2011-07-06

An Introduction to Bayesian Methodology via WinBUGS and PROC MCMC

Heidi Lula Lindsey
Brigham Young University - Provo

An Introduction to Bayesian Methodology via WinBUGS & PROC MCMC

Heidi L. Lindsey

A Project submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Gilbert W. Fellingham, Chair
William F. Christensen
Scott D. Grimshaw

Department of Statistics

Brigham Young University

August 2011

ABSTRACT

An Introduction to Bayesian Methodology
via WinBUGS & PROC MCMC

Heidi L. Lindsey
Department of Statistics, BYU
Master of Science

Bayesian statistical methods have long been computationally out of reach because the analysis often requires integration of high-dimensional functions. Recent advancements in computational tools to apply Markov Chain Monte Carlo (MCMC) methods are making Bayesian data analysis accessible for all statisticians. Two such computer tools are WinBUGS and SAS® 9.2's PROC MCMC. Bayesian methodology will be introduced through discussion of fourteen statistical examples with code and computer output to demonstrate the power of these computational tools in a wide variety of settings.

Keywords: Bayesian data analysis, WinBUGS, PROC MCMC, statistical examples

ACKNOWLEDGMENTS

I would like to thank Dr. Fellingham for giving me the opportunity to work on this project and for his patient tutelage. I would also like to acknowledge that my work here at BYU was made possible by the love and support of my family, Tobias, Victoria and Connor.

CONTENTS

CHAPTER 1

INTRODUCTION

The purpose of this project is to create a primer on the use of Bayesian statistical methods as implemented in the computer programs WinBUGS and PROC MCMC in SAS® 9.2. This primer will illustrate these computer tools by demonstrating fourteen examples.

Bayesian statistical methods are more prevalent than in the past because of computational advances. However, proper training in the use of Bayesian methods is not as readily available as training in frequentist methodology. Therefore, this primer will serve as a guide for statisticians who desire to implement Bayesian methods but lack training.

WinBUGS is software that was developed by the Bayesian inference Using Gibbs Sampling (BUGS) project (BUGS 1996-2008). This group was concerned with flexible software for Bayesian analysis of complex statistical models using Markov chain Monte Carlo (MCMC) methods. The project began in 1989 in the MRC Biostatistics Unit of Cambridge University under the direction of David Spiegelhalter and chief programmer, Andrew Thomas. In 1996 the project expanded to include the Imperial College School of Medicine at St Mary's, London with the influence of Nicky Best, Jon Wakefield, and Dave Lunn. In 2004, Andrew Thomas moved to the University of Helsinki, Finland and began work on OpenBUGS while Nicky Best, Jon Wakefield, and Dave Lunn continued work on WinBUGS. (see OpenBUGS 2004)

The MCMC proceedure in SAS® 9.2 also uses Markov chain Monte Carlo (MCMC) simulation. PROC MCMC is a general purpose tool in SAS® 9.2 which one can utilize to implement Bayesian methods.

In both computer applications, a likelihood function for the data is proposed along with prior distributions for the parameters. Then relying on the notion that the appropriate

posterior distributions for the parameters in question are scaled products of the likelihood times the prior, the programs draw from the appropriate posterior distributions, producing summary diagnostic statistics computed from these draws.

The fourteen examples include: (1) one sample gamma, (2) two sample t-test, (3) linear regression, (4) multiple regression, (5) one-way ANOVA, (6) factorial design, (7) analysis of covariance, (8) linear mixed model, (9) random coefficient model, (10) logistic regression, (11) logistic regression with random effect, (12) Poisson model, (13) Poisson regression, and (14) survival model with censored data. These examples will demonstrate how the implementation of Bayesian methods is supported by these computational tools. A discussion of the computer output will also be included.

CHAPTER 2

BACKGROUND

Bayesian data analysis employs practical methods for making inferences from data using probability models for observed quantities about which one desires to learn. These methods are based on the work of Thomas Bayes, an English mathematician and Presbyterian minister who lived from 1702 – 1761 and formulated a probability theorem that bears his name. In an essay that was published after his death in 1763, Thomas Bayes presented a rule based on probability according to which "we ought to estimate the chance that the probability for the happening of an event perfectly unknown, should lie between any two named degrees of probabilty."(see Price 1763)

He wanted to use a set of binomial data, comprising of the number of successes out of a fixed number of attempts, to learn about the underlying chance of success for any randomly chosen event. Bayes' key contribution was to use a probability distribution to represent all of the uncertainty involved in the event space. This distribution represents the uncertainty due to a lack of knowledge concerning the underlying relationships governing the probability of future events, such as the uncertainty in a game of chance or a medical outcome. The essential characteristic of Bayesian methods is the explicit handling of probability in such a way as to incorporate prior beliefs or prior events into the model for the purpose of quantifying the uncertainty associated with the event of interest in the statistical data analysis.

Bayes' theorem is founded in probability theory, uses probability in its structure, and the theorem's approach follows the scientific method when appropriately implemented by a researcher working to predict the chance of the occurrence of an event of interest. It is flexible in that it can be employed to analyze simple as well as complex situations. A

powerful result is that all conclusions from the use of Bayes' theorem strictly obey the laws of probability.

2.1 Probability

We now provide a review of probability–vocabulary, theorems, and examples–that might be useful to prepare someone for further study in Bayesian methods.

Outcome: The building blocks of events. A single happening.

Event: A combination of outcomes, or a set of outcomes that are of interest.

Universal Event: The event that includes all possible events or outcomes. Also referred to as *sample space*, which is the set of all possible outcomes of a particular experiment.

Experiment: Any process that facilitates researchers in obtaining observations.

Union: The union of two sets, A and B, written as $A \cup B$, is the set of outcomes that belong to A, B, or both. For example,

- Let $A = \{12, 24, 36\}$ and $B = \{8, 10, 12\}$,
- $A \cup B = \{8, 10, 12, 24, 36\}$.

Intersection: The intersection of two sets, A and B, written as $A \cap B$, is the set of outcomes that belong to both A and B. For example,

- Let $A = \{12, 24, 36\}$ and $B = \{8, 10, 12\}$,
- $A \cap B = \{12\}$.

Complement: The set of outcomes from the sample space that do not contain any outcomes that are in set A. The complement is written as $^{\sim}A$ and is read as "not A". For example,

- Let the universal set $U = \{8, 10, 12, 24, 36, 40, 48\}$ and let $A = \{12, 24, 36\}$,
- $^{\sim}A = \{8, 10, 40, 48\}$.

Empty Set: The set consisting of no outcomes and written as $\emptyset$. A related term is *Impossible Event* which is an event that cannot happen.

Mutually Exclusive Events: Events that have no outcomes in common; events that have no overlap in outcomes. For example,

- Let $A = \{12, 24, 36\}$ and $C = \{9, 11\}$,
- $A \cap C = \emptyset$.

Probability: A value that represents how likely it is that an event will occur.
The following probability statements are taken as axiomatic:

1. If A is an event (i.e., a combination of outcomes, or a set of outcomes that is of interest), then $P(A) \geq 0$.
2. If U is the largest event possible, then $P(U) = 1$ (a certain event, it has to happen). U is the sample space.
3. If events A and B are mutually exclusive events, then $P(A \cup B) = P(A) + P(B)$. The probability of the union of two mutually exclusive sets is the sum of their respective probabilities. This is sometimes referred to as the Law of Total Probability.

   Using our probability axioms, it may be shown that:

- $P(\emptyset) = 0$
- $P(A) \leq 1$
- $P(A \cup {}^{\sim}A) = P(A) + P({}^{\sim}A) = 1$
- $P({}^{\sim}A) = 1 - P(A)$
- Examples:
  - Event A: Using a single die, roll an odd number = {1,3,5}
  - Event B: Using a single die, roll a four = {4}
  - Event U: Using a single die, roll = {1, 2, 3, 4, 5, 6}
  - $P(A \cup B) = P(A) + P(B) = \frac{3}{6} + \frac{1}{6} = \frac{4}{6} = \frac{2}{3}$
  - For a football game between BYU and SDSU, P(BYU Win and SDSU Win) = $P(BYU_{win} \cap SDSU_{win}) = \emptyset$
  - $P(A \cap {}^{\sim}A) = P(\emptyset) = 0$
  - $P(BYU_{win} \cup SDSU_{win}) = 1$
  - $P(A \cup {}^{\sim}A) = 1$

Fundamental Theorem of Counting: If an event can happen in m ways and another event can happen in n ways, then the event of their union can happen in $m \cdot n$ ways.

- Examples:
  - Event A: Selecting one shirt from a closet of ten shirts.
  - Event B: Selecting one pair of pants from a closet of seven pairs.
  - Therefore, A may happen in ten ways and B may happen in seven ways.
  - Thus, $(A \cup B)$ can happen together in a total of $10 \cdot 7 = 70$ ways.

Probabilities may be assigned to outcomes. If all outcomes are equally likely, then each outcome may logically be given an equal probability. But sometimes events are not equally likely. What if a die is weighted or loaded? Then one side is more likely to land up than another side. Sometimes additional information is obtained that informs us as to the probability of an event.

There are different ways to assign probabilities to events:

1. Equally Likely, i.e., flip a fair coin or roll a fair die
2. Long Run Frequency, i.e., conduct an experiment 1,000 or more times and then count the frequency of the outcomes
3. Degree of Belief, i.e., ask someone to state their belief of the probability of an event, then you ask a series of further questions, a calibration experiment, to hone in their personal degree of belief relative to the probability of an event happening. (This one makes people uncomfortable because your belief could be different than my belief.)

Joint Probability: Two events happened at the same time; $P(A \cap B)$ is read as the "joint probability of A and B".

Example:

- Event A: Using a single die, roll an odd number, $A = \{1, 3, 5\}$
- Event B: Using a single die, roll a number greater than three, $B = \{4, 5, 6\}$
- $(A \cap B) = \{5\}$; five is the only outcome that is in both sets.
- $P(A \cap B) = \frac{1}{6}$
- Event C: Using a single die, roll an even number $C = \{2, 4, 6\}$
- $P(A \cap C) = \emptyset$; A and C are mutually exclusive because they have no events in common.

Conditional Probability: We take the following as a definition, although we will attempt to show that is is intuitive. For two events, A and B in a sample space S, and $P(B) > 0$, then the conditional probability of A given B has occurred, written as $P(A|B)$, is

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

Intuitively, knowing that event B happened may tell us something about event A. Note that in this calculation of the conditional probability, B shrinks the sample space of S such that B becomes the new sample space, see figure 2.1.

- Conditioning on B occurring, shrinks the probability space. We are only working in the space of B. Figure 2.1 shows this with a Venn Diagram.

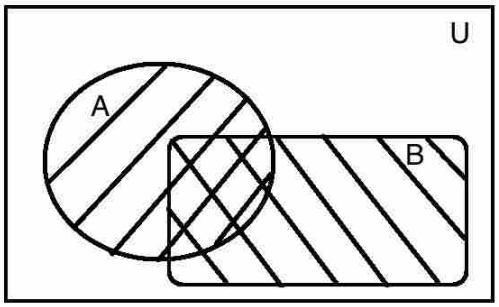


Figure 2.1: The conditional probability of event A given B is only the overlap space of A and B. The probability of the universal set is one: P(U)=1.

  - If A has occurred, it can only occur in the overlap space.
  - We need to scale the probability by dividing by $P(B)$.

- Consider the following sets of events:
  - Event A: Using a single die, roll an odd number, $A = \{1,3,5\}$
  - Event B: Using a single die, roll a number larger than three, $B = \{4, 5, 6\}$

- Event C: Using a single die, roll an even number, $C = \{2, 4, 6\}$

- If you know that event B happened, what is the probability now that an odd number was rolled, i.e. $P(A|B)$?
  - The $P(A)$, the unconditional probability that an odd number is rolled, $= \frac{1}{2}$.
  - The $P(B)$, the unconditional probability that a number larger than three is rolled, $= \frac{1}{2}$.
  - The $P(C)$, the unconditional probability that an even number is rolled, $= \frac{1}{2}$.
  - However, the conditional probability that an odd number was rolled given a number greater than three has occurred, is one-third.

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{\frac{1}{6}}{\frac{1}{2}} = \frac{1}{3}$$

- Bayes' Theorem

  Using the definition of conditional probability:

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

  Similarly,

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

- Thus,

$$P(A \cap B) = P(A|B)P(B)$$

$$P(B \cap A) = P(B|A)P(A)$$

$$\Longrightarrow$$

$$P(A|B)P(B) = P(B|A)P(A)$$

$$\Longrightarrow$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

Similarly,

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}.$$

These last two statements give Bayes' Theorem in its most basic form.

The following example demonstrates conditional probability and the fundamental theorem of counting.

- What is the probability that there is a common birthday among the individuals at any gathering of 25 people?
- $P(Common) = 1 - P(\sim Common)$
- $P(\sim Common) = \frac{365}{365} \cdot \frac{364}{365} \cdot \frac{363}{365} \ldots$ until the last person present, or $\frac{365-24}{365}$
  - The first person could have a birthday on any of the 365 days of the year. The second person cannot have a birthday on the same day that the first person has theirs, so this person has 364 days they could have a birthday. The third person has two days that their birthday cannot be on, so their birthday could be on any of the remaining 363 days. This continues until the last person present, then these conditional probability fractions can all be multiplied because $P(A \cap B) = P(B|A)P(A)$.
- This simplifies to: $\frac{\left(\frac{365!}{340!}\right)}{365^{25}}$
- The result is that $P(\sim Common) = 0.4313003$
- $P(Common) = 1 - P(\sim Common) = 1 - 0.4313003 = 0.5686997$

Consider mutually exclusive events, A and B. What is $P(A|B)$? Figure 2.2 demonstrates that this is an impossible event, $P(A|B) = 0$. Because A and B are mutually exclusive, knowing that B happened leads to the conclusion that A did not happen.

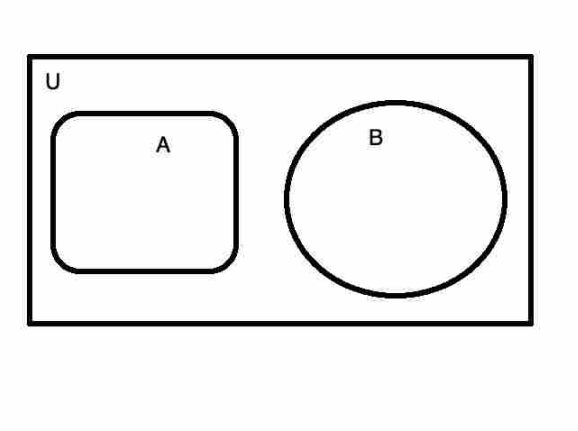


Figure 2.2: The conditional probability of A given B is zero here.

Law of Total Probability: Let's revisit the Law of Total Probability, the third axiom of probability. In figure 2.3:

$$\begin{aligned} P(B) &= P(B \cap A) \cup P(B \cap \ ^{\sim}A) \\ &= P(B \cap A) + P(B \cap \ ^{\sim}A) \\ &= P(B|A)P(A) + P(B|^{\sim}A)P(^{\sim}A). \end{aligned}$$

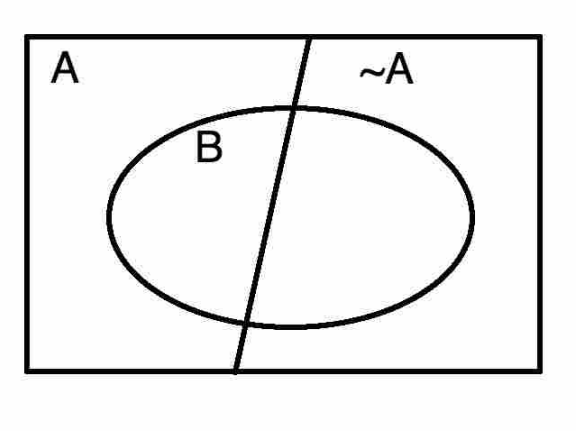


Figure 2.3: Demonstrating the law of total probability.

- Recall these conditional probability statements:

- $P(A|B) = \frac{P(A \cap B)}{P(B)}$
- $P(B|A) = \frac{P(A \cap B)}{P(A)}$
- $P(A \cap B) = P(B|A) \cdot P(A)$

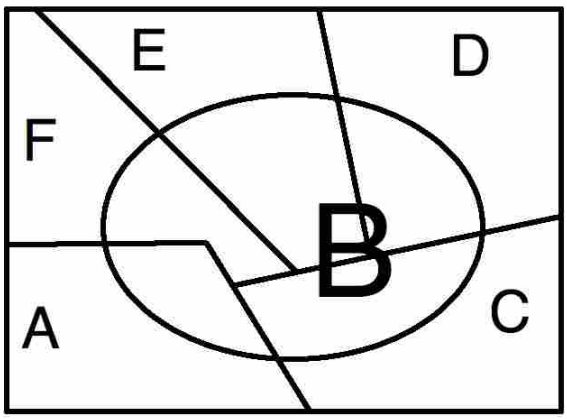


Figure 2.4: Extending the law of total probability.

In figure 2.4:

$$P(B) = P(B|A)P(A) + P(B|C)P(C) + P(B|D)P(D) + P(B|E)P(E) + P(B|F)P(F)$$

As long as the sample space is partitioned into mutually exclusive events, the individual probabilities can be summed, by the law of total probability.

The next example demonstrates this concept. Tovi is in a chess tournament and wants to know the probability he will win his next match, but there are two people he might play because they are still playing their game with each other and the winner is undetermined. Tovi wants to know the unconditional probability that he will win. Consider the probability of each part:

$$P(\text{Win}|\text{Play John}) = \frac{7}{10}$$
$$P(\text{Win}|\text{Play Maritza}) = \frac{4}{10}$$
$$P(\text{Play John}) = \frac{2}{5}$$
$$P(\text{Play Maritza}) = \frac{3}{5} = P(\sim \text{Play John})$$

$$
\begin{aligned}
P(\text{Win}) &= P(\text{Win}|\text{Play John})P(\text{Play John}) + P(\text{Win}|\sim \text{Play John})P(\sim \text{Play John}) \\
&= \frac{7}{10} \cdot \frac{2}{5} + \frac{4}{10} \cdot \frac{3}{5} \\
&= \frac{14}{50} + \frac{12}{50} \\
&= \frac{26}{50} = \frac{13}{25}
\end{aligned}
$$

Therefore, with a $\frac{13}{25}$ probability of winning, Tovi will win more often than lose if he plays over and over. This also means that if Tovi bets to win, he will win money if this scenario could be repeated over and over.

Bayes' Theorem: Now, combining the results from page 10 and page 12, we can state Bayes' Theorem another way. For any two events A and B, with $P(B) > 0$,

$$
\begin{aligned}
P(A|B) &= \frac{P(B|A)P(A)}{P(B)} && \text{from p.10} \\
&= \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\sim A)P(\sim A).} && \text{from p.12}
\end{aligned}
$$

However, the law of total probability allows for Bayes' Theorem to be extended to any partition of the sample space into mutually exclusive events. Let $A_1, A_2, \ldots A_i$ be such a partition and let B be any subset of the sample space. Then for each $j = 1, 2, \ldots, i$,

$$
\begin{aligned}
P(A_j|B) &= \frac{P(B|A_j)P(A_j)}{P(B)} \\
&= \frac{P(B|A_j)P(A_j)}{\sum_{j=1}^{i} P(B|A_j)P(A_j)}.
\end{aligned}
$$

Let's return to Tovi's chess tournament and suppose Tovi tells you he won. Can we determine the conditional probability he played John given he won? $P(\text{Played John}|\text{Tovi Won})$. Can this be unraveled?

Let's use Bayes' rule:

$$P(J_{\text{played John}}|W_{\text{win}}) = \frac{P(J \cap W)}{P(W)} = \frac{P(W|J)P(J)}{P(W|J)P(J) + P(W| \sim J)P(\sim J)}$$
$$= \frac{\frac{7}{10} \cdot \frac{2}{5}}{\frac{13}{25}}$$
$$= \frac{\frac{14}{50}}{\frac{26}{50}}$$
$$= \frac{14}{26} = \frac{7}{13}$$

Note, conditional probability allows us to formulate the following statements:

- $P(J \cap W) = P(J|W)P(W)$
- $P(W|J) = \frac{P(J \cap W)}{P(J)}$
- $P(J \cap W) = P(W|J)P(J)$

Bayes' rule will be further demonstrated through a discussion of the solution to the Monty Hall problem (Let's Make A Deal Game). Three boxes are presented to you as the contestant. One box has the key to a new car. Two boxes contain goats, or something equally non-desirable.

1. Play begins and you pick a box.
2. Before showing you what is in the box you picked, the MC shows you what is in one of the other two boxes that you did not pick. We will assume that he knows what is in the boxes and that the box shown to you will never have the key.
3. Now you are asked if you want to stay with your chosen box, or switch to the other box: stay or switch? What is the "right" choice? Is there a choice that can increase your probability of winning?

In the beginning of the game, you have no prior probability of preferring a given box.

$$P(\text{Box 1 wins}) = P(\text{Box 2 wins}) = P(\text{Box 3 wins}) = \frac{1}{3}$$

Let's say you chose box 2 as the box that holds the key and let's say the MC shows you box 1 because he knows the key is not in box 1:

$$\begin{aligned} P(\text{key in your box (2)}|\text{MC shows empty box (1)}) &= \frac{P(1|2) \cdot P(2)}{P(1)} \\ &= \frac{P(1|2) \cdot P(2)}{P(1|1) \cdot P(1) + P(1|2) \cdot P(2) + P(1|3) \cdot P(3)} \\ &= \frac{\frac{1}{2} \cdot \frac{1}{3}}{0 \cdot \frac{1}{3} + \frac{1}{2} \cdot \frac{1}{3} + 1 \cdot \frac{1}{3}} \\ &= \frac{\frac{1}{6}}{\frac{1}{6} + \frac{1}{3}} \\ &= \frac{\frac{1}{6}}{\frac{3}{6}} = \frac{1}{3} \end{aligned}$$

Probability has not changed.

–Additionally–

$$\begin{aligned} P(\text{key in unselected box (3)}|\text{MC shows empty box (1)}) &= \frac{P(1|3) \cdot P(3)}{P(1)} \\ &= \frac{P(1|3) \cdot P(3)}{P(1|1) \cdot P(1) + P(1|2) \cdot P(2) + P(1|3) \cdot P(3)} \\ &= \frac{1 \cdot \frac{1}{3}}{0 \cdot \frac{1}{3} + \frac{1}{2} \cdot \frac{1}{3} + 1 \cdot \frac{1}{3}} \\ &= \frac{\frac{1}{3}}{\frac{1}{6} + \frac{1}{3}} \\ &= \frac{\frac{1}{3}}{\frac{3}{6}} = \frac{1}{3} \cdot \frac{6}{3} = \frac{2}{3} \end{aligned}$$

$$\Rightarrow \text{If you stay, } P(\text{You Win}) = \frac{1}{3};$$

$$\Rightarrow \text{If you switch, } P(\text{You Win}) = \frac{2}{3}.$$

These new probability values are a result of the probability from Box 1 essentially being transferred to Box 3. Switching is the correct thing to do, but you still may not win. Switching raises the probability of a win, but it does not guarantee a win.

Independent Events: Intuitively, independence means that knowing something about one event informs nothing about the other event. By definition, two events are statistically independent if

$$P(A \cap B) = P(A) \cdot P(B).$$

Example, keeping Event A, Event B, and Event C as previously defined,

- Event A: Using a single die, roll an odd number, $A = \{1, 3, 5\}$
- Event B: Using a single die, roll a number greater than three, $B = \{4, 5, 6\}$
- Event C: Using a single die, roll an even number, $C = \{2, 4, 6\}$
- Recall from p.7 and p.8, $P(A \cap B) = \frac{1}{6}$, $P(A) = \frac{1}{2}$, and $P(B) = \frac{1}{2}$
- Are A and B independent events?
- Is $P(A) \cdot P(B) = \frac{1}{6}$?

  $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4} \neq \frac{1}{6}$
- No, A and B are not independent events.
- $B \cap C = \{4, 6\}$; four and six are elements in both sets
- $P(B \cap C) = \frac{2}{6} = \frac{1}{3}$
- Are B and C independent events?
- Is $P(B) \cdot P(C) = \frac{1}{3}$?

  $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4} \neq \frac{1}{3}$
- No, B and C are not independent events.

Independent probabilities can be extended beyond two events in the following manner:

$$P(A \cap B \cap C \ldots \cap Q) = P(A) \cdot P(B) \cdot P(C) \cdot \ldots \cdot P(Q).$$

Note: This extension of independent probabilities does not imply pairwise independence.

Exchangeability: Two experiments are considered exchangeable if three conditions are met.

1. Possible outcomes are the same in both experiments
2. Probability of each outcome is the same in both experiments
3. The conditional probability of the second given the first is the same as the conditional probability of the first given the second.

Some experiments are independent which is helpful when calculating probabilities. However, some experiments are not independent but they are exchangeable which is helpful for Bayesian statistics, because exchangeable experiments have the same properties as independent experiments. The characteristic of exchangeability is not as strong as independence.

- Let's say you select two cards from a bowl of four cards numbered 1 through 4 without replacing the first card.
- The first draw is not independent with the second draw.
  - The probability of any given number on the first draw is one-fourth.
  - The probability of the remaining numbers on the second draw is one-third.
  - Thus the probability for each of the possible pairs to be selected is one-twelveth, see figure 2.5.

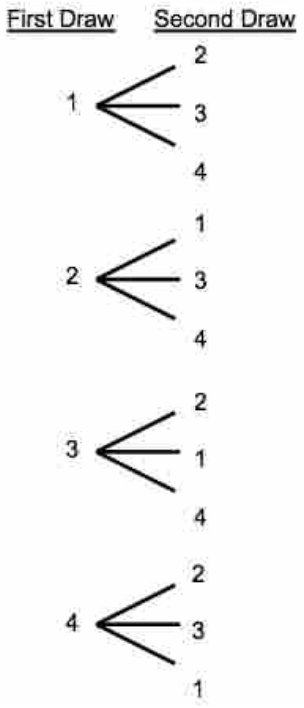


Figure 2.5:

Suppose two experiments were carried out on drawing cards from
the bowl. Both experiments have the same set of four cards. Remember there are three criteria for events to be exchangeable:

1. Possible outcomes are the same in both experiments.
   - $\{1, 2, 3, 4\}$ is the same as $\{2, 1, 3, 4\}$
2. The probability of each outcome is the same in both experiments.
   - The probability of drawing a 1 first is $\frac{1}{4}$

     $$P(1_1) = \frac{1}{4}$$

   - The probability of drawing a 1 second is $\frac{3}{12} = \frac{1}{4}$. This may also be shown with a tree diagram outlining all possible outcomes, see figure 2.5.

     $$P(1_2) = \frac{1}{12} + \frac{1}{12} + \frac{1}{12} = \frac{3}{12} = \frac{1}{4}$$

3. The conditional probability of the second given the first is the same as the conditional probability of the first given the second.
   - The conditional probability of drawing a 2 second given a 1 was drawn first is $\frac{1}{3}$

     $$P(2_2|1_1) = \frac{1}{3}$$

   - The conditional probability of drawing a 1 second given a 2 was drawn first is $\frac{1}{3}$

     $$P(1_2|2_1) = \frac{1}{3}$$

2.2 Probability Density Functions

Random Variable: A random variable is a function that assigns a single numerical value to each outcome of an experiment. The value is specific to the outcome from a given experiment. For example, if our experiment involved flipping a coin four times and recording

each outcome, the sample space includes outcomes of heads or tails and the random variable associated with this experiment could be to assign a numerical value of 1 if the coin landed heads and a 0 if the coin landed tails. These numerical values would be recorded as the experiment progresses and are referred to as random because we do not know what the next value is until after the experiment has been conducted.

Probability Density Function: A probability density function (PDF) is a function that assigns probability to each random variable in the data. Technically, if the random variable is discrete, the function is a probability mass function (pmf) and if the random variable is continuous, the function is a probability density function (pdf). However, we will refer to these functions collectively as PDFs. It is recommended that the reader become familiar with the common PDFs because they are a crucial part of how Baye's Theorem is utilized in the Bayesian approach to data analysis.

Parameter: Something that describes a population, is used in a PDF, and is represented with a Greek letter. The parameter controls the value of the function.

Statistic: A quantity we compute from the data.

PDF Examples.

- The Bernoulli(θ) PDF describes data limited to two possible outcomes, a success (1) or a failure (0). The parameter θ describes the probability of a success and $0 \leq \theta \leq 1$, while $x = \{0, 1\}$,

$$f(x|\theta) = \theta^x (1-\theta)^{1-x}.$$

  - Mean and variance:

$$EX = \theta, \qquad VarX = \theta(1-\theta).$$

- The Beta(α, β) PDF describes data limited to outcomes from 0 to 1 inclusive, $0 \leq x \leq 1$, with parameters $\alpha > 0$ and $\beta > 0$,

$$f(x|\alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}.$$

  - Mean and variance:

$$EX = \frac{\alpha}{\alpha + \beta}, \qquad VarX = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}.$$

- The Gaussian(μ, σ) PDF, also called the normal distribution, describes data that can fall anywhere in the $\Re$ number line with parameters $\sigma > 0$ and $-\infty \leq \mu \leq \infty$,

$$f(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}.$$

  - Mean and variance:

$$EX = \mu, \qquad VarX = \sigma^2.$$

- The Gamma(shape=α, scale=β) PDF describes data limited to positive outcomes, $0 \leq x < \infty$, with parameters $\alpha > 0$ and $\beta > 0$,

$$f(x|\alpha, \beta) = \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha-1} e^{-\frac{x}{\beta}}.$$

  - Mean and variance:

$$EX = \alpha\beta, \qquad VarX = \alpha\beta^2.$$

- The Inverse Gamma(shape=α, scale=β) PDF describes data limited to positive outcomes, $0 \leq x < \infty$, with parameters $\alpha > 0$ and $\beta > 0$,

$$f(x|\alpha, \beta) = \frac{1}{\Gamma(\alpha)\beta^\alpha} (x)^{-(\alpha+1)} e^{-\frac{1}{\beta x}}.$$

  - Mean and variance:

$$EX = \frac{1}{\beta(\alpha - 1)} \text{ for } \alpha > 1, \qquad VarX = \frac{1}{(\alpha - 2)\beta^2(\alpha - 1)^2} \text{ for } \alpha > 2.$$

- The Poisson(λ) PDF describes data limited to the whole numbers, $x = 0, 1, \ldots$, with parameter $0 \leq \lambda < \infty$,

$$f(x|\lambda) = \frac{e^{-\lambda}\lambda^x}{x!}.$$

  - Mean and variance:

$$EX = \lambda, \qquad VarX = \lambda.$$

Likelihood: Probability of obtaining a particular set of data. If data are independent, or exchangeable, the likelihood may be computed by multiplying the probabilities associated with each data point.

$$f(\mathbf{X}|\theta) = \prod_{i=1}^{n} f(x_i|\theta) = \text{Lik}(\mathbf{X}|\theta)$$

1. Frequentists make inferences on parameters from the likelihood function to obtain a possible value for parameters. Frequentists believe that parameters are fixed but unknown.

2. Bayesians put a prior distribution on the likelihood to obtain a posterior distribution describing each parameter. Bayesians believe that since we don't know the value of the parameter, our uncertainty about the value can be appropriately described using a PDF.

