COMPILED BY SHRAVAN VASISHTH

ADVANCED DATA ANALYSIS

FOR PSYCHOLINGUISTICS: BAYESIAN METHODS

VASISHTH LAB LECTURE NOTES

Contents

List of Figures

List of Tables

Preface

Version dated May 3, 2014.

These are lecture notes for a master's level course on data analysis taught at the Department of Linguistics, University of Potsdam, Germany. The notes are a compilation from various sources, with some material towards the end that's original and specific to psycholinguistic research. Please note that this is not a stand-alone textbook, so it is sometimes low on explanation, especially about details regarding code.

I sometimes quote almost verbatim from sources, but do not put the quote in double-quotes; I do cite the sources though.

I owe a huge debt to Reinhold Kliegl for patiently educating me on (frequentist) statistical issues in psycholinguistics over the last eight years. In addition, I'd like to thank the statisticians and mathematicians at the School of Mathematics and Statistics, University of Sheffield, in particular, Dr. Kevin Knight, Dr. Jeremy Oakley, and Dr. Fionntan Roukema for their superbly taught courses on statistics and mathematics. Obviously, none of these people are responsible for any errors in these notes.

Please be sure to let me know (vasishth@uni-potsdam.de) if (or perhaps when) you find mistakes.

1
Introduction

1.1 The aims of this course

I'm assuming that you know how to fit and interpret linear models and linear mixed models for analyzing psycholinguistic data. A minimal understanding of the underlying statistical theory is assumed; this is quite a modest requirement: the course I teach in summer is all the background I assume.

It feels like it's time psycholinguists made the leap to using bayesian methods as a standard tool for inference. For the kind of work most of us do, using bayesian tools is very much within our reach. This course will cover some of the key aspects of the bayesian approach to data analysis that are relevant to people who fit linear mixed models on repeated measures data.

1.2 Software needed

Apart from R, we will need JAGS (http://mcmc-jags.sourceforge.net/); and within R we will need the `rjags` package. Both were developed by the epidemiologist/statistician Martyn Plummer. JAGS and rjags are installed in the CIP pool in Haus 14, but you should install them on your own machines too. You might need some other packages; this will become clear as you work through the notes.

The best tutorial I have seen for setting up software for using JAGS is:

`http://spot.colorado.edu/~joka5204/bayes2013.html`

Just follow the instructions provided there to set yourself up.

1.3 How to read these notes

I am expecting that most people attending the course do not know calculus (or have forgotten it). This course is intended to be more of a

practical introduction than a theoretical one. I will therefore walk you through the more technical parts; at no point do you **have** to do any analysis using calculus. Your goal should be to just understand the general idea; I will try to facilitate that process.

The big barrier in understanding bayesian methods is that you need to understand some of the theory. With frequentist methods, you often can get by just knowing "which button to press" (or, at least, neither you nor your reviewer or your audience will ever know that anything went wrong). In Bayesian methodology it's important to know a bit about the what and why. I will try to minimize the mathematical load. It's probably easiest to read these notes (especially the first half) **after** hearing my lectures.

Finally, I should mention that I am still, at the time of writing this (2013), relatively new to bayesian methods, so don't be disappointed if you find that I don't know everything.

These notes are supposed to be self-contained, but I do point the reader to books on bayesian methods. All this reading is strictly optional.

1.4 Short review of linear (mixed) models

Our focus here is on linear mixed models, because this is our bread and butter in psycholinguistics. In this section our only goal is to have some idea of what the components of a linear (mixed) model are. Basically, we need to know how to talk about the variance components and the 'fixed' part because we will need to fit bayesian models using this specification.

The basic linear model that we know how to fit in R has the following linear relationship between the response variable y and the predictor (matrix) X:

$$y = X\beta + \varepsilon \tag{1.1}$$

where
$E(y) = X\beta = \mu \quad E(\varepsilon) = 0$
$Var(y) = \sigma^2 I_n \quad Var(\varepsilon) = \sigma^2 I_n$
I will spell this out a bit in class.

In linear modelling we model the mean of a response $y_1, \ldots, y_n$ as a function of a vector of predictors $x_1, \ldots, x_n$. We assume that the y_i are conditionally independent given X, β. When y's are not marginally independent, we have $Cor(y_1, y_2) \neq 0$, or $P(y_2 \mid y_1) \neq P(y_2)$.

Linear mixed models are useful for correlated data where $\mathbf{y} \mid X, \beta$ are not independently distributed.

Basic specification of LMMs

$$\underset{n\times 1}{Y_i} = \underset{n\times p}{X_i}\ \underset{p\times 1}{\beta} + \underset{n\times q}{Z_i}\ \underset{q\times 1}{b_i} + \underset{n\times 1}{\varepsilon_i} \tag{1.2}$$

where $i = 1,\ldots,m$, let n be total number of data points.

Distributional assumptions:

$b_i \sim N(0,D)$ and $\varepsilon_i \sim N(0,\sigma^2 I)$. D is a $q \times q$ matrix that does not depend on i, and b_i and ε_i are assumed to be independent.

Y_i has a multivariate normal distribution:

$$Y_i \sim N(X_i\beta, V(\alpha)) \tag{1.3}$$

where $V(\alpha) = Z_i D Z_i^T + \sigma^2 I$, and α is the variance component parameters.

Note:

1. D has to be symmetric and positive definite.
2. The Z_i matrix columns are a subset of X_i. In the random intercept model, $Z_i = 1_i$.
3. In the varying intercepts and varying slopes model, $X_i = Z_i = (1_i, X_i)$. Then:

$$Y_i = X_i(\beta + b_i) + \varepsilon_i \tag{1.4}$$

or

$$Y_i = X_i\beta_i + \varepsilon_i \quad \beta_i \sim N(\beta, D) \tag{1.5}$$

$$\begin{aligned} D &= \begin{pmatrix} d_{00} & d_{01} \\ d_{10} & d_{11} \end{pmatrix} \\ &= \begin{pmatrix} d_{00} = Var(\beta_{i0}) & d_{01} = Cov(\beta_{i0},\beta_{i1}) \\ d_{10} = Cov(\beta_{i0},\beta_{i1}) & d_{11} = Var(\beta_{i1}) \end{pmatrix} \end{aligned} \tag{1.6}$$

1.5 σ_b^2 describes both between-block variance, and within block covariance

Consider the following model, a varying intercepts model:

$$Y_{ij} = \mu + b_i + e_{ij}, \tag{1.7}$$

with $b_i \sim N(0,\sigma_b^2)$, $e_{ij} \sim N(0,\sigma^2)$.

Note that variance is a covariance of a random variable with itself, and then consider the model formulation. If we have

$$Y_{ij} = \mu + b_i + \varepsilon_{ij} \tag{1.8}$$

where i is the group, j is the replication, if we *define* $b_i \sim N(0, \sigma_b^2)$, and refer to σ_b^2 as the between group variance, then we must have

$$\begin{aligned} Cov(Y_{i1}, Y_{i2}) =& Cov(\mu + b_i + \varepsilon_{i1}, \mu + b_i + \varepsilon_{i2}) \\ =& \underset{=0}{\uparrow} Cov(\mu, \mu) + \underset{=0}{\uparrow} Cov(\mu, b_i) + \underset{=0}{\uparrow} Cov(\mu, \varepsilon_{i2}) + \\ & \underset{=0}{\uparrow} Cov(b_i, \mu) + \underset{+ve}{\uparrow} Cov(b_i, b_i) \dots \\ =& Cov(b_i, b_i) = Var(b_i) = \sigma_b^2 \end{aligned} \tag{1.9}$$

1.6 Two basic types of linear mixed model and their variance components

1.6.1 Varying intercepts model

The model for a categorical predictor is:

$$Y_{ijk} = \beta_j + b_i + \varepsilon_{ijk} \tag{1.10}$$

$i = 1, \dots, 10$ is subject id, $j = 1, 2$ is the factor level, k is the number of replicates (here 1). $b_i \sim N(0, \sigma_b^2), \varepsilon_{ijk} \sim N(0, \sigma^2)$.

For a continuous predictor:

$$Y_{ijk} = \beta_0 + \beta_1 t_{ijk} + b_{ij} + \varepsilon_{ijk} \tag{1.11}$$

The general form for any model in this case is:

$$\begin{pmatrix} Y_{i1} \\ Y_{i2} \end{pmatrix} \sim N\left(\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}, V \right) \tag{1.12}$$

where

$$V = \begin{pmatrix} \sigma_b^2 + \sigma^2 & \sigma_b^2 \\ \sigma_b^2 & \sigma_b^2 + \sigma^2 \end{pmatrix} = \begin{pmatrix} \sigma_b^2 + \sigma^2 & \rho \sigma_b \sigma_b \\ \rho \sigma_b \sigma_b & \sigma_b^2 + \sigma^2 \end{pmatrix} \tag{1.13}$$

Note also that the mean response for the subject, i.e., *conditional* mean of Y_{ij} given the subject-specific effect b_i is:

$$E(Y_{ij} \mid b_i) = X_{ij}^T \beta + b_i \tag{1.14}$$

The mean response in the population, i.e., the marginal mean of Y_{ij}:

$$E(Y_{ij}) = X_{ij}^T \beta \tag{1.15}$$

The marginal variance of each response is:

$$\begin{aligned} Var(Y_{ij}) =& Var(X_{ij}^T\beta + b_i + \varepsilon_{ij}) \\ =& Var(\beta + b_i + \varepsilon_{ij}) \\ =& \sigma_b^2 + \sigma^2 \end{aligned} \tag{1.16}$$

the covariance between any pair of responses Y_{ij} and $Y_{ij'}$ is given by

$$\begin{aligned} Cov(Y_{ij}, Y_{ij'}) =& Cov(X_{ij}^T\beta + b_i + \varepsilon_{ij}, X_{ij'}^T\beta + b_i + \varepsilon_{ij'}) \\ =& Cov(b_i + e_{ij}, b_i + e_{ij'}) \\ =& Cov(b_i, b_i) = \sigma_b^2 \end{aligned} \tag{1.17}$$

The correlation is

$$Corr(Y_{ij}, Y_{ij'}) = \frac{\sigma_b^2}{\sigma_b^2 + \sigma^2} \tag{1.18}$$

In other words, introducing a random intercept induces a correlation among repeated measurements.

$\hat{V}$ is therefore:

$$\begin{pmatrix} \hat{\sigma}_b^2 + \hat{\sigma}^2 & \hat{\rho}\hat{\sigma}_b\hat{\sigma}_b \\ \hat{\rho}\hat{\sigma}_b\hat{\sigma}_b & \hat{\sigma}_b^2 + \hat{\sigma}^2 \end{pmatrix} \tag{1.19}$$

Note: $\hat{\rho} = 1$. But this correlation is not *estimated* in the varying intercepts model (note: I am not 100% sure about this last sentence; I'm still working on understanding this part).

1.6.2 Varying intercepts and slopes (with correlation)

The model for a categorical predictor is:

$$Y_{ij} = \beta_1 + b_{1i} + (\beta_2 + b_{2i})x_{ij} + \varepsilon_{ij} \quad i = 1,...,M, j = 1,...,n_i \tag{1.20}$$

with $b_{1i} \sim N(0, \sigma_1^2), b_{2i} \sim N(0, \sigma_2^2)$, and $\varepsilon_{ij} \sim N(0, \sigma^2)$.

Note: I have seen this presentation elsewhere:

$$Y_{ijk} = \beta_j + b_{ij} + \varepsilon_{ijk} \tag{1.21}$$

$b_{ij} \sim N(0, \sigma_b)$. The variance σ_b must be a 2×2 matrix:

$$\begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix} \tag{1.22}$$

The general form for the model is:

$$\begin{pmatrix} Y_{i1} \\ Y_{i2} \end{pmatrix} \sim N\left(\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}, V \right) \tag{1.23}$$

where

$$V = \begin{pmatrix} \sigma^2_{b,A} + \sigma^2 & \rho\sigma_{b,A}\sigma_{b,B} \\ \rho\sigma_{b,A}\sigma_{b,B} & \sigma^2_{b,B} + \sigma^2 \end{pmatrix} \tag{1.24}$$

Basically, when we fit LMMs in JAGS we will simply state the above model in BUGS syntax.

Example: Reading Time data This is a two-condition experiment, self-paced reading data kindly released by Gibson and Wu.[1] This dataset is unique because we will make a replication attempt in an exercise I will give out later in the course. The full analysis is discussed in my PLoS One paper[2].

[1] E. Gibson and I. Wu. Processing Chinese relative clauses in context. Language and Cognitive Processes (in press), 2011

[2] Shravan Vasishth, Zhong Chen, Qiang Li, and Gueilan Guo. Processing Chinese relative clauses: Evidence for the subject-relative advantage. *PLoS ONE*, 8(10):e77006, 10 2013

Here is the head of the data frame. Note that we have full crossing between subjects and items, but there is some missing data from subj 27 (no idea why):

```
> head(critdata)

   subj item pos   rt region            type2   so
7     1   13   6 1140    de1   object relative  0.5
20    1    6   6 1197    de1  subject relative -0.5
32    1    5   6  756    de1   object relative  0.5
44    1    9   6  643    de1   object relative  0.5
60    1   14   6  860    de1  subject relative -0.5
73    1    4   6  868    de1  subject relative -0.5

> head(xtabs(~subj+item,critdata))

    item
subj 1 2 3 4 5 6 7 8 9 10 11 13 14 15 16
   1 5 5 5 5 5 5 5 5 5  5  5  5  5  5  5
   2 5 5 5 5 5 5 5 5 5  5  5  5  5  5  5
   3 5 5 5 5 5 5 5 5 5  5  5  5  5  5  5
   4 5 5 5 5 5 5 5 5 5  5  5  5  5  5  5
   5 5 5 5 5 5 5 5 5 5  5  5  5  5  5  5
   6 5 5 5 5 5 5 5 5 5  5  5  5  5  5  5

> ## subj 27:
> xtabs(~subj+item,critdata)[24,]

 1  2  3  4  5  6  7  8  9 10 11 13 14 15 16
 5  0  5  0  0  0  0  5  5  0  0  5  0  5  5
```

We fit two LMMs (varying intercepts, and varying intercepts and slopes):

```
> library(lme4)
> ## varying intercepts:
> m0<-lmer(rt~so+(1|subj),subset(critdata,region=="headnoun"))
> library(car)
> qqPlot(residuals(m0))
> qqPlot(ranef(m0)$subj)
> ## varying intercepts and slopes:
> m1<-lmer(rt~so+(1+so|subj),subset(critdata,region=="headnoun"))
> VarCorr(m1)

 Groups   Name        Std.Dev. Corr
 subj     (Intercept) 155
          so          186      -1.00
 Residual             569

>
>
```

The varying intercepts model can be written as:

$$Y_{ijk} = \beta_j + b_i + \varepsilon_{ijk} \tag{1.25}$$

$i = 1,\ldots,10$ is subject id, $j = 1,2$ is the factor level, k is the number of replicates. $b_i \sim N(0,\sigma_b^2), \varepsilon_{ijk} \sim N(0,\sigma^2)$.

The varying intercepts and slopes model can be written:

$$Y_{ij} = \beta_1 + b_{1i} + (\beta_2 + b_{2i})x_{ij} + \varepsilon_{ij} \quad i = 1,...,M, j = 1,...,n_i \tag{1.26}$$

with $b_{1i} \sim N(0,\sigma_1^2), b_{2i} \sim N(0,\sigma_2^2)$, and $\varepsilon_{ij} \sim N(0,\sigma^2)$.

For the varying intercepts and slopes model we can also say the following. Here, i indexes the subject id, subjects are assumed numbered from 1,...,37.

$$\begin{pmatrix} Y_{i1} \\ Y_{i2} \end{pmatrix} \sim N\left(\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}, V\right) \tag{1.27}$$

where

$$V = \begin{pmatrix} \sigma_{b,A}^2 + \sigma^2 & \rho\sigma_{b,A}\sigma_{b,B} \\ \rho\sigma_{b,A}\sigma_{b,B} & \sigma_{b,B}^2 + \sigma^2 \end{pmatrix} \tag{1.28}$$

Here is a class exercise to get us warmed up. For each of the two models, state

1. What the model matrices X and the estimates of β are (and what these estimates mean).

   Answer:

```
    (Intercept)   so
94            1  0.5
221           1 -0.5
341           1  0.5
461           1  0.5
621           1 -0.5
753           1 -0.5

    (Intercept)   so
94            1  0.5
221           1 -0.5
341           1  0.5
461           1  0.5
621           1 -0.5
753           1 -0.5

(Intercept)          so
     549.11     -122.12

[1] 548.44

 object relative subject relative
          122.86          -124.21

(Intercept)          so
     549.07     -123.51
```

2. What the estimated variance components of the covariance matrix V is.

   Answer:

3. Whether all model assumptions are met.

   Answer:

```
> library(car)
> ## residuals assumption not met:
> qqPlot(residuals(m0))
> qqPlot(residuals(m1))
> ## random effects (BLUPs') assumptions met:
> qqPlot(ranef(m0)$subj)
> qqPlot(ranef(m1)$subj[,1])
> qqPlot(ranef(m1)$subj[,2])
```

1.7 Bayesian data analysis

Background reading: A major source and reference for these notes is Scott Lynch's book[3]. I consider this book to be the best book introducing bayesian data analysis out there. However, it does assume that you know at least single variable differential and integral calculus. Incidentally, a good book that covers all the mathematics you would need to read the Lynch book is available: Gilbert and Jordan.[4]

In my notes I go a bit easier on the reader; it's possible to get by in this course without really knowing calculus (although I will have to show some equations).

If you are reasonably good with calculus and linear algebra and probability theory, read Lynch's book and the references I provide at the end rather than my notes.

The basic idea we will explore in this course is Bayes' theorem, which allows us to learn from experience and turn a prior distribution into a posterior distribution given data. I stole this neat summary from Lunn et al.[5]

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.

[3] Scott Michael Lynch. *Introduction to applied Bayesian statistics and estimation for social scientists*. Springer, 2007

[4] J. Gilbert and C.R. Jordan. *Guide to mathematical methods*. Macmillan, 2002

[5] David Lunn, Chris Jackson, David J Spiegelhalter, Nicky Best, and Andrew Thomas. *The BUGS book: A practical introduction to Bayesian analysis*, volume 98. CRC Press, 2012

1.8 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.

1.9 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 characterize 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.[6]

   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.

[6] Peter M Lee. *Bayesian statistics: An introduction*. John Wiley & Sons, 2012

Bayesians sometimes get pretty worked up about their approach. The goal in this course is not to convert you to a new religion, but rather to learn the methodology. Sometimes bayesian methods make sense, and other times frequentist methods do.

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 that a null hypothesis is true given the data; p-values doesn't tell you anything exactly about the actual **research** hypothesis.

Of course, it remains to be seen whether people would abuse Bayesian methods once they get the hang of it! (I guess one thing will never change: in the hands of an amateur, statistics is a devastating weapon. I should know! I have made horrible mistaks in the past, and probably will in the future too.)

1.10 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,[7] 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 $P(data \mid parameter)$.

[7] This is the maximum likelihood estimate.

1.11 How we will proceed

We will spend some time reviewing very basic probability theory, and spend some time understanding what a **probability density/mass function** is. Then we will look at the important concepts of **joint, marginal, and conditional distributions**; here, you don't need to know how to derive anything, you just need to know what these are. I will show some some derivations, though. Next, we will study **maximum likelihood**. Then we will get into **bayesian data analysis**; this will be the more practical part of the course. There will be one theoretical excursion in this practical part; the study of **Monte Carlo Markov Chain** sampling techniques. But even this will be relatively

painless. In the last part of the course we will work through several typical case studies and exercises, using examples from psycholinguistics, usually from my own lab, but we will also use other people's data when it's available.

At the end of the course, I will give some advice on what to read next. There are some really good books on bayesian data analysis, but there are also a lot of very obscure ones that only statistians or mathematicians can follow. Unfortunately, statisticians seem to generally write only for other statisticians; there are very few of them out there who can communicate to the lay world or even the semi-lay world. But there are a few.

2

Probability theory and probability distributions

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.[1] This chapter very closely based on this book.

As Christensen et al.[2] 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] G. Jay Kerns. *Introduction to Probability and Statistics Using R.* 2010

[2] Ronald Christensen, Wesley O Johnson, Adam J Branscum, and Timothy E Hanson. *Bayesian ideas and data analysis: An introduction for scientists and statisticians.* CRC Press, 2011

2.1 Kolmogorov Axioms of Probability

[I assume some knowledge of set theory here, but I will spell this out in class.]

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. **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

$$\mathbb{P}\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} \mathbb{P}(A_i) \text{ for every } n, \tag{2.1}$$

and furthermore,

$$\mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mathbb{P}(A_i). \tag{2.2}$$

2.2 Three important propositions

Proposition 1. *Let $E \cup E^c = S$. Then,*

$$1 = P(S) = P(E \cup E^c) = P(E) + P(E^c) \tag{2.3}$$

or:

$$P(E^c) = 1 - P(E) \tag{2.4}$$

Proposition 2. *If $E \subset F$ then $P(E) \leq P(F)$.*

Proposition 3.

$$P(E \cup F) = P(E) + P(F) - P(EF) \tag{2.5}$$

2.2.1 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

$$\mathbb{P}(B \mid A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)}, \quad \text{if } \mathbb{P}(A) > 0. \tag{2.6}$$

Theorem 1. *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:

$$\mathbb{P}\left(\bigcup_{k=1}^{\infty} B_k \,\middle|\, A\right) = \sum_{k=1}^{\infty} \mathbb{P}(B_k|A). \tag{2.7}$$

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,

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B|A). \tag{2.8}$$

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

$$\mathbb{P}(A_1 \cap A_2 \cap \cdots \cap A_n) = \mathbb{P}(A_1)\mathbb{P}(A_2|A_1)\cdots\mathbb{P}(A_n|A_1 \cap A_2 \cap \cdots \cap A_{n-1}). \tag{2.9}$$

2.3 Independence of events

[Taken nearly verbatim from Kerns.]

Definition 1. *Events A and B are said to be independent if*

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B). \tag{2.10}$$

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

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}. \tag{2.11}$$

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

$$\mathbb{P}(A|B) = \mathbb{P}(A) \text{ when } A, B \text{ are independent.} \tag{2.12}$$

Proposition 4. *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,

$$\begin{aligned} P(E) =& P(EF) + P(EF^c) \\ =& P(E)P(F) + P(EF^c) \end{aligned} \tag{2.13}$$

Equivalently:

$$\begin{aligned} P(EF^c) =& P(E)[1 - P(F)] \\ =& P(E)P(F^c) \end{aligned} \tag{2.14}$$

2.4 Bayes' rule

[Quoted nearly verbatim from Kerns.]

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*

$$\mathbb{P}(B_k|A) = \frac{\mathbb{P}(B_k)\mathbb{P}(A|B_k)}{\sum_{i=1}^{n} \mathbb{P}(B_i)\mathbb{P}(A|B_i)}, \quad k = 1, 2, \ldots, n. \tag{2.15}$$

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

$$\mathbb{P}(A)\mathbb{P}(B_k|A) = \mathbb{P}(B_k \cap A) = \mathbb{P}(B_k)\mathbb{P}(A|B_k). \tag{2.16}$$

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

$$\mathbb{P}(B_k|A) = \frac{\mathbb{P}(B_k)\mathbb{P}(A|B_k)}{\mathbb{P}(A)}. \tag{2.17}$$

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

$$\mathbb{P}(A) = \sum_{k=1}^{n} \mathbb{P}(B_k \cap A) = \sum_{k=1}^{n} \mathbb{P}(B_k)\mathbb{P}(A|B_k). \tag{2.18}$$

2.5 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.)

$$p_X : S_X \to [0, 1] \tag{2.19}$$

defined by

$$p_X(x) = P(X(\omega) = x), x \in S_X \tag{2.20}$$

[**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

$$X \sim g(\cdot) \tag{2.21}$$

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

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp[-\frac{(x-\mu)^2}{2\sigma^2}] \tag{2.22}$$

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:

$$P(x < X) = F(x < X) = \int_{-\infty}^{X} f(x)\, dx \tag{2.23}$$

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:

$$f(x) = \exp[-\frac{(x-\mu)^2}{2\sigma^2}] \tag{2.24}$$

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)

2.5066 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))))
+ }
> 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")
> ## area under the curve:
> integrate(norm,lower=-Inf,upper=Inf)
```

```
1 with absolute error < 9.4e-05
```

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,

$$P\{X \in B\} = \int_B f(x)\,dx \tag{2.25}$$

Kerns has the following to add about the above:

> Continuous random variables have supports that look like
>
> $$S_X = [a,b] \text{ or } (a,b), \tag{2.26}$$
>
> 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
>
> $$\mathbb{P}(X \in A) = \int_c^d f_X(x)\,\mathrm{d}x. \tag{2.27}$$

- 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:

$$\mathbb{P}(a \leq X \leq b) = \mathbb{P}(a < X \leq b) = \mathbb{P}(a \leq X < b) = \mathbb{P}(a < X < b). \quad (2.28)$$

- 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

$$1 = P\{X \in (-\infty, \infty)\} = \int_{-\infty}^{\infty} f(x)\,dx \quad (2.29)$$

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

$$P\{a \leq X \leq b\} = \int_{a}^{b} f(x)\,dx \quad (2.30)$$

If $a = b$, we get

$$P\{X = a\} = \int_{a}^{a} f(x)\,dx = 0 \quad (2.31)$$

Hence, for any continuous random variable,

$$P\{X < a\} = P\{X \leq a\} = F(a) = \int_{-\infty}^{a} f(x)\,dx \quad (2.32)$$

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

$$\frac{dF(a)}{da} = f(a) \quad (2.33)$$

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

Basic results (proofs omitted):

1.
$$E[X] = \int_{-\infty}^{\infty} x f(x)\, dx \tag{2.34}$$

2.
$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f(x)\, dx \tag{2.35}$$

3.
$$E[aX + b] = aE[X] + b \tag{2.36}$$

4.
$$Var[X] = E[(X - \mu)^2] = E[X^2] - (E[X])^2 \tag{2.37}$$

5.
$$Var(aX + b) = a^2 Var(X) \tag{2.38}$$

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.

2.6 What can you do with a pdf?

You can:

1. Calculate the mean:

   Discrete case:

$$E[X] = \sum_{i=1}^{n} x_i p(x_i) \tag{2.39}$$

   Continuous case:

$$E[X] = \int_{-\infty}^{\infty} x f(x)\, dx \tag{2.40}$$

2. Calculate the variance:

$$Var(X) = E[X^2] - (E[X])^2 \tag{2.41}$$

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

$$\int_{-\infty}^{Q} f(x)\, dx \tag{2.42}$$

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

$$\int_{0}^{1} f(x)\, dx \tag{2.43}$$

We can do this in R as follows:[3]

```
> integrate(function(x) dnorm(x, mean = 0, sd = 1),
+ lower=0,upper=1)

0.34134 with absolute error < 3.8e-15

> ## alternatively:
> pnorm(1)-pnorm(0)

[1] 0.34134
```

[3] 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.

Homework 1

This assignment is optional.

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 $1 = c\int_0^1 \theta^2 d\theta$.

1. Find c.
2. Find the mean, median (hint: what is the median in terms of quantiles?) and variance of the above pdf.
3. Find the 95% credible interval, i.e., the lower and upper values in $P(lower < \theta < upper) = 0.95$.

2.7 Some basic facts about expectation and variance

1. Computing expectation:

$$E[X] = \sum_{i=1}^{n} x_i p(x_i) \tag{2.44}$$

$$E[X] = \int_{-\infty}^{\infty} x f(x)\, dx \tag{2.45}$$

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

$$E[g(X)] = \sum_{i=1}^{n} g(x_i) p(x_i) \tag{2.46}$$

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f(x)\, dx \tag{2.47}$$

3. Computing variance:

$$Var(X) = E[(X-\mu)^2] \tag{2.48}$$

$$Var(X) = E[X^2] - (E[X])^2 \tag{2.49}$$

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

$$Var(aX+b) = a^2 Var(X) \tag{2.50}$$

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

5.

$$SD(X) = \sqrt{Var(X)} \tag{2.51}$$

6. For two independent random variables X and Y,

$$E[XY] = E[X]E[Y] \tag{2.52}$$

7. Covariance of two random variables:

$$Cov(X,Y) = E[(X-E[X])(Y-E[Y])] \tag{2.53}$$

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

Corollary in 4.1 of Ross:[4]

[4] Sheldon Ross. *A first course in probability*. Pearson Education, 2002

$$E[aX+b] = aE[X]+b \tag{2.54}$$

9. A related result is about **linear combinations of RVs**:

Theorem 3. *Given two* ***not necessarily independent*** *random variables X and Y:*

$$E[aX+bY] = aE[X]+bE[Y] \tag{2.55}$$

10. If X and Y are independent,

$$Var(X+Y) = Var[X]+Var[Y] \tag{2.56}$$

and

$$Var(aX+bY) = a^2 Var(X)+b^2 Var(Y) \tag{2.57}$$

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

$$Var(X-Y) = Var(X)+Var(Y) \tag{2.58}$$

11. If X and Y are not independent, then

$$Var(X-Y) = Var(X)+Var(Y)-2Cov(X,Y) \tag{2.59}$$

2.8 Three common (and, for us, important) distributions

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

$$P(x \mid n,p) = \binom{n}{k} p^k (1-p)^{n-k} \tag{2.60}$$

[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)
```

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

$$f_X(x) = \begin{cases} \frac{1}{\beta-\alpha}, & \alpha < x < \beta, \\ 0, & \text{otherwise} \end{cases} \tag{2.61}$$

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:

$$F_X(a) = \begin{cases} 0, & a < 0, \\ \frac{a-\alpha}{\beta-\alpha}, & \alpha \le t < \beta, \\ 1, & a \ge \beta. \end{cases} \tag{2.62}$$

$$E[X] = \frac{\beta+\alpha}{2} \tag{2.63}$$

$$Var(X) = \frac{(\beta-\alpha)^2}{12} \tag{2.64}$$

```
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)
```

Normal random variable

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \quad -\infty < x < \infty. \tag{2.65}$$

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

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:

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{-y^2}{2}} dy \quad \text{where } y = (x - \mu)/\sigma \tag{2.66}$$

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

$$\Phi(-x) = 1 - \Phi(x), \quad -\infty < x < \infty \tag{2.67}$$

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

$$p\{Z \leq -x\} = P\{Z > x\}, \quad -\infty < x < \infty \tag{2.68}$$

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:

$$F_X(a) = P\{X \leq a\} = P\left(\frac{X - \mu}{\sigma} \leq \frac{a - \mu}{\sigma}\right) = \Phi\left(\frac{a - \mu}{\sigma}\right) \tag{2.69}$$

The standardized version of a normal random variable X is used to compute specific probabilities relating to X (it's 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 $u = -x^2/2$.

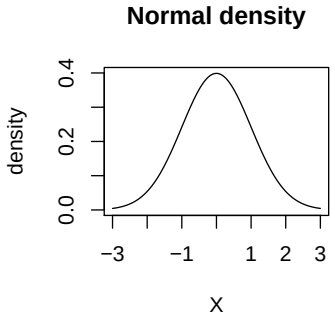


Figure 2.1: Normal distribution.

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

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

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

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \underset{u}{\underset{\uparrow}{x}} \underset{dv/dx}{\underset{\uparrow}{x e^{-x^2/2}}}\,dx = \frac{1}{\sqrt{2\pi}} \underset{u}{\underset{\uparrow}{x}} \underset{v}{\underset{\uparrow}{-e^{-x^2/2}}} - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \underset{v}{\underset{\uparrow}{-e^{-x^2/2}}} \underset{du/dx}{\underset{\uparrow}{1}}\,dx = 1$$

[Explained on p. 274 of Grinstead and Snell's online book[5]; it wasn't obvious to me, and Ross[6] is pretty terse]: "The first summand above can be shown to equal 0, since as $x \to \pm\infty$, $e^{x^2/2}$ 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."

[5] C.M. Grinstead and J.L. Snell. *Introduction to probability.* American Mathematical Society, 1997

[6] Sheldon Ross. *A first course in probability.* Pearson Education, 2002

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 $N(10, 16/4)$.

2.9 Distribution of a function of a random variable (transformations of random variables)

[This section can be skipped.]

A nice and intuitive description:

Consider a continuous RV Y which is a continuous differentiable increasing function of X:

$$Y = g(X) \tag{2.70}$$

Because g is differentiable and increasing, g' and g^{-1} are guaranteed to exist. Because g maps all $x \leq s \leq x + \Delta x$ to $y \leq s \leq y + \Delta y$, we can say:

$$\int_x^{x+\Delta x} f_X(s)\,ds = \int_y^{y+\Delta y} f_Y(t)\,dt \tag{2.71}$$

Therefore, for small Δx:

$$f_Y(y)\Delta y \approx f_X(x)\Delta x \tag{2.72}$$

Dividing by Δy we get:

$$f_Y(y) \approx f_X(x)\frac{\Delta x}{\Delta y} \tag{2.73}$$

Theorem 4 (**Theorem 7.1 in Ross**). *Let X be a continuous random variable having probability density function f_X. Suppose that $g(x)$ is a strict monotone (increasing or decreasing) function, differentiable and (thus continuous) function of x. Then the random variable Y defined by $Y = g(X)$ has a probability density function defined by*

$$f_Y(y) = \begin{cases} f_X(g^{-1}(y)) \mid \frac{d}{dx} g^{-1}(y) \mid & \text{if } y = g(x) \text{ for some } x \\ 0 & \text{if } y \neq g(x) \text{ for all } x. \end{cases}$$

where $g^{-1}(y)$ is defined to be equal to the value of x such that $g(y-y)$.

Proof:

Suppose $y = g(x)$ for some x. Then, with $Y = g(X)$,

$$\begin{aligned} F_Y(y) =& P(g(X) \leq y) \\ =& P(X \leq g^{-1}(y)) \\ =& F_X(g^{-1}(y)) \end{aligned} \tag{2.74}$$

Differentiation gives

$$f_Y(y) = f_X(g^{-1}(y))\frac{d(g^{-1}(y))}{dy} \tag{2.75}$$

Detailed explanation for the above equation: Since

$$\begin{aligned} F_Y(y) =& F_X(g^{-1}(y)) \\ =& \int f_X(g^{-1}(y))\, dy \end{aligned} \tag{2.76}$$

Differentiating:

$$\frac{d(F_Y(y))}{dy} = \frac{d}{dy}(F_X(g^{-1}(y))) \tag{2.77}$$

We use the chain rule. To simplify things, rewrite $w(y) = g^{-1}(y)$ (otherwise typesetting things gets harder). Then, let

$$u = w(y)$$

which gives

$$\frac{du}{dy} = w'(y)$$

and let

$$x = F_X(u)$$

This gives us

$$\frac{dx}{du} = F'_X(u) = f_X(u)$$

By the chain rule:

$$\frac{du}{dy} \times \frac{dx}{du} = w'(y) f_X(u) = \underset{\substack{\uparrow \\ \text{plugging in the variables}}}{\frac{d}{dy}(g^{-1}(y)) f_X(g^{-1}(y))}$$

■

Exercises (optional):

1. $Y = X^2$
2. $Y = \sqrt{X}$
3. $Y = \mid X \mid$
4. $Y = aX + b$

2.10 Class activities

2.10.1 Exercise 1: Actual coin toss experiment

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, we can write down the number of cases we have of each possible outcome, along with the theoretical probability.

Number of heads:	0	1	2	3	4	5	6	7	8	9	10
Theoretical probability:											
Count:											

2.10.2 Exercise 2: Probability theory

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.

Fill in the table below.

$f(\cdot)$	Prob.	Answer
Unif(0,10)	$P(X < 3) =$	
	$P(X > 11) =$	
	$P(X = 6) =$	
N(100,20)	$P(X < 3) =$	
	$P(X > 11) =$	
	$P(X = 6) =$	
Binom(p=.6,n=20)	$P(X < 3) =$	
	$P(X > 11) =$	
	$P(X = 6) =$	

2.10.3 Exercise 3: Normally distributed data with known variance

Here we will consider normally distributed data with unknown mean but known variance.

Before getting into the exercise, I want to first introduce the idea of the **likelihood function**.

Suppose $X_1, \ldots, X_n$ is a random variable that comes from a normally distributed population with mean μ and variance σ^2. Suppose that the sample values $x_1, \ldots, x_n$ are independent.

Because these values are independent, the joint probability of getting these values is just the product of the individual probabilities:

$$\begin{aligned} P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) &= P(X_1 = x_1)P(X_2 = x_2)\ldots P(X_n = x_n) \\ &= f(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n; \theta) \end{aligned} \tag{2.78}$$

The last line $f(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n; \theta)$ means: "the joint probability of the values $x_1, \ldots, x_n$ given a specific value for the parameter(s) θ." Here, θ is the vector $< \mu >$.[7]

[7] If σ^2 were unknown, θ would be $< \mu, \sigma^2 >$.

For different values of θ, we would get different values for the function $f(\cdot)$. In other words, we can talk about a function $L(\theta)$ given the data, that delivers different values for each value of θ. This is called the **likelihood function**.

In other words:

$$L(x \mid \theta) = \prod N(x_i; \mu, \sigma^2) \tag{2.79}$$

$$= (\tfrac{1}{\sigma\sqrt{2\pi}})^n \exp(-\tfrac{1}{2\sigma^2}\textstyle\sum(x_i - \mu)^2) \tag{2.80}$$

$$\tag{2.81}$$

We assume that we already know σ^2. Given the likelihood function, we can ask: What is the value of μ that would maximize the probability of this data?

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

$$\hat{\mu} = \frac{1}{n}\sum x_i = \bar{x} \tag{2.82}$$

Suppose that we have independent data as follows:

```
> x<-c(15.7,14.7,16.1,16.2,15.1)
```

We typically start by calculating the mean (among other things):

```
> mean(x)

[1] 15.56
```

Using this sample mean (and the given variance), we make a best guess as to the pdf that is assumed to have generated **each data point**:

$$X \sim N(\bar{x}, \sigma^2)$$

For n data points (as you will no doubt recall from the introductory course in the summer semester), we can characterize the sampling distribution of the sample means as:

$$\bar{X} \sim N(\bar{x}, \sigma^2/n)$$

So let us suppose we have independent and identically distributed data $x = \{x_1, \ldots, x_n\}$, where $X_i \sim N(\mu, \sigma^2)$. Assume that σ^2 is known as above, but μ is unknown.

It follows that the pdf for random variable representing any one data point is $N(\bar{x}, \sigma^2)$.

Now we make the big bayesian move. Let our prior be that the sampling distribution of μ is $N(m, v)$. Note that this is the prior for the sampling distribution of the mean μ. It represents what we believe to be true about μ, including our degree of uncertainty about our belief.

Then, given Bayes' theorem

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

or

$$\text{Posterior} \propto \underset{\substack{\uparrow \\ prior}}{N(m, v)} \times \underset{\substack{\uparrow \\ likelihood}}{N(\bar{x}, \sigma^2/n)}$$

it is straightforward to derive analytically (proof omitted, but see Lynch textbook) that the posterior will be $N(m*, v*)$, where

$$v* = \frac{1}{\frac{1}{v} + \frac{n}{\sigma^2}} \tag{2.83}$$

$$m* = v* \left(\frac{m}{v} + \frac{n\bar{x}}{\sigma^2} \right) \tag{2.84}$$

Importantly, we can rewrite m* as a weighted mean of the prior and the sample mean:

$$m* = \frac{w_1}{w_1 + w_2} m + \frac{w_2}{w_1 + w_2} \bar{x} \tag{2.85}$$

where $w_1 = v^{-1}$ and $w_2 = (\frac{\sigma^2}{n})^{-1}$

Because of results like the one above, it is easier to talk about variance in terms of **precision=1/variance**. In JAGS and pretty much any other bayesian data analysis software, everything is stated in terms of precision, e.g., normal distributions are defined as N(mean,precision), not N(mean,variance).

Example of how we can use the above analytical result:

Let the data be: 15.7,14.7,16.1,16.2,15.1. Variance (σ^2) known at 0.6. Let the prior be N(m=14,v=1). We will work out the posterior distribution using the formula and using JAGS.

Note the differences between R and JAGS/BUGS, and statistical conventions in textbooks:

1. In textbooks, the normal distribution is represented by $N(\mu, \sigma^2)$, i.e., the uncertainty is in terms of **variance**.
2. In R, we have $N(\mu, \sigma)$, i.e., the uncertainty is in terms of **standard deviation**.
3. In JAGS and BUGS in general, we have $N(\mu, \frac{1}{\sigma^2})$, i.e., the uncertainty is in terms of **precision**.

I'm sorry for this confusing mess, but it's the statisticans' fault that they don't keep things straight. **Don't forget this detail in the coming lectures.**

```
> derive.post<-function(n,x.bar,sigma2,m,v){
+ v.star<- 1/( (1/v) + n/sigma2 )
+ m.star<-v.star*(m/v + (n*x.bar/sigma2))
+ return(list(m.star,v.star))
+ }
> data<-list(y=c(15.7,14.7,16.1,16.2,15.1))
> ## let this data come from N(mu,sigma2=.6)
> ## Let prior be N(14,1)
>
> (post<-derive.post(n=length(data$y),
+                    x.bar=mean(data$y),
+                    sigma2=0.6,m=14,v=1))

[[1]]
[1] 15.393

[[2]]
[1] 0.10714
```

You can now play with the weighting of the prior and data.

First, make the prior variance very low; this yields a shift of the posterior towards the prior, with low variance:

```
> (post<-derive.post(n=length(data$y),x.bar=mean(data$y),
+                    sigma2=.6,m=14,v=0.0001))

[[1]]
[1] 14.001

[[2]]
[1] 9.9917e-05
```

Next, make the prior variance very high; this shifts the posterior towards the sample mean and variance:

```
> (post<-derive.post(n=length(data$y),x.bar=mean(data$y),
+                    sigma2=.6,m=14,v=100))

[[1]]
[1] 15.558

[[2]]
[1] 0.11986

> mean(data$y); var(data$y)

[1] 15.56
```

```
[1] 0.418
```

This basically sums up a situation we will encounter in this course: we will use priors that express low certainty ("non-informative" priors), letting "the data speak for themselves." But in cases where we do have prior information, we will use it! (This is impossible to do in frequentist settings) You will see how prior beliefs can be incorporated into your analysis.

Question: what role does sample size n play in the computation of the posterior? When sample size is increased in the above example, will the posterior lean towards the prior or the likelihood?

Next, we fit a JAGS model to do the same calculation that we did analytically above, but now using something called **Monte Carlo Markov Chain sampling**. Right now we can ignore what exactly that is and "just do it".

The JAGS model will have the following components:

1. The **model specification**: This will be a model written as a text file from R, using the `cat` command.
2. The **data specification**: This must be a list, see example below.
3. Define which parameters you want to look at the posterior distribution of; we will conventionally use a vector called `track.variables` to keep track of these.
4. Run the `jags.model` function with parameters and save the result as an object, say `normalex1.mod`. Conventionally we will use the extension .mod for model objects.

```
jags.model(
    ## your data as a list:
    data = data,
    ## model location in hard drive:
    file = "JAGSmodels/normalexercise1.jag",
    ## ignore all this right now:
    n.chains = 4,
    n.adapt =2000 ,quiet=T)
```

5. Run the command `coda.samples` using the following syntax, and save the result as say `normalex1.res`. We will conventionally save results with the extension .res.

```
normalex1.res<-coda.samples( normalex1.mod,
              var = track.variables,
              ## ignore all this for now:
              n.iter = 100000,
              thin = 50 )
```

6. Plot posterior distribution using

```
plot(normalex1.res)
```

7. Print out summary of posterior distribution using

```
summary(normalex1.res)
```

So here we go:

```
> ## model specification:
> cat("
+ model
+    {
+ for(i in 1:5){
+     ## note specification in terms of precision:
+   y[i] ~ dnorm(mu[i],1/0.6)
+   mu[i] <- beta0
+ }
+   ##prior
+   beta0 ~ dnorm(14,1)
+ }",
+     file="JAGSmodels/normalexercise1.jag" )
> ## data:
> data<-list(y=c(15.7,14.7,16.1,16.2,15.1))
> ## specify variables to track
> ## the posterior distribution of:
> track.variables<-c("beta0")
> ## load rjags library:
> library(rjags,quietly=T)
> ## define model:
> normalex1.mod <- jags.model(
+     data = data,
+     file = "JAGSmodels/normalexercise1.jag",
+     n.chains = 4,
+     n.adapt =2000 ,quiet=T)
> ## run model:
> normalex1.res <- coda.samples( normalex1.mod,
+                                    var = track.variables,
+                                    n.iter = 100000,
+                                    thin = 50 )
> ## summarize and plot:
> summary(normalex1.res)

Iterations = 50:1e+05
Thinning interval = 50
```

```
Number of chains = 4
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean             SD       Naive SE Time-series SE
      15.38461        0.32313        0.00361        0.00359

2. Quantiles for each variable:

 2.5%   25%   50%   75% 97.5%
 14.8  15.2  15.4  15.6  16.0
```

```
> plot(normalex1.res)
> ## convergence diagnostic:
> ## (to be explained later)
> gelman.diag(normalex1.res)

Potential scale reduction factors:

      Point est. Upper C.I.
beta0          1          1
```

We can visualize the prior, likelihood, posterior:

```
> theta<-seq(0,18,by=0.01)
> ## lik: the sampling distribution of the mean:
> plot(function(x) dnorm(x,mean=mean(data$y),sd=sqrt(0.6/5)),
+      12, 18,
+      ylab="density",xlab="X",ylim=c(0,1.3))
> ## prior
> lines(theta,dnorm(theta,mean=14,sd=1),lty=2)
> #posterior from JAGS:
> ## mean=15.40056, sd=0.32127
> lines(theta,dnorm(theta,mean=15.40056,sd=0.32127),col="red")
```

Final exercise: Visualize the prior, likelihood and posterior for v=0.001, v=100. Obtain the posterior distribution's parameters using both the formula and by generating the posterior sample using (modified) JAGS code.

