ## Sys.sleep(0.1)
}
```

We can plot the "trace plot" of each density, as well as the marginal density:

```
op <- par(mfrow = c(2, 2), pty = "s")
plot(1:2000, x)
hist(x, main = "Marginal density of X")
plot(1:2000, y)
hist(y, main = "Marginal density of Y")
```

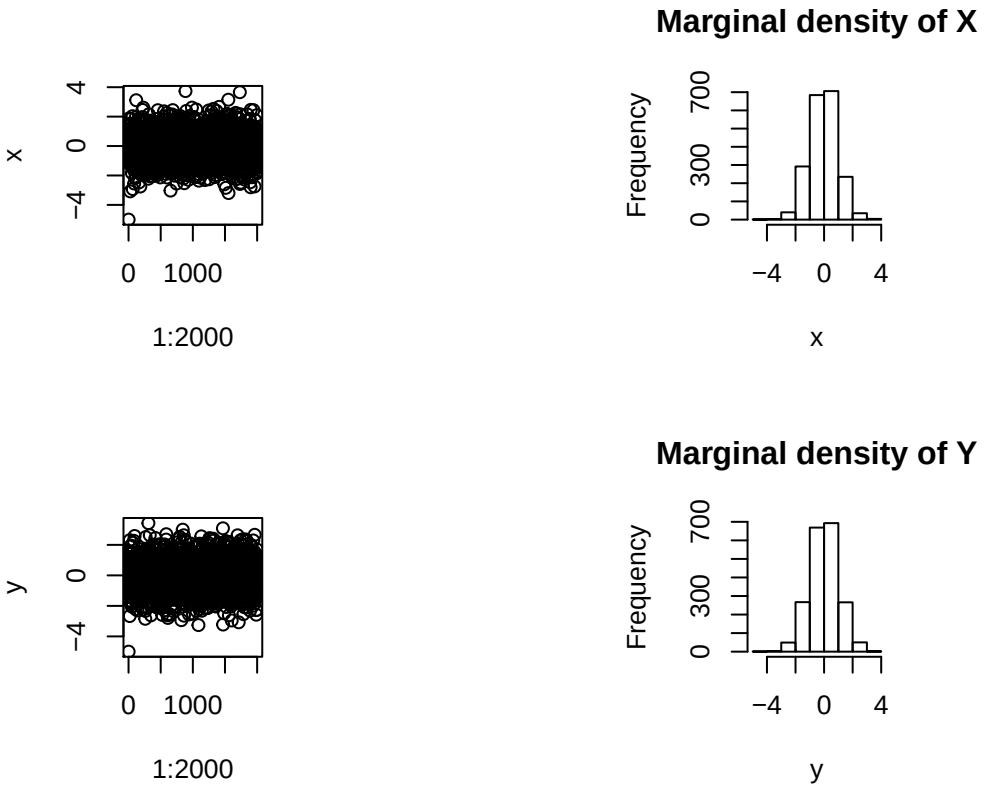


Figure 17: Trace plots.

The trace plots show the points that were sampled. The initial values (-5) were not realistic, and it could be that the first few iterations do not really yield representative samples. We discard these, and the initial period is called "burn-in" or (in Stan) warm-up.

The two dimensional trace plot traces the Markov chain walk (I don't plot it because it slows down the loading of the pdf):

We can also summarize the marginal distributions. One way is to compute the 95% highest posterior density interval (now you know why it's called that), and the mean or median.

```
quantile(x, 0.025)
```

```
##  2.5%
```

```
## -1.95
```

```
quantile(x, 0.975)
```

```
## 97.5%
##  1.87
```

```
mean(x)
```

```
## [1] -0.0447
```

```
quantile(y, 0.025)
```

```
##  2.5%
## -2.02
```

```
quantile(y, 0.975)
```

```
## 97.5%
##  2.01
```

```
mean(y)
```

```
## [1] 0.00193
```

These numbers match up pretty well with the theoretical values (which we know since we sampled from a bivariate with known means and sds; see above).

If we discard the first 500 runs as burn-in or warm-up, we get:

```
quantile(x[501:2000], 0.025)
```

```
##  2.5%
## -1.91
```

```
quantile(x[501:2000], 0.975)
```

```
## 97.5%
##  1.92
```

```
mean(x[501:2000])
```

```
## [1] -0.0288
```

```
quantile(y[501:2000], 0.025)
```

```
##  2.5%
## -2.02
```

```
quantile(y[501:2000], 0.975)
```

```
## 97.5%
##  2.03
```

```
mean(y[501:2000])
```

```
## [1] -0.000563
```

Here is Scott Lynch's code, with conditional distributions specified up to proportionality:

```
## R program for Gibbs sampling from a bivariate normal pdf
x <- rep(NA, 2000)
y <- rep(NA, 2000)
for (j in 2:2000) {
    ## sampling from x|y
    x[j] = rnorm(1, mean = (0.5 * y[j - 1]), sd = sqrt(1 - 0.5 *
        0.5))
    # sampling from y|x
    y[j] = rnorm(1, mean = (0.5 * x[j]), sd = sqrt(1 - 0.5 *
        0.5))
}
```

1.8.9 Sampling the parameters given the data

The Bayesian approach is that, conditional on having data, we want to sample parameters. For this, we need to figure out the conditional density of the parameters given the data.

The marginal for σ^2 happens to be:

$$p(\sigma^2 \mid X) \propto InverseGamma((n-1)/2, (n-1)var(x)/2) \tag{222}$$

The conditional distribution:

$$p(\sigma^2 \mid \mu, X) \propto InverseGamma(n/2, \sum (x_i - \mu)^2/2) \tag{223}$$

$$p(\mu \mid \sigma^2, \mid X) \propto N(\bar{x}, \sigma^2/n) \tag{224}$$

Now we can do Gibbs sampling on parameters given data. There are two ways, given the above equations:

1. Given the marginal distribution of σ^2, sample a vector of values for σ^2 and then sample μ conditional on each value of σ^2 from μ's conditional distribution. This approach is more efficient.
2. First sample a value for σ^2 conditional on μ, then sample a value of μ conditional on the new value for σ^2, etc.

1.8.9.1 Example of first approach

```
## R: sampling from marginal for variance and conditional for
## mean code by Scott Lynch, slightly modified by SV:
x <- as.matrix(read.table("data/education.dat", header = F)[,
    1])
sig2 <- rgamma(2000, (length(x) - 1)/2, rate = ((length(x) -
    1) * var(x)/2))
sig2 <- 1/sig2
mu <- rnorm(2000, mean = mean(x), sd = (sqrt(sig2/length(x))))
```

Note that we draw from a Gamma, and then invert to get an inverse Gamma.

Visualization:

```
bivar.kde <- kde2d(sig2, mu)
persp(bivar.kde, phi = 10, theta = 90, shade = 0, border = NA,
    main = "Simulated bivariate density using
      Gibbs sampling\n
      first approach (using marginals)")
```

```
op <- par(mfrow = c(1, 2), pty = "s")
hist(mu)
hist(sig2)
```

1.8.9.2 Example of second approach

Here, we sample μ and σ^2 sequentially from their conditional distributions.

```
## R: sampling from conditionals for both variance and mean
x <- as.matrix(read.table("data/education.dat", header = F)[,
    1])
mu <- rep(0, 2000)
```

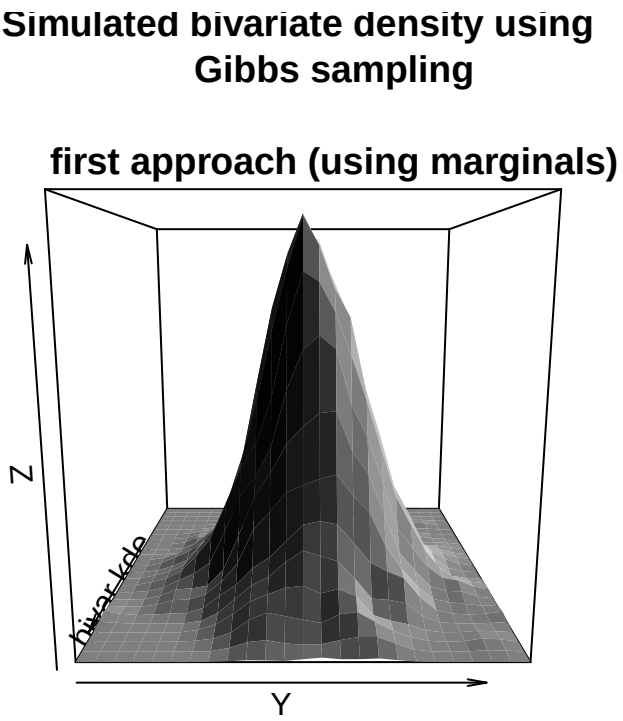


Figure 18: Gibbs sampling using marginals.

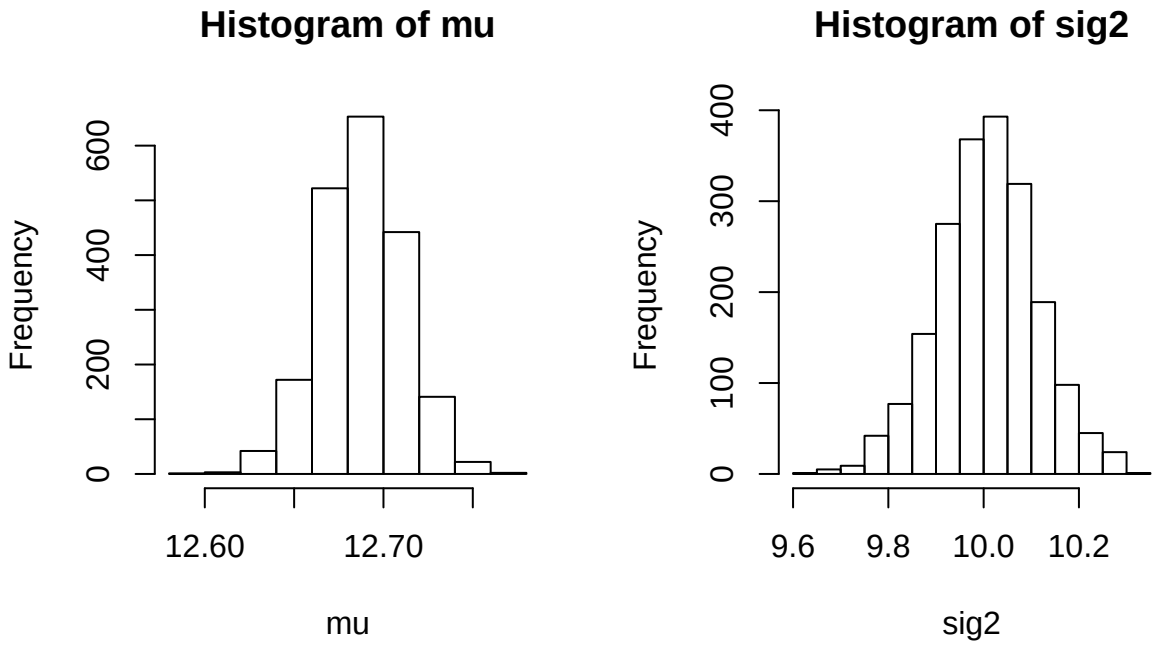


Figure 19: Marginal posterior distributions for mu and sigma squared.

```
sig2 <- rep(1, 2000)
for (i in 2:2000) {
    sig2[i] <- rgamma(1, (length(x)/2), rate = sum((x - mu[i -
        1])^2)/2)
    sig2[i] <- 1/sig2[i]
    mu[i] <- rnorm(1, mean = mean(x), sd = sqrt(sig2[i]/length(x)))
}
```

First, we drop the burn-in/warm-up values (in the conditionals sampling approach, the initial values will need to be dropped).

```
mu <- mu[1001:2000]
sig2 <- sig2[1001:2000]
```

Visualization:

```
bivar.kde <- kde2d(sig2, mu)
persp(bivar.kde, phi = 10, theta = 45, shade = 0, border = NA,
    main = "Simulated bivariate density using
      Gibbs sampling\n
      second approach (using conditionals)")
```

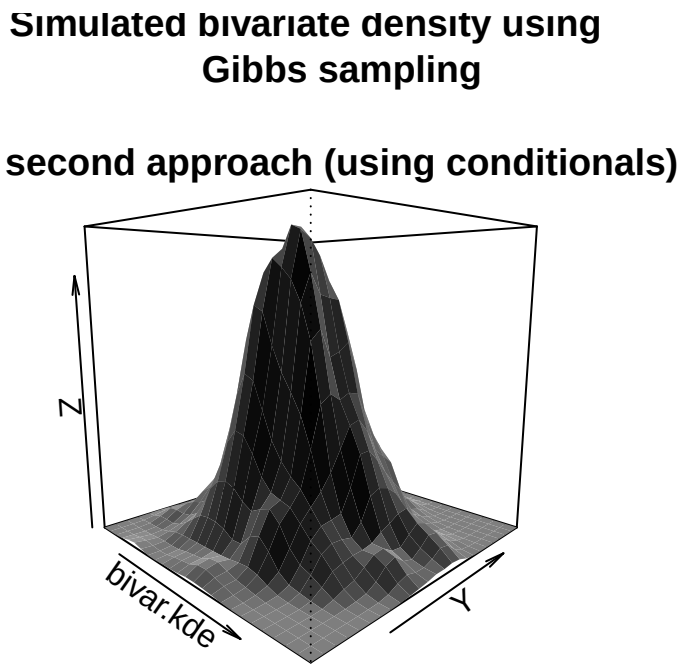


Figure 20: Posterior distribution using conditionals.

1.8.10 Summary of the story so far

This summary is taken, essentially verbatim but reworded a bit, from Lynch %[?] (pages 103-104).

1. Our main goal in the Bayesian approach is to summarize the posterior density.
2. If the posterior density can be analytically analyzed using integration, we are done.
3. If there are no closed-form solutions to the integrals, we resort to sampling from the posterior density. This is where Gibbs sampling comes in: the steps are:
   (a) Derive the relevant conditional densities: this involves (a) treating other variables as fixed (b) determining how to sample from the resulting conditional density. The conditional density either has a known form (e.g., normal) or it may take an unknown form. If it has an unknown form, we use inversion or rejection sampling in order to take samples from the conditional densities.
   (b) If inversion sampling from a conditional density is difficult or impossible, we use the Metropolis algorithm, which we discuss next.

1.9 Exercises

1. Suppose we are given that the pdf of θ, which ranges from 0 to 1, is proportional to θ^2. This means that there is some constant c (the constant of proportionality) such that $1 = c\int_0^1 \theta^2 d\theta$.
   (a) Find c.
   (b) Find the mean, median (hint: what is the median in terms of quantiles?) and variance of the above pdf.
   (c) Find the 95% credible interval, i.e., the lower and upper values in $P(lower < \theta < upper) = 0.95$.
2. Exercise on the Binomial distribution: Toss a coin 10 times and compute, using `pbinom`, the probability of getting the total numbers of heads you got, assuming that the coin is fair.
   Hint: given sample size n, your assumed probability of a heads $prob$, and the number of heads you got x, the `pbinom` function delivers $P(X \leq x)$, the probability of getting x heads or less. In other words, pbinom is the cumulative distribution function (textbooks often call this simply distribution function).
   Here is how you can compute $P(X \leq x)$:
   `pbinom(x,size=n,p=prob)`
   Note that you have to compute $P(X = x)$!
   Based on what everyone finds, write down the number of cases we have of each possible outcome, along with the theoretical probability.
3. Given $X \sim f(\cdot)$, where $f(\cdot)$ is (a) $Unif(0,10)$, (b) $N(\mu = 100, \sigma^2 = 20)$, (c) $Binom(p = .6, n = 20)$. Find the probability of $P(X < 3), P(X > 11), P(X = 6)$ for each distribution.
4. Plot the priors, likelihoods, and posterior distributions in the four Beta-Binomial cases discussed in the notes.

1.10 Further readings

Some good books introducing Bayesian data analysis methods are the following:

- Lynch, S. M. (2007). Introduction to applied Bayesian statistics and estimation for social scientists. Springer Science & Business Media.
- McElreath, R. (2015). Statistical Rethinking. Texts in Statistical Science.
- Kruschke, J. (2014). Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan. Academic Press.

2 Introduction to Stan probabilistic language

Stan is a probabilistic programming language for statistical inference written in C++ that can be accessed through several interfaces (e.g., R, Python, Matlab, etc.). We will focus on the package `rstan` that integrates Stan to R. In Stan, we first define how the structure of the data looks like, the parameters we want to estimate, and then the priors and likelihood. Stan uses a variant of Hamiltonian Monte Carlo (HMC), called No-U-Turn sampler (NUTS), which is much more efficient than most handcrafted samplers, and also than the traditional Gibbs sampler used in other probabilistic languages such as (Win)BUGS (Lunn et al. 2000) and JAGS (Plummer 2016).

2.1 A Stan program

A Stan program is usually saved as a `.stan` file and accessed through R (or other interfaces) and it is organized into a sequence of optional and obligatory blocks, which must be written in order.

```
functions {
// This is an optional block used for functions that can be used in other blocks.
}
data {
// Obligatory block that specifies the required data for the model.
}
transformed data {
// Optional block if we want to manipulate the data.
}
parameters {
// Obligatory block that specifies the model's parameters.
}
transformed parameters {
// Optional block if we want to manipulate the parameters (re-parametrize the model).
}
model {
// Obligatory block that specifies the model's likelihood and priors.
}
generated quantities {
// Optional block if we want to manipulate the output of our model.
```

```
}
```

A big difference compared to R is that every variable used needs to be declared first with its type (real, integer, vector, matrix, etc.). There are more types, but we'll start using the following ones. Another difference from R is that there must be a semi-colon (`;`) at the end of each line.

- If a variable `mu` is going to contain a real number, either positive or negative, we define it like this:

```
real mu;
```

- If a variable `X` is going to contain a real number that is bounded between two numbers (or by only one), we can add `lower` and/or `upper` to the declaration. Suppose `X` is some type of measurement that can only be between 0 and 1000. (We can add lower and/or upper to any type.)

```
real<lower = 0, upper = 1000>  X;
```

- If `N` is going to contain integers, such as the number of observations (which should be > 0):

```
int<lower = 0>  N;
```

- If `Y` is a going to contain multiple real values, we define it as a vector, and we need to define the number of elements that it will contain. This number can be a variable that was defined earlier (such as `N`, the number of observations, in our example). We can optionally specify a lower boundary for all the values inside the vector. (In Stan, the values inside a vector are always real.)

```
vector<lower = 0> [N] Y;
```

2.2 A first example

As a first example, we will look at on of the analytical example presented before. Recall that we wanted to study whether comprehension of non-canonical sentences in individuals with aphasia is at chance level. The data showed 46 correct responses out of 100. Save this as `firstmodel.stan` (don't run it in R).

```
data {
  int<lower = 1> N;  //Total number of questions
  int<lower = 0, upper = N> c;  //Number of correct responses
}
```

```
parameters {
  // theta is a probability, it has to be constrained between 0 and 1
  real<lower = 0, upper = 1> theta;
}
model {
  // Prior on theta:
  theta ~ beta(1, 1);
  // Likelihood:
  c ~ binomial(N, theta);
}
```

Use the following code to call the model from R. We first load the package `rstan`, and then we tell it that we want to save the models we compiled (this will speed up things), and that we want to run the chains in parallel using all the computer cores. (You may want to use all the cores but one, especially if you plan to use your computer while fitting models.) Then we need to save the data as a `list`, before fitting the model with it. In this simple case, the data is just the number of correct answers, and the total number of observations. We'll skip the fake data simulation for now. But see Exercise 1c.

```r
library(rstan)
# Save compiled models:
rstan_options(auto_write = TRUE)
# Parallelize the chains using all the cores:
options(mc.cores = parallel::detectCores())
# options(mc.cores = parallel::detectCores() - 1) # If you
# want to have an extra core free

# Create a list:
qresp_data <- list(c = 46, N = 100)
# Fit the model with the default values of number of chains
# and iterations chains = 4, iter = 2000
fit <- stan(file = "firstmodel.stan", data = qresp_data)
```

We can see a summary of the posterior by "printing" the model's fit.

```r
print(fit, pars = c("theta"))
```

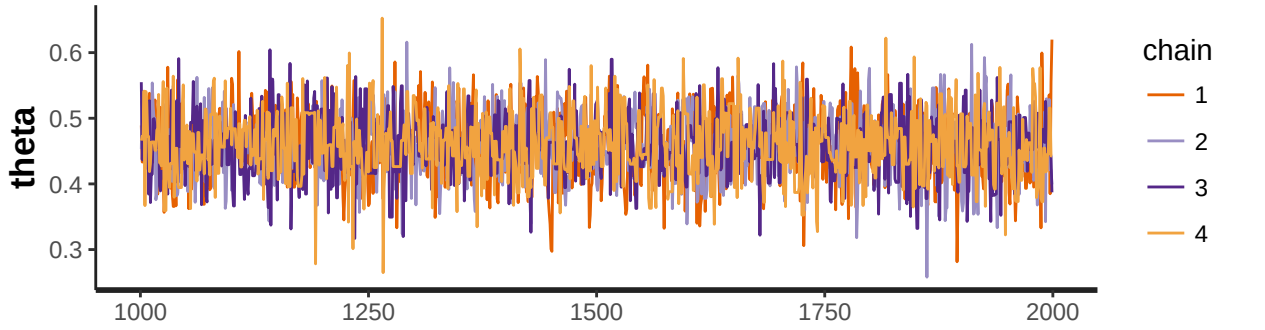
```
## Inference for Stan model: firstmodel.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
```

```
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##       mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
## theta 0.46       0 0.05 0.37 0.43 0.46 0.49  0.55  2334    1
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:45:07 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

The summary displayed by `print` includes means, standard deviations (`sd`), quantiles, Monte Carlo standard errors (`se_mean`), split Rhats, and effective sample sizes (`n_eff`). The summaries are computed after removing the warmup and merging together all chains. Notice that the `se_mean` is unrelated to the `se` of an estimate in the parallel frequentist model. We will see later how to communicate the posterior distribution.

And we should also inspect the chains, see the next section:

```
traceplot(fit, pars = c("theta"))
```



2.3 MCMC diagnostics: Convergence problems and Stan warnings

Before inferring anything from a model, we need to assess convergence. The most important checks or MCMC diagnostics are the following:

- The chains should look like a straight "fat hairy caterpillar": the chains should bounce around the same values and with the same variance.
- The potential scale reduction factors, $\hat{R}$s, of the parameters should be close to one (as a rule of thumb less than 1.1). This indicates that the chains have mixed and they are traversing the same distribution. $\hat{R}$s are printed in `rstan` summary in the column `Rhat` (see section 11.4 of Gelman et al. 2014).

- The effective sample size, n_{eff} should be large enough. The effective sample size is an estimate of the number of independent draws from the posterior distribution. Since the samples are not independent, n_{eff} will generally be smaller than the total number of samples, N. How large n_{eff} should be depends on the summary statistics that we want to use. But as a rule of thumb, $n_{eff}/N > 0.1$.
- There are no (important) warnings, such as, divergent transitions, Bayesian fraction of missing information (BFMI) that was too low, etc. These warning may indicate that the sampler is not adequately exploring the parameter space. See also http://mc-stan.org/misc/warnings.html

For useful graphical checks see https://cran.r-project.org/web/packages/bayesplot/vignettes/MCMC-diagnostics.html

These issues should not be ignored! See the Appendix 2.8 for some troubleshooting ideas to solve them.

2.4 A small experiment

Now we will focus on a more interesting example. The file `1.dat` contains data of a subject (actually, me) pressing the space bar without reading in a self-paced reading experiment.

2.4.1 Preprocessing of the data

```
noreading_data <- read.table(header = F, "data/1.dat", encoding = "latin1")
noreading_data <- noreading_data[c("V2", "V3", "V5", "V6", "V8")]
colnames(noreading_data) <- c("type", "item", "wordn", "word",
    "RT")
tail(noreading_data)
```

```
##       type item wordn           word  RT
## 356 filler    3     0     Vielleicht 214
## 357 filler    3     1          haben 182
## 358 filler    3     2 die_Zahnärztin 179
## 359 filler    3     3       aus_Bonn 177
## 360 filler    3     4  die_Patienten 183
## 361 filler    3     5      verklagt. 162
```

```r
summary(noreading_data$RT)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     110     156     166     169     181     409
```

```r
class(noreading_data)
```

```
## [1] "data.frame"
```

We can't use the `data.frame`, and we need to transform it to a `list`.

```r
# We save the RTs of each row, and also the total number of
# observations N.

noreading_list <- list(Y = noreading_data$RT, N = length(noreading_data$RT))
```

2.4.2 Probability model

Let's model the data with the following assumptions:

- There is a true underlying time, μ, that the participant needs to press the space-bar.
- There is some noise in this process.
- The noise is normally distributed.