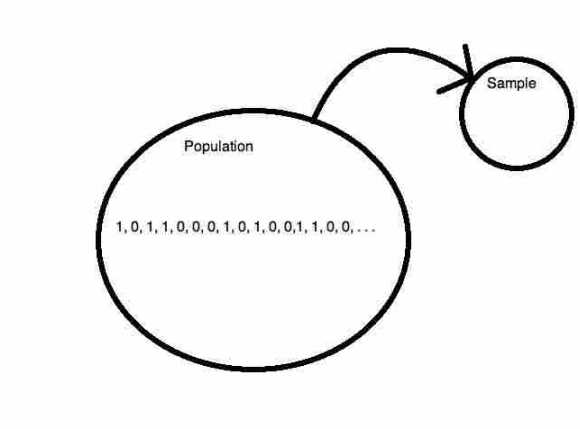


Figure 2.6: A population of Bernoulli data.

An example to demonstrate computing the likelihood.

Flip a coin once and compute the likelihood: Let's define tails = 0 and heads = 1. These data follow a Bernoulli likelihood, describing a series of successes, see figure 2.6.

Once we know what the data are, we can compute the vlue of the likelihood.

Computing the likelihood for a flip that yields a tail with θ=0.1:

$$\text{Lik}(x|\theta) = .1^0(1-0.1)^{(1-0)} = 0.9$$

Computing the likelihood for a flip that yields a tail with θ=0.2:

$$\text{Lik}(x|\theta) = .2^0(1-0.2)^{(1-0)} = 0.8$$

A different value for the parameter gives a different value for the likelihood.

Let's set $\theta = 0.1$ and gather more data by throwing a coin several more times. Thus we have the following data set:

$$\{0, 0, 0, 1, 1, 1, 0\}$$

Note: These trials are each independent; knowing one outcome doesn't tell me anything about the other outcomes. The event I am interested in is $P(T \cap T \cap T \cap H \cap H \cap H \cap T)$ and because these events are independent, I can multiply the individual probabilities for each event together: $P(T) \cdot P(T) \cdot P(T) \cdot P(H) \cdot P(H) \cdot P(H) \cdot P(T)$.

$$\text{Lik}(\text{data}|\theta = 0.1) = .1^0(1-.1)^{(1-0)} \cdot .1^0(1-.1)^{(1-0)} \cdot .1^0(1-.1)^{(1-0)} \cdot .1^1(1-.1)^{(1-1)} \cdot$$

$$.1^1(1-.1)^{(1-1)} \cdot .1^1(1-.1)^{(1-1)} \cdot .1^0(1-.1)^{(1-0)} \cdot$$

$$= .9 \cdot .9 \cdot .9 \cdot .1 \cdot .1 \cdot .1 \cdot .9$$

$$= 0.0006561$$

Computing the likelihood in general:

$$\begin{aligned}
\text{Lik} &= \prod_{i=1}^{n} \theta^{x_i}(1-\theta)^{(1-x_i)} \\
&= \theta^{x_1}(1-\theta)^{(1-x_1)} \cdot \theta^{x_2}(1-\theta)^{(1-x_2)} \cdot \theta^{x_3}(1-\theta)^{(1-x_3)} \cdot \ldots \cdot \theta^{x_n}(1-\theta)^{(1-x_n)} \\
&= \theta^{\sum x_i}(1-\theta)^{(n-\sum x_i)}.
\end{aligned}$$

Maximizing the Likelihood: It is possible to maximize the likelihood relative to the parameter, θ. To maximize the likelihood, take the derivative, set it equal to zero, and solve for the desired variable. However, we can make it easier to take the derivative of the function by taking the log of the function first. The resulting function will have the maximum y value at the same x value as the original function,

$$log(\text{Lik}) = \sum x_i log(\theta) + (n - \sum x_i) log(1 - \theta).$$

Now, remember that the x's are data and that we are maximizing with respect to θ. Thus, we will find the derivative with respect to θ,

$$\frac{\partial}{\partial \theta} = \frac{\sum x_i}{\theta} - \frac{(n - \sum x_i)}{(1 - \theta)}.$$

Now, set the derivative equal to zero,

$$\begin{aligned} 0 =& \frac{\sum x_i}{\hat{\theta}} - \frac{(n - \sum x_i)}{(1 - \hat{\theta})} \\ \frac{(n - \sum x_i)}{(1 - \hat{\theta})} =& \frac{\sum x_i}{\hat{\theta}}. \end{aligned}$$

Solving for $\hat{\theta}$,

$$\begin{aligned} \hat{\theta}(n - \sum x_i) &= \sum x_i (1 - \hat{\theta}) \\ n\hat{\theta} - \hat{\theta} \sum x_i &= \sum x_i - \hat{\theta} \sum x_i \\ \hat{\theta} &= \frac{\sum x_i}{n} \\ \hat{\theta} &= \bar{x}. \end{aligned}$$

Note: $\hat{\theta}$ is the maximum likelihood estimator and $\hat{\theta}$ maximizes the likelihood function. The statistic $\bar{x}$ estimates θ. Here then, according to the data, $\bar{x} = \frac{3}{7} \approx 0.429$.

Prior: The prior is uncertainty associated with a PDF selected to summarize previous belief about the parameter. Following is a sampling of PDF prior choices.

- If the data are described as a Bernoulli(π) PDF with $x \in (0, 1)$ and $0 \leq \pi \leq 1$, then a reasonable prior PDF to describe π is a Beta(α, β) with $0 \leq \pi \leq 1$, $\alpha > 0$, and $\beta > 0$.

- If the data are described as a Poisson(λ) PDF with $x = 0, 1, \ldots$ and $0 \leq \lambda < \infty$, then a reasonable prior PDF to describe λ is a Gamma(α, β) with $0 \leq \lambda < \infty$, $\alpha > 0$, and $\beta > 0$.

- If the data are described as a Normal(μ, σ) PDF with $-\infty < x < \infty$, $-\infty < \mu < \infty$, and $\sigma > 0$, then a reasonable prior PDF to describe μ is a Normal(μ_μ, σ_μ) with $-\infty < \mu < \infty$, $-\infty < \mu_\mu < \infty$, and $\sigma_\mu > 0$ and an Inverse Gamma($\alpha_\sigma, \beta_\sigma$) to describe σ with $0 \leq \sigma < \infty$, $\alpha_\sigma > 0$, and $\beta_\sigma > 0$.

Bayes' theorem can be thought of as a way of coherently updating our uncertainty in light of new evidence. This update is modeled with probability distributions that serve as a statement expressing uncertainty and results from a choice that is based on logical reasoning. Beginning with the assumption that a sample is an exchangeable sequence of random variables, $x_1, x_2, \ldots, x_n$, from a population of interest, means that the sequence at hand behaves like earlier samples, or that any order of the sample is equally likely. A sequence of independent and identically distributed random variables is exchangeable. Assumptions about exchangeability are equivalent to assuming events are independent conditional on some unknown parameter that has a prior probability distribution and a likelihood function describing the events.

The process of Bayesian data analysis follows three steps: (1) setting up a full probability model with an appropriate likelihood function to model the exchangeable sample conditioned on observed data

$$f(\mathbf{X}|\boldsymbol{\theta}) = \text{Lik}(\mathbf{X}|\boldsymbol{\theta});$$

(2) choosing prior probability distribution(s) to preserve the parameter space and model the prior probability associated with the parameter(s) in the likelihood

$$\pi(\boldsymbol{\theta});$$

and (3) evaluating the fit of the model and the implications of the resulting posterior distribution

$$p(\boldsymbol{\theta}|\mathbf{X}) = \frac{\prod_{i=1}^{n} f(\mathbf{X}|\boldsymbol{\theta}) \cdot \pi(\boldsymbol{\theta})}{\int_{\Omega} \prod_{i=1}^{n} f(\mathbf{X}|\boldsymbol{\theta}) \cdot \pi(\boldsymbol{\theta}) \partial \boldsymbol{\theta}}.$$

(Note that this final equation is a special use of Bayes' theorem.)

Posterior Probability Density Function part A: A probability density function describing the updated belief about the parameter that is based on the prior belief about the parameter and incorporates the data. Notice how this form follows Bayes' Rule:

$$p(\text{parameter}|\text{data}) = \frac{\text{Lik}(\text{data}|\text{parameter}) \cdot \text{Prior}(\text{parameter})}{\int \text{Lik}(\text{data}|\text{parameter})\text{Prior}(\text{parameter}) \partial \text{parameter}}$$

$$p(\boldsymbol{\theta}|\mathbf{X}) = \frac{\text{Lik}(\mathbf{X}|\boldsymbol{\theta}) \cdot \pi(\boldsymbol{\theta})}{\int_{\Omega} \text{Lik}(\mathbf{X}|\boldsymbol{\theta}) \cdot \pi(\boldsymbol{\theta}) \partial \boldsymbol{\theta}}$$

- Example: Back to the coin flip data set. This data was modeled with a Bernoulli likelihood. What is a reasonable choice for a prior distribution to model θ? The parameter here represents the probability of a coin flip and as such is limited to $0 \leq \theta \leq 1$. Therefore, a Beta PDF is a reasonable choice for a prior distribution on θ.

Normalizing Constant: The denominator in the posterior probability density function turns the numerator into a proper PDF because it appropriately scales the numerator.

$$\int \text{Lik}(\text{parameter}|\text{data})\text{Prior}(\text{parameter}) \partial \text{parameter}$$

Once the parameter has been integrated out, what remains is a constant. The constant can be put aside momentarily, as discussed later.

Posterior Probability Density Function part B: Putting the Bernoulli likelihood together with the Beta prior from the coin flip example:

$$\text{Post}(\theta|\text{data}) = \frac{\text{Lik}(\text{data}|\theta)\text{Prior}(\theta)}{\int \text{Lik}(\text{data}|\theta)\text{Prior}(\theta) \partial \theta}$$

$$\text{Post}(\theta|\text{data}) \propto \text{Lik}(\text{data}|\theta)\text{Prior}(\theta)$$

Note: the symbol $\propto$ means "is proportional to". The constants are put together, taken out, and "forgotten" about momentarily, while the variable parts are treated as proportional to what was there before.

And now, putting together the posterior density function, using Bayes' theorem.

$$\text{Post}(\theta|\text{data}) = \frac{\prod_{i=1}^{n} \theta^{x_i}(1-\theta)^{(1-x_i)} \cdot \dfrac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \cdot \theta^{a-1}(1-\theta)^{b-1}}{\displaystyle\int_0^1 \prod_{i=1}^{n} \theta^{x_i}(1-\theta)^{(1-x_i)} \cdot \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \cdot \theta^{a-1}(1-\theta)^{b-1}\partial\theta}$$

Any term that is a constant will be combined with the normalizing constant and momentarily ignored. The factors with $x's$ and $\theta's$ are of interest because they are variables, but constants will be ignored for now.

$$\text{Post}(\theta|\text{data}) = \frac{\prod_{i=1}^{n} \theta^{x_i}(1-\theta)^{(1-x_i)} \cdot \cancel{\dfrac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}} \cdot \theta^{a-1}(1-\theta)^{b-1}}{\displaystyle\int_0^1 \prod_{i=1}^{n} \theta^{x_i}(1-\theta)^{(1-x_i)} \cdot \cancel{\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}} \cdot \theta^{a-1}(1-\theta)^{b-1}\partial\theta}$$

constant

constant

$$\text{Post}(\theta|\text{data}) \propto \theta^{x_1}(1-\theta)^{1-x_1} \cdot \theta^{x_2}(1-\theta)^{1-x_2} \cdot \ldots \cdot \theta^{x_n}(1-\theta)^{1-x_n} \cdot \theta^{a-1}(1-\theta)^{b-1}$$

$$\text{Post}(\theta|\text{data}) \propto \theta^{(\sum x_i + a)-1}(1-\theta)^{(n-\sum x_i + b)-1}$$

This is a probability function whose support is from 0 to 1. The next step is to obtain a constant to multiply the function with so the function will integrate to 1. Compare the beta PDF with this last function. What is in place of the beta function's "a" and "b" in the last line above? Note that "a" $= (\sum x_i + a)$ and that "b" $= (n - \sum x_i + b)$.

Incorporating the new a and b into the beta PDF, we see that the posterior function is another Beta PDF with new values for a and b:

$$\text{Post}(\theta|\text{data}) = \theta^{(\sum x_i + a)-1}(1-\theta)^{(n-\sum x_i + b)-1} \frac{\Gamma(\sum x_i + a + n - \sum x_i + b)}{\Gamma(\sum x_i + a)\Gamma(n - \sum x_i + b)}$$

Conjugate Prior: The prior has the same functional form as the posterior. If the prior is a beta, the conjugate posterior will be a beta.

Practical use of the Bayesian approach requires careful consideration of challenging probability concepts, including the source of the prior distribution, the choice of a likelihood function, computation and summary of the posterior distribution in high-dimensional problems, and making a convincing presentation of the analysis. Advances in Bayesian data analysis have been made in the last twenty years due to the evolution of computational methods using the power of computers.

2.3 Markov chain Monte Carlo (MCMC)

A major limitation on the widespread implementation of Bayesian methods of data analysis was that obtaining the posterior distribution often required the integration of high-dimensional functions. This can be mathematically very difficult, and as such, inhibited the use of Bayesian methods since Bayes' first proposal on the subject in 1763. The advancement of computational methods has greatly simplified the application of Bayesian data analysis and made these methods more accessible for all statisticians. One such development is Markov chain Monte Carlo (MCMC).

Markov chain Monte Carlo methods include random walk Monte Carlo methods and are a class of algorithms for sampling from probability distributions based on constructing a Markov chain that has the desired distribution as its target distribution. The Monte Carlo method for multidimensional integrals simply consists of integrating over a random sampling of points instead of over a regular array of points. (Metropolis et al. 1953)

The chain begins at an initial value and is allowed to run for n iterations before the researcher keeps the draws. These first n iterations are referred to as a "burn-in", the value of n is usually a large number, and a trace plot of the drawn values against the iteration number guides in the selection of n. After n iterations or steps, the chain is kept and used

as a sample from the desired distribution. The quality of the sample improves as a function of the number of steps taken in the algorithm. MCMC is based on drawing values of $\boldsymbol{\theta}$ from approximate distributions and then correcting those draws to better approximate the target posterior distribution. Samples are drawn sequentially with the distribution of the sample draws depending on the last value drawn, thus forming a Markov chain. The next value drawn depends upon the current value.

Typically, it is not hard to construct a Markov chain that will have the desired properties. It is more difficult to determine the n steps that are needed to converge to the desired, stationary distribution within an acceptable error. The key to the method's success is not the Markov property, however, but rather that the approximate distributions are improved at each of the n steps in the simulation. Thus, the more steps that are taken, the closer is the convergence to the desired target distribution.

Metropolis Algorithm

Statistical MCMC methods have their roots in the Metropolis algorithm as presented by Metropolis et al. (1953) and later generalized and improved by Hastings from the University of Toronto (Hastings 1970). The Metropolis algorithm computes complex integrals by expressing them as expectations for some distribution and then estimating this expectation by drawing samples from that distribution. This method consists simply of "integrating over a random sampling of points instead of over a regular array of points." (Metropolis et al. 1953)

The Metropolis algorithm hinges on a function proportional to the distribution to be sampled. This function is a rejection/acceptance criteria and requires a candidate density from which draws are obtained and then fed into the function $h(\theta)$ to determine rejection or acceptance of that draw.

$$p(\boldsymbol{\theta}|\mathbf{X}) \propto g(\boldsymbol{\theta}) \equiv f(\mathbf{X}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}).$$

The algorithm begins by specifying a candidate or proposal density $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(t-1)})$ that is a valid density meeting all of the required conditions to be a valid density for every possible value of the conditioning variable $\boldsymbol{\theta^{(t-1)}}$ and also satisfies $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(t-1)}) = q(\boldsymbol{\theta}^{(t-1)}|\boldsymbol{\theta}^*)$, which means that q is symmetric in its arguments.

Here is a description of the Metropolis Algorithm: Given a starting value $\boldsymbol{\theta}^{(0)}$ at iteration $t = 0$, then for $t = 1, \ldots, T$, repeat:

1. Draw $\boldsymbol{\theta}^*$ from $q(\cdot|\boldsymbol{\theta}^{(t-1)})$
2. Compute the ratio $r = \frac{g(\boldsymbol{\theta}^*)}{g(\boldsymbol{\theta}^{(t-1)})}$
3. If $r \geq 1$, set $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^*$; if $r < 1$, set $\boldsymbol{\theta}^{(t)} = \begin{cases} \boldsymbol{\theta}^* \text{ with probability } r \\ \boldsymbol{\theta}^{(t-1)} \text{ with probability 1 - } r. \end{cases}$

It has been shown that a draw $\boldsymbol{\theta}^{(t)}$ converges in distribution to a draw from the true posterior density $p(\boldsymbol{\theta}|\mathbf{x})$. (Carlin and Louis 2009)

Gibbs Sampler

The Gibbs sampler, as introduced by Geman and Geman (1984), sparked a major increase in the application of Bayesian analysis, making Bayesian analysis feasible in practice. This method provides an approach that reduces the hard multivariate problem to a series of simple lower-dimensional problems. This method assumes the availability of all k full conditional distributions, one for each parameter, and is known to converge slowly in applications with a large number of k. The Gibbs sampler will sample from the full conditional distributions at each iteration and the collection of full conditional distributions uniquely determines the joint posterior distribution, $p(\boldsymbol{\theta}|\mathbf{X})$ along with all marginal posterior distributions $p(\theta_i|\mathbf{x})$, $i = 1, \ldots, k$.

Here is a description of the Gibbs Sampler Algorithm: For $t = 1, \ldots, T$, repeat:

Step 1: Draw $\theta_1^{(t)}$ from $p(\theta_1|\theta_2^{(t-1)}, \theta_3^{(t-1)}, \ldots, \theta_k^{(t-1)}, \mathbf{x})$

Step 2: Draw $\theta_2^{(t)}$ from $p(\theta_2|\theta_1^{(t)}, \theta_3^{(t-1)}, \ldots, \theta_k^{(t-1)}, \mathbf{x})$

$\vdots$

Step k: Draw $\theta_k^{(t)}$ from $p(\theta_k|\theta_1^{(t)}, \theta_2^{(t)}, \ldots, \theta_{k-1}^{(t)}, \mathbf{x})$

It has been shown that this k-tuple from the t^{th} iteration of this algorithm converges in distribution to a draw from the true joint posterior distribution $p(\boldsymbol{\theta}|\mathbf{x})$. Hence, for t sufficiently large, larger than the n "burn-in" iterations, $\boldsymbol{\theta}$ is a correlated sample from the true posterior from which posterior quantities of interest may be calculated. For example, a sample mean for $\hat{\theta}_3$ can estimate the posterior mean for θ_3. (Carlin and Louis 2009)

2.4 WinBUGS

In 1989, the BUGS (Bayesian inference Using Gibbs Sampling) project began under the direction of David Spiegelhalter and chief programmer Andrew Thomas in the MRC Biostatistics Unit, Cambridge, and led initially to the 'Classic' BUGS program. The Imperial College School of Medicine at St Mary's, London joined the project in 1996 with the work of Nicky Best, Jon Wakefield, and Dave Lunn (BUGS 1996-2008). Andrew Thomas moved to Helsinki, Finland in 2004 and began work on OpenBUGS at the University of Helsinki. (OpenBUGS 2004) Currently the program runs only in the Microsoft Windows operating system.

WinBUGS is a windows-based computer program designed to conduct Bayesian Analyses of complex statistical models using Markov chain Monte Carlo (MCMC) methods. It is a 'point-and-click' environment that utilizes Markov chain Monte Carlo computational power to analyze a wide class of Bayesian full probability models. Herein, models will be specified textually, but they may also be specified graphically. (Lunn et al. 2000)

WinBUGS is part of the BUGS project, which aims to make practical MCMC methods available to applied statisticians. In this program, the user specifies a model and starting values, and then a Markov chain simulation is automatically implemented for the resulting

posterior distribution. It can use either a standard 'point-and-click' windows interface for controlling the analysis, or can construct the model using a graphical interface called DoodleBUGS. WinBUGS is a stand-alone program that can also be called from other software, like R. For further information on this, see the OpenBUGS site.

MCMC algorithms are implemented in this program to generate simulated observations from the posterior distribution of the unknown quantities in the statistical model. With sufficiently many simulated observations, it is possible to get an accurate picture of the posterior distribution.

2.5 PROC MCMC

SAS® is a statistical program that was created to meet the need for a computerized statistics program to analyze vast amounts of agricultural data. The establishment of such software was all-important to members of the University Statisticians Southern Experiment Stations, a consortium of eight land-grant universities largely funded by the USDA. These schools came together under a grant from the National Institutes of Health in the development of SAS® . North Carolina State University became the leader of the consortium and the project found a home in the Statistics Department under the leadership of Jim Goodnight and Jim Barr (SAS 1976).

SAS® programs define a sequence of operations to be performed on data stored as tables. These operations are libraried as procedures or PROC commands. One of the procedures new to SAS® 9.2 is the PROC MCMC command. This is a general-purpose MCMC simulation procedure for fitting a wide range of Bayesian models. PROC MCMC uses a random walk Metropolis algorithm to obtain posterior samples. By default, PROC MCMC assumes that all observations in the data set are independent, and therefore exchangeable.

Unlike most other SAS® procedures, PROC MCMC is designed for Bayesian statistical analysis and inference. This procedure needs a likelihood function to be specified for the data and prior distributions for the parameters; hyperprior distributions are needed if the

model is hierarchical. Prior distributions for the parameters are specified with PRIOR statements and the likelihood function for the data is specified with a MODEL statement. This procedure bases its inferences from simulation rather than through analytic or numerical methods. The default algorithm is an adaptive blocked random walk Metropolis algorithm that uses a normal proposal distribution. Therefore, a second run of the same problem will produce slightly different answers from the first run unless the same random number seed is used. PROC MCMC saves the posterior sample draws in an output data set that can be used for further analysis and it also produces summary and diagnostic statistics.

CHAPTER 3

COMPUTER SYNTAX INTRODUCTION

3.1 WinBUGS

WinBUGS may be downloaded from the BUGS Project website at http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml. Users are not required to register in order to obtain a key for unrestricted use. Follow the installation instructions to obtain a key and appropriate download of WinBUGS.

In the help menu, users may access the user manual. It is here where you may read how WinBUGS parameterizes various distributions. Such knowledge is crucial as you specify prior distributions and likelihoods in your Bayesian models because WinBUGS syntax must be incorporated appropriately to obtain results for the desired model.

Computer syntax will be introduced by working through an example. This example contains seven data points that were simulated from a Gamma (shape=6, scale=8) distribution. It will be assumed that this distribution is unknown and the prior distributions for both α and β will be Gamma distributions.

$$x \sim \text{Gamma}(\alpha, \beta)$$
$$\alpha \sim \text{Gamma}(\alpha_\alpha, \beta_\alpha)$$
$$\beta \sim \text{Gamma}(\alpha_\beta, \beta_\beta)$$

The likelihood for the data is

$$f(\mathbf{X}|\alpha, \beta) = \prod_{i=1}^{n} \frac{1}{\Gamma(\alpha)\beta^\alpha} x_i^{\alpha-1} e^{-\frac{x_i}{\beta}}.$$

The prior distributions are

$$\pi(\alpha) = \frac{1}{\Gamma(\alpha_\alpha)\beta_\alpha^{\alpha_\alpha}}\alpha^{\alpha_\alpha - 1}e^{-\frac{\alpha}{\beta_\alpha}}$$
$$\pi(\beta) = \frac{1}{\Gamma(\alpha_\beta)\beta_\beta^{\alpha_\beta}}\beta^{\alpha_\beta - 1}e^{-\frac{\beta}{\beta_\beta}}.$$

Baye's theorem tells us that the posterior distribution is

$$p(\alpha, \beta|\mathbf{X}) = \frac{\prod_{i=1}^{n} f(\mathbf{X}|\alpha, \beta) \cdot \pi(\alpha) \cdot \pi(\beta)}{\int_\Omega \prod_{i=1}^{n} f(\mathbf{X}|\alpha, \beta) \cdot \pi(\alpha) \cdot \pi(\beta)\partial\alpha\partial\beta}$$

which, after some algebraic manipulation, removing of constants, and taking the log, is proportional to

$$p(\alpha, \beta|\mathbf{X}) \propto -n\log(\Gamma(\alpha)) - n\alpha\log(\beta) + (\alpha - 1)\sum_{i=1}^{n}\log(x_i) - \frac{\sum_{i=1}^{n} x_i}{\beta}$$
$$+ (\alpha_\alpha - 1)\log(\alpha) - \frac{\alpha}{\beta_\alpha} + (\alpha_\beta - 1)\log(\beta) - \frac{\beta}{\beta_\beta}$$

The first step is to type your model in a new document window and when saved it needs to be in *.odc format. It is necessary that the user is aware of how WinBUGS parameterizes distributions. In the user manual, it can be seen that WinBUGS parameterizes the gamma distribution with the inverse of the shape parameter. Thus, when defining your model in WinBUGS, it is necessary that you account for differences in parameterizing. One such way is in the following model statement that was saved in *.odc format.

```
model {
for (i in 1:7) {
# likehood
y[i] ~ dgamma(a,b);
}
# prior for a
a ~ dgamma(3, .5);
```

```
# prior for b
b <- 1/c ;
c ~ dgamma(4,.5);
}
```

Next, open another window to display the *.txt format of the data. The first line tells WinBUGS about each column in the dataset while the last line indicates when the program should stop looking for data. This dataset has only one column, which are the responses, y_i.

```
y[]
65.1
42.8
62.7
131.3
57.3
45.8
113.8
END{};
```

Opening a series of windows make up the next steps in the process. Click Model on the upper menu bar and choose Specification..., see figure 3.1, the Specification Tool window will appear. Activate the *.odc window where the model is typed, then click the check model button; look for the message "model is syntactically correct" in the lower left corner of the WinBUGS window. Activate the *.txt window where the data is displayed, then click the load data button; look for the message "data loaded". Click the compile button; look for the message "model compiled". Click the gen inits button and look for the message "initial values generated, model initialized". Click Model on the upper menu bar and choose Update..., see figure 3.2, the Update Tool window will appear with 1000 highlighted in the update field. Click the update button and look for the message "updates

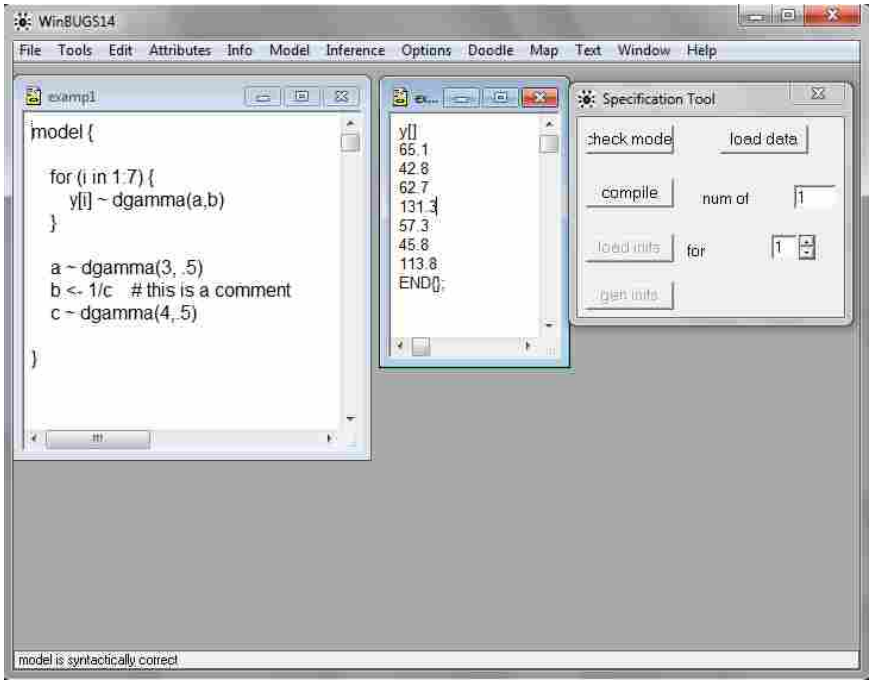


Figure 3.1: Model specification screen shot.

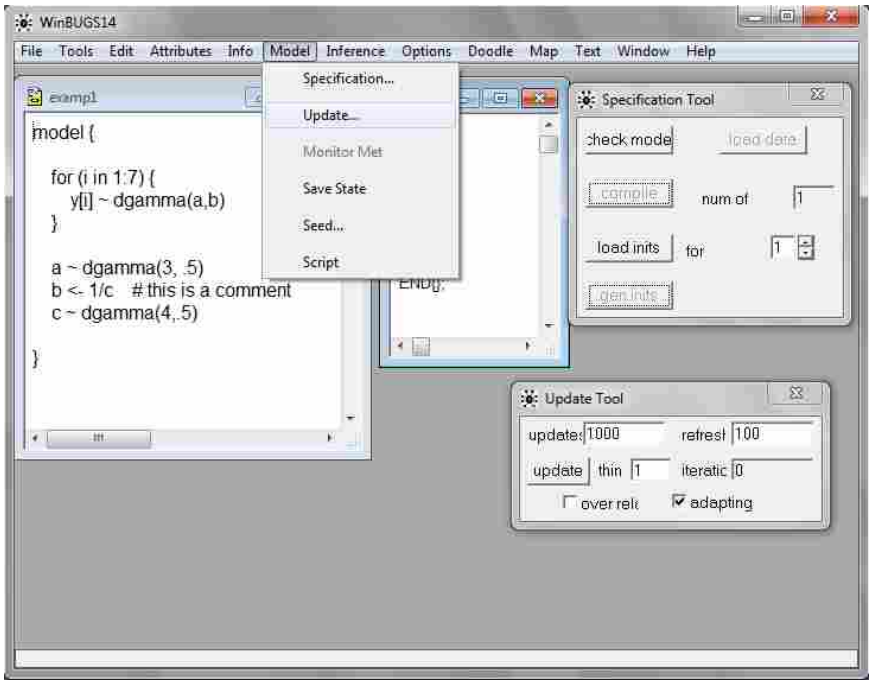


Figure 3.2: Model update screen shot.

took 0 s", which indicates that WinBUGS ran 1000 burn-in iterations through the MCMC algorithm. Click Inference on the upper menu bar and choose Samples..., see figure 3.3, the Sample Monitor Tool window will appear. In the nod field, indicate which variables from the model WinBUGS should keep track of. Type a, then click the set button; type b, then click the set button; type c, then click the set button. After all desired variables have been set, type * which will populate the other buttons in the window, see figure 3.4. Activate the Update Tool window and type the desired number of iterations for the MCMC algorithm, perhaps 10000, in the update field, then click update and look for the "updates took 6 s" message.

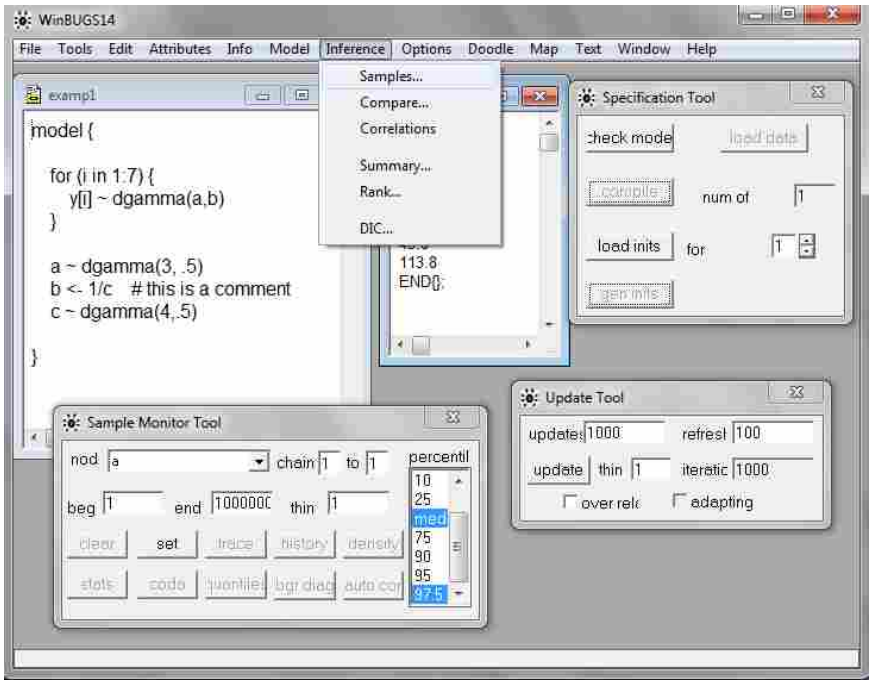


Figure 3.3: Sample monitor screen shot.