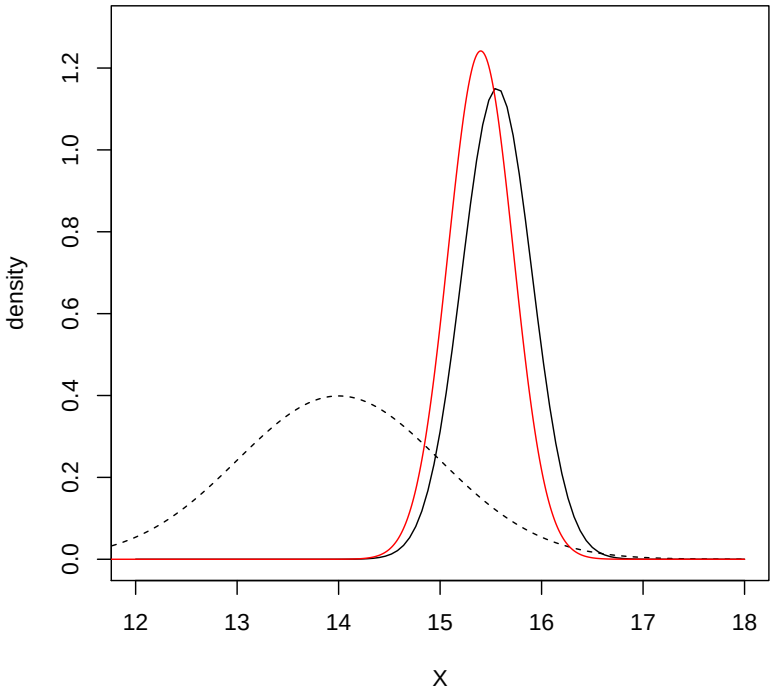


Figure 2.2: Visualization of prior, likelihood, posterior.

3
Important distributions

3.1 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

$$f_{X_1,\ldots,X_k}(x_1,\ldots,x_k) \quad = \quad \binom{n}{x_1\, x_2\, \cdots\, x_k} p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k}, \tag{3.1}$$

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

$$(X_1,\ldots,X_k) \sim \mathsf{multinom}(\mathtt{size} = n, \mathtt{prob} = \mathbf{p}_{\mathrm{k}\times 1}). \tag{3.2}$$

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

$$\binom{n}{x_1\, x_2\, \cdots\, x_k} = \frac{n!}{x_1!x_2!\cdots x_k!} \tag{3.3}$$

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:

$$f_{X_1,\ldots,X_k}(x_1,\ldots,x_k) = \binom{n}{x_1\, x_2 \cdots x_k} p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k} \quad (3.4)$$

Plugging in the values:

$$f_{X_1,\ldots,X_k}(x_1,\ldots,x_k) = \binom{9}{3\,2\,2\,1\,1\,0} \frac{1}{6}^3 \frac{1}{6}^2 \frac{1}{6}^2 \frac{1}{6}^1 \frac{1}{6}^1 \frac{1}{6}^0 \quad (3.5)$$

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

3.2 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

$$f_X(x) = \mathbb{P}(X = x) = \mathrm{e}^{-\lambda}\frac{\lambda^x}{x!}, \quad x = 0,1,2,\ldots \quad (3.6)$$

3.3 Geometric distribution

From Ross [1] (page 155):

[1] Sheldon Ross. *A first course in probability*. Pearson Education, 2002

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:

$$\sum_{i=1}^{\infty} P(X = n) = p\sum_{i=1}^{\infty}(1-p)^{n-1} \underset{\text{see geometric series section.}}{\uparrow} = \frac{p}{1-(1-p)} = 1$$

Mean and variance of the geometric distribution

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

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

For proofs, see Ross [2] (pages 156-157).

[2] Sheldon Ross. *A first course in probability.* Pearson Education, 2002

3.4 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,

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

3.5 Gamma distribution

[The text is an amalgam of Kerns and Ross[3] (page 215). I don't put it in double-quotes as a citation because it would look ugly.]

[3] Sheldon Ross. *A first course in probability.* Pearson Education, 2002

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:

$$\Gamma(\alpha) = \int_0^\infty e^{-y} y^{\alpha-1}\, dy \underset{\text{integration by parts}}{=} (\alpha-1)\Gamma(\alpha-1)$$

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.

```
> ## fn refers to the fact that it is a function in R,
> ## it does not mean that this is the gamma function:
> gamma.fn<-function(x){
+         lambda<-1
+         alpha<-1
+         (lambda * exp(1)^(-lambda*x) *
+         (lambda*x)^(alpha-1))/gamma(alpha)
+ }
> x<-seq(0,4,by=.01)
> plot(x,gamma.fn(x),type="l")
```

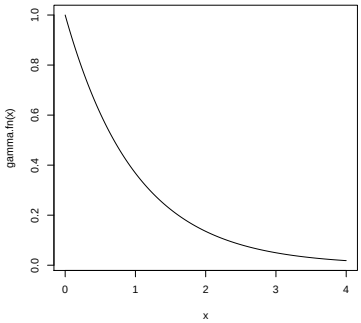


Figure 3.1: The gamma distribution.

.

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

```
> gamma.fn<-function(x){
+         lambda<-1/2
+         alpha<-8/2 ## n=4
+         (lambda * (exp(1)^(-lambda*x)) *
+         (lambda*x)^(alpha-1))/gamma(alpha)
+ }
```

```
> x<-seq(0,100,by=.01)
> plot(x,gamma.fn(x),type="l")
```

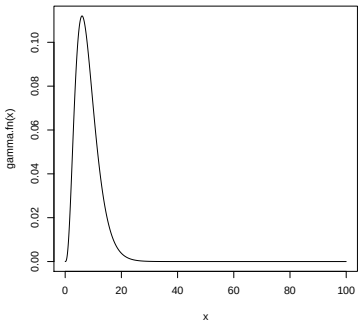


Figure 3.2: The chi-squared distribution.

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

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

3.6 Memoryless property (Poisson, Exponential, Geometric)

A nonnegative random variable is memoryless if

$$P(X > s+t \mid X > t) = P(X > s) \quad \text{for all } s,t \geq 0$$

Two equivalent ways of stating this:

$$\frac{P(X > s+t, X > t)}{P(X > t)} = P(X > s)$$

[just using the definition of conditional probability]
or

$$P(X > s+t) = P(X > s)P(X > t)$$

Recall definition of conditional probability:

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

What memorylessness means is: let $s = 10$ and $t = 30$. Then

$$\frac{P(X > 10+30, X \geq 30)}{P(X \geq 30)} = P(X > 10)$$

or

$$P(X > 10+30) = P(X > 10)P(X \geq 30)$$

It does **not** mean:

$$P(X > 10+30 \mid X \geq 30) = P(X > 40)$$

It's easier to see graphically what this means:

```
> fn<-function(x,lambda){
+         lambda*exp(1)^(-lambda*x)
+ }
> x<-seq(0,1000,by=1)
> plot(x,fn(x,lambda=1/100),type="l")
> abline(v=200,col=3,lty=3)
> abline(v=300,col=1,lty=3)


> x1<-seq(300,1300,by=1)
> plot(x1,fn(x1,lambda=1/100),type="l")
```

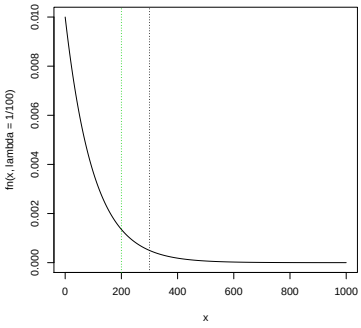


Figure 3.3: The memoryless property of the exponential distribution. The graph after point 300 is an exact copy of the original graph (this is not obvious from the graph, but redoing the graph starting from 300 makes this clear, see figure 3.4 below).

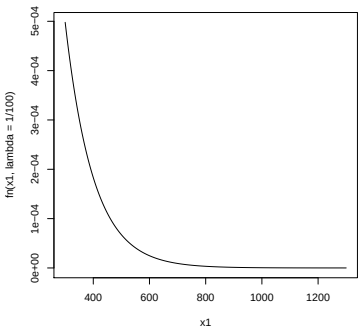


Figure 3.4: Replotting the distribution starting from 300 instead of 0, and extending the x-axis to 1300 instead of 1000 (the number in figure 3.3) gives us an exact copy of original. This is the meaning of the memoryless property of the distribution.

Examples of memorylessness

Suppose we are given that a discrete random variable X has probability function $\theta^{x-1}(1-\theta)$, where $x = 1,2,\ldots$. Show that

$$P(X > t+a \mid X > a) = \frac{P(X > t+a)}{P(X > a)} \tag{3.7}$$

hence establishing the 'absence of memory' property:

$$P(X > t+a \mid X > a) = P(X > t) \tag{3.8}$$

Proof:

First, restate the pdf given so that it satisfies the definition of a geometric distribution. Let $\theta = 1-p$; then the pdf is

$$(1-p)^{x-1}p \tag{3.9}$$

This is clearly a geometric random variable (see p. 155 of Ross). On p. 156, Ross points out that

$$P(X > a) = (1-p)^a \tag{3.10}$$

[Actually Ross points out that $P(X \geq k) = (1-p)^{k-1}$, from which it follows that $P(X \geq k+1) = (1-p)^k$; and since $P(X \geq k+1) = P(X > k)$, we have $P(X > k) = (1-p)^k$.]

Similarly,

$$P(X > t) = (1-p)^t \tag{3.11}$$

and

$$P(X > t+a) = (1-p)^{t+a} \tag{3.12}$$

Now, we plug in the values for the right-hand side in equation 3.7, repeated below:

$$P(X > t+a \mid X > a) = \frac{P(X > t+a)}{P(X > a)} = \frac{(1-p)^{t+a}}{(1-p)^a} = (1-p)^t \qquad (3.13)$$

Thus, since $P(X > t) = (1-p)^t$ (see above), we have proved that

$$P(X > t+a \mid X > a) = P(X > t) \qquad (3.14)$$

This is the definition of memorylessness (equation 5.1 in Ross, p. 210). Therefore, we have proved the memorylessness property.

■

Optional exercise (solutions below): Prove the memorylessness property for Gamma and Exponential distributions

Exponential:

The CDF is:

$$P(a) = 1 - e^{-\lambda a} \qquad (3.15)$$

Therefore:

$$\begin{aligned} P(X > s+t) &= 1 - P(s+t) \\ &= 1 - (1 - e^{-\lambda(s+t)}) \\ &= e^{-\lambda(s+t)} \\ &= e^{-\lambda s} e^{-\lambda t} \\ &= P(X > s) P(X > t) \end{aligned} \qquad (3.16)$$

The above is the definition of memorylessness.

■

Gamma distribution:

The CDF (not sure how this comes about, see Ross) is

$$F(x; \alpha, \beta) = 1 - \sum_{i=0}^{\alpha-1} \frac{1}{i!} (\beta x)^i e^{-\beta x} \qquad (3.17)$$

Therefore,

$$\begin{aligned} P(X > s+t) &= 1 - P(X < s+t) \\ &= 1 - (1 - \sum_{i=0}^{\alpha-1} \tfrac{1}{i!} (\beta(s+t))^i e^{-\beta(s+t)}) \\ &= \sum_{i=0}^{\alpha-1} \tfrac{1}{i!} (\beta(s+t))^i e^{-\beta(s+t)} \end{aligned} \qquad (3.18)$$

3.7 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

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}, \quad 0 < x < 1. \tag{3.19}$$

The mean and variance are

$$E[X] = \frac{a}{a+b} \text{ and } Var(X) = \frac{ab}{(a+b)^2(a+b+1)}. \tag{3.20}$$

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

3.8 t distribution

A random variable X with PDF

$$f_X(x) = \frac{\Gamma[(r+1)/2]}{\sqrt{r\pi}\,\Gamma(r/2)} \left(1 + \frac{x^2}{r}\right)^{-(r+1)/2}, \quad -\infty < x < \infty \tag{3.21}$$

is said to have Student's t distribution with r degrees of freedom, and we write $X \sim \mathtt{t}(\mathtt{df} = r)$. 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.

to-do: much more detail needed.

3.9 Inverse Gamma

to-do

3.10 Class activity

4
Jointly distributed random variables

4.1 Joint distribution functions

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

$$f_{X,Y}(x,y) = \mathbb{P}(X = x, Y = y), \quad \text{for } (x,y) \in S_{X,Y}. \tag{4.1}$$

Every joint PMF satisfies

$$f_{X,Y}(x,y) > 0 \text{ for all } (x,y) \in S_{X,Y}, \tag{4.2}$$

and

$$\sum_{(x,y)\in S_{X,Y}} f_{X,Y}(x,y) = 1. \tag{4.3}$$

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

$$\begin{aligned} f_X(x) &= \mathbb{P}(X = x), & (4.4)\\ &= \sum_{y\in S_Y} \mathbb{P}(X = x, Y = y), & (4.5)\\ &= \sum_{y\in S_Y} f_{X,Y}(x,y). & (4.6) \end{aligned}$$

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

$$f_Y(y) = \sum_{x \in S_X} f_{X,Y}(x,y). \tag{4.7}$$

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.

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 4.1. The joint CDF is typically not used in practice due to its inconvenient form; one can usually get by with the joint PMF alone.

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.

Continuous case

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

$$F(a,b) = P(X \leq a, Y \leq b) \quad -\infty < a,b < \infty \tag{4.8}$$

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

1. The CDF of X:

$$F_X(a) = P(X \leq a) = F_X(a,\infty) \tag{4.9}$$

2. The CDF of Y:

$$F_Y(a) = P(Y \leq b) = F_Y(\infty,b) \tag{4.10}$$

Definition 2. ***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:*

$$P((X,Y) \in C) = \iint\limits_{(x,y)\in C} f(x,y)\,dx\,dy \tag{4.11}$$

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

$$f(x,y) \geq 0 \text{ for all } (x,y) \in S_{X,Y}, \tag{4.12}$$

and

$$\iint\limits_{S_{X,Y}} f(x,y)\,\mathrm{dx}\mathrm{dy} = 1. \tag{4.13}$$

For any sets of real numbers A and B, and if $C = \{(x,y) : x \in A, y \in B\}$, it follows from equation 4.11 that

$$P((X \in A, Y \in B) \in C) = \int_B \int_A f(x,y)\,dx\,dy \tag{4.14}$$

Note that

$$F(a,b) = P(X \in (-\infty,a]), Y \in (-\infty,b])) = \int_{-\infty}^{b}\int_{-\infty}^{a} f(x,y)\,dx\,dy \tag{4.15}$$

Differentiating, we get the joint pdf:

$$f(a,b) = \frac{\partial^2}{\partial a \partial b} F(a,b) \tag{4.16}$$

One way to understand the joint PDF:

$$P(a < X < a+da, b < Y < b+db) = \int_b^{d+db}\int_a^{a+da} f(x,y)\,dx\,dy \approx f(a,b)dadb \tag{4.17}$$

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

Marginal probability distribution functions If X and Y are jointly continuous, they are individually continuous, and their PDFs are:

$$\begin{aligned} P(X \in A) =& P(X \in A, Y \in (-\infty,\infty)) \\ =& \int_A \int_{-\infty}^{\infty} f(x,y)\,dy\,dx \\ =& \int_A f_X(x)\,dx \end{aligned} \tag{4.18}$$

where

$$f_X(x) = \int_{-\infty}^{\infty} f(x,y)\,dy \tag{4.19}$$

Similarly:

$$f_Y(y) = \int_{-\infty}^{\infty} f(x,y)\,dx \tag{4.20}$$

4.2 Independent random variables

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

$$P(X \in A, Y \in B) = P(X \in A)P(Y \in B) \tag{4.21}$$

In the jointly continuous case:

$$f(x,y) = f_X(x) f_Y(y) \quad \text{for all } x,y \tag{4.22}$$

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.

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 ($f_X(x) f_Y(y)$).

4.3 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:

$$\begin{aligned} F_{X+Y}(a) =& P(X+Y \le a) \\ &= \iint\limits_{x+y\le a} f_{XY}(x,y)\,dx\,dy \\ &= \iint\limits_{x+y\le a} f_X(x) f_Y(y)\,dx\,dy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{a-y} f_X(x) f_Y(y)\,dx\,dy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{a-y} f_X(x)\,dx f_Y(y)\,dy \\ &= \int_{-\infty}^{\infty} F_X(a-y) f_Y(y)\,dy \end{aligned} \tag{4.23}$$

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}:

$$\begin{aligned} f_{X+Y} &= \frac{d}{dx}\int_{-\infty}^{\infty} F_X(a-y) f_Y(y)\, dy \\ &= \int_{-\infty}^{\infty} \frac{d}{dx} F_X(a-y) f_Y(y)\, dy \\ &= \int_{-\infty}^{\infty} f_X(a-y) f_Y(y)\, dy \end{aligned} \tag{4.24}$$

4.4 Conditional distributions

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

$$\mathbb{P}(B \mid A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(A)}, \quad \text{if } \mathbb{P}(A) > 0. \tag{4.25}$$

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

$$\begin{aligned} p_{X|Y}(x \mid y) &= P(X = x \mid Y = y) \\ &= \frac{P(X = x, Y = y)}{P(Y = y)} \\ &= \frac{p(x,y)}{p_Y(y)} \end{aligned} \tag{4.26}$$

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:

$$\begin{aligned} F_{X|Y} &= P(X \leq x \mid Y = y) \\ &= \sum_{a \leq x} p_{X|Y}(a \mid y) \end{aligned} \tag{4.27}$$

If X and Y are independent then

$$p_{X|Y}(x \mid y) = P(X = x) = p_X(x) \tag{4.28}$$

See the examples starting p. 264 of Ross.

An important thing to understand is the phrasing of the question (e.g., in P-Ass3): "Find the conditional distribution of X given all the possible values of Y".

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

$$f_{X|Y}(x \mid y) = \frac{f(x,y)}{f_Y(y)} \tag{4.29}$$

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

$$\begin{aligned} f_{X|Y}(x \mid y)\,dx &= \frac{f(x,y)}{f_Y(y)}\frac{dxdy}{dy} \\ &= \frac{f(x,y)dxdy}{f_Y(y)dy} \\ &= \frac{P(x < X < d + dx, y < Y < y + dy)}{y < P < y + dy} \end{aligned} \tag{4.30}$$

4.5 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:

$$\mathbb{E}\,g(X,Y) = \sum_{(x,y)\in S_{X,Y}}\sum g(x,y)\,f_{X,Y}(x,y). \tag{4.31}$$

Continuous case:

$$\mathbb{E}\,g(X,Y) = \iint_{S_{X,Y}} g(x,y)\,f_{X,Y}(x,y)\,\mathrm{d}x\,\mathrm{d}y, \tag{4.32}$$

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 3. *The **covariance** of X and Y is*

$$Cov(X,Y) = \mathbb{E}(X - \mathbb{E}X)(Y - \mathbb{E}Y). \tag{4.33}$$

Shortcut formula for covariance:

$$\mathrm{Cov}(X,Y) = \mathbb{E}(XY) - (\mathbb{E}X)(\mathbb{E}Y). \tag{4.34}$$

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

$$\text{Corr}(X,Y) = \frac{\text{Cov}(X,Y)}{\sigma_X \sigma_Y}. \tag{4.35}$$

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 $-1 \leq \rho_{X,Y} \leq 1$.

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

Discrete example: to-do

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

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

4.6 Conditional expectation

Recall that

$$f_{X|Y}(x \mid y) = P(X = x \mid Y = y) = \frac{p_{X,Y}(x,y)}{p_Y(y)} \tag{4.36}$$

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

It follows that

$$\begin{aligned} E[X \mid Y = y] &= \sum_x x P(X = x \mid Y = y) \\ &= \sum_x x p_{X|Y}(x \mid y) \end{aligned} \tag{4.37}$$

$E[X \mid Y]$ 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:

$$E\left[\sum_{i=1}^{n} X_i \mid Y = y\right] = \sum_{i=1}^{n} E[X_i \mid Y = y] \tag{4.38}$$

Proposition 5. ***Expectation of the conditional expectation***

$$E[X] = E[E[X \mid Y]] \tag{4.39}$$

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

$$E[X] = \sum_y E[X \mid Y = y] P(Y = y) \tag{4.40}$$

4.7 Multivariate normal distributions

Recall that in the univariate case:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e\{-\frac{(\frac{(x-\mu)}{\sigma})^2}{2}\} \quad -\infty < x < \infty \tag{4.41}$$

We can write the power of the exponential as:

$$(\frac{(x-\mu)}{\sigma})^2 = (x-\mu)(x-\mu)(\sigma^2)^{-1} = (x-\mu)(\sigma^2)^{-1}(x-\mu) = Q \quad (4.42)$$

Generalizing this to the multivariate case:

$$Q = (x-\mu)'\Sigma^{-1}(x-\mu) \quad (4.43)$$

So, for multivariate case:

$$f(x) = \frac{1}{\sqrt{2\pi det\Sigma}} e\{-Q/2\} \quad -\infty < x_i < \infty, i = 1,\ldots,n \quad (4.44)$$

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")
```

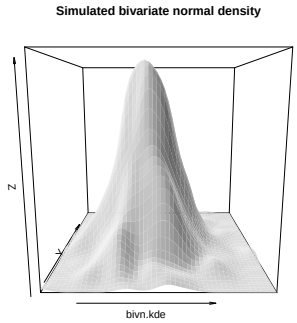


Figure 4.1: Visualization of two uncorrelated random variables.

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")
```

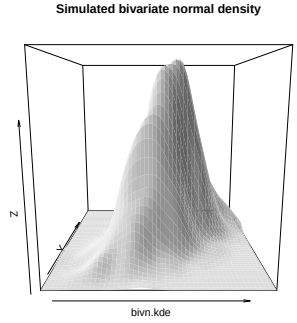


Figure 4.2: Visualization of two correlated random variables.

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(.5)
+ }
```

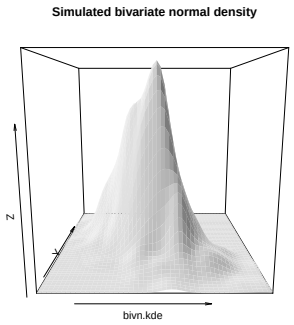


Figure 4.3: Visualization of two negatively correlated random variables.

5
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.

5.1 Discrete case

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

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = f(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n; \theta) \quad (5.1)$$

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**.

5.2 Continuous case

Here, f is the joint probability **density**, the rest is the same as above.

Definition 4. *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*

$$L(\theta) = f(x_1, x_2, \ldots, x_n; \theta) \quad (5.2)$$

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.

5.3 *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.

$$L(\sigma) = \prod f(x_i;\sigma) = \frac{1}{(2\sigma)^n} exp(-\sum \frac{|\,x_i\,|}{\sigma}) \tag{5.3}$$

Let ℓ be log likelihood. Then:

$$\ell(x;\sigma) = \sum \left[-\log 2 - \log \sigma - \frac{|\,x_i\,|}{\sigma} \right] \tag{5.4}$$

Differentiating and equating to zero to find maximum:

$$\ell'(\sigma) = \sum \left[-\frac{1}{\sigma} + \frac{|\,x_i\,|}{\sigma^2} \right] = -\frac{n}{\sigma} + \frac{|\,x_i\,|}{\sigma^2} = 0 \tag{5.5}$$

Rearranging the above, the MLE for σ is:

$$\hat{\sigma} = \frac{\sum |\,x_i\,|}{n} \tag{5.6}$$

Example 2: Exponential

$$f(x;\lambda) = \lambda exp(-\lambda x) \tag{5.7}$$

Log likelihood:

$$\ell = n \log \lambda - \sum \lambda x_i \tag{5.8}$$

Differentiating:

$$\ell'(\lambda) = \frac{n}{\lambda} - \sum x_i = 0 \tag{5.9}$$

$$\frac{n}{\lambda} = \sum x_i \tag{5.10}$$

I.e.,

$$\frac{1}{\hat{\lambda}} = \frac{\sum x_i}{n} \tag{5.11}$$

Example 3: Poisson

$$L(\mu;x) = \prod \frac{\exp^{-\mu} \mu^{x_i}}{x_i!} \tag{5.12}$$

$$= \exp^{-\mu} \mu^{\sum x_i} \frac{1}{\prod x_i!} \tag{5.13}$$

Log lik:

$$\ell(\mu;x) = -n\mu + \sum x_i \log \mu - \sum \log y! \tag{5.14}$$

Differentiating:

$$\ell'(\mu) = -n + \frac{\sum x_i}{\mu} = 0 \tag{5.15}$$

Therefore:

$$\hat{\lambda} = \frac{\sum x_i}{n} \tag{5.16}$$

Example 4: Binomial

$$L(\theta) = \binom{n}{x} \theta^x (1-\theta)^{n-x} \tag{5.17}$$

Log lik:

$$\ell(\theta) = \log \binom{n}{x} + x \log \theta + (n-x) \log(1-\theta) \tag{5.18}$$

Differentiating:

$$\ell'(\theta) = \frac{x}{\theta} - \frac{n-x}{1-\theta} = 0 \tag{5.19}$$

Thus:

$$\hat{\theta} = \frac{x}{n} \tag{5.20}$$

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.

$$L(\mu; \sigma^2) = \prod N(x_i; \mu, \sigma) \tag{5.21}$$

$$= \left(\tfrac{1}{2\pi\sigma^2}\right)^{n/2} \exp\left(-\tfrac{1}{2\sigma^2} \sum (x_i - \mu)^2\right) \tag{5.22}$$

$$\tag{5.23}$$

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

$$\hat{\mu} = \frac{1}{n} \sum x_i = \bar{x} \tag{5.24}$$

and

$$\hat{\sigma}^2 = \frac{1}{n} \sum (x_i - \bar{x})^2 \tag{5.25}$$

Example 6: Geometric

$$f(x; p) = (1-p)^{x-1} p \tag{5.26}$$

$$L(p) = p^n (1-p)^{\sum x - n} \tag{5.27}$$

Log lik:

$$\ell(p) = n\log p + (\sum x - n)\log(1-p) \tag{5.28}$$

Differentiating:

$$\ell'(p)\frac{n}{p} - \frac{\sum x - n}{1-p} = 0 \tag{5.29}$$

$$\hat{p} = \frac{1}{\bar{x}} \tag{5.30}$$

5.4 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))
```

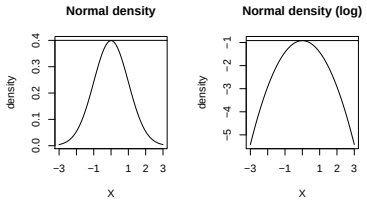


Figure 5.1: Maximum likelihood and log likelihood.

5.5 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.

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")
```

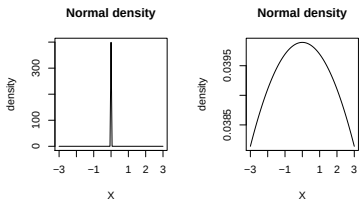


Figure 5.2: How variance relates to the second derivative.

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 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.

$$L(p) = \binom{n}{x} p^x (1-p)^{n-x} \tag{5.31}$$

Log lik:

$$\ell(p) = \log \binom{n}{x} + x \log p + (n-x) \log(1-p) \tag{5.32}$$

Differentiating:

$$\ell'(p) = \frac{x}{p} - \frac{n-x}{1-p} = 0 \tag{5.33}$$

Taking the second partial derivative with respect to p:

$$\ell''(p) = -\frac{x}{p^2} - \frac{n-x}{(1-p)^2} \tag{5.34}$$

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

$$E(\ell''(p)) = E(-\frac{x}{p^2} - \frac{n-x}{(1-p)^2}) \tag{5.35}$$

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

$$E(\ell''(p)) = E(-\frac{x}{p^2} - \frac{n-x}{(1-p)^2}) = -\frac{np}{p^2} - \frac{n-np}{(1-p)^2} \underset{\substack{\uparrow \\ exercise}}{=} -\frac{n}{p(1-p)} \quad (5.36)$$

Next, we negate and invert the expectation:

$$-\frac{1}{E(\ell''(\theta))} = \frac{p(1-p)}{n} \quad (5.37)$$

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

$$-\frac{1}{E(\ell''(\theta))} = \frac{\hat{p}(1-\hat{p})}{n} = \frac{1}{I(p)} \quad (5.38)$$

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

If we take the square root of the inverse Information Matrix

$$\sqrt{\frac{1}{I(p)}} = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \quad (5.39)$$

we have the **estimated standard error**.

Another example using the normal distribution:

Example 2: Normal distribution This example is based on Khuri[1] (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.

[1] André I Khuri. *Advanced calculus with applications in statistics*, volume 486. Wiley, 2003

$$L(x \mid \mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp[-\frac{1}{2\sigma^2} \sum (x-\mu)^2] \quad (5.40)$$

Taking log likelihood:

$$\ell = -\frac{n}{2} \log \frac{1}{(2\pi\sigma^2)} - \frac{1}{2\sigma^2} \sum (x-\mu)^2 \quad (5.41)$$

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

$$\frac{\partial \ell}{\partial \mu} = \frac{1}{\sigma^2} \sum (x-\mu) = 0 \Rightarrow n(\bar{x}-\mu) = 0 \quad (5.42)$$

$$\frac{\partial \ell}{\partial \sigma^2} = \frac{1}{2\sigma^4} \sum (x-\mu)^2 - \frac{n}{2\sigma^2} = 0 \Rightarrow \sum (x-\mu)^2 - n\sigma^2 = 0 \quad (5.43)$$

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[2] 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.)[3]

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

$$H = \begin{pmatrix} \frac{\partial^2 \ell}{\partial \mu^2} & \frac{\partial^2 \ell}{\partial \mu \partial \sigma^2} \\ \frac{\partial^2 \ell}{\partial \mu \partial \sigma^2} & \frac{\partial^2 \ell}{\partial \sigma^4} \end{pmatrix} \tag{5.44}$$

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:

$$\frac{\partial^2 \ell}{\partial \mu^2} = -\frac{n}{\hat{\sigma}^2} \tag{5.45}$$

$$\frac{\partial^2 \ell}{\partial \mu \partial \sigma^2} = -\frac{1}{\hat{\sigma}^2} \sum (x - \hat{\mu}) = 0 \tag{5.46}$$

$$\frac{\partial^2 \ell}{\partial \sigma^4} = -\frac{n}{2\hat{\sigma}^2} \tag{5.47}$$

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 ($\theta = (\mu, \sigma^2)$):

$$I(\theta) = \begin{pmatrix} \frac{n}{\sigma^2} & 0 \\ 0 & \frac{n}{2\sigma^4} \end{pmatrix} \tag{5.48}$$

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

$$\frac{1}{I(\theta)} = \begin{pmatrix} \frac{\hat{\sigma}^2}{n} & 0 \\ 0 & \frac{2\hat{\sigma}^4}{n} \end{pmatrix} \tag{5.49}$$

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!

[2] 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.

[3] The inverse of a matrix: $\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$

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.

5.6 MLE using R

One-parameter case **Example 1: Estimating λ in the Box-Cox transform**

Let's assume that there exists a λ such that

$$f_\lambda(y_i) = x_i^T\beta + \varepsilon_i \quad \varepsilon_i \sim N(0,\sigma^2) \tag{5.50}$$

We use maximum likelihood estimation to estimate λ. Note that $L(\beta_\lambda, \sigma^2_\lambda, \lambda; y) \propto$

$$(\frac{1}{\sigma})^n \exp[-\frac{1}{2\sigma^2}\sum[f_\lambda(y_i) - x_i^T\beta]^2][\underbrace{\prod f'_\lambda(y_i)}_{\text{Jacobian}}] \tag{5.51}$$

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

$$L(\hat{\beta}_\lambda, \hat{\sigma}^2_\lambda, \lambda; y) = S_\lambda^{-n/2}\prod f'_\lambda(y_i) \tag{5.52}$$

Taking logs:

$$\ell = c - \frac{n}{2}\log S_\lambda + \sum \log f'_\lambda(y_i) \tag{5.53}$$

One interesting case is the **Box-Cox family**.The relevant paper is by Box and Cox.[4] (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.[5])

[4] George E.P. Box and David R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 211–252, 1964

[5] George E.P. Box. *An accidental statistician: The life and memories of George EP Box*. Wiley, 2013

$$f_\lambda(y) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \lambda \neq 0 \\ \log y & \lambda = 0 \end{cases} \tag{5.54}$$

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:

$$\ell = c - \frac{n}{2}\log \underbrace{S_\lambda}_{\text{Residual sum of squares}} + (\lambda - 1)\sum \log(y_i) \tag{5.55}$$

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)))

$minimum
[1] -1.9999

$objective
[1] -69.311
```

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] 0.4
```

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] 0.4

$objective
[1] 6.7301
```

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] 676834

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

[1] 284515
```

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] 533.64
```

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] 502.633  50.204
```

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] 5.0204 3.5502
```

These values match our standard calculations:

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

```
[1] 5.0456
```

5.6.1 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(.1))
```

```
[1] 9.8425
```

```
> opt.vals.default<-optim(theta<-c(.5),
+                         negllbinom2,
+       method="Brent",
+       hessian=TRUE,lower=0,upper=1)
> ## SE:
> sqrt(solve(opt.vals.default$hessian))
```

```
        [,1]
[1,] 0.15492
```

The estimated SE matches our analytical result earlier:

$$\sqrt{\frac{1}{I(p)}} = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \tag{5.56}$$

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

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

```
[1] 0.15492
```

6
Basics of bayesian statistics

Recall Bayes rule:

Theorem 5. ***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*

$$\mathbb{P}(B_k|A) = \frac{\mathbb{P}(B_k)\mathbb{P}(A|B_k)}{\sum_{i=1}^{n} \mathbb{P}(B_i)\mathbb{P}(A|B_i)}, \quad k = 1,2,\ldots,n. \tag{6.1}$$

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

$$p(A \mid B) = \frac{p(B \mid A)p(A)}{p(B)} \tag{6.2}$$

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

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

$$f(\theta \mid \text{data}) = \frac{f(\text{data} \mid \theta)f(\theta)}{f(data)} \tag{6.3}$$

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.

$$f(y) = \int f(x,\theta)\,d\theta = \int \underset{\substack{\uparrow \\ likelihood}}{f(y \mid \theta)} \underset{\substack{\uparrow \\ prior}}{f(\theta)}\,d\theta \tag{6.4}$$

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

$$f(y) = \sum_{i=1}^{n} f(y \mid \theta_i)P(\theta = \theta_i) \tag{6.5}$$

Without the normalizing constant, we have the relationship:

$$f(\theta \mid \text{data}) \propto f(\text{data} \mid \theta)f(\theta) \tag{6.6}$$

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:

$$\text{Posterior} \propto \text{Likelihood} \times \text{Prior} \tag{6.7}$$

Our central goal is going to be to derive the posterior distribution and then summarize its properties (mean, median, 95% credible interval, etc.). 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.

"To a Bayesian, the best information one can ever have about θ is to know the posterior density." p. 31 of Christensen et al's book.

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.

Example 1: Proportions This is a contrived example, just meant to provide us with an entry point into bayesian data analysis. Suppose that an aphasic patient answered 46 out of 100 questions correctly in a particular 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] 0.057958
```

Note that

$$P(\text{data} \mid \theta) \propto \theta^{46}(1-\theta)^{54} \tag{6.8}$$

So, to get the posterior, we just need to work out a prior distribution $f(\theta)$.

$$f(\theta \mid \text{data}) \propto f(\text{data} \mid \theta) \underbrace{f(\theta)}_{prior} \tag{6.9}$$

For the prior, we need a distribution that can represent our uncertainty about p. 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[1]

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

where

[1] 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

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}, \quad 0 < x < 1. \tag{6.10}$$

Here, x refers to the probability p.

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

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

$$E[X] = \frac{a}{a+b} \text{ and } Var(X) = \frac{ab}{(a+b)^2(a+b+1)}. \tag{6.11}$$

The beta distribution's parameters a and b can be interpreted as (our beliefs about) prior successes and failures, and are called **hyper-parameters**. 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=2,shape2=2), 0,1,
+       main = "Beta density",
+               ylab="density",xlab="X",ylim=c(0,3))
> text(.5,1.1,"a=2,b=2")
> plot(function(x)
+   dbeta(x,shape1=3,shape2=3),0,1,add=T)
> text(.5,1.6,"a=3,b=3")
> plot(function(x)
+   dbeta(x,shape1=6,shape2=6),0,1,add=T)
> text(.5,2.6,"a=6,b=6")
```

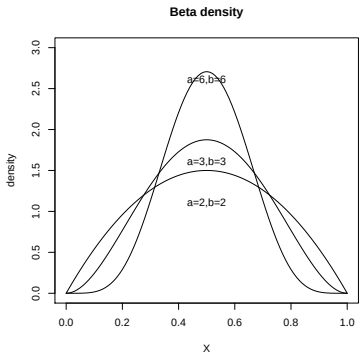


Figure 6.1: Examples of the beta distribution with different parameter values.

As the figure shows, as the a,b values are increased, the shape begins to resemble the normal distribution.

If we don't have much prior information, we could use a=b=2; 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 smaller 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 (except perhaps the first) 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:

$$f(\theta \mid \text{data}) \propto f(\text{data} \mid \theta) f(\theta) \tag{6.12}$$

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

$$f(\theta \mid \text{data}) \propto [p^{46}(1-p)^{54}][p^{2-1}(1-p)^{2-1}] = p^{47}(1-p)^{55} \tag{6.13}$$

$$f(\theta \mid \text{data}) \propto [p^{46}(1-p)^{54}][p^{3-1}(1-p)^{3-1}] = p^{48}(1-p)^{56} \tag{6.14}$$

$$f(\theta \mid \text{data}) \propto [p^{46}(1-p)^{54}][p^{6-1}(1-p)^{6-1}] = p^{51}(1-p)^{59} \tag{6.15}$$

$$f(\theta \mid \text{data}) \propto [p^{46}(1-p)^{54}][p^{21-1}(1-p)^{21-1}] = p^{66}(1-p)^{74} \tag{6.16}$$

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")
```

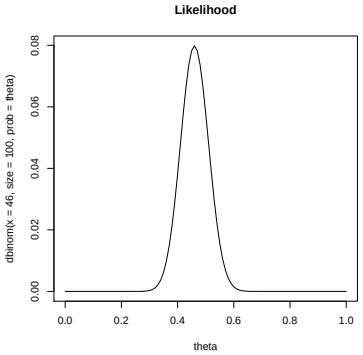
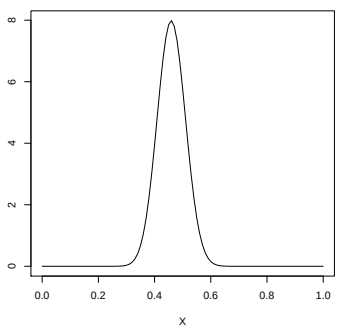


Figure 6.2: Binomial likelihood function.

Homework 2

Plot the priors, likelihoods, and posterior distributions in the four cases above.



Example 2: Proportions This example is taken from a Copenhagen course on bayesian data analysis that I attended in 2012.

For a single Bernoulli trial, the **likelihood** for possible parameters value θ_j (which we, for simplicity, fix at four values, 0.2, 0.4, 0.6, 0.8.) is:[2]

$$p(y \mid \theta_j) = \theta_j^y (1-\theta_j)^{1-y} \tag{6.17}$$

[2] See page 33 for the definition of the binomial distribution. The Bernoulli distribution is the binomial with n=1.

```
> y<-1
> n<-1
> thetas<-seq(0.2,0.8,by=0.2)
> likelihoods<-rep(NA,4)
> for(i in 1:length(thetas)){
+   likelihoods[i]<-dbinom(y,n,thetas[i])
+ }
```

Note that these do not sum to 1:

```
> sum(likelihoods)
```

```
[1] 2
```

Question: How can we make them sum to 1? I.e., how can we make the likelihood a proper pmf? Think about this before reading further.

Let the **prior** distribution of the parameters be $p(\theta_j)$; let this be a uniform distribution over the possible values of θ_j.

```
> (priors<-rep(0.25,4))
```

```
[1] 0.25 0.25 0.25 0.25
```

The prior is a proper probability distribution.

For any outcome y (which could be 1 or 0—this is a single trial), the posterior probability of success (a 1) is related to the prior and likelihood by the relation:

$$p(\theta_j \mid y) \propto p(\theta_j)\theta_j^y (1-\theta_j)^{1-y} \tag{6.18}$$

To get the posterior to be a proper probability distribution, you have to make sure that the RHS sums to 1.

```
> liks.times.priors<-likelihoods * priors
> ## normalizing constant:
> sum.lik.priors<-sum(liks.times.priors)
> posteriors<- liks.times.priors/sum.lik.priors
```

Note that the posterior distribution sums to 1, because we **normalized** it.

Now suppose our sample size was 20, and the number of successes 15. What does the posterior distribution look like now?

```
> n<-20
> y<-15
> priors<-rep(0.25,4)
> likelihoods<-rep(NA,4)
> for(i in 1:length(thetas)){
+   likelihoods[i]<-dbinom(y,n,thetas[i])
+ }
> liks.priors<-likelihoods * priors
> sum.lik.priors<-sum(liks.priors)
> (posteriors<- liks.priors/sum.lik.priors)

[1] 6.6456e-07 5.1676e-03 2.9799e-01 6.9684e-01
```

Now suppose that we had a non-zero prior probability to extreme values (0,1) to θ. The prior is now defined over six values, not four, so the probability distribution on the priors changes accordingly to 1/6 for each value of θ.

Given the above situation of n=20, y=15, what will change in the posterior distribution compared to what we just computed:

```
> posteriors

[1] 6.6456e-07 5.1676e-03 2.9799e-01 6.9684e-01
```

Let's find out:

```
> thetas<-seq(0,1,by=0.2)
> priors<-rep(1/6,6)
> y<-15
> n<-20
> likelihoods<-rep(NA,6)
> for(i in 1:length(thetas)){
+   likelihoods[i]<-dbinom(y,n,thetas[i])
+ }
> liks.priors<-likelihoods * priors
> sum.lik.priors<-sum(liks.priors)
> (posteriors<- liks.priors/sum.lik.priors)

[1] 0.0000e+00 6.6456e-07 5.1676e-03 2.9799e-01 6.9684e-01 0.0000e+00
```

How would the posteriors change if we had only one trial (a Bernoulli trial)? Let's find out:

```
> thetas<-seq(0,1,by=0.2)
> priors<-rep(1/6,6)
> y<-1
> n<-1
```

```
> j<-6 ## no. of thetas
> likelihoods<-rep(NA,6)
> for(i in 1:length(thetas)){
+   likelihoods[i]<-dbinom(y,n,thetas[i])
+ }
> liks.priors<-likelihoods * priors
> sum.lik.priors<-sum(liks.priors)
> posteriors<- liks.priors/sum.lik.priors
```

We have been using discrete prior distributions so far. We might want to use a continuous prior distribution if we have prior knowledge that, say, the true value lies between 0.2 and 0.6, with mean 0.4. If our prior should have mean 0.4 and sd 0.1, we can figure out what the corresponding parameters of the beta distribution should be. (Look up the mean and variance of the beta distribution, and solve for the parameters; see page 54 for information on the beta distribution.) You should be able to prove analytically that a= 9.2, b=13.8.

Let's plot the prior:

```
> x<-seq(0,1,length=100)
> plot(x,dbeta(x,shape1=9.2,shape2=13.8),type="l")
```

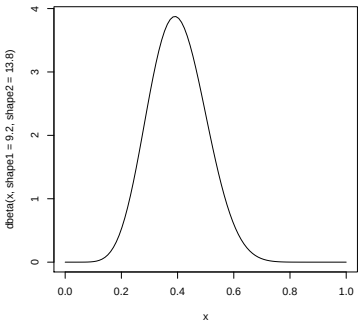


Figure 6.4: The prior in terms of the beta distribution.

Then the likelihood: Likelihood is a function of the parameters $\boldsymbol{\theta}$.

```
> thetas<-seq(0,1,length=100)
> probs<-rep(NA,100)
> for(i in 1:100){
+ probs[i]<-dbinom(15,20,thetas[i])
+ }
> plot(thetas,probs,main="Likelihood of y|theta_j",type="l")
```

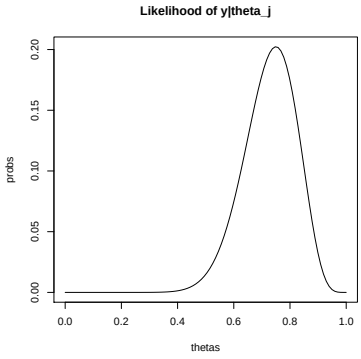


Figure 6.5: Likelihood.