This means that the likelihood for each observation i will be:

$$y_i \sim Normal(\mu, \sigma) \tag{225}$$

where $i = 1 \dots N$

And we are going to use the following priors:

$$\begin{aligned} \mu &\sim Normal(0, 2000) \\ \sigma &\sim Normal(0, 500) \text{ truncated so that } \sigma > 0 \end{aligned} \tag{226}$$

The prior for μ is encoding the following information: The model expects that the underlying time could be both positive and negative, and given that the scale of the prior (in this case the standard deviation of the normal distribution) is 2000, we are $\approx 68\%$ certain that the true value would be between 2000 ms and -2000 ms and $\approx 95\%$ certain that it would be between -4000 ms and 4000 ms (two standard deviations away from zero). But we obviously know that the time can't be negative! So we have more prior information than what we are

using for informing the model. We'll discuss this later. Regarding the prior for σ: It must be positive, and we are $\approx 68\%$ certain that the true value would be between 0 ms and 500 ms and $\approx 95\%$ certain that it would be between 0 ms and 1000 ms.

This can be translated to Stan in the following way:

```
data {
  int<lower = 1> N;
  vector[N] Y;
}
parameters {
  real mu;
  real<lower = 0> sigma;
}
model {
  mu ~ normal(0, 2000);
  // This will be a normal dist. truncated at 0,
  // because of our definition of the parameter sigma above:
  sigma ~ normal(0, 500);
  // This is the likelihood: The for-loop specifies the likelihood of each
  // observation in our dataset (from 1 to N),  as coming from a normal
 // distribution with mean mu and sd sigma. (The total log-likelihood will be the
  // sum of the pointwise log-likelihood.)
  for(i in 1:N){
    Y[i] ~ normal(mu, sigma);
  }
}
```

Save it as `no_reading_1.stan`.

2.4.3 Running the Stan model

Before we fit it to our real data, let's verify what happens with fake data.

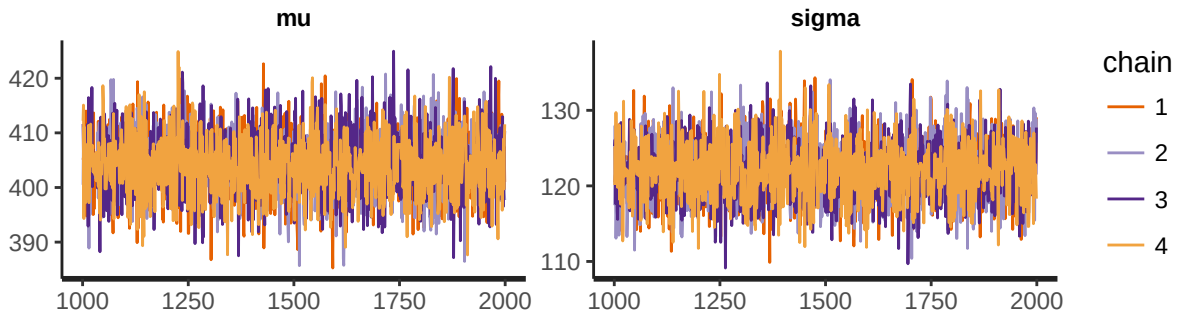
```r
set.seed(123)
N <- 500
true_mu <- 400
true_sigma <- 125
RT <- rnorm(N, true_mu, true_sigma)
```

```
RT <- round(RT)
fake_list <- list(Y = RT, N = N)
```

```
fit_fake <- stan(file = "noreading_1.stan", data = fake_list)
```

We see that the model converges:

```
traceplot(fit_fake)
```



And we get the following posterior distributions:

```
print(fit_fake, probs = c(0.025, 0.5, 0.975), pars = c("mu",
    "sigma"))
```

```
## Inference for Stan model: noreading_1.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##       mean se_mean   sd 2.5% 50% 97.5% n_eff Rhat
## mu     404    0.09 5.44  394 404   415  3568    1
## sigma  122    0.07 3.79  115 122   129  3307    1
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:45:11 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

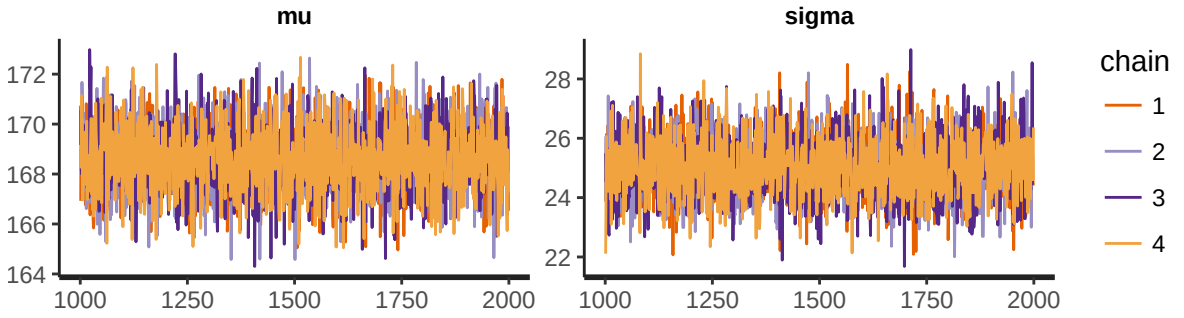
The successful recovery of the parameters means that the model is behaving as expected, now we can see what happens with real data:

```
fit_noreading <- stan(file = "noreading_1.stan", data = noreading_list)
```

```
print(fit_noreading, probs = c(0.025, 0.5, 0.975), pars = c("mu",
    "sigma"))
```

```
## Inference for Stan model: noreading_1.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##       mean se_mean   sd  2.5% 50% 97.5% n_eff Rhat
## mu     169    0.02 1.33 166.1 169   171  3400    1
## sigma   25    0.02 0.98  23.2  25    27  3478    1
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:45:15 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
traceplot(fit_noreading)
```



2.4.4 Summarizing the posterior

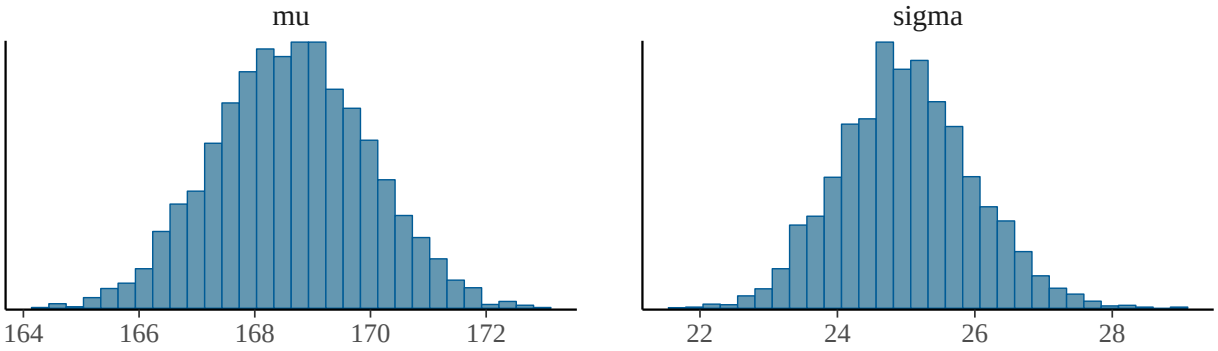
The package `bayesplot` provides very convenient plot functions (based on `ggplot2`). See the vignettes in https://cran.r-project.org/web/packages/bayesplot/index.html

```
library(bayesplot)
# We need to first extract the chains:
post_noreading <- as.array(fit_noreading)
dimnames(post_noreading)
```

```
## $iterations
## NULL
##
```

```
## $chains
## [1] "chain:1" "chain:2" "chain:3" "chain:4"
##
## $parameters
## [1] "mu"    "sigma" "lp__"
```

```
# And then we can plot them:
mcmc_hist(post_noreading, pars = c("mu", "sigma"))
```



The posterior distribution of μ is not the distribution of RTs!

We are assuming that there's a true underlying time it takes to press the space bar, μ, and there is normally distributed noise that generates the different RTs. This is encoded in our likelihood by assuming that RTs are distributed with an unknown true mean μ (and an unknown standard deviation σ). The objective of the Bayesian model is to learn about the possible values of μ, or in other words, to get a distribution that encodes what we know about the true mean of the distribution of RTs (and also another distribution that encodes what we know about true standard deviation, σ, of the distribution of RTs.)

Given our data, what can we learn from the posterior distribution of μ?

Our model allows us to have answers to questions such as:

What is the probability that the underlying value of the mindless press of the space bar would be over, say 170 ms?

Answer:

```
mu_samples <- post_noreading[, , "mu"]
# Another possibility would be
# mu_samples <- rstan::extract(fit_noreading)$mu
```

```r
# Proportion of samples over 170
mean(mu_samples > 170)
```

```
## [1] 0.158
```

Which range of values contains a specified amount of probability?

This type of interval is also known as a credible interval.

A credible interval demarcates the range within which we can be certain with a certain probability that the "true value" of a parameter lies given the data and the model. This is very different from the frequentist confidence interval! See for example, Hoekstra et al. (2014) and Morey et al. (2016).

The percentile interval is a type of credible interval (the most common one), where we assign equal probability mass to each tail. We generally report 95% credible intervals. But we can extract any interval, a 73% interval, for example, leaves 13.5% of the probability mass on each tail, and we can calculate it like this:

```r
print(fit_noreading, probs = c((1 - 0.73)/2, (1 + 0.73)/2),
    pars = c("mu"))
```

```
## Inference for Stan model: noreading_1.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##    mean se_mean   sd 13.5% 86.5% n_eff Rhat
## mu  169    0.02 1.33   167   170  3400    1
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:45:15 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

2.4.5 Influence of priors and sensitivity analysis

Everything was normally distributed in our example (or truncated normal), but the fact that we assumed that RTs were normally distributed is completely unrelated to our (truncated) normally distributed priors. Let's try with uniform priors without low or high boundaries,

these are improper distributions[7] that assign the same probability density to every outcome. In general, this is a bad idea for two reasons: (i) it is computationally expensive (the sampler has a huge parameter space), and (ii) it is providing information that we know it's not accurate (every value is equally likely). But in our very simple example these priors will work.

$$\begin{aligned} \mu &\sim Uniform(-\infty, \infty) \\ \sigma &\sim Uniform(0, \infty) \end{aligned} \tag{227}$$

This can be translated to Stan in the following way. If no priors are specified, uniform priors are assumed. I'm saving it as `noreading_2.stan`.

```
data {
  int<lower = 1> N;
  vector[N] Y;
}
parameters {
  real mu;
  real<lower = 0> sigma;
}
model {
  // If we don't define priors,
  // the priors are assumed to be uniform,
  // based on lower and upper constraint of the parameters.
  // We ALWAYS have priors.
  for(i in 1:N){
    Y[i] ~ normal(mu, sigma);
  }
}
```

```
fit_noreading_2 <- stan(file = "noreading_2.stan", data = noreading_list)
```

The output of the model will be more or less the same.

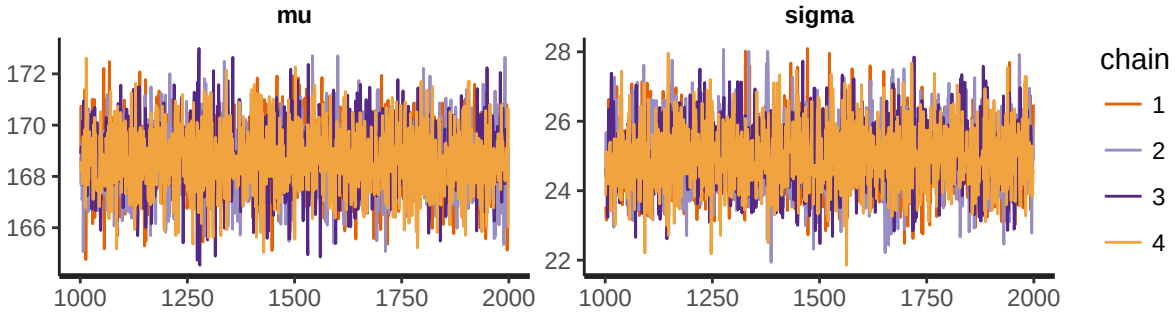
```
print(fit_noreading_2, probs = c(0.025, 0.5, 0.975), pars = c("mu",
    "sigma"))
```

```
## Inference for Stan model: noreading_2.
```

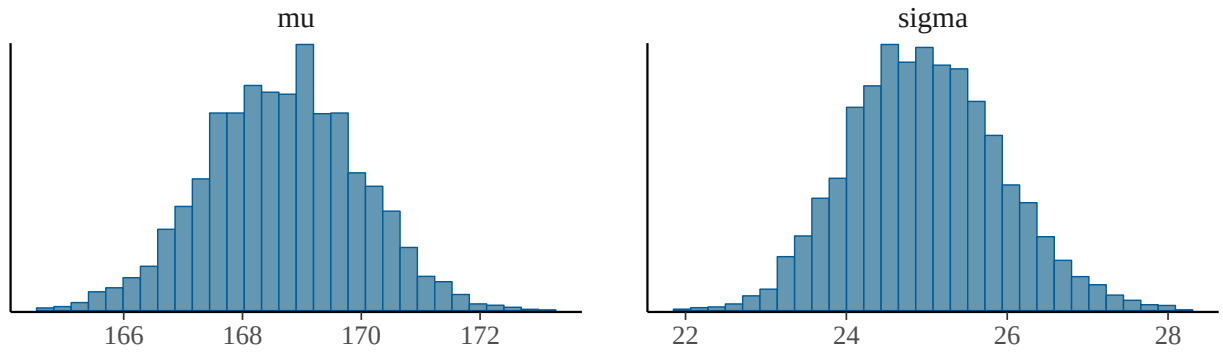
[7] They don't integrate to 1.

```
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##       mean se_mean   sd  2.5%   50% 97.5% n_eff Rhat
## mu     169    0.02 1.29 166.1 168.7 171.2  3507    1
## sigma   25    0.02 0.94  23.2  24.9  26.9  3393    1
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:45:19 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
traceplot(fit_noreading_2)
```




```
post_noreading_2 <- as.array(fit_noreading_2)
mcmc_hist(post_noreading_2, pars = c("mu", "sigma"))
```



In general we don't want our priors to have too much influence on our posterior. This is unless we have very good reasons for having informative priors, such as a very small sample and a lot of prior information; an example would be if we have data from an impaired population, which makes it hard to increase our sample size. In general, however, we only use priors to get reasonable posterior distributions. We center the priors in 0 and we let the data alone to "decide" whether an effect is positive or negative. This type of priors are called

weakly regularizing priors. Notice that a uniform prior is not a weakly regularizing prior, it assumes that every value is equally likely, zero is as likely as infinity. If you're unsure whether the priors you chose have too strong an effect on the posterior distribution, you can do a sensitivity analysis: You try different priors and verify that the posterior doesn't change drastically (for a published example, see Vasishth et al. 2013). See also Exercise 1a.

2.5 A slightly more interesting analysis of the small experiment

More realistically, we might have run the small experiment to know whether I (the participant) tended to speedup (practice effect) or slowdown (fatigue effect) while I was pressing the space bar. And maybe how strong this effect was.

2.5.1 Preprocessing the data

We need to have data about the number of times the space bar was pressed for each observation, and add it to our list. It's a good idea to center the number of presses (a covariate) to have a clearer interpretation of the intercept. In general, centering predictors is always a good idea, for interpretability and for computational reasons.

```
# We create the new column in the data frame
noreading_data$presses <- 1:nrow(noreading_data)
# We center the column
noreading_data$c_presses <- noreading_data$presses - mean(noreading_data$presses)
# We add it to the list that goes into rstan
noreading_list$presses <- noreading_data$c_presses
```

2.5.2 Probability model

Our model changes, because we have a new parameter.

$$RT_i \sim Normal(\alpha + presses_i \cdot \beta, \sigma) \tag{228}$$

where $i = 1 \ldots N$

And we are going to use the following priors:

$$
\begin{aligned}
\alpha &\sim Normal(0, 2000) \\
\sigma &\sim Normal(0, 500) \text{ truncated so that } \sigma > 0 \\
\beta &\sim Normal(0, 500)
\end{aligned} \tag{229}
$$

We are basically fitting a linear model, α represents the intercept (namely, the grand mean of the RTs), and β represents the slope. Which information are the priors encoding? Does it make sense?

We'll write this in Stan code as follows.

```
data {
  int<lower = 1> N;
  vector[N] Y;
  vector[N] presses;
}
parameters {
  real alpha;
  real beta;
  real<lower = 0> sigma;
}
model {
  alpha ~ normal(0, 2000);
  beta ~ normal(0, 500);
  // This will be a normal dist. truncated at 0,
  // because of our definition of the parameter sigma:
  sigma ~ normal(0, 500);
  for(i in 1:N){
    Y[i] ~ normal(alpha + beta * presses[i], sigma);
  }
}
```

Save it as `noreading_3.stan`.

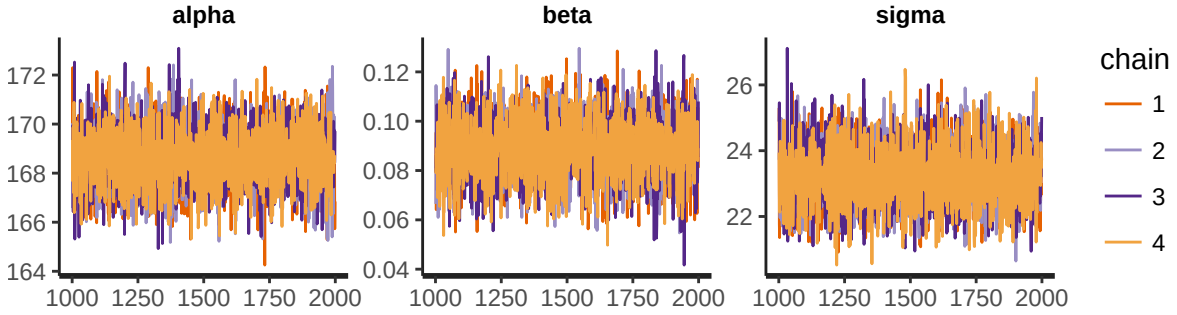
```
fit_noreading_3 <- stan(file = "noreading_3.stan", data = noreading_list)
```

```
print(fit_noreading_3, probs = c(0.025, 0.5, 0.975), pars = c("alpha",
    "beta", "sigma"))
```

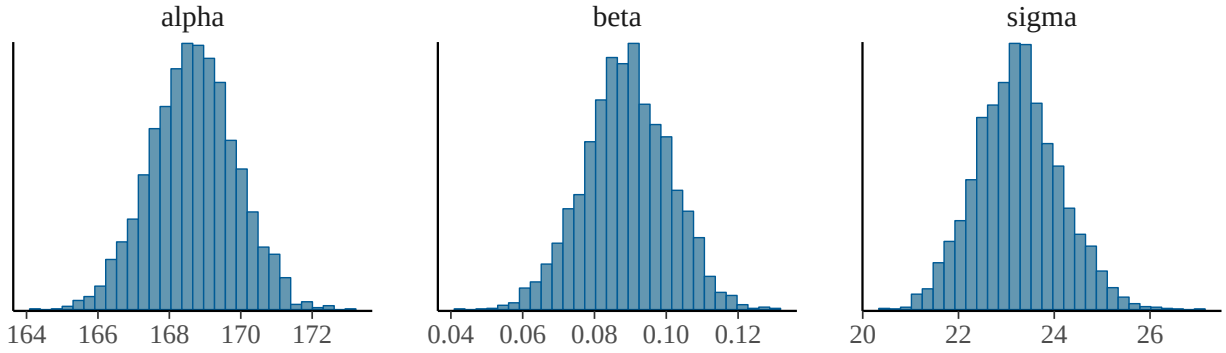
```
## Inference for Stan model: noreading_3.
```

```
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##         mean se_mean   sd   2.5%    50%  97.5% n_eff Rhat
## alpha 168.67    0.02 1.19 166.32 168.67 171.00  4000    1
## beta    0.09    0.00 0.01   0.06   0.09   0.11  4000    1
## sigma  23.24    0.01 0.88  21.55  23.24  25.02  4000    1
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:45:54 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
traceplot(fit_noreading_3)
```




```
post_noreading_3 <- as.array(fit_noreading_3)
mcmc_hist(post_noreading_3, pars = c("alpha", "beta", "sigma"))
```



2.5.3 Summarizing the posterior and inference

How can we answer our research question? What is the effect of pressing the bar on the participant's reaction time?

We'll need to examine what happens with β. The summary gives us important in-

formation, we can learn that the most likely values of β will be around the mean of the posterior 0.09, and we can be 95% certain that the true value of β given the model and the data lies between 0.06 and 0.11. We extract these values by doing `summary(fit_noreading_3)$summary["beta",column_name]` and replacing column_name by either `"mean"`, `"2.5%"`, or `"97.5%"`.

We see that as the number of times the space bar is pressed increases, the participant becomes slower. If we want to know how likely it is that the participant was slower rather than faster, we can examine the proportion of samples above zero:

```
beta_samples <- post_noreading_3[, , "beta"]


mean(beta_samples > 0)
```

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

We would report this in a paper as $\hat{\beta} = 0.09$, 95% CrI $= [0.06, 0.11]$, $P(\beta > 0) \approx 1$

Can we really conclude that there is a fatigue effect? It depends on how much we expect the fatigue to affect the RTs. Here we see that only after 100 button presses, we'll see a slowdown of 9 ms on average (0.09×100). We will need to think whether the size of this effect make sense considering the previous literature. Sometimes this requires a meta-analysis. See Jäger, Engelmann, and Vasishth (2017) for an example, and the use of this prior knowledge in Nicenboim et al. (Submitted).

2.5.4 Posterior predictive checks

Let's say we know that our model is working as expected, since we already used fake data to test the recovery of the parameters. How do we know if our model is "good"?

We will examine the descriptive adequacy of the models (Shiffrin et al. 2008; Gelman et al. 2014, Chapter 6): the observed data should look plausible under the posterior predictive distribution. The posterior predictive distribution is composed of one dataset for each sample from the posterior. (So it will generate as many datasets as iterations we have after the warm-up.) Achieving descriptive adequacy means that the current data could have been predicted by the model. Passing a test of descriptive adequacy is not strong evidence in favor of a model, but a major failure in descriptive adequacy can be interpreted as strong evidence against a model (Shiffrin et al. 2008).

To do posterior predictive checks we need to add the `generated_quantities` block at the end of our Stan model and we'll store the predictions in a vector called `pred_Y` with length

`N` (number of observations). We do this with a function called `normal_rng(mu, sigma)`; this function generates a random number (rng stands for random number generator) based on the two parameters of the function (similar to the R function `rnorm(1, mu, sigma)`). Notice that `pred_Y` will iterate over the entire dataset of length `N` on each iteration of the sampler. This will become clearer after running the code.

Save the model as `noreading_4.stan`.

```
data {
  int<lower = 1> N;
  vector[N] Y;
  vector[N] presses;
}
parameters {
  real alpha;
  real beta;
  real<lower = 0> sigma;
}
model {
  alpha ~ normal(0, 2000);
  beta ~ normal(0, 500);
  // This will be a normal dist. truncated at 0,
  // because of our definition of the parameter sigma:
  sigma ~ normal(0, 500);
  for(i in 1:N){
    Y[i] ~ normal(alpha + beta * presses[i], sigma);
  }
}
generated quantities {
  vector[N] pred_Y;
  for(i in 1:N){
    pred_Y[i] = normal_rng(alpha + beta * presses[i],sigma);
  }
}
```

We fit the model, and check its convergence as usual.

```
fit_noreading_4 <- stan(file = "noreading_4.stan", data = noreading_list)
```

```
traceplot(fit_noreading_4, pars = c("alpha", "beta", "sigma"))
print(fit_noreading_4, probs = c(0.025, 0.5, 0.975), pars = c("alpha",
    "beta", "sigma"))
```

But now, once we have fit the new model it is possible to extract the predicted RTs:

```
matrix_pred_Y <- rstan::extract(fit_noreading_4)$pred_Y
dim(matrix_pred_Y)
```

```
## [1] 4000  361
```

```
# first dimension: number of samples = 2000 iterations per chain /2 * 4 chains
# second dimension: number of original datapoints
```

This means we have 4000 simulations, each simulation consists of an entire dataset that is created with the values of `mu`, `beta` and `sigma` that the corresponds to a given iteration (after the warmup). For example, `matrix_pred_Y[1,]` is a complete simulated dataset generated from the first sample of the parameters.

We could recreate it in R with the code that follows.