At this point, the researcher might want to look at trace plots, density graphs, time series graphs, summary statistics for the variables, or perhaps the draws themselves. These may be accessed from the Sample Monitor Tool window. Trace plots may be viewed by clicking the trace button. Density plots for each variable may be viewed by clicking the density button. Time series graphs may be viewed by clicking the history button. Summary

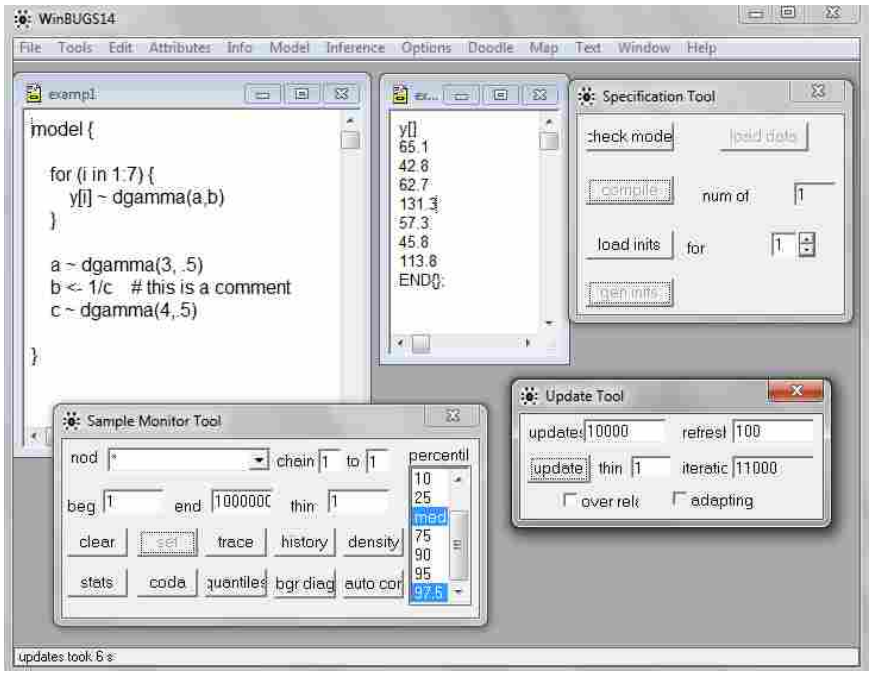


Figure 3.4: Update screen shot.

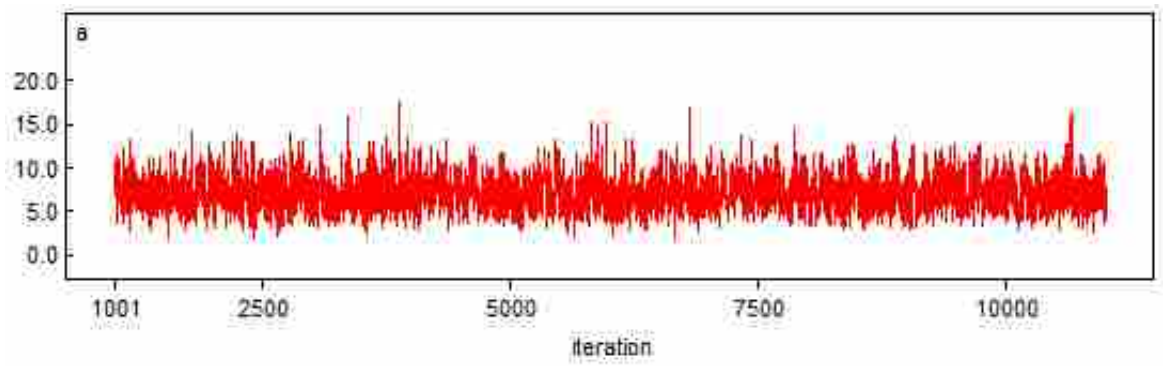
statistics may be viewed by clicking the stats button. The actual draws and an index may be accessed by clicking the coda button.

The summary stats for the analysis of this model are shown below in Table 3.1. As you can see, the estimate for shape=a is 7.09 and the estimate for scale=b is 10.96. These values are reasonably close to the original values of a=6 and b=8 from which the data were simulated.

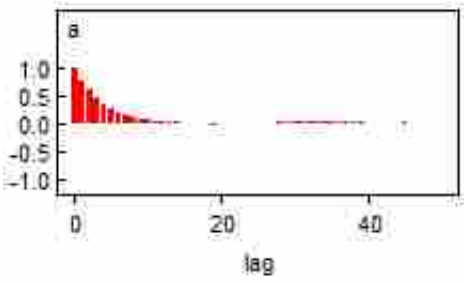
Table 3.1: Summary Statistics for Example 1 from WinBUGS.

	mean	sd	2.5%	25%	50%	75%	97.5%
a	7.09	2.04	3.80	5.63	6.86	8.25	11.76
b	0.10	0.03	0.05	0.08	0.09	0.11	0.16
c	10.96	3.12	6.15	8.70	10.53	12.75	18.36

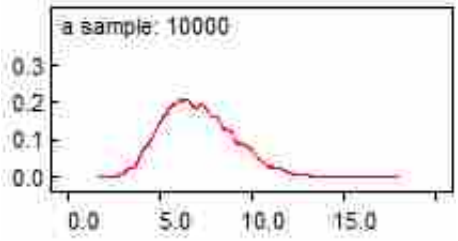
Figure 3.5 shows a sampling of the diagnostic plots that WinBUGS generates.



(a) Trace plot



(b) Autocorrelation



(c) Posterior density

Figure 3.5: Summary plots for α as generated by WinBUGS.

3.2 PROC MCMC

Computer syntax for SAS® 9.2 will be demonstrated by working through the same example that was shown for WinBUGS with the same priors placed on α and β. Below is SAS® 9.2 code demonstrating PROC MCMC with each line assigned a number for the purpose of this discussion. As was stressed earlier for WinBUGS, it is just as crucial that users familiarize themselves with the distributional forms SAS® 9.2 is programed to work with by looking through the user manual for PROC MCMC.

```
* read in the data file;
1 data example1;
2 infile 'c:\example1.txt';
3 input y;
4 run;
```

```
5
* print the data file for inspection;
6 proc print data=example1;
7 run;
8
* turn on graphics device;
9 ods graphics on;
10 proc mcmc data=example1  outpost=examp1out  nmc=10000 nbi=1000 seed=12345;
* set parameters and initial values;
11 parms a 5 b .2;
* define priors;
12 prior a~gamma(3, scale=2);
13 prior b~gamma(4, scale=2);
* likelihood;
14 model y~gamma(a, scale=b);
15 run;
16
* turn off graphics device;
17 ods graphics off;
```

Lines one through four direct SAS® 9.2 to read in the data file and tells SAS® 9.2 what it should find therein. Line one gives a name for SAS® 9.2 to refer to the data. Line two gives the file path where SAS® 9.2 can find the file. Line three tells SAS® 9.2 what variable(s) are located in the datafile and the variable name(s) for the column(s). Line four ends the directions to SAS® 9.2 by indicating to SAS® 9.2 that it should run lines one through four together. Line six directs SAS® 9.2 to print the data in the output window, line seven ends the direction and indicates that SAS® 9.2 should run line six. After running

lines one through six, take a moment and look over the printout of the data to check that it was read correctly by SAS® 9.2 and that you have correctly defined the variable(s).

The PROC MCMC statement is found in lines ten through fifteen. Line nine and seventeen together tell SAS® 9.2 to prepare to produce graphics in the next set of directions and when to stop being ready to produce graphics. Line 10 indicated that SAS® 9.2 should apply the MCMC procedure on the data referred to as example1, to name the posterior output as examp1out, to run 10000 MCMC iterations, to run 1000 burn-in iterations and to set the random seed generator at 12345. Initial values for the parameters are set in line eleven such that a begins at 5 and b begins at .2. Defining the distributional form of the priors is given in lines twelve and thirteen using the SAS® 9.2 definition of the gamma distribution. The model for y is defined in line fourteen using the SAS® 9.2 definition of the gamma distribution. The procedure is concluded in line fifteen.

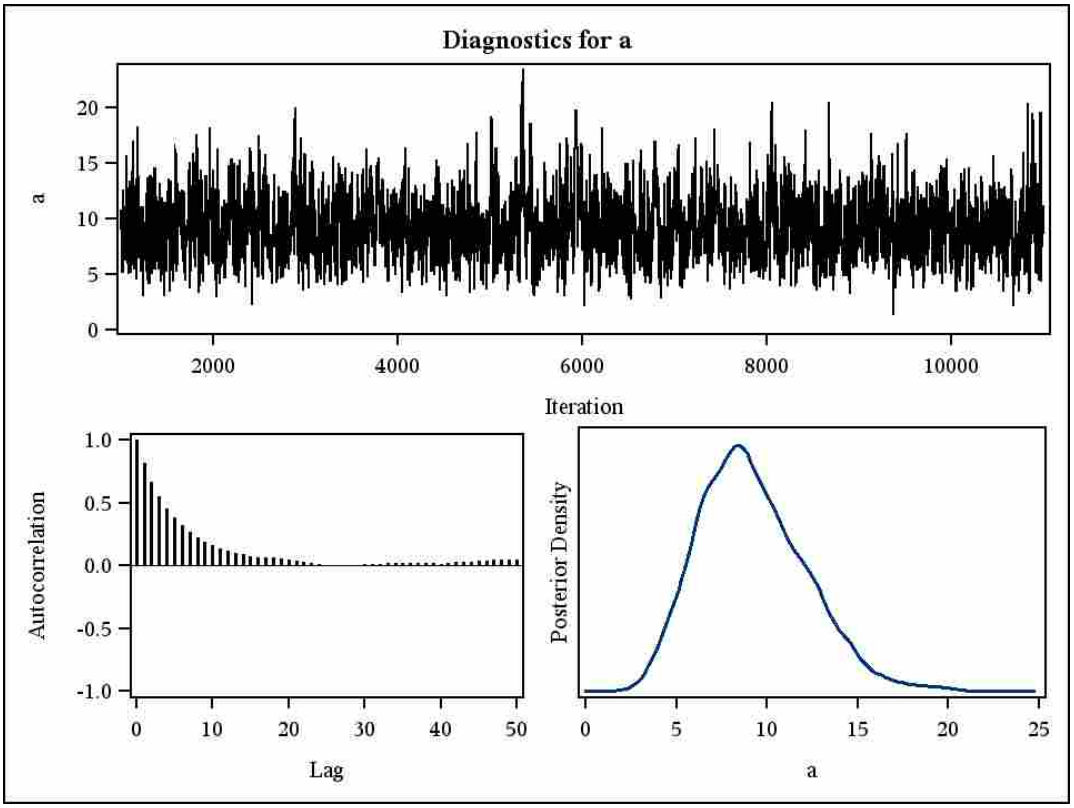
Table 3.2: Summary Statistics for Example 1 from PROC MCMC.

Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
a	10000	9.2144	2.9484	7.0373	8.8827	11.0280
b	10000	0.1324	0.0424	0.1009	0.1282	0.1593

Running lines nine through fifteen will produce several tables of output and diagnostic plots for each of the priors. Figure 3.6 shows a sample of the diagnostic plots that PROC MCMC generates. Of particular interest in the output is the table of posterior summaries as shown in table 3.2. Output tables also include tuning history, posterior intervals, Monte Carlo standard errors, posterior autocorrelations, among others. It should be noted that SAS® 9.2 has two possible parameterizations for the gamma distribution; the inverse scale

was utilized for this example. As such, take the inverse of b in table 3.2 to compare this posterior value with WinBUGS' value of c in table 3.1.

Figure 3.6: Summary plots for α as generated by PROC MCMC.



3.3 Side by Side Computer Code

WinBUGS Code:

```
model {
for (i in 1:7) {
   y[i] ~ dgamma(a,b) ;
   } a ~ dgamma(3, .5) ;
   b <- 1/c;
   c ~ dgamma(4,.5);
    }
```

SAS Code:

```
data example1;
infile 'c:\example1.txt';
input y;
run;

proc print data=example1;
run;

ods graphics on;
proc mcmc data=example1  outpost
  =examp1out
nmc=10000 nbi=1000 seed=12345;
parms a 5 b .2;
prior a~gamma(3, scale=2);
prior b~gamma(4, scale=2);
model y~gamma(a, scale=b);
run;

ods graphics off;
```

3.4 The General WinBUGS Procedure

Here is an outline of the steps to run the model in WinBUGS, provided as a general reference.

1. The model code should be saved in `.odc` format

2. The data code should be saved in `.txt` format

   a) The first row of the data file must be the column names followed by brackets.

   b) The last row of the data file must be `END{}`.

   `y[]`

```
65.1
42.8
62.7
131.3
57.3
45.8
113.8
END{};
```

3. Load the model and data: Model → Specification

   a) Click check model (make sure the `*.odc` window is selected).

   b) Click load data (make sure the `*.txt` window is selected).

   c) Click compile.

   d) Click gen inits.

4. Create burnin: Model → Update...

   a) Enter the number of burnin draws in the updates text field.

      Note: Usually 1,000 should be sufficient.

   b) Click update to create burnin draws.

5. Tell WinBUGS which parameters to keep track of: Inference → Samples...

   a) For each parameter you want to keep track of:

      i. In the node text field, enter the name of the parameter (as it appears in the `*.odc` file).

         Note: When you have entered an acceptable parameter, the set button will turn black.

      ii. Click set.

b) When you have entered all parameters you want to keep track of, enter * in the node text field.

Note: All buttons (except for the set button) will turn black.

6. Calculate the DIC (optional): Inference → DIC...

a) Click set.

7. Create joint posterior draws: Return to Update Tool window

a) Enter the number of posterior draws in the updates text field.

Note: Usually 10,000 should be sufficient.

Note: If you would like to watch the trace plot as the draws are updated, return to the Sample Monitor Tool and click trace.

b) Click update to create posterior draws.

8. View your results: Return to the Sample Monitor Tool

a) Click stats to see statistics for model parameters.

b) Click trace to see parameter trace plots.

c) Click density to see density plots of model parameters.

d) Click auto cor to see monitor autocorrelation within parameters.

9. View DIC results: Return to the DIC Tool

a) Click DIC to see results.

10. Saving the posterior draws: Return to the Sample Monitor Tool

a) Click coda, two windows will pop up.

i. The CODA for chain * window lists the posterior draws for all parameters that were set in Step 5. The draws are listed consecutively by parameter.

The first column is the observation number, and the second column is the posterior draws.

ii. The CODA index window gives the starting and ending index for each parameter in the second column of the CODA for chain * window.

b) File → Save As... (make sure the CODA index window is selected).

CHAPTER 4

TWO SAMPLE T-TEST

Consider the situation where two independent samples from two different normal distributions are obtained. Let x_1 have sample size n_1 and x_2 have sample size n_2; note that $n_1 \neq n_2$.

$$x_1 \sim \text{Normal}(\mu_1, \sigma_1^2)$$
$$x_2 \sim \text{Normal}(\mu_2, \sigma_2^2)$$

A typical Frequentist approach is to assume that the variances are equal and proceed with a two-sample t-test to obtain confidence intervals and test the equality of the two means. In this setting, the distribution of the test statistic under the null hypothesis is known and the methods are reliable. However, a problem with this approach is that the test is very sensitive to the assumption of equal variances and in practice it is almost impossible to satisfy the assumption. This situation is famous in the history of statistics and is referred to as the Behrens-Fisher problem. When the equal variance assumption cannot be satisfied, the distribution of the test statistic is unknown and must be approximated. This approximation is not pleasant and can result in incorrect conclusions because of the sensitivity of the test to the assumption (Casella and Berger 2002).

Within the Bayesian framework this setting poses none of the above problems. We can take a straightforward approach because the posterior distribution of $\mu_1 - \mu_2$ can be estimated while simultaneously taking into account the uncertainties of all parameters involved by treating them as random variables (SAS Institute Inc. 2008).

For this analysis, the data will be modeled as normal and the mean will have a prior distribution of normal while the variance will have a prior distribution of a gamma.

$$\begin{aligned} x_{ij} &\sim \text{Normal}(\mu_i, \sigma_i^2) \\ \mu_i &\sim \text{Normal}(150, \sigma_\mu^2 = 100,000,000) \\ \sigma_i^2 &\sim \text{Gamma}(2, \text{scale} = 25) \end{aligned}$$

The prior values for the mean and variance were selected to preserve the parameter space while not restricting the MCMC process in the random walk.

Equations for the likelihood, prior, and posterior distributions are omitted here where they were provided in Chapter 3 because the MCMC algorithms do not require finding the functional form of the posterior distribution. All that is required is the likelihood function and the distribution for all parameters in the model. The MCMC algorithms calculate the posterior distribution from there.

4.1 WinBUGS

The code below shows the model statement that should be saved as *.odc and the data file that should be saved as a *.txt file. Here the data are modeled as normal with each treatment having its own mean μ and precision τ. In WinBUGS's documentation, it can be seen that the parameterization for the normal distribution involves a precision which is the reciprocal of the variance σ^2. The treatment means are modeled as normal with mean 100 and variance of 100,000,000. The σ_i^2's are modeled with their own gamma distributions because we are not assuming they are equal. An approximate standard deviation for each treatment may be obtained by taking the range of the data and dividing by four. This look at the data tells us that treatment two possibly has a smaller variance, to account for this, σ^2 is allowed to vary for the two treatments. The model statement also instructs WinBUGS to compute the posterior distribution of the difference of means and the ratio of the variances.

The model statement:

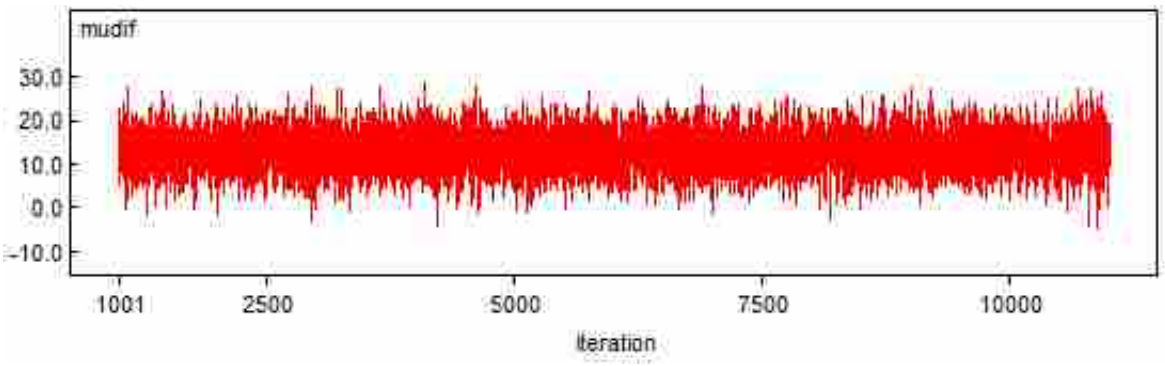
```
model {
# likelihood
for(i in 1:33)
{ y[i] ~ dnorm(mu[tmt[i]], prec[tmt[i]]);
}
for (i in 1:2)
{ prec[i] <- 1/var[i];
# the priors
mu[i] ~ dnorm(100, 0.00000001);
var[i] ~ dgamma(2, 0.04);
}
# variables of interest in the analysis
mudif <- mu[1] - mu[2];
varratio <- var[1]/var[2];
}
```

Table 4.1 shows a sample of the summary statistics for the posterior distribution and indicates that the means of the two samples are different because the distribution of the differences does not include zero. Additionally, the ratio of the two variances would be one if the two variances were the same, but the posterior distribution shown indicates that this ratio is not close to one. These results were obtained without any approximations as would be required in a Frequentist analysis of this same data.

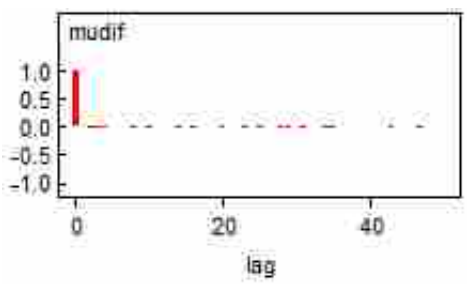
Table 4.1: Summary statistics from WinBUGS.

	mean	sd	2.5%	25%	50%	75%	97.5%
mu[1]	134.61	4.01	126.70	132.00	134.60	137.30	142.40
mu[2]	121.43	1.88	117.70	120.20	121.40	122.70	125.10
mudif	13.18	4.43	4.41	10.22	13.22	16.21	21.81
varratio	6.84	2.66	2.94	4.92	6.42	8.30	13.19
deviance	275.32	4.21	268.80	272.20	274.80	277.80	285.00

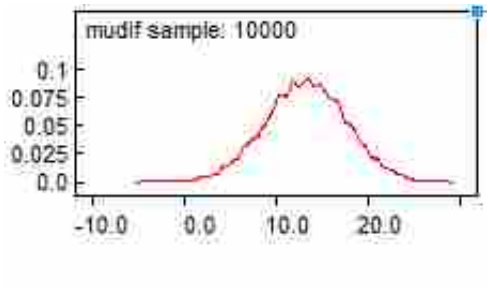
A sample of the posterior summary plots are shown in Figure 4.1. The trace plot indicates that convergence was reached. There were no problems with autocorrelation. The posterior density of the difference of means is also shown.



(a) Trace plot



(b) Autocorrelation



(c) Posterior density

Figure 4.1: WinBUGS summary plots for the posterior distribution of the difference of means.

4.2 PROC MCMC

As was done in chapter 3, each line of code has been numbered for the purpose of this discussion. Please note that line two below needs a directory path for the desired data file to be read into SAS if one is using the following code. Lines one through four direct SAS to read in the data file and tells SAS what it should find therein. Line one gives a name for SAS to refer to the data. Line two gives the file path where SAS can find the file. Line three tells SAS what variable(s) are located in the data file and the variable name(s) for the column(s). Line four ends the directions to SAS by indicating to SAS that it should run lines one through four together. Line six directs SAS to print the data in the output window, line seven ends the direction and indicates that SAS should run line six. After running lines one

through six, take a moment and look over the printout of the data to check that it was read correctly by SAS and that you have correctly defined the variable(s).

The PROC MCMC statement is found in lines ten through twenty-seven. Lines nine and twenty-eight together tell SAS to prepare to produce graphics in the next set of directions and when to stop being ready to produce graphics. Line ten indicated that SAS should apply the MCMC procedure on the data referred to as examp2, to name the posterior output as examp2out, to run 10000 MCMC iterations, to run 1000 burn-in iterations, and to set the random seed generator at 478. Initial values for the parameters are set in lines eleven and thirteen. Line twelve sets an array of length two for σ^2. Defining the distributional form of the priors is given in lines fourteen and fifteen using the SAS definition of the gamma distribution. Line sixteen defines mudif as the difference of the two samples means; line seventeen defines varratio as the ratio of the two population variances. Lines eighteen through twenty-five are for bookkeeping to keep track of which parameters go with which sample. The model for y is defined in line twenty-six using the SAS definition of the normal distribution. The procedure is concluded in line twenty-seven.

```
* read in the data file;
1 data examp2;
2 infile '   ';
3 input tmt y;
4 run;
5
* print the data for inspection;
6 proc print data=examp2;
7 run;
8
* turn on graphics device;
9 ods graphics on;
10 proc mcmc data=ex2 outpost=examp2out nmc=10000 seed=478 nbi=1000
   monitor=(_parms_ mudif varratio) dic;
* set parameters and initial values;
11 parm mu1 0 mu2 0;
* initialize an array of length 2 for sig2;
12 array sig2[2];
* set parameter and initial value;
13 parm sig2: 1;
```

```
* define priors;
14 prior mu: ~ normal(100, var=100000000);
15 prior sig2: ~ gamma(2, scale=25);
* define variables of interest;
16 mudif = mu1 - mu2;
17 varratio = sig2[1]/sig2[2];
* if-then to keep track of group membership;
18 if tmt = 1 then do;
19 mu=mu1;
20 vv=sig2[1];
21 end;
22 else do;
23 mu=mu2;
24 vv=sig2[2];
25 end;
* likelihood;
26 model y ~ normal(mu, var=vv);
27 run;
* turn off graphics device;
28 ods graphics off;
```

Running lines nine through twenty-eight will produce several tables of output and diagnostic plots for each of the priors. Figure 4.2 shows the diagnostic plots that PROC MCMC generates for the difference of means. Of particular interest in the output is the table of posterior summaries as shown in table 4.2. Other output tables are available, including tuning history, posterior intervals, Monte Carlo standard errors, posterior autocorrelations, among others.

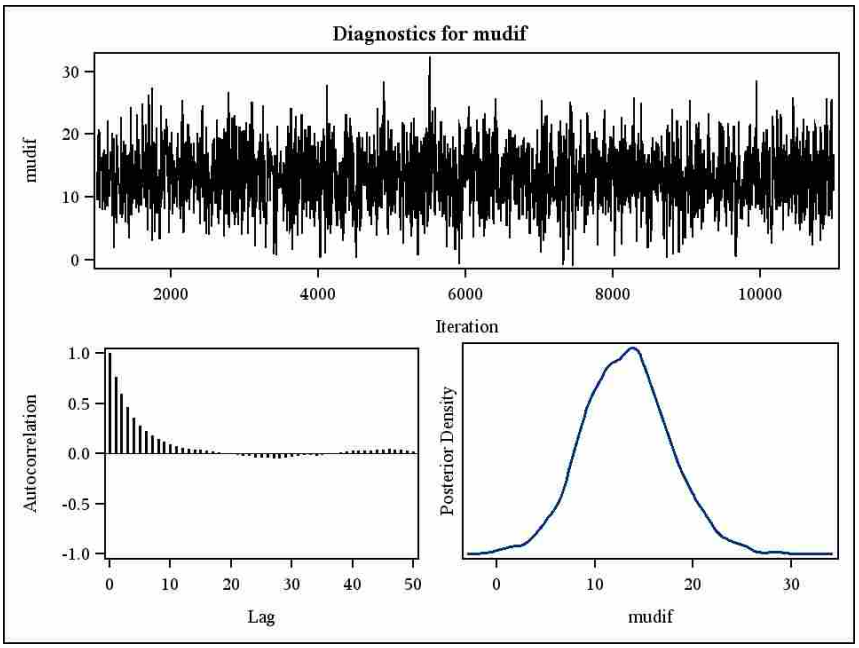
The code produced the following output as shown in Table 4.2 and Figure 4.2. The trace plot indicated that convergence was reached. The autocorrelation graph indicates no problems and the density plot shows the posterior distribution for the difference of the means.

SAS' output leads the researcher to the same conclusion as did WinBUGS that the means of the two samples are different because the distribution of the differences does not include zero. Additionally, the ratio of the two variances also indicates that the two variances are not the same.

Table 4.2: Summary Statistics for Example 2 from PROC MCMC.

Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
mu1	10000	134.6	3.9535	132.0	134.6	137.2
mu2	10000	121.4	1.8769	120.2	121.4	122.6
sig21	10000	298.6	53.8330	259.6	293.4	331.7
sig22	10000	49.7427	17.8083	37.1204	46.4437	58.4252
mudif	10000	13.2301	4.3869	10.1861	13.2190	16.0831
varratio	10000	6.7205	2.5853	4.8349	6.2929	8.1232

Figure 4.2: Summary plot for the posterior distribution of difference of means.



4.3 Side by Side Computer Code

WinBUGS Code:

```
model {
for(i in 1:33)
{ y[i] ~ dnorm(mu[tmt[i]], prec[
   tmt[i]]);
}
for (i in 1:2)
{ prec[i] <- 1/var[i];
mu[i] ~ dnorm(100, 0.00000001);
var[i] ~ dgamma(2, 0.04);
}
mudif <- mu[1] - mu[2];
varratio <- var[1]/var[2];
}
tmt[] y[]
1 121
1 94
1 119
1 122
1 142
1 168
1 116
1 172
1 155
```

SAS Code:

```
data examp2;
infile '   ';
input tmt y;
run;

proc print data=examp2;
run;

ods graphics on;
proc mcmc data=ex2 outpost=
   examp2out nmc=10000 seed=478
   nbi=1000 monitor=(_parms_ mudif
    varratio) dic;
parm mu1 0 mu2 0;
array sig2[2];
parm sig2: 1;
prior mu: ~ normal(100, var
   =100000000);
prior sig2: ~ gamma(2, scale=25);
mudif = mu1 - mu2;
varratio = sig2[1]/sig2[2];
if tmt = 1 then do;
mu=mu1;
```

```
1 107                                   vv=sig2[1];
1 180                                   end;
1 119                                   else do;
1 157                                   mu=mu2;
1 101                                   vv=sig2[2];
1 145                                   end;
1 148                                   model y ~ normal(mu, var=vv);
1 120                                   run;
1 147                                   ods graphics off;
1 125
2 126
2 125
2 130
2 130
2 122
2 118
2 118
2 111
2 123
2 126
2 127
2 111
2 112
2 121
END{};
```

CHAPTER 5

LINEAR REGRESSION

In this chapter, the data set will be used to demonstrate a simple linear regression setting. Here the desire is to understand the functional dependence of one variable on another and the model takes on the form

$$y_i = \beta_0 + \beta_1 x_i,$$

where y_i is the response variable and x_i is an observed variable that predicts y_i.

$$\mathbf{Y} \sim \text{Normal } (\beta_0 + \beta_1 x_i, \sigma^2)$$

The form of the above equation is like unto the slope-intercept line of $y = mx + b$ where β_0 equates to the intercept of the line and β_1 equates to the slope of the line. Hence, $\beta_0 + \beta_1 x_i$ is the mean at the line and σ^2 is the variance of the data around the line.

For this analysis, the data will be modeled as normal and the mean will have a prior for β_0 and for β_1. Three different approaches will be shown in WinBUGS for modeling σ^2, the variance of the data around the line.

$$\begin{aligned} y_i &\sim \text{Normal}(\mu_i, \sigma^2) \\ \mu_i &= \beta_0 + \beta_1 x_i \\ \beta_0 &\sim \text{Normal}(0, 1000000) \\ \beta_1 &\sim \text{Normal}(0, 1000000) \end{aligned}$$

Equations for the likelihood, prior, and posterior distributions are omitted here where they were provided in Chapter 3 because the MCMC algorithms do not require finding the functional form of the posterior distribution. All that is required is the likelihood function

and the distribution for all parameters in the model. The MCMC algorithms calculate the posterior distribution from there.

5.1 WinBUGS

In the code below, the first two lines are included because WinBUGS does not allow the model to exclude any column of the data set. Hence, dd1 and dd2 are needed as dummy variables because these two columns of the data set are not used to model the y_i's.

The model statement defines the y_i's to be normally distributed around the line as defined by μ with precision τ. Recall that WinBUGS parameterizes the normal distribution with precision which is the reciprocal of variance. The β_i's are also set to be normally distributed around zero with a very large variance.