This likelihood can equally well be presented as a beta distribution because the beta distribution's parameters a and b can be interpreted as prior successes and failures.

```
> x<-seq(0,1,length=100)
> plot(x,dbeta(x,shape1=15,shape2=5),type="l")
```

Since we multiply the beta distribution representing the prior and the beta distribution representing the likelihood:

$$Beta(9.2,13.8) * Beta(15,5) = Beta(a = 9.2+15, b = 13.8+5) \quad (6.19)$$

```
> thetas<-seq(0,1,length=100)
> a.star<-9.2+15
> b.star<-13.8+5
> plot(thetas,dbeta(thetas,shape1=a.star,shape2=b.star),
+      type="l")
```

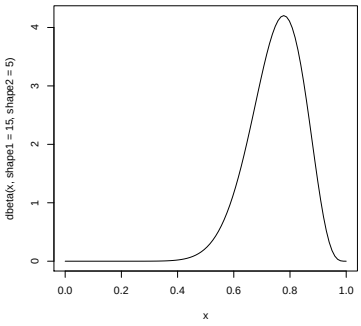


Figure 6.6: Likelihood in terms of the beta.

We can also plot all three (prior, likelihood, posterior) in one figure:

```
> par(mfrow=c(3,1))
> ## prior
> plot(thetas,dbeta(x,shape1=9.2,shape2=13.8),type="l",
+      main="Prior")
> ## lik
> probs<-rep(NA,100)
> for(i in 1:100){
+ probs[i]<-dbinom(15,20,thetas[i])
+ }
> plot(thetas,probs,main="Likelihood of y|theta_j",type="l")
> ## post
> x<-seq(0,1,length=100)
> a.star<-9.2+15
> b.star<-13.8+5
> plot(x,dbeta(x,shape1=a.star,shape2=b.star),type="l",
+      main="Posterior")
```

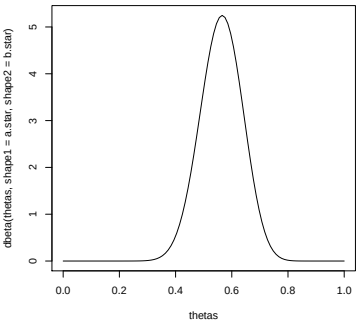


Figure 6.7: Example of posterior distribution.

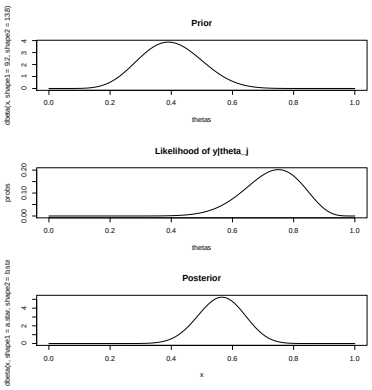


Figure 6.8: The prior, likelihood and posterior distributions in example 2.

Homework 3

Write a generic function that, for a given set of values for the prior (given mean's and sd's), and given the data (number of successes and failures), plots the appropriate posterior (alongside the priors and likelihood).

Homework 4

(This exercise is from the Copenhagen course of 2012. I quote it verbatim).

"The French mathematician Pierre-Simon Laplace (1749-1827) was the first person to show definitively that the proportion of female births in the French population was less then 0.5, in the late 18th century, using a Bayesian analysis based on a uniform prior distribution[3]). Suppose you were doing a similar analysis but you had more definite prior beliefs about the ratio of male to female births. In particular, if θ represents the proportion of female births in a given population, you are willing to place a Beta(100,100) prior distribution on θ.

1. Show that this means you are more than 95% sure that θ is between 0.4 and 0.6, although you are ambivalent as to whether it is greater or less than 0.5.

2. Now you observe that out of a random sample of 1,000 births, 511 are boys. What is your posterior probability that $\theta > 0.5$?"

Homework 5

[3]

[This problem is taken from Lunn et al.[4], their example 3.1.1.] Suppose that 1 in 1000 people in a population are expected to get HIV. Suppose a test is administered on a suspected HIV case, where the test has a true positive rate (the proportion of positives that are actually HIV positive) of 95% and true negative rate (the proportion of negatives are actually HIV negative) 98%. Use Bayes theorem to find out the probability that a patient testing positive actually has HIV.

[4] David Lunn, Chris Jackson, David J Spiegelhalter, Nicky Best, and Andrew Thomas. *The BUGS book: A practical introduction to Bayesian analysis*, volume 98. CRC Press, 2012

6.1 Class activities

6.1.1 The posterior is the weighted mean of the prior mean and the MLE

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:

$$f(x \mid \theta) = \frac{\exp(-\theta)\theta^x}{x!} \tag{6.20}$$

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.

Task 1 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 $\frac{a}{b^2}$. Hint: Since we are given values for the mean and variance, we can solve for a,b, which gives us the gamma density.

Task 2 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.

Example: 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.

Given that

$$\text{Posterior} \propto \text{Prior Likelihood} \tag{6.21}$$

and given that the likelihood is:

$$\begin{aligned} L(\mathbf{x} \mid \theta) &= \prod_{i=1}^{n} \frac{\exp(-\theta)\theta^{x_i}}{x_i!} \\ &= \frac{\exp(-n\theta)\theta^{\Sigma_i^n x_i}}{\prod_{i=1}^{n} x_i!} \end{aligned} \tag{6.22}$$

we can compute the posterior as follows:

$$\text{Posterior} = [\frac{\exp(-n\theta)\theta^{\Sigma_i^n x_i}}{\prod_{i=1}^{n} x_i!}][\frac{b^a \theta^{a-1} \exp(-b\theta)}{\Gamma(a)}] \tag{6.23}$$

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

$$\begin{aligned} \text{Posterior} &\propto \exp(-n\theta)\theta^{\Sigma_i^n x_i} \theta^{a-1} \exp(-b\theta) \\ &= \theta^{a-1+\Sigma_i^n x_i} \exp(-\theta(b+n)) \end{aligned} \tag{6.24}$$

Figure out the parameters of the posterior distribution, and show that it will be a gamma distribution.

Hint: 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*.

Task 3 Given 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

$$\begin{aligned} Gamma(a^* = a + \sum_i^n x_i, b^* = b + n) &= Gamma(10000/225 + 422, 4 + 100/225) \\ &= Gamma(466.44, 4.44) \end{aligned} \tag{6.25}$$

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 $\frac{a*}{b*^2} = 466.44/4.44^2 = 23.61$.

Task 4 Verify the above result by fitting a model in JAGS and running it.

Hint: the model will look like this:

```
model
   {
for(i in 1:4){
```

```
  y[i] ~ dpois(theta)
}
  ##prior
  ## gamma params derived from given information:
  theta ~ SPECIFY PRIOR DISTRIBUTION
}
```

Task 5 Plot the prior, likelihood, and posterior associated with the above model fit.

6.1.2 The posterior as 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:

$$\frac{a*}{b*} = \frac{a + \sum x_i}{n + b} \tag{6.26}$$

This can be rewritten as

$$\frac{a*}{b*} = \frac{a + n\bar{x}}{n + b} \tag{6.27}$$

Dividing both the numerator and denominator by b:

$$\frac{a*}{b*} = \frac{(a + n\bar{x})/b}{(n + b)/b} = \frac{a/b + n\bar{x}/b}{1 + n/b} \tag{6.28}$$

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

$$\frac{a/b + n\bar{x}/b}{1 + n/b} = \frac{m + \frac{n}{b}\bar{x}}{1 + \frac{n}{b}} \tag{6.29}$$

We can rewrite this as:

$$\frac{m + \frac{n}{b}\bar{x}}{1 + \frac{n}{b}} = \frac{m \times 1}{1 + \frac{n}{b}} + \frac{\frac{n}{b}\bar{x}}{1 + \frac{n}{b}} \tag{6.30}$$

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

$$m\frac{w_1}{w_1 + w_2} + \bar{x}\frac{w_2}{w_1 + w_2} \tag{6.31}$$

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.

This makes sense: as sample size increases, the likelihood will dominate in determining the posterior mean.

Regarding variance, since the variance of the posterior is:

$$\frac{a*}{b*^2} = \frac{(a + n\bar{x})}{(n + b)^2} \tag{6.32}$$

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

6.1.3 Problem 4

We solve this problem using the two methods suggested.

Using all the data together If we have additional data from two weeks, with a count of 200, $\sum x_i = 422 + 200 = 622$ and $n = 6$ (not 5, because it is two weeks' data).[5]

This means that the posterior would be a Gamma distribution with parameters: $a^{**} = a + \sum_i^6 x_i = 10000/225 + 622 = 666.44$ and $b^{**} = b + 6 = 100/225 + 6 = 6.44$. In other words, the mean of the posterior is 103.41 and the variance is 16.05.

[5] One can also demonstrate this by multiplying the likelihood of the five data points; for the two weeks' measurement, the likelihood would be proportional to $\exp(-2\theta)(2\theta)^x$. Given that $n = 4$ for the original data, this likelihood for the new data has the effect that the likelihood ends up being proportional to $\exp(-\theta(n+2))(\theta)^{\Sigma x}$.

Verification using JAGS

```
> dat2<-list(y=c(115,97,79,131,200))
> ## model specification:
> cat("
+ model
+    {
+ for(i in 1:4){
+   y[i] ~ dpois(theta)
+ }
+   y[5] ~ dpois(2*theta)
+
+   ##prior
+   ## gamma params derived from given info:
+   theta ~ dgamma(10000/225,100/225)
+ }",
+     file="poisexercise2.jag" )
> ## specify variables to track
> ## the posterior distribution of:
> track.variables<-c("theta")
> ## load rjags library:
> library(rjags,quietly=T)
> ## define model:
> poisex2.mod <- jags.model(
+     data = dat2,
+     file = "poisexercise2.jag",
+     n.chains = 4,
+     n.adapt =2000 ,quiet=T)
> ## run model:
> poisex2.res <- coda.samples( poisex2.mod,
```

```
+                 var = track.variables,
+                 n.iter = 100000,
+                 thin = 50 )
```

JAGS returns the mean as 103.46, and the variance as 4. This is quite close to the analytically computed values.

6.1.4 Using the posterior as our new prior

The posterior distribution given the old data is $Gamma(a = 466.44, b = 4.44)$. The new data is a count of 200 over two weeks, therefore the likelihood is proportional to:

$\exp(-2\theta)(2\theta)^{200}$

Multiplying the prior (the above posterior) with the likelihood, we get:

$$\begin{aligned}[\theta^{466.44-1}\exp(-4.44\theta)][\exp(-2\theta)(2\theta)^{200}] &= \theta^{666.44-1}\exp(-4.44\theta-2\theta)\\ &= \theta^{666.44-1}\exp(-6.44\theta)\end{aligned} \tag{6.33}$$

In other words, the posterior is $Gamma(666.44, 6.44)$.

This is identical to the posterior we obtained in the previous exercise. This situation will always be true when we have conjugate priors: the result will be the same regardless of whether we take all the data together or successively fit new data, using the posterior of the previous fit as our prior. This is because the multiplication of the terms will always give the same result regardless of how they are rearranged.

6.1.5 Summary of results for problems 1-4

	Prior	Lik.	Post.	Mean	Var
Prob. 1, 2	$Ga(\frac{10000}{225}, \frac{100}{225})$	$\exp(-n\theta)\theta^{\Sigma_i^n x_i}$	$Ga(a+\sum_i^n x_i, b+n)$	$\frac{a+\sum_i^n x_i}{b+n}$	$\frac{a+\sum_i^n x_i}{(b+n)^2}$
Prob. 3	$Ga(\frac{10000}{225}, \frac{100}{225})$	$\exp(-4\theta)\theta^{422}$	$Ga(\frac{10000}{225}+422, 4+\frac{100}{225})$	104.95	23.61
Prob. 4 (method 1)	$Ga(\frac{10000}{225}, \frac{100}{225})$	$\exp(-6\theta)\theta^{622}$	$Ga(666.44, 6.44)$	103.41	16.05
Prob. 4 (method 2)	$Ga(\frac{10000}{225}+422, 4+\frac{100}{225})$	$\exp(-2\theta)(2\theta)^{200}$	Ga(666.44, 6.44)	103.41	16.05

6.1.6 Problem 5

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

The posterior mean is:

$$\frac{a*}{b*} = \frac{a+\sum x_i}{n+b} \tag{6.34}$$

This can be rewritten as

$$\frac{a*}{b*} = \frac{a+n\bar{x}}{n+b} \tag{6.35}$$

Dividing both the numerator and denominator by b:

$$\frac{a*}{b*} = \frac{(a+n\bar{x})/b}{(n+b)/b} = \frac{a/b+n\bar{x}/b}{1+n/b} \tag{6.36}$$

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

$$\frac{a/b+n\bar{x}/b}{1+n/b} = \frac{m+\frac{n}{b}\bar{x}}{1+\frac{n}{b}} \tag{6.37}$$

We can rewrite this as:

$$\frac{m+\frac{n}{b}\bar{x}}{1+\frac{n}{b}} = \frac{m\times 1}{1+\frac{n}{b}} + \frac{\frac{n}{b}\bar{x}}{1+\frac{n}{b}} \tag{6.38}$$

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

$$m\frac{w_1}{w_1+w_2} + \bar{x}\frac{w_2}{w_1+w_2} \tag{6.39}$$

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.

This makes sense: as sample size increases, the likelihood will dominate in determining the posterior mean.

Regarding variance, since the variance of the posterior is:

$$\frac{a*}{b*^2} = \frac{(a+n\bar{x})}{(n+b)^2} \tag{6.40}$$

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

7
Gibbs sampling and the Metropolis-Hastings algorithm

7.1 A preview of where we are going

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:

$$E(f(X)) \approx \frac{1}{n}\sum f(X) \tag{7.1}$$

For example:

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

[1] 0.013602
```

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

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

```
> counts<-table(x<1.96)[2]
> ## pretty close to the theoretical value:
> counts/length(x)

 TRUE
0.977

> ## 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.

Markov chain sampling

We have been doing non-Markov chain sampling in the introductory course:

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

```
[1] -1.71635 -1.47290  2.18520  0.33206 -0.11327  1.28744
```

The vector of values sampled here are statistically independent.

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

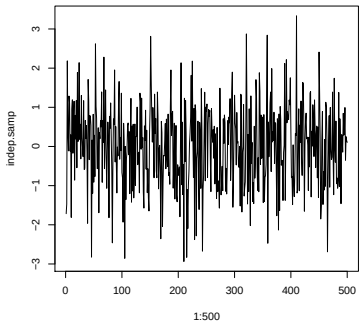


Figure 7.1: Example of independent samples.

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)
> 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")
```

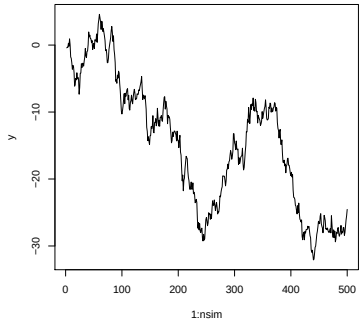


Figure 7.2: Example of markov chain.

7.2 Monte Carlo sampling

In the example with the beta distribution above, we can sample from the posterior distribution easily:

```
> 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
0.3458
```

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%
0.47831

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

  97.5%
0.51411
```

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] 0.47869

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

[1] 0.51436
```

Using calculus (well, we are still using R; I just mean that one could do this by hand, by solving the integral):

```
> integrate(function(x) dbeta(x,shape1=1498,shape2=1519),
+           lower=lower,upper=upper)

0.95 with absolute error < 9.9e-12
```

However—and here we finally get to a crucial point in this course—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),[1] 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 chapter elaborates on this statement.

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

[1] David Lunn, Chris Jackson, David J Spiegelhalter, Nicky Best, and Andrew Thomas. *The BUGS book: A practical introduction to Bayesian analysis*, volume 98. CRC Press, 2012

7.3 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:

$$u = \int_0^z \frac{1}{40}(2x+3) \tag{7.2}$$

```
> 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.

7.4 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

$$mg(z) > f(z) \quad m \text{ a constant} \tag{7.3}$$

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

2. Compute the ratio

$$R = \frac{f(z)}{mg(z)} \tag{7.4}$$

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

$$R = f(x)/mg(z) > u \tag{7.5}$$

```
> #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*.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))

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

 TRUE
0.496
```

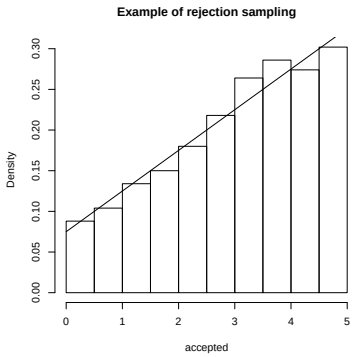


Figure 7.3: Example of rejection sampling.

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.

7.5 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 $\theta_2^j \mid \theta_1^j \theta_3^{j-1} \dots \theta_k^{j-1}$.

$\vdots$

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

4. Return to step 1.

Example: Consider the bivariate distribution:

$$f(x,y) = \frac{1}{28}(2x+3y+2) \tag{7.6}$$

We can analytically work out the conditional distributions:

$$f(x \mid y) = \frac{f(x,y)}{f(y)} = \frac{(2x+3y+2)}{6y+8} \tag{7.7}$$

$$f(y \mid x) = \frac{f(x,y)}{f(y)} = \frac{(2x+3y+2)}{4y+10} \tag{7.8}$$

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 to solve for z_1:

$$u = \int_0^{z_1} \frac{(2x+3y+2)}{6y+8}\, dx \tag{7.9}$$

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

$$u = \int_0^{z_2} \frac{(2x+3y+2)}{4y+10}\, dy \tag{7.10}$$

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")
```

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.

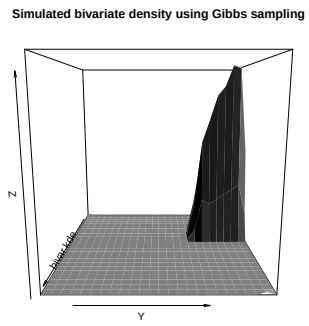


Figure 7.4: Example of posterior distribution of bivariate distribution

7.6 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)*.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)*.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)")
```

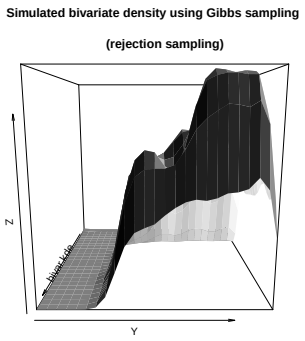


Figure 7.5: Sampling from posterior density, using rejection sampling.

7.7 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 an instructive example.

Here, we can compute the conditional distributions but we can't compute $F^{-1}()$ analytically (I don't understand yet why not—have to try it out).

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

$$f(X \mid Y) = N(\mu_x + \rho\sigma_x \frac{y-\mu_y}{\sigma_y}, \sigma_x\sqrt{(1-\rho^2)}) \qquad (7.11)$$

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")
```

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".

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

```
> plot(x,y,type="l",col="red")
```

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%
-2.0107

> quantile(x,0.975)

 97.5%
1.9951

> mean(x)

[1] -0.057775

> quantile(y,0.025)

   2.5%
-1.9209

> quantile(y,0.975)

 97.5%
1.8278

> mean(y)

[1] -0.023617
```

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, we get:

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

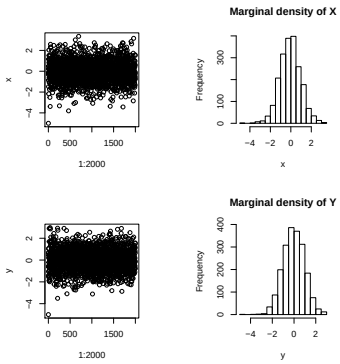


Figure 7.6: Trace plots.

```
   2.5%
-1.9461

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

 97.5%
1.9901

> mean(x[501:2000])

[1] -0.041811

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

   2.5%
-1.9192

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

 97.5%
1.7912

> mean(y[501:2000])

[1] -0.034134
```

7.8 Sampling the parameters given the data

The Bayesian approach is that, conditional of 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:

$$p(\sigma^2 \mid X) \propto InverseGamma((n-1)/2,(n-1)var(x)/2) \tag{7.12}$$

The conditional distribution:

$$p(\sigma^2 \mid \mu, X) \propto InverseGamma(n/2,\sum(x_i-\mu)^2/2) \tag{7.13}$$

$$p(\mu \mid \sigma^2, \mid X) \propto N(\bar{x},\sigma^2/n) \tag{7.14}$$

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.

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("ScottLynchData/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)
```

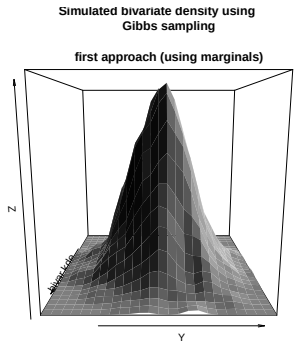


Figure 7.7: Gibbs sampling using marginals.

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("ScottLynchData/education.dat",
+                         header=F)[,1])
> mu<-rep(0,2000)
> 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)))
+ }
```

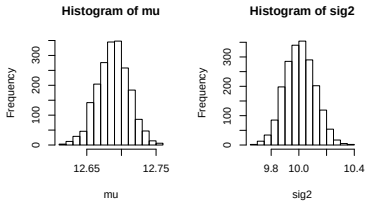


Figure 7.8: Marginal posterior distributions for μ and σ^2.

First, we drop the burn-in 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)")
```

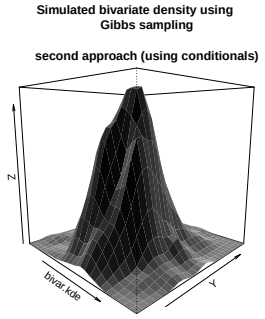


Figure 7.9: Posterior distribution using conditionals.

7.9 Summary of the story so far

This summary is taken, essentially verbatim but reworded a bit, from Lynch[2] (pages 103-104).

[2] Scott Michael Lynch. *Introduction to applied Bayesian statistics and estimation for social scientists.* Springer, 2007

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-Hastings algorithm**, which we discuss next.

7.10 Metropolis-Hastings sampling

I got this neat informal presentation of the idea from
http://www.faculty.biol.ttu.edu/strauss/bayes/LectureNotes/09_MetropolisAlgorithm.pdf
Quoted almost verbatim (slightly edited):

1. Imagine a politician visiting six islands:
   (a) Wants to visit each island a number of times proportional to its population.

(b) Doesn't know how many islands there are.

(c) Each day: might move to a new neighboring island, or stay on current island.

2. Develops simple heuristic to decide which way to move:

   (a) Flips a fair coin to decide whether to move to east or west.

   (b) If the proposed island has a larger population than the current island, moves there.

   (c) If the proposed island has a *smaller* population than the current island, moves there probabilistically, based on uniform distribution.

   Probability of moving depends on relative population sizes:

$$p_{move} = \frac{p_{proposed}}{p_{current}} \tag{7.15}$$

3. End result: probability that politician is on any one of the islands exactly matches the relative proportions of people on the island. We say that the probability distribution **converges** to a particular distribution.

Illustration of convergence using a discrete Markov chains [Adapted slightly from Jim Albert's book[3].]

[3] Jim Albert. *Bayesian computation with R*. Springer, 2009

The above politician example can be made concrete as follows in terms of a Markov chain. A Markov chain defines probabilistic movement from one state to the next. Suppose we have 6 states; a **transition matrix** can define the probabilities:

```
> ## Set up transition matrix:
> T<-matrix(rep(0,36),nrow=6)
> diag(T)<-0.5
> offdiags<-c(rep(0.25,4),0.5)
> for(i in 2:6){
+ T[i,i-1]<-offdiags[i-1]
+ }
> offdiags2<-c(0.5,rep(0.25,4))
> for(i in 1:5){
+ T[i,i+1]<-offdiags2[i]
+ }
> T

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,] 0.50 0.50 0.00 0.00 0.00 0.00
[2,] 0.25 0.50 0.25 0.00 0.00 0.00
```

```
[3,] 0.00 0.25 0.50 0.25 0.00 0.00
[4,] 0.00 0.00 0.25 0.50 0.25 0.00
[5,] 0.00 0.00 0.00 0.25 0.50 0.25
[6,] 0.00 0.00 0.00 0.00 0.50 0.50
```

Note that the rows sum to 1, i.e., the probability mass is distributed over all possible transitions from any one location:

```
> rowSums(T)

[1] 1 1 1 1 1 1
```

We can represent a current location as a probability vector: e.g., in state one, the transition probabilities for possible moves are:

```
> T[1,]

[1] 0.5 0.5 0.0 0.0 0.0 0.0
```

We can also simulate a random walk based on T:

```
> nsim<-500
> s<-rep(0,nsim)
> ## initialize:
> s[1]<-3
> for(i in 2:nsim){
+   s[i]<-sample(1:6,size=1,prob=T[s[i-1],])
+ }
> plot(1:nsim,s,type="l")
```

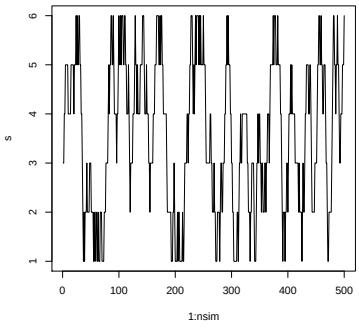


Figure 7.10: A random walk with a discrete markov chain.

Note that the above random walk is non-deterministic: if we rerun the above code multiple times, we will get different patterns of movement.

This Markov chain converges to a particular distribution of probabilities of visiting states 1 to 6. We can see the convergence happen by examining the proportions of visits to each state after blocks of steps that increase by 500 steps:

```
> nsim<-50000
> s<-rep(0,nsim)
> ## initialize:
> s[1]<-3
> for(i in 2:nsim){
+   s[i]<-sample(1:6,size=1,prob=T[s[i-1],])
+ }
> blocks<-seq(500,50000,by=500)
> n<-length(blocks)
```

```
> ## store transition probs over increasing blocks:
> store.probs<-matrix(rep(rep(0,6),n),ncol=6)
> ## compute relative frequencies over increasing blocks:
> for(i in 1:n){
+   store.probs[i,]<-table(s[1:blocks[i]])/blocks[i]
+ }
> ## convergence for state 1:
> for(j in 1:6){
+ if(j==1){
+ plot(1:n,store.probs[,j],type="l",lty=j,xlab="block",
+      ylab="probability")
+ } else {
+   lines(1:n,store.probs[,j],type="l",lty=j)
+ }
+ }
```

Note that each of the rows of the store.probs matrix is a probability mass function, which defines the probability distribution for the 6 states:

```
> store.probs[1,]
```

```
[1] 0.174 0.280 0.198 0.222 0.090 0.036
```

This distribution is settling down to a particular set of values; that's what we mean by convergence. This particular set of values is:

```
> (w<-store.probs[n,])
```

```
[1] 0.09786 0.19550 0.19440 0.20444 0.20590 0.10190
```

w is called a **stationary** distribution. If $wT = w$, then then w is the stationary distribution of the Markov chain.

```
> round(w%*%T,digits=2)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  0.1  0.2  0.2  0.2 0.21  0.1
```

```
> round(w,digits=2)
```

```
[1] 0.10 0.20 0.19 0.20 0.21 0.10
```

This discrete example gives us an intuition for what will happen in continuous distributions: we will devise a Markov chain such that the chain will converge to the distribution we are interested in sampling from.

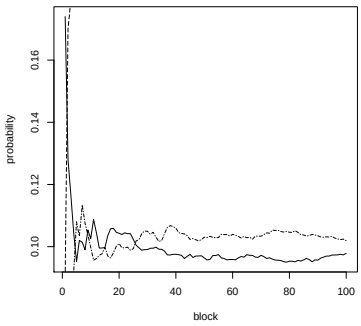


Figure 7.11: Convergence in the discrete markov chain example.

More formal presentation of Metropolis-Hastings One key point here is that in MH, we need only specify the unnormalized joint density for the parameters, we don't need the conditional densities. One price to be paid is greater computational overhead.

As Lynch[4] (page 108) puts it:

> "A key advantage to the MH algorithm over other methods of sampling, like inversion and rejection sampling, is that it will work with multivariate distributions (unlike inversion sampling), and we do not need an enveloping function (as in rejection sampling)."

[4] Scott Michael Lynch. *Introduction to applied Bayesian statistics and estimation for social scientists.* Springer, 2007

The algorithm (taken almost verbatim from Lynch):

1. Establish starting values S for the parameter: $\theta^{j=0} = S$, using MLE or some other method (apparently the starting values don't matter—the distribution to which the Markov chain converges will be the posterior distribution of interest—but poor starting values will affect speed of convergence). Set $j = 1$.

2. Draw a "candidate" parameter, θ^c from a "proposal density," $\alpha(\cdot)$.

3. Compute the ratio:

$$R = \frac{f(\theta^c)}{f(\theta^{j-1})} \frac{\alpha(\theta^{j-1} \mid \theta^c)}{\alpha(\theta^c \mid \theta^{j-1})} \tag{7.16}$$

   The first part of the ratio ($\frac{f(\theta^c)}{f(\theta^{j-1})}$) is called the **importance ratio**: the ratio of the unnormalized posterior density evaluated at the candidate parameter value (θ^c) to the posterior density evaluated at the previous parameter value (θ^{j-1}).

   The second part of the ratio is the ratio of the **proposal** densities evaluated at the candidate and previous points. This ratio adjusts for the fact that some candidate values may be selected more often than others.

4. Compare R with a $Unif(0,1)$ random draw u. If $R > u$, then set $\theta^j = \theta^c$. Otherwise, set $\theta^j = \theta^{j-1}$.

   An equivalent way of saying this: set an **acceptance probability** *aprob* as follows:

$$aprob = min(1, R) \tag{7.17}$$

   Then sample a value u from a uniform $u \sim Unif(0,1)$.

   If $u < aprob$, then accept candidate, else stay with current value (x_{i-1}). What this means is that if aprob is larger than 1, then we accept the candidate, and if it is less than 1, then we accept the candidate with probability *aprob*.

5. Set $j = j+1$ and return to step 2 until enough draws are obtained.

Step 2 is like drawing from an envelope function in rejection sampling, except that the proposal density (which is any density that it's easy to sample from) does not have to envelope the density of interest. As in rejection sampling, we have to check whether the draw of a parameter value from this proposal density can be considered to be from the density of interest. For example, a candidate value could be drawn from a normal distribution $\theta^c = \theta^{j-1} + N(0,c)$, where c is a constant. This gives us a "random walk Metropolis algorithm".

We can quickly implement such a random walk to get a feel for it. This is one of the things the MH algorithm will be doing.

```
> ## implementation of random walk Metropolis:
> theta.init<-5
> values<-rep(NA,1000)
> values[1]<-theta.init
> for(i in 2:1000){
+   values[i]<-values[i-1]+rnorm(1,mean=0,sd=1)
+ }
> plot(1:1000,values,type="l",main="Random walk")
```

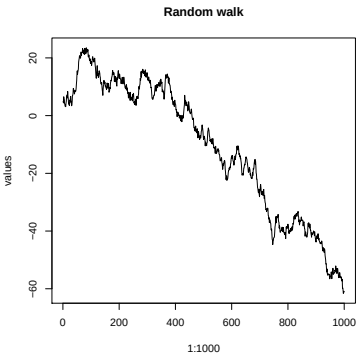


Figure 7.12: The random walk in the random walk Metropolis algorithm.

And here is an example of a draw from a proposal density: Suppose that at the $(j-1)$-th step, we have the value θ^{j-1} (in the very first step, when j=2, we will be using the starting value of θ). Suppose that our proposal density is a normal. Our candidate draw will then be made using θ^{j-1}: $\theta^c = \theta^{j-1} + N(0,c)$, c some constant. Suppose $\theta^{j-1} = 5$, and $c = 1$. We can draw a candidate value θ^c as follows:

```
> (theta.c<-5 + rnorm(1,mean=0,sd=1))

[1] 4.9246
```

This is now our θ^c.

For Step 3 in the algorithm, we can now calculate:

$$\alpha(\theta^c \mid \theta^{j-1}) \tag{7.18}$$

as follows:

```
> (p1<-dnorm(theta.c,mean=5,sd=1))

[1] 0.39781
```

Here is a visualization of what the above call does: given the candidate θ^c, we are computing the probability of transitioning to it (the probability of it being the next position moved to on the x-axis) given a normal distribution with mean 5, which is our starting value for x (sd of the normal distribution is 1, for simplicity).

```
> plot(function(x) dnorm(x,mean=5), 0, 10,
+     ylim=c(0,0.6),
+     ylab="density",xlab="X")
> points(theta.c,p1,pch=16,col="red")
```

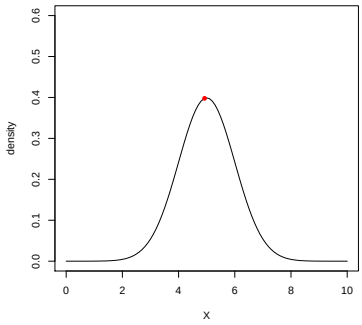


Figure 7.13: Illustration of calculation of $\alpha(\theta^c \mid x)$.

We can also calculate

$$\alpha(\theta^{j-1} \mid \theta^c) \tag{7.19}$$

as follows:

```
> (p2<-dnorm(5,mean=theta.c,sd=1))

[1] 0.39781
```

Let's visualize what is being computed here. First I re-plot the above visualization for comparison.

```
> op<-par(mfrow=c(1,2),pty="s")
> plot(function(x) dnorm(x,mean=5), 0, 10,
+     ylim=c(0,0.6),
+     ylab="density",xlab="X",main="alpha(theta.c|x)")
> points(theta.c,p1,pch=16,col="red")
> plot(function(x) dnorm(x,mean=theta.c), 0, 10,
+     ylim=c(0,0.6),
+     ylab="density",xlab="X",new=F,lty=2,,main="alpha(x|theta.c)")
> points(5,p2,pch=16,col="black")
```

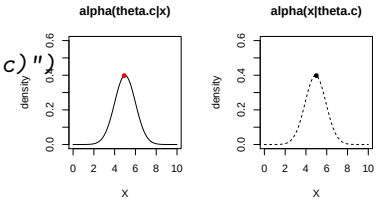


Figure 7.14: Illustration of calculation of $\alpha(x \mid \theta^c)$.

These two calculations allows us to compute the ratio $\frac{\alpha(\theta^{j-1}|\theta^c)}{\alpha(\theta^c|\theta^{j-1})}$ (which will turn up in the next step):

```
> p2/p1

[1] 1
```

Note that the ratio is 1. If you look at the figure above, you will see that for symmetric distributions (here, the normal distribution) this ratio will **always** be 1. In other words, for symmetric proposal distributions, we can ignore the second term in the calculation of R and focus only on the importance ratio (see above).

What does it mean for the p2/p1 ratio to be 1? It means is that the probability of transitioning to θ^{j-1} when sampling from $N(\theta^c,1)$ is the same as the probability of transitioning to θ^c when sampling from $N(\theta^{j-1},1)$. (Variance is set at 1 just for illustration; one could choose a different value.)

The ratio p2/p1 becomes relevant for asymmetric proposal densities. Lynch writes: "This ratio adjusts for the fact that, with asymmetric proposals, some candidate values may be selected more often

than others. . . " In such situations p1 would be larger than p2, and so p2/p1 will be smaller than 1. Adding this term to the calculation of R down-adjusts the value of R.

Let's try to visualize this. What we see below is that the probability of transitioning to the candidate is higher given x (LHS plot) than the probability of transitioning to x given the candidate (RHS).

```
> op<-par(mfrow=c(1,2),pty="s")
> plot(function(x) dgamma(x,shape=5), 0, 10,
+     #ylim=c(0,0.6),
+     ylab="density",xlab="X",main="alpha(theta.c|x)")
> p1<-dgamma(theta.c,shape=5)
> points(theta.c,p1,pch=16,col="red")
> plot(function(x) dgamma(x,shape=theta.c), 0, 10,
+     #ylim=c(0,0.6),
+     ylab="density",xlab="X",
+      lty=2,,main="alpha(x|theta.c)")
> p2<-dgamma(5,shape=theta.c)
> points(5,p2,pch=16,col="black")
```

This asymmetry is what the second term in the ratio R corrects for.

When the second term is needed (when it's 1 because we have a symmetric proposal), we have the Metropolis algorithm. When the second term is needed, we have the Metropolis-Hastings algorithm.

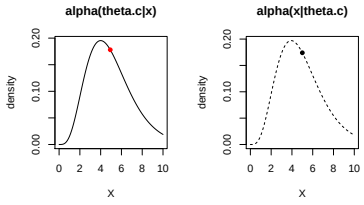


Figure 7.15: The effect of asymmetry in the MH algorithm.

7.11 Implementing and visualizing the MH algorithm

Here is a simple implementation of a Metropolis algorithm, which involves such a symmetric situation:

```
> ## source:
> ##http://www.mas.ncl.ac.uk/~ndjw1/teaching/sim/metrop/metrop.html
> ## slightly adapted by SV:
> #n is num. of simulations
> norm<-function (n)
+ {
+         vec <- vector("numeric", n)
+         x <- 0
+         vec[1] <- x
+         for (i in 2:n) {
+                 can <- x + rnorm(1,mean=0,sd=1)
+                 ## to-do need to anticipate this in notes:
+                 aprob <- min(1, dnorm(can)/dnorm(x))
+                 u <- runif(1)
+                 if (u < aprob){
```

```
+                         x <- can
+                 }
+                 vec[i] <- x
+         }
+         vec
+ }
```

```
> normvec<-norm(5000)
> op<-par(mfrow=c(2,1))
> plot(ts(normvec))
> hist(normvec,30,freq=F)
```

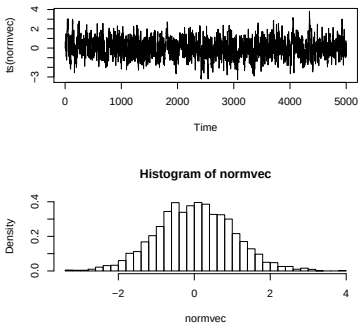


Figure 7.16: A demonstration of the Metropolis algorithm.

```
> ## we get the correct posterior distribution:
> mean(normvec)

[1] 0.019414

> sd(normvec)

[1] 1.009
```

Visualization of the unfolding move-no move decisions I do this in two ways. One shows how the choice between the candidate and the current x is resolved, and the other shows the end result of the algorithm's run.

I first set up a function to create transparent colors (there is probably a better way than this, need to look into that):

```
> ## I got this from the internet somewhere:
> addTrans <- function(color,trans)
+ {
+   # This function adds transparency to a color.
+   # Define transparency with an integer between 0 and 255
+   # 0 being fully transperant and 255 being fully visible
+   # Works with either color and trans a vector of equal length,
+   # or one of the two of length 1.
+
+   if (length(color)!=length(trans)&!any(c(length(color),
+                                            length(trans))==1))
+     stop("Vector lengths not correct")
+   if (length(color)==1 & length(trans)>1) color <-
+     rep(color,length(trans))
+   if (length(trans)==1 & length(color)>1) trans <-
+     rep(trans,length(color))
+
```

```
+   num2hex <- function(x)
+   {
+     hex <- unlist(strsplit("0123456789ABCDEF",split=""))
+     return(paste(hex[(x-x%%16)/16+1],hex[x%%16+1],sep=""))
+   }
+   rgb <- rbind(col2rgb(color),trans)
+   res <- paste("#",apply(apply(rgb,2,num2hex),2,
+                          paste,collapse=""),sep="")
+   return(res)
+ }
```

First, the incremental presentation, with 100 steps:

```
> n<-100
> norm<-function (n,display="candidate")
+ {
+         vec <- vector("numeric", n)
+         x <- 0
+         vec[1] <- x
+         plot(function(x) dnorm(x), -10, 10,
+         ylim=c(0,0.6),
+         ylab="density",xlab="X")
+         for (i in 2:n) {
+                if(display=="x"){
+                 points(x,jitter(0),pch=16,
+                        col=addTrans("black",100),cex=1.3)}
+                 can <- x + rnorm(1,mean=0,sd=1)
+                 aprob <- min(1, dnorm(can)/dnorm(x))
+                 u <- runif(1)
+                 if (u < aprob){
+                         x <- can
+                 }
+                 vec[i] <- x
+                if(display=="candidate"){
+                 points(can,jitter(0),pch=16,
+                        col=addTrans("red",100),cex=1.3)}
+ #                Sys.sleep(1)
+                if(display=="choice"){
+                 points(vec[i],jitter(0),pch=16,
+                        col=addTrans("green",100),cex=2)}
+                 Sys.sleep(0.5)
+         }
+         vec
+ }
> #op<-par(mfrow=c(1,3),pty="s")
```

```
> normvec<-norm(n=100,display="candidate")
> normvec<-norm(n=100,display="x")
> normvec<-norm(n=100,display="choice")
```

The red dots are the candidates, the black the X's, and the green ones are ones are the choices adopted at each step.

The next visualization shows the end result of the algorithm run:

```
> n<- 500
> can.store<-rep(NA,n)
> x.store<-rep(NA,n)
> norm<-function (n)
+ {
+         vec <- vector("numeric", n)
+         x <- 0
+         vec[1] <- x
+         for (i in 2:n) {
+                 can <- x + rnorm(1,mean=0,sd=1)
+                 can.store[i]<-can
+                 x.store[i]<-x
+                 aprob <- min(1, dnorm(can)/dnorm(x))
+                 u <- runif(1)
+                 if (u < aprob){
+                         x <- can
+                 }
+                 vec[i] <- x
+         }
+         list(vec,can.store,x.store)
+ }

> normvec<-norm(n)
> ## recover history of candidates and x's:
> ## discard first half:
> start<-n/2
> can.store<-normvec[2][[1]][start:n]
> x.store<-normvec[3][[1]][start:n]
> plot(function(x) dnorm(x), -10, 10,
+                   ylim=c(0,0.6),
+                   ylab="density",xlab="X")
> #ys<-seq(0,.4,by=0.4/(n/2))
> points(can.store,jitter(rep(0,length(can.store))),cex=1,pch=16,
+        col=addTrans(rep("red",length(can.store)),10))
> points(x.store,jitter(rep(0.05,length(x.store))),cex=1,pch=16,
+        col=addTrans(rep("black",length(can.store)),10))
> #Sys.sleep(2)
```

```
> #                plot(function(x) dnorm(x,mean=can), -10, 10,
> #                  ylim=c(0,1),
> #                  ylab="density",xlab="X",add=T,col="red")
> cand.prob<-dnorm(can.store)
> x.prob<-dnorm(x.store)
> ## cases where the candidate is chosen:
> move<-runif(1)<cand.prob/x.prob
> ## at which steps did we take the candidates or stayed at x:
> indices<-c(which(move),which(!move))
> ## erase earlier candidates:
> #points(can.store,ys,cex=2,pch=16,col="white")
> #points(x.store,ys,cex=2,pch=16,col="white")
>
> ## redraw target dist:
> #plot(function(x) dnorm(x), -10, 10,
> #                  ylim=c(0,0.6),
> #                  ylab="density",xlab="X")
>
>
> move.history<-data.frame(indices=indices,
+               points=c(can.store[which(move)],
+                        x.store[which(!move)]))
> ## reordered to reflect history:
> move.history<-move.history[order(move.history[1,])]
> points(move.history$points,
+        jitter(rep(0.1,length(move.history$points))),
+        pch=16,
+        col=addTrans(rep("green",
+                         length(move.history$points)),10),
+        cex=1.5)
> #legend(5,.6,pch=16,col=c("red","black"),
> #       legend=c(paste("candidate",round(cand.prob,digits=2)),
> #                paste("previous",round(x.prob,digits=2))))
>
> hist(move.history$points,freq=FALSE,add=T,
+      col=addTrans(rep("green",length(move.history$points)),10))
```

When we have an asymmetry, we use the Metropolis-Hastings algorithm:

```
> ## source: http://www.mas.ncl.ac.uk/~ndjw1/teaching/sim/metrop/indep.r
> # metropolis-hastings independence sampler for a
> # gamma based on normal candidates with the same mean and variance
>
> gamm<-function (n, a, b)
```

```
+ {
+ mu <- a/b
+ sig <- sqrt(a/(b * b))
+ vec <- vector("numeric", n)
+ x <- a/b
+ vec[1] <- x
+ for (i in 2:n) {
+     can <- rnorm(1, mu, sig)
+     importance.ratio<-(dgamma(can, a, b)/dgamma(x,a, b))
+     adjustment<-(dnorm(can, mu, sig)/dnorm(x,mu,sig))
+     aprob <- min(1,importance.ratio/adjustment)
+     u <- runif(1)
+     if (u < aprob)
+     x <- can
+     vec[i] <- x
+ }
+ vec
+ }
> vec<-gamm(10000,2.3,2.7)
> op<-par(mfrow=c(2,1))
> plot(ts(vec))
> hist(vec,30)
> # end
> vec.orig<-vec
```

We can see why we need the adjustment in the ratio R by simply removing it. Without the adjustment we get a mis-characterization of the posterior distribution of interest.