Notice that this won't return the exact same values than Stan gave us, however, because we are generating random numbers based on the distribution.

```
alpha_samples <- rstan::extract(fit_noreading_4)$alpha
beta_samples <- rstan::extract(fit_noreading_4)$beta
sigma_samples <- rstan::extract(fit_noreading_4)$sigma
rt_gen <- c()
for(i in 1:noreading_list$N) {
  mu <- alpha_samples[1] + beta_samples[1] * noreading_list$presses[i]
  sigma<- sigma_samples[1]
  rt_gen[i] <- rnorm(1, mu , sigma )
}

length(rt_gen)
```

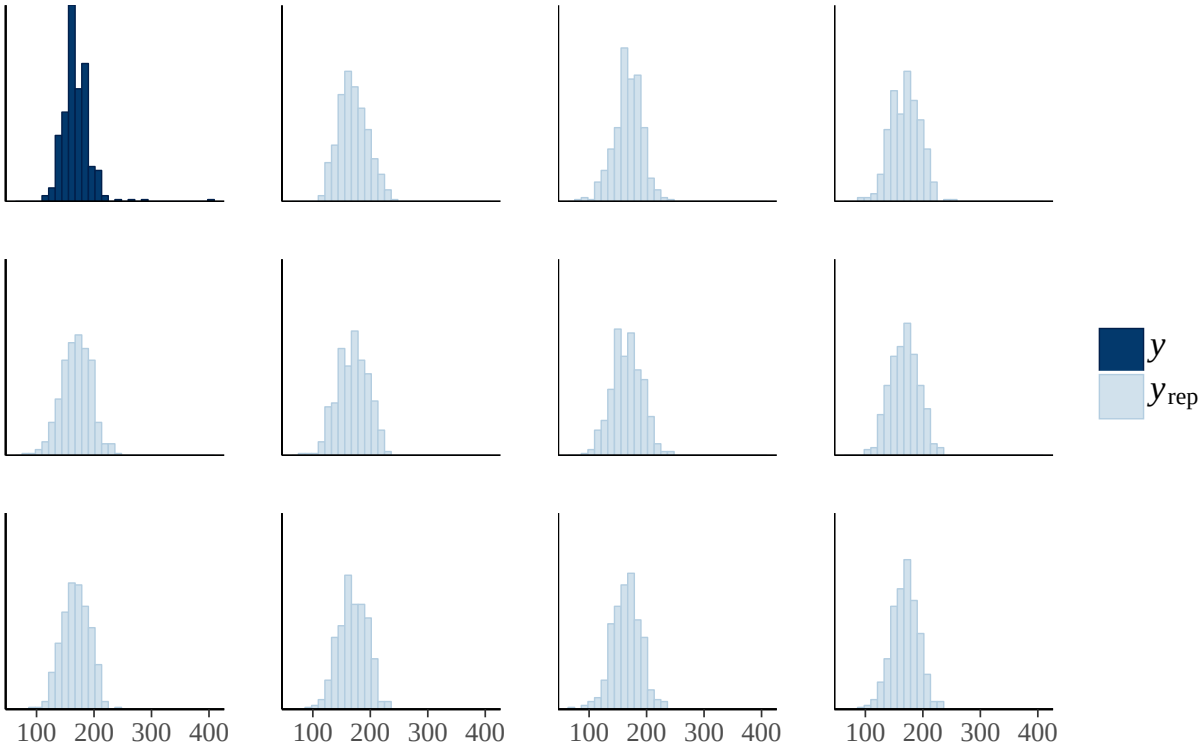
```
## [1] 361
```

```
head(rt_gen)
```

```
## [1] 152 141 150 144 154 170
```

We'll use the values generated by our Stan model to verify whether the general shape of the actual distribution matches the distributions from some of the generated datasets. Let's compare the real data against 11 of these 4000 datasets (called here `y_rep`):

```
# Let's pick 11 random simulations over the 4000 that we
# have
pick <- sample(1:4000, 11)
ppc_hist(noreading_list$Y, matrix_pred_Y[pick, ])
```



Is the simulated data similar to the real data?

Our dataset seems to be more skewed to the right than our predicted datasets. This is not too surprising, we assumed that the likelihood was a normal distribution, but latencies are not very normal-like, they can't be negative and can be arbitrarily long.

2.5.5 A better probability model using the log-normal distribution

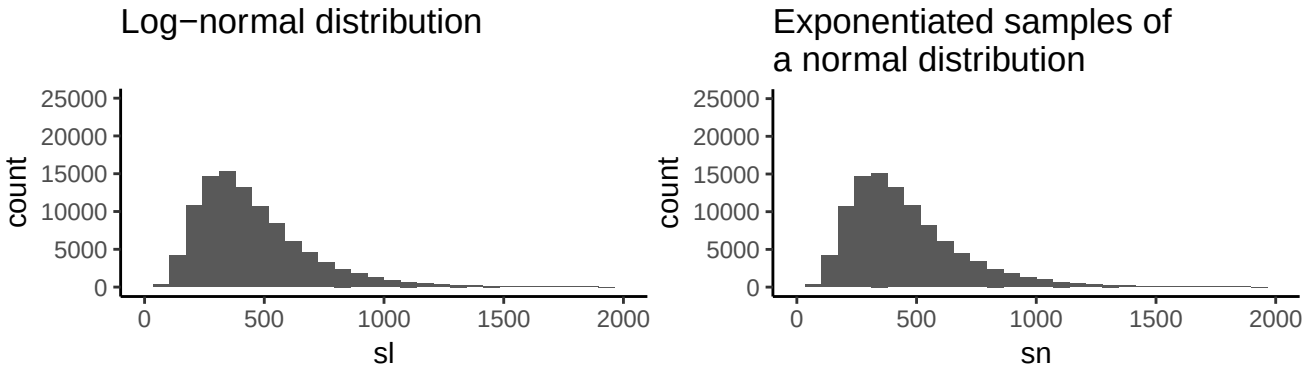
Since we know that the latencies shouldn't be normally distributed, we can choose a more realistic distribution for the likelihood. A good candidate is the log-normal distribution since a variable (such as time) that is log-normally distributed takes only positive real values.

If Y is log-normally distributed, this means that $log(Y)$ is normally distributed.[8] Something important to notice is that the log-normal distribution is defined using again μ and σ, but this corresponds to the mean and standard deviation of the normally distributed logarithm $log(Y)$. Thus μ and σ are on a different scale than the variable that is log-normally distributed.

This also means that you can create a log-normal distribution by exponentiating the samples of a normal distribution:

```
mu <- 6
sigma <- 0.5
N <- 100000
# We generate N random samples from a log-normal
# distribution.
sl <- rlnorm(N, mu, sigma)
lognormal_plot <- ggplot(data.frame(samples = sl), aes(sl)) +
    geom_histogram() + ggtitle("Log-normal distribution\n") +
    ylim(0, 25000) + xlim(0, 2000)
# We generate N random samples from a normal distribution,
# and then we exponentiate them
sn <- exp(rnorm(N, mu, sigma))
normalplot <- ggplot(data.frame(samples = sn), aes(sn)) + geom_histogram() +
    ggtitle("Exponentiated samples of\na normal distribution") +
    ylim(0, 25000) + xlim(0, 2000)

plot(lognormal_plot)
plot(normalplot)
```

[8] In fact, $log_e(Y)$ or $ln(Y)$, but since it is the most popular logarithm in statistics we'll write it as just $log()$



2.5.6 The slightly more realistic probability model

If we assume that RTs are log-normally distributed, we'll need to change our model:

$$Y_i \sim LogNormal(\alpha + presses_i \cdot \beta, \sigma) \tag{230}$$

where $i = 1 \ldots N$

But now the scale of our priors needs to change! They are no longer in milliseconds.

$$\begin{aligned} \alpha &\sim Normal(0, 10) \\ \sigma &\sim Normal(0, 2) \text{ truncated so that } \sigma > 0 \\ \beta &\sim Normal(0, 1) \end{aligned} \tag{231}$$

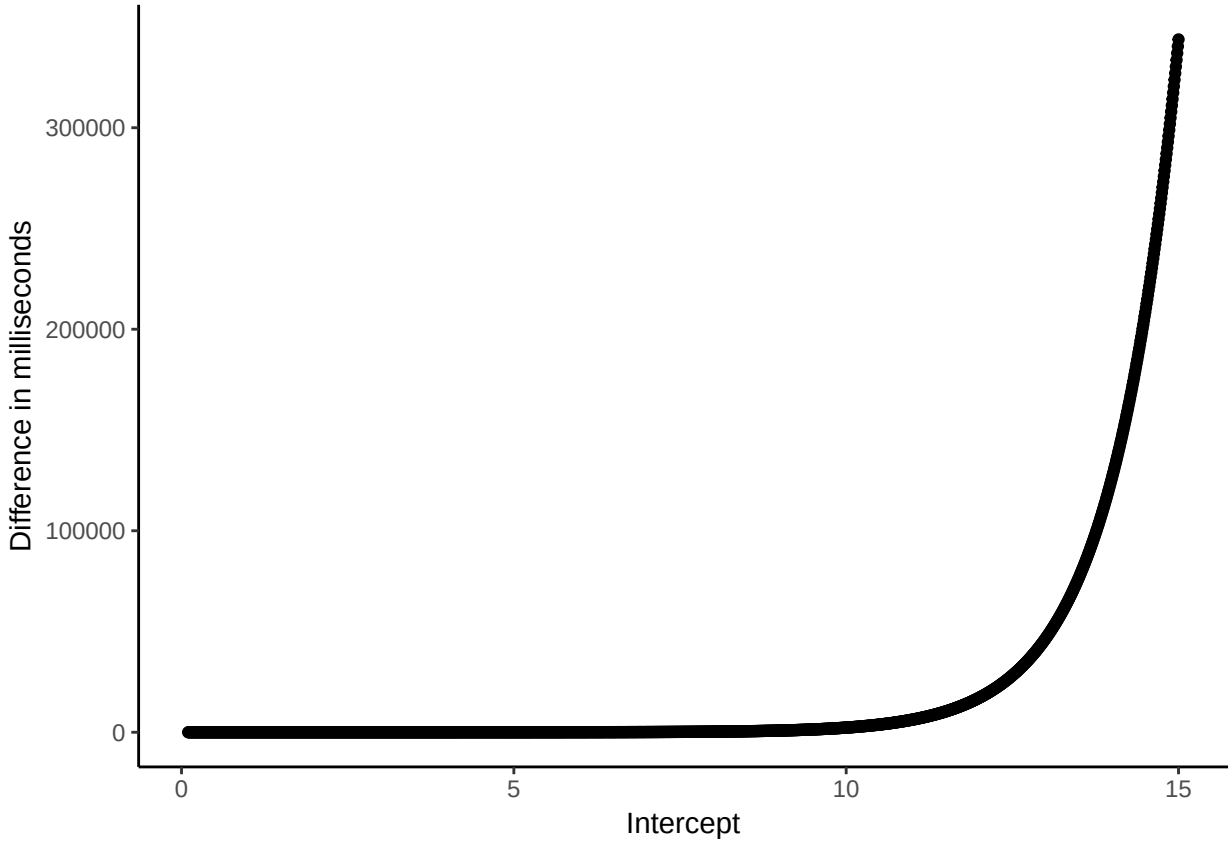
The interpretation of the parameters changes and it is more complex than if we were dealing with a linear model (that is, with a normal distribution):

- α. In our previous linear model, α represented the grand mean (or the grand median since in a normal distribution both coincide), and was equivalent to our previous μ (since β was multiplied by 0). But now, the grand mean needs to be calculated in the following way, $\exp(\alpha + \sigma^2/2)$. Interestingly, the grand median will just be $exp(\alpha)$,[9] and we could assume that this represents the underlying time it takes to press the space bar if there would be no noise, that is, if σ had no effect. This also means that the prior of α is not in milliseconds, but in log(milliseconds).
- β. In a linear model, β represents the slowdown for each time the space bar is pressed. Now β is the effect on the log-scale, and the effect in milliseconds depends on the intercept α: $exp(\alpha + \beta) - exp(\alpha)$. Notice that the log is not linear and the effect of β will have more impact on milliseconds as the intercept grows. For example, if we start

[9] You can check in Wikipedia (https://en.wikipedia.org/wiki/Log-normal_distribution) why.

with (i) $exp(5) = 148$, and we add 0.1 in log-scale, $exp(5+0.1) = 164$, we end up with a difference of 15 ms; if we start with (ii) $exp(6) = 400$, and we add 0.1, $exp(6+0.1) = 445$, we end up with a difference of 45 ms. You can also see it graphically below.

- σ. This is the standard deviation of the normal distribution of $log(y)$.



What kind of information are the priors encoding?

- For α: We are 95% certain that the grand median of the RTs will be between ≈ 0 and 485165195 milliseconds. This is a (very-)weakly regularizing prior because it won't affect our results, but it will down-weights values for the grand median of the RTs that are extremely large, and won't allow the grand median to be negative. We calculate the previous range by back-transforming the values that lie between two standard deviations of the prior (2×10) to millisecond scale: $exp(-10 \times 2)$ and $exp(10 \times 2)$).
- For β: This is more complicated, because the effect on milliseconds will depend on the estimate of α. However, we can assume some value for α and it will be enough to be in the right order of magnitude. So let's assume 500 ms. That will mean that we are 95% certain that the effect of pressing the space bar will be between -491 and 26799 milliseconds. It is asymmetric because the log-scale is asymmetric. But the prior is weak enough so that if we assume 1000 or 100 instead of 500, the possible estimates of

β will still be contained in the prior distribution. We calculate this by first finding out the value in milliseconds when we are two standard deviations away in both directions: (2×2), that is $exp(500 - 2 \times 2)$ and $exp(500 + 2 \times 2)$, and we subtract from that the value of α that we assumed, 500: $exp(500 - 2 \times 2) - 500$ and $exp(500 + 2 \times 2) - 500$.

- For σ. This indicates that we are 95% certain that the standard deviation of $log(y)$ will be between 0 and 2. So 95% of the RTs will be between $exp(log(500) - 1 \times 2) = 68$ and $exp(log(500) + 1 \times 2) = 3695$.

What happens if we replace 500 by 100, and by 1000? What happens if it is 10 instead? Does it still makes sense?

We'll code the model as follows. Notice that we can use the `generated quantities` also to transform our parameters in a scale we can understand more easily (milliseconds).

```
data {
  int<lower = 1> N;
  vector[N] Y;
  vector[N] presses;
}
parameters {
  real alpha;
  real beta;
  real<lower = 0> sigma;
}
model {
  alpha ~ normal(0, 10);
  beta ~ normal(0, 2);
  // This will be a normal dist. truncated at 0,
  // because of our definition of the parameter sigma:
  sigma ~ normal(0, 1);
  for(i in 1:N){
    Y[i] ~ lognormal(alpha + beta * presses[i], sigma);
  }
}
generated quantities {
  vector[N] pred_Y;
  real RT_under;
  real effect_of_1_press;
```

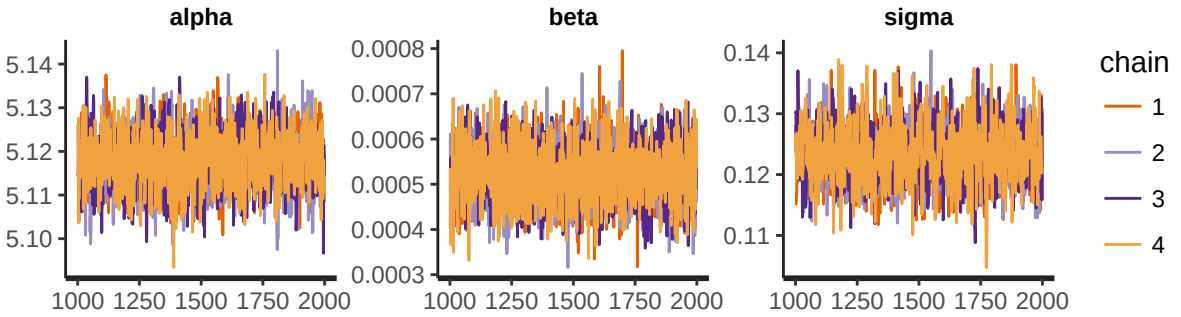
```
  for(i in 1:N){
    pred_Y[i] = lognormal_rng(alpha + beta * presses[i],sigma);
  }
  RT_under = exp(alpha);
  effect_of_1_press = exp(alpha + beta * 1 ) - exp(alpha);
}
```

Save the above code as `noreading_log.stan`.

We fit the model, and check its convergence as usual.

```
fit_noreading_log <- stan(file = "noreading_log.stan", data = noreading_list)
```

```
traceplot(fit_noreading_log, pars = c("alpha", "beta", "sigma"))
```




```
print(fit_noreading_log, probs = c(0.025, 0.5, 0.975), pars = c("alpha",
    "beta", "sigma", "RT_under", "effect_of_1_press"))
```

```
## Inference for Stan model: noreading_log.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                    mean se_mean   sd   2.5%    50%  97.5% n_eff Rhat
## alpha              5.12    0.00 0.01   5.11   5.12   5.13  1695    1
## beta               0.00    0.00 0.00   0.00   0.00   0.00  3948    1
## sigma              0.12    0.00 0.00   0.11   0.12   0.13  1413    1
## RT_under         167.00    0.03 1.08 164.89 166.99 169.18  1694    1
## effect_of_1_press  0.09    0.00 0.01   0.07   0.09   0.11  3935    1
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:47:13 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
```

```
## convergence, Rhat=1).
```

```
# We'll need more digits to see what's going on with beta
print(fit_noreading_log, probs = c(0.025, 0.5, 0.975), pars = c("beta"),
    digits = 5)
```

```
## Inference for Stan model: noreading_log.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##         mean se_mean      sd    2.5%     50%   97.5% n_eff Rhat
## beta 0.00052       0 0.00006 0.00041 0.00052 0.00064  3948    1
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:47:13 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

2.5.7 Summarizing the posterior and inference

We can summarize the posterior and do inference as before. If we want to talk about the effect estimated by the model, we summarize the posterior of β in the following way: $\hat{\beta} = 0$, 95% CrI $= [0, 0.001]$, $P(\beta > 0) \approx 1$

But sometimes, the effect in milliseconds is easier to interpret. We generated `effect_of_1_press` in the generated quantities block, which is not the same as the linear model's β. Our generated estimate will tell us the estimate of the slowdown produced by pressing the space bar in the middle of the experiment once, assuming that the RTs are log-normally distributed: 0.09 ms, 95% CrI $= [0.07, 0.11]$. Coincidentally, it is the same value as before, but this is not always the case, and since it's not linear the effect won't be the same across the whole experiment; see Exercise ?a.

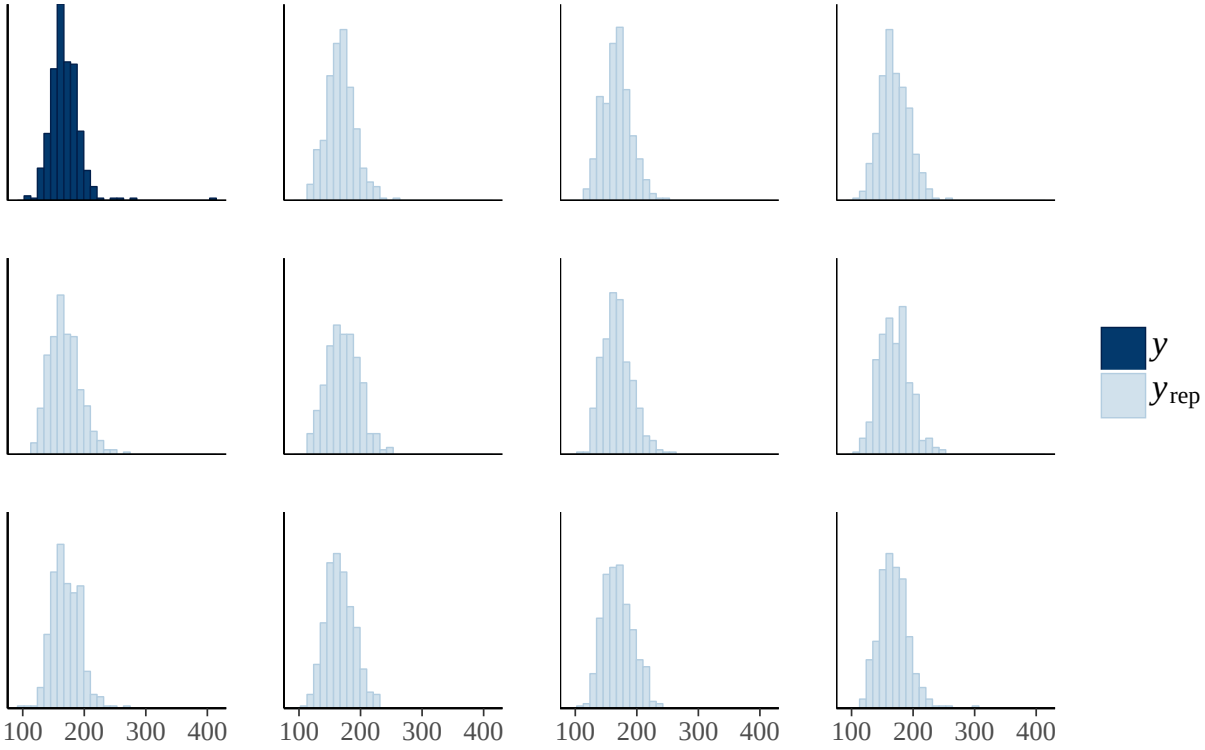
2.5.8 Posterior predictive checks and distribution of summary statistics

We can now verify whether our predicted datasets look more similar to the real dataset.

```
matrix_pred_Y_log <- rstan::extract(fit_noreading_log)$pred_Y
dim(matrix_pred_Y_log)
```

```
## [1] 4000  361
```

```
# Let's pick 11 random simulations over the 4000
pick <- sample(1:4000, 11)
ppc_hist(noreading_list$Y, matrix_pred_Y_log[pick, ])
```

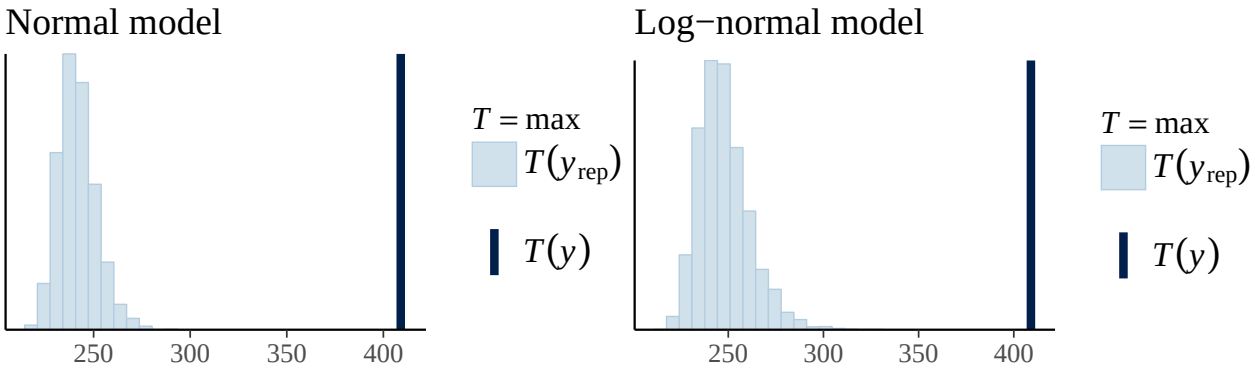


Is the simulated data now more similar to the real data?

It seems so, but it's not easy to tell. Another way to examine this would be to look at the distribution of summary statistics. The idea is to compare the distribution of representative summary statistics for the datasets generated by different models and compare them to the observed statistics. Since we suspect that the log-normal distribution may capture the long tail, we could use the maximum as a summary statistics.

We do it using the package `bayesplot`:

```
normalsum <- ppc_stat(noreading_list$Y, matrix_pred_Y, stat = "max") +
             ggtitle("Normal model")
lognormalsum <- ppc_stat(noreading_list$Y, matrix_pred_Y_log, stat = "max") +
                ggtitle("Log-normal model")
plot(normalsum)
plot(lognormalsum)
```



Here we see that both distributions are unable to capture the maximum value of the observed data, and that there's still room for improving the model. Another more advanced possibility is to compare the models using cross-validation techniques; see for example, Gelman, Hwang, and Vehtari (2014); Vehtari and Ojanen (2012); Vehtari, Gelman, and Gabry (2017).

2.5.9 General workflow

This is the general workflow that we recommend for a Bayesian model.

1. Define the full probability model:
    a. Think about the likelihood.
    b. Think about the priors.
    c. Write the Stan model.
2. Fake data simulations:
    a. Simulate data with known values for the parameters.
    b. Fit the model and do MCMC diagnostics.
    c. Verify that it recovers the parameters from simulated data.
3. Fit the model with real data and do MCMC diagnostics.
4. Evaluate the model's fit (e.g., posterior predictive checks, distribution of summary statistics). This may send you back to 1.
5. Inference/prediction.
6. Sometimes model comparison if there's an alternative model.

2.6 Key concepts

- Stan basic blocks and data types.
- How to identify convergence problems (MCMC diagnostics).
- Normal and log-normal distributions.
- Summaries of the posterior and inference.
- Model evaluation using posterior predictive checks.
- General workflow for doing Bayesian analysis.

2.7 Exercises

1. For the model in Section 2.2.
   (a) Recall that when we use the Beta distribution as a prior for the θ parameter in a Binomial, the two parameters of the Beta indicate previous outcomes and not only the most likely value of θ. For example, $Beta(2,2)$ and $Beta(50,50)$ both indicate that .5 is the most likely value of θ, but $Beta(50,50)$ indicates that we are more certain about it than $Beta(2,2)$. Assume that you are quite certain that the average accuracy in the task is 80% for individuals with aphasia. How would you change the priors for the model in 2.2? Fit the model several times, assuming the same average accuracy as prior information, but varying the amount of uncertainty? When do the results change?
   (b) Change the data to `qresp_data <- list(c = 460, N = 1000)`, and repeat 1a with the same priors.
   (c) Simulate data assuming that the true θ is exactly .5 and (i) `N = 100`, (ii) `N = 1000` and run the `firstmodel.stan`. (Tip: use the function `rbinom` to simulate data.) Is the true value of θ inside 95% credible interval?
2. For the model in Section 2.4
   (a) Change the priors of μ and σ in `noreading_1.stan` to a Cauchy and a truncated Cauchy distributions. A Cauchy distribution is basically a bell shaped distribution with very fat tails.[a] What are reasonable values for the location and scale? What's the difference between having a Cauchy or a normal distribution as priors?

Continues in the next page.

[a] You can read about this distribution in https://en.wikipedia.org/wiki/Cauchy_distribution.

3. For the models in Section 2.5.
   (a) Edit the `generated quantities` block in `noreading_log.stan` to estimate the slowdown in milliseconds (mean and 90% credible interval) for the last time the participant pressed the space bar in the experiment. In addition, predict the slowdown if the experiment would have had 500 observations.
   (b) Optional but very recommended: Simulate data that assumes a value of α of 400 and σ of 125 and a practice effect of (i) 0.00001 and (ii) 0.1 (both in log-scale). Fit these data with `noreading_log.stan`. Is the true value of the effect in the 95% credible interval in both cases?
   (c) Add word length as a covariate in `noreading_log.stan`, summarize the posterior. How does the length of the words affect RTs?

2.8 Appendix - Troubleshooting problems of convergence

1. Rhat > 1.1 First of all check that there are no silly mistakes in the model. Forgetting to put parenthesis, multiplying the wrong parameters, using the wrong operation, etc. can create a model that can't converge. As our models grow in complexity there are more places where to make mistakes. Start simple, see if the model works, add complexity slowly, checking if the model converges at every step. In very rare occasions, when Rhat >1.1 and the model is correct, it may help to increase the number of iterations, but then it's usually a better idea to re-parametrize the model, see 3.
2. Stan gives a warning. The solution may also be point 1. But if the model is correctly specified, you should check Stan's website, there is a very good guide to solve problems in: http://mc-stan.org/misc/warnings.html. If this doesn't work, you may need to re-parametrize the model, see 3.
3. Some models have convergence issues because the sampler struggles to explore the parameters space. This is specially relevant in complex hierarchical models. In this case, the solution might be to re-parametrize the model. This is by no means trivial. However, the simplest parametrization trick to try is to have all the priors on the same rough scale, that is priors shouldn't have different orders of magnitude. You can find some suggestions in the chapter 21 of Stan manual (Stan Development Team 2017), and the following case study: http://mc-stan.org/users/documentation/case-studies/qr_regression.html.

3 Bayesian hierarchical models (also known as multilevel or mixed-effects models)

3.1 Advantages (and disadvantages)

Why do we need them?

1. To adjust estimates for repeated sampling when more than one observation arises from the same "cluster": individual, item, word, etc…
2. To adjust estimates for imbalance in sampling. It is not extremely important in carefully designed experiments, but it can happen that we need to throw away data from items (i.e., because of a typo) or from subjects.
3. To study variation between participants.
4. To avoid averaging over participants (and items) to solve (1), that is, repeated sampling. Averaging artificially removes variation and leads to false confidence. Hierarchical models allow us to preserve the uncertainty in the original pre-averaged values, while still using the average to make predictions.

What are the costs of fitting hierarchical models?

- New assumptions: distributions from which the characteristics of the clusters arise.
- Time! It takes much longer than a non-hierarchical model.

3.2 Applied example: Attachment preference

Let's consider these two sentences taken from Swets et al. (2008), available through Pavel Logačev's github page.[10] (Notice that we are not focusing on the phenomenon that Logacev or Swets et al investigated.)

(1) The son of the princess who scratched himself in public was terribly humiliated (N1 attachment)

(2) The son of the princess who scratched herself in public was terribly humiliated. (N2 attachment)

There is evidence that English speakers have a preference for N2 attachment (also called low-attachment), that is, people tend to interpret that the relative clause starting with who will refer to the second noun ("the princess") rather than the first one (e.g., Carreiras and Clifton 1999). This preference is attested by English speakers showing processing difficulty

[10] https://github.com/plogacev/manuscript_LogacevVasishth_TQJEP_Underspecification/tree/master/Data_Swets_et_al

when faced with a disambiguating noun that contradicts their ~~initial analysis~~ preference. I'm trying to be neutral regarding the reason, such as garden path: Frazier and Rayner (1982); surprisal: Levy (2008); etc. We want to examine if the data at hand provide evidence for N2 attachment preference.

- We first load the data and clean it a bit:

```
swetsetal2008 <- read.table("data/Data_SwetsEtAl2008.csv", sep = ";",
    header = T)
head(swetsetal2008)
```

```
##   sub item trial cond..for.coding.purposes.in.column.G.
## 1   6    1   107                                      6
## 2   6    2    12                                      6
## 3   6    3   110                                      6
## 4   6    4    71                                      6
## 5   6    5     8                                      6
## 6   6    6    99                                      6
##   resp..1...yes..2...no. Q.answer.time
## 1                      1          2089
## 2                      1          3286
## 3                      1          1779
## 4                      1          1609
## 5                      1          5827
## 6                      1          4270
##  correct.inc..1...N1.interpretation.for.ambiguous.RC.questions.and.correct.answers.fo
## 1
## 2
## 3
## 4
## 5
## 6
##  qtype..1...detailed.questions..2...superficial.questions..3...occasional.superficial
## 1                                                                                    1
## 2                                                                                    1
## 3                                                                                    1
## 4                                                                                    1
## 5                                                                                    1
```

```
## 6                                                                    1
##   amb..1...ambiguous..2...N1.attachment..3...N2.attachment.
## 1                                                         3
## 2                                                         3
## 3                                                         3
## 4                                                         3
## 5                                                         3
## 6                                                         3
##   qtype2..1...question.about.N1..2...question.about.N2. the  n1  of the.1
## 1                                                     2 559 638 719   483
## 2                                                     2 554 550 477   563
## 3                                                     2 563 622 641   727
## 4                                                     2 463 700 712   495
## 5                                                     2 552 529 733   600
## 6                                                     2 539 583 484   433
##    n2  who sub.verb reflexive region9 matrix.verb  w11  w12  w13  w14 w15
## 1 541 2075      659       707    2379         618  744 1301   NA   NA  NA
## 2 700 1392      788      1269    2605         509 1551 1550 5985   NA  NA
## 3 745 1021     7721       538    1643         590  788  764  414  423 691
## 4 465  524      720       726    1602         555  794  533  470  766  NA
## 5 546  513      681      1200    1220         550  657  595  779 2123  NA
## 6 466 1445      546      1186     585         603  582  580  535 1560  NA
##   w16 w17 w18
## 1  NA  NA  NA
## 2  NA  NA  NA
## 3 796  NA  NA
## 4  NA  NA  NA
## 5  NA  NA  NA
## 6  NA  NA  NA
```

```
# We save this horribly long column name in another column
swetsetal2008$condition <- swetsetal2008$amb..1...ambiguous..2...N1.attachment..3...N2.a

# We remove the globally ambiguous sentences from our data
N1N2 <- swetsetal2008[swetsetal2008$condition != 1, ]

# We'll save RTs at 'reflexive' in a column with a clearer
```

```
# name:
N1N2$RT <- N1N2$reflexive

# Subset some subjects (to decrease computation time)
N1N2 <- N1N2[N1N2$sub <= 80, ]

# Total number of subjects:
length(unique(N1N2$sub))
```

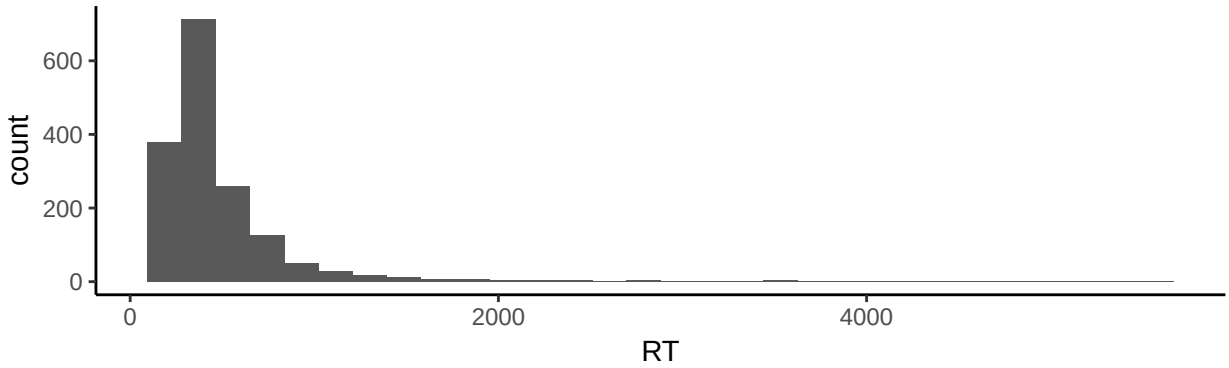
```
## [1] 68
```

We can inspect a bit the data:

```
library(ggplot2)
# Let's also see how the data is distributed
quantile(N1N2$RT)
```

```
##   0%  25%  50%  75% 100%
##  133  286  381  545 5524
```

```
ggplot(N1N2, aes(RT)) + geom_histogram()
```



We see that the data don't look normally distributed at all.

We also take a look at the average RTs at the critical region conditional on the attachment site.

```
# Show times at the reflexive verb for the two conditions 2
# is N1 attachment, 3 is N2 attachment
aggregate(RT ~ condition, data = N1N2, mean)
```

```
##   condition  RT
## 1         2 527
```

```
## 2         3 469
```

We see that N1 attachment causes longer RTs at the reflexive verb. Is it just in our sample because of noise? We want to examine the evidence that our data (and our model) presents for N1 attachment causing increased processing difficulty in comparison with N2 attachment.

- The data that we have consists of:
  - Reading times at the reflexive pronoun.
  - Two conditions: N1 (high) and N2 (low) attachment.
  - Number of observations (`length(N1N2$RT)`): 1632
  - Number of subjects (`length(unique(N1N2$sub))`): 68
  - Number of items (`length(unique(N1N2$item))`): 36

How should we model the data? Which are sensible assumptions? We'll build different models to analyze these data.

3.2.1 Complete pooling or fixed effects model (M_{cp})

We'll start from the simplest model which is basically a linear regression with a log-transformed dependent variable. Note that this model is incorrect for these data due to point 2 below.

- Model M_{cp} assumptions:

1. RTs are log-normally distributed.
2. Observations are independent.
3. There can be a difference in RTs between the sentences with N1 attachment and N2 attachment.
4. Likelihood:

$$RT_n \sim LogNormal(\mu_n, \sigma) \tag{232}$$

with $\mu_n = \alpha + x_n \cdot \beta$ and $n = 1, 2, \ldots, N$ observations

- n represents each observation, the nth row in the data frame
- $exp(\alpha)$ is the median of the log-normal distribution
- We use sum coding for the effect of attachment so that x_n is 1 for N1 attachment and -1 for N2 attachment:

```
N1N2$x <- ifelse(N1N2$condition == 2, 1, -1)
```

- $\exp(\alpha)$ is the median of the log-normal distribution when we average out the effect of attachment, β, in the log-scale. α is generally called the intercept.
- β represents the effect of the attachment site on the location of the distribution in log-scale. The effect in milliseconds is not $exp(\beta)$! If we want to know the difference in milliseconds between the two conditions, we need to take into account the intercept: $exp(\alpha+\beta)-exp(\alpha-\beta)$.

Why do we need to do $exp(\alpha+\beta)-exp(\alpha-\beta)$?

This is because on average (according to the geometric mean) our model behaves as follows:

$$log(RT_{condition}) = \alpha + x_{condition} \cdot \beta \tag{233}$$

And depending on the condition,
(a) $log(RT_{condition=N1}) = \alpha + x_{N1} \cdot \beta = \alpha + (+1) \cdot \beta$
(b) $log(RT_{condition=N2}) = \alpha + x_{N2} \cdot \beta = \alpha + (-1) \cdot \beta$
Exponentiating we get
(a) $exp(log(RT_{condition=N1})) = RT_{condition=N1} = exp(\alpha+\beta)$
(b) $exp(log(RT_{condition=N2})) = RT_{condition=N2} = exp(\alpha-\beta)$
Thus:

$$RT_{condition=N1} - RT_{condition=N2} = exp(\alpha+\beta) - exp(\alpha-\beta) \tag{234}$$

5. Priors:

$$\begin{aligned} \alpha &\sim Normal(0,10) \\ \beta &\sim Normal(0,1) \\ \sigma &\sim Normal(0,1) \text{ for } \sigma > 0 \end{aligned} \tag{235}$$

Pay attention to the scale we are using: α, β, and σ will be parameters inside a log-normal distribution, and thus they are not representing milliseconds. What do we know about the parameters according to these priors? (See the notes of the previous session.)

A model such as M_{cp} is sometimes called a fixed-effects model: all the parameters are fixed and do not vary from subject to subject or from item to item.[11]

This model is almost identical to the one we dealt at the end of the last session. But I'm introducing a new block as well, `transformed parameters`. Here I define μ which is not

[11] A similar frequentist model would correspond to fitting a simple linear model using the `lm` function: `lm(log(RT) ~ x, data=N1N2)`.

really a free parameter and depends completely on the values of α and β. Notice that I'm also generating the estimate of the overall difference between the two conditions `overall_difference` in milliseconds.

```
data {
  int<lower=1> N;
  vector[N] RT;
  vector<lower=-1,upper=1>[N] x;
}
parameters {
  real<lower=0> sigma;
  real alpha;
  real beta;
}
transformed parameters {
  vector[N] mu;
  for(n in 1:N){
    mu[n] = alpha + x[n] * beta;
  }
}
model {
  alpha ~ normal(0,10);
  beta  ~ normal(0,1);
  sigma ~ normal(0,1);
  for(n in 1:N){
    RT[n] ~ lognormal(mu[n], sigma);
  }
}
generated quantities {
  real overall_difference;
  overall_difference = exp(alpha + beta) - exp(alpha - beta);
}
```

The following code does the same but it's clearer and slightly faster. Stan allows us to omit for-loops if we deal with vectors.[12]

[12]Maybe you heard that you should never use for-loops in R; in Stan the situation is not that bad, since the code is compiled in C++. However, vectorized code is usually simpler, less prone to errors, and could be slightly faster.

```
data {
  int<lower=1> N;
  vector[N] RT;
  vector<lower=-1,upper=1>[N] x;
}
parameters {
  real<lower=0> sigma;
  real alpha;
  real beta;
}
transformed parameters {
  vector[N] mu;
  mu = alpha + x * beta;
}
model {
  alpha ~ normal(0,10);
  beta  ~ normal(0,1);
  sigma ~ normal(0,1);
  RT ~ lognormal(mu, sigma);
}
generated quantities {
  real overall_difference;
  overall_difference = exp(alpha + beta) - exp(alpha - beta);
}
```

We save the previous code as a text file called `Mcp.stan`. And we create a list to fit the data in `rstan`.

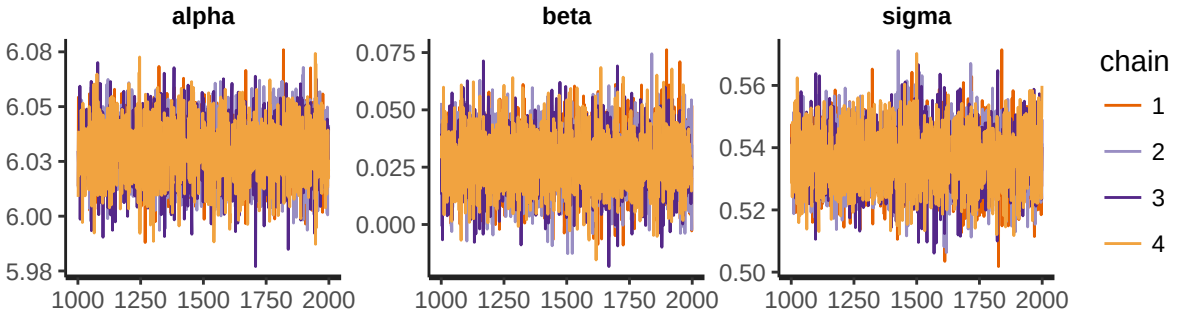
```
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
# We define a list with the data, notice that the names in
# the list need to match with the names in the data block of
# the model (case included)
Swets_list <- list(RT = N1N2$RT, x = N1N2$x, N = nrow(N1N2))
# We fit the model here.
fit_Mcp <- stan(file = "Mcp.stan", data = Swets_list)
```

If you did everything right,[13] you should be able to print the summary for the parameters and the difference between conditions. Verify that the model converged by looking at the warning messages, examining the `Rhat`s and `n_eff`, and looking at the `traceplot`.

```
print(fit_Mcp, probs = c(0.025, 0.5, 0.975), pars = c("alpha",
    "beta", "sigma", "overall_difference"), digits = 3)
```

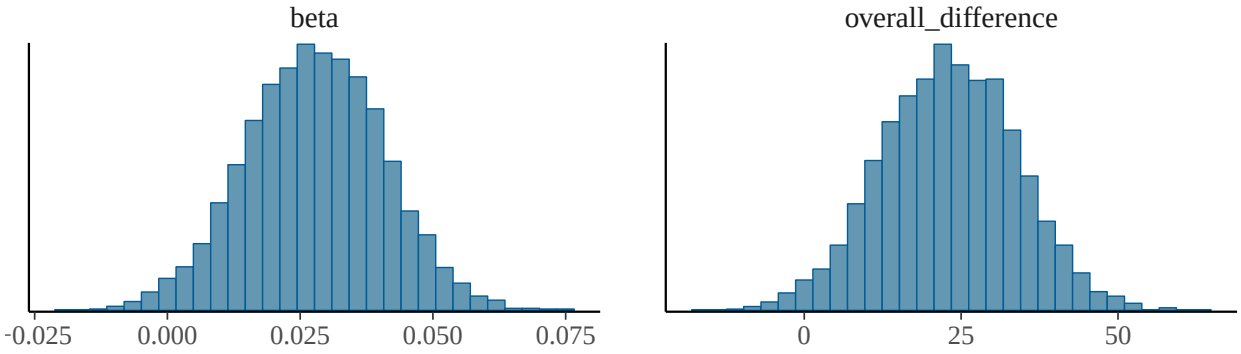
```
## Inference for Stan model: Mcp.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                      mean se_mean     sd  2.5%    50%  97.5% n_eff  Rhat
## alpha               6.029   0.000  0.013 6.003  6.029  6.055  4000 1.000
## beta                0.028   0.000  0.013 0.002  0.028  0.052  4000 1.000
## sigma               0.536   0.000  0.009 0.519  0.536  0.555  4000 0.999
## overall_difference 23.063   0.171 10.803 1.569 23.074 43.600  4000 1.000
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:47:29 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
traceplot(fit_Mcp, pars = c("alpha", "beta", "sigma"))
```




```
library(bayesplot)
posterior_Mcp <- as.array(fit_Mcp)
mcmc_hist(posterior_Mcp, pars = c("beta", "overall_difference"))
```

[13] Yes, you will get some exceptions but as long as the count << number of iterations, everything is fine.



We see that β seems to be different than 0. How sure can we be that this is so?

```
# Proportion of samples above 0 should be similar to the
# proportion of the posterior distribution above 0.
mean(rstan::extract(fit_Mcp)$beta > 0)
```

```
## [1] 0.983
```

3.2.2 No pooling model (M_{np})

One of the assumptions of the previous model is clearly wrong, observations are not independent. Observations depend on the participant, for example, faster participants will generally answer faster. The no pooling model assumes that each participant is completely independent from each other.[14]

- Model M_{np} assumptions:

1. RTs are log-normally distributed.
2. Observations depend completely on the participant. (Participants have nothing in common.)
3. For every participant, there can be a difference in RTs between the sentences with N1 attachment and N2 attachment.
4. Likelihood:

$$RT_n \sim LogNormal(\mu_n, \sigma_{i[n]}) \tag{236}$$

with $\mu_n = \alpha_{i[n]} + x_n \cdot \beta_{i[n]}$

5. Priors:

$$\alpha_i \sim Normal(0, 10)$$

[14] There is a parallel frequentist implementation in R called `lmlist`.

$$\beta_i \sim Normal(0,1)$$

$$\sigma_i \sim Normal(0,1) \text{ for } \sigma_n > 0$$

- n represents each obs, the nth row in the dataframe
- i represents each participant (our cluster), and $i[n]$ is the participant that corresponds to observation n. This follows Gelman and Hill (2007)'s notation.

Before we discuss the Stan implementation, let's see how the vector of μ's look like. There are in total 1632 observations, that means that $\mu = \{\mu_1, \mu_2, \ldots, \mu_{1632}\}$, and we have 68 participants which means that $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_{68}\}$ (and similarly for β and σ). There are also 24 observations for each subject,[15] so from $\alpha_{i[1]}$ to $\alpha_{i[24]}$ we are actually accessing to α associated with subject 1, that is, α_1, and from $\alpha_{i[25]}$ to $\alpha_{i[48]}$ we are accessing α_2, and so forth.

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \ldots \\ \mu_7 \\ \mu_8 \\ \ldots \\ \mu_{24} \\ \mu_{25} \\ \mu_{26} \\ \ldots \\ \mu_{73} \\ \ldots \\ \mu_{1632} \end{bmatrix} = \begin{bmatrix} \alpha_{i[1]} \\ \alpha_{i[2]} \\ \ldots \\ \alpha_{i[7]} \\ \alpha_{i[8]} \\ \ldots \\ \alpha_{i[24]} \\ \alpha_{i[25]} \\ \alpha_{i[26]} \\ \ldots \\ \alpha_{i[73]} \\ \ldots \\ \alpha_{i[1632]} \end{bmatrix} + \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_7 \\ x_8 \\ \ldots \\ x_{24} \\ x_{25} \\ x_{26} \\ \ldots \\ x_{73} \\ \ldots \\ x_{1632} \end{bmatrix} \cdot \begin{bmatrix} \beta_{i[1]} \\ \beta_{i[2]} \\ \ldots \\ \beta_{i[7]} \\ \beta_{i[8]} \\ \ldots \\ \beta_{i[24]} \\ \beta_{i[25]} \\ \beta_{i[26]} \\ \ldots \\ \beta_{i[73]} \\ \ldots \\ \beta_{i[1632]} \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_1 \\ \ldots \\ \alpha_1 \\ \alpha_1 \\ \ldots \\ \alpha_1 \\ \alpha_2 \\ \alpha_2 \\ \ldots \\ \alpha_4 \\ \ldots \\ \alpha_{68} \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \\ \ldots \\ 1 \\ 1 \\ \ldots \\ -1 \\ 1 \\ 1 \\ \ldots \\ 1 \\ \ldots \\ -1 \end{bmatrix} \circ \begin{bmatrix} \beta_1 \\ \beta_1 \\ \ldots \\ \beta_1 \\ \beta_1 \\ \ldots \\ \beta_1 \\ \beta_2 \\ \beta_2 \\ \ldots \\ \beta_4 \\ \ldots \\ \beta_{68} \end{bmatrix} \tag{237}$$

with $\circ$ indicating an element-wise multiplication. This is done with `.*` in Stan but just `*` in R.

The code of the no pooling model (M_{np}) appears below.

```
data {
  int<lower = 1> N;
  vector[N] RT;
  vector<lower = -1, upper = 1>[N] x;
  int<lower = 1> N_subj;
```

[15] There are 36 items, but we removed the ones associated with the third condition of the actual experiment.

```
  // The following line creates an array of integers;
  int<lower = 1, upper = N_subj> subj[N];
}
parameters {
  vector<lower = 0>[N_subj] sigma;
  vector[N_subj] alpha;
  vector[N_subj] beta;
}
transformed parameters {
  vector[N] mu;
    mu = alpha[subj] + x .* beta[subj];
    // .* indicates an element-wise multiplication (in contrast with a matrix
    // multiplication).

    // Notice that the vectors alpha and beta have N_subj elements, but the
    // vectors alpha[subj] and beta[subj] have N elements. This is because we
    // are using the content of subj (integers) as indexes for these vectors,
    // and subj has N elements.