There are three different model statements here to allow for three different approaches to the variance around the line or the modeling of σ^2. The first model sets the variance to be constant for each x_i. The second model defines the precision to be a linear function of the standard deviation. The third model defines the precision to be a linear function of the variance.

The data file is shown below in the Side-By-Side section. These models could be adjusted to replace with y as $y1$ for the response variable or again as $y2$ in the data set to predict the line for these other columns of responses. Note, that the second and third models are using $y2$ as the response variable.

```
model {
  # dummy variables to use all columns in data set
  dd1 <- y1[1];
  dd2 <- y2[1];

  for(i in 1:10){
     # likelihood
     y[i] ~ dnorm(mu[i], prec);
     # define the mean
     mu[i] <- b[1] + b[2]*x[i];
     }
  # the priors for betai
```

```
    b[1] ~ dnorm(0, 0.000001);
    b[2] ~ dnorm(0,0.000001);
    # adjust the variance in terms of precision
    prec <- 1/sig2;
    # prior for variance
    sig2 ~ dgamma(1,0.1);
}

# A second model:
model {
    # dummy variables to use all columns in data set
    dd1 <- y[1];
    dd2 <- y1[1];

    for(i in 1:10){
        # likelihood, allowing variance to change with each xi
        y2[i] ~ dnorm(mu[i], prec[i]);
        # define the mean
        mu[i] <- b[1] + b[2]*x[i];
        cc[i] <- (1/(b[3]*x[i]*sqrt(varreg)));
        # cci relates to the standard deviation of the data
        # the new precision is a linear function of the standard
            deviation
        prec[i] <- cc[i]*cc[i];
        }
    # the priors for betai
    b[1] ~ dnorm(0, 0.0001);
    b[2] ~ dnorm(0,.01);
    b[3] ~ dgamma(1,.2);
    #b[3] is a dgamma because it has to be positive
    #the dgamma has a positive support
    varreg ~ dgamma(2,.2);
}

# A third model:
model {
    # dummy variables to use all columns in data set
    dd1 <- y[1];
    dd2 <- y1[1];

    for(i in 1:10){
        # likelihood, allowing variance to change with each xi
        y2[i] ~ dnorm(mu[i], prec[i]);
        # define the mean
        mu[i] <- b[1] + b[2]*x[i];
        cc[i] <- (1/(b[3]*x[i]*(varreg)));
        #cci relates to the variance of the data
```

```
        # the new precision is a linear function of the variance
        prec[i] <- cc[i]*cc[i];
        }
    # the priors for betai
    b[1] ~ dnorm(0, 0.0001);    b[2] ~ dnorm(0,.01);
    b[3] ~ dgamma(1,.2);
    #b[3] is a dgamma because it has to be positive
    #the dgamma has a positive support
    varreg ~ dgamma(2,.2);
}
```

Table 5.1 shows a sample of the summary statistics for the posterior distribution from the first model and indicates that the intercept of the regression line is about 10 and the slope is 2. A sample of the posterior summary plots is shown in Figure 5.1. The trace plot indicates that convergence was reached. There were no problems with autocorrelation. The posterior density of the variance around the line is also shown.

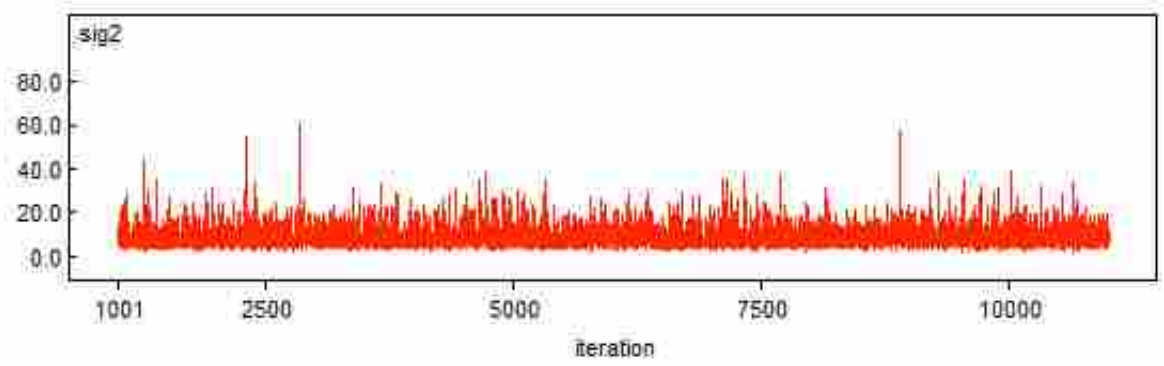
Table 5.1: Summary statistics from WinBUGS.

	mean	sd	2.5%	25%	50%	75%	97.5%
b[1]	9.98	2.77	4.46	8.22	10.01	11.75	15.49
b[2]	2.00	0.05	1.90	1.97	2.00	2.03	2.10
sig2	9.34	4.77	3.63	6.08	8.20	11.31	21.63
deviance	49.53	2.51	46.54	47.65	48.92	50.78	55.82

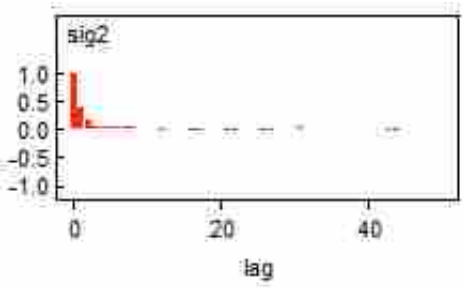
5.2 PROC MCMC

Here in PROC MCMC, only the model with constant variance is shown. Other code could be created to model the behavior of the variance as in the ways shown above in WinBUGS.

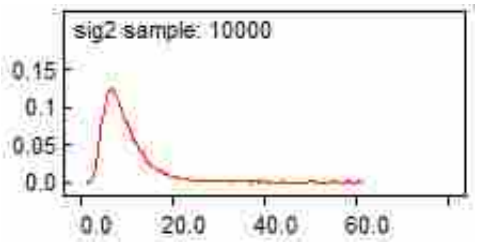
As was done in chapter 3, each line of code has been numbered for the purpose of this discussion. Please note that lines two and twenty-one both have space for the directory path for the file name to be read and the file to be saved. Lines one through four direct SAS to read in the data file and tell SAS what it should find therein. Line one gives a name for SAS to refer to the data. Line two gives the file path where SAS can find the file. Line three tells SAS what variable(s) are located in the data file and the variable name(s) for the



(a) Trace plot



(b) Autocorrelation



(c) Posterior density

Figure 5.1: WinBUGS summary plots for the posterior distribution of the variance around the line.

column(s). Line four ends the directions to SAS by indicating to SAS that it should run lines one through four together.

The PROC MCMC statement is found in lines seven through sixteen. Lines six and seventeen together tell SAS to prepare to produce graphics in the next set of directions and when to stop being ready to produce graphics. Line seven indicates that SAS should apply the MCMC procedure on the data referred to as ex3, to name the posterior postex3, to run 510000 MCMC iterations, to have 10000 burn-in iterations, and to thin the draws by taking only every fiftieth one. Line eight continues the MCMC procedure by setting the random seed generator at 123 and asking SAS to monitor β_0, β_1, and σ^2. Initial values for the parameters are set in lines nine and ten. Lines eleven through thirteen define the distributional form of the priors using the SAS definition of distributions. Line fourteen defines μ and line fifteen defines the model. The procedure is concluded in line sixteen.

Lines nineteen through twenty-four ask SAS to export the posterior draws to a .csv file that can be read into another program for further analysis.

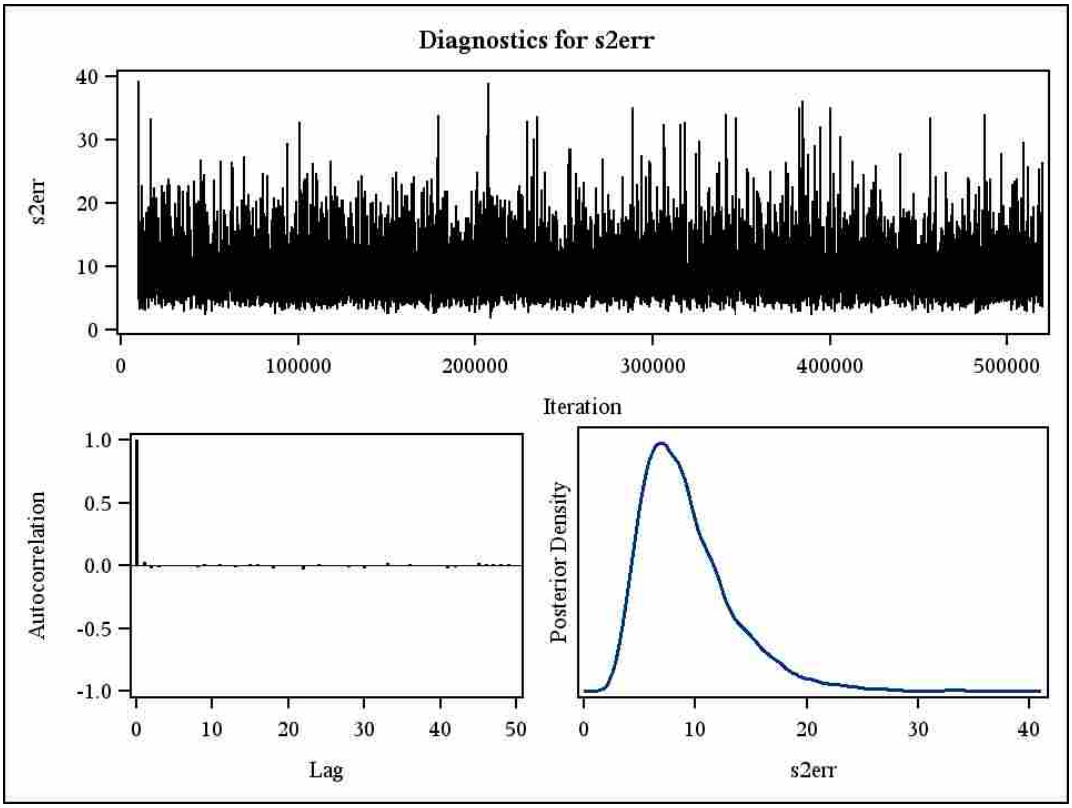
The code produced the following output as shown in Table 5.2 and Figure 5.2. Notice that SAS predicts the intercept of the line to be about 10 and the slope to be about 2. The trace plot indicated that convergence was reached. The autocorrelation graph indicates no problem and the density plot shows the posterior distribution for σ^2.

Table 5.2: Summary Statistics for Example 3 from PROC MCMC.

Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
b0	10200	9.9615	2.8027	8.1657	9.9702	11.7410
b1	10200	2.0005	0.0529	1.9667	2.0008	2.0345
s2err	10200	9.2820	4.1895	6.3273	8.4157	11.2738

```
* read in the data file;
1  data ex3;
2      infile '    ';
3      input x y y1 y2;
4  run;
5
* turn on graphics device;
6  ods graphics on;
7  proc mcmc data=ex3 outpost=postex3 nmc=510000 nbi=10000 thin=50
8    seed=123 monitor=(b0 b1 s2err);
* set parameters and initial values;
9       parms b0 0 b1 1;
10     parms s2err 10;
* define priors;
11     prior b0 ~ normal(0,var=10000);
12     prior b1 ~ normal(0, var=100);
13     prior s2err ~ gamma(2, scale=5);
* define the mean, which is the line;
14     mu=b0 + b1*x;
* likelihood;
15     model y ~ normal(mu, var=s2err);
```

Figure 5.2: Summary plot for the posterior distribution of the variance around the line.




```
16  run;
* turn off graphics device;
17  ods graphics off;
18
19  /* creating the chain of draws: */
20  proc export data=postex3
21     outfile='     '
22     dbms=csv
23     replace;
24 run;
```

5.3 Side by Side Computer Code

WinBUGS Code:

```
model {
    dd1 <- y1[1];
    dd2 <- y2[1];

        for(i in 1:10){
        y[i] ~ dnorm(mu[i], prec);
        mu[i] <- b[1] + b[2]*x[i];
        }

    b[1] ~ dnorm(0, 0.000001);
     #a very small
    #precision gets a very big
       variance
    b[2] ~ dnorm(0,0.000001);
    prec <- 1/sig2;
    sig2 ~ dgamma(1,0.1);
}

# A second model:
model {
    dd1 <- y[1];
    dd2 <- y1[1];
```

SAS Code:

```
    infile '    ';
    input x y y1 y2;
run;

ods graphics on;
proc mcmc data=ex3 outpost=
   postex3 nmc=510000 nbi=10000
   thin=50
     seed=123 monitor=(b0 b1 s2err
        );
     parms b0 0 b1 1;
     parms s2err 10;
     prior b0 ~ normal(0,var
        =10000);
     prior b1 ~ normal(0, var=100)
        ;
     prior s2err ~ gamma(2, scale
        =5);
     mu=b0 + b1*x;
     model y ~ normal(mu, var=
        s2err);
run;
ods graphics off;
```

```
for(i in 1:10){
   y2[i] ~ dnorm(mu[i], prec[i
      ]);
   #allowing variance to
      change with each xi
   mu[i] <- b[1] + b[2]*x[i];
   cc[i] <- (1/(b[3]*x[i]*sqrt
      (varreg)));
   #cci relates to the
      standard deviation of the
       data
   # the new precision is a
      linear function of the
      standard deviation
   prec[i] <- cc[i]*cc[i];
   }

b[1] ~ dnorm(0, 0.0001);
 #a very small
#precision gets a very big
   variance
b[2] ~ dnorm(0,.01);
b[3] ~ dgamma(1,.2);
#b[3] is a dgamma because it
   has to be positive
#the dgamma has a positive
   support
```

```
/* creating the chain of draws:
   */
proc export data=postex3
     outfile='     '
     dbms=csv
     replace;
run;
```

The data file:

```
x[] y[] y1[] y2[]
30 73 41 88
20 50 82 53
60 128 131 140
80 170 157 204
40 87 87 92
50 108 88 96
60 135 130 114
30 69 59 81
70 148 160 200
60 132 168 161
END{};
```

```
    varreg ~ dgamma(2,.2);
}

# A third model:
model {
    dd1 <- y[1];
    dd2 <- y1[1];

    for(i in 1:10){
        y2[i] ~ dnorm(mu[i], prec[i
            ]);
        #allowing variance to
            change with each xi
        mu[i] <- b[1] + b[2]*x[i];
        cc[i] <- (1/(b[3]*x[i]*(
            varreg)));
        #cci relates to the
            standard deviation of the
             data
        # the new precision is a
            linear function of the
            standard deviation
        prec[i] <- cc[i]*cc[i];
        }

    b[1] ~ dnorm(0, 0.0001);
     #a very small
```

```
    #precision gets a very big
       variance
    b[2] ~ dnorm(0,.01);
    b[3] ~ dgamma(1,.2);
    #b[3] is a dgamma because it
       has to be positive
    #the dgamma has a positive
       support
    #prec <- 1/varreg;
    varreg ~ dgamma(2,.2);
}
```

CHAPTER 6

MULTIPLE REGRESSION

In the previous chapter, the desire was to relate a single dependent response variable, y to a single independent predictor variable x. This chapter expands this approach to include multiple independent predictor variables x_m which model the behavior of a single dependent response variable y. For each y_i in the data set, there are m columns of the x_m's that will be used for prediction. The model takes on the form

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_m x_m$$

with

$$Y \sim \text{Normal}(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_j x_m, \sigma^2).$$

The form of the above equation is like unto the slope-intercept line of $y = mx + b$ where the intercept is β_0 and the idea of slope is expanded to include the sum of the remaining β_j that are coefficients on the x_m's. Hence, $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_j x_m$ is the mean at the line and σ^2 is the variance of the data around the line.

For this analysis, the data will be modeled as normal and the mean will have a prior for each of the β_i's. Three different models will show three approaches for modeling the mean; the first model is given below.

$$
\begin{aligned}
y_i &\sim \text{Normal}(\mu_i, \sigma^2) \\
\mu_i &= \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} \\
\beta_0 &\sim \text{Normal}(0, 10000) \\
\beta_1 &\sim \text{Normal})(0, 100) \\
\beta_2 &\sim \text{Normal}(0, 100) \\
\sigma^2 &\sim \text{Gamma}(3, \text{scale} = 10)
\end{aligned}
$$

Equations for the likelihood, prior, and posterior distributions are omitted here where they were provided in Chapter 3 because the MCMC algorithms do not require finding the functional form of the posterior distribution. All that is required is the likelihood function and the distribution for all parameters in the model. The MCMC algorithms calculate the posterior distribution from there.

Here in this chapter the concept of model selection will be introduced by discussing DIC which means Deviance Information Criterion. Three different models will be looked at and evaluated for their goodness of fit as calculated by DIC = -2log(likelihood) + (the effective number of parameters). The model with the smallest DIC has the best fit. For further information about DIC, see the WinBUGS user manual, the SAS User's Guide, or Carlin and Louis (2009).

6.1 WinBUGS

This data set has three columns and is shown in the side-by-side code section below. The objective is to predict the hours as a function of number of interviews and number of miles driven. The first model in the code is the full model using all three columns of the data set such that hours = β_0 + β_1*miles + β_2*interviews. The second model removes the interviews column and just models hours = β_0 + β_1*miles. The third model removes the miles column

and just models hours = $\beta_0 + \beta_2$*interviews. For all three models, WinBUGS will calculate DIC and this value will be used to determine which model has a better goodness of fit.

```
 model {
    for (i in 1:14) {
           # likelihood
            hours[i] ~ dnorm(mu[i],prec);
            # define the mean
      mu[i] <- b[1] + b[2]*miles[i] + b[3]*ints[i];
      }
      # the priors for betai
      b[1] ~ dnorm(0, 0.0001);
      b[2] ~ dnorm(0,.01);
      b[3] ~ dnorm(0,.01);
      # adjust the variance in terms of precision
      prec <- 1/varreg;
      # prior for variance
      varreg ~ dgamma(3,.1);
      }

# A second model:
model {
    # dummy variable to use all columns in data set
      dummy1 <- ints[1];
    for (i in 1:14) {
           # likelihood
            hours[i] ~ dnorm(mu[i],prec);
            # define the mean
      mu[i] <- b[1] + b[2]*miles[i];
      }
      # the priors for betai
      b[1] ~ dnorm(0, 0.0001);
      b[2] ~ dnorm(0,.01);
      #b[3] ~ dnorm(0,.01);
      # adjust the variance in terms of precision
      prec <- 1/varreg;
      # prior for variance
      varreg ~ dgamma(3,.1);
      }

# A third model:
model {
   # dummy variable to use all columns in data set
     dummy1 <- miles[1];
    for (i in 1:14) {
           # likelihood
            hours[i] ~ dnorm(mu[i],prec);
```

```
        # define the mean
    mu[i] <- b[1]  + b[3]*ints[i];
  }
  # the priors for betai
  b[1] ~ dnorm(0, 0.0001);
  #b[2] ~ dnorm(0,.01);
  b[3] ~ dnorm(0,.01);
  # adjust the variance in terms of precision
  prec <- 1/varreg;
  # prior for variance
  varreg ~ dgamma(3,.1);
}
```

Table 6.1 shows a sample of the summary statistics for the posterior distribution of the third model. Figure 6.1 gives a sample of the posterior summary plots for β_0 in the third model. Convergence was reached, there were no problems with autocorrelation, and the posterior distribution is shown.

Table 6.1: Summary statistics for all three models from WinBUGS.

	mean	sd	2.5%	25%	50%	75%	97.5%
b[1]	0.77	4.30	-7.76	-2.08	0.77	3.62	9.29
b[2]	0.69	0.51	-0.32	0.36	0.69	1.03	1.70
b[3]	1.82	1.11	-0.40	1.09	1.81	2.55	4.04
varreg	35.61	11.96	18.13	26.96	33.67	42.01	64.23
deviance	90.20	2.62	86.87	88.26	89.62	91.55	96.70

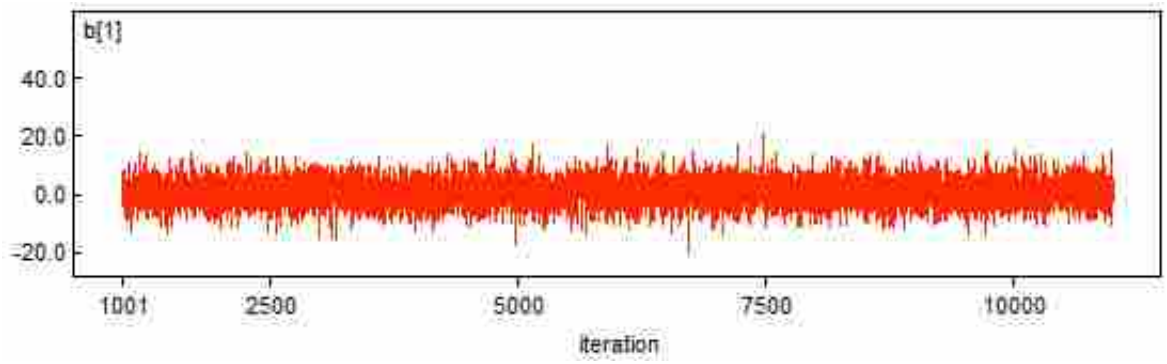
	mean	sd	2.5%	25%	50%	75%	97.5%
b[1]	2.57	4.32	-6.19	-0.28	2.60	5.40	11.20
b[2]	1.49	0.15	1.19	1.39	1.49	1.59	1.79
varreg	38.78	12.29	20.68	29.92	36.81	45.40	68.49
deviance	92.04	2.18	89.63	90.48	91.45	93.01	97.74

	mean	sd	2.5%	25%	50%	75%	97.5%
b[1]	0.77	4.37	-7.99	-2.14	0.78	3.65	9.33
b[3]	3.27	0.32	2.63	3.06	3.27	3.48	3.90
varreg	37.15	12.09	19.41	28.51	35.12	43.76	66.35
deviance	91.10	2.19	88.64	89.52	90.51	92.07	96.74

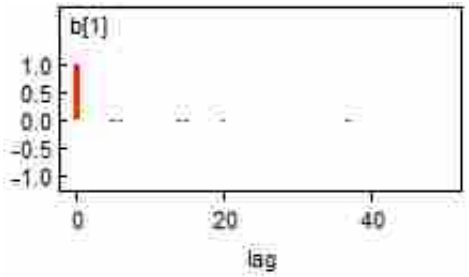
DIC for model 1: 93.643

DIC for model 2: 94.583

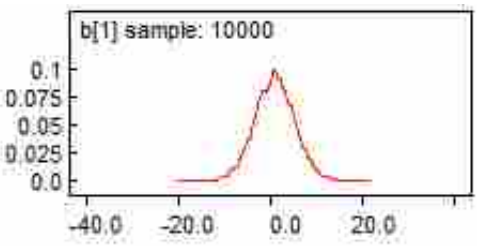
DIC for model 3: 93.641



(a) Trace plot



(b) Autocorrelation



(c) Posterior density

Figure 6.1: WinBUGS summary plots for the posterior distribution of the regression line's intercept from model three.

These DIC values give a measure of model fit and allow the researcher to compare how different models perform their ability to model the data satisfactorily. The lower DIC value indicates which model fits the data best. Here, model 3 has the lowest DIC, so this model is the best choice of the three.

6.2 PROC MCMC

The code for the first and third models are presented here and the summary statistics are shown in table 6.2. The second model was not included because it had the highest DIC

and it is very similar to the code for model 3. Notice on lines nine and thirty-seven that nmc has been increased along with nbi and we added an option to thin the draws every 50. These changes can be made when the posterior distribution has a little trouble reaching convergence and shows problems with autocorrelation. These chosen values for the number of MCMC iterations and number of burn-in values have allowed the posterior here to reach convergence satisfactorily and have no autocorrelation issues as seen in figure 6.2.

As was done in chapter 3, each line of code has been numbered for the purpose of this discussion. Please note that lines two, twenty-three, thirty, and fifty-one have space for the directory path for the file name to be read and the file to be saved. Lines one through four and twenty-nine through thirty-two direct SAS to read in the data file and tell SAS what it should find therein. Lines one and twenty-nine give a name for SAS to refer to the data. Lines two and thirty give the file path where SAS can find the file. Lines three and thirty-one tell SAS what variables are located in the data file and the variable names for the columns. Lines four and thirty-two end the directions to SAS by indicating to SAS that it should run lines one through four and lines twenty-seven through thirty-two together respectively.

The PROC MCMC statement is found in lines nine through twenty-one and again in lines thirty-seven through forty-nine. Initial values for the parameters are set in lines eleven through fourteen and thirty-nine through forty-two. The distributional forms of the priors are set in lines fifteen through eighteen and forty-three through forty-six. Lines nineteen and forty-seven define μ and lines twenty and forty-eight define the model. The procedures are concluded in lines twenty-one and forty-nine. Lines twenty-three through twenty-four and fifty-one through fifty-two ask SAS to export the posterior draws to a .csv file that can be read into another program for further analysis.

The code produced the following output as shown in Table 6.2 and Figure 6.2.

```
 * read in the data file;
1  data no4;
2  infile ''     '';
3  input hours ints miles;
4  run;
```

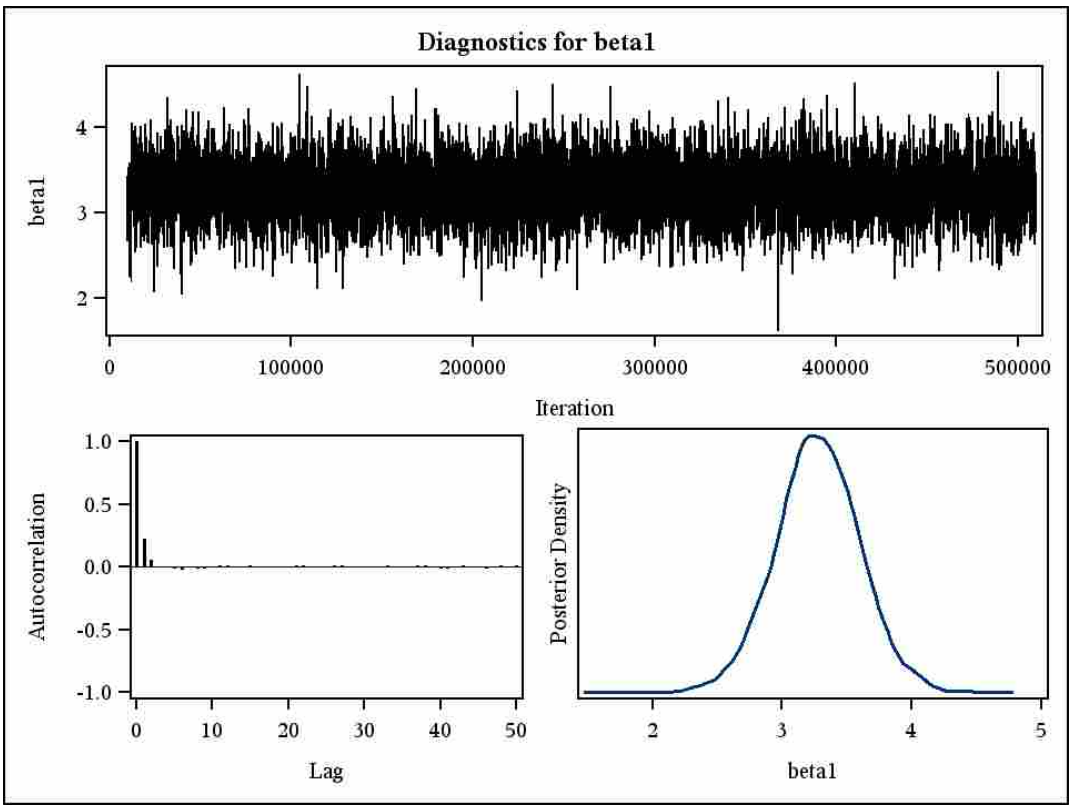
Table 6.2: Summary Statistics for Example 4 from PROC MCMC. Posterior summaries for the first and third models are shown.

Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
beta0	10000	0.8569	4.3578	-1.9874	0.7965	3.7143
beta1	10000	1.8286	1.1145	1.0678	1.8230	2.5532
beta2	10000	0.6851	0.5080	0.3558	0.6841	1.0243
s2err	10000	35.5924	11.7582	27.1990	33.6351	41.8653

Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
beta0	10000	0.7521	4.3466	-2.1088	0.7501	3.5427
beta1	10000	3.2755	0.3219	3.0649	3.2754	3.4917
s2err	10000	37.0810	11.9183	28.4832	35.1127	43.3988

```
5
* print the data file for inspection;
6  proc print data=no4;
7  run;
8
9  proc mcmc  data=no4 outpost=no4post nmc=500000 nbi=10000 thin=50
   seed=1234
10       monitor=(_parms_)  dic;
* set parameters and initial values;
11  parms beta0 0;
12  parms beta1 0;
13  parms beta2 0;
14  parms s2err 10;
* define priors;
15  prior beta0 ~ normal(0,prec=.0001);
16  prior beta1 ~ normal(0,prec=.01);
17  prior beta2 ~ normal(0, prec=.01);
18  prior s2err ~ gamma(3, iscale=.1);
* define the mean, which is the line;
19  mu = beta0 + beta1*ints + beta2*miles;
```

Figure 6.2: Summary plots for the posterior distribution of the regression line's intercept from model three.




```
* likelihood;
20  model hours ~ normal(mu, var=s2err);
21  run;
22
* export the posterior MCMC draws and save the .csv file;
23  proc export data=no4post outfile='' '' dbms=csv replace;
24  run;
25
26  /*  The third model:  */
27
28
* read in the data file;
29  data no4;
30  infile ''  '';
31  input hours ints miles;
32  run;
33
```

```
* print the data file for inspection;
34  proc print data=no4;
35  run;
36
37  proc mcmc  data=no4 outpost=no4post nmc=500000 nbi=10000 thin=50
   seed=1234
38       monitor=(_parms_)  dic;
* set parameters and initial values;
39  parms beta0 0;
40  parms beta1 0;
41  *parms beta2 0;
42  parms s2err 10;
* define priors;
43  prior beta0 ~ normal(0,prec=.0001);
44  prior beta1 ~ normal(0,prec=.01);
45  *prior beta2 ~ normal(0, prec=.01);
46  prior s2err ~ gamma(3, iscale=.1);
* define the mean;
47  mu = beta0 + beta1*ints;
* likelihood;
48  model hours ~ normal(mu, var=s2err);
49  run;
50
* export the posterior MCMC draws and save the .csv file;
51  proc export data=no4post outfile=''     '' dbms=csv replace;
52  run;
```

6.3 Side by Side Computer Code

WinBUGS Code:

```
model {
   for (i in 1:14) {
         hours[i] ~ dnorm(mu[i
            ],prec);
      mu[i] <- b[1] + b[2]*
         miles[i] + b[3]*ints[i
         ];
   }
```

SAS Code:

```
data no4;
infile ''     '';
input hours ints miles;
run;

proc print data=no4;
run;
```

```
      b[1] ~ dnorm(0, 0.0001);
      b[2] ~ dnorm(0,.01);
      b[3] ~ dnorm(0,.01);
      prec <- 1/varreg;
      varreg ~ dgamma(3,.1);
      }

# A second model:
model {
      dummy1 <- ints[1];
    for (i in 1:14) {
          hours[i] ~ dnorm(mu[i
            ],prec);
        mu[i] <- b[1] + b[2]*
           miles[i];
      }
      b[1] ~ dnorm(0, 0.0001);
      b[2] ~ dnorm(0,.01);
      #b[3] ~ dnorm(0,.01);
      prec <- 1/varreg;
      varreg ~ dgamma(3,.1);
      }

# A third model:
model {
     dummy1 <- miles[1];
    for (i in 1:14) {
```

```
proc mcmc  data=no4 outpost=
   no4post nmc=500000 nbi=10000
   thin=50 seed=1234
    monitor=(_parms_)  dic;
parms beta0 0;
parms beta1 0;
parms beta2 0;
parms s2err 10;
prior beta0 ~ normal(0,prec
   =.0001);
prior beta1 ~ normal(0,prec=.01);
prior beta2 ~ normal(0, prec=.01)
   ;
prior s2err ~ gamma(3, iscale=.1)
   ;
mu = beta0 + beta1*ints + beta2*
   miles;
model hours ~ normal(mu, var=
   s2err);
run;

proc export data=no4post outfile
   =''    '' dbms=csv replace;
run;

/*   The third model:  */
data no4;
```

```
        hours[i] ~ dnorm(mu[i
            ],prec);
        mu[i] <- b[1] + b[3]*
            ints[i];
    }
    b[1] ~ dnorm(0, 0.0001);
    #b[2] ~ dnorm(0,.01);
    b[3] ~ dnorm(0,.01);
    prec <- 1/varreg;
    varreg ~ dgamma(3,.1);
}
```

```
# The data set:
hours[] ints[] miles[]
52.1   17    35.7
24.6    6    11.4
49.2   13    28.6
30.0   11    25.8
82.2    23    50.6
42.4   16    27.2
55.7   15    31.3
21.1    5     10
27.7    10   18.9
36.3    12    25.2
69.1    20    39.9
38.8    12    32.5
```

```
infile ''     '';
input hours ints miles;
run;

proc print data=no4;
run;

proc mcmc  data=no4 outpost=
   no4post nmc=500000 nbi=10000
   thin=50 seed=1234
    monitor=(_parms_)  dic;
parms beta0 0;
parms beta1 0;
*parms beta2 0;
parms s2err 10;
prior beta0 ~ normal(0,prec
   =.0001);
prior beta1 ~ normal(0,prec=.01);
*prior beta2 ~ normal(0, prec
   =.01);
prior s2err ~ gamma(3, iscale=.1)
   ;
mu = beta0 + beta1*ints;
model hours ~ normal(mu, var=
   s2err);
run;
```

```
22.8    8    13.6
34.7    8    19
END{};
```

```
proc export data=no4post outfile
   =''     '' dbms=csv replace;
run;
```

CHAPTER 7

ONE-WAY ANOVA

A one-way analysis of variance (ANOVA) is a factorial design with one treatment at multiple levels. Of interest are the sample means and whether or not there are differences among them. ANOVA is a way to test the null hypotheses that samples from two or more treatment groups are drawn from the same population under the assumption that the variances of the populations are equal. Under the Bayesian paradigm we can test two models, one where the variances are allowed to be different for each group and the other where the variance is the same for all groups. The performance of these models will be compared using DIC to determine which model fits better.