```
> gamm<-function (n, a, b)
+ {
+ mu <- a/b
+ sig <- sqrt(a/(b * b))
+ vec <- vector("numeric", n)
+ x <- a/b
+ vec[1] <- x
+ for (i in 2:n) {
+     can <- rnorm(1, mu, sig)
+     importance.ratio<-(dgamma(can, a, b)/dgamma(x,a, b))
+     adjustment<-(dnorm(can, mu, sig)/dnorm(x,mu,sig))
+     #aprob <- min(1,importance.ratio/adjustment)
+     aprob <- min(1,importance.ratio)
+     u <- runif(1)
+     if (u < aprob)
+     x <- can
```

```
+     vec[i] <- x
+ }
+ vec
+ }
> vec<-gamm(10000,2.3,2.7)

> hist(vec,30,col=addTrans(rep("blue",length(vec)),10),freq=F)
> hist(vec.orig,30,col=addTrans(rep("red",length(vec.orig)),10),
+      freq=F,add=T)
```

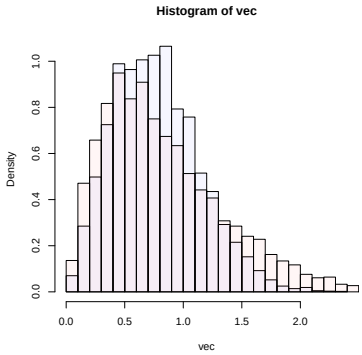


Figure 7.17: The effect of ignoring asymmetry on the posterior distribution.

Note that we have not explained why the Metropolis-Hastings algorithm works. A full explanation apparently requires Markov chain theory, but an intuitive explanation is that once a sample from the target distribution has been obtained, all subsequent samples will be from the target distribution. There are more details in Gilks et al.[5]

[5] Walter R Gilks, Sylvia Richardson, and David J Spiegelhalter. *Markov chain Monte Carlo in practice*, volume 2. CRC press, 1996

7.12 Assessing convergence: Fat hairy caterpillars

Visually, you can assess convergence by looking at the chain to see if it looks like a "fat hairy caterpillar" (This is how Lunn et al. describe it.)

If you use multiple chains with different starting values, and pool all the chains to assess convergence, you will generally get convergence (at least in the models we consider in this course). In our linear mixed models (coming up) we use 3 or 4 chains. The downside of running multiple chains is computational overhead. However, for regular psycholinguistic data this is not going to be a major problem. (It will probably be a major problem for ERP data; but we'll see how that goes).

The Gelman-Rubin (or Brooks-Gelman-Rubin) diagnostic involves sampling from multiple chains with "overdispersed" original values, and then comparing between and within group variability.[6] Within variance is represented by the mean width of the 95% posterior Credible Intervals (CrI) of all chains, from the final T iterations. Within variance is represented by the width of the 95% CrI using all chains pooled together (for the T iterations). If the ratio $\hat{R} = B/W$ is approximately 1, we have convergence. Alternatively, you can run the model on incrementally increasing blocks of iterations T_i, e.g., $T_1 = 1000, T_2 = 2000$, etc., and assess the emergence of convergence.

[6] Andrew Gelman and Donald B Rubin. Inference from iterative simulation using multiple sequences. *Statistical science*, pages 457–472, 1992

7.13 Other sampling methods

There are other methods of sampling available than the ones presented above. Examples are slice sampling and adaptive rejection sampling. But for our purposes we only need to know how the Gibbs and MH algorithms work, and that too only *in principle*; we will never need

to implement them for data analysis (I guess I should have mentioned this earlier). JAGS will do the sampling for us once we define the model.

8
Using JAGS for modeling

8.1 Introduction to JAGS

8.2 A simple example

This example is taken from Lunn et al (their example 2.4.1).[1] It illustrates basic BUGS syntax, using JAGS. We can also run this model within WinBUGS, as shown below briefly. Stan is another option; I may show examples in class of Stan usage.

[1] David Lunn, Chris Jackson, David J Spiegelhalter, Nicky Best, and Andrew Thomas. *The BUGS book: A practical introduction to Bayesian analysis*, volume 98. CRC Press, 2012

Suppose you have $Z \sim N(0,1)$. Assume that we transform this random variable as follows:

$$Y = (2Z+1)^3 \tag{8.1}$$

We can work out the distribution of Y analytically (but it's hard, at least for me), or we can simulate it and look at the distribution, and compute any summary statistic we want.

First, we write the model into a text file and save it (we do this within R). Note that in the normal distribution, the scale parameter is stated in terms of precision=1/variance. I don't know why BUGS has this convention, but it's (unfortunately) the way it is.

My guess as to why variance is defined in terms of precision is that it's convenient for algebraic manipulation.

```
> cat("
+ # Fixing data to be used in model definition
+ model
+    {
+     ## the second parameter is precision=1/variance
+     Z ~ dnorm(0,1)
+     Y <- pow(2*Z + 1, 3)
+     P10 <- step(Y-10)
+    }",
+      file="JAGSmodels/binomialexample1.jag" )
```

Then we specify the variables we want to track the distribution of:

```
> track.variables<-c("Y")
```

Next, we create an object representing the model. I use system.time to keep track of how long this takes. The timing issue will be interesting when we compare performance with other software like WinBUGS and Stan.

```
> library(rjags,quietly=T)
> system.time(
+ binomialexample1.mod <- jags.model(
+   file = "JAGSmodels/binomialexample1.jag",
+                       n.chains = 1,
+                        n.adapt =2000 ,quiet=T)
+             )

   user  system elapsed
  0.001   0.001   0.011
```

Next, we generate samples:

```
> system.time(
+ binomialexample1.res <- coda.samples( binomialexample1.mod,
+                                    var = track.variables,
+                                  n.iter = 10000,
+                                    thin = 20 ) )

   user  system elapsed
  0.006   0.001   0.014
```

Finally, we summarize the distribution:

```
> summary( binomialexample1.res )

Iterations = 20:10000
Thinning interval = 20
Number of chains = 1
Sample size per chain = 500

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean             SD       Naive SE Time-series SE
         11.42          34.72           1.55           1.55

2. Quantiles for each variable:

    2.5%      25%      50%      75%    97.5%
-28.2812  -0.0658   1.0081  13.0309  95.4793
```

We can also plot the distribution and the chain, see Fig 8.1.

Incidentally, you can plot the posterior distributions etc. in `ggplot2` if you are so inclined. See below for examples, and http://xavier-fim.net/packages/ggmcmc/ for a tutorial.

```
> library(ggmcmc)
> res<-ggs(binomialexample1.res)
> #ggmcmc(res)
> ggs_histogram(res)
> ggs_density(res)
> ggs_traceplot(res)
```

Of course, we could have written the above JAGS code in R:

```
> X<-rnorm(10000)
> Y<-(2*X+1)^3
> summary(Y)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-277.00   -0.04    1.05   13.20   12.80  619.00
```

8.3 Integrating out an unknown parameter

Suppose we have a sampling distribution $p(y \mid \theta)$. This could be, for example, $y \sim N(0,1)$ (here, θ is a vector representing the mean and variance). Let our certainty about the parameters be expressed by the probability distribution $p(\theta)$. In this situation, we can produce a predictive distribution by "integrating out the unknown parameter":[2,3]

$$p(y) = \int p(y,\theta)\,d\theta = \int p(y \mid \theta)p(\theta)\,d\theta \tag{8.4}$$

This kind of "integrating out" can be done in BUGS quite easily. Suppose we have a random variable Y that has a binomial distribution with success probability θ and sample size n, and our uncertainty about θ is expressed as the distribution Beta(a,b). For specific values of a,b, and n (see code below), we can write the model as:

```
> ## remove stray Y object, otherwise JAGS gets confused:
> rm(Y)
> cat("
+ model
+     {
+      theta ~ dbeta(3,27)
+      Y ~ dbin(theta,20)
+     }",
+       file="JAGSmodels/integratingout.jag")
```

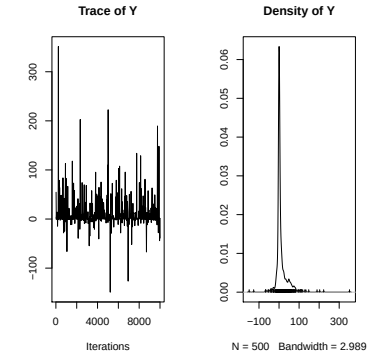


Figure 8.1: The distribution of the transformed binomial random variable, using JAGS.

```
> plot(density(Y))
```

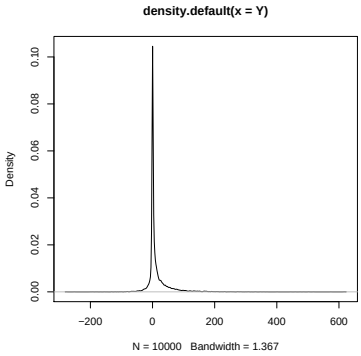


Figure 8.2: The distribution of the transformed binomial random variable, using R.

[2] Recall the discussion about marginal pdfs, page 60.

[3] This trick, of integrating out the unknown parameter turns up in the estimation of linear mixed model (LMM) fixed effects parameters.

In LMMs, the model is:

$$Y_i = X_i\beta + Z_i\beta_i + \varepsilon_i, \quad i = 1,\ldots,M \tag{8.2}$$

where $b_i \sim N(0,\Psi), \varepsilon_i \sim N(0,\sigma^2 I)$. Let θ be the parameters that determine Ψ.

$$\begin{aligned} L(\beta,\theta,\sigma^2 \mid y) &= p(y:\beta,\theta,\sigma^2) \\ &= \prod_i^M p(y_i:\beta,\theta,\sigma^2) \\ &= \prod_i^M \int p(y_i \mid b_i,\beta,\sigma^2)p(b_i:\theta,\sigma^2)\,db_i \end{aligned} \tag{8.3}$$

we want the density of the observations (y_i) given the parameters β,θ and σ^2 only. We can integrate out the nuisance parameter b_i, which are the BLUPs (best linear unbiased predictors) in a linear mixed model. to-do: spell this out with an example.

Then we generate samples from $p(Y)$ in the usual way:

```
> track.variables<-c("Y")
> library(rjags)
> intout.mod <- jags.model(
+   file = "JAGSmodels/integratingout.jag",
+                       n.chains = 2,
+                        n.adapt =2000 , quiet=T)
> intout.res <- coda.samples( intout.mod,
+                                  var = track.variables,
+                                n.iter = 10000,
+                                  thin = 20 )
> summary( intout.res )

> plot(intout.res)

Iterations = 20:10000
Thinning interval = 20
Number of chains = 2
Sample size per chain = 500

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean             SD       Naive SE Time-series SE
        1.9340         1.6794         0.0531         0.0515

2. Quantiles for each variable:

 2.5%   25%   50%   75% 97.5%
    0     1     2     3     6
```

to-do: show to calculate $P(Y \geq 6)$.

8.4 Posterior predictive distributions

This section is taken from Lunn et al., Section 3.2; slighly reworded.[4]

Once we have the posterior distribution $f(\theta \mid y)$, we can derive the predictions based on this posterior distribution using the same trick as above.

$$p(y_{pred} \mid y) = \int p(y_{pred}, \theta \mid y)\, d\theta = \int p(y_{pred} \mid \theta, y) p(\theta \mid y)\, d\theta \qquad (8.5)$$

Assuming that past and future observations are conditionally inde-

[4] David Lunn, Chris Jackson, David J Spiegelhalter, Nicky Best, and Andrew Thomas. *The BUGS book: A practical introduction to Bayesian analysis*, volume 98. CRC Press, 2012

Here is an interesting quote on prediction: "... it does not really matter whether a scientific theory is correct.... What matters is whether the scientific theory allows us to make useful predictions about future observations." p. 32 of Christensen et al. Statisticans are often very interested in the predictive accuracy of their statistical models. Note however that in planned experiments we never derive predictions from the statistical model, but rather derive predictions from our theories, which we are investigating the predictions of using the statistical model. So, we don't really care much about the predictive accuracy of our

pendent given θ, i.e., $p(y_{pred} \mid \theta, y) = p(y_{pred} \mid \theta)$, we can write:

$$p(y_{pred} \mid y) = \int p(y_{pred} \mid \theta) p(\theta \mid y)\, d\theta \tag{8.6}$$

We use this in the next example.

8.5 Computing posterior predictive distributions using JAGS

This is very similar to the example on p. 82. Suppose our prior is Beta(2,2) and the data are Beta(46,54), and the posterior is Beta(47,55).

We first define the data and model. The data includes the parameters for the prior as well (a,b), and the sample size of the predicted values.

```
> ## the data:
> data<-list(a=3,b=27,y=0,n=10,n.pred=20)
> cat("
+ model
+    {
+     ## prior
+     theta ~ dbeta(a,b)
+     ## likelihood
+     y ~ dbin(theta,n)
+     ## predicted posterior distribution
+     y.pred ~ dbin(theta,n.pred)
+    }",
+      file="JAGSmodels/predictionexample1.jag" )
> track.variables<-c("y.pred","theta")
> library(rjags)
> predeg.mod <- jags.model(
+   file = "JAGSmodels/predictionexample1.jag",
+                       data=data,
+                       n.chains = 4,
+                        n.adapt =2000 , quiet=T)
> predeg.res <- coda.samples( predeg.mod,
+                             var = track.variables,
+                             n.iter = 10000,
+                             thin = 20 )
```

The summary of the posterior distribution $p(\theta)$ and predicted values:

```
> summary( predeg.res )

Iterations = 20:10000
Thinning interval = 20
```

```
Number of chains = 4
Sample size per chain = 500

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

         Mean     SD Naive SE Time-series SE
theta  0.0745 0.0404 0.000904       0.000888
y.pred 1.4865 1.4056 0.031431       0.032072

2. Quantiles for each variable:

         2.5%    25%   50%    75% 97.5%
theta  0.0154 0.0441 0.067 0.0971 0.171
y.pred 0.0000 0.0000 1.000 2.0000 5.000
```

Exercise: Compare the above estimated means and standard deviations with the theoretical means and standard deviations. The distributions can also be plotted, see Fig. 8.3.

In the above example, JAGS is sampling directly from the Beta(3,37):

We can also compute the probability of 2 or more successes in the predicted distribution of sample size 20 is:

```
> post.pred<-jags.samples(predeg.mod,var=track.variables,n.iter=10000)
> ## compute probability of theta > 0
> counts.pred<-table(post.pred$y.pred[,,1]>=2)
> counts.pred[2]/sum(counts.pred)

  TRUE
0.4182
```

Similarly, you can also compute any probablistic value from the posterior distribution of θ, which you would normally do with `pbeta`.

8.6 Normal data with unknown mean, known variance

to-do

8.7 MCMC integration in JAGS

This example is from Lunn et al., example 4.1.1.[5] Suppose we have 10 successes from a sample size of 100, assuming a binomial process. Instead of a beta prior on θ, we could could use a non-conjugate prior on a transformation of θ: $logit(\theta) \sim N(\mu, \omega^2)$. Let $\omega^2 = 2.71$. Figuring out the posterior distribution of θ is not possible analytically; but MCMC methods allow us to simulate it.

```
> plot(predeg.res)
```

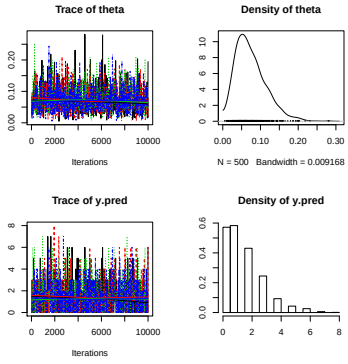


Figure 8.3: Example of predictive posterior distribution.

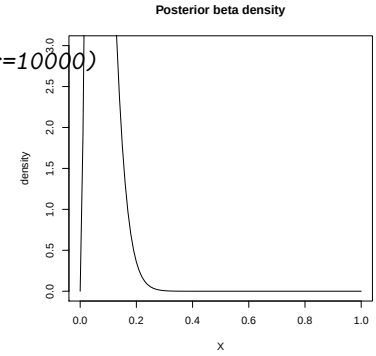


Figure 8.4: The posterior distribution computed in R.

[5] David Lunn, Chris Jackson, David J Spiegelhalter, Nicky Best, and Andrew Thomas. *The BUGS book: A practical introduction to Bayesian analysis*, volume 98. CRC Press, 2012

```
> ## the data:
> data<-list(y=10,n=100)
> cat("
+ model
+    {
+     ## likelihood
+     y ~ dbin(theta,n)
+     ## prior
+     logit(theta) <- logit.theta
+     ## precision 0.368 = 1/2.71
+     logit.theta ~ dnorm(0,0.368)
+    }",
+      file="JAGSmodels/mcmcexample1.jag" )
> track.variables<-c("theta")
> library(rjags)
> mcmceg.mod <- jags.model(
+   file = "JAGSmodels/mcmcexample1.jag",
+                       data=data,
+                       n.chains = 1,
+                        n.adapt =2000 , quiet=T)
> mcmceg.res <- coda.samples( mcmceg.mod,
+                             var = track.variables,
+                             n.iter = 10000,
+                             thin = 20 )
> summary(mcmceg.res)

Iterations = 2020:12000
Thinning interval = 20
Number of chains = 1
Sample size per chain = 500

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean             SD       Naive SE Time-series SE
       0.10822        0.03108        0.00139        0.00155

2. Quantiles for each variable:

  2.5%    25%    50%    75%  97.5%
0.0562 0.0853 0.1055 0.1253 0.1765
```

Homework 6

Modify the above code so that (a) the prior for θ is $Unif(0,1)$, and compute the posterior predictive distribution of y (call it y.pred) for

```
> densityplot(mcmceg.res)
```

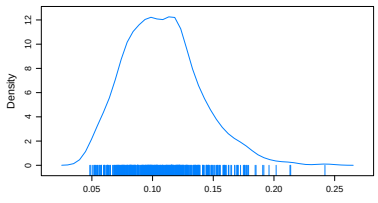


Figure 8.5: Density plot of posterior distribution of θ.

100 future observation. Does the posterior distribution of θ change? Are the predicted values reasonable?

8.8 Multi-parameter models in JAGS

Suppose that we have data y_i which we believe comes from a heavy-tailed t-distribution.

```
> y.sim<-rt(100,d=4)
```

Let's assume that we don't know any of the parameters and set as priors for this data:

1. $\mu \sim N(\gamma, \omega^2)$
2. $1/\sigma^2 \sim Gamma(\alpha, \beta)$
3. $df \sim Unif(2,30)$

Then, the joint posterior distribution of each parameter is:

$$p(\mu, \sigma^2, df \mid y) \propto \text{Likelihood} \times [\text{prior } \mu] \times [\text{prior } \sigma^2] \times [\text{prior } d] \quad (8.7)$$

Now, if we want the marginal distributions, $p(\mu \mid y), p(\sigma^2 \mid y), p(df \mid y)$, we can't compute these analytically. JAGS will compute these for you using MCMC sampling.

```
> n.samples<-100
> y.sim<-rt(n.samples,df=4)
> ## the data:
> data<-list(n=n.samples,gamma=0,
+            inv.omega.squared=0.0001,
+            alpha=0.001,beta=0.001,y=y.sim)
> cat("
+ model
+    {
+     for(i in 1:n){ y[i] ~ dt(mu,r,d)}
+     mu ~ dnorm(gamma,inv.omega.squared)
+     r ~ dgamma(alpha,beta)
+     d ~ dcat(p[])
+     p[1]<-0
+     for(i in 2:30){p[i]<-1/29}
+    }",
+      file="JAGSmodels/multimcmcexample2.jag" )
> track.variables<-c("mu","r","d")
> library(rjags)
```

```
> inits <- list (list(mu=0,r=1,d=10))
> mcmceg.mod <- jags.model(
+   file = "JAGSmodels/multimcmcexample2.jag",
+                       data=data,
+                       inits=inits,
+                       n.chains = 1,
+                        n.adapt =10000 ,quiet=T)
> mcmceg.res <- coda.samples( mcmceg.mod,
+                                  var = track.variables,
+                                n.iter = 10000,
+                                  thin = 20 )
>
> #summary(mcmceg.res)
```

Incidentally, such a model can be fit in WinBUGs as well. Because the original WinBUGS interface is so tedious (so many windows, so little time), I use Andrew Gelman's approach to running WinBUGS (which works only in Windows or Windows emulators like WINE—see Gelman's home page for more details and examples) via R. Below is the code in WinBUGS that does the same as the JAGS code above. We won't be using WinBUGS much, but I include this example because most books seem to use WinBUGS and so some familiarity with WinBUGS models will eventually be needed if you read textbooks on bayesian methods.

```
cat("model {
     for (i in 1:n){
       y[i] ~ dt (mu, r, d)}
       mu ~ dnorm(gamma,inv.omega.squared)
       r ~ dgamma(alpha,beta)
       d ~ dcat(p[])
       p[1]<-0
       for(j in 2:30){p[j]<-1/29}
   }",
     file="WinBUGsmodels/multimcmcexample2.bug" )

library(arm)

n.samp<-100
y.sim<-rt(n.samp,df=4)

data<-list(n=n.samp,gamma=0,inv.omega.squared=0.0001,
           alpha=0.001,beta=0.001,y=y.sim)

inits <- list (list(mu=0,r=1,d=10))
```

```
> post<-jags.samples(mcmceg.mod,
+                       var=track.variables
+                       n.iter=10000)
> op<-par(mfrow=c(1,3),pty="s")
> hist(post$d)
> hist(post$mu)
> hist(post$r)
```

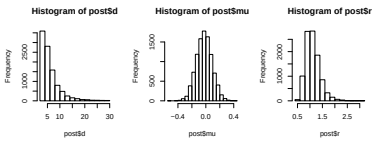


Figure 8.6: Example of MCMC sampling with (intractable) multiparameter models.

```
parameters <- c("mu", "r", "d")

## change this for your system:
mybugsdir<-"C:/Users/Shravan/Desktop/winbugs14/WinBUGS14/"

model.sim <- bugs (data, inits, parameters,
             "WinBUGsmodels/multimcmcexample2.bug",
              n.chains=1, n.iter=10000,debug=TRUE,
              bugs.directory=mybugsdir)

print(model.sim)
plot(model.sim)
```

Here is a comparison of JAGS and WinBUGS output (I ran the latter on Windows):

```
## JAGS:
     Mean    SD Naive SE Time-series SE
mu  0.824 98.57    1.559          1.534
r   1.590 39.39    0.623          0.623
d  16.190  8.42    0.133          0.141

   2.5%   25%       50%       75%    97.5%
mu -190 -63.5  1.61e+00  6.76e+01 1.95e+02
r     0   0.0 1.76e-299 4.46e-128 7.28e-10
d     2   9.0  1.60e+01  2.40e+01 3.00e+01
##WinBUGS:
        mean  sd  2.5%   25%   50%   75% 97.5%
mu       0.0 0.1  -0.2  -0.1   0.0   0.1   0.3
r        0.8 0.2   0.5   0.6   0.8   0.9   1.3
d        9.1 7.0   3.0   4.0   6.0  12.0  28.0
```

9
Priors

We will use either non-informative (diffuse) or informative priors; details coming in the next chapters when we look at psycholinguistic data. In linear mixed models we will use diffuse priors for the fixed effects (which is what we are interested in), but "fairly informative priors" (following Lunn et al, Section 5.1) for random effects. Lunn et al say:

> Often there is limited evidence in the immediate data concerning such parameters and hence there can be considerable sensitivity to the prior distribution, in which case we recommend thinking carefully about reasonable values in advance and so specifying fairly informative priors—the inclusion of such external information is unlikely to bias the main estimates arising from a study, although it may have some influence on the precision of the estimates and this needs to be careully explored through sensitivity analysis.

One rule of thumb seems to be: If you have very little data you probably should not try to use a vague/diffuse prior. When you have little data, your prior is going to dominate the posterior, so you need prior knowledge or expert opinion. By contrast, if you have lots of data, your data will dominate, and the posterior is not going to be too sensitive to the specification of the prior (we will test this later; but I think it's generally true).

9.1 Example illustrating how increasing sample size reduces the impact of the prior

This is an exercise for you. We return to an earlier example (p. 127). Try to obtain the posterior distribution using the following different priors. Summarize the posterior distributions in a way that that the effect of the prior is easy to see; this could be a table or (a set of) figures.

1. $\theta \sim Beta(3,27)$ (this is already given below)

2. $\theta \sim Unif(0,1)$
3. $\theta \sim Beta(0,0)$
4. The so-called Jeffreys prior: $\theta \sim Beta(0.5,0.5)$
5. A logit-normal prior (it's an exercise to figure out how to do this; a hint is that we have seen it before)

```
> ## the data:
> data<-list(a=3,b=27,y=0,n=10,n.pred=20)
> cat("
+ model
+    {
+     ## prior
+     theta ~ dbeta(a,b)
+
+     ## likelihood
+     y ~ dbin(theta,n)
+     ## predicted posterior distribution
+     y.pred ~ dbin(theta,n.pred)
+    }",
+      file="JAGSmodels/predictionexample1.jag" )
> track.variables<-c("y.pred","theta")
> library(rjags)
> system.time(
+ predeg.mod <- jags.model(
+   file = "JAGSmodels/predictionexample1.jag",
+                       data=data,
+                       n.chains = 4,
+                        n.adapt =2000 )
+             )

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 7

Initializing model

   user  system elapsed
  0.001   0.001   0.014

> system.time(
+ predeg.res <- coda.samples( predeg.mod,
+                             var = track.variables,
```

```
+                              n.iter = 10000,
+                              thin = 20 ) )

   user  system elapsed
  0.024   0.000   0.028
```

Next, investigate what happens with each of the priors you specified above when you increase sample size to 1000 (instead of 10), with number of successes equal to 100 rather than 0. What you should find is that the likelihood starts to dominate in determining the posterior, the prior has basically no influence on the posterior when sample size is relatively large.

9.2 Priors for σ^2 of the observable variables

When specifying priors for the observable variable's variance, we are generally going to use either $\tau \sim Gamma(0.001, 0.001)$, where $\tau = \frac{1}{\sigma^2}$, or a Uniform distribution with an appropriate range, such as Unif(0,100). The Gamma makes the prior for σ^2 an inverse gamma. See Lunn et al., page 87[1] for discussion on why.

[1] David Lunn, Chris Jackson, David J Spiegelhalter, Nicky Best, and Andrew Thomas. *The BUGS book: A practical introduction to Bayesian analysis*, volume 98. CRC Press, 2012

9.3 Priors for σ^2 of the random effects variance components

Lunn et al. say that an apparently vague prior for random effects variance components can have undesirable consequences. You can use a Gamma(a,b) with a,b small, or a uniform Unif(-10,10) on the log of the sd, but the choice of parameters can affect the posterior. **Note however that this statement does not seem to be true for standard datasets in psycholinguistics, perhaps because we have so much data.**

Gelman[2] recommends using a **uniform prior** on a plausible range of standard deviation, or a **half-normal with large variance**. When an informative prior is needed, he recommends working with a half-t family of distributions, which includes the half-Cauchy as a special case. The half-Cauchy can be defined indirectly: if z is a random variable with $z \sim N(0, B^2)$, and a random variable γ is chi-squared with df=1 (i.e., γ is Gamma(0.5,0.5)), then $z/\sqrt{\gamma}$ is Cauchy distributed with mean 0 and sd B. This can be defined in BUGS as follows:

[2] Andrew Gelman. Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper). *Bayesian analysis*, 1(3):515–534, 2006

```
omega <- abs(z)/sqrt(gamma)
z ~ dnorm(0,inv.B.squared)
inv.B.squared<- 1/pow(B,2)
gamma ~ dgamma(0.5,0.5)
```

To simplify matters, in this course, we will only use the uniform prior on the random effects variance components. For reading times

on the log scale, I use Unif(0,10), i.e., I am agnostic about the value of the variance components, but I know it's going to be 0 or positive. This makes sense because we will almost never have any prior beliefs about these components (our focus is on the fixed effects).

9.4 Priors for variance-covariance matrices

Suppose we have a repeated measures design with two conditions. Let i range over subjects ($i = 1,\ldots,n$); j ranges over the two conditions ($j = 1,2$); and k range over the items ($k = 1,\ldots,m$).

We are going to specify a model like the familiar one below:

```
lmer(log.rt~condition + (1|subj)+(condition -1 | subj)+(1|item))
```

The specification of this model with varying intercepts for subjects (u_{0i}) and items (w_{0k}), and varying slopes (u_{1i}) by subject (without a correlation component) is as follows.

$$y_{ijk} = (\beta_0 + u_{0i} + w_{0k}) + (\beta_1 + u_{1i}) + e_{ijk} \tag{9.1}$$

where

$$\begin{pmatrix} u_{0i} \\ u_{1i} \end{pmatrix} \sim N(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma^2_{u_{0i}} & \rho\sigma_{u_{0i}}\sigma_{u_{1i}} \\ \rho\sigma_{u_{0i}}\sigma_{u_{1i}} & \sigma^2_{u_{1i}} \end{pmatrix}) \quad \text{and } w_{0k} \sim N(0, \sigma_{w0k}) \tag{9.2}$$

and

$$\varepsilon_{ijk} \sim N(0, \sigma^2) \tag{9.3}$$

In other words, the variance covariance matrix for the **by subjects** varying intercepts is:

$$\Sigma = \begin{pmatrix} \sigma^2_{u_{0i}} & \rho\sigma_{u_{0i}}\sigma_{u_{1i}} \\ \rho\sigma_{u_{0i}}\sigma_{u_{1i}} & \sigma^2_{u_{1i}} \end{pmatrix} \tag{9.4}$$

with $\rho = 0$.

What kind of prior can we define for this vcov matrix? For Ω, the **precision matrix**[3] we can take a Wishart(R,k) prior; R denotes the **inverse scale matrix**, and k the degrees of freedom.[4]

To get a feel for what a distribution for a variance-covariance matrix might be like, recall that we can plot the distribution of a normally distributed random variable by sampling from it:

```
> x<-seq(-4,4,by=0.01)
> plot(x,dnorm(x))
```

Similarly, we can sample from a Wishart distribution, but this time we will get random matrices as output:

[3] Recall that precision is the inverse of variance.

[4] The Wishart distribution is the multivariate analog of the gamma distribution.

```
> library(MCMCpack)
> draw <- rwish(3, matrix(c(1,.3,.3,1),2,2))
```

I have no experience in plotting distributions of covariance matrices,[5] but let me take a shot at it. Run the following code:

[5] See *Visualizing Distributions of Covariance Matrices*, by Tomoki Tokuda, Ben Goodrich, Iven Van Mechelen, Andrew Gelman, Francis Tuerlinckx, for a more authoritative study.

```
> nsim<-1000
> store22<-store21<-store12<-store11<-rep(NA,nsim)
> for(i in 1:nsim){
+         draw <- rwish(3, matrix(c(1,0.9,0.9,1),2,2))
+         store11[i]<-draw[1,1]
+         store12[i]<-draw[1,2]
+         store21[i]<-draw[2,1]
+         store22[i]<-draw[2,2]
+ }
> library(MASS)
> variances<-as.matrix(data.frame(store11,store22))
> bivn.kde<-kde2d(variances[,1],variances[,2],n=nsim)
> persp(bivn.kde,phi=10,theta=0,shade=0.2,border=NA,
+       main="Simulated variances' joint distribution")
```

You can play with the correlation in the off-diagonals and the variances to see what happens to the joint distribution of the variances.

In our example above, the inverse scale matrix R would be:

$$R = \begin{pmatrix} \tau_{subjintercept} & 0 \\ 0 & \tau_{subjslope} \end{pmatrix} \tag{9.5}$$

and $k = 2$.

The way we will define this in JAGS is, first we create the zero matrix for the means, and the R matrix, and then generate varying intercepts and slopes (for subjects) from a multivariate normal.

```
## partial BUGS code focusing on the by-subject
## variance component priors:
data
   {
   zero[1] <- 0
   zero[2] <- 0
   R[1,1] <- 0.1
   R[1,2] <- 0
   R[2,1] <- 0
   R[2,2] <- 0.5
   }
model
   {
```

```
# Intercept and slope for each person, including random effects
  for( i in 1:I )
  {
  u[i,1:2] ~ dmnorm(zero,Omega.u)
  }

# Define prior for the variance-covariance matrix of the
# random effects for subjects
  Sigma.u <- inverse(Omega.u)
  Omega.u  ~ dwish( R, 2 )
```

Another way to define this model would be to simply define a prior for each variance component, without defining any prior for the variance-covariance matrix. A fully worked example of the above case will appear in the chapter on linear mixed models.

9.4.1 Priors for variance-covariance matrices with dimensions greater than 2x2

[I'm grateful to Sergio Polini for helping me understand this point. Also see Gelman's blog entry: http://andrewgelman.com/2012/08/29/more-on-scaled-inverse-wishart-and-prior-independence/]

Defining priors for $n \times n, n > 2$, covariance matrices is difficult because of constraints among the correlation parameters. So you have to define a prior for a large covariance matrix in one go.

One option is to use the Wishart distribution as we did above, but its parameters are difficult to interpret and a single parameter controls the precision of all elements of the covariance matrix.[6] An alternative, proposed by Barnard, McCulloch and Meng[7] is to use a separation strategy, $\Sigma = DRD$, where D is a diagonal matrix of standard deviations and R is a correlation matrix. This allows you to express beliefs about the variance components in D while remaining agnostic about beliefs regarding the correlation matrix.

To model a correlation matrix Q, O'Malley and Zaslavsky suggest a scaled inverse-Wishart:

$\Sigma = diag(xi)Qdiag(xi), Q \sim Inv - Wishart(...)$

If we set degrees of freedom to n+1, we get a marginal uniform distribution for the correlations, then "correct" the constrained variances by a vector xi of scale parameters. See Gelman & Hill, pp. 286-287.

Lewandowski, Kurowicka and Joe[8] found efficient algorithms to generate random correlation matrices whose distribution depends on a single η parameter. If $\eta = 1$ then their distribution is jointly uniform. If η is greater than 1, then the distribution is concentrated around the identity matrix, which has 1's in the diagonals and 0's in the off-diagonals (recall that it's a multivariate distribution), and

[6] A. James O'Malley and Alan M. Zaslavsky. Domain-level covariance analysis for multilevel survey data with structured nonresponse. *Journal of the American Statistical Association*, 103:1405–1418, 2008

[7] John Barnard, Robert McCulloch, and Xiao-Li Meng. Modeling covariance matrices in terms of standard deviations and correlations, with application to shrinkage. *Statistica Sinica*, 10:1281–1311, 2000

[8]

as η increases, the distribution becomes more sharply concentrated around the identity matrix. If η lies between 0 and 1, then there is a trough at the identity matrix. This makes the separation strategy proposed by Barnard, McCulloch and Meng efficient. Stan implements this algorithm; in Stan, it is called lkj_corr. We will use this later in the chapter on linear mixed models.

9.4.2 Visualizing the lkj_corr matrix

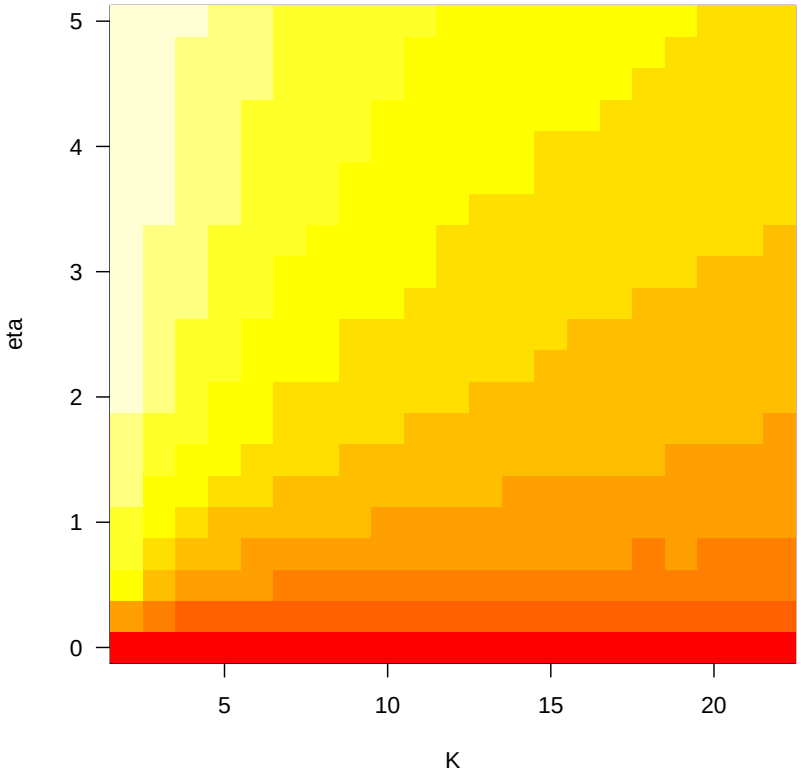
I quote verbatim from Ben Goodrich's response to a question from me about how to visualize the correlation matrix.

The following code

> "randomly draws from the LKJ distribution (quickly).
>
> When the shape parameter is not equal to 1, then the correlations do not all have the same marginal distribution, unless you do a symmetric permutation of the rows and columns of the correlation matrix. But a marginal correlation is a red herring anyway.
>
> In my opinion, the thing to visualize is "effective independence" — which is the determinant of the correlation matrix raised to the power of $1/K$ or equivalently the geometric mean of the eigenvalues — as K and the shape parameter varies. So, something like:

```
> ## this is a file provided by Goodrich:
> source("lkj.R")
> stopifnot(require(parallel))
> K <- 2:22
> eta <- seq(from = 0, to = 5, length.out = 21)
> eg <- expand.grid(K = K, eta = eta)
> ei_lkj <- function(K, eta, SIMS = 1000) {
+   mean(replicate(SIMS,
+                  exp(2 * mean(log(diag(rcorvine(K, eta, cholesky = TRUE)))))) )
+ }
> theta_lkj <- mcmapply(ei_lkj, K = eg$K, eta = eg$eta, mc.cores = detectCores())
> par(mar = c(5,4,1,1) + .1)
> image(K, eta, matrix(theta_lkj, length(K), length(eta)), las = 1)
> contour(K, eta, matrix(theta_lkj, length(K), length(eta)), las = 1)
```

Figure 9.1: Visualizing the LKJ correlation matrix, code by Ben Goodrich.



Homework 7

How would you define a prior for a model with an intercept-slope correlation for subjects random effects? Write down the model mathematically, and then the code in BUGS syntax. The key point here is that you need to define a prior for ρ.

In R we would write:

```
lmer(log.rt~condition + (1+condition | subj)+(1|item))
```

How would you define a random intercepts and slopes model for item? In R this would be:

```
lmer(log.rt~condition + (1+condition | subj)+(1+condition|item))
```

9.5 Mixture of priors

When we have conflicting opinions about priors, we can include competing priors and let the data decide which one is better.

We return to the example on page 126. Suppose there are two possible priors, Beta(3,27) and Beta(3,3), and the probability of each prior (relative belief) being the appropriate one is 0.5. Suppose we observe 2 successes out of 5. What is the better prior for 10 future observations? Intuitively, it's going to be Beta(3,3) (because it represents a success probability of 2/5=0.4 better than a Beta(3,27), which represents a success probability of 3/27.

Note: The `dcat` function in JAGS below represents the distribution $\frac{\pi_x}{\sum pi_i}$.

```
> ## the data:
> data<-list(## current data:
+            y=2,n=5,
+             ## future data sample size:
+             m=10,
+            ## possible priors:
+             a=c(3,3),b=c(27,3))
> cat("
+ model
+    {
+     ## prior
+     theta ~ dbeta(a[pick],b[pick])
+     pick ~ dcat(q[1:2])
+     q[1] <- 0.5
+     q[2] <- 0.5
+     ## 1 if prior 1 is picked:
+     q.post[1] <- equals(pick,1)
```

```
+     q.post.1 <- q.post[1]
+     ## 1 if prior 2 is picked:
+     q.post[2] <- equals(pick,2)
+     q.post.2 <- q.post[2]
+     ## likelihood:
+     y ~ dbin(theta,n) ## sampling distrn
+     ## predicted posterior distribution
+     y.pred ~ dbin(theta,m) ## predictive distrn
+    }",
+      file="JAGSmodels/mixtureexample1.jag" )
> track.variables<-c("y.pred","theta","q.post.1","q.post.2")
> library(rjags)
> system.time(
+ mixtureeg.mod <- jags.model(
+   file = "JAGSmodels/mixtureexample1.jag",
+                       data=data,
+                       n.chains = 4,
+                        n.adapt =2000 )
+             )

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 19

Initializing model

   user  system elapsed
  0.005   0.003   0.022

> system.time(
+ mixtureeg.res <- coda.samples( mixtureeg.mod,
+                               var = track.variables,
+                               n.iter = 10000,
+                               thin = 20 ) )

   user  system elapsed
  0.175   0.004   0.259

> summary(mixtureeg.res)

Iterations = 20:10000
Thinning interval = 20
Number of chains = 4
Sample size per chain = 500
```

```
1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean    SD Naive SE Time-series SE
q.post.1 0.258 0.438  0.00979        0.00979
q.post.2 0.742 0.438  0.00979        0.00979
theta    0.378 0.189  0.00422        0.00436
y.pred   3.746 2.332  0.05214        0.05613

2. Quantiles for each variable:

          2.5%   25%   50%   75% 97.5%
q.post.1 0.000 0.000 0.000 1.000 1.000
q.post.2 0.000 0.000 1.000 1.000 1.000
theta    0.071 0.209 0.388 0.519 0.733
y.pred   0.000 2.000 4.000 5.000 9.000
```

```
> plot(mixtureeg.res)
```

The summary of the posterior distribution $p(\theta)$ and predicted values:

```
> summary( mixtureeg.res )[1]

$statistics
            Mean      SD  Naive SE Time-series SE
q.post.1 0.25800 0.43764 0.0097860      0.0097895
q.post.2 0.74200 0.43764 0.0097860      0.0097895
theta    0.37768 0.18883 0.0042224      0.0043581
y.pred   3.74600 2.33157 0.0521354      0.0561338
```

The relative probabilities of q.post.1 and q.post.2 show that prior 2 (Beta(3,3)) is more appropriate given the present data, 2 successes out of 5. The posterior distribution of a future dataset with sample size 10 is represented by y.pred.

Homework 8

Suppose our data was 1 success out of 10. For the same problem as above (predicting the posterior distribution for a future 10 measurements), which of the two priors, Beta(3,27) and Beta(3,3), is more appropriate?

9.6 Univariate conjugate prior distributions

Here is a listing of some conjugate priors,taken from Lunn et al.

Univariate conjugate prior distributions for various one-parameter likelihoods from a sample of size n. Also given are the corresponding posterior parameters and the predictive distribution for a single new observation $\tilde{y}^{\dagger}$. See Appendix C and/or Bernardo and Smith (1994), pp. 427–435, for definitions of distributions.

Sampling distribution	**Conjugate prior**	**Posterior parameters**	**Predictive distribution**
$y\|\theta \sim \text{Binomial}(\theta, n)$ including Bernoulli ($n = 1$)	$\theta \sim \text{Beta}(a, b)$	$a_n = a + y,$ $b_n = b + n - y$	Beta-Binomial(a_n, b_n, n)
$y\|\mu \sim \prod_{i=1}^{n} \text{Normal}(\mu, \sigma^2)$	$\mu \sim \text{Normal}(\gamma, \omega^2 = \frac{\sigma^2}{n_0})$	$\gamma_n = \frac{n_0\gamma + n\bar{y}}{n_0 + n},$ $\omega_n^2 = \frac{\sigma^2}{n_0 + n}$	Normal$(\gamma_n, \omega_n^2 + \sigma^2)^{\ddagger}$
$y\|\sigma^2 \sim \prod_{i=1}^{n} \text{Normal}(\mu, \sigma^2)$	$\sigma^{-2} \sim \text{Gamma}(a, b)$	$a_n = a + \frac{n}{2},$ $b_n = b + \frac{1}{2}\sum_i (y_i - \mu)^2$	Student-$t(\mu, \frac{b_n}{a_n}, 2a_n)^{\S}$
$y\|\theta \sim \prod_{i=1}^{n} \text{Poisson}(\theta)$	$\theta \sim \text{Gamma}(a, b)$	$a_n = a + n\bar{y},$ $b_n = b + n$	NegBin$(\frac{b_n}{b_n+1}, a_n)$
$y\|\theta \sim \prod_{i=1}^{n} \text{Gamma}(\alpha, \theta)$ including Exponential ($\alpha = 1$)	$\theta \sim \text{Gamma}(a, b)$	$a_n = a + n\alpha,$ $b_n = b + n\bar{y}$	Gamma-Gamma(a_n, b_n, α)
$y\|\theta \sim \prod_{i=1}^{n} \text{Uniform}(0, \theta)$	$\theta \sim \text{Pareto}(a, b)$	$a_n = a + n,$ $b_n = \max\{b, y\}$	$\begin{cases} \frac{a_n}{a_n+1}\text{Uniform}(0, b_n), & \tilde{y} \leq b_n \\ \frac{1}{a_n+1}\text{Pareto}(a_n, b_n), & \tilde{y} > b_n \end{cases}$
$y\|\theta \sim \text{NegBin}(\theta, r)$ including Geometric ($r = 1$)	$\theta \sim \text{Beta}(a, b)$	$a_n = a + r,$ $b_n = b + y$	Negative-Binomial-Beta(a_n, b_n, r_p)

10
Regression models

This section is based on Lunn et al.[1]

We begin with a simple example. Let the response variable be $y_i, i = 1,\ldots,n$, and let there be p predictors, $x_{1i},\ldots,x_{pi}$. Also, let

$$y_i \sim N(\mu_i, \sigma^2), \quad \mu_i = \beta_0 + \sum \beta x_{ki} \tag{10.1}$$

(the summation is over the p predictors, i.e., $k = 1,\ldots,p$).

We need to specify a prior distribution for the parameters:

$$\beta_k \sim Normal(0, 100^2) \quad \log\sigma \sim Unif(-100, 100) \tag{10.2}$$

Recall that one should do a sensitivity analysis, and if the choice of vague prior is influential, then we should be looking at more informative priors based on prior work.

Lunn et al advise us to center our predictors because this reduces the posterior correlation between each coefficient (the slopes) and the intercept (because the intercept is relocated to the center of the data). Lunn et al say that high levels of posterior correlation are problematic for Gibbs sampling.

Let's take the beauty data as an example. We saw this data-set (taken from Gelman and Hill,[2] if I recall correctly) in the introductory course: nicer looking professors get better teaching evaluations. The predictor is already centered. Notice that we literally follow the specification of the linear model given above. We specify the model for the data frame row by row, using a for loop, so that for each dependent variable value y_i (the evaluation score) we specify how it was generated.

Note that I have to set priors on σ in order to "recover" the estimates that the lm function delivers in R for σ.