    // The whole row is equivalent to:
    // for(n in 1:N) mu[n] = alpha[subj[n]] + x[n] * beta[subj[n]];
}
model {
  alpha ~ normal(0, 10);
  beta  ~ normal(0, 1);
  sigma ~ normal(0, 1);
  // The paremeters are vectors now, it is equivalent to:
  // for(i in 1:N_subj){
  //  alpha[i] ~ normal(0, 10);
  //  beta[i]  ~ normal(0, 1);
  //  sigma[i] ~ normal(0, 1);
  //}
  RT ~ lognormal(mu, sigma[subj]);
}
generated quantities {
  vector[N_subj] difference_by_subj;
```

```
  real overall_difference;
  real average_beta;
  for(i in 1:N_subj){
   difference_by_subj[i] = exp(alpha[i] + beta[i]) - exp(alpha[i] - beta[i]);
  }
  overall_difference = mean(difference_by_subj);
  average_beta = mean(beta);
}
```

We have also some new Stan code:

- `int<lower = 1, upper = N_subj> subj[N];` defines a one-dimensional array of `N` elements that contains integers (bounded between 1 and `N_subj`). The difference between vectors and one-dimensional arrays is that vectors can only contain real numbers and can be used with matrix algebra functions, and arrays can contain any type but can't be used in matrix algebra.
- `mu = alpha[subj] + x .* beta[subj];` is summing two vectors of the same length as `mu` (`N`). The first summand is basically the α column in Eq. (237), the second summand is the element-wise multiplication of the α column in Eq. (237) with the vector `x`.

Save the model as `Mnp.stan`.

Now the model doesn't assume only one effect, but a different effect for each subject. We can then calculate the average of the β's, but the model doesn't assume that there's one common β.

Before we can fit the model, we'll need to add to the list that goes into `rstan`, the information about the subjects, namely, which observation corresponds to each subject (in `subj`) and the total number of subjects (in `N_subj`).

```
# This way I'm sure that the subjects are a sequence of
# numbers starting from 1
subj <- as.numeric(as.factor(N1N2$sub))
N_subj <- length(unique(subj))
# We add this to the previous list
Swets_list <- c(Swets_list, list(subj = subj, N_subj = N_subj))

fit_Mnp <- stan(file = "Mnp.stan", data = Swets_list)
```

As usual we inspect convergence:

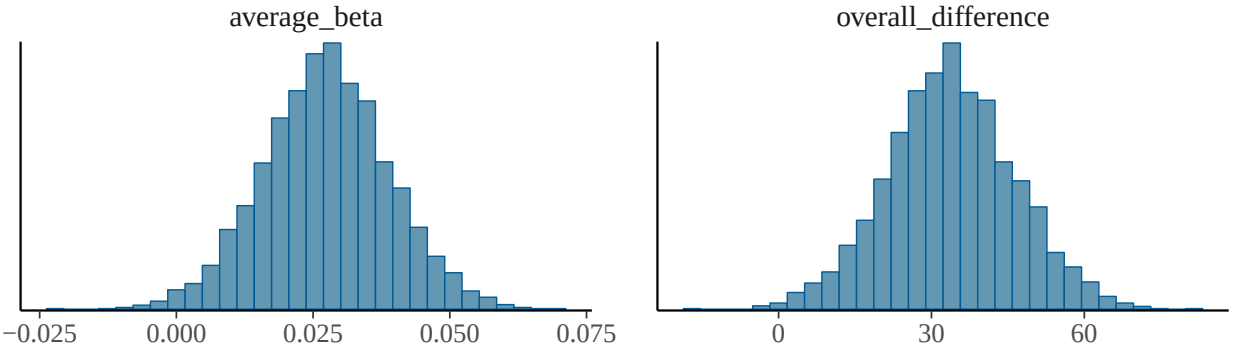
```
print(fit_Mnp, probs = c(0.025, 0.5, 0.975), pars = c("alpha",
    "beta", "sigma"))
traceplot(fit_Mnp, pars = c("alpha", "beta", "sigma"))
```

And we'll examine the marginal posterior, and the posteriors of our generated quantities.

```
print(fit_Mnp, probs = c(0.025, 0.5, 0.975), pars = c("overall_difference",
    "average_beta"), digits = 3)
```

```
## Inference for Stan model: Mnp.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                      mean se_mean     sd  2.5%    50% 97.5% n_eff Rhat
## overall_difference 33.853   0.198 12.512 8.885 33.782 58.93  4000    1
## average_beta        0.027   0.000  0.012 0.004  0.027  0.05  4000    1
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:48:09 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
# Let's look at our generated quantities
posterior_Mnp <- as.array(fit_Mnp)
mcmc_hist(posterior_Mnp, pars = c("average_beta", "overall_difference"))
```



We can also calculate how certain we are that the average effect is larger than 0.

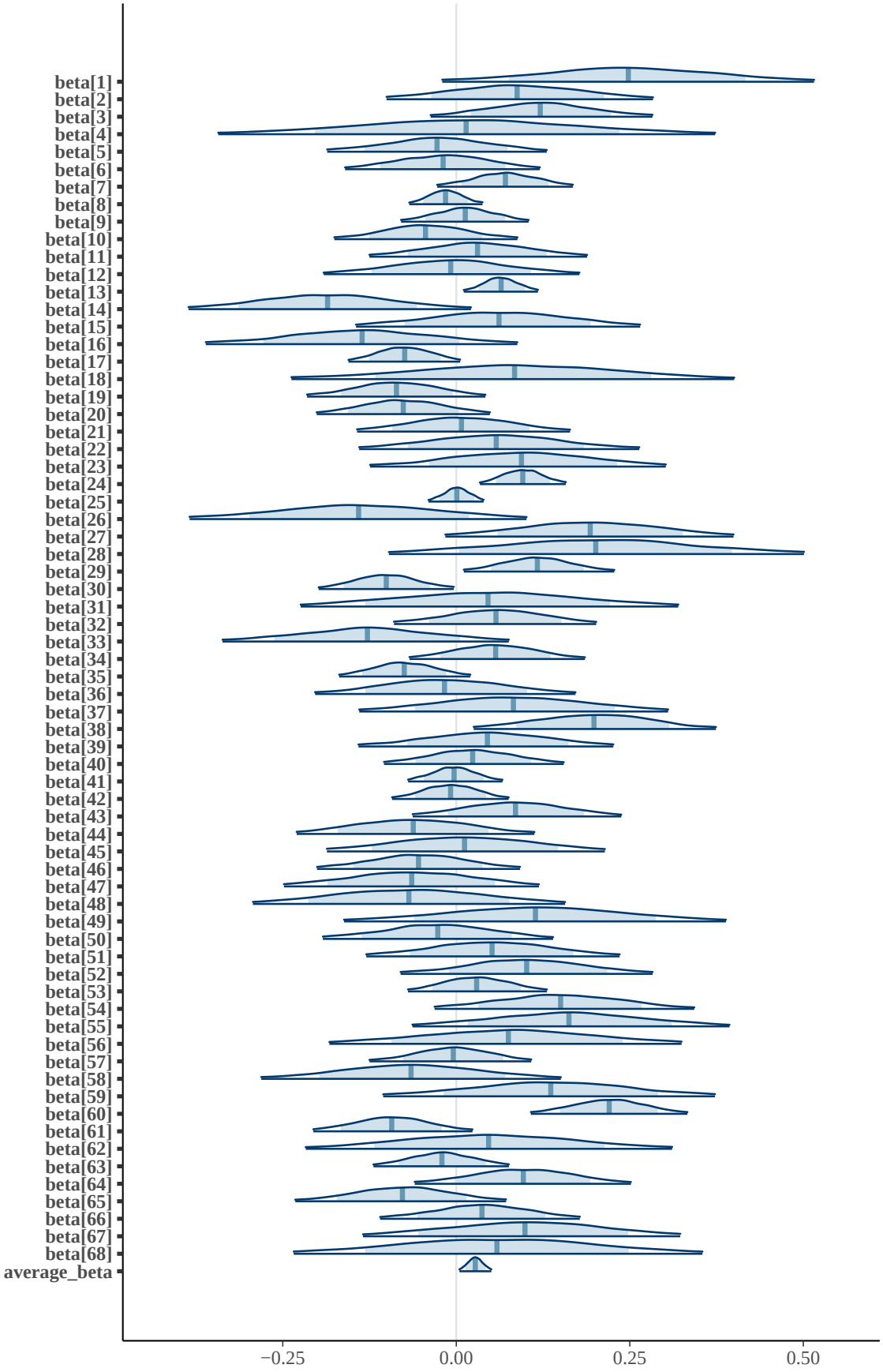
```
mean(rstan::extract(fit_Mnp)$average_beta > 0)
```

```
## [1] 0.989
```

We might have built a model like this, because we assume that participants are different,

and that's what we want to examine.

```
# We plot the 95% and 80% areas of the posterior
# distributions of the 'beta' generated quantity. Notice
# that we use regex_pars, this is because the columns of
# as.array(fit_Mvi) are beta[1], beta[2], etc. With
# regex_pars we match all the beta's.
mcmc_areas(posterior_Mnp, regex_pars = c("beta"), prob = 0.8,
    prob_outer = 0.95, point_est = "mean")
```


beta[1]
beta[2]
beta[3]
beta[4]
beta[5]
beta[6]
beta[7]
beta[8]
beta[9]
beta[10]
beta[11]
beta[12]
beta[13]
beta[14]
beta[15]
beta[16]
beta[17]
beta[18]
beta[19]
beta[20]
beta[21]
beta[22]
beta[23]
beta[24]
beta[25]
beta[26]
beta[27]
beta[28]
beta[29]
beta[30]
beta[31]
beta[32]
beta[33]
beta[34]
beta[35]
beta[36]
beta[37]
beta[38]
beta[39]
beta[40]
beta[41]
beta[42]
beta[43]
beta[44]
beta[45]
beta[46]
beta[47]
beta[48]
beta[49]
beta[50]
beta[51]
beta[52]
beta[53]
beta[54]
beta[55]
beta[56]
beta[57]
beta[58]
beta[59]
beta[60]
beta[61]
beta[62]
beta[63]
beta[64]
beta[65]
beta[66]
beta[67]
beta[68]
average_beta
−0.25
0.00
0.25
0.50

Or we may want to see how many of the subjects have a mean effect that is positive:

```
betas <- rstan::extract(fit_Mnp)$beta
# Each column represents a participant, each row a sample:
dim(rstan::extract(fit_Mnp)$beta)
```

```
## [1] 4000   68
```

```
# We check for each column (participant), the proportion of
# samples above 0
mean(colMeans(betas) > 0)
```

```
## [1] 0.603
```

Or more "Bayesian-ly" we may want to know how many lower intervals of the 95% CrI of the participants are above 0:

```
# We apply for each column (2), the function
# quantile(x,.025)
lowerCrI <- apply(betas, 2, function(x) quantile(x, 0.025))
# We check the proportion of columns with lower quantiles
# (.025) above 0
mean(lowerCrI > 0)
```

```
## [1] 0.0735
```

Let's see now what can we learn from models with more realistic assumptions.

3.2.3 Varying intercept model (M_{vi})

We see that one main problem with the no pooling model is that it's not modeling what we care about (the overall effect)! (We just averaged participant's effects to obtain it, but the model didn't assume any commonality between participants). Another problem is in the estimation: We ignore completely that the participants were after all doing the same experiment. We fit each subject's data ignoring the information available in the other subjects' data. The previous model is very likely to overfit the data, we are likely to ignore the generalities of the data and we may end up modeling the noise.

A sensible assumption (that will improve the estimation) is that there are commonalities between the subjects. This will result in the estimation of posteriors for each participant being also influenced by what we know about all the participants. We'll fit first the simplest

kind of hierarchical model, a varying intercepts model.[16] This model assumes that only the intercept is affected by the participant, but that all the participants are affected equally by the experimental manipulation.

- Model M_{vi} assumptions:

1. RTs are log-normally distributed.
2. Some aspects of the reading speed depend on the participant.
3. There can be a difference in RTs between the sentences with N1 attachment and N2 attachment.
4. Likelihood:

$$RT_n \sim LogNormal(\mu_n, \sigma) \tag{238}$$

with $\mu_n = \alpha + u_{i[n]} + x_n \cdot \beta$

5. Priors:

$$\begin{aligned} \alpha &\sim Normal(0, 10) \\ \beta &\sim Normal(0, 1) \\ \sigma &\sim Normal(0, 1) \text{ for } \sigma > 0 \\ \tau_u &\sim Normal(0, 1) \text{ for } \tau_u > 0 \\ u_i &\sim Normal(0, \tau_u) \end{aligned} \tag{239}$$

- n represents each obs, the nth row in the data frame.
- i represents each subj, and $i[n]$ is the subject that corresponds to observation n, following Gelman and Hill (2007)'s notation.

In this model each subject has his or her own adjustment u_i, if u_i is positive, the subject will be slower than the average subject, if it's negative he or she will be faster. Notice that since we are estimating α and u at the same time and we assume that the average of the u's is 0 (since it is assumed to be normally distributed with mean of 0), whatever the subjects have in common "goes" to α, and u only "absorbs" the differences between participants through the variance component τ_u. This model has then two variance components: σ and τ_u. Parameters of parameters such as τ_u are often called hyperparameters and their priors, hyperpriors.

Some important (and sometimes confusing) points:

[16] You can fit a parallel frequentist model with `lmer` from the package `lme4`, using `(1|subj)` for the random effects.

- Why does u have a mean of 0?

Because we want u to capture only differences between subjects, we could achieve the same by assuming that $\mu_n = \alpha_{i[n]} + \beta \cdot x_n$ and $\alpha_i \sim Normal(\alpha, \tau_u)$; $\alpha \sim Normal(0, 10)$. And in fact, that's another common way to write the model.

- Why do the adjustments u have a normal distribution?

Mostly because of "convention", that's the way it's implemented in most frequentist mixed models. But also because if we don't know anything about the distribution besides its mean and variance, the normal distribution is the most conservative assumption (see also chapter 9 of McElreath 2015).

We can implement this in Stan in the following way:

```
data {
  int<lower = 1> N;
  vector[N] RT;
  vector<lower = -1, upper = 1>[N] x;
  int<lower = 1> N_subj;
  int<lower = 1, upper = N_subj> subj[N];
}
parameters {
  real<lower = 0> sigma;
  real<lower = 0> tau_u;
  real alpha;
  real beta;
  vector[N_subj] u;
}
transformed parameters {
  vector[N] mu;
  mu = alpha + u[subj] +  x * beta;
  // In this model, alpha and beta are real, not vectors.
  // The whole row is equivalent to:
  // for(n in 1:N) mu[n] = alpha + u[subj[n]] + x[n] * beta;
}

model {
  alpha ~ normal(0, 10);
  beta  ~ normal(0, 1);
```

```
  sigma ~ normal(0, 1);
  tau_u ~ normal(0, 1);
  // Now u is a vector with N_subj elements
  u ~ normal(0, tau_u);
  RT ~ lognormal(mu, sigma);

}
generated quantities {
  real overall_difference;
  vector[N_subj] difference_by_subj;
  for(i in 1:N_subj){
   difference_by_subj[i] = exp(alpha + u[i] + beta ) - exp(alpha + u[i] - beta);
  }
  overall_difference = exp(alpha + beta) - exp(alpha - beta);
}
```

We can get the difference between high and low (N1 vs. N2) attachment in ms for each subject by generating `overall_difference`.

Save the model as `Mvi.stan`.

```
fit_Mvi <- stan(file = "Mvi.stan", data = Swets_list)
```

As usual we inspect convergence:

```
print(fit_Mvi, probs = c(0.025, 0.5, 0.975), pars = c("alpha",
    "beta", "sigma", "tau_u"))
traceplot(fit_Mvi, pars = c("alpha", "beta", "sigma", "tau_u"))
```

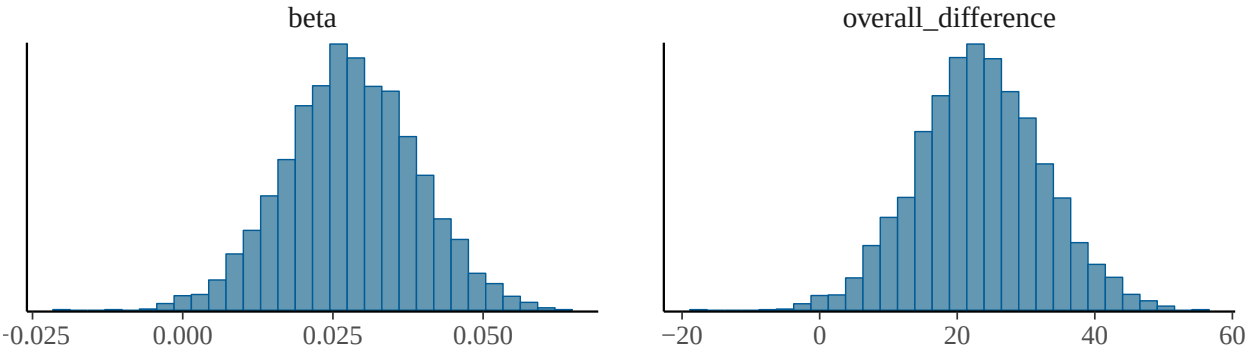
And we'll examine the marginal posterior, and the posteriors of our generated quantities.

```
print(fit_Mvi, probs = c(0.025, 0.5, 0.975), pars = c("beta",
    "overall_difference"), digits = 3)
```

```
## Inference for Stan model: Mvi.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                      mean se_mean    sd  2.5%    50%  97.5% n_eff  Rhat
## beta                0.028   0.000 0.011 0.007  0.028  0.049  4000 1.000
## overall_difference 23.294   0.145 9.175 5.480 23.224 41.891  4000 0.999
```

```
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:48:33 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
posterior_Mvi <- as.array(fit_Mvi)

mcmc_hist(posterior_Mvi, pars = c("beta", "overall_difference"))
```



We can also calculate how certain we are that the effect across subjects is larger than 0.

```
mean(rstan::extract(fit_Mvi)$beta > 0)
```

```
## [1] 0.994
```

3.2.4 (Uncorrelated) varying intercept varying slopes model (M_{uvivs})

Even though we may only care about β, that is the effect of the manipulation in general and not on specific subjects, the previous model makes the strong assumption that every participant will be affected equally by the manipulation (on the log-scale). One problem with this approach is that if a few subjects are strongly affected by the manipulation while the rest aren't, we will think that all of them are affected. In addition, we may be interested in the differences between the participants: are all participants equally affected? How general is the effect of attachment site in the population? We can relax the strong assumption of only one effect by letting the effect vary by participants. This is also referred to as adding a varying slope (since in the linear model the effect is the slope of the line that represents the fit).

- Model M_{uvivs} assumptions:

1. RTs are log-normally distributed.

2. Some aspects of the reading speed and the effect of the site of attachment depend on the participant. But these two aspects are unrelated.
3. There can be a difference in RTs between the sentences with N1 attachment and N2 attachment.
4. Likelihood:

$$RT_n \sim LogNormal(\mu_n, \sigma) \tag{240}$$

with $\mu_n = \alpha + u_{1i[n]} + x_n \cdot (\beta + u_{2i[n]})$

5. Priors:

$$\begin{aligned} \alpha &\sim Normal(0, 10) \\ \beta &\sim Normal(0, 1) \\ \sigma &\sim Normal(0, 1) \text{ for } \sigma > 0 \\ \tau_{u_1} &\sim Normal(0, 1) \text{ for } \tau_{u_1} > 0 \\ \tau_{u_2} &\sim Normal(0, 1) \text{ for } \tau_{u_2} > 0 \\ u_{1i} &\sim Normal(0, \tau_{u_1}) \\ u_{2i} &\sim Normal(0, \tau_{u_2}) \end{aligned} \tag{241}$$

- n represents each obs, the nth row in the data frame
- i represents each subj, and $i[n]$ is the subject that corresponds to observation n.

In this model, while τ_{u_1} represents the variation on the intercept (general speed) across subjects, τ_{u_2} represents the variation on the effect of the manipulation across subjects.

We can implement this in Stan in the following way:

```
data {
  int<lower = 1> N;
  vector[N] RT;
  vector<lower = -1, upper = 1>[N] x;
  int<lower = 1> N_subj;
  int<lower = 1, upper = N_subj> subj[N];
}
parameters {
  real<lower = 0> sigma;
  real<lower = 0> tau_u1;
  real<lower = 0> tau_u2;
```

```
  real alpha;
  real beta;
  vector[N_subj] u1;
  vector[N_subj] u2;
}
transformed parameters {
  vector[N] mu;
  mu = alpha + u1[subj] +  x .* (beta + u2[subj]);
  // The whole row is equivalent to:
 // for(n in 1:N) mu[n] = alpha + u1[subj[n]] + x[n] * beta + x[n] * u2[subj[n]];
}

model {
  alpha ~ normal(0, 10);
  beta  ~ normal(0, 1);
  sigma ~ normal(0, 1);
  tau_u1 ~ normal(0, 1);
  tau_u2 ~ normal(0, 1);
  // Now u1 and u2 are vectors with N_subj elements
  u1 ~ normal(0, tau_u1);
  u2 ~ normal(0, tau_u2);
  RT ~ lognormal(mu, sigma);
}
generated quantities {
  real overall_difference;
  vector[N_subj] difference_by_subj;
  for(i in 1:N_subj){
    difference_by_subj[i] = exp(alpha + u1[i] + (beta + u2[i])) -
                            exp(alpha + u1[i] - (beta+ u2[i]));
  }
  overall_difference = exp(alpha + beta) - exp(alpha - beta);
}
```

Save the model as `Muvivs.stan`.

```
fit_Muvivs <- stan(file = "Muvivs.stan", data = Swets_list)
```

```
## Warning: There were 4 chains where the estimated Bayesian Fraction of Missing Information
```

```
## http://mc-stan.org/misc/warnings.html#bfmi-low
```

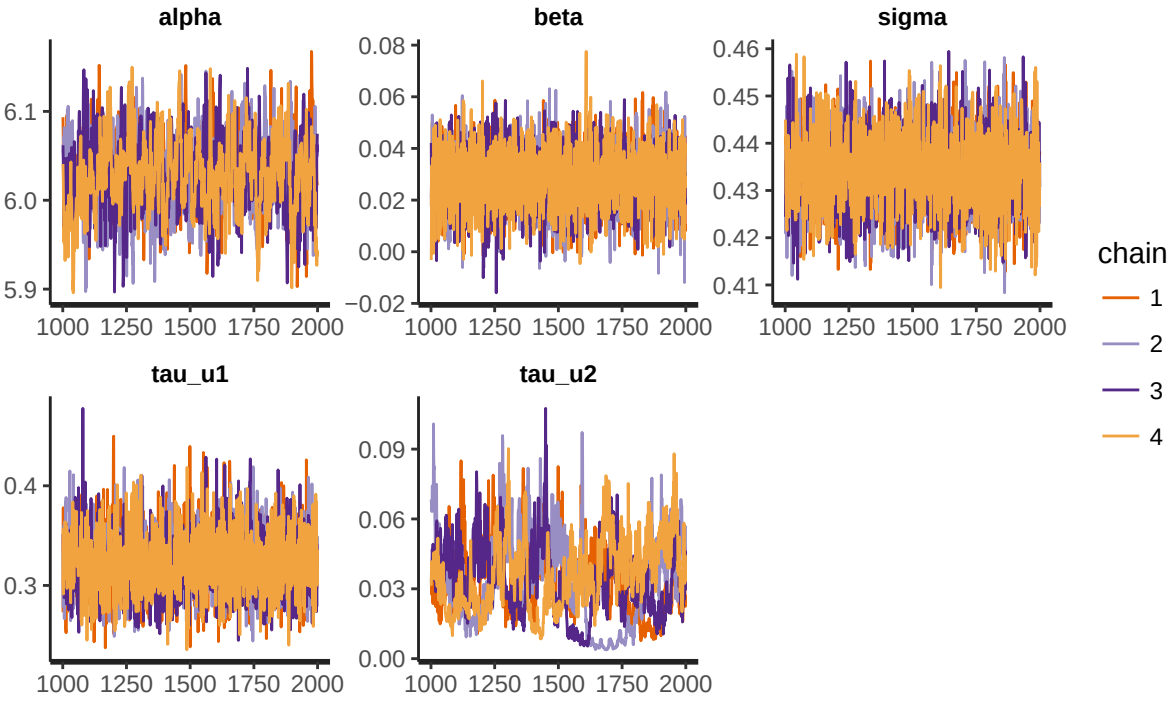
```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

We see that there are warnings. As we increase the complexity and the number of parameters, the sampler has a harder time exploring the parameter space:

```
print(fit_Muvivs, probs = c(0.025, 0.5, 0.975), pars = c("alpha",
    "beta", "sigma", "tau_u1", "tau_u2"))
```

```
## Inference for Stan model: Muvivs.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##        mean se_mean   sd 2.5%  50% 97.5% n_eff Rhat
## alpha  6.03       0 0.04 5.95 6.03  6.11   503 1.01
## beta   0.03       0 0.01 0.01 0.03  0.05  4000 1.00
## sigma  0.43       0 0.01 0.42 0.43  0.45  4000 1.00
## tau_u1 0.32       0 0.03 0.27 0.32  0.39  4000 1.00
## tau_u2 0.03       0 0.02 0.01 0.03  0.07    57 1.08
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:48:54 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
traceplot(fit_Muvivs, pars = c("alpha", "beta", "sigma", "tau_u1",
    "tau_u2"))
```



There's something wrong in the estimation of `tau_u2`. This parameter is specially problematic because it is bounded by 0 (it's a standard deviation), there is not too much information about it (every subject is providing only one data point), and it is quite small. This makes the exploration of the sampler quite hard. There are two options, we might just remove the varying slope since it's not giving us much information anyway (and I would recommend this), or we can alleviate this problem by re-parameterizing the model. In general, this would be the trickiest part of modeling, and I suggest to read chapter 21 of Stan manual (Stan Development Team 2017) for more about re-parametrization. The following box explains the specific re-parametrization we use for the improved version of our Stan code.