For this analysis, the data will be modeled as normal with a prior for each treatment's μ and a prior for σ^2. Model one will allow each treatment its own σ^2 while model two will only model a single σ^2 for all treatments. Here is the outline for model one, model two's outline is very similar.

$$\begin{aligned} y_i &\sim \text{Normal}(\mu_j, \sigma_j^2) \\ \mu_j &\sim \text{Normal}(20, 10000) \\ \sigma_j^2 &\sim \text{Gamma}(2, \text{scale} = 25) \end{aligned}$$

Equations for the likelihood, prior, and posterior distributions are omitted here where they were provided in Chapter 3 because the MCMC algorithms do not require finding the functional form of the posterior distribution. All that is required is the likelihood function and the distribution for all parameters in the model. The MCMC algorithms calculate the posterior distribution from there.

7.1 WinBUGS

This data set contains responses from four treatment groups and the objective is to determine if the responses among these four treatments are the same. The first model in the code allows the variances for each group to be different. The second model in the code treats the variances for the groups as equal.

```
model{
   # dummy variable to use all columns of data set
   dummy1 <- tmt[1];

   for(i in 1:28){
   # likelihood
   y[i] ~ dnorm(mu[trt[i]], prec[trt[i]]);
   }

   for (i in 1:4){
   # the priors
   mu[i] ~ dnorm(20, 0.0001);
   s2[i] ~ dgamma(2,0.04);
   # adjust the variance in terms of precision
   prec[i] <- 1/s2[i];
   }
}

# Second model restricts sigma2 to be same for all groups:
model{
   # dummy variable to use all columns of data set
   dummy1 <- tmt[1];

   for(i in 1:28){
   # likelihood
   y[i] ~ dnorm(mu[trt[i]], prec);
   }

   for (i in 1:4){
   # the priors
   mu[i] ~ dnorm(20, 0.0001);
   }
   s2 ~ dgamma(2,0.04);
   # adjust the variance in terms of precision
   prec <- 1/s2;
}
```

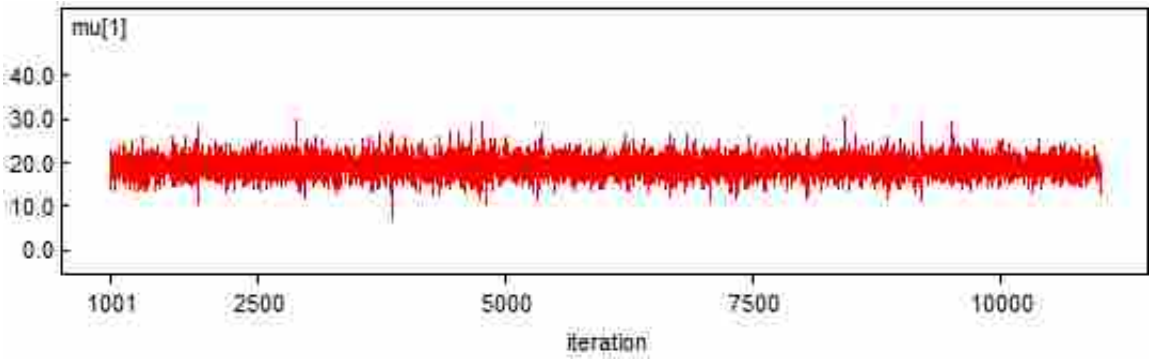
Table 7.1 shows the summary statistics for the posterior distribution of these two models. As you can see, the first model gives values for each samples' variance while the second model has only one variance. Figure 7.1 gives a sample of the posterior summary plots for group one's mean response from the first model. Convergence was reached, there were no problems with autocorrelation, and the posterior distribution is shown. However, model 2 is the better fitting model because DIC is lower here which indicates that it is appropriate to assume the variance for these four groups to be the same. Further analysis of the posterior distributions may be done to determine if the mean responses are the same among the four groups.

Please note the use of a dummy variable in these two models. WinBUGS requires that every column in the data set be utilized in some way in the model code. However, sometimes, a column of observations is not needed as part of the analysis. One option is to remove this column from the data set. Another option is to assign the unused column to a dummy variable as we have done here with "dummy1".

Table 7.1: Summary statistics for both models from WinBUGS.

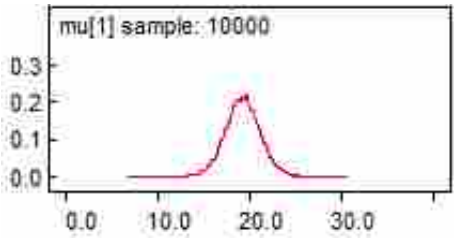
	mean	sd	2.5%	25%	50%	75%	97.5%
mu[1]	19.19	2.01	15.22	17.92	19.18	20.43	23.27
mu[2]	25.80	2.42	20.93	24.30	25.82	27.30	30.71
mu[3]	11.46	2.67	6.12	9.79	11.48	13.10	16.75
mu[4]	15.26	2.47	10.35	13.69	15.23	16.83	20.16
s2[1]	28.97	17.41	8.79	16.82	24.46	36.31	74.58
s2[2]	41.48	21.89	14.75	26.18	36.59	50.82	99.60
s2[3]	49.68	24.22	18.61	32.48	44.11	60.79	111.50
s2[4]	42.20	21.52	14.99	27.12	37.19	51.82	97.86
deviance	176.19	3.86	170.20	173.30	175.70	178.40	184.90

	mean	sd	2.5%	25%	50%	75%	97.5%
mu[1]	19.14	2.18	14.83	17.70	19.13	20.58	23.46
mu[2]	25.79	2.15	21.49	24.38	25.75	27.19	30.08
mu[3]	11.43	2.18	7.18	10.02	11.43	12.89	15.74
mu[4]	15.26	2.18	10.91	13.82	15.24	16.70	19.59
s2	32.90	10.02	18.72	25.82	31.16	38.04	57.28
deviance	174.81	3.56	170.10	172.20	174.10	176.70	183.60



(a) Trace plot

(b) Autocorrelation



(c) Posterior density

Figure 7.1: WinBUGS summary plots for the posterior distribution of group one's mean from model one.

DIC for model 1: 181.2

DIC for model 2: 179.4

7.2 PROC MCMC

The code for both models is given. Of note is line fifteen and sixteen from our first model where we define an array for μ and σ^2. Since we will be working with four group means and four group variances, we need to define an array of length four for both variables. Additionally, lines seventeen and eighteen have a colon after the variable names to indicate that the starting value should be applied to all array entries. Similarly, lines nineteen and twenty have a colon after the variable names to indicate that the distributions should be applied to all array entries. In the model statement on line twenty-one, we are telling SAS

that the "trt" column indicates how the mean and variance are grouped with the response values.

The second model defines only an array for μ, line thirty-three, because we are working with a single σ^2 for each group here. Notice that the likelihood statement on line thirty-eight allows only μ to vary by group while σ^2 is held constant. Lines thirty-four and thirty-five set the parameters and give their initial values. Lines thirty-six and thirty-seven define the prior distributions for μ and σ^2.

As an aside, lines ten, eleven, and twenty-four along with lines twenty-eight, twenty-nine and forty-two ask SAS to save the graphics and output tables as a *.pdf file. Line ten or twenty-eight begins this command, line eleven or twenty-nine gives the file path, and line twenty-five or forty-two ends the command. This is a nice tool to have in one's SAS toolbox.

```
* read in the data file;
1  data no5;
2  infile "  ";
3  input tmt y trt;
4  run;
5
* print the data file for inspection;
6  proc print data=no5;
7  run;
8
9  /* Model 1 */
* initializes saving of output as a pdf file;
10  ods pdf
11     file="   ";
* turn on graphics device;
12  ods graphics on;
13  proc mcmc data=no5 outpost=no5post nmc=100000 nbi=1000 thin=10 seed
   =1234
14  monitor=(_parms_) dic;
* define arrays of length 4;
15       array mu[4];
16       array s2[4];
* set parameters and initial values;
17       parms mu: 0;
18       parms s2: 1;
* define priors;
19       prior mu: ~normal(20, prec=0.0001);
20       prior s2: ~gamma(2, iscale=0.04);
```

```
* likelihood;
21       model y ~ normal(mu[trt], var=s2[trt]);
22   run;
23
* turn off graphics device;
24   ods graphics off;
* stops saving output file;
25   ods pdf close;
26
27   /* Model 2 */
* initializes saving of output as a pdf file;
28   ods pdf
29       file="     ";
* turn on graphics device;
30   ods graphics on;
31   proc mcmc data=no5 outpost=no5post nmc=100000 nbi=1000 thin=10 seed
   =1234
32   monitor=(_parms_) dic;
* initialize array of length 4;
33       array mu[4];
* set parameters and initial values;
34       parms mu: 0;
35       parms s2 1;
* define priors;
36       prior mu: ~normal(20, prec=0.0001);
37       prior s2 ~gamma(2, iscale=0.04);
* likelihood;
38       model y ~ normal(mu[trt], var=s2);
39   run;
40
* turn off graphics device;
41   ods graphics off;
* stop saving output file;
42   ods pdf close;
```

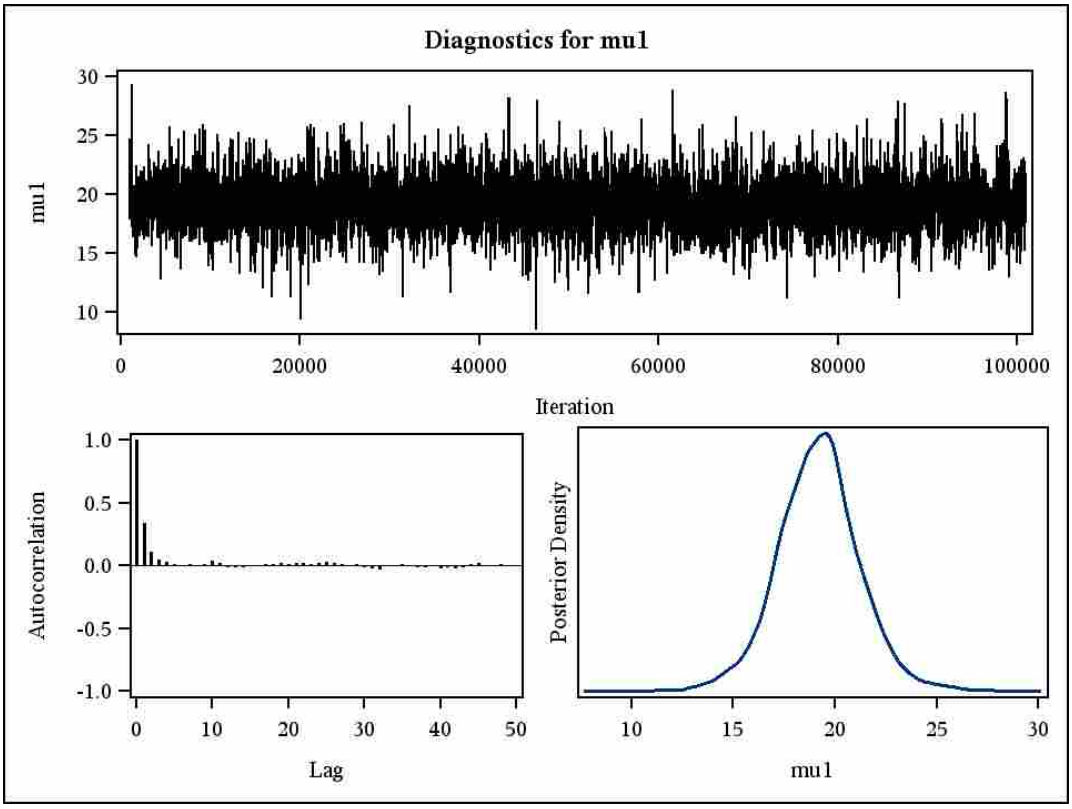
Table 7.2 shows the summary statistics for both models. In comparison with the summary statistics from WinBUGS, the posterior values are very close for all of the variables in both models. Figure 7.2 gives the posterior plots for group one's mean response from the first model. As was found in WinBUGS, model 2 with a single variance fits better than model 1, DIC for model 1 is 181.035 and DIC for model 2 is 179.473. Recall that with DIC, the smaller value indicates a better fitting model.

Table 7.2: Summary Statistics for Example 5 from PROC MCMC, both models are shown.

Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
mu1	10000	19.2038	2.0267	17.9202	19.2192	20.4216
mu2	10000	25.7793	2.4627	24.2456	25.8094	27.3148
mu3	10000	11.4125	2.5924	9.7964	11.3877	13.0384
mu4	10000	15.3112	2.4385	13.7688	15.2712	16.8677
s21	10000	28.4665	16.9253	16.7966	23.9472	35.4976
s22	10000	41.4464	22.1698	26.1303	36.3526	50.7291
s23	10000	48.4412	22.7821	32.3070	43.6142	59.3121
s24	10000	42.6589	21.6666	27.2798	37.7725	52.7968

Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
mu1	10000	19.2049	2.1820	17.7772	19.1837	20.6437
mu2	10000	25.6874	2.1629	24.2729	25.6894	27.1613
mu3	10000	11.4488	2.1625	10.0221	11.4538	12.8734
mu4	10000	15.2360	2.1896	13.8050	15.2139	16.7090
s2	10000	32.9959	9.8810	25.9839	31.4092	38.1260

Figure 7.2: Summary plots for the posterior distribution group one's mean response from model 1.



7.3 Side by Side Computer Code

WinBUGS Code:

```
model{
    dummy1 <- tmt[1];

    for(i in 1:28){
    y[i] ~ dnorm(mu[trt[i]], prec[
```

SAS Code:

```
/* Model 1 */
ods pdf
     file=''     '';
ods graphics on;
proc mcmc data=no5 outpost=
```

```
    trt[i]]);
  }

  for (i in 1:4){
  mu[i] ~ dnorm(20, 0.0001);
  s2[i] ~ dgamma(2,0.04);
  prec[i] <- 1/s2[i];
  }
}

# Second model:
model{
  dummy1 <- tmt[1];

  for(i in 1:28){
  y[i] ~ dnorm(mu[trt[i]], prec)
    ;
  }

  for (i in 1:4){
  mu[i] ~ dnorm(20, 0.0001);
  }
  s2 ~ dgamma(2,0.04);
  prec <- 1/s2;
}
```

```
  no5post nmc=100000 nbi=1000
  thin=10 seed=1234 monitor=(
  _parms_) dic;
    array mu[4];
    array s2[4];
    parms mu: 0;
    parms s2: 1;
    prior mu: ~normal(20, prec
      =0.0001);
    prior s2: ~gamma(2, iscale
      =0.04);
    model y ~ normal(mu[trt], var
      =s2[trt]);
run;

ods graphics off;
ods pdf close;

/* Model 2 */
ods pdf
    file=''  '';
ods graphics on;
proc mcmc data=no5 outpost=
  no5post nmc=100000 nbi=1000
  thin=10 seed=1234 monitor=(
  _parms_) dic;
    array mu[4];
```

```
# The data set:
  tmt[]    y[]    trt[]
    1 13.225551    1
    1 16.381329    1
    1 23.350272    1
    1 24.639941    1
    1 19.277215    1
    1 19.905660    1
    1 17.503872    1
    2 31.442618    2
    2 25.883180    2
    2 20.322800    2
    2 23.431178    2
    2 24.424148    2
    2 34.437840    2
    2 20.375937    2
    3 18.949881    3
    3  3.731357    3
    3 16.063343    3
    3 16.670093    3
    3  6.878069    3
    3  4.329623    3
    3 13.505717    3
    4  8.880663    4
    4  8.762337    4
    4 20.413043    4
    4 17.576046    4
```

```
        parms mu: 0;
        parms s2 1;
        prior mu: ~normal(20, prec
          =0.0001);
        prior s2 ~gamma(2, iscale
          =0.04);
        model y ~ normal(mu[trt], var
          =s2);
    run;

    ods graphics off;
    ods pdf close;
```

```
4  16.054745    4
4  12.249461    4
4  22.734480    4
END{};
```

CHAPTER 8

FACTORIAL DESIGN

A factorial design is used when the objective is to understand the effect of two or more independent treatment variables upon a single dependent response variable. These are also called two-way ANOVA. Of interest is determining the optimal combination of variables to give the most desired response. It is often helpful to see the data arranged in a table as shown in table 8.1. Participants will be assigned to one of the sixteen treatment combinations and their responses will be recorded. The most challenging part of this analysis is the bookkeeping to keep track of which response values should be grouped with which $\boldsymbol{\mu}$ values because there are sixteen of them.

Table 8.1: Arrangement of a 4x4 factorial design experiment.

		Treatment B Levels			
		1	2	3	4
Treatment A Levels	1	$\mu_{1,1}$	$\mu_{1,2}$	$\mu_{1,3}$	$\mu_{1,4}$
	2	$\mu_{2,1}$	$\mu_{2,2}$	$\mu_{2,3}$	$\mu_{2,4}$
	3	$\mu_{3,1}$	$\mu_{3,2}$	$\mu_{3,3}$	$\mu_{3,4}$
	4	$\mu_{4,1}$	$\mu_{4,2}$	$\mu_{4,3}$	$\mu_{4,4}$

For this analysis, the data will be modeled as normal with a prior for each treatment combination's μ. The data will be modeled with a single variance, σ^2.

$$y_i \sim \text{Normal}(\mu_{[A,B]}, \sigma^2)$$

$$\mu_{[A,B]} \sim \text{Normal}(1, 10000)$$

$$\sigma^2 \sim \text{Gamma}(2, \text{scale}{=}0.5)$$

Equations for the likelihood, prior, and posterior distributions are omitted here where they were provided in Chapter 3 because the MCMC algorithms do not require finding the

functional form of the posterior distribution. All that is required is the likelihood function and the distribution for all parameters in the model. The MCMC algorithms calculate the posterior distribution from there.

An advantage of the Bayesian approach to this analysis is that posterior draws are obtained and available to compute any linear combination of the cell means without worrying about multiple test adjustments.

8.1 WinBUGS

This data set contains responses of thirty-three participants in a feeding trial where they were tested for their response to two treatments that each have four possible levels as shown in table 8.1. We have sixteen different treatment combinations, (4x4), to consider as we try to determine the optimal treatment combination.

```
model{
   # dummy variable to use all columns of data set
   dummy1 <- tmt[1];
   for (i in 1:33){
   # likelihood
   gain[i] ~ dnorm(mu[tmtA[i],tmtB[i]],prec);
   }
   # defining the 16 priors for mu
   for (i in 1:4){
   for(j in 1:4) {
   mu[i,j] ~ dnorm(1,0.0001);
   }

   }
   # prior for sigma2 and adjusting variance in terms of precision
   s2 ~ dgamma(2,2);
   prec <- 1/s2;
   }
```

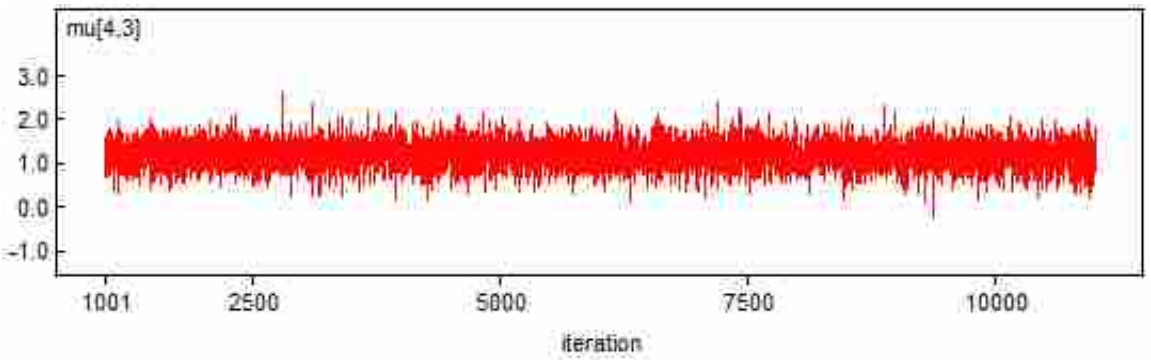
The summary statistics for the posterior distribution are given in table 8.2. As you can see, there is quite a range of mean responses among the different cells. Figure 8.1 gives a sample of the posterior summary plots, showing the posterior distribution of treatment A

at the fourth level and treatment B at the third level. Convergence was reached, there were no problems with autocorrelation, and the posterior distribution is shown.

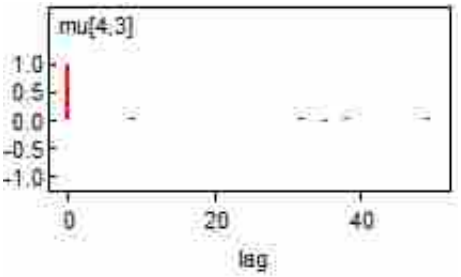
A researcher could save the posterior draws and read them into another program to conduct further analysis to determine the optimal treatment combination. The posterior draws could be used to find the marginal means for both treatments along with confidence intervals. Density plots of the posterior distributions for these marginal means could be created along with posterior distributions for each of the sixteen cell means from the posterior draws. Contrast statements would indicate that there is an interaction term to account for which, after further analysis, would lead to the conclusion that $\mu_{[4,3]}$ is the optimal treatment combination.

Table 8.2: Summary statistics from WinBUGS.

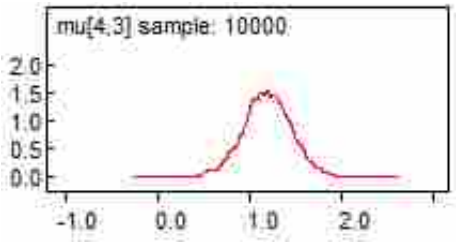
	mean	sd	2.5%	25%	50%	75%	97.5%
mu[1,1]	1.11	0.27	0.58	0.94	1.11	1.29	1.64
mu[1,2]	0.88	0.22	0.46	0.74	0.88	1.03	1.33
mu[1,3]	1.14	0.38	0.38	0.90	1.14	1.38	1.89
mu[1,4]	1.10	0.27	0.56	0.92	1.10	1.27	1.62
mu[2,1]	0.77	0.27	0.23	0.59	0.76	0.94	1.30
mu[2,2]	1.30	0.22	0.86	1.16	1.30	1.44	1.73
mu[2,3]	1.01	0.27	0.48	0.83	1.00	1.18	1.54
mu[2,4]	1.58	0.27	1.04	1.41	1.58	1.75	2.11
mu[3,1]	0.79	0.27	0.25	0.61	0.79	0.96	1.32
mu[3,2]	1.02	0.22	0.58	0.88	1.02	1.17	1.46
mu[3,3]	1.79	0.27	1.27	1.62	1.79	1.96	2.33
mu[3,4]	0.97	0.27	0.45	0.80	0.97	1.15	1.50
mu[4,1]	1.38	0.38	0.63	1.13	1.38	1.63	2.12
mu[4,2]	1.25	0.27	0.72	1.07	1.25	1.42	1.78
mu[4,3]	1.20	0.27	0.64	1.03	1.20	1.37	1.74
mu[4,4]	1.45	0.27	0.92	1.28	1.45	1.63	1.99
s2	0.14	0.06	0.07	0.10	0.13	0.17	0.29
deviance	24.12	9.43	8.41	17.44	23.10	29.91	45.26



(a) Trace plot



(b) Autocorrelation



(c) Posterior density

Figure 8.1: WinBUGS summary plots for the posterior distribution of treatment A at the fourth level and treatment B at the third level.

8.2 PROC MCMC

In SAS, when one of the variables is categorical, this fact needs to be indicated as done in line three with the $ signs after tmtA and tmtB. Notice that line fourteen defines an array of length sixteen for μ, and in line nineteen's model, μ is grouped by the sixteen treatment combinations while the entire dataset is modeled with a single σ^2. Lines fifteen and seventeen have a colon after μ to indicate that the initial value and the prior distribution should be applied to each of the sixteen array entries.

```
* read in the data file;
1  data no5;
2  infile ''  '';
3  input gain tmtA $ tmtB $ tmt;
4  run;
5
* print the data file for inspection;
```

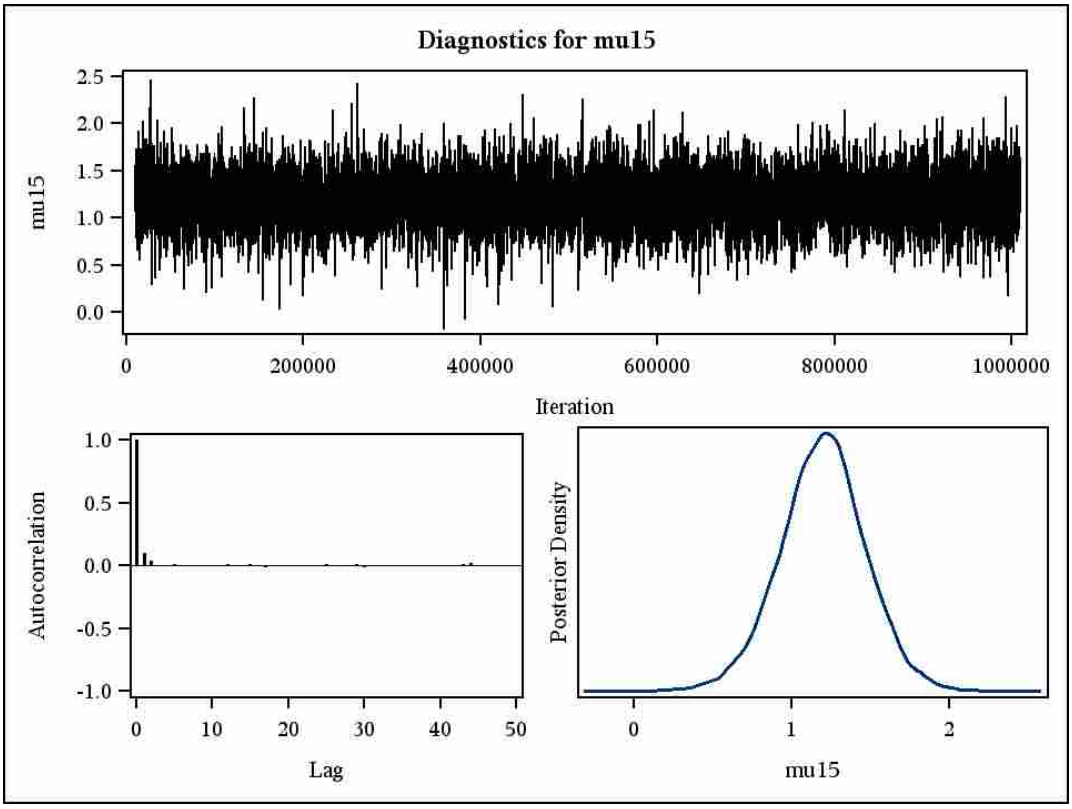
```
6  proc print data=no5;
7  run;
8
* initializes saving of output as a pdf file;
9  ods pdf
10  file ``    '';
* turn on graphics device;
11  ods graphics on;
12  proc mcmc data=no5 outpost=no5post nmc=1000000 nbi=10000 seed=4826
   thin=100
13       monitor=(_parms_) dic;
* create an array of length 16 for mu;
14       array mu[16];
* set parameters and initial values;
* the colon on mu indicated that the initial value be applied to all
   array entries;
15       parms mu: 0;
16       parms s2 1;
* define priors;
* the colon on mu indicates that the prior be applied to all array
   entries;
17       prior mu: ~normal(1,prec=.0001);
18       prior s2~gamma(2,iscale=2);
* likelihood;
19       model gain~normal(mu[tmt], var=s2);
20  run;
21
* export the posterior MCMC draws and save the .csv file;
22  proc export data=no5post outfile=``    '' dbms=csv replace;
23  run;
24
* turn off graphics device;
25  ods graphics off;
* stop saving output file;
26  ods pdf close;
```

Table 8.3 shows the summary statistics for this analysis. In comparison with the summary from WinBUGS, it can be seen that both programs are providing similar posterior summaries for all seventeen variables. Figure 8.2 gives the posterior plots which indicate convergence was reached and there were no problems with autocorrelation.

Table 8.3: Summary Statistics for Example 6 from PROC MCMC.

Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
mu1	10000	1.1194	0.2665	0.9460	1.1170	1.2905
mu2	10000	0.8819	0.2211	0.7412	0.8820	1.0223
mu3	10000	1.1505	0.3806	0.9081	1.1460	1.3935
mu4	10000	1.0915	0.2678	0.9240	1.0938	1.2645
mu5	10000	0.7624	0.2703	0.5902	0.7637	0.9352
mu6	10000	1.2974	0.2266	1.1514	1.2954	1.4431
mu7	10000	1.0034	0.2740	0.8286	1.0038	1.1757
mu8	10000	1.5842	0.2689	1.4092	1.5837	1.7569
mu9	10000	0.7876	0.2700	0.6150	0.7900	0.9603
mu10	10000	1.0229	0.2201	0.8809	1.0244	1.1668
mu11	10000	1.7966	0.2718	1.6188	1.7961	1.9758
mu12	10000	0.9745	0.2735	0.8004	0.9758	1.1473
mu13	10000	1.3839	0.3846	1.1367	1.3842	1.6313
mu14	10000	1.2429	0.2710	1.0697	1.2431	1.4160
mu15	10000	1.2000	0.2726	1.0292	1.2046	1.3728
mu16	10000	1.4537	0.2728	1.2803	1.4548	1.6310
s2	10000	0.1464	0.0634	0.1033	0.1321	0.1734

Figure 8.2: Summary plots for the posterior distribution of treatment A at the fourth level and treatment B at the third level.