```
> beautydata<-read.table("data/beauty.txt",header=T)
> #summary(fm<-lm(evaluation~beauty,beautydata))
>
```

[1] David Lunn, Chris Jackson, David J Spiegelhalter, Nicky Best, and Andrew Thomas. *The BUGS book: A practical introduction to Bayesian analysis*, volume 98. CRC Press, 2012

[2] A. Gelman and J. Hill. *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press, Cambridge, UK, 2007

```
> ## restate the data as a list for JAGS:
> data<-list(x=beautydata$beauty,y=beautydata$evaluation)
> cat("
+ model
+    {
+     ## specify model for data:
+     for(i in 1:463){
+     y[i] ~ dnorm(mu[i],tau)
+     mu[i] <- beta0 + beta1 * (x[i])
+     ## residuals: to-do
+     ##res[i] <- y[i] - mu[i]
+     }
+     # priors:
+     beta0 ~ dunif(-10,10)
+     beta1 ~ dunif(-10,10)
+     tau <- 1/sigma2
+     sigma2<-pow(sigma,2)
+     #log(sigma2) <- 2* log.sigma
+     #log.sigma ~ dunif(-10,10)
+     sigma ~ dunif(0,100)
+    }",
+      file="JAGSmodels/beautyexample1.jag" )
> track.variables<-c("beta0","beta1","sigma")
> library(rjags)
> inits <- list (list(beta0=0,beta1=0))
> beauty.mod <- jags.model(
+   file = "JAGSmodels/beautyexample1.jag",
+                       data=data,
+                       inits=inits,
+                       n.chains = 1,
+                        n.adapt =10000 , quiet=T)
> beauty.res <- coda.samples( beauty.mod,
+                                  var = track.variables,
+                                n.iter = 10000,
+                                  thin = 20 )
> summary(beauty.res)

Iterations = 10020:20000
Thinning interval = 20
Number of chains = 1
Sample size per chain = 500

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
```

```
       Mean     SD Naive SE Time-series SE
beta0 4.008 0.0257 0.001148       0.001159
beta1 0.134 0.0318 0.001422       0.001328
sigma 0.547 0.0169 0.000755       0.000817

2. Quantiles for each variable:

        2.5%   25%   50%   75% 97.5%
beta0 3.9611 3.988 4.007 4.025 4.058
beta1 0.0739 0.112 0.133 0.154 0.200
sigma 0.5161 0.536 0.548 0.558 0.582
```

```
> densityplot(beauty.res)
```

In the BUGS book by Lunn et al., they give a smaller dataset as an example, involving five measurements of a rat's weight, in grams, as a function of some x (not sure what the predictor is). Note that here the centering of the predictor happens in the model code.

First we load/enter the data:

```
> data<-list(x=c(8,15,22,29,36),y=c(177,236,285,350,376))
```

Then we fit the linear model using `lm`, for comparison with the bayesian model:

```
> summary(fm<-lm(y~x,data))

Call:
lm(formula = y ~ x, data = data)

Residuals:
    1     2     3     4     5
 -5.4   2.4   0.2  14.0 -11.2

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  123.886     11.879    10.4  0.00188 **
x              7.314      0.492    14.8  0.00066 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Residual standard error: 10.9 on 3 degrees of freedom
Multiple R-squared:  0.987,        Adjusted R-squared:  0.982
F-statistic:  221 on 1 and 3 DF,  p-value: 0.000662
```

Then define the bayesian model; note that I put a prior on σ, not σ^2.

```
> cat("
+ model
+    {
+     ## specify model for data:
+     for(i in 1:5){
+     y[i] ~ dnorm(mu[i],tau)
+     mu[i] <- beta0 + beta1 * (x[i]-mean(x[]))
+     }
+     # priors:
+     beta0 ~ dunif(-500,500)
+     beta1 ~ dunif(-500,500)
+     tau <- 1/sigma2
+     sigma2 <-pow(sigma,2)
+     sigma ~ dunif(0,200)
+    }",
+      file="JAGSmodels/ratsexample2.jag" )
> track.variables<-c("beta0","beta1","sigma")
> library(rjags)
> inits <- list (list(beta0=10,beta1=10,sigma=10))
> rats.mod <- jags.model(
+   file = "JAGSmodels/ratsexample2.jag",
+                       data=data,
+                       inits=inits,
+                       n.chains = 1,
+                        n.adapt =10000 , quiet=T)
> rats.res <- coda.samples( rats.mod,
+                                  var = track.variables,
+                                n.iter = 10000,
+                                  thin = 20 )
> summary(rats.res)

Iterations = 10020:20000
Thinning interval = 20
Number of chains = 1
Sample size per chain = 500

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

        Mean   SD Naive SE Time-series SE
beta0 285.03 17.1    0.766         0.9505
beta1   7.29  1.7    0.076         0.0596
sigma  23.45 22.5    1.004         1.5948
```

```
2. Quantiles for each variable:

        2.5%    25%    50%    75% 97.5%
beta0 260.16 279.61 284.39 289.89 312.8
beta1   4.49   6.77   7.29   7.78  10.5
sigma   7.02  11.67  16.36  25.42  83.2
```

```
> densityplot(rats.res)
```

This model gets pretty similar summaries compared to the lm function in R (check this). The estimate for sd is a bit high compared to the lm model fit, which means that the standard errors in the bayesian model will be larger than in the lm fit.

Homework 9

For the above example, first, put a suitable uniform prior on σ^2. Compare the posterior summary with the linear model output in R.

Next, put a uniform prior on log σ^2. Here's a hint:

```
log(sigma2) <- 2* log.sigma
## need some numerical values in place of xx
log.sigma ~ dunif(xx,xx) ## complete this and run model
```

What changes in our estimates of σ compared to the first case above?

10.1 Multiple regression

This is the lexical decision dataset from Baayen's book.[3] We fit log reading time to Trial id (centered), Native Language, and Sex. The categorical variables are centered as well.

[3] R. H. Baayen. *Practical data analysis for the language sciences*. Cambridge University Press, 2008

```
> library(languageR)
> data<-lexdec[,c(1,2,3,4,5,9)]
> contrasts(data$NativeLanguage)<-contr.sum(2)
> contrasts(data$Sex)<-contr.sum(2)
> summary(fm<-lm(RT~scale(Trial,scale=F)+NativeLanguage+Sex,data))

Call:
lm(formula = RT ~ scale(Trial, scale = F) + NativeLanguage +
    Sex, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-0.5702 -0.1541 -0.0275  0.1140  1.1356

Coefficients:
```

```
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)              6.407285   0.006036 1061.53   <2e-16 ***
scale(Trial, scale = F) -0.000283   0.000118   -2.39    0.017 *
NativeLanguage1         -0.083705   0.005750  -14.56   <2e-16 ***
Sex1                    -0.030711   0.006036   -5.09    4e-07 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Residual standard error: 0.227 on 1655 degrees of freedom
Multiple R-squared:  0.119,        Adjusted R-squared:  0.117
F-statistic: 74.4 on 3 and 1655 DF,  p-value: <2e-16
```

The JAGS model:

```
> dat<-list(y=data$RT,
+           Trial=(data$Trial-mean(data$Trial)),
+           Lang=ifelse(data$NativeLanguage=="English",1,-1),
+           Sex=ifelse(data$NativeLanguage=="F",1,-1))
> cat("
+ model
+    {
+     ## specify model for data:
+     for(i in 1:1659){
+     y[i] ~ dnorm(mu[i],tau)
+     mu[i] <- beta0 +
+              beta1 * Trial[i]+
+              beta2 * Lang[i] +  beta3 * Sex[i]
+     }
+     # priors:
+     beta0 ~ dunif(-10,10)
+     beta1 ~ dunif(-5,5)
+     beta2 ~ dunif(-5,5)
+     beta3 ~ dunif(-5,5)
+
+     tau <- 1/sigma2
+     sigma2 <-pow(sigma,2)
+     sigma ~ dunif(0,200)
+    }",
+      file="JAGSmodels/multregexample1.jag" )
> track.variables<-c("beta0","beta1",
+                    "beta2","beta3","sigma")
> library(rjags)
> inits <- list (list(beta0=0,
+                     beta1=0,
+                     beta2=0,beta3=0,sigma=10))
```

```
> lexdec.mod <- jags.model(
+   file = "JAGSmodels/multregexample1.jag",
+                       data=dat,
+                       inits=inits,
+                       n.chains = 1,
+                        n.adapt =10000 , quiet=T)
> lexdec.res <- coda.samples( lexdec.mod,
+                                  var = track.variables,
+                                n.iter = 10000,
+                                  thin = 20 )
> summary(lexdec.res)

Iterations = 10020:20000
Thinning interval = 20
Number of chains = 1
Sample size per chain = 500

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

           Mean       SD Naive SE Time-series SE
beta0  4.265138 0.250538 1.12e-02       1.46e-01
beta1 -0.000277 0.000117 5.22e-06       5.22e-06
beta2 -0.077789 0.005387 2.41e-04       2.21e-04
beta3 -2.131421 0.250295 1.12e-02       1.43e-01
sigma  0.228733 0.004150 1.86e-04       1.86e-04

2. Quantiles for each variable:

           2.5%       25%      50%     75%     97.5%
beta0  3.935429  4.061238  4.23483  4.3691  4.87e+00
beta1 -0.000513 -0.000354 -0.00027 -0.0002 -6.13e-05
beta2 -0.087886 -0.081258 -0.07775 -0.0744 -6.67e-02
beta3 -2.462409 -2.333245 -2.16195 -2.0303 -1.53e+00
sigma  0.220899  0.225813  0.22849  0.2315  2.37e-01
```

Here is the lm fit for comparison (I center all predictors):

```
> contrasts(lexdec$NativeLanguage)<-contr.sum(2)
> contrasts(lexdec$Sex)<-contr.sum(2)
> summary(fm<-lm(RT~scale(Trial,scale=F)+
+                  NativeLanguage+Sex,
+                lexdec))

Call:
lm(formula = RT ~ scale(Trial, scale = F) + NativeLanguage +
```

```
    Sex, data = lexdec)

Residuals:
    Min      1Q  Median      3Q     Max
-0.5702 -0.1541 -0.0275  0.1140  1.1356

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)               6.407285   0.006036 1061.53   <2e-16 ***
scale(Trial, scale = F) -0.000283   0.000118   -2.39    0.017 *
NativeLanguage1         -0.083705   0.005750  -14.56   <2e-16 ***
Sex1                    -0.030711   0.006036   -5.09    4e-07 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Residual standard error: 0.227 on 1655 degrees of freedom
Multiple R-squared:  0.119,        Adjusted R-squared:  0.117
F-statistic: 74.4 on 3 and 1655 DF,  p-value: <2e-16
```

Note: We should have fit a linear mixed model here; I will return to this later.

10.2 Generalized linear models

We consider the model

$$y_i \sim Binomial(p_i, n_i) \quad logit(p_i) = \beta_0 + \beta_1(x_i - \bar{x}) \tag{10.3}$$

For example, beetle data from Dobson et al.[4]:

[4] Annette J Dobson. *An introduction to generalized linear models*. CRC press, 2010

```
> beetledata<-read.table("data/beetle.txt",header=T)
```

The JAGS model:

```
> dat<-list(x=beetledata$dose-mean(beetledata$dose),
+           n=beetledata$number,
+           #p=beetledata$killed/beetledata$number,
+           y=beetledata$killed)
> cat("
+ model
+    {
+ for(i in 1:8){
+     y[i] ~ dbin(p[i],n[i])
+     logit(p[i]) <- beta0 + beta1 * x[i]
+     ## for assessing fitted values:
+     #p.hat[i]<-y[i] / n[i]
```

```
+     #y.hat[i]<-n[i] * p[i]
+ }
+     # priors:
+     beta0 ~ dunif(0,pow(100,2))
+     beta1 ~ dunif(0,pow(100,2))
+    }",
+      file="JAGSmodels/glmexample1.jag" )
> track.variables<-c("beta0","beta1")
> library(rjags)
> inits <- list (list(beta0=0,
+                     beta1=0))
> glm.mod <- jags.model(
+   file = "JAGSmodels/glmexample1.jag",
+                      data=dat,
+                      inits=inits,
+                      n.chains = 1,
+                       n.adapt =10000 , quiet=T)
> glm.res <- coda.samples( glm.mod,
+                                 var = track.variables,
+                               n.iter = 10000,
+                                 thin = 20 )
> summary(glm.res)

Iterations = 10020:20000
Thinning interval = 20
Number of chains = 1
Sample size per chain = 500

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

       Mean    SD Naive SE Time-series SE
beta0  0.75 0.144  0.00642        0.00642
beta1 34.72 2.928  0.13093        0.14173

2. Quantiles for each variable:

        2.5%    25%    50%    75% 97.5%
beta0  0.492  0.646  0.748  0.856  1.03
beta1 29.505 32.671 34.609 36.683 40.68

> plot(glm.res)
```

The values match up with glm output:

```
> coef(glm(killed/number~scale(dose,scale=F),
```

```
+          weights=number,
+            family=binomial(),beetledata))

          (Intercept) scale(dose, scale = F)
               0.7438                34.2703
```

Homework 10

Fit the beetle data again, using suitable normal distribution priors for the coefficients beta0 and beta1. Is the posterior distribution sensitive to the prior?

There is actually much more to say here, because we can fit a lot of other models with different link functions, but for psycholinguistic work we mostly stay with the logit. If there is interest I will give a lecture on GLMs in general, but for the most common cases in psycholinguistics we won't need that knowledge. See my GLM notes (Github) for more detail.

10.3 Prediction

One interesting thing we can do is to predict the posterior distribution of future or missing data. One easy to way to this is to define a node for the predictor, as shown below. This example revisits the earlier toy example from Lunn et al. on the rat's data (page 147).

```
> data<-list(x=c(8,15,22,29,36),y=c(177,236,285,350,376))
> cat("
+ model
+    {
+     ## specify model for data:
+     for(i in 1:5){
+     y[i] ~ dnorm(mu[i],tau)
+     mu[i] <- beta0 + beta1 * (x[i]-mean(x[]))
+     }
+     ## prediction
+     mu45 <- beta0+beta1 * (45-mean(x[]))
+     y45 ~ dnorm(mu45,tau)
+     # priors:
+     beta0 ~ dunif(-500,500)
+     beta1 ~ dunif(-500,500)
+     tau <- 1/sigma2
+     sigma2 <-pow(sigma,2)
+     sigma ~ dunif(0,200)
+    }",
+      file="JAGSmodels/ratsexample2pred.jag" )
> track.variables<-c("beta0","beta1","sigma","mu45")
```

```
> library(rjags)
> inits <- list (list(beta0=10,beta1=10,sigma=10))
> rats.mod <- jags.model(
+   file = "JAGSmodels/ratsexample2pred.jag",
+                       data=data,
+                       inits=inits,
+                       n.chains = 1,
+                        n.adapt =10000, quiet=T)
> rats.res <- coda.samples( rats.mod,
+                                  var = track.variables,
+                                n.iter = 10000,
+                                  thin = 20 )
> summary(rats.res)

Iterations = 10020:20000
Thinning interval = 20
Number of chains = 1
Sample size per chain = 500

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

        Mean    SD Naive SE Time-series SE
beta0 285.68 12.57   0.5624         0.6101
beta1   7.38  1.15   0.0516         0.0468
mu45  455.38 30.72   1.3739         1.3739
sigma  20.20 13.82   0.6182         0.5583

2. Quantiles for each variable:

        2.5%    25%    50%    75%  97.5%
beta0 263.22 280.20 285.49 290.45 311.19
beta1   5.31   6.83   7.33   7.85   9.95
mu45  401.22 441.36 454.15 466.89 519.09
sigma   6.72  11.25  16.39  23.59  59.33

> densityplot(rats.res)
```

11
Linear mixed models

We start with a simple example, the analysis of Gibson and Wu's 2012 dataset on Chinese relative clauses.[1] This data will be one of our running examples in this course.

[1] E. Gibson and I. Wu. Processing Chinese relative clauses in context. Language and Cognitive Processes (in press), 2011

11.1 Linear mixed models analysis using lmer

First we load the data and set things up for analysis.

```
> expt1 <- read.table("data/example1spr.txt")
> colnames(expt1) <- c("subj","item","type",
+                      "pos","word", "correct","rt")
> data<-expt1[ , c(1, 2, 3, 4, 6, 7)]
> ## rename data frame
> dat<-data
>
> #head(dat)
>
> ## 37 subjects
> #length(unique(dat$subj))
> #xtabs(~subj+item,dat)
> #length(sort(unique(dat$subj)))
```

Next we do some pre-processing to get ready for analysis:

We can look at the distributions of reading time in each region of interest:

```
> library(lattice)
> bwplot(rt~type2,subset(critdata,region=="headnoun"),
+        xlab="relative clause type",ylab="reading time (ms)")
> bwplot(log(rt)~type2,subset(critdata,region=="headnoun"),
+        xlab="relative clause type",ylab="log reading time (log ms)")
> bwplot(-1000/rt~type2,subset(critdata,region=="headnoun"),
+        xlab="relative clause type",ylab="negative reciprocal reading time (-1/sec)")
```

Set up the contrast coding:

```
> library(lme4)
> library(car)
> library(ggplot2)
> ## treatment contrasts should be OK for analysis...
> #contrasts(critdata$type)
>
> ## but let's do a regular anova style centering, sum contrasts:
> critdata$so<- ifelse(critdata$type%in%c("subj-ext"),-0.5,0.5)
```

The Box-Cox transform suggests using the inverse for the head noun and the region after:

```
> par( mfrow=c(2,2) )
> library(MASS)
> boxcox(rt~type*subj,
+        data=critdata[critdata$region=="de1", ])
> boxcox(rt~type*subj,
+        data=critdata[critdata$region=="de", ])
> boxcox(rt~type*subj,
+        data=critdata[critdata$region=="headnoun", ])
> boxcox(rt~type*subj,
+        data=critdata[critdata$region=="headnoun1", ])
> ## transform:
> critdata$rrt <- -1000/critdata$rt
> means.rrt<-round(with(critdata,
+                       tapply(rrt,IND=list(region,type),mean)),
+                  digits=3)
> means.rt<-round(with(critdata,
+                      tapply(rt,IND=list(region,type),mean)),
+                 digits=0)
> library(xtable)
> #xtable(cbind(means.rt,means.rrt))


> boxplot(rrt~type,subset(critdata,region=="de1"))
> boxplot(rrt~type,subset(critdata,region=="de"))
> boxplot(rrt~type,subset(critdata,region=="dehnoun"))
> boxplot(rrt~type,subset(critdata,region=="headnoun"))
> boxplot(rrt~type,subset(critdata,region=="headnoun1"))
```

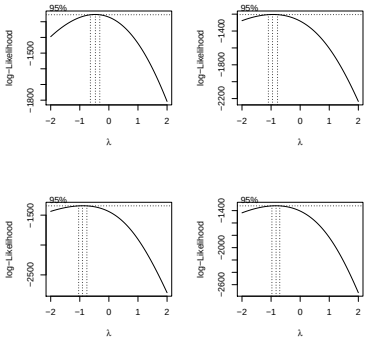


Figure 11.1: The results of the Box-Cox procedure.

We have predictions for the head noun and the word after that, but with a variance stabilizing transform (reciprocal RT) these are not borne out, cf. the published paper's results based on raw RTs (also see below). For more detail see Vasishth et al 2013.[2]

[2] Shravan Vasishth, Zhong Chen, Qiang Li, and Gueilan Guo. Processing Chinese relative clauses: Evidence for the subject-relative advantage. *PLoS ONE*, 8(10):e77006, 10 2013

```
> ## no effect at headnoun.
> ## cf. spurious effect at head noun in raw rts:
> gw.hn.rrt <- lmer(rrt~so+(1|item)+(1+so|subj),
+                   subset(critdata,region=="headnoun"))
> summary(gw.hn.rrt)

Linear mixed model fit by REML ['lmerMod']
Formula: rrt ~ so + (1 | item) + (1 + so | subj)
   Data: subset(critdata, region == "headnoun")

REML criterion at convergence: 1596.7

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 subj     (Intercept) 0.3717   0.610
          so          0.0526   0.229    -0.51
 item     (Intercept) 0.1098   0.331
 Residual             0.8941   0.946
Number of obs: 547, groups: subj, 37; item, 15

Fixed effects:
            Estimate Std. Error t value
(Intercept)  -2.6722     0.1379  -19.37
so           -0.0762     0.0895   -0.85

Correlation of Fixed Effects:
   (Intr)
so -0.158

> headnoun.dat<-subset(critdata,region=="headnoun")
> gw.hn.rt <- lmer(rt~so+(1|item)+(1+so|subj),
+                   headnoun.dat)
> gw.hn.lrt <- lmer(log(rt)~so+(1|item)+(1+so|subj),
+                   headnoun.dat)
> #summary(gw.hn.rt)
>
> gw.hn.rt.trimmed <- lmer(rt~so+(1|item)+(1+so|subj),
+                   headnoun.dat[abs(scale(residuals(gw.hn.rt)))<2.5,]
+ )
> #summary(gw.hn.rt.trimmed)
>
> ## check how many data points we have by subject and item:
> #xtabs(~subj+so,subset(critdata,region=="headnoun"))
> #xtabs(~item+so,subset(critdata,region=="headnoun"))
>
```

```
> ## contrast coding:
> #critdata$so<- ifelse(critdata$type%in%c("subj-ext"),-0.5,0.5)
>
> gw.hn.rt <- lmer(rt~so+(1+so|item)+(1+so|subj),
+                  subset(critdata,region=="headnoun"))
> gw.hn.logrt <- lmer(log(rt)~so+(1+so|item)+(1+so|subj),
+                     subset(critdata,region=="headnoun"))
> gw.hn.rrt <- lmer(-1000/rt~so+(1+so|item)+(1+so|subj),
+                   subset(critdata,region=="headnoun"))
```

Classical aggregated analysis assuming non-equal variance:

```
> hnoun<-subset(critdata,region=="headnoun")
> ## classical aggregrated analysis:
> hnoun.subj.means<-aggregate(rt~subj+type,
+                             mean,data=hnoun)
> t.test(subset(hnoun.subj.means,type=="subj-ext")$rt,
+        subset(hnoun.subj.means,type=="obj-ext")$rt,paired=T,var.equal=FALSE)

        Paired t-test

data:  subset(hnoun.subj.means, type == "subj-ext")$rt and subset(hnoun.subj.means, type == "obj-ext")$r
t = 2.6301, df = 36, p-value = 0.01248
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  28.199 218.207
sample estimates:
mean of the differences
                  123.2

> ## examine residuals:
> library(car)
> op<-par(mfrow=c(1,3),pty="s")
> qqPlot(residuals(gw.hn.rt),
+        ylab="residuals of raw reading times")
> qqPlot(residuals(gw.hn.logrt),
+        ylab="residuals of log reading times")
> qqPlot(residuals(gw.hn.rrt),
+        ylab="residuals of negative reciprocal reading times")
```

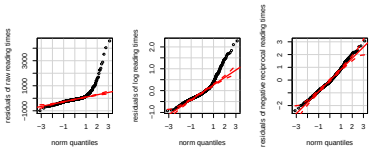


Figure 11.2: Residuals in lmer models with raw RTs, log RTs, and negative reciprocals.

On this raw reading time scale, the differences in rt are about 178 ms at the head noun (OR advantage):

```
> means<-with(critdata,tapply(rt,IND=list(region,type),mean))
```

However, standard deviation is not comparable:

```
> sds<-with(critdata,tapply(rt,IND=list(region,type),sd))
```

At the head noun, the ratio of variances is:

```
> round(sds[4,2]/sds[4,1],digits=1)
```

```
[1] 2.2
```

Note that Gibson and Wu fit raw reading times, and got significant effects (OR advantage). Here is an lmer fit analogous (but not identical) to what they did:

```
> ##head noun:
> gw.hn <- lmer(rt~so+(1|item)+(1|subj),subset(critdata,region=="headnoun"))
```

The model estimates that ORs are about 120 ms easier to process than SRs at the head noun.

However, statistical significance here is a consequence of the normality assumption (of residuals) not being satisfied; I think that, more precisely, it's the equal variance assumption that's an issue (SR variance is much higher due to those extreme values).

```
> qqPlot(residuals(gw.hn))
```

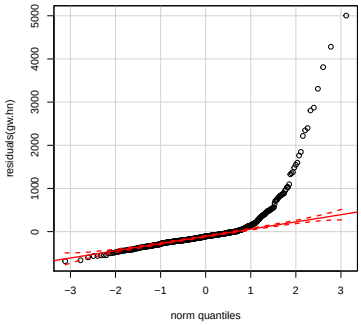


Figure 11.3: Residuals in the model using raw reading times.

Compare this with the reciprocal rt's residuals.

In the remaining sections, I will fit the three major types of model used in psycholinguistics (a) crossed varying intercepts by subject and by item, (b) varying intercepts and slopes by subject and varying intercepts by item, without a correlation estimated, (c) and varying intercepts and slopes by subject and varying intercepts by item, with correlation estimated.

For more complex models, which will come up in later case studies, we will switch to Stan.

11.2 Bayesian analysis: Crossed varying intercepts for subject and item

We will use the lmer fit to decide on initial values when we do Gibbs sampling.

```
> ## using linear mixed model to figure out init values:
> m1<-lmer(rrt~so+(1|subj)+(1|item),subset(critdata,region=="headnoun"))
> ## estimated sd of varying intercept:
> sigma.u<-attr(VarCorr(m1)$subj,"stddev")
> sigma.w<-attr(VarCorr(m1)$item,"stddev")
> ## estimated residual sd:
> sigma.e<-attr(VarCorr(m1),"sc")
> beta<-fixef(m1)
> headnoun<-subset(critdata,region=="headnoun")
```

Set up data for JAGS. **Note that the subject and item should be relabeled as integers in increasing order. This is important for the JAGS model.**

```
> headnoun.dat <- list( subj = sort(as.integer( factor(headnoun$subj) )),
+                       item = sort(as.integer( factor(headnoun$item) )),
+                    rrt = headnoun$rrt,
+                     so = headnoun$so,
+                     N = nrow(headnoun),
+                     I = length( unique(headnoun$subj) ),
+                     K = length( unique(headnoun$item) )
+                       )
```

Set up four chains:

```
> headnoun.ini <- list( list( sigma.e = sigma.e/3,
+                             sigma.u = sigma.u/3,
+                             sigma.w = sigma.w/3,
+                                beta = beta   /3 ),
+                    list( sigma.e = sigma.e*3,
+                             sigma.u = sigma.u*3,
+                             sigma.w = sigma.w*3,
+                                beta = beta   *3 ),
+                    list( sigma.e = sigma.e/3,
+                             sigma.u = sigma.u*3,
+                             sigma.w = sigma.w*3,
+                                beta = beta   /3 ),
+                    list( sigma.e = sigma.e*3,
+                             sigma.u = sigma.u/3,
+                             sigma.w = sigma.w/3,
+                                beta = beta   *3 ) )
>
```

The JAGS model below assumes that we have reciprocal rts.

```
> cat("
+ # Fixing data to be used in model definition
+ model
+    {
+    # The model for each observational unit
+   #   (each row is a subject's data point)
+      for( j in 1:N )
+      {
+      mu[j] <- beta[1] + beta[2] * ( so[j] )  + u[subj[j]] + w[item[j]]
+      rrt[j] ~ dnorm( mu[j], tau.e )
+       }
```

```
+
+    # Random effects for each subject
+      for( i in 1:I )
+      {
+      u[i] ~ dnorm(0,tau.u)
+      }
+
+      # Random effects for each item
+      for( k in 1:K )
+      {
+      w[k] ~ dnorm(0,tau.w)
+      }
+
+    # Uninformative priors:
+
+    # Fixed effect intercept and slope
+      beta[1] ~ dnorm(0.0,1.0E-5)
+      beta[2] ~ dnorm(0.0,1.0E-5)
+
+    # Residual (within-person) variance
+      tau.e <- pow(sigma.e,-2)
+      sigma.e  ~ dunif(0,10)
+
+    # Between-person variation
+      tau.u <- pow(sigma.u,-2)
+      sigma.u  ~ dunif(0,10)
+
+      # Between-item variation
+      tau.w <- pow(sigma.w,-2)
+      sigma.w  ~ dunif(0,10)
+     ## I get normal residuals only if I assume a N(0,100) prior
+     ## or one with larger variance
+
+    }",
+      file="JAGSmodels/gwheadnouncrossedrandom.jag" )
```

Define which variables you want to track:

```
> track.variables<-c("beta","sigma.e","sigma.u","sigma.w")
```

Run model:

```
> library(rjags)
> headnoun.mod <- jags.model( file = "JAGSmodels/gwheadnouncrossedrandom.jag",
+                          data = headnoun.dat,
+                     n.chains = 4,
```

```
+                          inits = headnoun.ini,
+                        n.adapt =2000 , quiet=T)
>
```

Generate posterior samples:

```
> headnoun.res <- coda.samples( headnoun.mod,
+                                  var = track.variables,
+                                n.iter = 10000,
+                                  thin = 20 )
```

One way to assess convergence is with the Gelman-Rubin (or Brooks-Gelman-Rubin) diagnostic. This involves sampling from multiple chains and then comparing between and within group variability. It's analogous to the F-score in anova. Within variance is represented by the mean width of the 95% posterior Credible Intervals (CrI) of all chains, from the final T iterations. Between variance is represented by the width of the 95% CrI using all chains pooled together (for the T iterations). If the ratio $\hat{R} = B/W$ is approximately 1, we have convergence.

```
> ## Gelman-Rubin convergence diagnostic:
> gelman.diag(headnoun.res)

Potential scale reduction factors:

        Point est. Upper C.I.
beta[1]      1.003       1.01
beta[2]      0.999       1.00
sigma.e      1.001       1.00
sigma.u      1.005       1.02
sigma.w      1.009       1.03

Multivariate psrf

1.01
```

And finally, plot results:

```
> plot(headnoun.res)
```

```
> plot(headnoun.res)
```

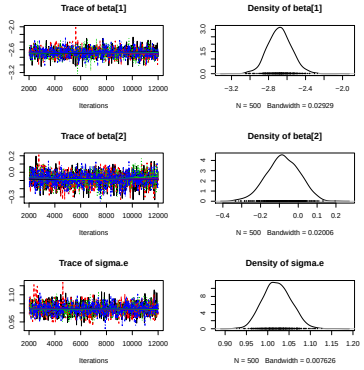


Figure 11.4: Posterior distribution for reading times at head noun.

11.3 Bayesian analysis: Varying intercepts and slopes by subject, varying intercepts by item, no correlation estimated

We now fit a crossed varying intercepts model (items and subject) with varying slopes for subjects. We don't estimate the correlation

between intercepts and slopes; I show two ways to fit this model in JAGS.

We use sum contrast coding as above.

First we define the model. Let i range over subjects ($i = 1,\ldots,n$); j ranges over the two conditions ($j = 1,2$); and k range over the items ($k = 1,\ldots,m$).

Specify JAGS model with varying intercepts for subjects (u_{0i}) and items (w_{0k}), and varying slopes (u_{1i}) by subject.

$$y_{ijk} = (\beta_0 + u_{0i} + w_{0k}) + (\beta_1 + u_{1i}) + e_{ijk} \tag{11.1}$$

where

$$\begin{pmatrix} u_{0i} \\ u_{1i} \end{pmatrix} \sim N(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma^2_{u_{0i}} & \rho\sigma_{u_{0i}}\sigma_{u_{1i}} \\ \rho\sigma_{u_{0i}}\sigma_{u_{1i}} & \sigma^2_{u_{1i}} \end{pmatrix}) \quad \text{and } w_{0k} \sim N(0, \sigma_{w0k}) \tag{11.2}$$

and

$$\varepsilon_{ijk} \sim N(0, \sigma^2) \tag{11.3}$$

In other words, the variance covariance matrix for the varying intercepts and slopes by subject is:

$$\Sigma = \begin{pmatrix} \sigma^2_{u_{0i}} & \rho\sigma_{u_{0i}}\sigma_{u_{1i}} \\ \rho\sigma_{u_{0i}}\sigma_{u_{1i}} & \sigma^2_{u_{1i}} \end{pmatrix} \tag{11.4}$$

with ρ not estimated.

We are going to derive this Σ in JAGS. We do this by modeling the precision matrix Ω, which is the inverse of the variance-covariance matrix Σ, as a Wishart distribution (which takes a positive definite $p \times p$ matrix R, and $k \geq p$, as parameters; see JAGS manual). The Wishart distribution is the multivariate generalization of the gamma distribution. Recall that the inverse gamma is a conjugate prior density for the variance in a univariate normal model; the inverse Wishart is a conjugate prior density for variance in the multivariate case above. So, we can model a precision matrix using the Wishart, and the variance-covariance matrix using the inverse Wishart. By taking the inverse of Ω we get Σ.

Here is the model:

```
> cat("
+ # Fixing data to be used in model definition
+ data
+    {
+    zero[1] <- 0
+    zero[2] <- 0
```

```
+ ## some prior guess for precision matrix:
+    R[1,1] <- 0.1
+    R[1,2] <- 0
+    R[2,1] <- 0
+    R[2,2] <- 0.5
+    }
+ # Then define model
+ model
+    {
+    # Intercept and slope for each person, including random effects
+      for( i in 1:I )
+      {
+      u[i,1:2] ~ dmnorm(zero,Omega.u)
+      }
+
+          # Random effects for each item
+      for( k in 1:K )
+      {
+      w[k] ~ dnorm(0,tau.w)
+      }
+
+    # Define model for each observational unit
+      for( j in 1:N )
+      {
+      mu[j] <- ( beta[1] + u[subj[j],1] ) +
+               ( beta[2] + u[subj[j],2] ) * ( so[j] ) + w[item[j]]
+      rrt[j] ~ dnorm( mu[j], tau.e )
+      }
+
+    #------------------------------------------------------------
+    # Priors:
+
+    # Fixed intercept and slope (uninformative)
+      beta[1] ~ dnorm(0.0,1.0E-5)
+      beta[2] ~ dnorm(0.0,1.0E-5)
+
+    # Residual variance
+      tau.e <- pow(sigma.e,-2)
+    sigma.e  ~ dunif(0,100)
+
+    # Define prior for the variance-covariance matrix of the random effects for subjects
+      Sigma.u <- inverse(Omega.u)
+      Omega.u  ~ dwish( R, 2 )
+
```

```
+     # Between-item variation
+      tau.w <- pow(sigma.w,-2)
+      sigma.w  ~ dunif(0,10)
+    }",
+      file="JAGSmodels/gwintslopeuninf.jag" )
```

We re-do the data set-up just for completeness:

```
> ##
> #headnoun<-subset(data,region=="headnoun")
> #headnoun$region<-factor(headnoun$region)
>
> headnoun.dat <- list(subj =
+               sort(as.integer(factor(headnoun$subj))),
+                       item =
+               sort(as.integer(factor(headnoun$item))),
+                     rrt = headnoun$rrt,
+                     so = headnoun$so,
+                     N = nrow(headnoun),
+                     I = length(unique(headnoun$subj)),
+                     K = length(unique(headnoun$item)))
```

Set up four chains (initial values derived from lmer model):

```
> m2<-lmer(rrt~so+(1|subj)+(0+so|subj)+(1|item),headnoun)
> ( sigma.e <- attr(VarCorr(m2),"sc") )

[1] 0.94575

> ## Build precision matrix, inverse of VarCorr mat:
> Sigma<-matrix(rep(NA,4),ncol=2)
> Sigma[1,1]<-attr(VarCorr(m2)$subj.1,"stddev")
> Sigma[2,2]<-attr(VarCorr(m2)$subj,"stddev")
> ## uncorrelated standard errors of variance components:
> Sigma[1,2]<-Sigma[2,1]<-0
> ( Omega.u <- solve( Sigma ))

       [,1]   [,2]
[1,] 1.6429 0.0000
[2,] 0.0000 4.3777

> (sigma.w<-attr(VarCorr(m2)$item,"stddev"))

(Intercept)
    0.33058

> ( beta    <- fixef( m2 ) )
```

```
(Intercept)          so
  -2.672003   -0.075126
```

Set up initial values:

```
> headnoun.ini <- list( list( sigma.e = sigma.e/3,
+                          Omega.u = Omega.u/3,
+                             sigma.w = sigma.w/3,
+                             beta = beta   /3 ),
+                     list( sigma.e = sigma.e*3,
+                          Omega.u = Omega.u*3,
+                          sigma.w = sigma.w*3,
+                             beta = beta   *3 ),
+                     list( sigma.e = sigma.e/3,
+                          Omega.u = Omega.u*3,
+                          sigma.w = sigma.w/3,
+                             beta = beta   /3 ),
+                     list( sigma.e = sigma.e*3,
+                          Omega.u = Omega.u/3,
+                          sigma.w = sigma.w*3,
+                             beta = beta   *3 ) )
```

Set up JAGS model:

```
> library(rjags)
> headnoun.mod <- jags.model(
+                 file = "JAGSmodels/gwintslopeuninf.jag",
+                 data = headnoun.dat,
+                 n.chains = 4,
+                 inits = headnoun.ini,
+                 n.adapt = 5000 ,quiet=T)
```

Define variables to track, and sample from posterior:

```
> track.variables<-c("beta","sigma.e","Sigma.u","sigma.w")
> headnoun.res <- coda.samples( headnoun.mod,
+                                  var = track.variables,
+                                n.iter = 10000,
+                                  thin = 20 )
```

Then, summarize results, check convergence, and plot (not shown).

```
> summary( headnoun.res )
> ## compare with lmer:
> # Groups   Name        Std.Dev.
> # subj     so          0.228
> # subj.1   (Intercept) 0.609 <- a bit larger in lmer
```

```
> # item     (Intercept) 0.331
> # Residual             0.946
> #                Mean     SD Naive SE Time-series SE
> ## Note that Sigma contains variances
> ## but lmer prints out sds:
> #Sigma.u[1,1]  0.2555 0.0934 0.002088       0.002120
> #Sigma.u[2,1] -0.0251 0.0584 0.001306       0.001349
> #Sigma.u[1,2] -0.0251 0.0584 0.001306       0.001349
> #Sigma.u[2,2]  0.1171 0.0544 0.001216       0.001156
> #sigma.e       0.9585  0.9958  1.0159  1.03892  1.0831
> #sigma.w       0.3100 0.1499 0.003351       0.004996
> ## multivariate=F: see help.
> gelman.diag(headnoun.res,multivariate=F)
> par( mfrow=c(3,3) )
> plot(headnoun.res)
```

Here is an alternative way to fit this model, which does not estimate the correlation between intercepts and slopes. We could have defined the model without a prior for the precision matrix, because the two variance components by subject (varying intercepts and varying slopes) are independent:

```
> cat("
+ model
+    {
+    # Intercept and slope for each person, including random effects
+      for( i in 1:I )
+      {
+      u[i,1] ~ dnorm(0,tau.u.int)
+      u[i,2] ~ dnorm(0,tau.u.slopes)
+      }
+
+          # Random effects for each item
+      for( k in 1:K )
+      {
+      w[k] ~ dnorm(0,tau.w)
+      }
+
+    # Define model for each observational unit
+      for( j in 1:N )
+      {
+      mu[j] <- ( beta[1] + u[subj[j],1] ) +
+               ( beta[2] + u[subj[j],2] ) * ( so[j] ) + w[item[j]]
+      rrt[j] ~ dnorm( mu[j], tau.e )
+      }
```

```
+
+    #-------------------------------------------------------------
+    # Priors:
+
+    # Fixed intercept and slope (uninformative)
+      beta[1] ~ dnorm(0.0,1.0E-5)
+      beta[2] ~ dnorm(0.0,1.0E-5)
+
+    # Residual variance
+      tau.e <- pow(sigma.e,-2)
+    sigma.e  ~ dunif(0,100)
+
+     # Between-subj variation
+     tau.u.int <- pow(sigma.u.int,-2)
+     sigma.u.int ~ dunif(0,10)
+     tau.u.slopes <- pow(sigma.u.slopes,-2)
+     sigma.u.slopes ~ dunif(0,10)
+
+     # Between-item variation
+      tau.w <- pow(sigma.w,-2)
+      sigma.w  ~ dunif(0,10)
+    }",
+      file="JAGSmodels/gwintslopeuninfv2.jag" )
```

We need to work out the initial values again because we have to set them up differently now (the precision matrix is no longer in the picture):

```
> ## the other initial values remain unchanged.
> ## now we have two independent std devs for int and slope:
> (sigma.u.int <-attr(VarCorr(m2)$subj.1,"stddev"))

(Intercept)
    0.60869

> (sigma.u.slopes <-attr(VarCorr(m2)$subj,"stddev"))

     so
0.22843
```

Set up initial values:

```
> headnoun.ini <- list( list( sigma.e = sigma.e/3,
+                          sigma.u.int = sigma.u.int/3,
+                          sigma.u.slopes = sigma.u.slopes/3,
+                             sigma.w = sigma.w/3,
```

```
+                            beta = beta   /3 ),
+                    list( sigma.e = sigma.e*3,
+                          sigma.u.int = sigma.u.int*3,
+                          sigma.u.slopes = sigma.u.slopes*3,
+                          sigma.w = sigma.w*3,
+                            beta = beta   *3 ),
+                    list( sigma.e = sigma.e/3,
+                          sigma.u.int = sigma.u.int*3,
+                          sigma.u.slopes = sigma.u.slopes*3,
+                          sigma.w = sigma.w/3,
+                            beta = beta   /3 ),
+                    list( sigma.e = sigma.e*3,
+                          sigma.u.int = sigma.u.int/3,
+                          sigma.u.slopes = sigma.u.slopes/3,
+                          sigma.w = sigma.w*3,
+                            beta = beta   *3 ) )
```

Set up JAGS model:

```
> library(rjags)
> headnoun.mod <- jags.model(
+                 file = "JAGSmodels/gwintslopeuninfv2.jag",
+                 data = headnoun.dat,
+                 n.chains = 4,
+                 inits = headnoun.ini,
+                 n.adapt = 5000 ,quiet=T)
```

Define variables to track, and sample from posterior:

```
> track.variables<-c("beta","sigma.e","sigma.u.int","sigma.u.slopes","sigma.w")
> headnoun.res <- coda.samples( headnoun.mod,
+                                 var = track.variables,
+                               n.iter = 10000,
+                                 thin = 20 )
```

Then, summarize results, check convergence, and plot (not shown).