A simple re-parametrization

The parser can explore the parameter space more easily if it doesn't have to worry about the scale. We want to assume the following

$$\mathbf{u}_2 \sim Normal(0, \tau_{u_2}) \tag{242}$$

where $\mathbf{u}_2$ is the column vector of u_{2i}'s.
We can transform u_2 to z-scores as follows

$$\mathbf{u}_{raw2} = \frac{\mathbf{u}_2 - 0}{\tau_{u_2}} \tag{243}$$

where

$$\mathbf{u}_{raw2} \sim Normal(0, 1) \tag{244}$$

Now $\mathbf{u}_{raw2}$ is easier to sample because it doesn't depend on another parameter and its scale is 1. We can derive the actual parameter we care about by doing the following

$$\mathbf{u}_2 = \mathbf{u}_{raw2} \cdot \tau_{u_2} \tag{245}$$

The following Stan code uses the previous re-parametrization.

```
data {
  int<lower= 1 > N;
  vector[N] RT;
  vector<lower = -1, upper = 1>[N] x;
  int<lower = 1> N_subj;
  int<lower = 1,upper=N_subj> subj[N];
}
parameters {
  real<lower = 0> sigma;
  real<lower = 0> tau_u1;
  real<lower = 0> tau_u2;
  real alpha;
  real beta;
  vector[N_subj] u1;
  vector[N_subj] u2_raw;
}
transformed parameters {
```

```
  vector[N] mu;
  vector[N_subj] u2;
  u2 = u2_raw * tau_u2;
  mu = alpha + u1[subj] +  x .* (beta + u2[subj]);
  // The whole row is equivalent to:
  // for(n in 1:N) mu[n] = alpha + u1[subj[n]] +
  //                       x[n] * beta + x[n] * u2[subj[n]];
}
model {
  alpha ~ normal(0, 10);
  beta  ~ normal(0, 1);
  sigma ~ normal(0, 1);
  tau_u1 ~ normal(0, 1);
  tau_u2 ~ normal(0, 1);
  // Now u1 and u2 are vectors with N_subj elements
  u1 ~ normal(0, tau_u1);
  u2_raw ~ normal(0, 1);
  RT ~ lognormal(mu, sigma);
}
generated quantities {
  real overall_difference;
  vector[N_subj] difference_by_subj;
  for(i in 1:N_subj){
    difference_by_subj[i] = exp(alpha + u1[i] + (beta + u2[i])) -
                            exp(alpha + u1[i] - (beta + u2[i]));
  }
  overall_difference = exp(alpha + beta) - exp(alpha - beta);
}
```

Save the above code as `Muvivs_reparam.stan`.

```
fit_Muvivs_reparam <- stan(file = "Muvivs_reparam.stan", data = Swets_list)
```

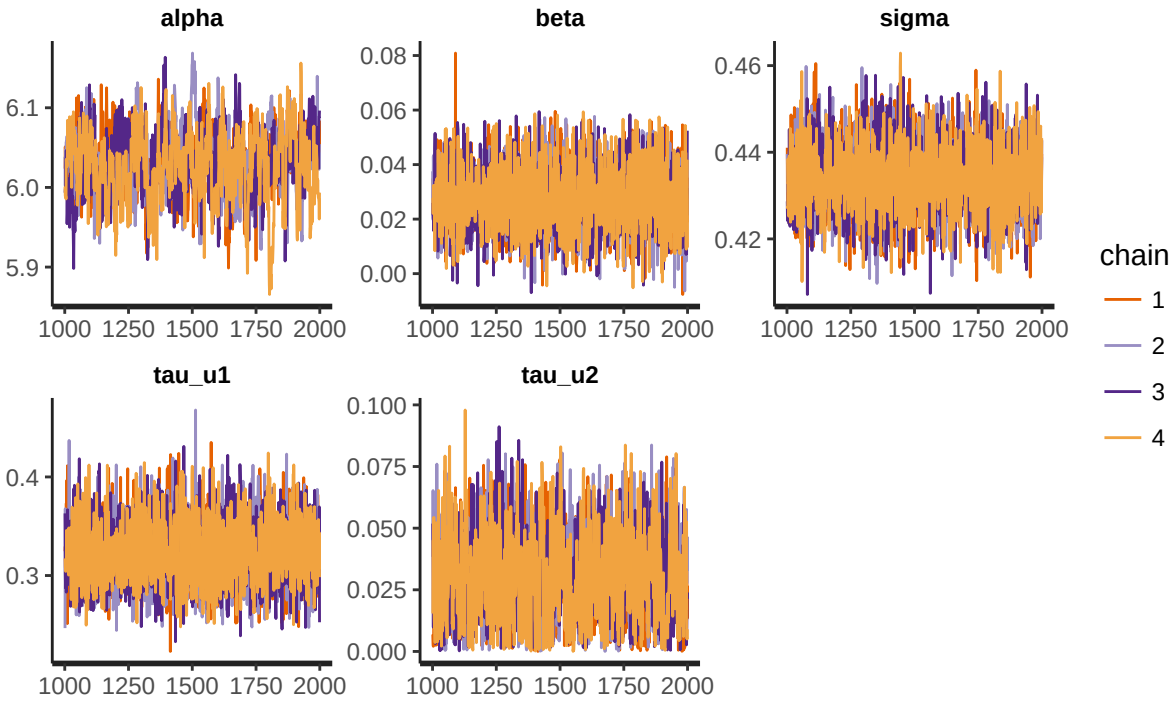
We see that the n_{eff} for `tau2` increased and traceplots look better.

```
print(fit_Muvivs_reparam, probs = c(0.025, 0.5, 0.975), pars = c("alpha",
    "beta", "sigma", "tau_u1", "tau_u2"))
```

```
## Inference for Stan model: Muvivs_reparam.
```

```
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##        mean se_mean   sd 2.5%  50% 97.5% n_eff Rhat
## alpha  6.03       0 0.04 5.95 6.03  6.11   352    1
## beta   0.03       0 0.01 0.01 0.03  0.05  4000    1
## sigma  0.43       0 0.01 0.42 0.43  0.45  4000    1
## tau_u1 0.32       0 0.03 0.27 0.32  0.39  4000    1
## tau_u2 0.03       0 0.02 0.00 0.03  0.07   903    1
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:49:18 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
traceplot(fit_Muvivs_reparam, pars = c("alpha", "beta", "sigma",
    "tau_u1", "tau_u2"))
```

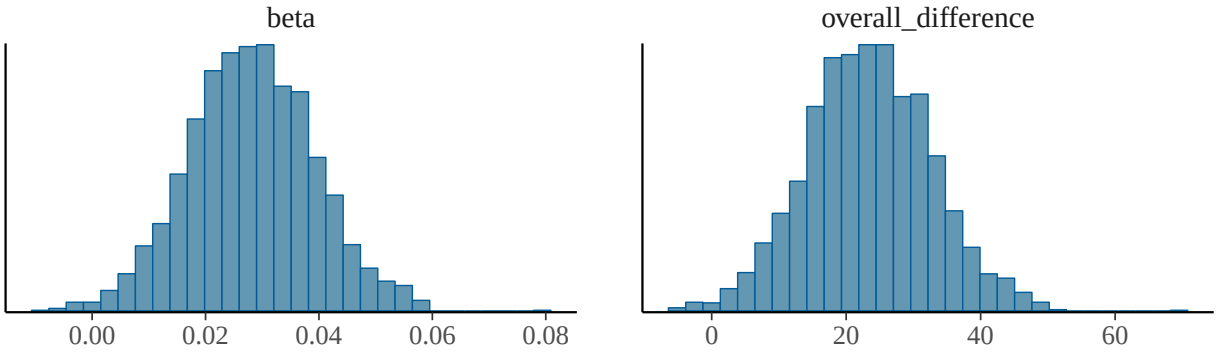


Only now we can examine the marginal posterior, and the posteriors of our generated quantities.

```
print(fit_Muvivs_reparam, probs = c(0.025, 0.5, 0.975), pars = c("beta",
    "overall_difference"), digits = 3)
```

```
## Inference for Stan model: Muvivs_reparam.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                       mean se_mean    sd  2.5%    50%  97.5% n_eff Rhat
## beta                 0.028   0.000 0.011 0.007  0.028  0.051  4000    1
## overall_difference 23.380   0.147 9.274 5.238 23.349 42.440  4000    1
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:49:18 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
posterior_Muvivs <- as.array(fit_Muvivs_reparam)

mcmc_hist(posterior_Muvivs, pars = c("beta", "overall_difference"))
```



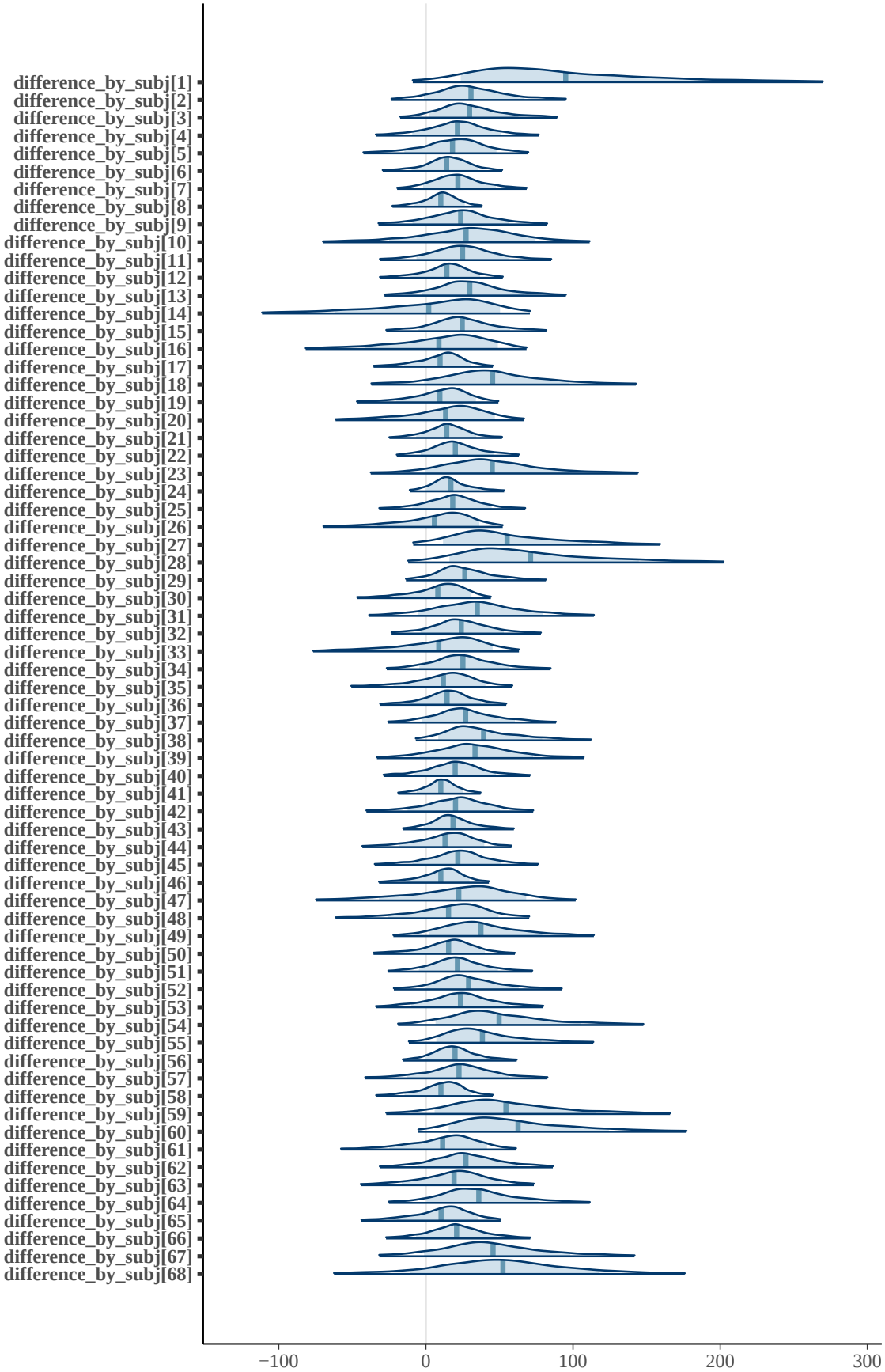
We can also calculate how certain we are that the common effect across subjects is larger than 0.

```
mean(rstan::extract(fit_Muvivs_reparam)$beta > 0)
```

```
## [1] 0.994
```

And we could examine how each subject is being affected by the manipulation (on the millisecond scale):

```
mcmc_areas(posterior_Muvivs, regex_pars = c("difference_by_subj"),
    prob = 0.8, prob_outer = 0.95, point_est = "mean")
```


difference_by_subj[1]
difference_by_subj[2]
difference_by_subj[3]
difference_by_subj[4]
difference_by_subj[5]
difference_by_subj[6]
difference_by_subj[7]
difference_by_subj[8]
difference_by_subj[9]
difference_by_subj[10]
difference_by_subj[11]
difference_by_subj[12]
difference_by_subj[13]
difference_by_subj[14]
difference_by_subj[15]
difference_by_subj[16]
difference_by_subj[17]
difference_by_subj[18]
difference_by_subj[19]
difference_by_subj[20]
difference_by_subj[21]
difference_by_subj[22]
difference_by_subj[23]
difference_by_subj[24]
difference_by_subj[25]
difference_by_subj[26]
difference_by_subj[27]
difference_by_subj[28]
difference_by_subj[29]
difference_by_subj[30]
difference_by_subj[31]
difference_by_subj[32]
difference_by_subj[33]
difference_by_subj[34]
difference_by_subj[35]
difference_by_subj[36]
difference_by_subj[37]
difference_by_subj[38]
difference_by_subj[39]
difference_by_subj[40]
difference_by_subj[41]
difference_by_subj[42]
difference_by_subj[43]
difference_by_subj[44]
difference_by_subj[45]
difference_by_subj[46]
difference_by_subj[47]
difference_by_subj[48]
difference_by_subj[49]
difference_by_subj[50]
difference_by_subj[51]
difference_by_subj[52]
difference_by_subj[53]
difference_by_subj[54]
difference_by_subj[55]
difference_by_subj[56]
difference_by_subj[57]
difference_by_subj[58]
difference_by_subj[59]
difference_by_subj[60]
difference_by_subj[61]
difference_by_subj[62]
difference_by_subj[63]
difference_by_subj[64]
difference_by_subj[65]
difference_by_subj[66]
difference_by_subj[67]
difference_by_subj[68]
−100
0
100
200
300

Something important to notice is that this model takes into account that participants are different, but also reduces the uncertainty of their estimates because it assumes that there are commonalities between them. This is usually called shrinkage. We can see the shrinkage of the estimates of this model when we compare them with the no pooling model (M_{np}).

```
# We'll need to make the plot 'manually'
shrinkage <- rstan::extract(fit_Muvivs_reparam)$difference_by_subj
dim(shrinkage)
```

```
## [1] 4000   68
```

```
# We apply the following function by columns to get the
# mean, and 95% credible interval of each subject's
# difference in ms
(shrinkage_mean <- apply(shrinkage, 2, mean))
```

```
##  [1] 95.03 30.68 29.69 21.47 18.11 14.12 21.77 10.16 23.66 27.32 24.95
## [12] 14.26 29.83  1.95 24.72  8.89  9.67 45.35  9.52 13.42 14.19 20.04
## [23] 45.11 17.00 18.25  5.83 55.19 71.16 26.48  8.12 34.95 24.07  8.80
## [34] 25.19 11.90 14.40 27.05 39.25 33.41 19.93 10.16 20.15 18.46 13.01
## [45] 21.73 10.18 22.37 15.41 37.43 15.45 21.47 29.06 23.52 49.75 38.43
## [56] 19.83 22.52 10.24 54.41 62.66 11.38 27.24 19.17 35.97 10.34 20.90
## [67] 45.66 52.37
```

```
(shrinkage_lower <- apply(shrinkage, 2, quantile, 0.025))
```

```
##  [1]   -8.91  -23.34  -17.48  -34.02  -42.45  -29.33  -19.53  -22.70
##  [9]  -32.18  -69.91  -31.21  -31.21  -28.10 -111.30  -26.90  -81.63
## [17]  -35.55  -36.99  -46.84  -61.36  -24.79  -19.78  -37.44  -10.93
## [25]  -31.63  -69.61   -8.58  -12.01  -13.54  -46.65  -38.49  -23.29
## [33]  -76.60  -26.57  -50.67  -31.04  -25.56   -6.85  -33.24  -28.74
## [41]  -18.79  -40.41  -15.48  -43.16  -34.72  -31.79  -74.63  -61.29
## [49]  -22.05  -35.60  -25.49  -21.75  -33.92  -18.88  -11.55  -15.64
## [57]  -41.20  -33.77  -26.95   -5.06  -57.65  -31.27  -44.31  -25.11
## [65]  -43.77  -27.06  -31.66  -62.42
```

```
(shrinkage_upper <- apply(shrinkage, 2, quantile, 0.975))
```

```
##  [1] 269.8  95.1  89.3  76.7  69.6  51.9  68.5  38.0  82.4 111.3  85.2
## [12]  52.3  95.1  70.7  81.9  68.4  45.5 142.7  49.3  66.7  51.7  63.2
```

```
## [23] 144.1  53.2  67.5  52.2 159.2 202.3  81.6  44.1 114.2  78.2  63.3
## [34]  84.8  58.8  54.6  88.4 112.2 107.3  70.8  37.2  73.0  59.8  58.2
## [45]  76.2  42.9 101.8  70.2 114.2  60.5  72.3  92.4  79.8 147.8 113.8
## [56]  61.7  82.6  45.5 166.0 177.2  61.2  86.3  73.4 111.4  50.8  71.0
## [67] 141.8 175.9
```

```
shrinkage_df <- data.frame(subj = 1:length(shrinkage_mean),
    mean = shrinkage_mean, lower = shrinkage_lower, upper = shrinkage_upper)
shrinkage_df$Model <- "Varying intercept and slope"

no_shrinkage <- rstan::extract(fit_Mnp)$difference_by_subj
dim(no_shrinkage)
```

```
## [1] 4000   68
```

```
(no_shrinkage_mean <- apply(no_shrinkage, 2, mean))
```

```
##  [1]  462.690   77.637   92.324   11.733  -23.341  -11.594   45.771
##  [8]   -6.544   11.368  -61.724   26.496   -4.690   59.137 -228.497
## [15]   48.826 -141.502  -42.249  111.465  -55.894  -68.797    4.087
## [22]   36.440  121.768   42.367    0.507 -114.810  230.020  308.426
## [29]   79.695  -62.585   53.542   43.583 -129.128   45.967  -56.486
## [36]  -10.145   65.514  162.357   49.451   16.596   -1.258   -6.951
## [43]   44.320  -45.623    9.453  -29.675  -86.617  -67.537  113.462
## [50]  -19.123   35.733   80.370   25.102  182.288  142.249   42.920
## [57]   -3.875  -37.607  193.112  284.613  -75.199   42.999  -18.389
## [64]   97.999  -52.385   26.648  129.381  105.825
```

```
(no_shrinkage_lower <- apply(no_shrinkage, 2, quantile, 0.025))
```

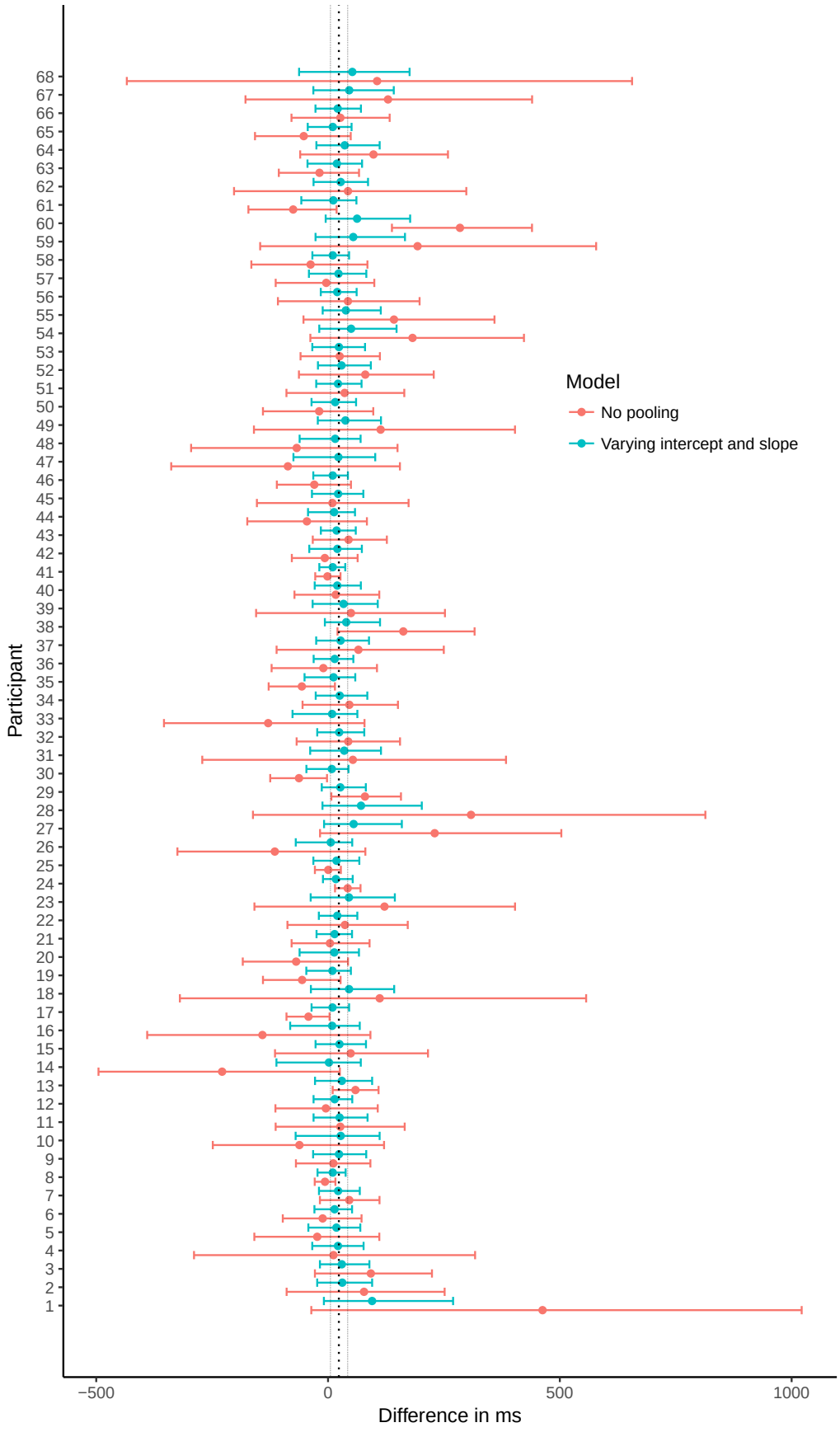
```
##  [1]  -36.10  -89.32  -28.28 -289.09 -158.95  -97.62  -17.51  -28.71
##  [9]  -69.25 -248.58 -113.00 -113.48   10.02 -495.08 -114.34 -390.09
## [17]  -89.61 -319.60 -140.33 -183.91  -78.57  -87.55 -158.75   15.09
## [25]  -28.34 -324.85  -17.26 -161.92    7.19 -124.68 -271.28  -67.73
## [33] -353.89  -54.99 -128.05 -121.71 -110.97   20.46 -155.17  -72.41
## [41]  -27.60  -78.07  -32.96 -174.18 -153.42 -110.40 -338.27 -295.34
## [49] -159.79 -140.40  -89.70  -62.77  -59.31  -38.12  -52.97 -108.12
## [57] -112.92 -165.40 -146.37  137.85 -171.73 -202.75 -106.18  -60.05
## [65] -157.54  -78.84 -178.03 -433.98
```

```
(no_shrinkage_upper <- apply(no_shrinkage, 2, quantile, 0.975))
```

```
##  [1] 1021.66  251.49  224.20  317.21  110.62   72.29  111.04   15.89
##  [9]   91.39  120.86  165.34  107.24  108.97   25.51  215.62   91.46
## [17]    3.26  556.77   27.07   43.09   89.58  171.95  403.36   70.09
## [25]   27.68   80.55  503.24  813.96  157.31   -2.20  383.92  155.28
## [33]   78.70  150.84   14.90  105.59  249.69  316.14  252.20  110.56
## [41]   26.87   63.85  126.87   83.87  173.79   49.47  155.05  149.90
## [49]  403.32   97.58  164.49  228.01  111.88  422.64  359.04  197.48
## [57]   99.81   84.99  578.35  440.00   18.49  298.28   66.87  258.70
## [65]   48.96  132.96  440.16  655.80
```

```
no_shrinkage_df <- data.frame(subj = 1:length(no_shrinkage_mean),
    mean = no_shrinkage_mean, lower = no_shrinkage_lower, upper = no_shrinkage_upper)

no_shrinkage_df$Model <- "No pooling"

models_df <- rbind(shrinkage_df, no_shrinkage_df)

overall_difference <- rstan::extract(fit_Muvivs_reparam)$overall_difference

plot_subj <- ggplot(models_df, aes(ymin = lower, ymax = upper,
    x = subj, y = mean, color = Model))
plot_subj <- plot_subj + geom_errorbar(position = position_dodge(1)) +
    geom_point(position = position_dodge(1))

# We'll also add the mean and 95% CrI of the overall
# difference to the plot:
plot_subj <- plot_subj + geom_hline(yintercept = mean(overall_difference),
    linetype = "dotted")
plot_subj <- plot_subj + geom_hline(yintercept = quantile(overall_difference,
    0.025), linetype = "dotted", size = 0.1)
plot_subj <- plot_subj + geom_hline(yintercept = quantile(overall_difference,
    0.975), linetype = "dotted", size = 0.1)

# Finally we make the plot prettier
plot_subj <- plot_subj + scale_x_continuous(name = "Participant",
```

```
    breaks = 1:length(shrinkage_mean))
plot_subj <- plot_subj + scale_y_continuous("Difference in ms") +
    theme(legend.position = c(0.8, 0.7))
plot_subj <- plot_subj + coord_flip()

plot_subj
```


Participant
Difference in ms
Model
No pooling
Varying intercept and slope
−500
0
500
1000

3.2.5 Correlated varying intercept varying slopes model (M_{cvivs})

The model M_{uvivs} allowed for different reading speeds and effects across subjects, but it has the implicit assumption that these are independent. It is in principle possible that slower participants will show stronger effects, or that slower participants will pay more attention and will be less affected, or that there is no relationship at all between reading speed and the effect of the site of attachment. We would like a model that can account for that. We model the correlation between varying intercepts and slopes, by defining a covariance relationship Σ between by-subject varying intercepts and slopes, and by assuming that both adjustments (intercept and slope) come from a multivariate normal distribution $\mathcal{N}$.