8.3 Side by Side Computer Code

WinBUGS Code:

```
model{
dummy1 <- tmt[1];
for (i in 1:33){
gain[i] ~ dnorm(mu[tmtA[i],tmtB[i
   ]],prec);
```

SAS Code:

```
data no5;
infile 'z:\my documents\
   stat595R_bayesian\hmwk\hmwk_5\
   hmwk5.txt';
input gain tmtA $ tmtB $ tmt;
```

```
}
for (i in 1:4){
for(j in 1:4) {
mu[i,j] ~ dnorm(1,0.0001);
}
}
s2 ~ dgamma(2,2);
prec <- 1/s2;
}

# The data set:
gain[]  tmtA[]  tmtB[]  tmt[]
1.06601955230029 1 1 1
1.1717077129235 1 1 1
1.13299485302514 1 2 2
0.725519010925458 1 2 2
0.790168559937335 1 2 2
1.14275707613406 1 3 3
1.26275163486356 1 4 4
0.924987112824956 1 4 4
0.635748752652977 2 1 5
0.89040183526832 2 1 5
1.40850641235961 2 2 6
1.05334779462587 2 2 6
1.42454603147834 2 2 6
0.876869178935861 2 3 7
1.13417236833904 2 3 7
```

```
run;

proc print data=no5;
run;

ods pdf
file = 'z:\my documents\
   stat595r_bayesian\hmwk\hmwk_5\
   SASoutput.pdf';
ods graphics on;
proc mcmc data=no5 outpost=
   no5post nmc=1000000 nbi=10000
   seed=4826 thin=100
     monitor=(_parms_) dic;
     array mu[16];
     parms mu: 0;
     parms s2 1;
     prior mu: ~normal(1,prec
        =.0001);
     prior s2~gamma(2,iscale=2);
     model gain~normal(mu[tmt],
        var=s2);
run;

proc export data=no5post outfile
   ='z:\my documents\
   stat595r_bayesian\hmwk\hmwk_5\
```

```
1.5307166136822 2 4 8
1.62876422747221 2 4 8
0.91869954925839 3 1 9
0.651574142026445 3 1 9
0.964795917174517 3 2 10
1.13567909147059 3 2 10
0.971842851689181 3 2 10
1.86789905425025 3 3 11
1.71858834375702 3 3 11
0.828461308629336 3 4 12
1.12405391192776 3 4 12
1.37895018583156 4 1 13
0.79139772956555 4 2 14
1.70889873968731 4 2 14
0.585982054786358 4 3 15
1.81194876837391 4 3 15
1.21187262044929 4 4 16
1.69330934499011 4 4 16
END{};
```

```
    no5post.csv' dbms=csv replace;
run;

ods graphics off;
ods pdf close;
```

CHAPTER 9

ANALYSIS OF COVARIANCE

Analysis of Covariance, ANCOVA, combines one-way or two-way ANOVA with linear regression. ANCOVA is used when there is a continuous response variable and two or more predictor variables, one being categorical and one being continuous. Here, we are comparing one variable in two or more groups taking into account variability of other variables, called covariates. Since ANCOVA is a method based on linear regression, the relationship of the dependent variable to the independent variable(s) must be linear in the parameters. Figure 9.1 shows that the relationship in our data set is indeed linear. In fact it might be possible that each group has its own y-intercept and slope. Two models will be presented that will explore these possibilities. The first model allows for two intercepts and two slopes while the second model allows for two intercepts but restricts the slopes to be the same. The two models will be compared via DIC values to determine which fits the data better. Here is an outline of the first model.

$$
\begin{aligned}
y_i &\sim \text{Normal}(\mu_j, \sigma^2) \\
\mu_j &= \beta_{0j} + \beta_{1j} x_i \\
\beta_{0j} &\sim \text{Normal}(10, 10000) \\
\beta_{1j} &\sim \text{Normal}(0, 100) \\
\sigma^2 &\sim \text{Gamma}(7, scale = 25)
\end{aligned}
$$

Equations for the likelihood, prior, and posterior distributions are omitted here where they were provided in Chapter 3 because the MCMC algorithms do not require finding the functional form of the posterior distribution. All that is required is the likelihood function

and the distribution for all parameters in the model. The MCMC algorithms calculate the posterior distribution from there.

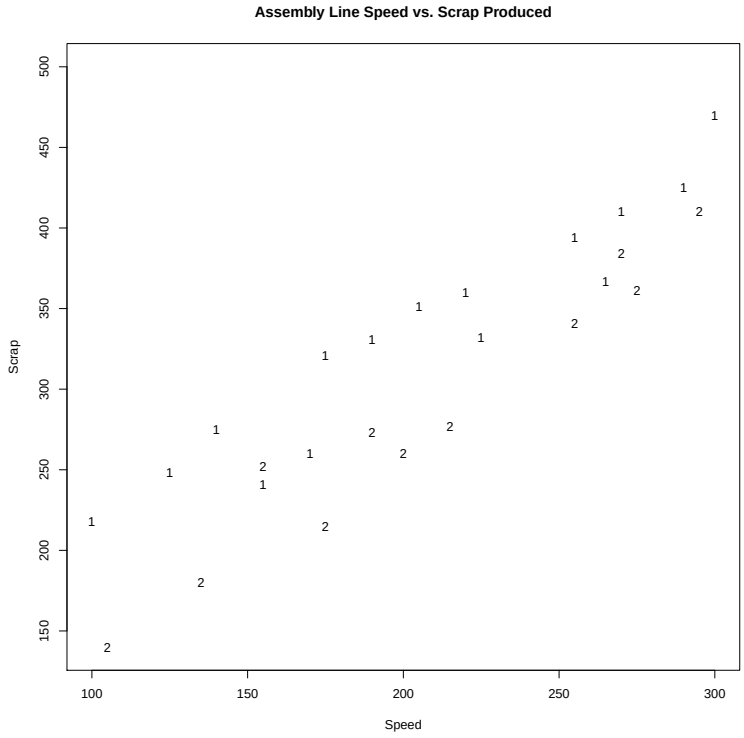


Figure 9.1: Linear relationship between variables.

9.1 WinBUGS

This analysis will look at production data from two different production lines with the objective of determining how much scrap is produced as the speed of an assembly line increases. We have three variables in the data set, line number (1 or 2), line speed, and amount of scrap produced. The data are grouped by line number and the response variable is the amount of scrap with a covariate of line speed.

```
# model one has two intercepts and two slopes
model{
    for (i in 1:27){
       # likelihood
       scrap[i] ~ dnorm(mu[i],prec);
       # define the mean
```

```
        mu[i] <- b[line[i]] + b1[line[i]]*speed[i];
        }
          # the priors for betai
        b[1] ~ dnorm(10, 0.0001);
        b[2] ~ dnorm(10, 0.0001);
        b1[1] ~ dnorm(0, 0.01);
        b1[2] ~ dnorm(0, 0.01);
        # prior for variance and adjust it in terms of precision
        s2 ~ dgamma(7, 0.04);
        prec <- 1/s2;
    }

# model two has two intercepts and one slope
model{
    for (i in 1:27){
        # likelihood
        scrap[i] ~ dnorm(mu[i],prec);
        # define the mean
        mu[i] <- b[line[i]] + b1[1]*speed[i]
        }
          # the priors for betai
        b[1] ~ dnorm(10, 0.0001);
        b[2] ~ dnorm(10, 0.0001);
        b1[1] ~ dnorm(0, 0.01);
        # prior for variance and adjust it in terms of precision
        s2 ~ dgamma(5, 0.01);
        prec <- 1/s2;
    }
```

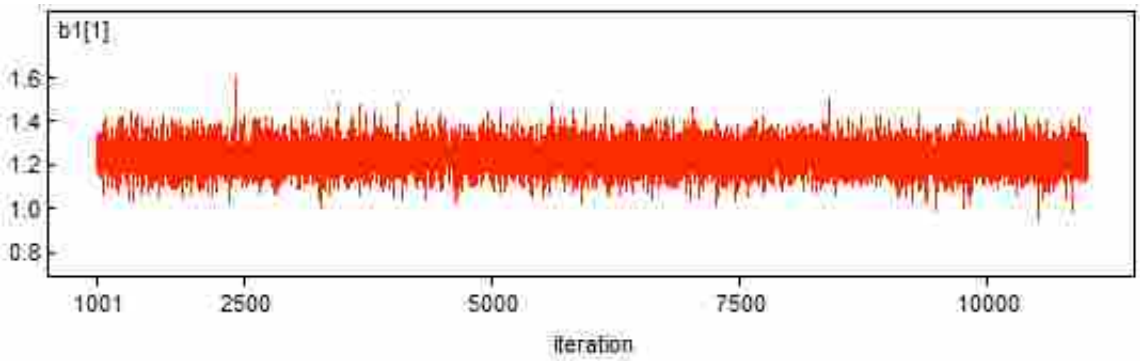
Table 9.1: Summary statistics from WinBUGS for both models.

	mean	sd	2.5%	25%	50%	75%	97.5%
b[1]	94.11	20.81	52.60	80.37	94.45	108.10	134.50
b[2]	7.82	22.68	-36.68	-7.06	7.78	22.81	52.41
b1[1]	1.16	0.10	0.97	1.10	1.16	1.23	1.36
b1[2]	1.32	0.10	1.12	1.25	1.32	1.39	1.52
s2	534.02	137.45	321.19	435.07	514.60	614.62	855.40
deviance	242.03	3.45	237.10	239.50	241.40	244.00	250.30

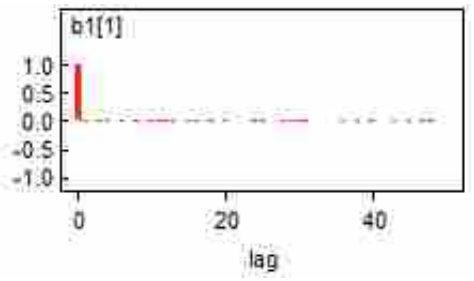
	mean	sd	2.5%	25%	50%	75%	97.5%
b[1]	78.47	14.75	49.14	68.75	78.53	88.16	107.50
b[2]	25.42	15.70	-6.13	14.92	25.33	35.72	55.84
b1[1]	1.24	0.07	1.11	1.20	1.24	1.28	1.37
s2	482.11	122.42	293.70	395.37	465.60	549.40	769.60
deviance	242.20	2.77	238.70	240.20	241.60	243.60	249.10

Model 1 DIC: 246.292

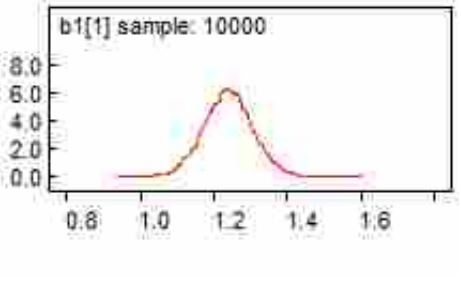
Model 2 DIC: 245.72



(a) Trace plot



(b) Autocorrelation



(c) Posterior density

Figure 9.2: WinBUGS summary plots for the posterior distribution of the single slope parameter from the second model.

The summary statistics for both models are shown in table 9.1. In comparing these two models, we see that DIC for model two is lower, therefore model two with the single slope fits better. Figure 9.2 gives a sample of the posterior summary plots, showing the posterior distribution of the single slope parameter from the second model. Convergence was reached, there were no problems with autocorrelation, and the posterior distribution is shown.

9.2 PROC MCMC

The models are presented in the code below in the same order as they appear in WinBUGS above. The data are read in and defined with lines one through four and printed for review before the analysis in lines six and seven. Lines nine, ten, and thirty-nine prepare a *.pdf file where SAS will save the output tables and graphs. Model one is coded in lines twelve through twenty-three and model two in lines twenty-six through thirty-six. Compare how μ is defined in lines twenty-one and thirty-four. In model one, μ is defined to have a slope and intercept for each group while in model two, it is defined to have two intercepts but the same slope for the two groups. As such, model one has an array of length two for both β_0 and β_1 while model two only needs an array of length two for β_0. Also notice that on lines twelve and twenty-six, the number of MCMC iterations has been increased along with the number of burn-in iterations and the thin option has been defined so that the simulation is thinned to take only every fiftieth one to reduce autocorrelation and reach convergence satisfactorily as seen in figure 9.3. The summary statistics for both models are shown in table 9.2.

```
* read in the data file;
1  data ancova;
2  infile ''     '';
3  input tmt speed scrap;
4  run;
5
* print the data file for inspection;
6  proc print data=ancova;
7  run;
8
* initializes saving of output as a pdf file;
9  ods pdf
10  file = ''     '';
* turn on graphics device;
11  ods graphics on;
12  proc mcmc data=ancova outpost=examp7out nmc=500000  nbi=1000 seed
   =1234 thin=50  monitor=(_parms_) dic;
* define arrays of length 2 for intercept and slope;
13       array beta0[2];
14       array beta1[2];
* set parameters and initial values;
```

```
* the colon on the betai's indicate that the initial values be applied
  to all array entries;
15       parms beta0: 150;
16       parms beta1: 0;
17       parms s2 500;
* define priors;
* the colon on the betai's indicate that the prior be applied to all
  array entries;
18       prior beta0: ~ normal(100,var=10000);
19       prior beta1: ~ normal(0, var=10);
20       prior s2 ~ gamma(7, scale=75);
* define the mean, which is the line;
21      mu = beta0[tmt] + beta1[tmt]*speed;
* likelihood;
22       model scrap ~ normal(mu, var=s2);
23   run;
24
25
* change code to have one slope now;
26   proc mcmc data=ancova outpost=examp7out nmc=500000  nbi=1000 seed
  =1234 thin=50  monitor=(_parms_) dic;
* define one array of length 2 for intercept;
27       array beta0[2];
* set parameters and initial values;
* the colon on beta0 indicates that the initial values be applied to
  all array entries;
28       parms beta0: 150;
29       parms beta1 0;
30       parms s2 500;
* define priors;
* the colon on beta0 indicates that the prior be applied to all array
  entries;
31       prior beta0: ~ normal(100,var=10000);
32       prior beta1 ~ normal(0, var=10);
33       prior s2 ~ gamma(7, scale=75);
* define the mean, which is the line;
34      mu = beta0[tmt] + beta1*speed;
* likelihood;
35       model scrap ~ normal(mu, var=s2);
36   run;
37
* turn off graphics device;
38   ods graphics off;
* stop saving output file;
39   ods pdf close;
```

Table 9.2: Summary Statistics for Example 7 from PROC MCMC.

Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
beta01	10000	100.2	20.2927	86.7662	100.4	113.4
beta02	10000	12.1241	21.8431	-2.2602	12.1257	26.6854
beta11	10000	1.1284	0.0946	1.0661	1.1282	1.1914
beta12	10000	1.3026	0.0966	1.2397	1.3017	1.3661
s2	10000	487.7	119.1	403.3	471.4	556.1

Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
beta01	10000	82.5032	15.2356	72.8622	82.6148	92.4310
beta02	10000	31.1614	16.2140	20.5151	31.0183	41.8791
beta1	10000	1.2142	0.0688	1.1688	1.2146	1.2584
s2	10000	501.0	120.0	416.8	486.7	569.6

9.3 Side by Side Computer Code

WinBUGS Code:

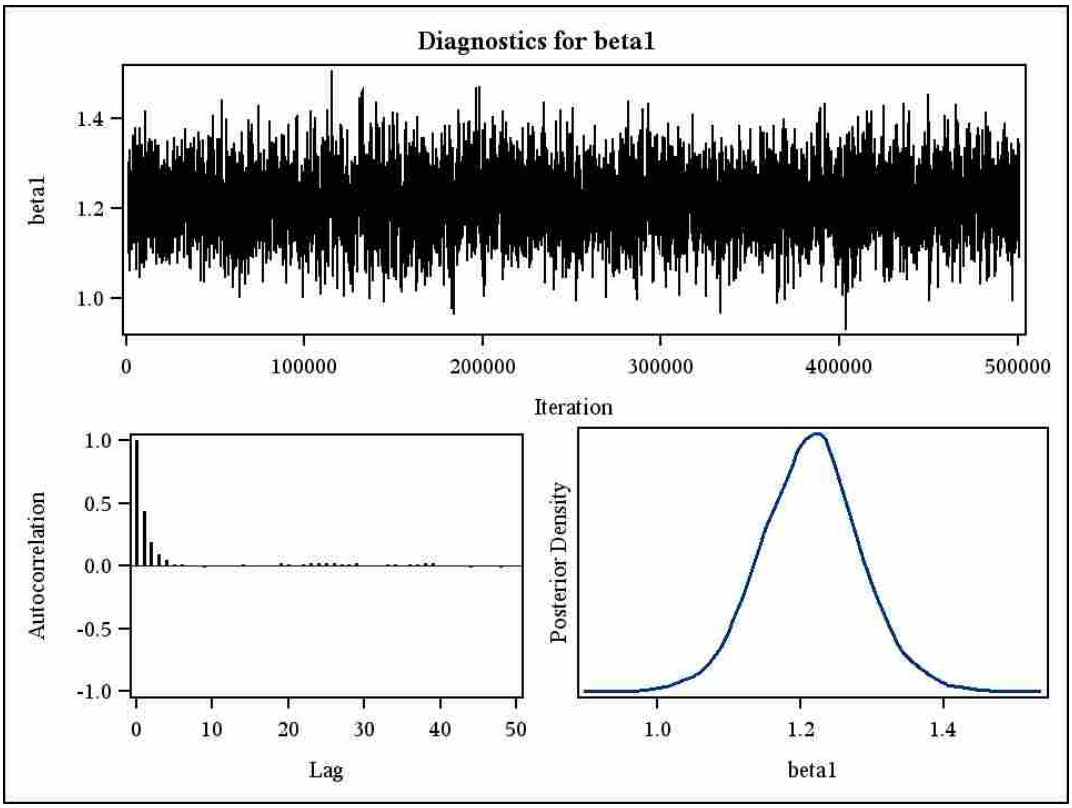
```
# model one has two intercepts
   and two slopes
model{
    for (i in 1:27){
       scrap[i] ~ dnorm(mu[i],prec
          );
       mu[i] <- b[line[i]] + b1[
```

SAS Code:

```
data ancova;
infile ‘‘     ’’;
input tmt speed scrap;
run;

proc print data=ancova;
run;
```

Figure 9.3: Summary plots for the posterior distribution of the single slope parameter from the second model.




```
    line[i]]*speed[i];
  }
  b[1] ~ dnorm(10, 0.0001);
  b[2] ~ dnorm(10, 0.0001);
  b1[1] ~ dnorm(0, 0.01);
  b1[2] ~ dnorm(0, 0.01);
  s2 ~ dgamma(7, 0.04);
  prec <- 1/s2;
}
```

```
ods pdf
file = ''     '';
ods graphics on;
proc mcmc data=ancova outpost=
   examp7out nmc=500000  nbi=1000
   seed=1234 thin=50  monitor=(
   _parms_) dic;
    array beta0[2];
    array beta1[2];
```

```
# model two has two intercepts
   and one slope
model{
    for (i in 1:27){
       scrap[i] ~ dnorm(mu[i],prec
          );
       mu[i] <- b[line[i]] + b1
          [1]*speed[i]
       }
       b[1] ~ dnorm(10, 0.0001);
       b[2] ~ dnorm(10, 0.0001);
       b1[1] ~ dnorm(0, 0.01);
       s2 ~ dgamma(5, 0.01);
       prec <- 1/s2;
    }
# The data set:
line[]  speed[]  scrap[]
1 100 218
1 125 248
1 220 360
1 205 351
1 300 470
1 255 394
1 225 332
1 175 321
1 270 410
1 170 260
       parms beta0: 150;
       parms beta1: 0;
       parms s2 500;
       prior beta0: ~ normal(100,var
          =10000);
       prior beta1: ~ normal(0, var
          =10);
       prior s2 ~ gamma(7, scale=75)
          ;
       mu = beta0[tmt] + beta1[tmt]*
          speed;
       model scrap ~ normal(mu, var=
          s2);
run;

* change code to have one slope
   now;
proc mcmc data=ancova outpost=
   examp7out nmc=500000  nbi=1000
   seed=1234 thin=50  monitor=(
   _parms_) dic;
    array beta0[2];
    parms beta0: 150;
    parms beta1 0;
    parms s2 500;
    prior beta0: ~ normal(100,var
       =10000);
```

```
1 155 241
1 190 331
1 140 275
1 290 425
1 265 367
2 105 140
2 215 277
2 270 384
2 255 341
2 175 215
2 135 180
2 200 260
2 275 361
2 155 252
2 320 422
2 190 273
2 295 410
END{};
```

```
        prior beta1 ~ normal(0, var
          =10);
        prior s2 ~ gamma(7, scale=75)
          ;
        mu = beta0[tmt] + beta1*speed
          ;
        model scrap ~ normal(mu, var=
          s2);
    run;

    ods graphics off;
    ods pdf close;
```

CHAPTER 10

LINEAR MIXED MODEL

In all of the previous models, the data were assumed to be independent and exchangeable, meaning that the order in which our sample was taken makes no difference in the probability of the sample occurring. The order of the indexes has no influence on the calculation of the probability.

However, it is not plausible to make the assumption for every data set. Here in the linear mixed model, the data are not independent which means that more than one source of variability must be accounted for. There is variability due to random error and fixed error. This is just an extension of the linear model where the linear predictor contained all of the variability. The usual model for the linear regression is

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}, \quad \mathbf{e} \sim \text{Normal}(0, \sigma^2 I).$$

The mixed model setting, however, is more complicated because the errors are not independent. The name is mixed because both random and fixed effects are mixed in the model, where before only fixed effects were modeled. Mixed models are applicable to settings where repeated measurements are taken on the same statistical unit, or where measurements are made on clusters of related statistical units. Often the goal of the researcher is to make inference on the entire population that these statistical units come from, and not just the sample itself.

The Bayesian paradigm easily adjusts for this form of analysis. A term must be added to the model that will account for the extra variability due to randomness. It is imperative that the extra source(s) of variability be accounted for so that the inference is valid. Typically, fixed effects are terms that have one level of priors modeling their parameters while random effects have priors modeling their priors, called hyperpriors. The

model now is

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Zu} + \mathbf{e}, \quad \mathbf{e} \sim \text{Normal}(0, \mathbf{R}), \mathbf{u} \sim \text{Normal}(0, \mathbf{G})$$

where $\mathbf{X}$ and $\mathbf{Z}$ are known design matrices and the covariance matrices $\mathbf{R}$ and $\mathbf{G}$ may depend upon a set of unknown variance components.

For this analysis, the data will be modeled as normal with priors and hyperpriors as shown below where i indicates the metal type, j indicates the ingot, and u is the effect of each ingot.

$$\begin{aligned}
y_{ijk} &\sim \text{Normal}(\mu_i, \sigma^2) \\
\mu_i &= \beta_i x_i + u_j \\
\beta_i &\sim \text{Normal}(72, 100) \\
u_j &\sim \text{Normal}(o, \sigma_u^2) \\
\sigma_u^2 &\sim \text{Gamma}(3, \text{scale} = \frac{1}{3}) \\
\sigma^2 &\sim \text{Gamma}(3, \text{scale} = \frac{1}{3})
\end{aligned}$$

Equations for the likelihood, prior, and posterior distributions are omitted here where they were provided in Chapter 3 because the MCMC algorithms do not require finding the functional form of the posterior distribution. All that is required is the likelihood function and the distribution for all parameters in the model. The MCMC algorithms calculate the posterior distribution from there.

10.1 WinBUGS

This data set comes from a study to determine the pressure required to break a metal's bond. The general goal was to compare the bond break pressure of the metals. The data set contained three columns of observations, ingot, metal, and pressure as can be seen in the side-by-side code section below. An ingot is a block of metal, typically oblong in shape,

and it is assumed that the seven ingots in the sample represent a much larger population of ingots to which the researcher desires to make inference.

The metals are nickel, iron, and calcium, indicated with n, i, and c in the data set. However, this poses a problem for WinBUGS because the program is not able to manage character entries. Therefore, a fourth column was added to the data set where n was given a numerical value of 1, i was given a numerical value of 2, and c a value of 3. This column will be used to inform WinBUGS as to which metal the observation belongs. The data set must be further adjusted such that the metal column is omitted prior to reading the data into WinBUGS because the program cannot work with character entries.

```
model{
   for (i in 1:21){
      # likelihood;
      pressure[i] ~ dnorm(mu[i],prec);
      # define the mean
      mu[i] <- beta[met[i]] + u[ingot[i]]
      }
      # the priors for mean and random effect
      for (i in 1:3){
      beta[i] ~ dnorm(72, .01);
      }
      for(i in 1:7){
      u[i] ~ dnorm(0, precing);
      }
      # prior and hyperprior for the variances and adjusting them in
         terms of precision
      s2 ~ dgamma(3,3);
      prec <- 1/s2;
      s2ing ~ dgamma(3,3);
      precing <- 1/s2ing;
   }
```

There are seven different ingots in this study and these are considered the random effects while the three metals are considered to be the fixed effects. These observations cannot be assumed independent because we have repeated measurements which means there are two sources of variability to account for in the model, the fixed effect error due to metal as indicated by σ^2 and the random effect error due to each ingot as indicated by σ_u^2.

Table 10.1: Summary statistics from WinBUGS.

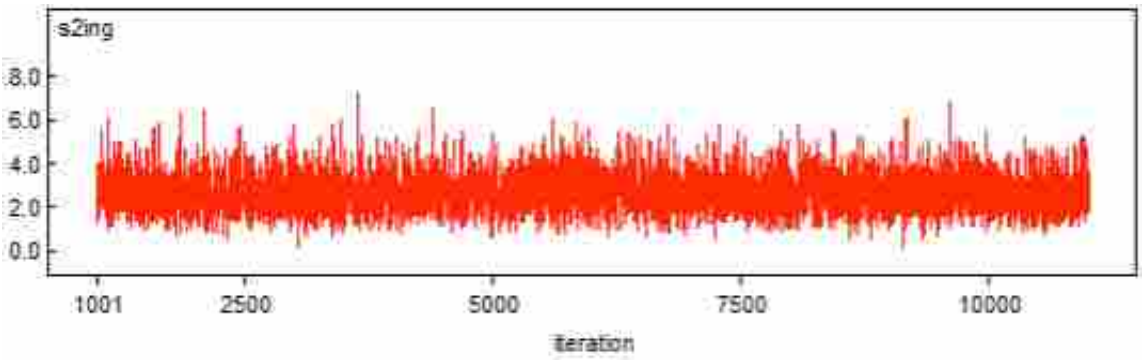
	mean	sd	2.5%	25%	50%	75%	97.5%
beta[1]	71.10	1.02	69.09	70.40	71.11	71.77	73.11
beta[2]	75.89	1.02	73.90	75.23	75.88	76.56	77.88
beta[3]	70.20	1.02	68.18	69.51	70.21	70.89	72.17
s2	4.67	0.92	3.14	4.02	4.57	5.22	6.70
s2ing	2.63	0.85	1.16	2.04	2.56	3.16	4.49
deviance	113.64	6.33	103.30	109.10	112.90	117.42	128.30

The summary statistics are shown in table 10.1. The mean pressure for breaking nickel was 71.10, the mean pressure for breaking iron was 75.89, and the mean pressure for breaking calcium was 70.20. Figure 10.1 gives a sample of the posterior summary plots, showing the posterior distribution of the error due to each ingot. Convergence was reached, there were no problems with autocorrelation, and the posterior distribution is shown. Even though plots for the variance components are not shown here, convergence of these components must be monitored carefully because variances are very challenging to model correctly and obtain convergence. These components were monitored in this analysis and convergence was indeed reached with no autocorrelation concerns.

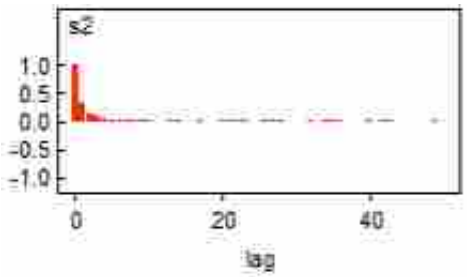
It should be noted that even though the random effect of ingot was accounted for in the model, this variable is not of concern because the goal was to generalize the results to all ingots. The mixed model and this hierarchical Bayesian model allow for the results to be applied to the entire population of ingots, and not just the seven in the study.

10.2 PROC MCMC

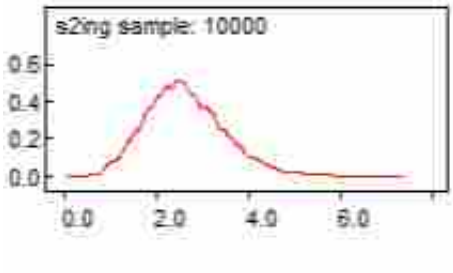
The code shown below asks SAS to create the same numeric column in lines four through six for the metal as was used for WinBUGS. The MCMC command begins on line fifteen and utilizes thinning of increased number of iterations after 10,000 burn-in iterations in an effort to reduce autocorrelation and reach convergence. Lines sixteen and seventeen create the needed arrays of length three for metal and of length seven for ingot. Lines eighteen, twenty-



(a) Trace plot



(b) Autocorrelation



(c) Posterior density

Figure 10.1: WinBUGS summary plots for the posterior distribution of the error due to the ingot.

one, twenty-two and twenty-three include a colon to indicate that the starting values and priors should be applied to all array entries. Line twenty-seven gives the model's likelihood statement.

```
* read in the data file;
1  data bond;
2       infile ``     '';
3            input ingot metal $ pressure;
* create a treatment column of numerical values;
4            if metal= 'n' then tmt=1;
5            if metal='i' then tmt=2;
6            if metal='c' then tmt=3;
7  run;
8
* print the data file for inspection;
9  proc print;
10  run;
11
* Initializes saving of output as a pdf file;
```

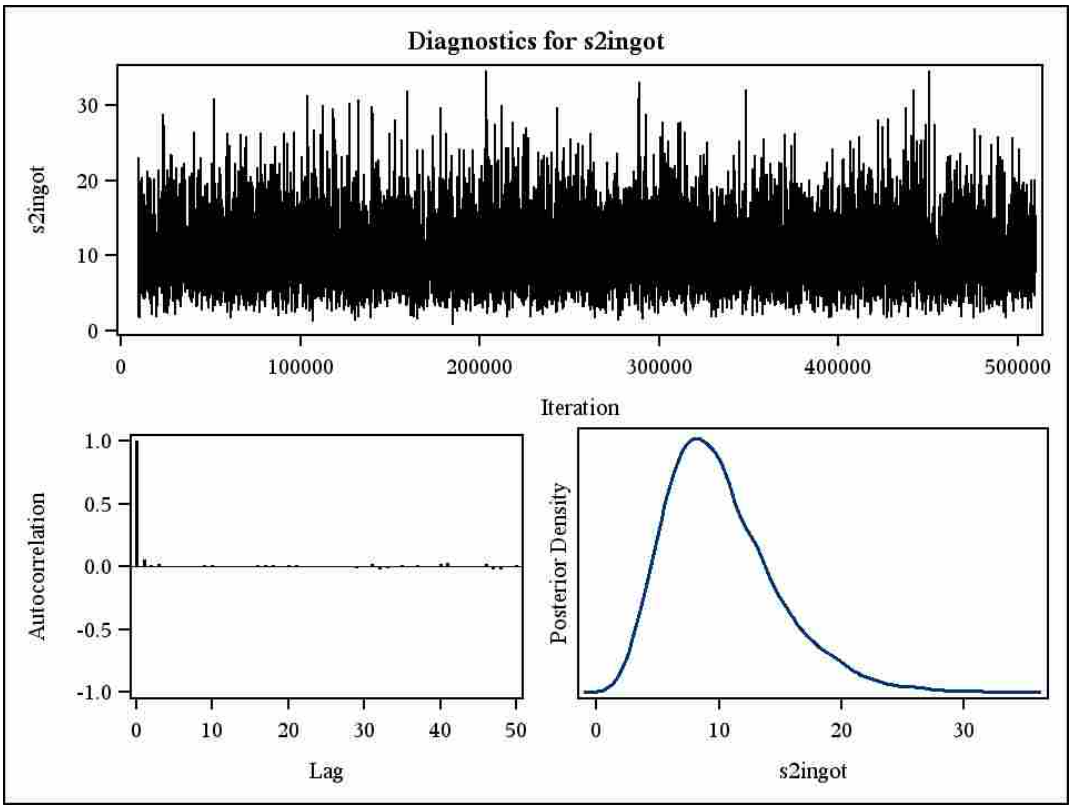
```
12  ods pdf
13  file = ''     '';
* turn on graphics device;
14  ods graphics on;
15  proc mcmc data=bond outpost=bondout nmc=500000 thin=50 nbi=10000
   monitor=(mu s2error s2ingot) dic seed=1234;
* create arrays for mean and random effect;
16       array mu[3];
17       array u[7];
* set parameters and initial values;
* the colon on mu and u indicate that the initial values be applied to
   all array entries;
18       parms mu: 70;
19       parms s2error 10;
20       parms s2ingot 10;
21       parms u: 0;
* define priors;
* the colon on mu and u indicate that the prior be applied to all array
    entries;
22       prior mu: ~ normal(72, var=100);
23       prior u: ~ normal(0, var=s2ingot);
24       prior s2error ~ gamma(3,scale=3);
25       prior s2ingot ~ gamma(3, scale=3);
* define the mixed model line;
26       line = mu[tmt] + u[ingot];
* likelihood;
27       model pressure ~ normal(line, var=s2error);
28  run;
29
* turn off graphics device;
30  ods graphics off;
* stop saving output file;
31  ods pdf close;
```

The summary statistics are shown in table 10.2 and give posterior values very similar to WinBUGS' for the pressure for breaking the bond, 71.09 for nickel, 75.84 for iron, and 70.18 for calcium. Figure 10.2 shows the posterior distribution of the error due to the ingot, indicating that convergence was reached and there was no problem with autocorrelation.