```
> summary( headnoun.res )
> ## compare with lmer:
> # Groups   Name        Std.Dev.
> # subj     so          0.228
> # subj.1   (Intercept) 0.609 <- a bit larger in lmer
> # item     (Intercept) 0.331
> # Residual             0.946
> #                Mean     SD Naive SE Time-series SE
> #Sigma.u[1,1]  0.2555 0.0934 0.002088       0.002120
```

```
> #Sigma.u[2,1] -0.0251 0.0584 0.001306       0.001349
> #Sigma.u[1,2] -0.0251 0.0584 0.001306       0.001349
> #Sigma.u[2,2]  0.1171 0.0544 0.001216       0.001156
> #sigma.e       0.9585  0.9958  1.0159  1.03892  1.0831
> #sigma.w       0.3100 0.1499 0.003351       0.004996
> ## V2, without precision matrix prior:
> #                   Mean     SD Naive SE Time-series SE
> #sigma.e         1.0187 0.0321 0.000717       0.000758
> ## note that these are standard deviations:
> ## cf Sigma[1,1] and Sigma[2,2] above, which are
> ## variances!
> #sigma.u.int     0.5360 0.0980 0.002190       0.002503
> #sigma.u.slopes  0.1549 0.1055 0.002358       0.005612
> #sigma.w         0.2766 0.1487 0.003326       0.005627
```

We get very similar results using both methods.

11.3.1 The correlation between varying intercepts and slopes

In the above model, we do not estimate ρ. Recall that the correlation is that between the varying intercepts and slopes (the BLUPs). Even though ρ is not estimated in the model, it can be calculated, as I discuss below.

First, a refresher: If we were interesting in estimating the relationship between the different intercepts and slopes for each subject for the factor so, we could fit a separate linear model for each subject, compute the intercept and slope for each subject, and just plot the relationship. We can also compute the correlation, see below.

In linear mixed models, we don't compute these intercepts and slopes separately for each subject, but rather we "borrow strength from the grand mean" and conservatively cause the intercepts and slopes to gravitate towards the grand mean intercept and slope. Gelman and Hill present a nice discussion of this kind of "shrinkage" (but everyone who took my summer course should have seen this before).

To estimate the above correlation from our model, we just need to figure out what the estimated value is from the model of the off-diagonal in Σ. The summary output of the model shows all the components of Σ. Then we can use the fact that the off-diagonal value is the covariance, i.e., $Covariance = \rho\sigma_A\sigma_B$, where σ_A is the standard deviation of the intercepts (this can be gotten from the posterior estimate for Σ, from the [1,1] cell), and σ_B is the standard deviation of the slopes. **Do this calculation now for this data**.

The point here is that even though we did not estimate the correlation in the model, we can calculate it. In other words, the difference between the two specifications we use in psycholinguistics with lmer:

```
> intslopes<-lmList(rrt~so|subj,subset(
> intercepts<-coef(intslopes)[,1]
> slopes<-coef(intslopes)[,2]
> plot(intercepts~slopes,xlab="slopes",
> cor(intercepts,slopes)
[1] -0.21697
```

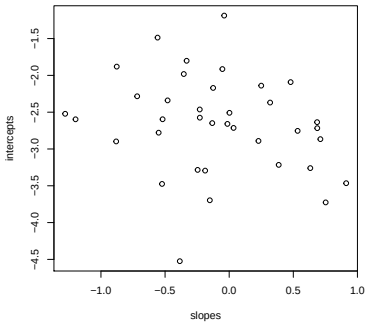


Figure 11.5: Intercepts and slopes by subject.

`(1|subj) + (0+so|subj)` and `(1+so|subj)`

is only that in the first case no correlation is printed out, and in the second it is. Note that the value get in the second case is not totally identical to the one computed by hand in the first case, but it's quite similar.

Homework 11

Fit a varying intercepts and slopes model (no correlation estimated) for the region following the head noun.

Homework 12

Return to the lexdec dataset from the regression chapter, and fit a model with varying intercepts for subject, and varying intercepts for Word (i.e., subject and Word are random effects).

11.4 Bayesian analysis: Fitting a varying intercepts, varying slopes model for subjects, including correlation term

The model will be:

```
> m3 <- lmer(rrt~so+(1+so|subj)+(1|item),
+           subset(critdata,region=="headnoun"))
```

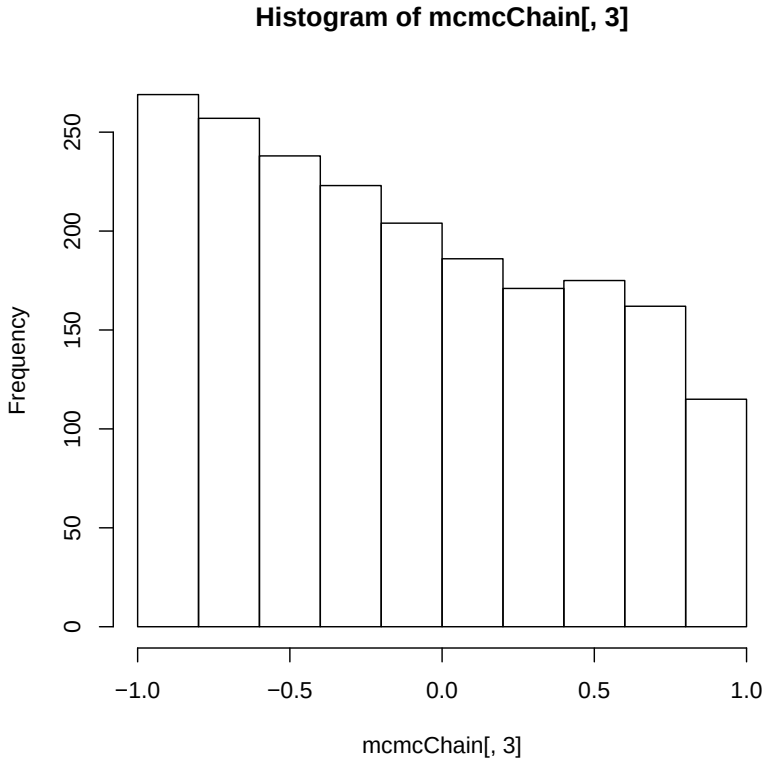
Here is the JAGS code:

```
> cat("
+ # Fixing data to be used in model definition
+ data
+ {
+     zero[1] <- 0
+     zero[2] <- 0
+ ## the var-cov matrix is defined below
+ }
+     # Then define model
+     model
+ {
+     # Intercept and slope for each person, including random effects
+     for( i in 1:I )
+ {
+     u[i,1:2] ~ dmnorm(zero,Omega.u)
+ }
+
+     # Random effects for each item
+     for( k in 1:K )
+ {
+     w[k] ~ dnorm(0,tau.w)
```

```
+ }
+
+     # Define model for each observational unit
+     for( j in 1:N )
+ {
+     mu[j] <- ( beta[1] + u[subj[j],1] ) +
+     ( beta[2] + u[subj[j],2] ) * ( so[j] ) + w[item[j]]
+     rrt[j] ~ dnorm( mu[j], tau.e )
+ }
+
+     #------------------------------------------------------------
+     # Priors:
+
+     # Fixed intercept and slope (uninformative)
+     beta[1] ~ dnorm(0.0,1.0E-5)
+     beta[2] ~ dnorm(0.0,1.0E-5)
+
+     # Residual variance
+     tau.e <- pow(sigma.e,-2)
+     sigma.e  ~ dunif(0,100)
+
+     # Define prior for the variance-covariance matrix of the random effects for subjects
+     ## precision:
+     Omega.u  ~ dwish( R, 2 )
+     ## R matrix:
+     R[1,1] <- pow(sigma.a,2)
+     R[2,2] <- pow(sigma.b,2)
+     R[1,2] <- rho*sigma.a*sigma.b
+     R[2,1] <- R[1,2]
+     ## Vcov matrix:
+     Sigma.u <- inverse(Omega.u)
+     ## priors for var int. var slopes
+     sigma.a ~ dunif(0,10)
+     sigma.b ~ dunif(0,10)
+
+     ## prior for correlation:
+     rho ~ dunif(-1,1)
+
+     # Between-item variation
+     tau.w <- pow(sigma.w,-2)
+     sigma.w  ~ dunif(0,10)
+ }",
+      file="JAGSmodels/gwintslopeuninfcorr.jag" )
```

```
Iterations = 2020:12000
Thinning interval = 20
Number of chains = 4
Sample size per chain = 500

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

           Mean     SD Naive SE Time-series SE
beta[1] -2.6839 0.1335 0.002985       0.003045
beta[2] -0.0791 0.0942 0.002106       0.002029
rho     -0.1276 0.5611 0.012546       0.020519
sigma.a  0.7712 0.3772 0.008434       0.009359
sigma.b  0.2104 0.1836 0.004106       0.008564
sigma.e  1.0193 0.0321 0.000718       0.000705
sigma.w  0.2744 0.1454 0.003252       0.005218

2. Quantiles for each variable:

           2.5%     25%     50%     75%   97.5%
beta[1] -2.9677 -2.7665 -2.6783 -2.5978 -2.4313
beta[2] -0.2704 -0.1424 -0.0778 -0.0134  0.0984
rho     -0.9541 -0.6228 -0.1886  0.3397  0.9019
sigma.a  0.1944  0.4846  0.7152  1.0034  1.6171
sigma.b  0.0122  0.0732  0.1577  0.2913  0.6869
sigma.e  0.9585  0.9972  1.0182  1.0403  1.0842
sigma.w  0.0292  0.1659  0.2674  0.3648  0.5879
```

```
> ## for later comparison:
> hist(mcmcChain[,3])
```



11.5 Summary

We can now fit the three classical types of psycholinguistic models in JAGS:

1. Varying intercepts:

```
> m1<-lmer(rrt~so+(1|subj)+(1|item),headnoun)
```

2. Varying intercepts and slopes, no correlation estimated:

```
> m1<-lmer(rrt~so+(1|subj)+(0+so|subj)+(1|item),
+          headnoun)
```

3. Varying intercepts and slopes, correlation estimated:

```
> m2<-lmer(rrt~so+(1|subj)+(0+so|subj)+(1|item),
+          headnoun)
```

Gelman and Hill 2007 (p. 375) say that the second type is inappropriate. But we will often have to fit them because we get a convergence error in the third type of model (not enough data). I present one solution to this issue next.

11.6 What to do about ±1 correlation estimates

We often find ourselves with a varying intercepts, varying slopes model where we ask lmer to estimate the correlation and it returns a +1 or −1 correlation. Normally, we just back off to not estimating the correlation. Chung et al have an unpublished paper where they solve this problem in the bayesian setting:

"The key problem solved by our method is the tendency of maximum likelihood estimates of Σ to be degenerate, that is, on the border of positive-definiteness, which corresponds to **zero variance or perfect correlation** among some linear combinations of the parameters. ... When the maximum likelihood estimate of a hierarchical covariance matrix is degenerate, this arises from a likelihood that is nearly flat in the relevant dimension and **just happens to have a maximum at the boundary**."

Their recommendation:

"A small amount of regularization puts the posterior mode inside the allowable space and reduces mean squared error. Our solution is a class of weakly informative prior densities for Σ that go to zero on the boundary as Σ becomes degenerate, thus ensuring that the posterior mode (i.e., the maximum penalized likelihood estimate) is always nondegenerate."

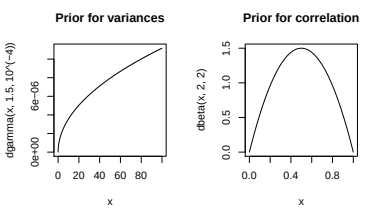
"We recommend a class of Wishart priors with a default choice of hyperparameters: the degrees of freedom is the dimension of b_j plus two and the scale matrix is the identity matrix multiplied by a large enough number. This prior can be expressed as ... a product of gamma(1.5, θ) priors on variances of the varying effects with rate parameter $\theta \to 0$ and a function of the correlations (a beta prior the two-dimensional case)."

Later on they say:

"independent gamma(1.5,θ) priors on both σ_a and σ_b, and a beta(1.5,1.5) prior on $(\rho + 1)/2$."

This means that if we have a σ_a, σ_b as varying intercept and varying slope correlations ρ:

```
## varying intercepts' variance:
sigma_a ~ dgamma(1.5,10^(-4))
## varying slopes' variance:
sigma_b ~ dgamma(1.5,10^(-4))
## correlation:
rho <- rho2*2-1
rho2 ~ dbeta(1.5,1.5)
```



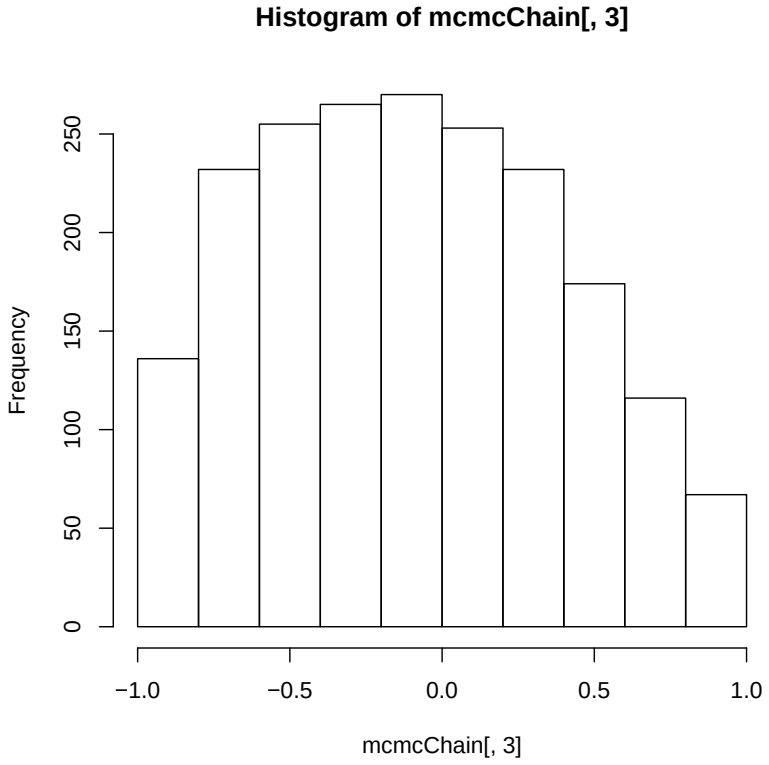
Here is what these priors look like.

And here is a JAGS model with the correlation prior shown above:

```
> cat("
+ # Fixing data to be used in model definition
+ data
+ {
+     zero[1] <- 0
+     zero[2] <- 0
+ ## the var-cov matrix is defined below
+ }
+     # Then define model
+     model
+ {
+     # Intercept and slope for each person, including random effects
+     for( i in 1:I )
+ {
+     u[i,1:2] ~ dmnorm(zero,Omega.u)
+ }
+
+     # Random effects for each item
+     for( k in 1:K )
+ {
+     w[k] ~ dnorm(0,tau.w)
+ }
+
+     # Define model for each observational unit
+     for( j in 1:N )
+ {
+     mu[j] <- ( beta[1] + u[subj[j],1] ) +
+     ( beta[2] + u[subj[j],2] ) * ( so[j] ) + w[item[j]]
+     rrt[j] ~ dnorm( mu[j], tau.e )
+ }
+
+     #-------------------------------------------------------------
+     # Priors:
+
+     # Fixed intercept and slope (uninformative)
+     beta[1] ~ dnorm(0.0,1.0E-5)
+     beta[2] ~ dnorm(0.0,1.0E-5)
+
+     # Residual variance
+     tau.e <- pow(sigma.e,-2)
+     sigma.e  ~ dunif(0,100)
+
+     # Define prior for the variance-covariance matrix of the random effects for subjects
+     ## precision:
```

```
+     Omega.u  ~ dwish( R, 2 )
+     ## R matrix:
+     R[1,1] <- pow(sigma.a,2)
+     R[2,2] <- pow(sigma.b,2)
+     R[1,2] <- rho*sigma.a*sigma.b
+     R[2,1] <- R[1,2]
+     ## Vcov matrix:
+     Sigma.u <- inverse(Omega.u)
+     ## priors for var int. var slopes
+     sigma.a ~ dunif(0,10)
+     sigma.b ~ dunif(0,10)
+
+     ## prior for correlation:
+     #rho ~ dunif(-1,1)
+     rho <- rho2*2-1
+     rho2 ~ dbeta(1.5,1.5)
+
+     # Between-item variation
+     tau.w <- pow(sigma.w,-2)
+     sigma.w  ~ dunif(0,10)
+ }",
+      file="JAGSmodels/gwintslopeuninfcorr.jag" )
```

```
> hist(mcmcChain[,3])
```



11.7 Fitting LMMs in Stan

As soon as we transition to more complex models, we will need Stan; JAGS has its limits. I will cover three basic types of models that we will fit in Stan:

1. Varying intercepts for subject and item
2. Varying intercepts and varying slopes for subject, and varying intercepts for items.

Recall that in textbooks and in these lecture notes, we refer to the normal distribution in terms of the variance: $N(\mu, \sigma^2)$. In R, we refer to it in terms of standard deviation: $N(\mu, \sigma)$. Then came WinBUGS and JAGS, which refers to the normal in terms of precision: $N(\mu, 1/\sigma^2)$. In Stan, the normal distribution is like R, referred to in terms of the standard deviation. Make sure you keep that in mind.

First make sure you have RStan installed. See the RStan website for details. I begin by fitting a model with varying intercept by subject and by item. I leave this uncommented because it's pretty obvious what the code is doing, given that we have seen similar JAGS code before.

```
> library(rstan)
> set_cppo("fast")  # for best running speed
> #set_cppo('debug') # make debug easier
>
> ## Gibson and Wu in RStan
> ## varying intercepts for subjects and items:
>
> data<-read.table("data/gibsonwu2012data.txt",header=T)
> ## convert to reciprocal rt:
> data$rrt<- -1000/data$rt
> ## contrast coding:
> data$so <- ifelse(
+   data$type%in%c("subj-ext"),-0.5,0.5)
> ## reference values from lmer:
> library(lme4)
> (m1 <- lmer(rrt~so+(1|subj),
+             subset(data,region=="headnoun")))
> headnoun<-subset(data,region=="headnoun")
> ## Stan ready data:
> headnoun.dat <- list(subj=
+                      sort(as.integer(
+                      factor(headnoun$subj) )),
+                      item=sort(as.integer(
+                           factor(headnoun$item) )),
+                       rrt = headnoun$rrt,
+                       so = headnoun$so,
+                       N = nrow(headnoun),
+                       I =
+                         length( unique(headnoun$subj) ),
+                       K =
+                         length( unique(headnoun$item) )
+ )
> ## Stan code:
> varying.int_code<-'
+   data {
+ int<lower=1> N; //no. of rows
+ real so[N];     // predictor
+ real rrt[N];   //outcome
+ int<lower=1> I;   //number of subjects
+ int<lower=1, upper=I> subj[N];  //subject id
+ int<lower=1> K;   //number of items
+ int<lower=1, upper=K> item[N];  //item id
+ }
+ parameters {
```

```
+ real alpha;      // intercept
+ real beta;       // slope
+ real u[I];   // random intercept subj
+ real w[K];   // random intercept item
+ real<lower=0> sigma_e;  // residual variance
+ real<lower=0> sigma_u;   //  subject var
+ real<lower=0> sigma_w;   //  item var
+ }
+ model {
+ real mu[N];
+ for (i in 1:N) {
+ mu[i] <- alpha + beta*so[i] + u[subj[i]]+w[item[i]];
+ }
+ rrt ~ normal(mu,sigma_e);    // likelihood
+ alpha ~ normal(0,5);
+ beta ~ normal(0,5);
+ u ~ normal(0,sigma_u);
+ w ~ normal(0,sigma_w);
+ ## could have used uniform:
+ sigma_u ~ cauchy(0,5);
+ sigma_w ~ cauchy(0,5);
+ sigma_e ~ cauchy(0,5);
+ }
+ '
> fit <- stan(model_code = varying.int_code,
+             data = headnoun.dat,
+             iter = 500, chains = 2)
> #print(fit)
> #plot(fit)
>
> ## matches up:
> m1<-lmer(rrt~so+(1|subj)+(1|item),headnoun)
```

Next we look at the model with varying intercepts and varying slopes by subject. Here, we need to know how obtain the variance-covariance matrix (Σ) from the correlation matrix which, just to confuse the issue, we will call Ω (not to be confused with Ω, the precision matrix we used earlier to fit the varying intercept and varying slope model in JAGS).

In order to define a prior for the variance-covariance matrix, we will proceed as follows. We will first define a prior for the correlation matrix Ω using the Stan function:

```
Omega ~ lkj_corr(2.0)
```

This function generates a random correlation matrix whose distribution depends on a single "eta" parameter. This eta parameter can be treated as the shape parameter of a symmetric beta distribution. If eta = 1 then its distribution is jointly uniform; if it's greater than 1, then the lkj_corr distribution is concentrated around the identity matrix, which has 1's in the diagonals and 0's in the off-diagonals (recall that it's a multivariate distribution), and as eta increases, the distribution becomes more sharply concentrated around the identity matrix. If eta lies between 0 and 1, then there is a trough at the identity matrix.

Let's say we have defined a prior for the correlation matrix, and a prior for each variance component in the variance-covariance matrix (e.g., one for the standard deviation of the varying intercepts, and one for the standard deviation of the varying slopes). We can now derive the prior distribution of the variance-covariance matrix. Suppose that we have a 2x2 correlation matrix, Omega, and two standard deviations, let's say these have value 1 and 2. We can simply multiply the correlation matrix with the standard deviations to get the variance covariance matrix.

```
> ## 2x2 correlation matrix:
> Omega <- matrix(c(1.00, 0.25,
+                   0.25, 1.00),ncol=2)
> sigmas <- c(1,2)
> ## create empty 2x2 matrix:
> Sigma <- matrix(rep(NA,4),ncol=2)
> ##
> for (r in 1:2) {
+ for (c in 1:2) {
+ print(sigmas[r] * sigmas[c] * Omega[r,c])
+ Sigma[r,c] <- sigmas[r] * sigmas[c] * Omega[r,c]
+ }
+ }

[1] 1
[1] 0.5
[1] 0.5
[1] 4
```

This is how we are going to specify the prior for the variance-covariance matrix. Recall that in the JAGS model we used the Wishart distribution; we could have done that here too (exercise).

```
> ## varying intercepts and varying slopes model:
> ## of Gibson and Wu data:
>
> headnoun.dat <- list(zero=c(0,0),
```

```
+                       subj=
+                         sort(as.integer(
+                           factor(headnoun$subj) )),
+                       item=sort(as.integer(
+                         factor(headnoun$item) )),
+                       rrt = headnoun$rrt,
+                       so = headnoun$so,
+                       N = nrow(headnoun),
+                       I =
+                         length( unique(headnoun$subj) ),
+                       K =
+                         length( unique(headnoun$item) )
+ )
> varyingintslopes_code<- 'data {
+ int<lower=1> N;
+ real rrt[N];   //outcome
+ real so[N];   //predictor
+ int<lower=1> I;   //number of subjects
+ int<lower=1> K;   //number of items
+ int<lower=1, upper=I> subj[N];  //subject id
+ int<lower=1, upper=K> item[N];  //item id
+ vector[2] zero; //vector of zeros passed in from R
+ }
+ parameters {
+ vector[2] beta;      // intercept and slope
+ vector[2] u[I];   // random intercept and slope
+ real w[K];   // random intercept item
+ real<lower = 0> sigma_e;  // residual sd
+ vector<lower=0>[2] sigma_u;   // subj sd
+ real<lower=0> sigma_w;   // item sd
+ // Note: Omega is the correlation matrix, not the precision matrix.
+ // We are going to build Sigma from Omega.
+ corr_matrix[2] Omega;     // correlation matrix for random intercepts and slopes
+ }
+ transformed parameters {
+ cov_matrix[2] Sigma;  // constructing the variance/cov matrix for random effects
+ for (r in 1:2) {
+ for (c in 1:2) {
+ Sigma[r,c] <- sigma_u[r] * sigma_u[c] * Omega[r,c];
+ }
+ }
+ }
+ model {
+ real mu[N];   // mu for likelihood
```

```
+ for (i in 1:I) u[i] ~ multi_normal(zero, Sigma);  // loop for subj random effects
+ for (k in 1:K) w[k] ~ normal(0,sigma_w);
+ // loop for item random effects
+ for (n in 1:N) {
+ mu[n] <- beta[1] + beta[2]*so[n] + u[subj[n], 1] + u[subj[n], 2]*so[n];
+ }
+ rrt ~ normal(mu,sigma_e);    // likelihood
+ beta ~ normal(0,5);
+ sigma_e ~ cauchy(0,2);
+ sigma_u ~ cauchy(0,2);
+ sigma_w ~ cauchy(0,2);
+ Omega ~ lkj_corr(2.0);
+ }
+ '
> library(rstan)
> set_cppo("fast")  # for best running speed
> fit <- stan(model_code = varyingintslopes_code,
+             data = headnoun.dat,
+             iter = 500, chains = 2)

TRANSLATING MODEL 'varyingintslopes_code' FROM Stan CODE TO C++ CODE NOW.
COMPILING THE C++ CODE FOR MODEL 'varyingintslopes_code' NOW.
SAMPLING FOR MODEL 'varyingintslopes_code' NOW (CHAIN 1).

Iteration:   1 / 500 [  0%]  (Warmup)
Iteration:  50 / 500 [ 10%]  (Warmup)
Iteration: 100 / 500 [ 20%]  (Warmup)
Iteration: 150 / 500 [ 30%]  (Warmup)
Iteration: 200 / 500 [ 40%]  (Warmup)
Iteration: 250 / 500 [ 50%]  (Warmup)
Iteration: 300 / 500 [ 60%]  (Sampling)
Iteration: 350 / 500 [ 70%]  (Sampling)
Iteration: 400 / 500 [ 80%]  (Sampling)
Iteration: 450 / 500 [ 90%]  (Sampling)
Iteration: 500 / 500 [100%]  (Sampling)
Elapsed Time: 13.079 seconds (Warm-up)
              6.00743 seconds (Sampling)
              19.0865 seconds (Total)

SAMPLING FOR MODEL 'varyingintslopes_code' NOW (CHAIN 2).

Iteration:   1 / 500 [  0%]  (Warmup)
Iteration:  50 / 500 [ 10%]  (Warmup)
Iteration: 100 / 500 [ 20%]  (Warmup)
```

```
Iteration: 150 / 500 [ 30%]  (Warmup)
Iteration: 200 / 500 [ 40%]  (Warmup)
Iteration: 250 / 500 [ 50%]  (Warmup)
Iteration: 300 / 500 [ 60%]  (Sampling)
Iteration: 350 / 500 [ 70%]  (Sampling)
Iteration: 400 / 500 [ 80%]  (Sampling)
Iteration: 450 / 500 [ 90%]  (Sampling)
Iteration: 500 / 500 [100%]  (Sampling)
Elapsed Time: 8.87183 seconds (Warm-up)
              10.5865 seconds (Sampling)
              19.4584 seconds (Total)

>
> #print(fit)
```

11.8 Fitting models without correlation estimated for varying intercepts vs varying slopes

Earlier I had discussed how to fit models with varying intercepts and slopes, but without a correlation estimated. This corresponds to models with a specification like:

```
... (1|subj)+(so-1|subj)...
```

Here are two ways of fitting such a model. Compare this to the JAGS models fit earlier (page 164).

```
> ## varying intercepts and varying slopes
>
> headnoun.dat <- list(zero=c(0,0),
+                      subj=
+                        sort(as.integer(
+                          factor(headnoun$subj) )),
+                      item=sort(as.integer(
+                        factor(headnoun$item) )),
+                      rrt = headnoun$rrt,
+                      so = headnoun$so,
+                      N = nrow(headnoun),
+                      I =
+                        length( unique(headnoun$subj) ),
+                      K =
+                        length( unique(headnoun$item) )
+ )
> varyingintslopes_code<- 'data {
+ int<lower=1> N;
```

```
+ real rrt[N];   //outcome
+ real so[N];   //predictor
+ int<lower=1> I;   //number of subjects
+ int<lower=1> K;   //number of items
+ int<lower=1, upper=I> subj[N];  //subject id
+ int<lower=1, upper=K> item[N];  //item id
+ vector[2] zero; //vector of zeros passed in from R
+ }
+ parameters {
+ vector[2] beta;      // intercept and slope
+ vector[2] u[I];   // random intercept and slope
+ real w[K];   // random intercept item
+ real<lower = 0> sigma_e;  // residual sd
+ vector<lower=0>[2] sigma_u;   // subj sd
+ real<lower=0> sigma_w;   // item sd
+ // Note: Omega is the correlation matrix, not the precision matrix.
+ // We are going to build Sigma from Omega.
+ corr_matrix[2] Omega;     // correlation matrix for random intercepts and slopes
+ }
+ transformed parameters {
+ cov_matrix[2] Sigma;  // constructing the variance/cov matrix for random effects
+ for (r in 1:2) {
+ for (c in 1:2) {
+ Sigma[r,c] <- sigma_u[r] * sigma_u[c] * Omega[r,c];
+ }
+ }
+ }
+ model {
+ real mu[N];   // mu for likelihood
+ for (i in 1:I) u[i] ~ multi_normal(zero, Sigma);  // loop for subj random effects
+ for (k in 1:K) w[k] ~ normal(0,sigma_w);
+ // loop for item random effects
+ for (n in 1:N) {
+ mu[n] <- beta[1] + beta[2]*so[n] + u[subj[n], 1] + u[subj[n], 2]*so[n];
+ }
+ rrt ~ normal(mu,sigma_e);    // likelihood
+ beta ~ normal(0,10);
+ sigma_e ~ normal(0,10);
+ sigma_u ~ normal(0,10);
+ sigma_w ~ normal(0,10);
+ Omega ~ lkj_corr(2.0);
+ }
+ '
> library(rstan)
```

```
> set_cppo("fast")  # for best running speed
> fit <- stan(model_code = varyingintslopes_code,
+             data = headnoun.dat,
+             iter = 500, chains = 2)
> ##print(fit)
> ##              Stan  lmer
> #subjintercept   0.6  0.610
> #subjslope       0.1  0.229
> #item            6.3  0.331 <= don't match
> #error           1.0  0.946
> #m2<-lmer(rrt~so+(1+so|subj)+(1|item),headnoun)
> #VarCorr(m2)
>
> ## second version:
> headnoun.dat <- list(subj=
+                        sort(as.integer(
+                          factor(headnoun$subj) )),
+                      item=sort(as.integer(
+                        factor(headnoun$item) )),
+                      rrt = headnoun$rrt,
+                      so = headnoun$so,
+                      N = nrow(headnoun),
+                      I =
+                        length( unique(headnoun$subj) ),
+                      K =
+                        length( unique(headnoun$item) )
+ )
> varyingintslopes_codev2 <- 'data {
+ int<lower=1> N;
+ real rrt[N];   //outcome
+ real so[N];   //predictor
+ int<lower=1> I;   //number of subjects
+ int<lower=1> K;   //number of items
+ int<lower=1, upper=I> subj[N];  //subject id
+ int<lower=1, upper=K> item[N];  //item id
+ }
+ parameters {
+ vector[2] beta;      // intercept and slope
+ vector[2] u[I];   // random intercept and slope
+ real w[K];   // random intercept item
+ real<lower=0> sigma_e;  // residual sd
+ vector<lower=0>[2] sigma_u;   // subj sd
+ real<lower=0> sigma_w;   // item sd
+ }
```

```
+ model {
+ real mu[N];   // mu for likelihood
+ for (i in 1:I) u[i,1] ~ normal(0, sigma_u[1]);
+ for (i in 1:I) u[i,2] ~ normal(0, sigma_u[2]);
+ for (k in 1:K) w[k] ~ normal(0,sigma_w);
+ for (n in 1:N) {
+ mu[n] <- beta[1] + beta[2]*so[n] + u[subj[n], 1] + u[subj[n], 2]*so[n];
+ }
+ rrt ~ normal(mu,sigma_e);    // likelihood
+ beta ~ normal(0,5);
+ sigma_e ~ normal(0,5);
+ sigma_u ~ normal(0,5);
+ sigma_w ~ normal(0,5);
+ }
+ '
> library(rstan)
> set_cppo("fast")  # for best running speed
> fitv2 <- stan(model_code = varyingintslopes_codev2,
+             data = headnoun.dat,
+             iter = 1000, chains = 2)
> ##print(fitv2)
> ##              Stan            lmer
> #subjintercept   0.6            0.610
> #subjslope       0.2 (was 0.1)  0.229
> #item            4.2 (was 6.3)  0.331 <= don't match
> #error           1.0            0.946
> m3<-lmer(rrt~so+(1|subj)+(so-1|subj)+(1|item),headnoun)
> VarCorr(m3)
```

11.9 Fitting even more complex linear mixed models

If our random effects exceeds 2, i.e., if we have more than two variance components for random effects, we will probably be better off doing the modeling in Stan because defining priors for var-cov matrices of dimension greater than 2x2 is easier in Stan.

11.9.1 Example: Kliegl et al 2011

I benefitted greatly from discussion with members of the Stan mailing list for this example. In particular, thanks go to Sergio Polini for showing me how to write efficient code for such problems.

This is an example of a fairly involved dataset which can't easily be fit in JAGS. We also show here how parallelization of chains can be carried out when multiple cores are available on your machine.

One thing to note here is that we use the lkj_corr function in Stan. This was discussed earlier in the notes on priors for variance-covariance matrices (see page 138).

The Stan code (save this as a text file called "kliegl2.stan" and put it in the same directory that you call the R code from below):

```
data {
    int<lower=1> N;
    real rt[N];                     //outcome
    real c1[N];                     //predictor
    real c2[N];                     //predictor
    real c3[N];                     //predictor
    int<lower=1> I;                 //number of subjects
    int<lower=1, upper=I> id[N];    //subject id
    vector[4] mu_prior;             //vector of zeros passed in from R
}
transformed data {
    real ZERO;                      // like #define ZERO 0 in C/C++
    ZERO <- 0.0;
}
parameters {
    vector[4] beta;                 // intercept and slope
    vector[4] u[I];                 // random intercept and slopes
    real<lower=0> sigma_e;          // residual sd
    vector<lower=0>[4] sigma_u;     // subj sd
    corr_matrix[4] Omega;           // correlation matrix for random intercepts and slopes
}
transformed parameters {
    matrix[4,4] D;
    D <- diag_matrix(sigma_u);
}
model {
    matrix[4,4] L;
    matrix[4,4] DL;
    real mu[N]; // mu for likelihood
    //priors:
    beta ~ normal(0,50);
    sigma_e ~ normal(0,100);
    sigma_u ~ normal(0,100);
    Omega ~ lkj_corr(4.0);

    L <- cholesky_decompose(Omega);
    for (m in 1:4)
        for (n in 1:m)
```

```
            DL[m,n] <- L[m,n] * sigma_u[m];
    for (m in 1:4)
        for (n in (m+1):4)
            DL[m,n] <- ZERO;

    for (i in 1:I)                  // loop for subj random effects
        u[i] ~ multi_normal_cholesky(mu_prior, DL);

    for (n in 1:N) {
        mu[n] <- beta[1] + beta[2]*c1[n] + beta[3]*c2[n] + beta[4]*c3[n]
            + u[id[n], 1] + u[id[n], 2]*c1[n] + u[id[n], 3]*c2[n] + u[id[n], 4]*c3[n];
    }
    rt ~ normal(mu,sigma_e);        // likelihood
}
generated quantities {
    cov_matrix[4] Sigma;
    Sigma <- D * Omega * D;
}
```

And here is the model fit (not run, because it takes quite long).

```
> library(rstan)
> library(parallel)
> load("KWDYZ_test.rda")
> dat2 <- list(mu_prior=c(0,0,0,0),
+                      id=sort(as.integer(factor(dat$id))),
+                      rt = dat$rt,
+                      c1 = dat$c1,
+                      c2 = dat$c2,
+                      c3 = dat$c3,
+                      N = nrow(dat),
+                      I = length(unique(dat$id)))
> kliegl2.sm <- stan_model("kliegl2.stan", model_name = "kliegl")
> sflist <- mclapply(1:4, mc.cores = detectCores(),
+                    function(i) sampling(kliegl2.sm, data = dat2,
+                        chains = 1, chain_id = i, seed = 12345))
> kliegl2.sf <- sflist2stanfit(sflist)
> print(kliegl2.sf)
```

The results look like this:

```
Stan:
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.
                mean se_mean
beta[1]        382.1     0.8
```

```
beta[2]         32.1     0.2
beta[3]         14.1     0.1
beta[4]          2.9     0.1
sigma_u[1]      56.8     0.1
sigma_u[2]      23.6     0.1
sigma_u[3]       9.3     0.4
sigma_u[4]       9.5     0.2
sigma_e         69.8     0.0

lmer:
          Estimate Std. Error t value
beta[1]   389.73   7.15    54.5
beta[2]   33.78    3.31    10.2
beta[3]   13.98    2.32     6.0
beta[4]   2.75     2.22     1.2
                         Var     sd
# id       (Intercept) 3098.3   55.66
#          c1           550.8   23.47     0.603
#          c2           121.0   11.00    -0.129 -0.014
#          c3            93.2    9.65    -0.247 -0.846  0.376
# Residual             4877.1   69.84
```

These results are fairly similar to the lmer model's.

Here is another version of the Stan code, due to Sergio Polini:

```
data {
    int<lower=1> N;
    real rt[N];                     //outcome
    real c1[N];                     //predictor
    real c2[N];                     //predictor
    real c3[N];                     //predictor
    int<lower=1> I;                 //number of subjects
    int<lower=1, upper=I> id[N];    //subject id
    vector[4] mu_prior;             //vector of zeros passed in from R
}
transformed data {
    real ZERO;                      // like #define ZERO 0 in C/C++
    ZERO <- 0.0;
}
parameters {
    vector[4] beta;                 // intercept and slope
    vector[4] u[I];                 // random intercept and slopes
    real<lower=0> sigma_e;          // residual sd
    vector<lower=0>[4] sigma_u;     // subj sd
    corr_matrix[4] Omega;           // correlation matrix for random intercepts and slopes
```

```
}
transformed parameters {
    matrix[4,4] D;
    D <- diag_matrix(sigma_u);
}
model {
    matrix[4,4] L;
    matrix[4,4] DL;
    real mu[N]; // mu for likelihood
    //priors:
    beta ~ normal(0,50);
    sigma_e ~ normal(0,100);
    sigma_u ~ normal(0,100);
    Omega ~ lkj_corr(4.0);

    L <- cholesky_decompose(Omega);
    for (m in 1:4)
        for (n in 1:m)
            DL[m,n] <- L[m,n] * sigma_u[m];
    for (m in 1:4)
        for (n in (m+1):4)
            DL[m,n] <- ZERO;

    for (i in 1:I)                  // loop for subj random effects
        u[i] ~ multi_normal_cholesky(mu_prior, DL);

    for (n in 1:N) {
        mu[n] <- beta[1] + beta[2]*c1[n] + beta[3]*c2[n] + beta[4]*c3[n]
            + u[id[n], 1] + u[id[n], 2]*c1[n] + u[id[n], 3]*c2[n] + u[id[n], 4]*c3[n];
    }
    rt ~ normal(mu,sigma_e);        // likelihood
}
generated quantities {
    cov_matrix[4] Sigma;
    for (i in 1:4)                  // diagonal and lower triangle
        for (j in 1:i)
            Sigma[i,j] <- Omega[i,j] * sigma_u[i] * sigma_u[j];
    for (i in 1:3)                  // upper triangle
        for (j in (i+1):4)
            Sigma[i,j] <- Sigma[j,i];
}
```

11.10 Developing a workflow for Stan modeling

Sergio Polini uses the following conventions for modeling. This seems very sensible because every data analysis will have a very predictable format. Currently this is an issue because even leading statisticians like Andrew Gelman report (various blog entries) that they are unable to keep their files in an orderly enough manner to release them to the public domain. I myself have a great deal of trouble trying to maintain and release code related to published research.

Given the serious (non-)replicability issues we face in psycholinguistics, it would be helpful if we release all data and code on publication. The workflow shown below might make this job easier.

```
Sergio's method for maintaining data and code:
- "foo": model_name
- "foo.stan": model source
- "foo.data.R": data file
- "foo.sm": compiled model
- "foo.sm.RData": saved compiled model
- "foo.sf": final stanfit object
- "foo.sf.RData": eventual saved stanfit object

and a function:

mystan <- function(model, data=model, iter = 2000, seed = 12345, new=FALSE, ...) {
    stopifnot(require(parallel)) # should never happen!
    model.data <- paste(data, ".data.R", sep='')
    model.file <- paste(model, ".stan", sep='')
    model.sm <- paste(model, ".sm", sep='')
    model.saved <- paste(model.sm, ".RData", sep='')
    model.sf <- paste(model, ".sf", sep='')
    if (!file.exists(model.file))
        stop(paste(model.stan, "not found"))
    if (!file.exists(model.data))
        stop(paste(model.data, "not found"))
    if (new) { # you have a previous compiled model, but you've changed the source
        if (exists(model.sm))
            rm(list=model.sm, envir = .GlobalEnv)
        if (file.exists(model.saved))
            unlink(model.saved)
    }
    if (!exists(model.sm)) {
        if (file.exists(model.saved)) {
            # load the saved model and put its name in .GlobalEnv
            load(model.saved, envir = .GlobalEnv, verbose = FALSE)
```

```
        } else {
            rt <- stanc(model.file, model_name = model)
            sm <- stan_model(stanc_ret = rt)
            # create the compiled model variable name
            assign(model.sm, sm, envir = .GlobalEnv)
            save(list=model.sm, file = model.saved)
        }
    }
    sm <- get(model.sm) # sm <- the model to whom its variable name refers
    sflist <- mclapply(1:4, mc.cores = detectCores(),
                       function(i) sampling(sm, data = read_rdump(model.data),
                           chains = 1, chain_id = i,
                           iter = iter, seed = seed, ...))
    sf <- sflist2stanfit(sflist)
    # create the stanfit object variable name
    assign(model.sf, sf, .GlobalEnv)
}
```

An example session:

```
> library(rstan)
...
> source("mystan.R")
> N <- 100
> x <- rnorm(N,0,4)
> y <- 2 + 0.5 * x - 1.5 * x^2 + rnorm(N)
> stan_rdump(c("N", "x", "y"), file = "linreg.data.R")
> ls()
[1] "mystan" "N"      "x"      "y"
> mystan("linreg1", "linreg")
...
> ls()
[1] "linreg1.sf" "linreg1.sm" "mystan"     "N"          "x"
[6] "y"
> print(linreg1.sf)
...
> mystan("linreg2", "linreg")
...
> ls()
[1] "linreg1.sf" "linreg1.sm" "linreg2.sf" "linreg2.sm" "mystan"
[6] "N"          "x"          "y"
> print(linreg2.sf)
...
```

12
Model checking

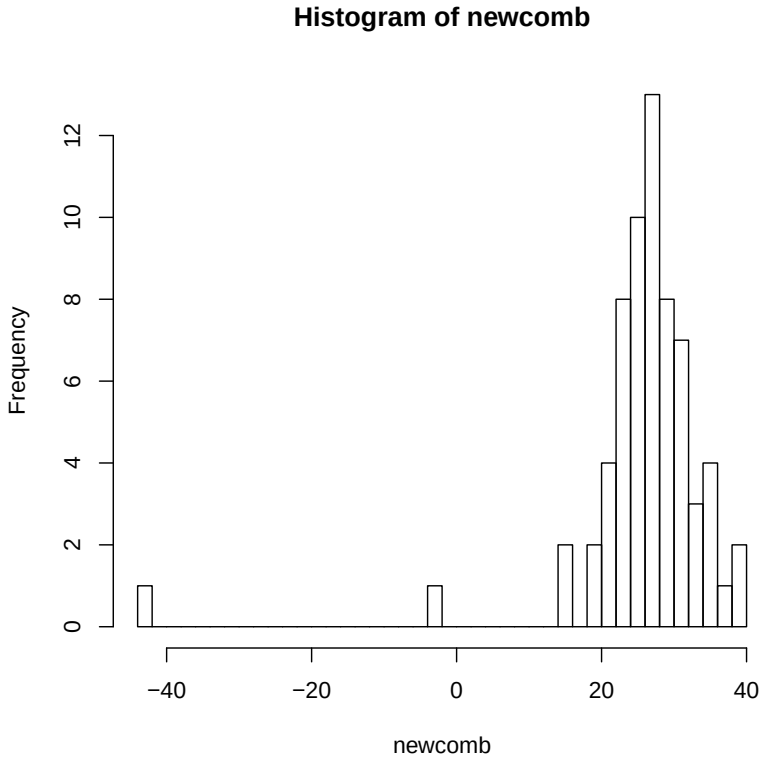
12.1 Example: Newcomb's speed of light data

Simon Newcomb measured the time (recorded as deviations from 24,800 nanosecs) it takes for light to travel 7442 meters.