- Model M_{cvivs} assumptions:

1. RTs are log-normally distributed.
2. Some aspects of the reading speed and the effect of the site of attachment depend on the participant, and these two aspects may be related.
3. There can be a difference in RTs between the sentences with N1 attachment and N2 attachment.
4. Likelihood:

$$RT_n \sim LogNormal(\mu_n, \sigma) \tag{246}$$

with $\mu_n = \alpha + u_{1i[n]} + x_n \cdot (\beta + u_{2i[n]})$

5. Priors:

$$\begin{aligned} \alpha &\sim Normal(0, 10) \\ \beta &\sim Normal(0, 1) \\ \sigma &\sim Normal(0, 1) \text{ for } \sigma > 0 \\ \begin{pmatrix} u_{1,i} \\ u_{2,i} \end{pmatrix} &\sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma_u \right) \end{aligned} \tag{247}$$

- n represents each obs, the nth row in the data frame.
- i represents each subj, and $i[n]$ is the subject that corresponds to observation n.

In this model, we define the vector $\mathbf{u}$ as coming from a multivariate normal distribution with a variance-covariance matrix Σ_u. This matrix has the variances of the adjustment to the intercept and to the slopes respectively along the diagonal, and the covariances on the off-diagonal (lower and upper triangles). The covariance $Cov(X, Y)$ between two variables X

and Y is defined as the product of their correlation ρ and their standard deviations σ_X and σ_Y, such that, $Cov(X,Y) = \rho\sigma_X\sigma_Y$.[17]

$$\Sigma_u = \begin{pmatrix} \tau_{u_1}^2 & \rho_u\tau_{u_1}\tau_{u_2} \\ \rho_u\tau_{u_1}\tau_{u_2} & \tau_{u_2}^2 \end{pmatrix} \tag{248}$$

In addition, it has a correlation matrix associated:[18]

$$\begin{pmatrix} 1 & \rho_u \\ \rho_u & 1 \end{pmatrix} \tag{249}$$

We still need to define a prior for Σ_u. We can decompose our prior into the scales and correlation matrix in the following way:

$$\begin{aligned} \Sigma_u &= diag\_matrix(\tau_u) \cdot \rho_u \cdot diag\_matrix(\tau_u) \\ &= \begin{pmatrix} \tau_{u_1} & 0 \\ 0 & \tau_{u_2} \end{pmatrix} \begin{pmatrix} 1 & \rho_u \\ \rho_u & 1 \end{pmatrix} \begin{pmatrix} \tau_{u_1} & 0 \\ 0 & \tau_{u_2} \end{pmatrix} \end{aligned} \tag{250}$$

And now we need priors for the τ_us and for ρ_u:

$$\begin{aligned} \tau_{u_1} &\sim Normal(0,1) \text{ for } \tau_{u_1} > 0 \\ \tau_{u_2} &\sim Normal(0,1) \text{ for } \tau_{u_2} > 0 \\ \rho_u &\sim LKJcorr(2) \end{aligned} \tag{251}$$

The basic idea of the LKJ correlation distribution is that as its parameter (usually called eta, η, here is 2) increases, the prior increasingly concentrates around the unit correlation matrix (i.e., favors less correlation: ones in the diagonals and values close to zero in the lower and upper triangles). At $\eta = 1$, the LKJ correlation distribution is uninformative (similar to $Beta(1,1)$), at $\eta < 1$, it favors extreme correlations (similar to $Beta(a < 1, b < 1)$).

We would end up with a model that looks like this (I'm omitting the data declaration, which is as before, and the re-parametrization).

```
parameters {
  real<lower = 0> sigma;
  vector<lower = 0>[2] tau_u;
```

[17] Notice that I'm calling the SD of the u's τ and not σ, don't get too attached to the specific Greek letter!

[18] Since we have two random variables, there is only one correlation in the matrix between u_1 and u_2. With 3 random variables (i.e., adding u_3), we end up with a 3×3 correlation matrix that includes three different correlations $\rho_{u_{1,2}} = \rho_{u_{2,1}}$, $\rho_{u_{1,3}} = \rho_{u_{3,1}}$, and $\rho_{u_{2,3}} = \rho_{u_{3,2}}$.

```
  real alpha;
  real beta;
  corr_matrix[2] rho_u;
  matrix[2, N_subj] u_t;
  }
transformed parameters {
  matrix[N_subj, 2]  u_t; // matrix of 2 rows and N_subj columns
  vector[N] mu;
  mu = alpha + u_t[subj, 1] +  x .* (beta + u_t[subj, 2]);
}
model {
  vector[N_obs] mu;
  rho_u ~ lkj_corr(2);
  alpha ~ normal(0,10);
  beta  ~ normal(0,1);
  sigma ~ normal(0,1);
  tau_u ~ normal(0,1);
  for(i in 1:N_subj)
    u_t[,i] ~ multi_normal(rep_vector(0, 2), quad_form_diag(rho_u, tau_u));
  RT ~ lognormal(mu, sigma);
}
```

There are a couple of new things in the previous code.

- `u_t` is a transposed version of the u that I presented before. The transposition allows us to do the element-wise multiplication `x .* u_t[subj,2]` since the two vectors(`x` and `u_t[subj,2]`) have the same number of rows.
- `matrix[n,m] M;` defines a matrix of n rows and m columns called M.
- `corr_matrix[n] M;` defines a (square) matrix of n rows and n columns called M, symmetrical around a diagonal of ones.
- `rep_vector(X, n)` creates a vector with n columns filled with X.
- `quad_form_diag(M, V)` a quadratic form using the column vector V as a diagonal matrix, i.e., in Stan notation: `diag_matrix(V) * M * diag_matrix(V)`. Notice that `*` in Stan is matrix multiplication, and `.*` is element-wise multiplication.[19]
- `M[,i]` creates a vector of the column i of matrix M.

This model is for illustration purposes, there are more efficient ways to code this. And the

[19] In R, `%*%` is matrix multiplication and `*` is element-wise multiplication.

efficiency considerations start to be important in hierarchical models. In practice, we will use a Cholesky factorization that we detail below (see also Sorensen 2016).

Given a correlation matrix ρ_u,[20] we can get a lower triangular matrix $\mathbf{L_u}$ such that $\mathbf{L_u}\mathbf{L_u}^T = \rho_u$. The matrix $\mathbf{L_u}$ is called the Cholesky factor of $\rho_\mathbf{u}$.[21]

$$\mathbf{L_u} = \begin{pmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{pmatrix} \tag{252}$$

The following R code is only for illustration, we'll end up coding everything directly in Stan. As an example, let's assume a correlation of 0.8.

```
rho <- 0.8
# Correlation matrix
(rho_u <- matrix(c(1, rho, rho, 1), ncol = 2))
```

```
##      [,1] [,2]
## [1,]  1.0  0.8
## [2,]  0.8  1.0
```

```
# Cholesky factor: (we transpose it so that it looks the
# same as in Stan)
(L_u <- t(chol(rho_u)))
```

```
##      [,1] [,2]
## [1,]  1.0  0.0
## [2,]  0.8  0.6
```

```
# We verify that we recover rho_u, %*% indicates matrix
# multiplication (=! element-wise multiplication)
L_u %*% t(L_u)
```

```
##      [,1] [,2]
## [1,]  1.0  0.8
## [2,]  0.8  1.0
```

For the previous model M_{uvivs}, it was enough to produce the adjustment of the intercept and of the slope from normal distributions, but now we want the adjustments to be correlated.

[20] a symmetric positive definite or semi-definite matrix

[21] You can think of it as the square root of the matrix ρ_u.

We can use the Cholesky factor to generate correlated random variables in the following way.

1. We generate uncorrelated values that will be associated with each adjustment u from a $Normal(0,1)$. In our case we have u_1 and u_2, so we will generate:

$$z_{u_1} \sim Normal(0,1)$$
$$z_{u_2} \sim Normal(0,1)$$

For example, assuming only 10 subjects.

```
N_subj <- 10
(z_u1 <- rnorm(N_subj, 0, 1))
```

```
##  [1]  1.1496 -0.6489 -0.0864  1.0593 -0.3386  0.0655  0.4269 -0.8201
##  [9] -1.9114  1.5859
```

```
(z_u2 <- rnorm(N_subj, 0, 1))
```

```
##  [1] -0.810  0.562 -0.974  0.916 -0.688 -0.193  0.779  1.538  0.776 -1.665
```

---

2. By multiplying the Cholesky factor by our z's we generate a matrix of correlated variables (with standard deviation of 1).

$$\mathbf{L_u} \cdot \mathbf{z_u} = \begin{pmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{pmatrix} \begin{pmatrix} z_{u_{1,subj=1}} & z_{u_{1,subj=2}} & \dots & z_{u_{1,subj=N_{subj}}} \\ z_{u_{2,subj=1}} & z_{u_{2,subj=2}} & \dots & z_{u_{2,subj=N_{subj}}} \end{pmatrix}$$

$$\mathbf{L_u} \cdot \mathbf{z_u} = \begin{pmatrix} l_{11} \cdot z_{u_{1,1}} + 0 \cdot z_{u_{2,1}} & l_{11} \cdot z_{u_{1,2}} & \dots & l_{11} \cdot z_{u_{1,N_{subj}}} \\ l_{21} \cdot z_{u_{1,1}} + l_{22} \cdot z_{u_{2,1}} & l_{21} \cdot z_{u_{1,2}} + l_{22} \cdot z_{u_{2,2}} & \dots & l_{11} \cdot z_{u_{1,N_{subj}}} + l_{22} \cdot z_{u_{2,N_{subj}}} \end{pmatrix}$$

A very informal explanation of why this works is that we are making the variable that corresponds to the slope to be a function of a scaled version of the intercept.

For example:

```
# matrix z_u
(z_u <- matrix(c(z_u1, z_u2), ncol = N_subj, byrow = T))
```

```
##        [,1]   [,2]    [,3]  [,4]   [,5]    [,6]  [,7]  [,8]   [,9] [,10]
## [1,]  1.15 -0.649 -0.0864 1.059 -0.339  0.0655 0.427 -0.82 -1.911  1.59
## [2,] -0.81  0.562 -0.9737 0.916 -0.688 -0.1930 0.779  1.54  0.776 -1.67
```

```
L_u %*% z_u
```

```
##        [,1]   [,2]    [,3] [,4]   [,5]    [,6]  [,7]   [,8]  [,9] [,10]
## [1,] 1.150 -0.649 -0.0864 1.06 -0.339  0.0655 0.427 -0.820 -1.91  1.59
## [2,] 0.434 -0.182 -0.6533 1.40 -0.684 -0.0634 0.809  0.267 -1.06  0.27
```

---

3. The last step is to scale the previous matrix to the desired standard deviation. We define the diagonalized matrix $diag\_matrix(\tau_u)$ as before:

$$\begin{pmatrix} \tau_{u_1} & 0 \\ 0 & \tau_{u_2} \end{pmatrix}$$

For example:

```
tau_u1 <- 0.2
tau_u2 <- 0.01
(diag_matrix_tau <- matrix(c(tau_u1, 0, 0, tau_u2), ncol = 2))
```

```
##      [,1] [,2]
## [1,]  0.2 0.00
## [2,]  0.0 0.01
```

And we pre-multiply it by the correlated variables with SD of 1 from before:

$$\mathbf{u} = diag\_matrix(\tau_u) \cdot \mathbf{L_u} \cdot \mathbf{z_u}$$

$$\mathbf{u} = \begin{pmatrix} \tau_{u_1} & 0 \\ 0 & \tau_{u_2} \end{pmatrix} \begin{pmatrix} l_{11} \cdot z_{u_{1,1}} & l_{11} \cdot z_{u_{1,2}} & \dots & l_{11} \cdot z_{u_{1,N_{subj}}} \\ l_{21} \cdot z_{u_{1,1}} + l_{22} \cdot z_{u_{2,1}} & l_{21} \cdot z_{u_{1,2}} + l_{22} \cdot z_{u_{2,2}} & \dots & l_{11} \cdot z_{u_{1,N_{subj}}} + l_{22} \cdot z_{u_{2,N_{subj}}} \end{pmatrix}$$

$$\mathbf{u} = \begin{pmatrix} \tau_{u_1} \cdot l_{11} \cdot z_{u_{1,1}} & \tau_{u_1} \cdot l_{11} \cdot z_{u_{1,2}} & \dots & \tau_{u_1} \cdot l_{11} \cdot z_{u_{1,N_{subj}}} \\ \tau_{u_2} \cdot (l_{21} \cdot z_{u_{1,1}} + l_{22} \cdot z_{u_{2,1}}) & \tau_{u_2} \cdot (l_{21} \cdot z_{u_{1,2}} + l_{22} \cdot z_{u_{2,2}}) & \dots & \tau_{u_2} \cdot (l_{11} \cdot z_{u_{1,N_{subj}}} + l_{22} \cdot z_{u_{2,N_{subj}}}) \end{pmatrix}$$

For example:

```
(u <- diag_matrix_tau %*% L_u %*% z_u)
```

```
##         [,1]     [,2]     [,3]  [,4]     [,5]      [,6]    [,7]     [,8]
## [1,] 0.22991 -0.12978 -0.01728 0.212 -0.06773  0.013104 0.08539 -0.16402
## [2,] 0.00434 -0.00182 -0.00653 0.014 -0.00684 -0.000634 0.00809  0.00267
##         [,9]  [,10]
## [1,] -0.3823 0.3172
## [2,] -0.0106 0.0027
```

```
# We should find that the rows are correlated ~.8
cor(u[1, ], u[2, ])
```

```
## [1] 0.701
```

```
# We should be able to recover the tau's as well:
sd(u[1, ])
```

```
## [1] 0.211
```

```
sd(u[2, ])
```

```
## [1] 0.00743
```

Now we can code it in Stan and save it as `Mcvivs.stan`.

```
data {
  int<lower = 1> N;
  vector[N] RT;
  vector<lower = -1, upper = 1>[N] x;
  int<lower = 1> N_subj;
  int<lower = 1, upper = N_subj> subj[N];
}
parameters {
  real<lower = 0> sigma;
  real alpha;
  real beta;
  vector<lower = 0> [2]  tau_u;
  cholesky_factor_corr[2] L_u;
```

```
  matrix[2, N_subj] z_u;
}
transformed parameters {
  matrix[2,N_subj]  u; // matrix of N_subj rows and 2 columns
  matrix[N_subj, 2]  u_t; // matrix of 2 rows and N_subj columns
  vector[N] mu;
  u = diag_pre_multiply(tau_u, L_u) * z_u;
  u_t = u'; // I transpose u so that I can use it later in the multiplication:
  mu = alpha + u_t[subj, 1] +  x .* (beta + u_t[subj, 2]);
}
model {
  alpha ~ normal(0, 10);
  beta  ~ normal(0, 1);
  sigma ~ normal(0, 1);
  tau_u ~ normal(0, 1);
  // Notice that the prior is on z_u and not on u or u_t
  to_vector(z_u) ~ normal(0, 1);
  L_u ~ lkj_corr_cholesky(2);
  RT ~ lognormal(mu, sigma);
}
generated quantities {
  real overall_difference;
  vector[N_subj] difference_by_subj;
  matrix[2,2] rho_u;
  rho_u = L_u * L_u';
  for(i in 1:N_subj){
   difference_by_subj[i] = exp(alpha + u_t[i,1] + (beta + u_t[i,2]) ) - exp(alpha + u_t[i,1]
  }
  overall_difference = exp(alpha + beta) - exp(alpha - beta);
}
```

The new things here are:

- `cholesky_factor_corr[2] L_u`, which defines `L_u` as a lower triangular (of 2×2) matrix which has to be the Cholesky factor of a correlation.
- `diag_pre_multiply(tau_u,L_u)` which simply makes a diagonal matrix out of the vector `tau_u` and multiplies it by `L_u`.

- `to_vector(z_u)` makes a long vector out the matrix `z_u`
- `L_u ~ lkj_corr_cholesky(2);` is the Cholesky factor associated with the lkj correlation distribution, such that it implies that `L_u * L_u' ~ lkj_corr(2.0);`. Notice that `'` indicates transposition (while it is `t(.)` in R).

We can recover the correlation by adding in the `generated quantities` section a 2×2 matrix `rho_u`, defined as `rho_u = L_u * L_u';`.

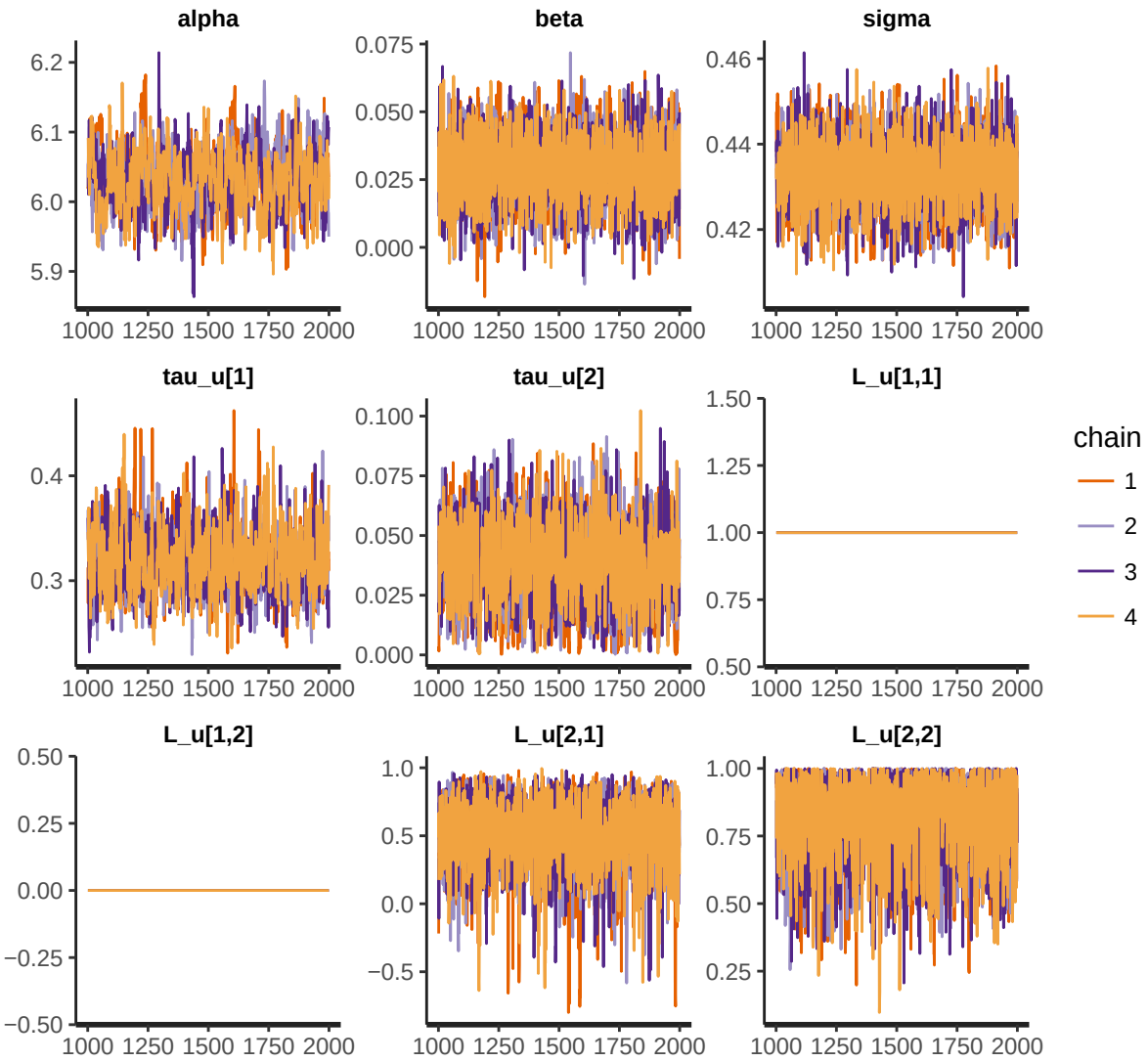
```
fit_Mcvivs <- stan(file = "Mcvivs.stan", data = Swets_list)
```

We do MCMC-diagnostics as always.

```
print(fit_Mcvivs, probs = c(0.025, 0.5, 0.975), pars = c("alpha",
    "beta", "sigma", "tau_u", "L_u"))
```

```
## Inference for Stan model: Mcvivs.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd  2.5%  50% 97.5% n_eff Rhat
## alpha    6.03    0.00 0.04  5.95 6.04  6.12   410 1.01
## beta     0.03    0.00 0.01  0.00 0.03  0.05  4000 1.00
## sigma    0.43    0.00 0.01  0.42 0.43  0.45  4000 1.00
## tau_u[1] 0.32    0.00 0.03  0.27 0.32  0.38   702 1.00
## tau_u[2] 0.04    0.00 0.02  0.00 0.04  0.07  1366 1.00
## L_u[1,1] 1.00    0.00 0.00  1.00 1.00  1.00  4000  NaN
## L_u[1,2] 0.00    0.00 0.00  0.00 0.00  0.00  4000  NaN
## L_u[2,1] 0.50    0.01 0.26 -0.09 0.54  0.89  2119 1.00
## L_u[2,2] 0.81    0.00 0.15  0.45 0.84  1.00  2686 1.00
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:49:51 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
traceplot(fit_Mcvivs, pars = c("alpha", "beta", "sigma", "tau_u",
    "L_u"))
```



Why are `L_u[1,1]` and `L_u[1,2]` just straight lines?