Table 10.2: Summary Statistics for Example 8 from PROC MCMC.

Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
mu1	10000	71.0869	1.6876	69.9875	71.0633	72.2041
mu2	10000	75.8358	1.7014	74.7305	75.8333	76.9608
mu3	10000	70.1805	1.7016	69.0329	70.1591	71.3149
s2error	10000	10.8084	3.4976	8.3146	10.2674	12.7592
s2ingot	10000	10.2217	4.5561	6.9417	9.4957	12.7537

Figure 10.2: Summary plots for the posterior distribution of the error due to the ingot.



10.3 Side by Side Computer Code

WinBUGS Code:

```
model{
   for (i in 1:21){
      pressure[i] ~ dnorm(mu[i],
         prec);
      mu[i] <- beta[met[i]] + u[
         ingot[i]]
      }

   for (i in 1:3){
      beta[i] ~ dnorm(72, .01);
      }
      for(i in 1:7){
      u[i] ~ dnorm(0, precing);
      }

      s2 ~ dgamma(3,3);
      prec <- 1/s2;
      s2ing ~ dgamma(3,3);
      precing <- 1/s2ing;
   }
# The data set:
ingot[] metal[] pressure[] met[]
1      n      67.0    1
```

SAS Code:

```
data bond;
    infile ''     '';
        input ingot metal $
           pressure;
        if metal= 'n' then tmt=1;
        if metal='i' then tmt=2;
        if metal='c' then tmt=3;
run;

proc print;
run;

ods pdf
file = ''     '';
ods graphics on;
proc mcmc data=bond outpost=
   bondout nmc=500000 thin=50 nbi
   =10000 monitor=(mu s2error
   s2ingot) dic seed=1234;
     array mu[3];
     array u[7];
     parms mu: 70;
     parms s2error 10;
```

```
1     i     71.9   2
1     c     72.2   3
2     n     67.5   1
2     i     68.8   2
2     c     66.4   3
3     n     76.0   1
3     i     82.6   2
3     c     74.5   3
4     n     72.7   1
4     i     78.1   2
4     c     67.3   3
5     n     73.1   1
5     i     74.2   2
5     c     73.2   3
6     n     65.8   1
6     i     70.8   2
6     c     68.7   3
7     n     75.6   1
7     i     84.9   2
7     c     69.0   3
END{};
```

```
        parms s2ingot 10;
        parms u: 0;
        prior mu: ~ normal(72, var
          =100);
        prior u: ~ normal(0, var=
          s2ingot);
        prior s2error ~ gamma(3,scale
          =3);
        prior s2ingot ~ gamma(3,
          scale=3);
        line = mu[tmt] + u[ingot];
        model pressure ~ normal(line,
           var=s2error);
    run;

    ods graphics off;
    ods pdf close;
```

CHAPTER 11

RANDOM COEFFICIENT MODEL

The random coefficient model is an extension of the linear mixed model. Here, the notion is that the regression equation will have fixed effects terms for overall intercept and for overall slope, but because the data consist of different groups of observations, there will also be terms for a random slope and a random intercept. Thus the coefficients in the model are allowed to vary for the random effects of the different groups. The design of this analysis is such as to allow for inference beyond the groups that are found in the sample data. The equation for a random coefficient model is

$$y_{ij} = \beta_0 + \beta_1 x_1 + \alpha_{0j} + \alpha_{1j} x_1 + \mathbf{e}.$$

This model not only allows for the adjustment of extra variation from the different groups, but also allows for the adjustment of different intercepts and slopes within each group. As can be seen in the graph of the data set in figure 11.1, it is plausible that there could be both an overall intercept and slope along with both an intercept and slope unique to each group. The result of the random coefficient model is that the researcher can generalize the analysis to include all possible groups in the population and not just those found in the sample. Sometimes, this extension is a very desirable attribute when conducting research. The population's average slope and intercept is calculated as $\beta_0 + \beta_1 x_1$ and each group's values are calculated as $\beta_0 + \alpha_{0j} + (\beta_1 + \alpha_{1j})x_1$.

For this analysis, the data will be modeled hierarchically to have a normal likelihood with priors and hyper-priors as shown below where i indicates the subject and j indicates group membership. Equations for the likelihood, prior, and posterior distributions are omitted here where they were provided in Chapter 3 because the MCMC algorithms do not require finding the functional form of the posterior distribution. All that is required is

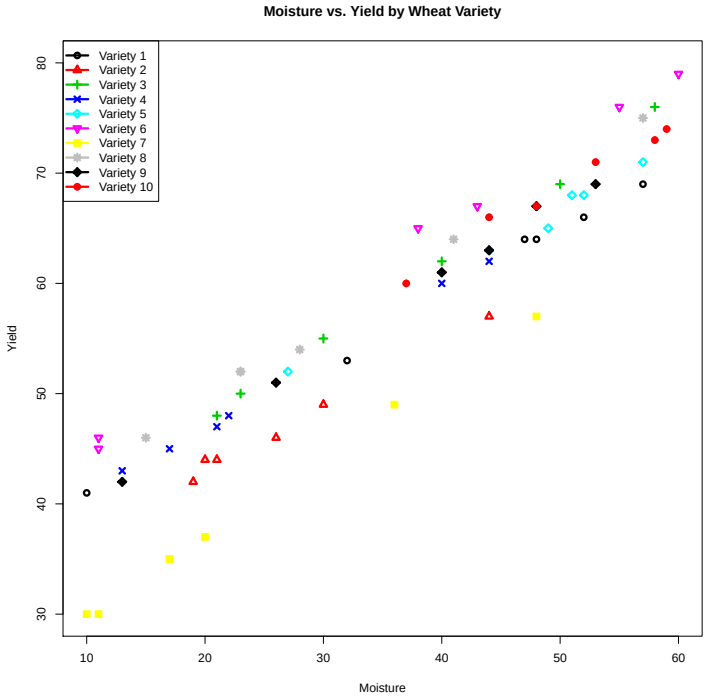


Figure 11.1: Linear relationship between moisture and yield. The different colors and shapes indicate group membership.

the likelihood function and the distribution for all parameters in the model. The MCMC algorithms calculate the posterior distribution from there.

$$
\begin{aligned}
y_{ij} &\sim \text{Normal}(\mu_{ij}, \sigma^2) \\
\mu &= \beta_0 + \beta_1 x_1 + \alpha_{0j} + \alpha_{1j} x_1 \\
\beta_0 &\sim \text{Normal}(30, 10000) \\
\beta_1 &\sim \text{Normal}(0, 100) \\
\alpha_0 &\sim \text{Normal}(0, \sigma^2_{intercept}) \\
\alpha_1 &\sim \text{Normal}(0, \sigma^2_{slope}) \\
\sigma^2 &\sim \text{Uniform}(0, 2) \\
\sigma^2_{intercept} &\sim \text{Uniform}(0, 200) \\
\sigma^2_{slope} &\sim \text{Uniform}(0, 0.2)
\end{aligned}
$$

11.1 WinBUGS

The data for this analysis are from an agriculture study on wheat varieties. The purpose was to predict yield based on moisture while taking into account an effect for different varieties of wheat. The data include observations on ten different varieties, but because of the hierarchical model, inference can be made beyond these ten varieties to the entire population of wheat varieties. The wheat varieties are random effects and the moisture variable is the fixed effect in the model. The fixed effect will have specific priors while the random effects will have a mean of zero and a hierarchical structure for the variance. Remember to choose specific priors that preserve the parameter space, and as such, the variance priors must be modeled with positive values.

```
model{
   # dummy variable to use all columns of data set
   dummy <- obs[1];
   for(i in 1:60){
      # likelihood
      yield[i] ~ dnorm(mu[i], prec);
```

```
    # define the mean
    mu[i] <- b0 + b1*moisture[i] + a0[variety[i]] + a1[variety[i]]*
       moisture[i];
}
# the priors for betai
b0 ~ dnorm(30, .001);
b1 ~ dnorm(0, .01);
# the priors for alphai
for(i in 1:10){
    a0[i] ~ dnorm(0,precint);
    a1[i] ~ dnorm(0, precslp);
    }
    # priors for variance parameters and adjusting them in terms of
       precision
    s2 ~ dunif(0, 2);
    prec <- 1/s2;
    s2int ~ dunif(0, 200);
    precint <- 1/s2int;
    s2slp ~ dunif(0,.2);
    precslp <- 1/s2slp;
}
```

Because variance values are positive real numbers, the researcher should thoughtfully choose appropriate prior distributions to model them, drawing upon previous experience or knowledge of the data. Possible variance priors are the gamma and uniform distributions, however, modeling hierarchical variances can be very difficult unless the researcher has a good sense of the data's behavior. When the researcher does have a good sense, then appropriate gamma priors could be thoughtfully selected. However, since we do not have a good sense of this data, uniform priors were selected as a good alternate choice because the parameter space could still be preserved.

A word of caution though, when using uniform priors on variance parameters, it is important to monitor the trace plots closely because the uniform could prevent the algorithm from moving into values beyond the bounds of the distribution even if the MCMC random walk attempts such movement. Watch for a trace plot that looks like a butch hair cut. When such a trace plot is found, return to the code and make adjustments on the prior values as needed to allow the MCMC random walk to cover the parameter space as needed. Trace

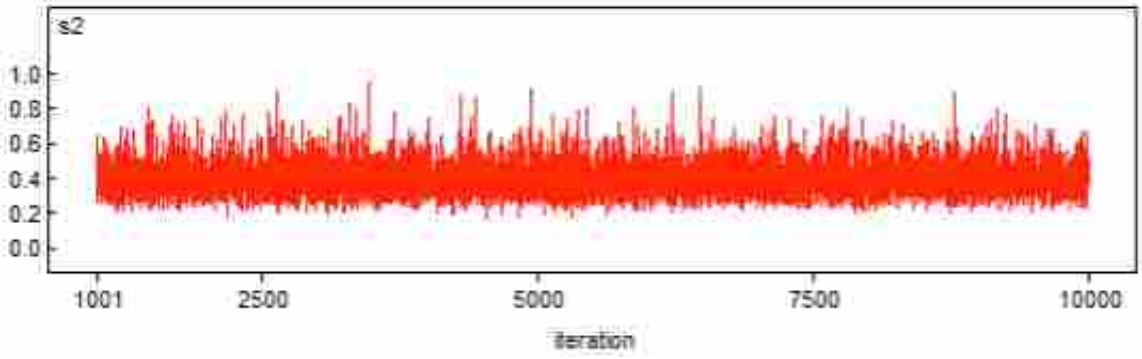
plots can also guide in narrowing the uniform interval if the interval is too broad and allows the MCMC random walk too much movement.

Table 11.1: Summary statistics from WinBUGS.

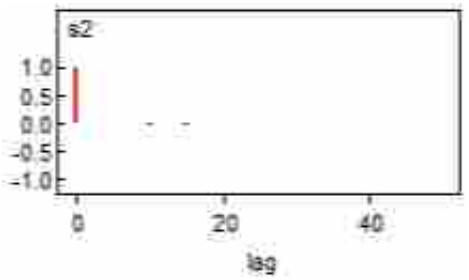
	mean	sd	2.5%	25%	50%	75%	97.5%
b0	33.51	1.80	29.99	32.42	33.49	34.58	37.25
b1	0.66	0.02	0.62	0.65	0.66	0.67	0.71
s2	0.39	0.09	0.25	0.32	0.38	0.44	0.62
s2int	31.80	21.82	10.15	18.05	25.56	38.06	94.05
s2slp	0.00	0.00	0.00	0.00	0.00	0.01	0.01
a0[1]	0.88	1.90	-2.99	-0.29	0.89	2.06	4.61
a0[2]	-2.24	1.92	-6.11	-3.42	-2.21	-0.99	1.50
a0[3]	-0.50	1.91	-4.42	-1.65	-0.47	0.70	3.15
a0[4]	0.62	1.87	-3.22	-0.51	0.67	1.78	4.26
a0[5]	0.99	2.05	-3.15	-0.30	0.98	2.27	5.04
a0[6]	4.53	1.87	0.74	3.40	4.54	5.68	8.18
a0[7]	-10.73	1.85	-14.60	-11.82	-10.69	-9.56	-7.12
a0[8]	2.29	1.86	-1.54	1.18	2.31	3.45	5.98
a0[9]	-0.24	1.91	-4.13	-1.42	-0.21	0.96	3.48
a0[10]	3.60	2.19	-0.73	2.17	3.58	5.04	7.95
a1[1]	-0.05	0.03	-0.10	-0.07	-0.05	-0.03	0.00
a1[2]	-0.07	0.03	-0.14	-0.09	-0.07	-0.05	-0.01
a1[3]	0.07	0.03	0.01	0.05	0.07	0.09	0.12
a1[4]	-0.02	0.03	-0.08	-0.04	-0.02	-0.01	0.03
a1[5]	-0.02	0.03	-0.08	-0.04	-0.02	0.00	0.04
a1[6]	0.02	0.02	-0.02	0.01	0.02	0.04	0.08
a1[7]	0.05	0.03	-0.00	0.04	0.05	0.07	0.11
a1[8]	0.02	0.03	-0.03	0.01	0.02	0.04	0.08
a1[9]	0.02	0.03	-0.03	0.01	0.02	0.04	0.08
a1[10]	-0.03	0.03	-0.10	-0.05	-0.03	-0.01	0.03
deviance	110.40	8.37	96.31	104.40	109.60	115.50	128.90

The summary statistics are shown in table 11.1. Notice that, as expected, the analysis gives posterior distributions for an overall intercept and overall slope along with posterior distributions for three variance parameters, ten variety specific intercepts, and ten variety specific slopes. Figure 11.2 gives a sample of the posterior summary plots, showing the posterior distribution of the fixed effect's variance. This trace plot shows that

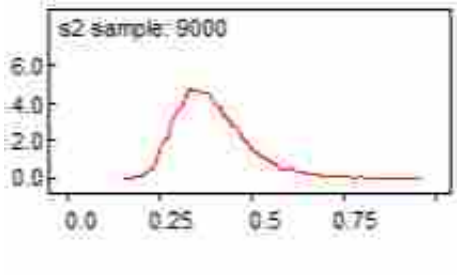
convergence was reached, indicating that the selected uniform prior was indeed appropriate for this parameter. There were no problems with autocorrelation.



(a) Trace plot



(b) Autocorrelation



(c) Posterior density

Figure 11.2: WinBUGS summary plots for the posterior distribution of the fixed effect error.

11.2 PROC MCMC

The coding of the random coefficient model in SAS is done similarly as in previous models. Lines one through four read in the data file and tell SAS what is should find therein. Lines six, seven, and thirty-five create and close a *.pdf file where SAS will save the posterior summary tables and plots that lines eight and thirty-four initiated and closed. The MCMC procedure consists of lines nine through twenty-nine. Of note on line nine is the number of burn-in iterations and the number of MCMC iterations along with the indication to thin every 100. The number of iterations was increased here and the thinning was increased to 100 so as to reduce autocorrelation and aid in the convergence process. Arrays are created in lines ten and eleven for the random slope and intercept parameters. Lines twelve through

eighteen give initial values for all parameters while lines nineteen through twenty-six define prior distributions for them. The random coefficient equation is defined in line twenty-seven. The likelihood is given in line twenty-eight. The posterior draws are created and saved in lines thirty-one and thirty-two.

The table of summary statistics is presented in table 11.2 and gives posterior values very similar to WinBUGS. Figure 11.3 gives the posterior plots for the fixed effect's variance. These plots indicate that convergence was reached, no autocorrelation problems were encountered and the density of the posterior is drawn.

```
* read in the data file;
1  data wheat;
2       infile ``     '' firstobs=2;
3       input obs variety yield moisture;
4  run;
5
* initializes saving of output as a pdf file;
6  ods pdf
7  file = ``     '';
* turn on graphics device;
8  ods graphics on;
9  proc mcmc data=wheat nbi=100000 nmc=1000000 thin=100 outpost=
   postwheat dic seed=1234 monitor=(_parms_);
* define arrays of length 10 for alphi's;
10       array a0[10];
11       array a1[10];
* set parameters and initial values;
* the colon on the alphai's indicate that the initial values be applied
    to all array entries;
12       parms b0 30;
13       parms b1 0;
14       parms a0: 0;
15       parms a1: 0;
16       parms s2 1;
17       parms s2slp .004;
18       parms s2int 30;
* define the priors;
* the colon on the alphai's indicate that the prior be applied to all
   array entries;
19       prior a0: ~ normal(0,var=s2int);
20       prior a1: ~ normal(0, var=s2slp);
21       prior b0 ~ normal(30, var=1000);
22       *variance is reciprocal of WinBUGS precision;
```

```
23        prior b1 ~ normal(0, var=100);
24        prior s2 ~ uniform(0,2);
25        prior s2int ~ uniform(0,200);
26        prior s2slp ~ uniform(0, .2);
* define the random coefficients line;
27       mu = b0 + b1 * moisture + a0[variety] + a1[variety]*moisture;
* likelihood;
28        model yield ~ normal(mu, var=s2);
29   run;
30
* export the posterior MCMC draws and save the .csv file;
31   proc export data=postwheat outfile=''     '' dbms=csv replace;
32   run;
33
* turn off graphics device;
34   ods graphics off;
* stop saving output file;
35   ods pdf close;
```

Figure 11.3: Summary plots for the posterior distribution of the fixed effect error.

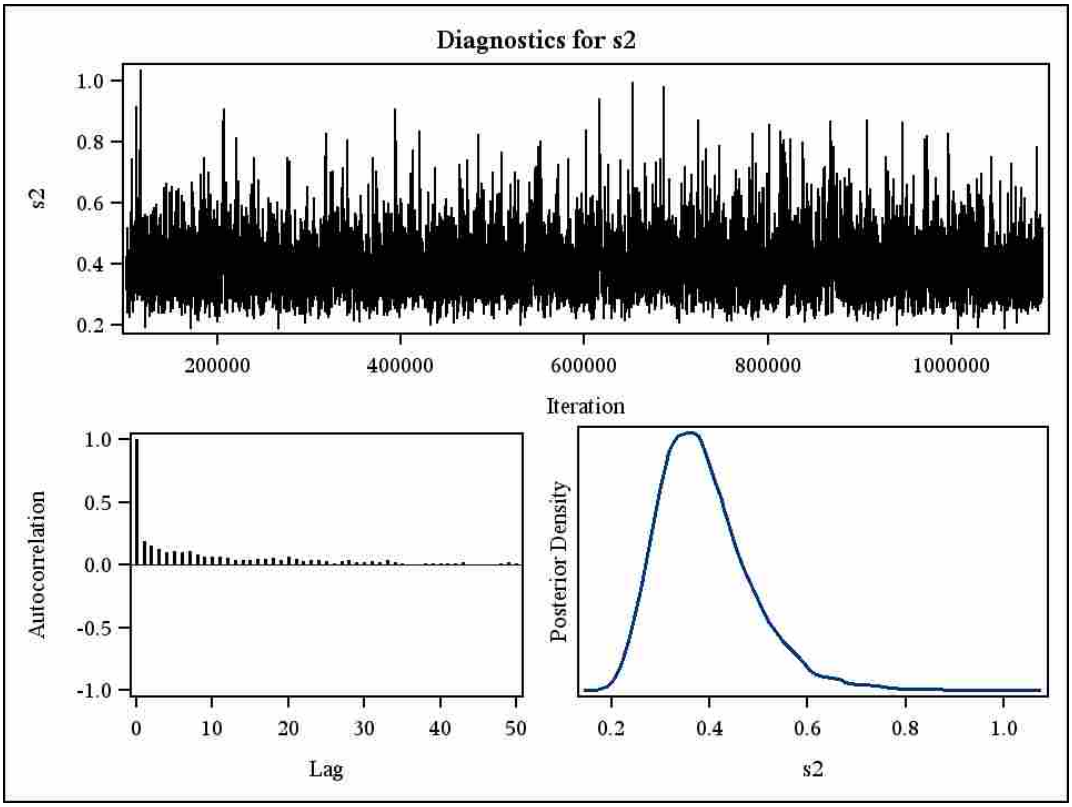


Table 11.2: Summary Statistics for Example 9 from PROC MCMC.

Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
b0	10000	33.3915	1.8131	32.2157	33.2575	34.4000
b1	10000	0.6628	0.0229	0.6482	0.6628	0.6769
a01	10000	0.9742	1.9520	-0.1256	1.1024	2.2535
a02	10000	-2.0732	1.8871	-3.1367	-2.0134	-0.8854
a03	10000	-0.4205	1.9880	-1.5569	-0.3098	0.9436
a04	10000	0.7638	1.8416	-0.2493	0.8688	1.9479
a05	10000	1.0764	2.1092	-0.2014	1.0126	2.3757
a06	10000	4.6656	1.8665	3.5841	4.7933	5.8752
a07	10000	-10.6084	1.8753	-11.6604	-10.4881	-9.3950
a08	10000	2.3814	1.9063	1.2861	2.4574	3.6102
a09	10000	-0.1549	1.9177	-1.2741	-0.0376	1.1177
a010	10000	3.7665	2.0910	2.4769	3.7975	5.0587
a11	10000	-0.0498	0.0262	-0.0664	-0.0496	-0.0329
a12	10000	-0.0745	0.0337	-0.0964	-0.0732	-0.0512
a13	10000	0.0678	0.0285	0.0485	0.0667	0.0857
a14	10000	-0.0253	0.0288	-0.0435	-0.0247	-0.00575
a15	10000	-0.0193	0.0313	-0.0383	-0.0177	0.00158
a16	10000	0.0235	0.0257	0.00730	0.0235	0.0395
a17	10000	0.0520	0.0285	0.0334	0.0515	0.0699
a18	10000	0.0233	0.0274	0.00590	0.0234	0.0410
a19	10000	0.0229	0.0273	0.00506	0.0227	0.0406
a110	10000	-0.0340	0.0333	-0.0557	-0.0339	-0.0119
s2	10000	0.3906	0.0961	0.3224	0.3765	0.4415
s2slp	10000	0.00453	0.00380	0.00231	0.00349	0.00545
s2int	10000	32.4943	22.5842	18.3688	26.0373	38.7539

11.3 Side by Side Computer Code

WinBUGS Code:

```
model{
   dummy <- obs[1];
   for(i in 1:60){
      yield[i] ~ dnorm(mu[i],
         prec);
      mu[i] <- b0 + b1*moisture[i
         ] + a0[variety[i]] + a1[
         variety[i]]*moisture[i];
   }

   b0 ~ dnorm(30, .001);
   b1 ~ dnorm(0, .01);

   for(i in 1:10){
      a0[i] ~ dnorm(0,precint);
      a1[i] ~ dnorm(0, precslp);
      }
      s2 ~ dunif(0, 2);
      prec <- 1/s2;
      s2int ~ dunif(0, 200);
      precint <- 1/s2int;
      s2slp ~ dunif(0,.2);
      precslp <- 1/s2slp;
```

SAS Code:

```
data wheat;
    infile ''     '' firstobs=2;
    input obs variety yield
       moisture;
run;

ods pdf
file = ''     '';
ods graphics on;
proc mcmc data=wheat nbi=100000
   nmc=1000000 thin=100 outpost=
   postwheat dic seed=1234 monitor
   =(_parms_);
     array a0[10];
     array a1[10];
     parms b0 30;
     parms b1 0;
     parms a0: 0;
     parms a1: 0;
     parms s2 1;
     parms s2slp .004;
     parms s2int 30;
     prior a0: ~ normal(0,var=
```

```
   }
#The data set:
 obs[]  variety[]   yield[]
    moisture[]
 1        1          41        10
 2        1          69        57
 3        1          53        32
 4        1          66        52
 5        1          64        47
 6        1          64        48
 7        2          49        30
 8        2          44        21
 9        2          44        20
10        2          46        26
11        2          57        44
12        2          42        19
13        3          69        50
14        3          62        40
15        3          50        23
16        3          76        58
17        3          48        21
18        3          55        30
19        4          48        22
20        4          60        40
21        4          45        17
22        4          47        21
23        4          62        44
```

```
      s2int);
    prior a1: ~ normal(0, var=
      s2slp);
    prior b0 ~ normal(30, var
      =1000);
    *variance is reciprocal of
      WinBUGS precision;
    prior b1 ~ normal(0, var=100)
      ;
    prior s2 ~ uniform(0,2);
    prior s2int ~ uniform(0,200);
    prior s2slp ~ uniform(0, .2);
    mu = b0 + b1 * moisture + a0[
      variety] + a1[variety]*
      moisture;
    model yield ~ normal(mu, var=
      s2);
run;

proc export data=postwheat
   outfile=''     '' dbms=csv
   replace;
run;

ods graphics off;
ods pdf close;
```

24	4	43	13
25	5	65	49
26	5	63	44
27	5	71	57
28	5	68	51
29	5	52	27
30	5	68	52
31	6	76	55
32	6	46	11
33	6	45	11
34	6	67	43
35	6	65	38
36	6	79	60
37	7	35	17
38	7	37	20
39	7	30	11
40	7	30	10
41	7	57	48
42	7	49	36
43	8	75	57
44	8	64	41
45	8	46	15
46	8	54	28
47	8	52	23
48	8	52	23
49	9	51	26
50	9	63	44

51	9	42	13
52	9	61	40
53	9	67	48
54	9	69	53
55	10	60	37
56	10	73	58
57	10	66	44
58	10	71	53
59	10	67	48
60	10	74	59

END{};

CHAPTER 12

LOGISTIC REGRESSION WITH A BINOMIAL LIKELIHOOD

When the researcher is looking at success/failure data or even count data, a normal likelihood is not the appropriate choice of distribution to model the data. So it is for this logistic regression with a binomial likelihood example.

The binomial likelihood models discrete data counting the number of successes in a sequence of n independent yes/no experiments. Each experiment will yield a success with probability p. A single experiment, when $n = 1$, is called a Bernoulli trial. A binomial distribution consists of n such experiments with success probability p; n is fixed or set and the parameter of interest is p, the probability of success. As such, p is restricted to be in the interval $0 \leq p \leq 1$. The maximum likelihood estimator for p is (number of successes)/n. However, a Bayesian model will be explained herein.

Recall that the odds for an experiment are found as $\frac{p}{1-p}$. The log of the odds will be set equal to the regression line with an intercept and coefficients for each of the covariates,

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3.$$

The log of the odds, or a logit transformation, is used because this function keeps things in their proper domain. The logit transformation allows for values in the regression equation along the entire real line, but also keeps p in its restricted interval. Thus, p is transformed from the real line to the interval $0 \leq p \leq 1$ and the parameter space is preserved. The logit transformation allows the β's to be any real number, while preserving the parameter space of the binomial p.

The data will be modeled with a binomial likelihood and normal priors as shown. Additionally, the logit transformation links the regression line to the binomial probability p.

$$
\begin{aligned}
y \sim & \text{Binomial}(n, p) \\
\text{logit}(p) = & \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 \\
\beta_0 \sim & \text{Normal}(0, 1) \\
\beta_1 \sim & \text{Normal}(0, 1) \\
\beta_2 \sim & \text{Normal}(0, 1) \\
\beta_3 \sim & \text{Normal}(0, 1)
\end{aligned}
$$

Equations for the likelihood, prior, and posterior distributions are omitted here where they were provided in Chapter 3 because the MCMC algorithms do not require finding the functional form of the posterior distribution. All that is required is the likelihood function and the distribution for all parameters in the model. The MCMC algorithms calculate the posterior distribution from there.

Table 12.1: Arrangement of data is like a three-way ANOVA.

		Young	Old
Low	Normal ECG	P_1	P_2
Catecholamine	Abnormal ECG	P_3	P_4
High	Normal ECG	P_5	P_6
Catecholamine	Abnormal ECG	P_7	P_8

12.1 WinBUGS

The data for this example include eight different groups of patients in an observational study who are at risk of developing coronary heart disease (CHD) and are shown in the side-by-side code section below. These patients were stratified into eight groups as determined by how they exhibited four characteristics, or covariates. There are five columns, one response column and four covariate columns. The values in the response column of CHD are a count of

the number of patients who developed coronary heart disease. The covariates are n, the total number of patients in each group; catecholamine, low=0 or high=1; age group, young=0 or old=1; and abnormal ECG, no=0 or yes=1. The data may be placed in a table like unto a three-way ANOVA as shown in table 12.1, indicating that the analysis will look for eight different binomial probabilities. Indeed, the binomial likelihood appears to be appropriate because the data set gives the number of patients who developed the disease out of a total number of patients at risk of possibly developing the disease.

```
model{
   for(i in 1:8){
   # likelihood -- note that WinBUGS requires p first for dbin()
   CHD[i] ~ dbin(p[i], nRisk[i]);
   # logit transformation to preserve parameter space of p
   logit(p[i]) <- bint + bcat*Cat[i] + bage*agegrp[i] + becg*abECG[i];
   }
   # priors for each beta_i
   bint ~ dnorm(0,1);
   bcat ~ dnorm(0,1);
   bage ~ dnorm(0,1);
   becg ~ dnorm(0,1);
   }
```

Table 12.2: Summary statistics from WinBUGS

	mean	sd	2.5%	25%	50%	75%	97.5%
bint	-2.50	0.19	-2.89	-2.63	-2.50	-2.37	-2.13
bcat	0.58	0.30	-0.02	0.39	0.58	0.79	1.18
bage	0.51	0.26	0.00	0.33	0.51	0.69	1.02
becg	0.30	0.28	-0.25	0.12	0.30	0.48	0.83
p[1]	0.08	0.01	0.05	0.07	0.08	0.09	0.11
p[2]	0.12	0.02	0.08	0.11	0.12	0.14	0.18
p[3]	0.10	0.03	0.06	0.08	0.10	0.12	0.16
p[4]	0.16	0.04	0.09	0.13	0.16	0.18	0.24
p[5]	0.13	0.04	0.07	0.10	0.13	0.16	0.22
p[6]	0.20	0.05	0.12	0.17	0.20	0.23	0.30
p[7]	0.17	0.05	0.10	0.14	0.17	0.20	0.27
p[8]	0.25	0.05	0.17	0.22	0.25	0.28	0.35
deviance	33.74	2.87	30.14	31.62	33.08	35.18	41.03

Please note that the parameterization of the binomial likelihood in WinBUGS takes p first and n second. It is crucial that the researcher become aware of the distributional definitions WinBUGS is programed with along with those SAS is programmed with. Their parameterizations are not always equivalent and adjustments need to be made when needed.