```
> newcomb <-
+ c(28,26,33,24,34,-44,27,16,40,-2,
+ 29,22,24,21,25,30,23,29,31,19,
+ 24,20,36,32,36,28,25,21,28,29,
+ 37,25,28,26,30,32,36,26,30,22,
+ 36,23,27,27,28,27,31,27,26,33,
+ 26,32,32,24,39,28,24,25,32,25,
+ 29,27,28,29,16,23)
```

Let's look at the data. Two points looks like outliers:

```
> hist(newcomb,breaks=50)
```



The JAGS model is:

```
> # Data as a list
> a.dat <- list( y=newcomb, I=length(newcomb) )
> # Inits as a list of lists
> a.ini <- list( list( alpha=20, sigma=8 ),
+                list( alpha=23, sigma=10 ),
+                list( alpha=26, sigma=12 ) )
> # Names of the parameters to monitor
> a.par <- c("alpha","sigma" )

> # The bugs-code for the model
> cat( "model
+       {
+       for( i in 1:I )
+          {
+          y[i] ~ dnorm(mu[i],tau)
+          mu[i] <- alpha
+          }
+       alpha ~ dnorm(0, 1.0E-6)
+       sigma ~ dunif(0,100)
+       tau  <- 1/pow(sigma,2)
```

```
+       }",
+       file="JAGSmodels/light.jag" )

> library(rjags)
> a.mod <- jags.model( file = "JAGSmodels/light.jag",
+                      data = a.dat,
+                 n.chains = 3,
+                    inits = a.ini,
+                  n.adapt = 20000,
+                  quiet=T)
> # Sampling from the posterior
> a.res <- coda.samples( a.mod,
+                           var = a.par,
+                        n.iter = 20000,
+                          thin = 10 )
```

And here are the results:

```
> summary(a.res)

Iterations = 20010:40000
Thinning interval = 10
Number of chains = 3
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

      Mean    SD Naive SE Time-series SE
alpha 26.2 1.356   0.0175         0.0177
sigma 11.0 0.975   0.0126         0.0129

2. Quantiles for each variable:

       2.5%  25%  50%  75% 97.5%
alpha 23.63 25.3 26.2 27.2  28.9
sigma  9.22 10.3 10.9 11.6  13.0
```

12.2 Model checking

Gelman et al 2014 say (p. 143):

> If [I guess they mean: if and only if] the model fits, then replicated data generated under the model should look similar to observed data. . . . the observed data should look plausible under the posterior predictive distribution.

We will follow Gelman et al and track the distribution of the minimum values under the posterior predictive distribution. Let's modify the code to track (properties of) posterior predictions:

```
> a.par <- c("alpha","sigma", "smallest","mn")
> # The jags code for the model
> cat( "model
+       {
+       for( i in 1:I )
+          {
+          y[i] ~ dnorm(mu[i],tau)
+          mu[i] <- alpha
+      ##generates predicted values:
+          y.pred[i] ~ dnorm(mu[i],tau)
+          }
+       alpha ~ dnorm(0, 1.0E-6)
+       sigma ~ dunif(0,100)
+       tau  <- 1/pow(sigma,2)
+       smallest <- min(y.pred)
+       mn <- mean(y.pred)
+       }",
+       file="JAGSmodels/light2.jag" )
```

The posterior distribution from this model will turn out to not reflect the observed data. We can change the model from normal to t-distribution with df=2 to model extreme values.

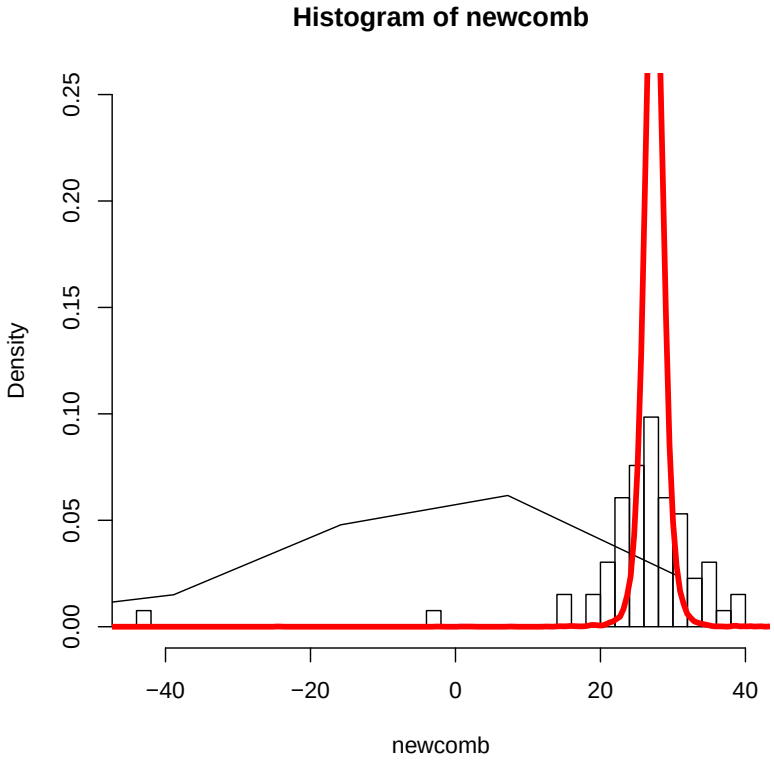
```
> cat( "model
+       {
+       for( i in 1:I )
+          {
+          y[i] ~ dt(mu[i],tau,2)
+          mu[i] <- alpha
+      ##GENERATES predicted values:
+          #y.pred[i] ~ dnorm(mu[i],tau)
+           y.pred[i] ~ dt(mu[i],tau,2)
+          }
+       alpha ~ dnorm(0, 1.0E-6)
+       sigma ~ dunif(0,100)
+       tau  <- 1/pow(sigma,2)
+       smallest <- min(y.pred)
+       mn <- mean(y.pred)
+       }",
+       file="JAGSmodels/tdistrlight2.jag" )
```

Here are the models and posterior samples for the two models above.

```
> # Model compilation:
> a.mod <- jags.model( file = "JAGSmodels/light2.jag",
+                      data = a.dat,
+                 n.chains = 3,
+                    inits = a.ini,
+                  n.adapt = 20000,
+                  quiet=T)
> a.res <- coda.samples( a.mod,
+                           var = a.par,
+                         n.iter = 40000,
+                           thin = 10 )

> summary(a.res)

Iterations = 20010:60000
Thinning interval = 10
Number of chains = 3
Sample size per chain = 4000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

           Mean    SD Naive SE Time-series SE
alpha    26.224 1.355  0.01237        0.01237
mn       26.227 1.920  0.01752        0.01742
sigma    10.969 0.978  0.00893        0.00893
smallest  0.419 5.629  0.05139        0.05072

2. Quantiles for each variable:

           2.5%   25%    50%   75% 97.5%
alpha     23.60 25.32 26.202 27.13  28.9
mn        22.49 24.95 26.219 27.49  30.0
sigma      9.28 10.27 10.890 11.58  13.1
smallest -12.06 -3.01  0.958  4.38  10.1
```

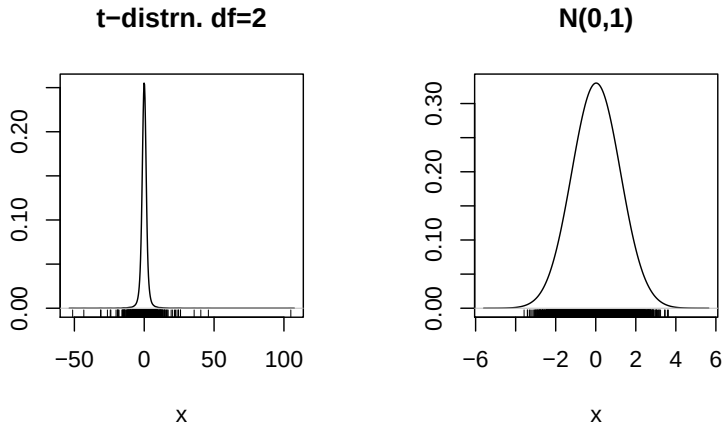
The distribution of the minimum values in the posterior predictive distribution compared with the observed distribution:

```
[1] "alpha"    "mn"       "sigma"    "smallest"
```



You can see that the t-distribution with df=2 has fatter tails than a normal distribution, and as a result models the extreme values observed.

```
> ## a smoothing function needed below:
> kdeq3m.fun <- function(x, mult = 2,mainlab)
+ { iq <- IQR(x)
+ wid <- mult * iq
+ n <- length(x)
+ plot(density(x, width = wid), xlab = "x", ylab = "", type = "l",main=mainlab)
+ rug(x)
+ out <- list(Sample.size = n, IQR = iq, Smoothing.par = wid)
+ #out
+ }
```



12.3 Model checking with raw reading times (Gibson and Wu data)

Fit the Gibson and Wu data using **raw reading times** as your dependent variable. **Fit a varying intercepts model only (no varying slopes)**.

Obtain the distribution of the **largest** values from the posterior predictive distribution and find out whether this distribution of largest values is realistic given the data.

Here is a solution to this task:

```
> data<-read.table("data/gibsonwu2012data.txt",header=T)
> ## head(data)
> data$so <- ifelse(
+   data$type%in%c("subj-ext"),-0.5,0.5)
> library(lme4)
> summary(m1 <- lmer(rt~so+(1|item)+(1|subj),
+           subset(data,region=="headnoun")))

Linear mixed model fit by REML ['lmerMod']
Formula: rt ~ so + (1 | item) + (1 | subj)
```

```
   Data: subset(data, region == "headnoun")

REML criterion at convergence: 8499.5

Random effects:
 Groups   Name        Variance Std.Dev.
 subj     (Intercept)  21738   147
 item     (Intercept)  22396   150
 Residual             314128   560
Number of obs: 547, groups: subj, 37; item, 15

Fixed effects:
            Estimate Std. Error t value
(Intercept)    548.4       51.6   10.64
so            -120.4       48.0   -2.51

Correlation of Fixed Effects:
   (Intr)
so -0.003

> (sigma.u<-as.integer(attr(VarCorr(m1)$subj,"stddev")))

[1] 147

> (sigma.w<-as.integer(attr(VarCorr(m1)$item,"stddev")))

[1] 149

> ## estimated residual sd:
> (sigma.e<-as.integer(attr(VarCorr(m1),"sc")))

[1] 560

> (beta<-fixef(m1))

(Intercept)          so
     548.43     -120.39

> headnoun<-subset(data,region=="headnoun")
> headnoun.dat <- list(subj=
+                 sort(as.integer(
+                       factor(headnoun$subj) )),
+                      item=
+                        sort(as.integer(
+                         factor(headnoun$item) )),
+                    rt = headnoun$rt,
+                     so = headnoun$so,
```

```
+                     N = nrow(headnoun),
+                     I =
+                     length( unique(headnoun$subj) ),
+                     K =
+                     length( unique(headnoun$item) )
+                       )
> ## data for another model, separate models for SR and OR:
> hnoun.sr<-subset(headnoun,type=="subj-ext")
> hnoun.or<-subset(headnoun,type=="obj-ext")
> N.sr<-dim(hnoun.sr)[1]## 272
> N.or<-dim(hnoun.or)[1]## 275
> headnoun.separate<-rbind(hnoun.sr,hnoun.or)
> headnounsep.dat <- list(subj=
+                 sort(as.integer(
+                       factor(headnoun.separate$subj) )),
+                      item=
+                        sort(as.integer(
+                         factor(headnoun.separate$item) )),
+                    rt = headnoun.separate$rt,
+                     so = headnoun.separate$so,
+                     N = nrow(headnoun.separate),
+                     I =
+                     length( unique(headnoun.separate$subj) ),
+                     K =
+                     length( unique(headnoun.separate$item) ),
+                     N.sr=N.sr,
+                     N.or=N.or
+                       )
> ## random initial values:
> gen.inits<-function(s){
+   init<-rnorm(1,mean=s,sd=50)
+   round(abs(init),digits=0)
+ }
> gen.inits.beta<-function(b){
+   k<-length(b)
+   store<-rep(NA,k)
+   for(i in 1:k){
+   init<-rnorm(1,mean=b[i],sd=50)
+   store[i]<-init
+   }
+   round(store,digits=0)
+ }
> ## testing:
> #gen.inits.beta(beta)
```

```
> #gen.inits(sigma.e)
>
> generate<-function(){
+   list( sigma.e = gen.inits(sigma.e),
+         sigma.u = gen.inits(sigma.u),
+         sigma.w = gen.inits(sigma.w),
+         beta = gen.inits.beta(beta))
+ }
> headnoun.ini <- list(generate(),
+                      generate(),
+                      generate(),
+                      generate())
> cat("
+ # Fixing data to be used in model definition
+ model
+    {
+    # The model for each observational unit
+   #   (each row is a subject's data point)
+      for( j in 1:N )
+      {
+      mu[j] <- beta[1] + beta[2] * ( so[j] )  + u[subj[j]] + w[item[j]]
+      rt[j] ~ dnorm( mu[j], tau.e )
+
+     ##generate predicted values:
+      rt.pred[j] ~ dnorm(mu[j],tau.e)
+      }
+
+    # Random effects for each subject
+      for( i in 1:I )
+      {
+      u[i] ~ dnorm(0,tau.u)
+      }
+
+      # Random effects for each item
+      for( k in 1:K )
+      {
+      w[k] ~ dnorm(0,tau.w)
+      }
+
+    # Uninformative priors:
+
+    # Fixed effect intercept and slope
+      beta[1] ~ dnorm(0.0,1.0E-5)
+      beta[2] ~ dnorm(0.0,1.0E-5)
```

```
+
+    # Residual (within-person) variance
+      tau.e <- pow(sigma.e,-2)
+      sigma.e  ~ dunif(0,1000)
+
+    # Between-person variation
+      tau.u <- pow(sigma.u,-2)
+      sigma.u  ~ dunif(0,500)
+
+      # Between-item variation
+      tau.w <- pow(sigma.w,-2)
+      sigma.w  ~ dunif(0,500)
+
+     ## track predicted values:
+     smallest <- min(rt.pred)
+     mn      <- mean(rt.pred)
+     largest  <- max(rt.pred)
+
+    }",
+      file="JAGSmodels/rtgwheadnouncrossedrandom.jag" )
> ## model with separate generating functions for SRs and ORs:
> cat("
+ # Fixing data to be used in model definition
+ model
+    {
+   # Residual (within-person) variance
+      tausr.e <- pow(sigmasr.e,-2)
+      tauor.e <- pow(sigmaor.e,-2)
+      sigmasr.e  ~ dunif(0,1000)
+      sigmaor.e  ~ dunif(0,1000)
+
+    # The model for each observational unit
+   #   (each row is a subject's data point)
+      for( j in 1:N )
+      {
+      mu[j] <- beta[1] + beta[2] * ( so[j] )  + u[subj[j]] + w[item[j]]
+
+     ##generate predicted values:
+      #rt.pred[j] ~ dnorm(mu[j],tausr.e)
+      }
+     for(s in 1:N.sr){
+     rt[s] ~ dnorm( mu[s], tausr.e)
+     }
+     for(r in (N.sr+1):(N.sr+N.or)){
```

```
+     rt[r] ~ dnorm( mu[r], tauor.e)
+     }
+
+    # Random effects for each subject
+      for( i in 1:I )
+      {
+      u[i] ~ dnorm(0,tau.u)
+      }
+
+      # Random effects for each item
+      for( k in 1:K )
+      {
+      w[k] ~ dnorm(0,tau.w)
+      }
+
+    # Uninformative priors:
+
+    # Fixed effect intercept and slope
+      beta[1] ~ dnorm(0.0,1.0E-5)
+      beta[2] ~ dnorm(0.0,1.0E-5)
+
+
+    # Between-person variation
+      tau.u <- pow(sigma.u,-2)
+      sigma.u  ~ dunif(0,500)
+
+      # Between-item variation
+      tau.w <- pow(sigma.w,-2)
+      sigma.w  ~ dunif(0,500)
+
+     ## track predicted values:
+   #  smallest <- min(rt.pred)
+   #  mn      <- mean(rt.pred)
+   #  largest  <- max(rt.pred)
+    }",
+      file="JAGSmodels/rtgwheadnouncrossedrandomsep.jag" )
> ## version 2: with t distribution
> ## model with separate generating functions for SRs and ORs:
> cat("
+ # Fixing data to be used in model definition
+ model
+    {
+   # Residual (within-person) variance
+      tausr.e <- pow(sigmasr.e,-2)
```

```
+      tauor.e <- pow(sigmaor.e,-2)
+      sigmasr.e  ~ dunif(0,1000)
+      sigmaor.e  ~ dunif(0,1000)
+
+    # The model for each observational unit
+   #   (each row is a subject's data point)
+      for( j in 1:N )
+      {
+      mu[j] <- beta[1] + beta[2] * ( so[j] )  + u[subj[j]] + w[item[j]]
+
+     ##generate predicted values:
+      #rt.pred[j] ~ dnorm(mu[j],tausr.e)
+      }
+     for(s in 1:N.sr){
+     #rt[s] ~ dnorm( mu[s], tausr.e)
+     ## truncated t-distribution with df=2
+     rt[s] ~ dt( mu[s], tausr.e,2) T(0,)
+     }
+     for(r in (N.sr+1):(N.sr+N.or)){
+     #rt[r] ~ dnorm( mu[r], tauor.e)
+     rt[r] ~ dt( mu[r], tauor.e,2) T(0,)
+     }
+
+    # Random effects for each subject
+      for( i in 1:I )
+      {
+      u[i] ~ dnorm(0,tau.u)
+      }
+
+      # Random effects for each item
+      for( k in 1:K )
+      {
+      w[k] ~ dnorm(0,tau.w)
+      }
+
+    # Uninformative priors:
+
+    # Fixed effect intercept and slope
+      beta[1] ~ dnorm(0.0,1.0E-5)
+      beta[2] ~ dnorm(0.0,1.0E-5)
+
+
+    # Between-person variation
+      tau.u <- pow(sigma.u,-2)
```

```
+      sigma.u  ~ dunif(0,500)
+
+      # Between-item variation
+      tau.w <- pow(sigma.w,-2)
+      sigma.w  ~ dunif(0,500)
+
+     ## track predicted values:
+   #  smallest <- min(rt.pred)
+   #  mn      <- mean(rt.pred)
+   #  largest  <- max(rt.pred)
+    }",
+      file="JAGSmodels/rtgwheadnouncrossedrandomsept.jag" )
> ## ex-gaussian:
> cat("
+ # Fixing data to be used in model definition
+ model
+    {
+   # Residual (within-person) variance
+      tau.e ~ dgamma(0.001,0.001)
+ ## tau.e = 1/sigma.e^2
+ ## sigma.e^2 = 1/tau.e
+ ## sigma.e = 1/tau.e^{1/2}
+ #     tau.e <- pow(sigma.e,-2)
+ #     sigma.e  ~ dunif(0,100)
+       sigma.e <- pow(tau.e,-0.5)
+
+    # The model for each observational unit
+   #   (each row is a subject's data point)
+      for( j in 1:N )
+      {
+      mu[j] <- beta[1] + beta[2] * ( so[j] )  + u[subj[j]] + w[item[j]]
+
+     ##generate predicted values:
+      #rt.pred[j] ~ dnorm(mu[j],tausr.e)
+      }
+     for(s in 1:N.sr){
+      ex[s] ~ dexp(rate.sr) ## rate is 1/lambda.sr
+      mn[s] <- mu[s] + ex[s]
+      rt[s] ~ dnorm(mn[s],tau.e)T(0,)
+     }
+     for(r in (N.sr+1):(N.sr+N.or)){
+      ex[r] ~ dexp(rate.or) ## rate is 1/lambda.or
+      mn[r] <- mu[r] + ex[r]
+      rt[r] ~ dnorm(mn[r],tau.e)T(0,)
```

```
+     }
+
+    # Random effects for each subject
+      for( i in 1:I )
+      {
+      u[i] ~ dnorm(0,tau.u)
+      }
+
+      # Random effects for each item
+      for( k in 1:K )
+      {
+      w[k] ~ dnorm(0,tau.w)
+      }
+
+    # Uninformative priors:
+
+    # Fixed effect intercept and slope
+      beta[1] ~ dnorm(0.0,1.0E-5)
+      beta[2] ~ dnorm(0.0,1.0E-5)
+
+
+    # Between-person variation
+ #     tau.u <- pow(sigma.u,-2)
+ #     sigma.u  ~ dunif(0,100)
+       tau.u ~ dgamma(0.001,0.001)
+       sigma.u <- pow(tau.u,-0.5)
+
+      # Between-item variation
+ #     tau.w <- pow(sigma.w,-2)
+ #     sigma.w  ~ dunif(0,100)
+       tau.w ~ dgamma(0.001,0.001)
+       sigma.w <- pow(tau.w,-0.5)
+
+     ## priors for rates:
+     rate.sr ~ dgamma(0.001,0.001)
+     rate.or ~ dgamma(0.001,0.001)
+
+     ## track predicted values:
+   #  smallest <- min(rt.pred)
+   #  mn      <- mean(rt.pred)
+   #  largest  <- max(rt.pred)
+    }",
+      file="JAGSmodels/rtgwheadnouncrossedrandomexg.jag" )
> ## ex-gaussian version 2, priors on sds are unif:
```

```
> cat("
+ # Fixing data to be used in model definition
+ model
+    {
+     # The model for each observational unit
+   #   (each row is a subject's data point)
+      for( j in 1:N )
+      {
+      mu[j] <- beta[1] + beta[2] * ( so[j] )  + u[subj[j]] + w[item[j]]
+     }
+     for(s in 1:N.sr){
+      ex[s] ~ dexp(rate.sr) ## rate is 1/lambda.sr
+      mn[s] <- mu[s] + ex[s]
+      rt[s] ~ dnorm(mn[s],tau.e)T(0,)
+     }
+     for(r in (N.sr+1):(N.sr+N.or)){
+      ex[r] ~ dexp(rate.or) ## rate is 1/lambda.or
+      mn[r] <- mu[r] + ex[r]
+      rt[r] ~ dnorm(mn[r],tau.e)T(0,)
+     }
+
+    # Random effects for each subject
+      for( i in 1:I )
+      {
+      u[i] ~ dnorm(0,tau.u)
+      }
+
+      # Random effects for each item
+      for( k in 1:K )
+      {
+      w[k] ~ dnorm(0,tau.w)
+      }
+
+    # Uninformative priors:
+
+    # Fixed effect intercept and slope
+      beta[1] ~ dnorm(0.0,1.0E-5)
+      beta[2] ~ dnorm(0.0,1.0E-5)
+
+   # Residual (within-person) variance
+      tau.e <- pow(sigma.e,-2)
+      sigma.e  ~ dunif(0,1000)
+
+    # Between-person variation
```

```
+      tau.u <- pow(sigma.u,-2)
+      sigma.u  ~ dunif(0,200)
+
+      # Between-item variation
+      tau.w <- pow(sigma.w,-2)
+      sigma.w  ~ dunif(0,200)
+
+     ## priors for rates:
+     rate.sr ~ dgamma(0.001,0.001)
+     rate.or ~ dgamma(0.001,0.001)
+
+     ## track predicted values:
+   #  smallest <- min(rt.pred)
+   #  mn      <- mean(rt.pred)
+   #  largest  <- max(rt.pred)
+    }",
+      file="JAGSmodels/rtgwheadnouncrossedrandomexgv2.jag" )
> ## visualizing the ex-gaussian:
> samp<-rep(NA,10000)
> for(i in 1:10000){
+ ex<-rexp(1,0.001)
+ #ex<-0
+ mn<-rnorm(1,mean=500,sd=40)
+ samp[i]<-ex+mn
+ }
> hist(samp)
> ## visualizing the t-distribution:
> samp<-rep(NA,10000)
> for(i in 1:10000){
+ samp[i]<-500+rt(1,df=2)
+ }
> hist(samp)
> track.variables<-c("beta","sigma.e",
+                    "sigma.u","sigma.w","smallest","mn","largest")
> track.variables.sep<-c("beta","sigmasr.e","sigmaor.e",
+                    "sigma.u","sigma.w")
> track.variables.exg<-c("beta","sigma.e",
+                    "sigma.u","sigma.w","rate.sr","rate.or")
> library(rjags)
> headnoun.mod <- jags.model(
+   file="JAGSmodels/rtgwheadnouncrossedrandom.jag",
+   data = headnoun.dat,
+   n.chains = 4,
+ #  inits = headnoun.ini,
```

```
+   n.adapt =5000, quiet=T)
> headnoun.mod.sep <- jags.model(
+   file="JAGSmodels/rtgwheadnouncrossedrandomsep.jag",
+   data = headnounsep.dat,
+   n.chains = 4,
+   n.adapt =5000, quiet=T)
> headnoun.mod.sept <- jags.model(
+   file="JAGSmodels/rtgwheadnouncrossedrandomsept.jag",
+   data = headnounsep.dat,
+   n.chains = 4,
+   n.adapt =5000, quiet=T)
> headnoun.mod.exg <- jags.model(
+   file="JAGSmodels/rtgwheadnouncrossedrandomexg.jag",
+   data = headnounsep.dat,
+   n.chains = 4,
+   n.adapt =20000, quiet=T)
> headnoun.mod.exgv2 <- jags.model(
+   file="JAGSmodels/rtgwheadnouncrossedrandomexgv2.jag",
+   data = headnounsep.dat,
+   n.chains = 4,
+   n.adapt =2000, quiet=T)
> headnoun.res <- coda.samples(headnoun.mod,
+                 var = track.variables,
+                 n.iter = 10000,
+                 thin = 20 )
> summary(headnoun.res)$statistics[2,1:2]

    Mean       SD
-119.599   50.223

> summary(headnoun.res)$quantiles[2,c(1,3,5)]

    2.5%      50%    97.5%
-214.964 -120.189  -24.693

> headnounsep.res <- coda.samples(headnoun.mod.sep,
+                 var = track.variables.sep,
+                 n.iter = 10000,
+                 thin = 20 )
> headnounsept.res <- coda.samples(headnoun.mod.sept,
+                 var = track.variables.sep,
+                 n.iter = 10000,
+                 thin = 20 )
> headnounexg.res <- coda.samples(headnoun.mod.exg,
+                 var = track.variables.exg,
+                 n.iter = 40000,
```

```
+                 thin = 20 )
> headnoun.mod.exgv2 <- jags.model(
+   file="JAGSmodels/rtgwheadnouncrossedrandomexgv2.jag",
+   data = headnounsep.dat,
+   n.chains = 4,
+   n.adapt =50000, quiet=T)
> headnounexgv2.res <- coda.samples(headnoun.mod.exgv2,
+                 var = track.variables.exg,
+                 n.iter = 100000,
+                 thin = 100 )
> gelman.diag(headnounexgv2.res)

Potential scale reduction factors:

        Point est. Upper C.I.
beta[1]       1.00       1.00
beta[2]       1.00       1.00
rate.or       1.24       1.73
rate.sr       1.04       1.11
sigma.e       1.00       1.00
sigma.u       1.00       1.00
sigma.w       1.00       1.00

Multivariate psrf

1.14

> plot(headnounexgv2.res)
> summary(headnounexgv2.res)

Iterations = 50100:150000
Thinning interval = 100
Number of chains = 4
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

        Mean    SD Naive SE Time-series SE
beta[1] -779 108.2    1.710          1.741
beta[2] -386 166.8    2.637          2.662
rate.or  234 425.1    6.722         26.271
rate.sr  264 457.2    7.229         35.218
sigma.e  982  15.9    0.252          0.251
sigma.u  142  51.5    0.815          1.275
```

```
sigma.w  117  57.2    0.905          1.048

2. Quantiles for each variable:

            2.5%    25%    50%  75% 97.5%
beta[1] -983.183 -855.1 -778.4 -705  -567
beta[2] -710.722 -497.4 -387.9 -276   -60
rate.or    1.050   12.3   48.7  280  1433
rate.sr    0.641   17.1   75.8  305  1620
sigma.e  941.176  974.6  986.6  994   999
sigma.u   16.642  111.3  158.2  183   199
sigma.w    7.942   70.4  125.2  167   197
```

```
> ## Gelman-Rubin convergence diagnostic:
> #gelman.diag(headnoun.res)
>
> plot(headnoun.res)
> plot(headnounsep.res)
> plot(headnounsept.res)
> plot(headnounexg.res)
> gelman.diag(headnounsep.res)
```

```
Potential scale reduction factors:

          Point est. Upper C.I.
beta[1]        1.001      1.005
beta[2]        1.000      1.003
sigma.u        1.006      1.018
sigma.w        1.005      1.016
sigmaor.e      0.999      0.999
sigmasr.e      1.000      1.001

Multivariate psrf

1.01
```

```
> gelman.diag(headnounsept.res)
```

```
Potential scale reduction factors:

          Point est. Upper C.I.
beta[1]        1.001       1.00
beta[2]        0.999       1.00
sigma.u        1.000       1.00
sigma.w        1.011       1.03
sigmaor.e      1.000       1.00
```

```
sigmasr.e      1.000       1.00

Multivariate psrf

1.01

> gelman.diag(headnounexg.res)

Potential scale reduction factors:

        Point est. Upper C.I.
beta[1]       2.64       4.45
beta[2]       1.13       1.36
rate.or       1.28       2.01
rate.sr       1.11       1.23
sigma.e       1.57       3.12
sigma.u       1.36       3.32
sigma.w       1.37       2.58

Multivariate psrf

2.48

> summary(headnounsep.res)$statistics[2,1:2]

   Mean      SD
-108.24   63.44

> summary(headnounsep.res)$quantiles[2,c(1,3,5)]

    2.5%      50%    97.5%
-231.938 -110.233   25.604

> summary(headnounsept.res)$statistics[2,1:2]

  Mean     SD
-5.377 28.789

> summary(headnounsept.res)$quantiles[2,c(1,3,5)]

    2.5%      50%    97.5%
-61.9138  -5.5512  50.5107

> summary(headnounexg.res)$statistics[,1:2]

             Mean        SD
beta[1]  -344.476    565.45
beta[2]   -77.041    323.53
```

```
rate.or    91.707    255.75
rate.sr   114.924    275.82
sigma.e  4049.941   3243.99
sigma.u   813.570   2348.68
sigma.w 53895.888 100491.02
```

```
> summary(headnounexg.res)$quantiles[,c(1,3,5)]
```

```
               2.5%        50%     97.5%
beta[1] -1.4715e+03  -210.5074    537.65
beta[2] -6.7980e+02   -86.8193    573.43
rate.or  7.7333e-02     6.2703    730.03
rate.sr  2.4988e-01    18.9507    948.60
sigma.e  1.0554e+03  3276.4211  14374.48
sigma.u  4.6587e-02    63.9492   6906.05
sigma.w  1.1596e-01 20397.1766 412901.86
```

```
> summary(lmer(rt~so+(1|subj)+(1|item),headnoun))
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: rt ~ so + (1 | subj) + (1 | item)
   Data: headnoun

REML criterion at convergence: 8499.5

Random effects:
 Groups   Name        Variance Std.Dev.
 subj     (Intercept)  21738   147
 item     (Intercept)  22396   150
 Residual             314128   560
Number of obs: 547, groups: subj, 37; item, 15

Fixed effects:
            Estimate Std. Error t value
(Intercept)    548.4       51.6   10.64
so            -120.4       48.0   -2.51

Correlation of Fixed Effects:
   (Intr)
so -0.003
```

```
> mcmcChain <- as.matrix( headnoun.res )
> ## max:
> hist(mcmcChain[,3],xlim=c(0,8000))
> abline(v=max(data$rt))
>
> #summary(headnoun$rt)
```

Incidentally, an error message you might see in the Gibson and Wu varying intercepts and slopes model when using the Gelman-Rubin convergence diagnostic:

```
Error in chol.default(W) :
  the leading minor of order 3 is not positive definite
```

This message occurs because two of the parameters being monitored are highly correlated.

Just set multivariate=FALSE in the gelman.diag function.

Another problem is convergence failure. The solution to this is supposed to be easy: run longer simulations. Following Gelman et al (2014), we discard the first half of all simulations as burn in (Gelman et al call it warm up). I will add a lot more detail on convergence failure in a later version of these notes.

Here is an exercise: Try to find out how good the model fit is using the negative reciprocal reading time and the distribution of the maximum value from the posterior predictive distribution.

12.4 Modeling in the raw reading time scale, assuming a Cauchy distribution for errors

To be elaborated on later.

```
> data<-read.table("data/gibsonwu2012data.txt",header=T)
> ## convert to reciprocal rt:
> data$rrt<- -1000/data$rt
> ## contrast coding:
> data$so <- ifelse(
+   data$type%in%c("subj-ext"),-0.5,0.5)
> ## reference values from lmer:
> library(lme4)
> (m1 <- lmer(rrt~so+(1|subj),
+             subset(data,region=="headnoun")))

Linear mixed model fit by REML ['lmerMod']
Formula: rrt ~ so + (1 | subj)
   Data: subset(data, region == "headnoun")
REML criterion at convergence: 1633.3
Random effects:
 Groups   Name        Std.Dev.
 subj     (Intercept) 0.604
 Residual             1.009
Number of obs: 547, groups: subj, 37
Fixed Effects:
```

```
(Intercept)           so
    -2.6698      -0.0802

> headnoun<-subset(data,region=="headnoun")
> ## Stan ready data:
> headnoun.dat <- list(subj=
+                      sort(as.integer(
+                      factor(headnoun$subj) )),
+                      item=sort(as.integer(
+                           factor(headnoun$item) )),
+                       rrt = headnoun$rt,
+                       so = headnoun$so,
+                       N = nrow(headnoun),
+                       I =
+                         length( unique(headnoun$subj) ),
+                       K =
+                         length( unique(headnoun$item) )
+ )
> ## Stan code:
> varying.int_code<-'
+   data {
+ int<lower=1> N; //no. of rows
+ real so[N];     // predictor
+ real rrt[N];   //outcome
+ int<lower=1> I;   //number of subjects
+ int<lower=1, upper=I> subj[N];  //subject id
+ int<lower=1> K;   //number of items
+ int<lower=1, upper=K> item[N];  //item id
+ }
+ parameters {
+ real alpha;      // intercept
+ real beta;       // slope
+ real u[I];   // random intercept subj
+ real w[K];   // random intercept item
+ real<lower=0> sigma_e;  // residual variance
+ real<lower=0> sigma_u;   //  subject var
+ real<lower=0> sigma_w;   //  item var
+ }
+ model {
+ real mu[N];
+ for (i in 1:N) {
+ mu[i] <- alpha + beta*so[i] + u[subj[i]]+w[item[i]];
+ }
+ rrt ~ cauchy(mu,sigma_e);    // likelihood
```

```
+ alpha ~ normal(0,800); // was 5
+ beta ~ normal(0,300); // was 5
+ u ~ normal(0,sigma_u);
+ w ~ normal(0,sigma_w);
+ ## could have used uniform:
+ sigma_u ~ cauchy(0,300);
+ sigma_w ~ cauchy(0,300);
+ sigma_e ~ cauchy(0,800);
+ }
+ '
> fit <- stan(model_code = varying.int_code,
+             data = headnoun.dat,
+             iter = 500, chains = 2)

TRANSLATING MODEL 'varying.int_code' FROM Stan CODE TO C++ CODE NOW.
COMPILING THE C++ CODE FOR MODEL 'varying.int_code' NOW.
SAMPLING FOR MODEL 'varying.int_code' NOW (CHAIN 1).

Iteration:   1 / 500 [  0%]  (Warmup)
Iteration:  50 / 500 [ 10%]  (Warmup)
Iteration: 100 / 500 [ 20%]  (Warmup)
Iteration: 150 / 500 [ 30%]  (Warmup)
Iteration: 200 / 500 [ 40%]  (Warmup)
Iteration: 250 / 500 [ 50%]  (Warmup)
Iteration: 300 / 500 [ 60%]  (Sampling)
Iteration: 350 / 500 [ 70%]  (Sampling)
Iteration: 400 / 500 [ 80%]  (Sampling)
Iteration: 450 / 500 [ 90%]  (Sampling)
Iteration: 500 / 500 [100%]  (Sampling)
Elapsed Time: 11.844 seconds (Warm-up)
              4.61307 seconds (Sampling)
              16.457 seconds (Total)

SAMPLING FOR MODEL 'varying.int_code' NOW (CHAIN 2).

Iteration:   1 / 500 [  0%]  (Warmup)
Iteration:  50 / 500 [ 10%]  (Warmup)
Iteration: 100 / 500 [ 20%]  (Warmup)
Iteration: 150 / 500 [ 30%]  (Warmup)
Iteration: 200 / 500 [ 40%]  (Warmup)
Iteration: 250 / 500 [ 50%]  (Warmup)
Iteration: 300 / 500 [ 60%]  (Sampling)
Iteration: 350 / 500 [ 70%]  (Sampling)
Iteration: 400 / 500 [ 80%]  (Sampling)
```

```
Iteration: 450 / 500 [ 90%]  (Sampling)
Iteration: 500 / 500 [100%]  (Sampling)
Elapsed Time: 8.68756 seconds (Warm-up)
              3.25862 seconds (Sampling)
              11.9462 seconds (Total)
```

```
> mcmcChain<-as.matrix(fit)
> hist(mcmcChain[,2])
>
> #plot(fit)
```

13
Bayesian statistics course notes from Sheffield

13.1 Eliciting priors

13.1.1 Location and dispersion elicitation

Elicit m, elicit 95% credible region, i.e., [m-a,m+a], then $a/2$ is SD, and $(a/2)^2$ (assuming normal distrn).

13.1.2 Bisection method

Elicit 25th, 50th, 75th percentile. More generally, to elicit an N(m,v) distribution to someone's beliefs about a parameter, in theory we need two percentiles, say $P(\theta < x_1) = p_1$ and $P(\theta > x_1) = p_2$. We can then find m and v as solutions of

$$\int_{-\infty}^{x_1} \frac{1}{\sqrt{2\pi v}} \exp -\frac{1}{2v}(m-\theta)^2 \, d\theta = p_1 \tag{13.1}$$

$$\int_{-\infty}^{x_2} \frac{1}{\sqrt{2\pi v}} \exp -\frac{1}{2v}(m-\theta)^2 \, d\theta = p_1 \tag{13.2}$$

Because people are inaccurate in delivering percentiles, eliciting several percentiles and minimizing the following quantity may be better:

$$\sum [\int_{-\infty}^{x_i} \frac{1}{\sqrt{2\pi v}} \exp -\frac{1}{2v}(m-\theta)^2 \, d\theta - p_i] \tag{13.3}$$

13.2 Exercises

13.2.1 Ex 4.2

A farmer is conducting a trial of a new crop variety. He expects the population mean yield of this variety (per plant) to be 0.5kg, and is 90% sure that the mean yield will be between 0.3kg and 0.7kg. Fit a normal distribution to these beliefs. The standard deviation of the plant yields is known to be 0.2kg. The yields in kg of the first

4 plants are 0.6, 0.8, 0.85 and 0.9. Assuming that these yields are normally distributed, derive the farmerâĂŹs posterior distribution for the population mean yield, and obtain a 95% posterior interval.

```
> data<-list(y=c(0.6,0.8,0.85,0.9))
> x.bar<-mean(data$y)
> ## priors:
> m<-0.5
> v<-(0.2/1.65)^2
> ## data size:
> n<-4
> ## variance of data:
> sigma2<-0.2^2
> derive.post<-function(n,x.bar,sigma2,m,v){
+ v.star<- 1/( (1/v) + n/sigma2 )
+ m.star<-v.star*(m/v + (n*x.bar/sigma2))
+ return(list(m.star,v.star))
+ }
> (post<-derive.post(n=length(data$y),
+                    x.bar=mean(data$y),
+                    sigma2=sigma2,m=m,v=v))

[[1]]
[1] 0.67107

[[2]]
[1] 0.0059502

> ## posterior 95% credible interval
>  post[1][[1]]-2*sqrt( post[2][[1]]); post[1][[1]]+2*sqrt( post[2][[1]])

[1] 0.51679

[1] 0.82534
```

JAGS:

```
> ## specify data: see above
>
> ## model specification:
>
> cat("
+ model
+    {
+ for(i in 1:4){
+     y[i] ~ dnorm(mu,1/(0.2^2))
+     } ## likelihood
```

```
+    mu ~ dnorm(0.5,1/(0.2/1.65)^2) ## prior
+ }",
+     file="exercise4point2.jag" )
> ## specify variables to track
> ## the posterior distribution of:
> track.variables<-c("mu")
> ## load rjags library:
> library(rjags,quietly=T)
> ## define model:
> ex4point2.mod <- jags.model(
+     data = data,
+     file = "exercise4point2.jag",
+     n.chains = 4,
+     n.adapt =2000 ,quiet=T)
> ## run model:
> ex4point2.res <- coda.samples( ex4point2.mod,
+                var = track.variables,
+                n.iter = 100000,
+                thin = 50 )
> ## summarize and plot:
> summary(ex4point2.res)

Iterations = 50:1e+05
Thinning interval = 50
Number of chains = 4
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean             SD       Naive SE Time-series SE
      0.670590       0.077419       0.000866       0.000845

2. Quantiles for each variable:

 2.5%   25%   50%   75% 97.5%
0.518 0.618 0.671 0.723 0.824

> ## intervals:
> 0.671364-2*0.076182 ; 0.671364+2*0.076182

[1] 0.519

[1] 0.82373

> #[0.520, 0.818]
```

```
> plot(ex4point2.res)
> gelman.diag(ex4point2.res)

Potential scale reduction factors:

   Point est. Upper C.I.
mu          1          1
```

The analytical and JAGS intervals match up.
Analytical interval: mean 0.67107 [0.520, 0.818]
JAGS interval: mean 0.671364, [0.520, 0.818]

13.3 JAGS model for sheep data in 4.3.5

```
> data<-list(x=18,n=30,a=19,b=1)
> cat("
+ model
+    {
+  x ~ dbin(theta,n) ## likelihood
+  theta ~  dbeta(a,b) ## prior
+ }",
+     file="sheep.jag" )
> ## specify variables to track
> ## the posterior distribution of:
> track.variables<-c("theta")
> ## load rjags library:
> library(rjags,quietly=T)
> ## define model:
> sheep.mod <- jags.model(
+     data = data,
+     file = "sheep.jag",
+     n.chains = 4,
+     n.adapt =2000 ,quiet=T)
> ## run model:
> sheep.res <- coda.samples( sheep.mod,
+                var = track.variables,
+                n.iter = 100000,
+                thin = 50 )
> ## summarize and plot:
> summary(sheep.res)

Iterations = 50:1e+05
Thinning interval = 50
Number of chains = 4
Sample size per chain = 2000
```

```
1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean             SD       Naive SE Time-series SE
      0.739117       0.062024       0.000693       0.000693

2. Quantiles for each variable:

 2.5%   25%   50%   75% 97.5%
0.610 0.698 0.743 0.783 0.849
```

```
> plot(sheep.res)
```

Skeptic's prior:

```
> data<-list(x=18,n=30,a=1,b=19)
> cat("
+ model
+    {
+  x ~ dbin(theta,n) ## likelihood
+  theta ~  dbeta(a,b) ## prior
+ }",
+     file="sheep.jag" )
> ## specify variables to track
> ## the posterior distribution of:
> track.variables<-c("theta")
> ## load rjags library:
> library(rjags,quietly=T)
> ## define model:
> sheep.mod <- jags.model(
+     data = data,
+     file = "sheep.jag",
+     n.chains = 4,
+     n.adapt =2000 ,quiet=T)
> ## run model:
> sheep.res <- coda.samples( sheep.mod,
+                var = track.variables,
+                n.iter = 100000,
+                thin = 50 )
> ## summarize and plot:
> summary(sheep.res)

Iterations = 50:1e+05
Thinning interval = 50
Number of chains = 4
```

```
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean             SD       Naive SE Time-series SE
      0.379762       0.067017       0.000749       0.000749

2. Quantiles for each variable:

 2.5%   25%   50%   75% 97.5%
0.255 0.332 0.378 0.425 0.515
```

```
> plot(sheep.res)
```

With much more data, the disagreement can be resolved:

```
> ## prior 1
> data<-list(x=610,n=1000,a=19,b=1)
> ## prior 2
> data<-list(x=610,n=1000,a=19,b=1)
> cat("
+ model
+    {
+  x ~ dbin(theta,n) ## likelihood
+  theta ~  dbeta(a,b) ## prior
+ }",
+     file="sheep.jag" )
> ## specify variables to track
> ## the posterior distribution of:
> track.variables<-c("theta")
> ## load rjags library:
> library(rjags,quietly=T)
> ## define model:
> sheep.mod <- jags.model(
+     data = data,
+     file = "sheep.jag",
+     n.chains = 4,
+     n.adapt =2000 ,quiet=T)
> ## run model:
> sheep.res <- coda.samples( sheep.mod,
+                var = track.variables,
+                n.iter = 100000,
+                thin = 50 )
> ## summarize and plot:
> summary(sheep.res)
```

```
Iterations = 50:1e+05
Thinning interval = 50
Number of chains = 4
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean             SD       Naive SE Time-series SE
      0.616565       0.015251       0.000171       0.000172

2. Quantiles for each variable:

 2.5%   25%   50%   75% 97.5%
0.586 0.606 0.617 0.627 0.646
```

```
> plot(sheep.res)
```

Perhaps compare priors.

13.4 Exercise 4.3.7

```
> post.mean.beta<-function(a,b,x,n){
+   m<-(a+x)/(a+b+n)
+   return(m)
+ }
> post.var.beta<-function(a,b,x,n){
+   v<-((a+x)*(b+n-x))/((a+b+n)^2 * (a+b+n+1))
+   return(v)
+ }
> post.mean.beta(a=3,b=12,x=2,n=15)

[1] 0.16667

> post.var.beta(a=3,b=12,x=2,n=15)

[1] 0.0044803

> sqrt(post.var.beta(a=3,b=12,x=2,n=15))

[1] 0.066935

>  0.16667-2*sqrt(post.var.beta(a=3,b=12,x=2,n=15))

[1] 0.0328

>  0.16667+2*sqrt(post.var.beta(a=3,b=12,x=2,n=15))
```

```
[1] 0.30054
```

Using JAGS: checks out.