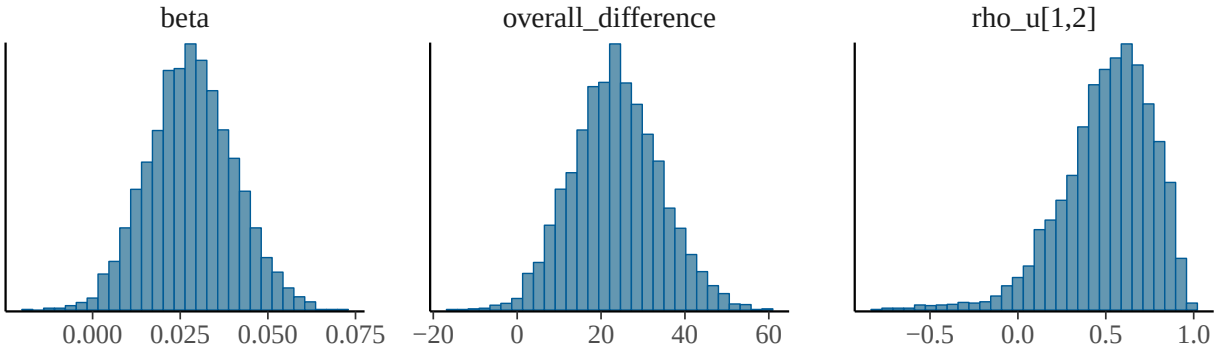
Only now we can examine the posterior distributions to do inference.

```
print(fit_Mcvivs, probs = c(0.025, 0.5, 0.975), pars = c("overall_difference",
    "rho_u"), digits = 3)
```

```
## Inference for Stan model: Mcvivs.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                      mean se_mean     sd   2.5%    50%  97.5% n_eff  Rhat
## overall_difference 23.573   0.165 10.434  3.740 23.364 44.854  4000 1.001
## rho_u[1,1]          1.000   0.000  0.000  1.000  1.000  1.000  4000   NaN
## rho_u[1,2]          0.504   0.006  0.262 -0.093  0.537  0.893  2119 1.001
```

```
## rho_u[2,1]          0.504   0.006  0.262 -0.093  0.537  0.893  2119 1.001
## rho_u[2,2]          1.000   0.000  0.000  1.000  1.000  1.000  4000 0.999
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:49:51 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
posterior_Mcvivs <- as.array(fit_Mcvivs)
mcmc_hist(posterior_Mcvivs, pars = c("beta", "overall_difference",
    "rho_u[1,2]"))
```



We can calculate, as before, how certain we are that the effect across subjects is larger than 0.

```
mean(rstan::extract(fit_Mcvivs)$beta > 0)
```

```
## [1] 0.992
```

But more interestingly, there is some weak evidence that slower participants are more likely to show a stronger effect, $\hat{\rho}_u = 0.5, 95\%$ credible interval $= [-0.09, 0.89]$. How sure should we be that there is a positive correlation?

```
# We need to extract the samples in the following way,
# because rho is actually a matrix. When we use the function
# extract, the first dimension consists of the samples.  See
# dim(rstan::extract(fit_Mcvivs)$rho_u)
mean(rstan::extract(fit_Mcvivs)$rho_u[, 1, 2] > 0)
```

```
## [1] 0.959
```

3.2.6 Recovery of the parameters

We are working with quite a complex model, we need to be sure that the model is doing what we build it for. We'll generate fake data, which is similar to our data, but where we know the true values of the parameters, and we'll examine how well the model recovers them.

```
library(MASS)

# We decide how the data should look like.  We'll start with
# fake data similar to our data
N_subj <- 70
N_obs <- 24 * N_subj
x <- rep(c(1, -1), N_obs/2)
# We create a vector similar to the one in Stan, that looks
# like 1,1,1,...,2,2,2,2...  Be careful to pay attention if
# you're repeating using each= or times=. If you forget to
# add each then you will generate 1,2,3,.., N_subj, 1,2,3...
subj <- rep(1:N_subj, each = N_obs/N_subj)

# We could also completely imitate the format of our data.

# N_subj <- length(unique(N1N2$sub))

# N_obs <- length(N1N2$RT)

# x <- N1N2$x

# subj <- as.numeric(as.factor(N1N2$sub))

# These will be the TRUE values, not necessarily the mean of
# the posterior
alpha <- 6
beta <- 0.05
sigma <- 0.4
tau_u <- c(0.3, 0.05)
rho <- 0.5
```

```
# This matrix is symmetric, so it won't matter, but be
# careful with how R arranges the matrix values
Cor_u <- matrix(c(1, rho, rho, 1), nrow = 2)

# Variance covariance matrix for 'subj':
Sigma_u <- diag(tau_u, 2, 2) %*% Cor_u %*% diag(tau_u, 2, 2)

# Let's create the correlated adjustments (here I'm creating
# a matrix of u, instead of joining two vectors):
u <- mvrnorm(n = N_subj, c(0, 0), Sigma_u)

# Are they correlated as expected?
cor(u)
```

```
##       [,1]  [,2]
## [1,] 1.000 0.576
## [2,] 0.576 1.000
```

```
# Are the SDs as expected?
sd(u[, 1])
```

```
## [1] 0.333
```

```
sd(u[, 2])
```

```
## [1] 0.0474
```

```
# We create a vector of the location for the log-normal
# distribution:
mu <- alpha + u[subj, 1] + x * (beta + u[subj, 2])
RT <- rlnorm(N_obs, mu, sigma)

fake_data_list <- list(RT = RT, N = N_obs, subj = subj, N_subj = N_subj,
    x = x)
```

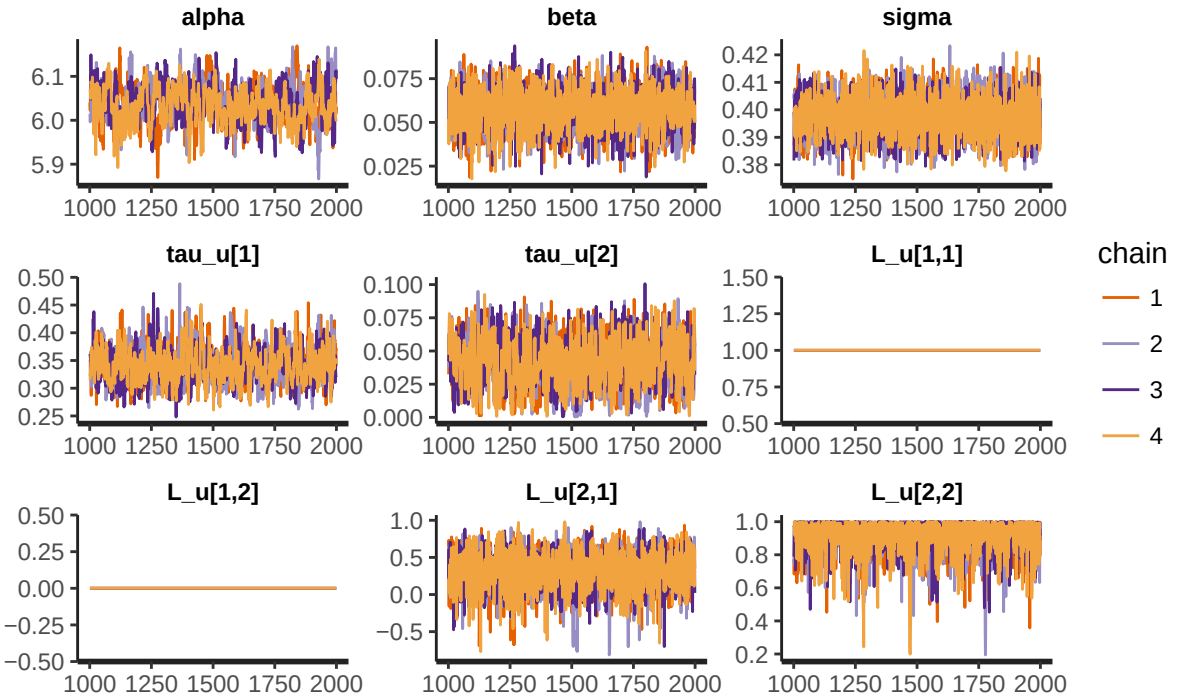
```
fit_fake_Mcvivs <- stan(file = "Mcvivs.stan", data = fake_data_list)
```

We do MCMC-diagnostics:

```
print(fit_fake_Mcvivs, probs = c(0.025, 0.5, 0.975), pars = c("alpha",
    "beta", "sigma", "tau_u", "L_u"))
```

```
## Inference for Stan model: Mcvivs.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd  2.5%  50% 97.5% n_eff Rhat
## alpha    6.04       0 0.04  5.95 6.04  6.12   281 1.01
## beta     0.06       0 0.01  0.03 0.06  0.08  4000 1.00
## sigma    0.40       0 0.01  0.38 0.40  0.41  4000 1.00
## tau_u[1] 0.34       0 0.03  0.29 0.34  0.41   601 1.00
## tau_u[2] 0.04       0 0.02  0.01 0.04  0.07   737 1.00
## L_u[1,1] 1.00       0 0.00  1.00 1.00  1.00  4000  NaN
## L_u[1,2] 0.00       0 0.00  0.00 0.00  0.00  4000  NaN
## L_u[2,1] 0.32       0 0.25 -0.21 0.33  0.76  2794 1.00
## L_u[2,2] 0.91       0 0.10  0.65 0.94  1.00  1801 1.00
##
## Samples were drawn using NUTS(diag_e) at Thu Sep 14 21:50:22 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

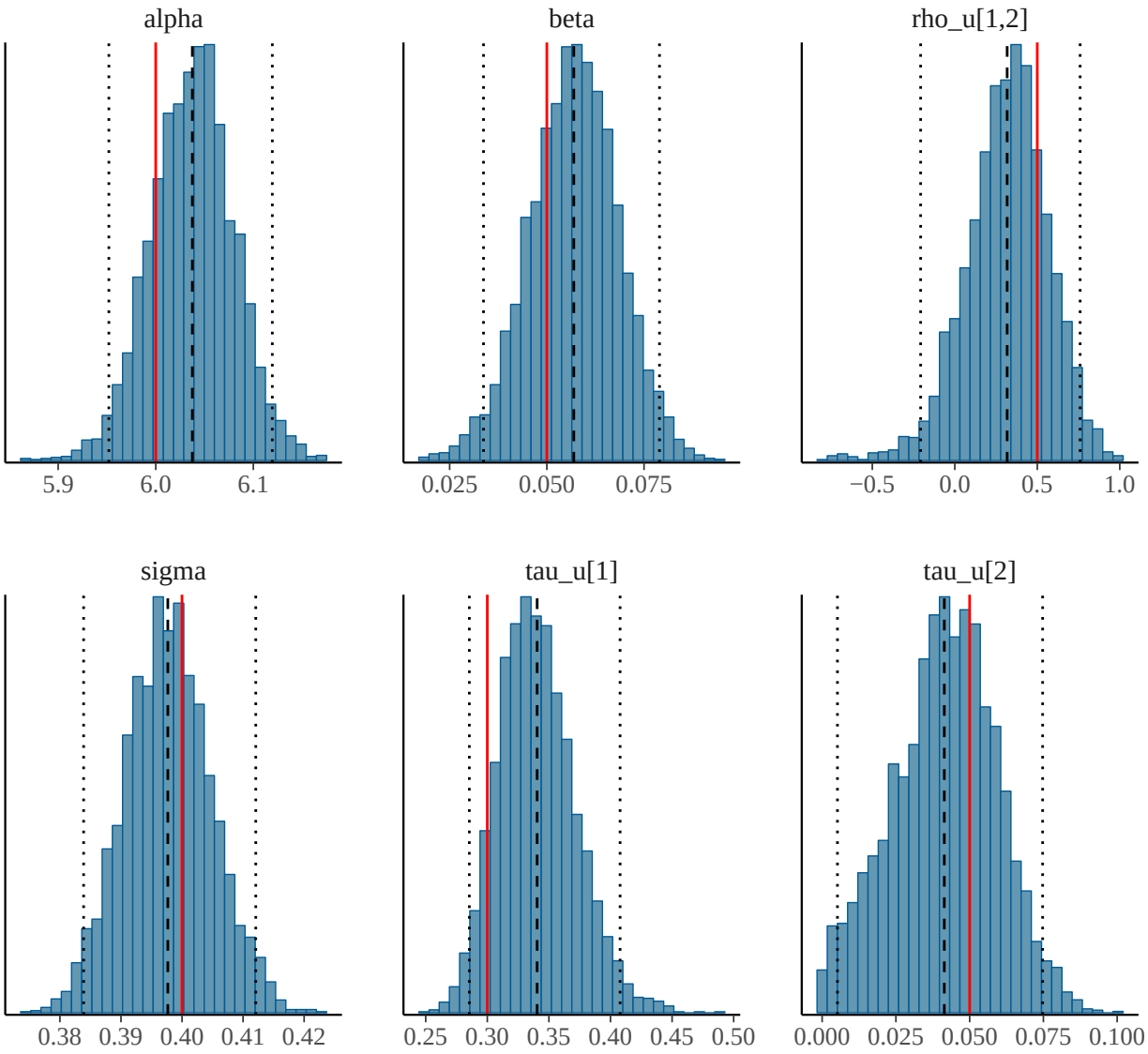
```
traceplot(fit_fake_Mcvivs, pars = c("alpha", "beta", "sigma",
    "tau_u", "L_u"))
```



Let's see if the true values are generally inside the 95% credible intervals.

```
# I save the true values with the same names as the Stan
# model use
true_values <- data.frame(Parameter = c("alpha", "beta", "sigma",
    "tau_u[1]", "tau_u[2]", "rho_u[1,2]"), value = c(alpha,
    beta, sigma, tau_u, rho))
posterior_fakeMcvivs <- as.array(fit_fake_Mcvivs)

# Data frame with the summary
fakeMcvivs_summary <- data.frame(summary(fit_fake_Mcvivs)$summary)
fakeMcvivs_summary$Parameter <- rownames(fakeMcvivs_summary)
# For some reason, R transforms the 2.5% and 97.5% into
# X2.5. and X97.5. when we transform the summary to a data
# frame
fakeMcvivs_summary <- fakeMcvivs_summary[c("Parameter", "mean",
    "X2.5.", "X97.5.")]
fakeMcvivs_summary <- subset(fakeMcvivs_summary, Parameter %in%
    c("alpha", "beta", "sigma", "tau_u[1]", "tau_u[2]", "rho_u[1,2]"))

# We plot the historgram
p_hist <- mcmc_hist(posterior_fakeMcvivs, pars = c("alpha",
    "beta", "sigma", "tau_u[1]", "tau_u[2]", "rho_u[1,2]"))

# We add the summaries (notice that columns with strange
# characters need to go inside ``)
p_hist <- p_hist + geom_vline(data = fakeMcvivs_summary, aes(xintercept = mean),
    linetype = "dashed")
p_hist <- p_hist + geom_vline(data = fakeMcvivs_summary, aes(xintercept = X2.5.),
    linetype = "dotted")
p_hist <- p_hist + geom_vline(data = fakeMcvivs_summary, aes(xintercept = X97.5.),
    linetype = "dotted")

# We add the true values in red
p_hist <- p_hist + geom_vline(data = true_values, aes(xintercept = value),
    color = "red")
```

```
p_hist
```



We see that the true value is in general inside the 95% credible interval (and in fact it should be there for 95% of the parameters), but there's a lot of uncertainty in the parameter we care about, `beta`. Take into account that this is the best case scenario, we are assuming that the data are generated in the same manner as the likelihood. This will never happen in real life, but it can give us an idea whether the model is doing what it should be doing, and what's our upper limit on certainty in the inference: The model may work fine when the data is generated similarly to the likelihood that we assume, but may not work well if the data is generated in a very different way.

3.3 Why should we take the trouble of fitting a Bayesian hierarchical model?

During these last two sessions we have discussed a way to add a hierarchical structure to the location of the normal and the log-normal distributions (by having these u's that go inside the parameter μ of the distributions). Carrying out Bayesian data analysis clearly requires much more effort than fitting a frequentist model: we have to define priors, verify that our model works, and decide how to interpret the results. By comparison, fitting a linear mixed model using `lme4` consists of only one single line of code; there are, of course, many assumptions made, but they are hidden from us.

We want to emphasize the following points regarding the reason for fitting Bayesian hierarchical models (we discuss more thoroughly the advantages of Bayesian modeling in general in Nicenboim and Vasishth 2016).

- For linear mixed models, one strength of Bayesian models is that we can fit models with a full random structure that would not converge with frequentist methods or would yield overestimates of correlations between the random effects (Bates et al. 2015). Some examples are Hofmeister and Vasishth (2014), Husain, Vasishth, and Srinivasan (2014), and Frank, Trompenaars, and Vasishth (2015).
- The same approach we used here can be used to extend any generalized linear model (such as logistic regression when our dependent variable is binary such as response accuracy) or non-linear model. This includes useful models that are rarely used in psycholinguistics such as multi-logistic regression (e.g., accuracy in some task with more than two answers), ordered logistic (e.g., ratings), and models with a shifted log-normal distribution (see Nicenboim et al. 2016; Rouder 2005).
- Complex cognitive models can be extended hierarchically in a straightforward way, see M. D. Lee (2011) and M. D. Lee and Wagenmakers (2014).[22] Some examples of hierarchical computational cognitive models in psycholinguistics are Logačev and Vasishth (2016), Nicenboim and Vasishth (Submitted), Vasishth et al. (2017), and Vasishth, Jäger, and Nicenboim (2017).

[22]Notice the distinction between using Bayesian methods for modeling cognitive processes and assuming that (some aspect of) the mind is Bayesian.

3.4 Key concepts

- Advantages (and disadvantages) of hierarchical models.
- Complete pooling vs. no pooling.
- Varying intercepts and slopes and their correlation.
- Cholesky factorization.
- Simulation of data from a hierarchical model.

3.5 Exercises

1. By-items random effects. Everything that we said about subjects is also relevant for items, it could be that some items are harder than others, biasing the preference more towards N1 or N2 attachment.
   (a) Assume that the observations depend completely on the specific item (M'_{np}). (No pooling model). What is the posterior for the average beta (Mean and 95% credible interval)? For how many items are you 95% sure that there is an effect?
   (b) Now build a model with by-participant and by-item correlated varying intercept and varying slopes (M'_{cvivs}). Is there more variance in the by-participants or by-items adjustments of this model? How is the uncertainty of the `overall_difference` in comparison with original M_{cvivs}?
   (c) Compare the `effect_by_item` of M'_{cvivs} with M'_{np}, plotting the items of both models together. Is there shrinkage?
   (d) Optional. The original experiment had 3 conditions, N1 attachment, N2 attachment, and ambiguous attachment. It was hypothesized that N2 would be faster than N1 and than the ambiguous condition would be faster than N2. Try to fit a by-participant and by-item correlated varying intercept and varying slopes model (a.k.a full random effects model). Tip: You will need two vectors of predictors now, two parameters for the effect, and thus two "effect" adjustments for items and two for subjects.
2. Fake data simulation.
   (a) Run the fake data simulation a couple of times as it is and fit it with the model M_{cvivs}. Verify that 50% of the time the true values are inside the 50% credible intervals.
   (b) Run the fake data simulation with 20 subjects. What happens?
   (c) Change the true value of β to 0.005 and run the fake data simulation. What is the true value of the difference between conditions in milliseconds? What is the posterior distribution of β now according to the model?

References

Bates, Douglas, Reinhold Kliegl, Shravan Vasishth, and Harald Baayen. 2015. "Parsimonious Mixed Models." http://arxiv.org/abs/1506.04967.

Carreiras, Manuel, and Charles Clifton. 1999. "Another Word on Parsing Relative Clauses: Eyetracking Evidence from Spanish and English." Memory & Cognition 27 (5): 826–33.

Frank, Stefan L., Thijs Trompenaars, and Shravan Vasishth. 2015. "Cross-Linguistic Differences in Processing Double-Embedded Relative Clauses: Working-Memory Constraints or Language Statistics?" Cognitive Science, n/a.

Frazier, Lyn, and Keith Rayner. 1982. "Making and Correcting Errors During Sentence Comprehension: Eye Movements in the Analysis of Structurally Ambiguous Sentences." Cognitive Psychology 14 (2). Elsevier: 178–210.

Gelman, Andrew, and Jennifer Hill. 2007. Data Analysis Using Regression and Multilevel/Hierarchical Models. Cambridge University Press.

Gelman, Andrew, John B Carlin, Hal S Stern, and Donald B Rubin. 2014. Bayesian Data Analysis. Third Edition. Taylor & Francis.

Gelman, Andrew, Jessica Hwang, and Aki Vehtari. 2014. "Understanding Predictive Information Criteria for Bayesian Models." Statistics and Computing 24 (6). Springer: 997–1016. doi:10.1007/s11222-013-9416-2.

Hoekstra, Rink, Richard D Morey, Jeffrey N. Rouder, and Eric-Jan Wagenmakers. 2014. "Robust Misinterpretation of Confidence Intervals." Psychonomic Bulletin & Review 21 (5). Springer: 1157–64. doi:10.3758/s13423-013-0572-3.

Hofmeister, Philip, and Shravan Vasishth. 2014. "Distinctiveness and Encoding Effects in Online Sentence Comprehension." Frontiers in Psychology 5: 1–13. doi:doi: 10.3389/fpsyg.2014.01237.

Husain, Samar, Shravan Vasishth, and Narayanan Srinivasan. 2014. "Strong Expectations Cancel Locality Effects: Evidence from Hindi." PLoS ONE 9 (7). Public Library of Science: 1–14.

Jäger, Lena Ann, Felix Engelmann, and Shravan Vasishth. 2017. "Similarity-Based Interference in Sentence Comprehension: Literature Review and Bayesian Meta-Analysis." Journal of Memory and Language 94: 316–39.

Lee, Michael D. 2011. "How Cognitive Modeling Can Benefit from Hierarchical Bayesian Models." Journal of Mathematical Psychology 55 (1). Elsevier BV: 1–7.

doi:10.1016/j.jmp.2010.08.013.

Lee, Michael D., and Eric-Jan Wagenmakers. 2014. Bayesian Cognitive Modeling: A Practical Course. Cambridge University Press.

Levy, Roger P. 2008. "Expectation-Based Syntactic Comprehension." Cognition 106 (3): 1126–77. doi:10.1016/j.cognition.2007.05.006.

Logačev, Pavel, and Shravan Vasishth. 2016. "A Multiple-Channel Model of Task-Dependent Ambiguity Resolution in Sentence Comprehension." Cognitive Science 40 (2): 266–98. doi:10.1111/cogs.12228.

Lunn, D.J., A. Thomas, N. Best, and D. Spiegelhalter. 2000. "WinBUGS-A Bayesian Modelling Framework: Concepts, Structure, and Extensibility." Statistics and Computing 10 (4). Springer: 325–37.

McElreath, Richard. 2015. Statistical Rethinking: A Bayesian Course with R Examples. Chapman; Hall/CRC.

Morey, Richard D, Rink Hoekstra, Jeffrey N. Rouder, Michael D Lee, and Eric-Jan Wagenmakers. 2016. "The Fallacy of Placing Confidence in Confidence Intervals" 23 (1): 103–23. doi:10.3758/s13423-015-0947-8.

Nicenboim, Bruno, and Shravan Vasishth. 2016. "Statistical methods for linguistic research: Foundational Ideas - Part II." Language and Linguistics Compass 10 (11): 591–613. doi:10.1111/lnc3.12207.

———. Submitted. "Models of Retrieval in Sentence Comprehension: A Computational Evaluation Using Bayesian Hierarchical Modeling." https://arxiv.org/abs/1612.04174.

Nicenboim, Bruno, Felix Engelmann, Katja Suckow, and Shravan Vasishth. Submitted. "Number Interference in German: Evidence for Cue-Based Retrieval." Open Science Framework. doi:10.17605/OSF.IO/MMR7S.

Nicenboim, Bruno, Pavel Logačev, Carolina Gattei, and Shravan Vasishth. 2016. "When High-Capacity Readers Slow down and Low-Capacity Readers Speed up: Working Memory and Locality Effects." Frontiers in Psychology 7 (280). doi:10.3389/fpsyg.2016.00280.

Plummer, Martin. 2016. "JAGS Version 4.2.0 User Manual."

Rouder, Jeffrey N. 2005. "Are Unshifted Distributional Models Appropriate for Response Time?" Psychometrika 70 (2). Springer Science + Business Media: 377–81. doi:10.1007/s11336-005-1297-7.

Shiffrin, Richard M., Michael Lee, Woojae Kim, and Eric-Jan Wagenmakers. 2008. "A

Survey of Model Evaluation Approaches with a Tutorial on Hierarchical Bayesian Methods." Cognitive Science 32 (8). Wiley-Blackwell: 1248–84. doi:10.1080/03640210802414826.

Sorensen, Sven AND Vasishth, Tanner AND Hohenstein. 2016. "Bayesian Linear Mixed Models Using Stan: A Tutorial for Psychologists, Linguists, and Cognitive Scientists." The Quantitative Methods for Psychology 12 (3). TQMP: 175–200. doi:10.20982/tqmp.12.3.p175.

Stan Development Team. 2017. "Stan: A C++ Library for Probability and Sampling, Version 2.16.0." http://mc-stan.org/.

Swets, Benjamin, Timothy Desmet, Charles Clifton, and Fernanda Ferreira. 2008. "Underspecification of Syntactic Ambiguities: Evidence from Self-Paced Reading." Memory & Cognition 36 (1): 201–16. doi:10.3758/MC.36.1.201.

Vasishth, S., L. A. Jäger, and B. Nicenboim. 2017. "Feature overwriting as a finite mixture process: Evidence from comprehension data." In Proceedings of Mathpsych/Iccm Conference. Warwick, UK. https://arxiv.org/abs/1703.04081.

Vasishth, Shravan, Zhong Chen, Qiang Li, and Gueilan Guo. 2013. "Processing Chinese Relative Clauses: Evidence for the Subject-Relative Advantage." PLoS ONE 8 (10). Public Library of Science: 1–14.

Vasishth, Shravan, Nicolas Chopin, Robin Ryder, and Bruno Nicenboim. 2017. "Modelling Dependency Completion in Sentence Comprehension as a Bayesian Hierarchical Mixture Process: A Case Study Involving Chinese Relative Clauses." In Proceedings of Cognitive Science Conference. London, UK. https://arxiv.org/abs/1702.00564v2.

Vehtari, Aki, and Janne Ojanen. 2012. "A Survey of Bayesian Predictive Methods for Model Assessment, Selection and Comparison." Statistics Surveys 6 (0). Institute of Mathematical Statistics: 142–228. doi:10.1214/12-ss102.

Vehtari, Aki, Andrew Gelman, and Jonah Gabry. 2017. "Practical Bayesian Model Evaluation Using Leave-One-Out Cross-Validation and WAIC." Statistics and Computing 27 (5): 1413–32. doi:10.1007/s11222-016-9696-4.