```
> data<-list(x=2,n=15,a=3,b=12)
> cat("
+ model
+    {
+  x ~ dbin(theta,n) ## likelihood
+  theta ~  dbeta(a,b) ## prior
+ }",
+     file="grades.jag" )
> ## specify variables to track
> ## the posterior distribution of:
> track.variables<-c("theta")
> ## load rjags library:
> library(rjags,quietly=T)
> ## define model:
> grades.mod <- jags.model(
+     data = data,
+     file = "grades.jag",
+     n.chains = 4,
+     n.adapt =2000 ,quiet=T)
> ## run model:
> grades.res <- coda.samples( grades.mod,
+                var = track.variables,
+                n.iter = 100000,
+                thin = 50 )
> ## summarize and plot:
> summary(grades.res)

Iterations = 50:1e+05
Thinning interval = 50
Number of chains = 4
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean             SD       Naive SE Time-series SE
      0.166057       0.066461       0.000743       0.000736

2. Quantiles for each variable:

  2.5%    25%    50%    75%  97.5%
0.0582 0.1166 0.1594 0.2070 0.3155
```

```
> plot(grades.res)
```

13.5 Normal distribution, mean unknown, variance known

13.6 Section 4.4.5, Normal distribution, mean and variance unknown, using JAGS

13.7 Decision theory

Peter Tryfos:

"the decision problem is how to select the best of the available alternatives. The elements of the problem are the possible alternatives (ac- tions, acts), the possible events (states, outcomes of a random process), the probabilities of these events, the consequences associated with each possible alternative-event combination, and the criterion (decision rule) according to which the best alternative is selected."

Types of problems:

"The simplest decision problems can be resolved by listing the possible monetary consequences and the associated probabilities for each alternative, calculating the expected monetary values of all alternatives, and selecting the alternative with the highest expected monetary value. The determina- tion of the optimal alternative becomes a little more complicated when the alternatives involve sequences of decisions."

"In another class of problems, it is possible to acquireÃśoften at a certain costÃśadditional information about an uncertain variable. This additional information is rarely entirely accurate. Its valueÃśhence, also the maximum amount one would be willing to pay to acquire itÃśshould depend on the dif- ference between the best one expects to do with the help of this information and the best one expects to do without it."

Another type of problem is: "non-recurring decision problems in which the consequences cannot be measured in monetary terms"

```
> compute.utility<-function(p){
+   p*0 + (1-p)*(-800)
+ }
> ps<-seq(0,1,by=0.01)
> plot(ps,compute.utility(ps))
> abline(h=-300)
> ## for p=0.62-0.63 utility the same as S
>
```

Regret:

```
> payoff<-c(2,3,4,5,6,1)
> ## utilities identical:
> sum(payoff*(1/6))

[1] 3.5

> sum((1:6)*1/6)

[1] 3.5

> ## regret different:
> roll<-function(){
+   payoff<-c(2,3,4,5,6,1)
+   x<-sample(1:6,size=1)
+   return(c(payoff[x],x))
+ }
> roll()

[1] 3 2

> n.sim<-100000
> store.A<-rep(NA,n.sim)
> store.B<-rep(NA,n.sim)
> for(i in 1:n.sim){
+   x<-roll()
+  store.A[i]<-x[1]
+  store.B[i]<-x[2]
+ }
> 5/6

[1] 0.83333

> results<-table(store.A>store.B)/n.sim
> results<-as.data.frame(results)
> results$regret<-c("no.regret","regret")
>
```

13.8 Predictive posterior distributions

p. 39 lecture notes:

```
> computeposterior<-function(y){
+ (choose(10,y) / (gamma(37)*gamma(13)/gamma(50))) * ((gamma(37+y)*gamma(23-y))/gamma(60))
+ }
> plot(0:10,computeposterior(0:10),type="h")
>
```

13.9 Conjugacy notes

13.10 Ch 10: Hierarchical models

```
> schools<-c(28.39,7.94,-2.75,6.82,-0.64,0.63,18.01,12.16)
> id<-1:8
> ## equation 242
> mean(schools)

[1] 8.82

> schools.dat<-list(x=schools,m=0,t=10^(-6),a=0.001,b=0.001)
```

JAGS:

```
> ## schools model:
> cat("
+ model
+    {
+ for(i in 1:8){
+     ## note specification in terms of precision:
+   x[i] ~ dnorm(theta[i],pow(36,-1))
+   theta[i] ~ dnorm(mu,tau)
+ }
+ ## priors:
+ mu ~ dnorm(m,t)
+ tau ~ dgamma(a,b)
+ sigma <- pow(tau,-2)
+ }",
+     file="JAGSmodels/schools.jag" )
> track.variables<-c("mu","sigma")
> ## load rjags library:
> library(rjags,quietly=T)
> ## define model:
> schools.mod <- jags.model(
+     data = schools.dat,
+     file = "JAGSmodels/schools.jag",
+     n.chains = 4,
+     n.adapt =2000)

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 29

Initializing model
```

```
> ## run model:
> schools.res <- coda.samples( schools.mod,
+                 var = track.variables,
+                 b.burnin = 50000,
+                 n.iter = 100000,
+                 thin = 50 )
> gelman.diag(schools.res)

Potential scale reduction factors:

      Point est. Upper C.I.
mu          1.00       1.00
sigma       1.07       1.07

Multivariate psrf

1

> ## summarize and plot:
> summary(schools.res)

Iterations = 50:1e+05
Thinning interval = 50
Number of chains = 4
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean       SD Naive SE Time-series SE
mu        8.86     3.81   0.0426         0.0426
sigma 17247.79 83095.01 929.0305       928.9566

2. Quantiles for each variable:

          2.5%    25%     50%     75%    97.5%
mu    1.32e+00   6.64    8.83    11.1     16.6
sigma 8.32e-05 515.89 3211.40 11355.7 114613.0

> plot(schools.res)
```

Why is sigma so high?

13.11 Exercise 10.2.1: Power plant failure rates

```
> pump<-1:10
> failures<-c(3,1,1,5,4,6,2,1,4,9)
```

```
> pump.dat<-list(x=failures,a=1,b=1,c=1)
> cat("
+ model
+    {
+ for(i in 1:10){
+     ## note specification in terms of precision:
+   x[i] ~ dpois(lambda[i])
+   lambda[i] ~ dgamma(alpha,beta)
+ }
+   ##prior
+   alpha ~ dexp(a)
+   beta ~ dgamma(b,c)
+ }",
+     file="JAGSmodels/pumpfailure.jag" )
> track.variables<-c("alpha","beta","lambda")
> ## load rjags library:
> library(rjags,quietly=T)
> ## define model:
> pump.mod <- jags.model(
+     data = pump.dat,
+     file = "JAGSmodels/pumpfailure.jag",
+     n.chains = 4,
+     n.adapt =2000 ,quiet=T)
> ## run model:
> pump.res <- coda.samples( pump.mod,
+                var = track.variables,
+                n.iter = 100000,
+                thin = 50 )
> gelman.diag(pump.res)

Potential scale reduction factors:

           Point est. Upper C.I.
alpha               1       1.00
beta                1       1.00
lambda[1]           1       1.00
lambda[2]           1       1.00
lambda[3]           1       1.01
lambda[4]           1       1.00
lambda[5]           1       1.00
lambda[6]           1       1.00
lambda[7]           1       1.00
lambda[8]           1       1.00
lambda[9]           1       1.00
```

```
lambda[10]          1       1.00

Multivariate psrf

1
```

```
> ## summarize and plot:
> summary(pump.res)

Iterations = 2050:102000
Thinning interval = 50
Number of chains = 4
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

            Mean    SD Naive SE Time-series SE
alpha      2.356 1.111   0.0124         0.0123
beta       0.686 0.358   0.0040         0.0040
lambda[1]  3.199 1.447   0.0162         0.0161
lambda[2]  1.969 1.158   0.0130         0.0127
lambda[3]  1.969 1.143   0.0128         0.0128
lambda[4]  4.410 1.727   0.0193         0.0200
lambda[5]  3.809 1.567   0.0175         0.0183
lambda[6]  5.049 1.883   0.0211         0.0210
lambda[7]  2.574 1.287   0.0144         0.0148
lambda[8]  1.956 1.151   0.0129         0.0128
lambda[9]  3.814 1.595   0.0178         0.0175
lambda[10] 6.915 2.269   0.0254         0.0254

2. Quantiles for each variable:

            2.5%  25%   50%   75% 97.5%
alpha      0.811 1.55 2.154 2.928  5.08
beta       0.196 0.43 0.616 0.869  1.58
lambda[1]  1.010 2.15 3.000 4.002  6.58
lambda[2]  0.342 1.12 1.769 2.590  4.79
lambda[3]  0.337 1.13 1.783 2.599  4.74
lambda[4]  1.731 3.18 4.149 5.394  8.36
lambda[5]  1.409 2.68 3.572 4.686  7.52
lambda[6]  2.134 3.69 4.791 6.134  9.48
lambda[7]  0.669 1.63 2.384 3.280  5.64
lambda[8]  0.357 1.11 1.744 2.597  4.74
lambda[9]  1.359 2.67 3.564 4.744  7.49
```

```
lambda[10] 3.315 5.26 6.617 8.284 12.08
```

```
> plot(pump.res)
```

We can compute mean and var of pump failures from alpha, beta.

13.12 MCMC

Suppose that we want to know the mean of a distribution but cannot solve $\int x(fx),dx$ can somehow sample from a distribution but can't

13.13 Monitoring convergence

From notes:

" 1. visually inspect to see if each random walk has reached equilibrium. If you have a very large sample size, you may need to increase the value next to thin, so not every value is plotted.

2. Try doubling the number of iterations from n to 2n, and compare means, variances, percentiles etc based on the two sample sizes. Note that each time you click on stats in the sample monitor tool, a new window is created.

3. Try different starting values. After finding the 2.5th and 97.5th percentiles of each variable the first time round, you could try new starting values well outside these intervals. Then compare the new sample means, variances and percentiles with the old sample."

Simple normal distrn model:

```
> cat("model{
+   for(i in 1:N){
+     y[i] ~ dnorm(mu[i], tau)
+     mu[i] <- alpha + beta * (x[i] - mean(x[]))
+   }
+   sigma <- 1/sqrt(tau)
+   alpha ~ dnorm(0, 1.0E-6)
+   beta ~ dnorm(0, 1.0E-6)
+   tau ~ dgamma(1.0E-3,1.0E-3)
+ }",file="JAGSmodels/normalexamplenotes.jag")
> dat<-list(x=c(1,2,3,4,5),y=c(0.8,2.3,3,3.7,5),N=5)
> # Number of steps to "tune" the samplers.
> n.adapt <- 1000 ## was 10000
> # Number of steps to "burn-in" the samplers.
> n.burnin <- 2000 ## was 20000
> # Number of chains to run.
> n.chains <- 2
> # Total number of steps in chains to save.
```

```
> n.savedsteps <- 4000 ## 80000
> # Number of steps to "thin" (1=keep every step).
> n.thin<-1
> # Steps per chain.
> n.perchain = ceiling( ( n.savedsteps * n.thin) / n.chains)
> ## vary inits:
> inits<-list(list(alpha=-3,beta=-4,tau=0.001),
+             list(alpha=5,beta=4,tau=10))
> mod <- jags.model(
+   file="JAGSmodels/normalexamplenotes.jag",
+   data = dat,
+   inits= inits,
+   n.chains = n.chains,
+   n.adapt =n.adapt , quiet=T)
> update(mod,n.iter=n.burnin)
> track.variables=c("alpha","beta","sigma")
> res <- coda.samples(mod,
+                       var = track.variables,
+                       n.iter = n.savedsteps,
+                       thin = n.thin)
> plot(res)
> gelman.diag(res)

Potential scale reduction factors:

      Point est. Upper C.I.
alpha       1.01       1.01
beta        1.00       1.00
sigma       1.00       1.00

Multivariate psrf

1

> summary(res,quantiles=c(0.025,0.5,0.975))

Iterations = 2001:6000
Thinning interval = 1
Number of chains = 2
Sample size per chain = 4000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

       Mean    SD Naive SE Time-series SE
```

```
alpha 2.962 0.210  0.00235        0.00244
beta  0.978 0.145  0.00163        0.00156
sigma 0.373 0.281  0.00314        0.00639

2. Quantiles for each variable:

      2.5%   50% 97.5%
alpha 2.58 2.958  3.36
beta  0.69 0.979  1.25
sigma 0.15 0.302  1.05

>
> #     Mean    SD Naive SE Time-series SE
> #alpha 2.962 0.192  0.00214        0.00231
> #beta  0.978 0.147  0.00165        0.00168
> #sigma 0.357 0.251  0.00281        0.00536
>
> #       2.5%   50% 97.5%
> #alpha 2.616 2.959 3.327
> #beta  0.719 0.981 1.228
> #sigma 0.148 0.294 0.941
```

Write a routine to take HPDs and then reset init values to be outside the HPD n times, then compare and print results.

From notes: "Recall that the aim here is to conduct inference for the unknown parameters α,β and σ^2 given the (x,y) data..., i.e., we want the posterior distribution $f(\alpha,\beta,\sigma^2 \mid x,y)$, with x and y as given. The chosen prior distributions are not conjugate to the likelihood $f(y \mid \alpha,\beta,\sigma^2,x)$ (treating x as given), so we cannot analytically derive $f(\alpha,\beta,\sigma^2 \mid x,y)$.

Gibbs sampling can be used to obtain a sample from $f(\alpha,\beta,\sigma^2 \mid x,y)$, by sampling from the full conditionals $f(\alpha \mid \beta,\sigma^2,x,y)$, $f(\beta|\alpha,\sigma^2,x,y)$ and $f(\sigma^2|\alpha,\beta,x,y)$. Once the model has been specified, WinBUGS (JAGS) will identify the full conditional distributions and sample from them directly if possible (i.e., if they are in the form of a standard distribution), or indirectly otherwise (e.g. using adaptive rejection sampling)."

13.14 Cholesterol data

A small study has been conducted to investigate the effect of a drug in lowering cholesterol levels. For each patient, their baseline cholesterol level and level after treatment are recorded. It is also recorded whether or not each patient suffered a heart attack during the duration of the study.

```
> chol.data<-read.table("data/cholesterol.txt",header=T)
> chol.data<-rbind(chol.data,c(4.0,NA,NA))
> dat<-as.list(chol.data)
> cat("model{
+ for(j in 1:101){
+    cr[j] ~ dnorm(mu,tau)
+    mi[j] ~ dbern(theta[j])
+ ##   logit(theta[j]) <- alpha + beta * (cb[j] - cr[j])
+ ## to decorrelate predictors:
+    logit(theta[j]) <- alpha +
+            beta * ((cb[j]-cr[j]) - 2.4973)
+ }
+ betapos<-step(beta)
+
+ ## priors:
+ mu ~ dnorm(0,0.00001)
+ tau ~ dgamma(0.001,0.001)
+ alpha ~ dnorm(0,0.00001)
+ beta ~ dnorm(0,0.00001)
+ sigma2 <- pow(tau,-1)
+ sigma <- sqrt(sigma2)
+ }
+     ",file="JAGSmodels/cholesterol.jag")
> # Number of steps to "tune" the samplers.
> n.adapt <- 1000 ## was 10000
> # Number of steps to "burn-in" the samplers.
> n.burnin <- 2000 ## was 20000
> # Number of chains to run.
> n.chains <- 2
> # Total number of steps in chains to save.
> n.savedsteps <- 4000 ## 80000
> # Number of steps to "thin" (1=keep every step).
> n.thin<-1
> # Steps per chain.
> n.perchain = ceiling( ( n.savedsteps * n.thin) / n.chains)
> ## vary inits:
> inits<-list(list(mu=0,alpha=-3,beta=-4,tau=0.001),
+             list(mu=10,alpha=5,beta=4,tau=10))
> mod <- jags.model(
+   file="JAGSmodels/cholesterol.jag",
+   data = dat,
+   inits= inits,
+   n.chains = n.chains,
+   n.adapt =n.adapt , quiet=T)
```

```
> update(mod,n.iter=n.burnin)
> track.variables=c("mu","alpha","beta","sigma2","sigma","betapos","mi[101]","cr[101]")
> res <- coda.samples(mod,
+                       var = track.variables,
+                       n.iter = n.savedsteps,
+                       thin = n.thin)
> plot(res)
> gelman.diag(res)

Potential scale reduction factors:

        Point est. Upper C.I.
alpha            1          1
beta             1          1
betapos          1          1
cr[101]          1          1
mi[101]          1          1
mu               1          1
sigma            1          1
sigma2           1          1

Multivariate psrf

1

> summary(res,quantiles=c(0.025,0.5,0.975))

Iterations = 3001:7000
Thinning interval = 1
Number of chains = 2
Sample size per chain = 4000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

           Mean      SD Naive SE Time-series SE
alpha   -0.2501 0.20890 2.34e-03       3.04e-03
beta     0.6966 0.36259 4.05e-03       5.15e-03
betapos  0.9748 0.15689 1.75e-03       2.23e-03
cr[101]  1.0006 0.21060 2.35e-03       2.35e-03
mi[101]  0.5186 0.49968 5.59e-03       5.59e-03
mu       1.0010 0.02089 2.34e-04       2.42e-04
sigma    0.2083 0.01496 1.67e-04       1.72e-04
sigma2   0.0436 0.00631 7.06e-05       7.15e-05
```

```
2. Quantiles for each variable:

            2.5%    50%  97.5%
alpha   -0.66176 -0.249 0.1483
beta    -0.00172  0.696 1.4187
betapos  0.00000  1.000 1.0000
cr[101]  0.59008  1.003 1.4163
mi[101]  0.00000  1.000 1.0000
mu       0.95997  1.001 1.0419
sigma    0.18106  0.207 0.2400
sigma2   0.03278  0.043 0.0576
```

```
> autocorr.plot(res)
> mcmcChain<-as.matrix(res)
> ## correlation:
> plot(mcmcChain[,1],mcmcChain[,2])
> ## prob that beta is positive:
> pnorm(0,mean=0.7011, sd=0.37719,lower.tail=F)

[1] 0.96847

> ## matches up:
> counts<-table(mcmcChain[,2]>0)
> counts[2]/(sum(counts))

   TRUE
0.97475
```

"Using your re-parameterised model, obtain the posterior probability that β is positive. (Hint: the function step(x) returns 1 if x is positive and 0 otherwise)." Notice that I can even talk about my uncertainty about this quantity. Look up step in BUGS book.

"If a new patient is recruited to the study (drawn from the same subpopulation of interest), with a baseline cholesterol level of 4.0, what is the probability that they will suffer a heart attack during the trial?"

We can also say what the likely reduction is going to be for this patient, right?

13.15 Ch 15: Hospital deaths

```
> # n: no of operations
> # x: no of deaths
> # N: no of hospitals
> dat<-list(n=c(47,211,810,148,196,360,119,207,97,
+               256,148,215),
+           x=c(0,8,46,9,13,24,8,14,8,29,18,31),
```

```
+           N=12)
> cat("model
+     {
+     for(i in 1:N){
+     x[i] ~ dbinom(p[i],n[i])
+     logit(p[i]) <- gamma[i]
+     gamma[i] ~ dnorm(mu,tau)
+     }
+     ## priors:
+     mu ~ dnorm(0,1.0E-3)
+     #tau ~ dgamma(0.001,0.001)
+     sigma ~ dunif(0,10)
+     ## half-normal prior:
+     ##sigma ~ dnorm(0,0.001)T(0,)
+     ##sigma <- 1/sqrt(tau)
+     tau <- 1/pow(sigma,2)
+     for(i in 1:N){
+     theta.pred[i] <- exp(gamma[i])/(1+exp(gamma[i]))
+     }
+ }",file="JAGSmodels/hospitaldeaths.jag")
> # Number of steps to "tune" the samplers.
> n.adapt <- 1000 ## was 10000
> # Number of steps to "burn-in" the samplers.
> n.burnin <- 2000 ## was 20000
> # Number of chains to run.
> n.chains <- 2
> # Total number of steps in chains to save.
> n.savedsteps <- 4000 ## 80000
> # Number of steps to "thin" (1=keep every step).
> n.thin<-1
> # Steps per chain.
> n.perchain = ceiling( ( n.savedsteps * n.thin) / n.chains)
> ## vary inits:
> #inits<-list(list(mu=0,alpha=-3,beta=-4,tau=0.001),
> #            list(mu=10,alpha=5,beta=4,tau=10))
>
> mod <- jags.model(
+   file="JAGSmodels/hospitaldeaths.jag",
+   data = dat,
+ #  inits= inits,
+   n.chains = n.chains,
+   n.adapt =n.adapt , quiet=T)
> update(mod,n.iter=n.burnin)
> track.variables=c("mu","sigma","theta.pred")
```

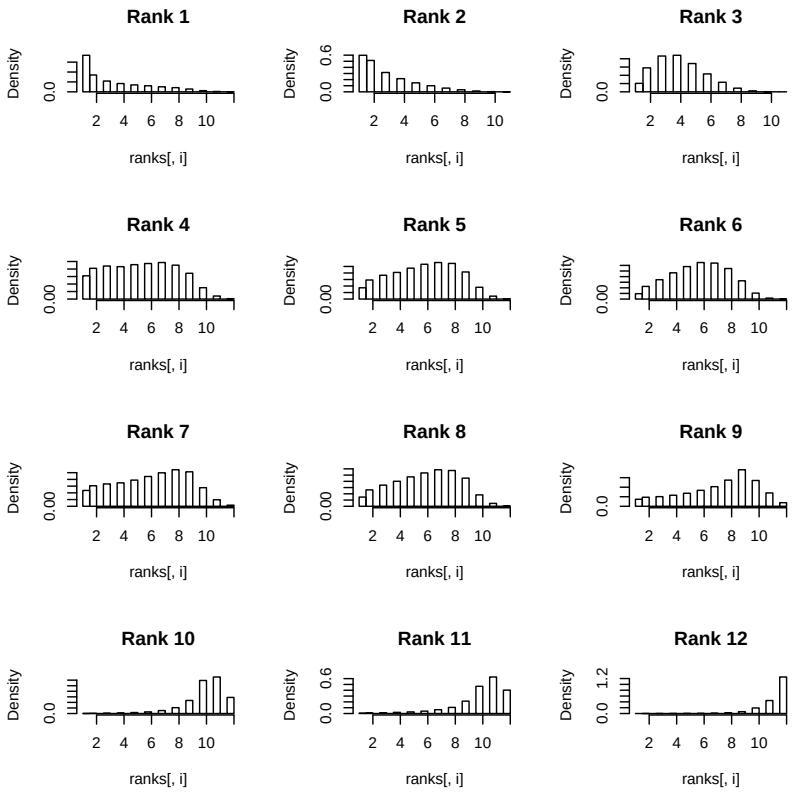
```
> res <- coda.samples(mod,
+                     var = track.variables,
+                     n.iter = n.savedsteps,
+                     thin = n.thin)
> plot(res)
> gelman.diag(res)

Potential scale reduction factors:

               Point est. Upper C.I.
mu                      1       1.00
sigma                   1       1.01
theta.pred[1]           1       1.00
theta.pred[2]           1       1.00
theta.pred[3]           1       1.00
theta.pred[4]           1       1.00
theta.pred[5]           1       1.00
theta.pred[6]           1       1.00
theta.pred[7]           1       1.00
theta.pred[8]           1       1.00
theta.pred[9]           1       1.01
theta.pred[10]          1       1.00
theta.pred[11]          1       1.00
theta.pred[12]          1       1.01

Multivariate psrf

1

> summary(res,quantiles=c(0.025,0.5,0.975))

Iterations = 3001:7000
Thinning interval = 1
Number of chains = 2
Sample size per chain = 4000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

                  Mean      SD Naive SE Time-series SE
mu             -2.5678 0.17073 1.91e-03       0.002936
sigma           0.4677 0.18440 2.06e-03       0.006174
theta.pred[1]   0.0493 0.02022 2.26e-04       0.000480
theta.pred[2]   0.0495 0.01313 1.47e-04       0.000256
theta.pred[3]   0.0584 0.00793 8.86e-05       0.000122
```

```
theta.pred[4]   0.0654 0.01658 1.85e-04       0.000247
theta.pred[5]   0.0686 0.01523 1.70e-04       0.000239
theta.pred[6]   0.0680 0.01202 1.34e-04       0.000178
theta.pred[7]   0.0700 0.01828 2.04e-04       0.000281
theta.pred[8]   0.0694 0.01486 1.66e-04       0.000231
theta.pred[9]   0.0786 0.02076 2.32e-04       0.000322
theta.pred[10]  0.1044 0.01826 2.04e-04       0.000293
theta.pred[11]  0.1060 0.02307 2.58e-04       0.000372
theta.pred[12]  0.1268 0.02243 2.51e-04       0.000453

2. Quantiles for each variable:

                  2.5%     50%   97.5%
mu             -2.9320 -2.5582 -2.2499
sigma           0.2111  0.4354  0.8943
theta.pred[1]   0.0141  0.0480  0.0920
theta.pred[2]   0.0261  0.0488  0.0771
theta.pred[3]   0.0438  0.0581  0.0749
theta.pred[4]   0.0368  0.0640  0.1023
theta.pred[5]   0.0420  0.0674  0.1024
theta.pred[6]   0.0462  0.0674  0.0932
theta.pred[7]   0.0387  0.0684  0.1107
theta.pred[8]   0.0431  0.0685  0.1011
theta.pred[9]   0.0437  0.0765  0.1252
theta.pred[10]  0.0728  0.1031  0.1427
theta.pred[11]  0.0672  0.1039  0.1552
theta.pred[12]  0.0863  0.1259  0.1740
```

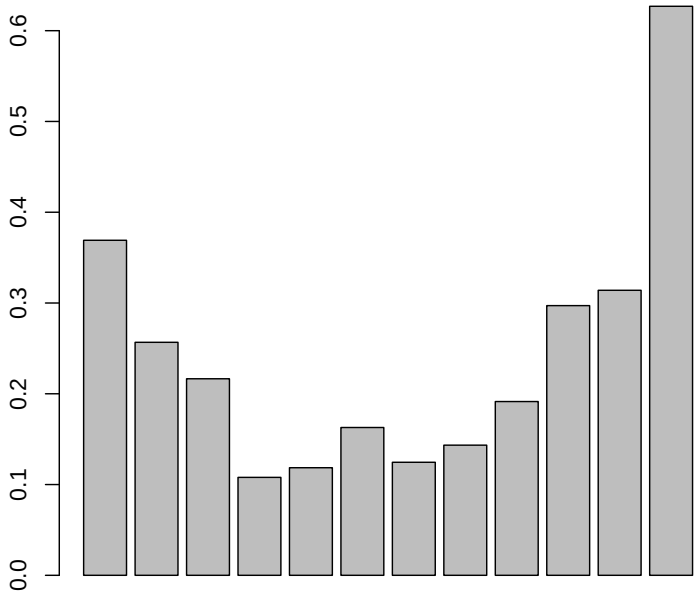
```
> mcmcChain<-as.matrix(res)
> hosp<-LETTERS[1:12]
> hosp.res<-data.frame(hosp,summary(res)$statistics[3:14,1:2])
> ## sample from 12 distributions, derive
> ## distrn. of ranks
> n.sim<-100000
> samp<-matrix(c(rep(NA,12*n.sim)),ncol=12)
> for(i in 1:n.sim){
+   for(j in 1:12){
+     samp[i,j] <- rnorm(1,mean=hosp.res[j,2],
+                        sd=hosp.res[j,3])
+   }
+ }
> samp<-as.data.frame(samp)
> colnames(samp)<-LETTERS[1:12]
> ranks<-matrix(c(rep(NA,12*n.sim)),ncol=12)
```

```
> for(i in 1:n.sim){
+ ranks[i,]<-rank(samp[i,])
+ }
```

```
> op <- par(mfrow=c(4,3))
> for(i in 1:12){
+ hist(ranks[,i],freq=F,main=(paste("Rank ",i,sep="")))
+ }
> probs<-rep(NA,12)
> for(i in 1:12){
+ counts<-table(ranks[,i]==i)
+ ##P(rank.I==9)
+ probs[i]<-counts[2]/sum(counts)
+ }
```




```
> ## probabilities of each rank:
> barplot(probs)
```




```
> ## 95\% region for I:
> counts<-table(2<ranks[,9] & ranks[,9]>11)
> 1-counts[2]/sum(counts)

   TRUE
0.98109
```

13.16 Ch 16: Meta analysis

```
> # r and n are defined as 2x22 matrices. The first row of each is control, the second is treatment.
> dat<-list(r=structure(.Data=c(3, 14, 11, 127, 27, 6, 152, 48, 37, 188, 52, 47, 16, 45, 31, 38, 12, 6,
+ n=structure(.Data=c(39, 116, 93, 1520, 365, 52, 939, 471, 282, 1921, 583, 266, 293, 883, 147, 213, 12
+ I=structure(.Data=c(1,0,0,1),.Dim=c(2,2)),
+ malpha=c(0,0),
+ palpha=structure(.Data=c(1.0E-6, 0, 0, 1.0E-6), .Dim = c(2, 2)))
> # Number of steps to "tune" the samplers.
> n.adapt <- 1000 ## was 10000
> # Number of steps to "burn-in" the samplers.
> n.burnin <- 2000 ## was 20000
> # Number of chains to run.
> n.chains <- 2
```

```
> # Total number of steps in chains to save.
> n.savedsteps <- 4000 ## 80000
> # Number of steps to "thin" (1=keep every step).
> n.thin<-1
> # Steps per chain.
> n.perchain = ceiling( ( n.savedsteps * n.thin) / n.chains)
> cat("
+ model
+ {
+ for(i in 1:22){
+ ## controls
+ r[1,i] ~ dbinom(p[1,i],n[1,i])
+ ## treatment
+ r[2,i] ~ dbinom(p[2,i],n[2,i])
+ logit(p[1,i]) <- gamma[1,i]
+ logit(p[2,i]) <- gamma[2,i]
+ ## define gamma in terms of beta and then model beta
+ #beta[1,i] <- gamma[1,i] + gamma[2,i]
+ #beta[2,i] <- gamma[1,i] - gamma[2,i]
+ gamma[1,i] <- 0.5 * (beta[1,i] + beta[2,i])
+ gamma[2,i] <- 0.5 * (beta[1,i] - beta[2,i])
+ #b1 = g1 + g2
+ #b2 = g1 - g2
+ #So:
+ #b1 = g1 + g2  and g2 = g1-b2
+ #So: b1 = g1 + (g1-b2)
+ ## 2g1 = b1 + b2
+ ## g1 = 0.5 (b1 + b2)
+ #g2 = g1 - b2 = 0.5 (b1 + b2) - b2
+ #-> g2 = 0.5 b1 + 0.5 b2 - b2 = 0.5b1 - 0.5 b2
+ #-> g2 = 0.5(b1-b2)
+ beta[1:2,i] ~ dmnorm(alpha[1:2],
+                      Sigmainv[1:2,1:2])
+ }
+ ## priors:
+ Sigmainv[1:2,1:2] ~ dwish(I[1:2,1:2],2)
+ Sigma <- inverse(Sigmainv)
+ alpha[1] ~ dnorm(0,0.0001)
+ alpha[2] ~ dnorm(0,0.0001)
+ ## prediction:
+ newbeta ~ dmnorm(alpha[1:2],Sigmainv[1:2,1:2])
+ newgamma1 <- 0.5 * (newbeta[1] + newbeta[2])
+ newgamma2 <- 0.5 * (newbeta[1] - newbeta[2])
+ logit(newp1) <- newgamma1
```

```
+ logit(newp2) <- newgamma2
+ ## diff in log odds:
+ diffgamma <- newgamma1 - newgamma2
+ diffp<-newp1-newp2
+ }
+ ",file="JAGSmodels/metaanalysis.jag")
> ## Paul's solution:
> cat("
+ model
+ {
+ alpha[1:2] ~ dmnorm(malpha[1:2],palpha[1:2 , 1:2])
+ Salpha[1:2 , 1:2] ~ dwish(I[1:2 , 1:2],2)
+ # Put gamma priors on each *precision*
+ ## ALTERNATIVE TO WISHART:
+ #tau1~dgamma(0.001,0.001)
+ #tau2~dgamma(0.001,0.001)
+ #Salpha[1,1] <-tau1
+ #Salpha[2,2] <-tau2
+ #Salpha[1,2] <-0
+ #Salpha[2,1] <-0
+ # Alternatively use uniform priors on each sd
+ #s1~dunif(0,10)
+ #s2~dunif(0,10)
+ #tau1<-1/(s1*s1)
+ #tau2<-1/(s2*s2)
+
+ for( j in 1 : 22 ) {
+ beta[j , 1:2] ~ dmnorm(alpha[1:2],Salpha[1:2 , 1:2])
+ gamma[j , 1] <- 0.5 * (beta[j , 1] - beta[j , 2])
+ gamma[j , 2] <- 0.5 * (beta[j , 1] + beta[j , 2])
+ logit(theta[j , 1]) <- gamma[j , 1]
+ logit(theta[j , 2]) <- gamma[j , 2]
+ r[1 , j] ~ dbin(theta[j , 1],n[1 , j])
+ r[2 , j] ~ dbin(theta[j , 2],n[2 , j])
+ }
+ # predictive distribution for new treatment
+ newbeta[1:2] ~ dmnorm(alpha[1:2],Salpha[1:2 , 1:2])
+ newg1 <- 0.5 * (newbeta[1] - newbeta[2])
+ newg2 <- 0.5 * (newbeta[1] + newbeta[2])
+ logit(newtheta1) <- newg1
+ logit(newtheta2) <- newg2
+ tdiff <- newtheta1 - newtheta2
+ }
+ ",file="JAGSmodels/metaanalysisSOL.jag")
```

```
> mod <- jags.model(
+   file="JAGSmodels/metaanalysis.jag",
+   data = dat,
+   n.chains = n.chains,
+   n.adapt =n.adapt , quiet=T)
> mod2 <- jags.model(
+   file="JAGSmodels/metaanalysisSOL.jag",
+   data = dat,
+   n.chains = n.chains,
+   n.adapt =n.adapt , quiet=T)
> update(mod,n.iter=n.burnin)
> update(mod2,n.iter=n.burnin)
> track.variables=c("alpha","Sigma","diffgamma","diffp")
> track.variables=c("alpha","tdiff")
> res <- coda.samples(mod,
+                       var = track.variables,
+                       n.iter = n.savedsteps,
+                       thin = n.thin)
> res <- coda.samples(mod2,
+                       var = track.variables,
+                       n.iter = n.savedsteps,
+                       thin = n.thin)
> summary(res)


Iterations = 3001:7000
Thinning interval = 1
Number of chains = 2
Sample size per chain = 4000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

             Mean     SD Naive SE Time-series SE
alpha[1] -4.66457 0.1887 0.002110        0.00332
alpha[2]  0.03717 0.1744 0.001950        0.00364
tdiff    -0.00353 0.0673 0.000753        0.00078

2. Quantiles for each variable:

           2.5%     25%      50%     75%  97.5%
alpha[1] -5.041 -4.7893 -4.65985 -4.5394 -4.289
alpha[2] -0.302 -0.0760  0.03662  0.1518  0.385
tdiff    -0.149 -0.0394 -0.00134  0.0344  0.129
```

```
> plot(res)
> gelman.diag(res,multivariate=F)

Potential scale reduction factors:

         Point est. Upper C.I.
alpha[1]          1          1
alpha[2]          1          1
tdiff             1          1

> ## correlation: almost none,
> ## so why did we fit it?
> -0.0478/sqrt(0.6675)*sqrt(0.5183)

[1] -0.04212

> summary(res,quantiles=c(0.025,0.5,0.975))

Iterations = 3001:7000
Thinning interval = 1
Number of chains = 2
Sample size per chain = 4000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

             Mean     SD Naive SE Time-series SE
alpha[1] -4.66457 0.1887 0.002110        0.00332
alpha[2]  0.03717 0.1744 0.001950        0.00364
tdiff    -0.00353 0.0673 0.000753        0.00078

2. Quantiles for each variable:

           2.5%      50%  97.5%
alpha[1] -5.041 -4.65985 -4.289
alpha[2] -0.302  0.03662  0.385
tdiff    -0.149 -0.00134  0.129

> mcmcChains<-as.matrix(res)
> pnorm(0,mean=-0.0459,sd=0.739)

[1] 0.52476
```

Conclusion doesn't match claims.

To-do: "Investigate the robustness of your inferences to your prior distribution for Σ. For alternative choices of prior, consider a diagonal matrix Σ with independent priors for the diagonal elements."

13.17 Ch 17: Model comparison

```
> chol.data<-read.table("data/cholesterol.txt",header=T)
> chol.data<-rbind(chol.data,c(4.0,NA,NA))
> dat<-as.list(chol.data)
> cat("model{
+ for(j in 1:101){
+    cr[j] ~ dnorm(mu,tau)
+    mi[j] ~ dbern(theta[j])
+ ##   logit(theta[j]) <- alpha + beta * (cb[j] - cr[j])
+ ## to decorrelate predictors:
+    logit(theta[j]) <- alpha +
+            beta * ((cb[j]-cr[j]) - 2.4973)
+
+ ##   probit(theta[j]) <- alpha +
+ ##           beta * ((cb[j]-cr[j]) - 2.4973)
+ }
+ betapos<-step(beta)
+ ## priors:
+ mu ~ dnorm(0,0.00001)
+ tau ~ dgamma(0.001,0.001)
+ alpha ~ dnorm(0,0.00001)
+ beta ~ dnorm(0,0.00001)
+ sigma2 <- pow(tau,-1)
+ sigma <- sqrt(sigma2)
+ ## predicted:
+ gamma101<-alpha + beta * ((cb[101]-cr[101]) - 2.4973)
+ theta.pred101 <- exp(gamma101)/(1+exp(gamma101))
+ }
+     ",file="JAGSmodels/cholesterol.jag")
> # Number of steps to "tune" the samplers.
> n.adapt <- 1000 ## was 10000
> # Number of steps to "burn-in" the samplers.
> n.burnin <- 2000 ## was 20000
> # Number of chains to run.
> n.chains <- 2
> # Total number of steps in chains to save.
> n.savedsteps <- 4000 ## 80000
> # Number of steps to "thin" (1=keep every step).
> n.thin<-1
> # Steps per chain.
> n.perchain = ceiling( ( n.savedsteps * n.thin) / n.chains)
> ## vary inits:
> inits<-list(list(mu=0,alpha=-3,beta=-4,tau=0.001),
```

```
+             list(mu=10,alpha=5,beta=4,tau=10))
> mod <- jags.model(
+   file="JAGSmodels/cholesterol.jag",
+   data = dat,
+   inits= inits,
+   n.chains = n.chains,
+   n.adapt =n.adapt , quiet=T)
> update(mod,n.iter=n.burnin)
> track.variables=c("mu","alpha","beta","sigma2","sigma","betapos","mi[101]","cr[101]","theta.pred101")
> res <- coda.samples(mod,
+                       var = track.variables,
+                       n.iter = n.savedsteps,
+                       thin = n.thin)
> plot(res)
> gelman.diag(res)

Potential scale reduction factors:

              Point est. Upper C.I.
alpha                  1       1.00
beta                   1       1.00
betapos              NaN        NaN
cr[101]                1       1.00
mi[101]                1       1.00
mu                     1       1.00
sigma                  1       1.01
sigma2                 1       1.01
theta.pred101          1       1.00

Multivariate psrf

1

> #res.probit<-summary(res,quantiles=c(0.025,0.5,0.975))
> summary(res,quantiles=c(0.025,0.5,0.975))

Iterations = 3001:7000
Thinning interval = 1
Number of chains = 2
Sample size per chain = 4000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

                Mean      SD Naive SE Time-series SE
```

```
alpha         -0.2566 0.20583 2.30e-03       2.87e-03
beta           0.6995 0.37245 4.16e-03       5.36e-03
betapos        0.9754 0.15499 1.73e-03       2.28e-03
cr[101]        1.0024 0.20648 2.31e-03       2.31e-03
mi[101]        0.5424 0.49823 5.57e-03       5.41e-03
mu             1.0008 0.02103 2.35e-04       2.35e-04
sigma          0.2078 0.01495 1.67e-04       1.70e-04
sigma2         0.0434 0.00631 7.05e-05       7.17e-05
theta.pred101  0.5229 0.07654 8.56e-04       9.91e-04

2. Quantiles for each variable:

                  2.5%     50%  97.5%
alpha         -0.66036 -0.2553 0.1412
beta           0.00186  0.6918 1.4467
betapos        1.00000  1.0000 1.0000
cr[101]        0.59849  1.0053 1.4125
mi[101]        0.00000  1.0000 1.0000
mu             0.95939  1.0008 1.0415
sigma          0.18081  0.2068 0.2392
sigma2         0.03269  0.0428 0.0572
theta.pred101  0.38510  0.5195 0.6858

> ## logit:
> #           Mean      SD
> #alpha   -0.1608 0.12990
> #beta     0.4285 0.22266
> #betapos  0.9730 0.16209
> #cr[101]  1.0026 0.20976
> #mi[101]  0.5205 0.49961
> #mu       1.0007 0.02058
> #sigma    0.2083 0.01495
> #sigma2   0.0436 0.00631
> ## probit:
> #           Mean      SD
> #alpha   -0.1571 0.12639
> #beta     0.4248 0.22018
> #betapos  0.9754 0.15499
> #cr[101]  1.0037 0.20865
> #mi[101]  0.5108 0.49992
> #mu       1.0008 0.02084
> #sigma    0.2082 0.01492
> #sigma2   0.0436 0.00629
>
```

```
> exp(-0.2519 + 0.7130)/(1+exp(-0.2519 + 0.7130))

[1] 0.61328

> autocorr.plot(res)
> mcmcChain<-as.matrix(res)
> ## correlation:
> plot(mcmcChain[,1],mcmcChain[,2])
> ## prob that beta is positive:
> pnorm(0,mean=0.7011, sd=0.37719,lower.tail=F)

[1] 0.96847

> ## matches up:
> counts<-table(mcmcChain[,2]>0)
> counts[2]/(sum(counts))

   TRUE
0.97537
```

14
Homework assignments

15
Closing remarks

This course only scratches the surface of bayesian methods.

What books should you read to learn more? Kruschke[1] is good and worth reading; it aims at a technical level below Lynch's book and could perhaps be read before Lynch (although I must admit I did not read Kruschke; I found the Lynch book much more helpful). Kruschke also has a good website accompanying the book:

http://www.indiana.edu/ kruschke/DoingBayesianDataAnalysis/

There are also lots and lots of amazing websites. Simon Jackman's web page for example, and this one by Patrick Breheny:

http://web.as.uky.edu/statistics/users/pbreheny/701/S13/notes.html

As is probably evident from these notes, I greatly enjoyed the BUGS book by Lunn et al., and Lynch's book (which also has online code). If you want to get more serious about this material, you should read Gelman and Hill (esp. the second half) and Gelman et al.'s new edition of Bayesian Data Analysis (BDA).

One major hurdle in reading Gelman et al.'s BDA and books of that type is the mathematics and probability theory. I find Kerns very good for the latter; for math the best book I know is Salas et al.[2] It is hard work and the book is very thick, but it gives you a thorough grounding. For probability theory, Ross' book is OK, but might give you an inferiority complex. In this context, I highly recommend doing the Graduate Certificate in Statistics taught online at Sheffield. This is a nine-month course and will give you almost all the background you need to read books like Gelman et al's BDA. You can read my review of this course on my statistics blog.

Finally, one major problem in my opinion with reading bayesian textbooks is the distracting ferocity of some of the bayesians out there. Some of them are pretty religious about bayesian methods, to the extent that one wonders whether they might be better off in a theology department. It is better to filter this out in your reading; we don't want to become religious bayesians, or, indeed, religious anythings. We just want to do science the best we can, and for that purpose it is

[1] John Kruschke. *Doing Bayesian Data Analysis: A Tutorial Introduction with R.* Academic Press, 2010

[2] S.L. Salas, E. Hille, and G.J. Etgen. *Calculus: One and several variables.* Wiley, 2003

enough to recognize that bayesian methods have their place and value. Religious bayesians keep coming up with ways to sell their position, by claiming things like: bayes can solve the crisis in psychology. It can't; it's possible to do bad statistics even with bayes. The real problem is the lack of education about statistical theory, whether frequentist or bayesian. That's the problem we should be trying to solve.

16
Bibliography

[1] Jim Albert. *Bayesian computation with R*. Springer, 2009.

[2] R. H. Baayen. *Practical data analysis for the language sciences*. Cambridge University Press, 2008.

[3] John Barnard, Robert McCulloch, and Xiao-Li Meng. Modeling covariance matrices in terms of standard deviations and correlations, with application to shrinkage. *Statistica Sinica*, 10:1281–1311, 2000.

[4] George E.P. Box. *An accidental statistician: The life and memories of George EP Box*. Wiley, 2013.

[5] George E.P. Box and David R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 211–252, 1964.

[6] Ronald Christensen, Wesley O Johnson, Adam J Branscum, and Timothy E Hanson. *Bayesian ideas and data analysis: An introduction for scientists and statisticians*. CRC Press, 2011.

[7] Annette J Dobson. *An introduction to generalized linear models*. CRC press, 2010.

[8] A. Gelman and J. Hill. *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press, Cambridge, UK, 2007.

[9] Andrew Gelman. Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper). *Bayesian analysis*, 1(3):515–534, 2006.

[10] Andrew Gelman and Donald B Rubin. Inference from iterative simulation using multiple sequences. *Statistical science*, pages 457–472, 1992.

[11] E. Gibson and I. Wu. Processing Chinese relative clauses in context. Language and Cognitive Processes (in press), 2011.

[12] J. Gilbert and C.R. Jordan. *Guide to mathematical methods.* Macmillan, 2002.

[13] Walter R Gilks, Sylvia Richardson, and David J Spiegelhalter. *Markov chain Monte Carlo in practice*, volume 2. CRC press, 1996.

[14] C.M. Grinstead and J.L. Snell. *Introduction to probability.* American Mathematical Society, 1997.

[15] G. Jay Kerns. *Introduction to Probability and Statistics Using R.* 2010.

[16] André I Khuri. *Advanced calculus with applications in statistics*, volume 486. Wiley, 2003.

[17] John Kruschke. *Doing Bayesian Data Analysis: A Tutorial Introduction with R.* Academic Press, 2010.

[18] Peter M Lee. *Bayesian statistics: An introduction.* John Wiley & Sons, 2012.

[19] David Lunn, Chris Jackson, David J Spiegelhalter, Nicky Best, and Andrew Thomas. *The BUGS book: A practical introduction to Bayesian analysis*, volume 98. CRC Press, 2012.

[20] Scott Michael Lynch. *Introduction to applied Bayesian statistics and estimation for social scientists.* Springer, 2007.

[21] A. James O'Malley and Alan M. Zaslavsky. Domain-level covariance analysis for multilevel survey data with structured nonresponse. *Journal of the American Statistical Association*, 103:1405–1418, 2008.

[22] Sheldon Ross. *A first course in probability.* Pearson Education, 2002.

[23] S.L. Salas, E. Hille, and G.J. Etgen. *Calculus: One and several variables.* Wiley, 2003.

[24] Shravan Vasishth, Zhong Chen, Qiang Li, and Gueilan Guo. Processing Chinese relative clauses: Evidence for the subject-relative advantage. *PLoS ONE*, 8(10):e77006, 10 2013.