The summary statistics are shown in table 12.2, giving summaries for the four β parameters and the eight binomial probabilities, p. Figure 12.1 gives a sample of the posterior summary plots, showing the posterior distribution of the intercept parameter. The plots indicate that convergence was reached and that there were no problems with autocorrelation.

One of the most useful mathematical properties of Bayesian logistic regression is that the parameters can be unraveled in the output. The β_i's and p_i's are related to each other and as such, can be calculated from the other

$$\mathbf{p} = \frac{1}{1 + e^{-\mathbf{X}\boldsymbol{\beta}}}.$$

The $\mathbf{X}$ matrix is the design matrix of zeros and ones that "turns on" each β_i value when it applies to a treatment combination. The following equations give each of the equivalencies particular to this analysis.

$$
\begin{aligned}
p_1 &= \frac{1}{1+e^{-\beta_{int}}} \\
p_2 &= \frac{1}{1+e^{-\beta_{int}-\beta_{age}}} \\
p_3 &= \frac{1}{1+e^{-\beta_{int}-\beta_{ecg}}} \\
p_4 &= \frac{1}{1+e^{-\beta_{int}-\beta_{ecg}-\beta_{age}}} \\
p_5 &= \frac{1}{1+e^{-\beta_{int}-\beta_{cat}}} \\
p_6 &= \frac{1}{1+e^{-\beta_{int}-\beta_{cat}-\beta_{age}}} \\
p_7 &= \frac{1}{1+e^{-\beta_{int}-\beta_{cat}-\beta_{ecg}}} \\
p_8 &= \frac{1}{1+e^{-\beta_{int}-\beta_{cat}-\beta_{age}-\beta_{ecg}}}
\end{aligned}
$$

12.2 PROC MCMC

The coding of logistic regression with a binomial likelihood follows the same pattern as previous models. New to this model is the inclusion of the logistic function as found in line twenty-two. The likelihood given in line twenty-three gives the binomial parameterization SAS is programmed for. The MCMC procedure consists of lines thirteen through twenty-four.

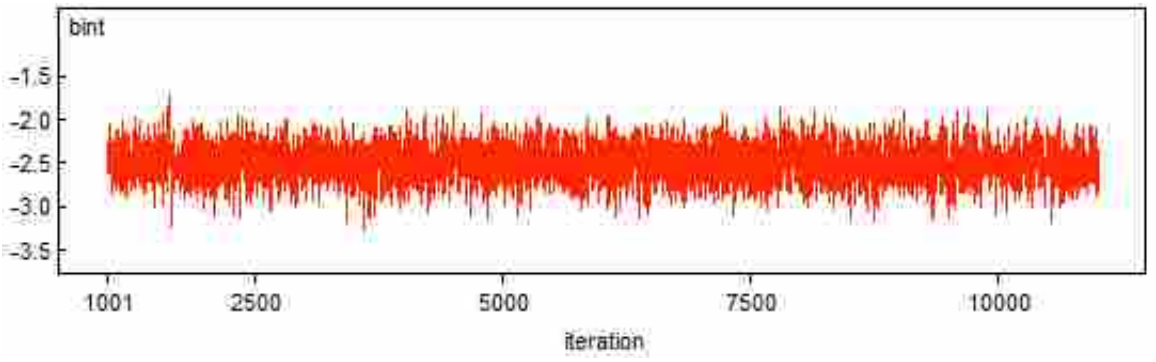
The summary statistics are given in table 12.3 and give summaries for the four β's but only one p. The eight p's can be calculated using the above equations and the results will be very similar to those given by WinBUGS. Figure 12.2 gives the posterior plots for the intercept parameter. These plots indicate that convergence was reached, no autocorrelation problems were encountered and the density of the posterior is drawn.

```
* read in the data file;
1  data heart;
2       infile ''     '' firstobs=2;
3       input CHD nRisk Cat agegrp abECG;
```

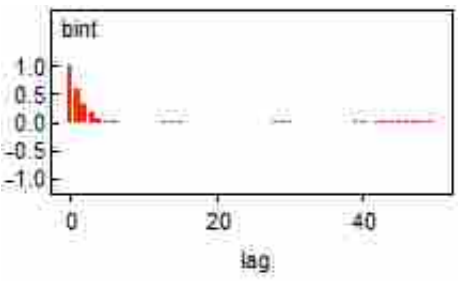
```
4  run;
5
* print the data file for inspection;
6  proc print;
7  run;
8
* initializes saving of output as a pdf file;
9  ods pdf
10     file="    ";
* turn on graphics device;
11 ods graphics on;
12
13 proc mcmc data=heart outpost=heartout nmc=500000 thin=50 nbi=10000
   monitor=(_parms_ pi) dic seed=1234;
* set parameters and initial values;
14     parms bint 0;
15     parms bcat 0;
16     parms bage 0;
17     parms becg 0;
* define priors;
18     prior bint ~ normal(0,var=1);
19     prior bcat ~ normal(0,var=1);
20     prior bage ~ normal(0,var=1);
21     prior becg ~ normal(0,var=1);
* logit transform equation;
22     pi = logistic(bint + bcat*CAT + bage*agegrp + becg*abECG);
* likelihood;
23     model CHD ~ binomial(n=nRisk, p=pi);
24 run;
25
* turn off graphics device;
26 ods graphics off;
* stop saving output file;
27 ods pdf close;
```

Table 12.3: Summary Statistics for Example 10 from PROC MCMC.

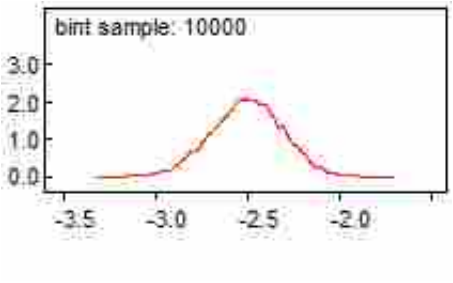
Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
bint	10000	-2.5046	0.1983	-2.6363	-2.5008	-2.3718
bcat	10000	0.5882	0.3041	0.3884	0.5918	0.7939
bage	10000	0.5125	0.2687	0.3302	0.5130	0.6928
becg	10000	0.3040	0.2734	0.1234	0.3053	0.4914
pi	10000	0.2525	0.0454	0.2207	0.2507	0.2829



(a) Trace plot



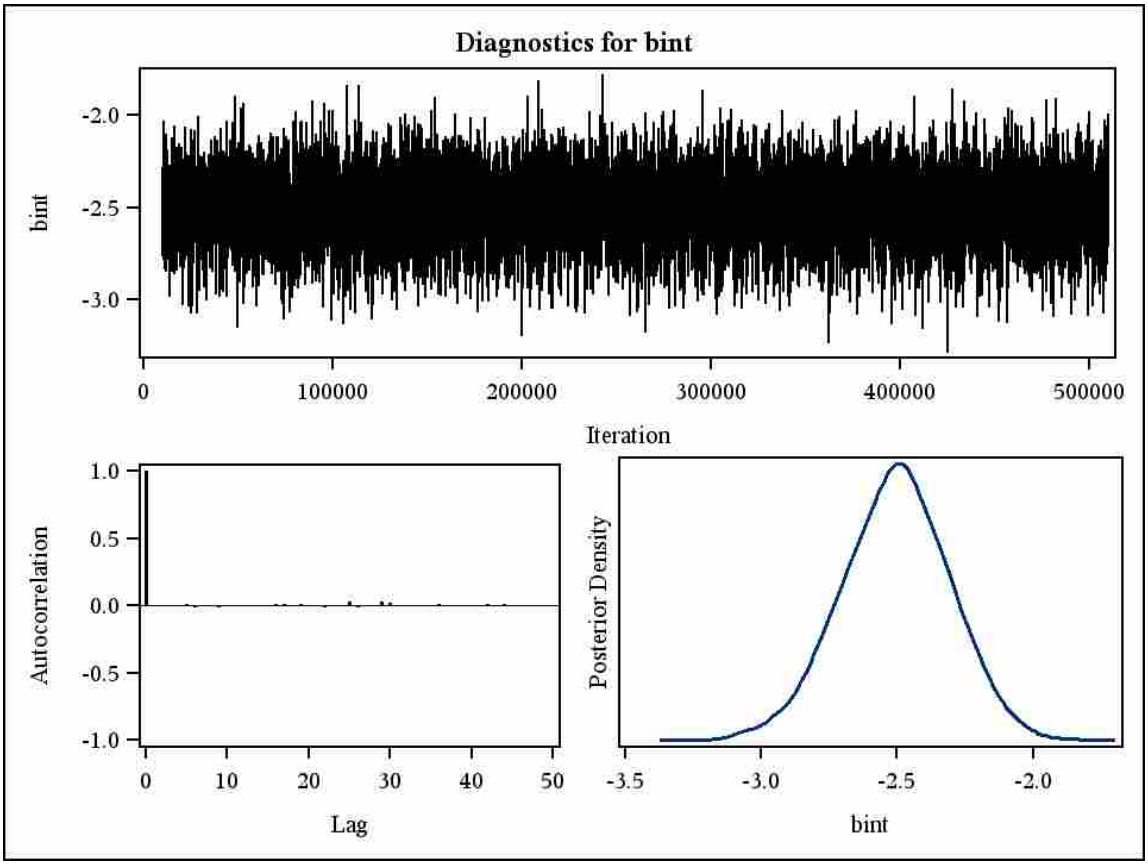
(b) Autocorrelation



(c) Posterior density

Figure 12.1: WinBUGS summary plots for the posterior distribution of the intercept parameter.

Figure 12.2: Summary plots for the posterior distribution of the intercept parameter.



12.3 Side by Side Computer Code

WinBUGS code:

```
model{
   for(i in 1:8){
   CHD[i] ~ dbin(p[i], nRisk[i]);
   logit(p[i]) <- bint + bcat*Cat
      [i] + bage*agegrp[i] + becg*
      abECG[i];
   }
   bint ~ dnorm(0,1);
   bcat ~ dnorm(0,1);
   bage ~ dnorm(0,1);
   becg ~ dnorm(0,1);
   }

#The data set:
CHD[] nRisk[] Cat[] agegrp[]
   abECG[]
17  274  0  0  0
15  122  0  1  0
 7   59  0  0  1
 5   32  0  1  1
 1    8  1  0  0
 9   39  1  1  0
 3   17  1  0  1
```

SAS code:

```
data heart;
    infile ''    '' firstobs=2;
    input CHD nRisk Cat agegrp
       abECG;
run;

proc print;
run;

ods pdf
    file="    ";
ods graphics on;

proc mcmc data=heart outpost=
   heartout nmc=500000 thin=50 nbi
   =10000 monitor=(_parms_ pi) dic
    seed=1234;
    parms bint 0;
    parms bcat 0;
    parms bage 0;
    parms becg 0;
    prior bint ~ normal(0,var=1);
    prior bcat ~ normal(0,var=1);
```

```
14   58   1   1   1
END{};
```

```
        prior bage ~ normal(0,var=1);
        prior becg ~ normal(0,var=1);
        pi = logistic(bint + bcat*CAT
            + bage*agegrp + becg*abECG
           );
        model CHD ~ binomial(n=nRisk,
            p=pi);
    run;

    ods graphics off;
    ods pdf close;
```

CHAPTER 13

LOGISTIC REGRESSION WITH RANDOM EFFECT

When the dependent response variable is a proportion, the traditional approach is to perform a logit transformation on the data. This approach is appropriate when the data give the number of successes out of the total number of trials as in a binomial likelihood.

In this setting π, the binomial probability, is modeled as

$$\pi = \frac{1}{1 + e^{-\mathbf{X}\boldsymbol{\beta}}}.$$

The logit transformation links π and the $\boldsymbol{\beta}$ parameters with the function

$$\log\left(\frac{\pi}{1-\pi}\right) = \mathbf{X}\boldsymbol{\beta}.$$

The covariates will be obtained by using a two-by-two factorial designed cell means model with a cell for each treatment combination as shown in table 13.1.

The analysis for this example, however, will also deal with replicates in the treatment combinations and as such is an extension of the mixed model. Two models will be presented and compared using the Deviance Information Criteria (DIC). The first model will simply model the binomial probability with the logit transform. The second model will extend this model to include the added variability of the replicates in each treatment combination.

The extra variability that might exist in this data set may or may not be adequately modeled with the added variance term. If this were a linear regression model, the σ^2 term typically accounts for the amount of noise in the data. The question in this setting is can this noise be sufficiently captured in the binomial likelihood or should an error term be added to the model to account for the added variability explicitly? Calculating the DIC values and comparing them will answer this question.

The data will be modeled with a binomial likelihood and priors as shown below. Additionally, the logit transformation links the binomial π with the parameters. The first

model is

$$\begin{aligned}
\pi &\sim \text{Binomial(n, p)} \\
\text{logit}(p) &= \mathbf{X}\boldsymbol{\beta} \\
\boldsymbol{\beta} &\sim \text{Normal(0, 1)},
\end{aligned}$$

and the second model is

$$\begin{aligned}
\pi &\sim \text{Binomial(n, p)} \\
\text{logit}(p) &= \mathbf{X}\boldsymbol{\beta} + e \\
\boldsymbol{\beta} &\sim \text{Normal(0, 1)}, \\
e &\sim \text{Normal}(0, \sigma^2) \\
\sigma^2 &\sim \text{Uniform}(0, 1).
\end{aligned}$$

Equations for the likelihood, prior, and posterior distributions are omitted here where they were provided in Chapter 3 because the MCMC algorithms do not require finding the functional form of the posterior distribution. All that is required is the likelihood function and the distribution for all parameters in the model. The MCMC algorithms calculate the posterior distribution from there.

Table 13.1: A two-by-two factorial cell means model.

	Bean	Cuc
a75	$\pi_{1,1}$	$\pi_{1,2}$
a73	$\pi_{2,1}$	$\pi_{2,2}$

$\Leftrightarrow$

	Bean	Cuc
a75	π_1	π_2
a73	π_3	π_4

13.1 WinBUGS

The data come from an experiment monitoring germination rates of seed varieties and seed types with the goal of estimating the proportion of seeds that will germinate in each treat-

ment combination. It includes an identifier for seed variety, a75 or a73; an identifier for seed type, bean or cuc; the number of seeds that germinated on a particular plate; and the number of seeds that were on the plate initially. Since the first two columns were character value entries and WinBUGS is not able to handle this type of data, a fifth column was added to the data to identify treatment combination membership of which there are four. Before reading the data into WinBUGS, these two character columns must be omitted to prevent errors.

An interesting feature of this data set is that each treatment combination was replicated five or six times for a total of twenty-one observations. When the data set is structured like this, it is often helpful to draw a table of the experimental design, as is shown in table 13.1, for use as a bookkeeping tool to keep track of treatment combination membership and linking this correctly with the corresponding probability.

The first model shown in the code is set up to predict the four binomial probabilities, one for each treatment combination while ignoring the replicates in the data. The second model takes into account the extra variability of the replicates by adding an error term to the logit equation and hierarchically placing priors on its variance parameter. Model comparison via DIC will determine which model sufficiently captures all of the variability here.

```
#Model 1
model{
    for(i in 1:21){
    # likelihood -- note that WinBUGS requires p first for dbin()
    r[i] ~ dbin(p[i], n[i]);
    # logit transformation to preserve parameter space of p
    logit(p[i]) <- b[tmt[i]];
    }
    # priors for each b
    for(i in 1:4){
    b[i] ~ dnorm(0, 1);
    }
    }
#Model 2
model{
    for(i in 1:21){
    # likelihood -- note that WinBUGS requires p first for dbin()
```

```
r[i] ~ dbin(p[i], n[i]);
# logit transformation to preserve parameter space of p
logit(p[i]) <- b[tmt[i]] + e[i];
}
# priors for each b
for(i in 1:4){
b[i] ~ dnorm(0,1);
}
# priors for random error term
for(i in 1:21){
e[i] ~ dnorm(0, prec);
}
# hyperprior for variance and also adjusting variance in terms of
   precision
s2 ~ dunif(0,1);
prec <- 1/s2;
}
```

Please note that the parameterization of the binomial likelihood in WinBUGS takes p first and n second. It is crucial that the researcher become aware of the distributional definitions WinBUGS is programmed with along with those SAS is programmed with. Their parameterizations are not always equivalent and adjustments need to be made as needed.

The evaluation of model fit between the two models compares the two deviance information criteria values with the lower number indicating the model with the better fit. The first model gives a DIC value of 115.4 while the second model gives a DIC value of 111.7. Thus the second model with the term accounting for added variability among the replicates fits the data more accurately. This demonstrates that mixed models indeed are a powerful tool in data analysis as a researcher searches for a model that best fits the data.

The summary statistics are shown in table 13.2, giving posterior summaries of the four β parameters and the twenty-one π parameters from model one. Figure 13.1 gives a sample of the posterior summary plots, showing the posterior distribution of β_1 from model one. The plots indicate that convergence was reached and that there were no problems with autocorrelation.

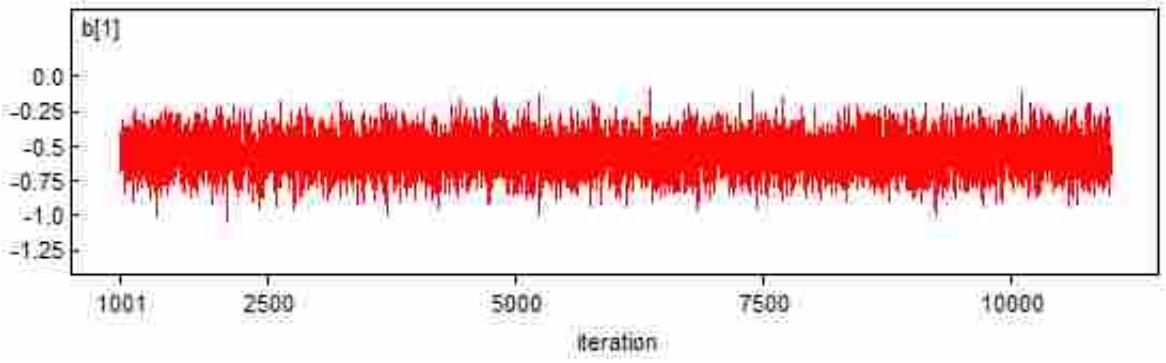
Table 13.2: Summary statistics from model 1 in WinBUGS.

	mean	sd	2.5%	25%	50%	75%	97.5%
b[1]	-0.55	0.12	-0.80	-0.64	-0.55	-0.47	-0.30
b[2]	0.80	0.12	0.56	0.71	0.80	0.88	1.05
b[3]	-0.40	0.18	-0.76	-0.52	-0.40	-0.28	-0.04
b[4]	0.13	0.17	-0.19	0.01	0.12	0.24	0.45
p[1]	0.37	0.03	0.31	0.35	0.37	0.38	0.42
p[2]	0.37	0.03	0.31	0.35	0.37	0.38	0.42
p[3]	0.37	0.03	0.31	0.35	0.37	0.38	0.42
p[4]	0.37	0.03	0.31	0.35	0.37	0.38	0.42
p[5]	0.37	0.03	0.31	0.35	0.37	0.38	0.42
p[6]	0.69	0.03	0.64	0.67	0.69	0.71	0.74
p[7]	0.69	0.03	0.64	0.67	0.69	0.71	0.74
p[8]	0.69	0.03	0.64	0.67	0.69	0.71	0.74
p[9]	0.69	0.03	0.64	0.67	0.69	0.71	0.74
p[10]	0.69	0.03	0.64	0.67	0.69	0.71	0.74
p[11]	0.69	0.03	0.64	0.67	0.69	0.71	0.74
p[12]	0.40	0.04	0.32	0.37	0.40	0.43	0.49
p[13]	0.40	0.04	0.32	0.37	0.40	0.43	0.49
p[14]	0.40	0.04	0.32	0.37	0.40	0.43	0.49
p[15]	0.40	0.04	0.32	0.37	0.40	0.43	0.49
p[16]	0.40	0.04	0.32	0.37	0.40	0.43	0.49
p[17]	0.53	0.04	0.45	0.50	0.53	0.56	0.61
p[18]	0.53	0.04	0.45	0.50	0.53	0.56	0.61
p[19]	0.53	0.04	0.45	0.50	0.53	0.56	0.61
p[20]	0.53	0.04	0.45	0.50	0.53	0.56	0.61
p[21]	0.53	0.04	0.45	0.50	0.53	0.56	0.61
deviance	111.52	2.79	108.10	109.40	110.90	112.90	118.40

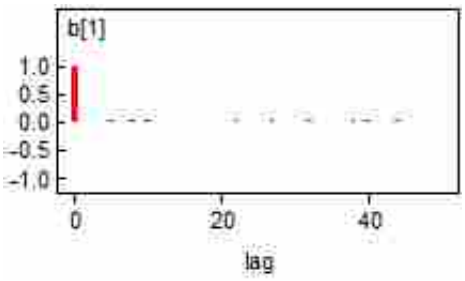
A useful mathematical property of the logit transform is that, like in logistic regression, one can convert each of the β's to estimates of the binomial probabilities π_j as

$$\pi_j = \frac{1}{1 + e^{-\beta_j}}.$$

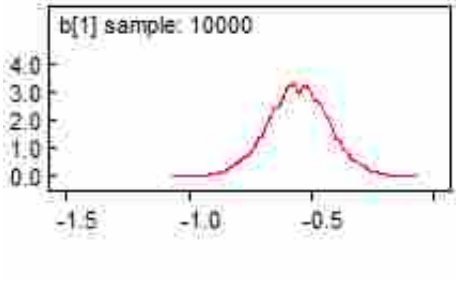
Thus, β_1 gives a binomial probability $\pi_1 = 0.3659$. Notice that the first five p estimates shown in table 13.2 are equivalent. The reason for this is that for each treatment combination, there are five or six replicates in the data but only one probability estimate. Each of the other three π_j estimates can be transformed in like manner.



(a) Trace plot



(b) Autocorrelation



(c) Posterior density

Figure 13.1: WinBUGS summary plots for the posterior distribution of β_1 from model 1.

13.2 PROC MCMC

As was coded in WinBUGS, two models are presented here for SAS to run. Notice how these models are very similar to the models for logistic regression. However, before reading in the data file to SAS, a column with indicator values from 1 to 21 that refer to the observation numbers was added to the data. This column was needed to include the error term in the second model. The MCMC procedures begin on lines thirteen and twenty-three. The second model needed more thinning than the first model to reduce autocorrelation and reach convergence satisfactorily. Notice that these two models have the logit transform in lines eighteen and thirty-three with model two adding the error term. Model two also has priors on this error term and its variance parameter. The likelihood statements are in lines nineteen and thirty-four.

```
* read in the data file;
1  data seeds;
2       infile ''     ''   firstobs=2;
3       input seed $ type $ r n tmt observ;
4  run;
5
* print the data file for inspection;
6  proc print;
7  run;
8
* initializes saving of output as a pdf file;
9  ods pdf
10      file=''     '' ;
* turn on graphics device;
11   ods graphics on;
12   *Model 1;
13   proc mcmc data=seeds outpost=seedsout nmc=500000 thin=50 nbi=10000
   monitor=(b pi) dic seed=1234;
* define arrays of length 4;
14       array b[4];
15       array pi[4];
* set parameter and initial value;
* the colon indicates that the initial value be applied to all array
   entries;
16       parms b: 0;
* define prior;
* the colon indicates that the distribution be applied to all array
   entries;
17       prior b: ~ normal(0, var=1);
* logit transformation equation to preserve parameter space of p;
18       pi[tmt] = logistic(b[tmt]);
* likelihood;
19       model r ~ binomial(n=n, p=pi[tmt]);
20   run;
21
22   *Model 2;
23   proc mcmc data=seeds outpost=seedsout nmc=5000000 thin=500 nbi
   =10000 monitor=(b pi) dic seed=1234;
* define arrays of length 4 and 21;
24       array b[4];
25       array pi[4];
26       array e[21];
* set parameters and initial values;
* the colon indicates that the initial values be applied to all array
   entries;
27       parms b:0;
28       parms e: 0;
```

```
29      parms s2 .5;
* define prior;
* the colon indicates that the distribution be applied to all array
   entries;
30      prior b: ~ normal(0, var=1);
31      prior e: ~ normal(0, var=s2);
32      prior s2 ~ uniform(0, 1);
* logit transformation equation to preserve parameter space of p;
33      pi[tmt] = logistic(b[tmt] + e[observ]);
* likelihood;
34      model r ~ binomial(n=n, p=pi[tmt]);
35  run;
36
* turn off graphics device;
37  ods graphics off;
* stop saving output file;
38  ods pdf close;
```

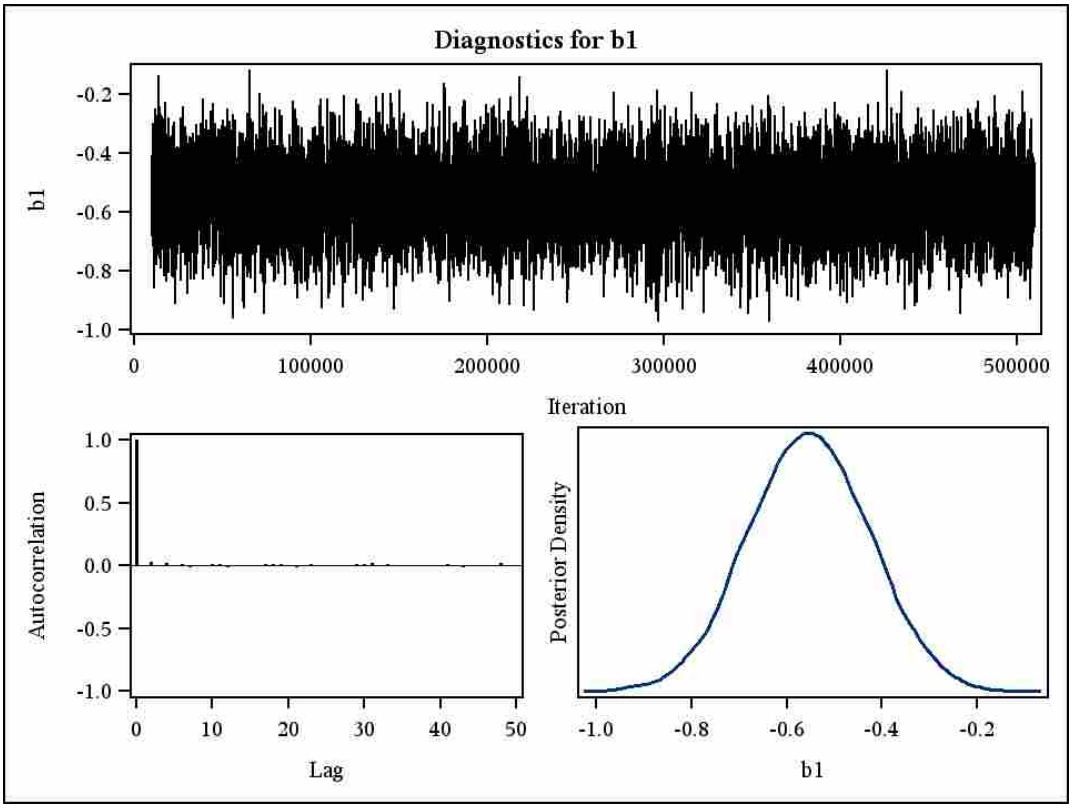
Table 13.3: Summary Statistics for model 1 from PROC MCMC.

Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
b1	10000	-0.5529	0.1256	-0.6380	-0.5530	-0.4671
b2	10000	0.7976	0.1238	0.7142	0.7973	0.8794
b3	10000	-0.4036	0.1814	-0.5247	-0.4022	-0.2793
b4	10000	0.1234	0.1661	0.0112	0.1226	0.2324

The summary statistics are given in table 13.3 for model one, showing posterior summaries of the four β parameters. Figure 13.2 gives the posterior plots for the distribution of β_1 from model one. These plots indicate that convergence was reached, no autocorrelation problems exist, and the density of the posterior is drawn. The DIC values calculated in SAS also indicate that model two is the better fitting model for this data.

Even though the summaries do not give the specific π_j values, they can be calculated from the β_j values as was discussed in the WinBUGS section. The logit transform of these

Figure 13.2: Summary plots for the posterior distribution of β_1 from model 1.



SAS values gives very similar π_j values as calculated from the WinBUGS' output, once again showing that WinBUGS and SAS produce very similar results in the calculation of the posterior distributions of the parameters and demonstrating that the MCMC algorithms indeed converge in distribution to the desired posterior distribution of the conditional probability of the parameters given the data.

13.3 Side by Side Computer Code

WinBUGS code:

```
#Model 1
model{
    for(i in 1:21){
    r[i] ~ dbin(p[i], n[i]);
    logit(p[i]) <- b[tmt[i]];
    }
    for(i in 1:4){
    b[i] ~ dnorm(0, 1);
    }
    }
#Model 2
model{
    for(i in 1:21){
    r[i] ~ dbin(p[i], n[i]);
    logit(p[i]) <- b[tmt[i]] + e[i
       ];
    }
    for(i in 1:4){
    b[i] ~ dnorm(0,1);
    }
    for(i in 1:21){
    e[i] ~ dnorm(0, prec);
    }
```

SAS code:

```
data seeds;
    infile ''    '' firstobs=2;
    input seed $ type $ r n tmt
       observ;
run;

proc print;
run;

ods pdf
    file='' '';
ods graphics on;
*Model 1;
proc mcmc data=seeds outpost=
   seedsout nmc=500000 thin=50 nbi
   =10000 monitor=(b pi) dic seed
   =1234;
     array b[4];
     array pi[4];
     parms b: 0;
     prior b: ~ normal(0, var=1);
     pi[tmt] = logistic(b[tmt]);
     model r ~ binomial(n=n, p=pi[
```

```
    s2 ~ dunif(0,1);
    prec <- 1/s2;
    }
#The data set:
seed[] type[]  r[]  n[] tmt[]
a75 bean 10 39   1
a75 bean 23 62   1
a75 bean 23 81   1
a75 bean 26 51   1
a75 bean 17 39   1
a75  cuc  5  6   2
a75  cuc 53 74   2
a75  cuc 55 72   2
a75  cuc 32 51   2
a75  cuc 49 79   2
a75  cuc 10 13   2
a73 bean  8 16   3
a73 bean 10 30   3
a73 bean  8 28   3
a73 bean 23 45   3
a73 bean  0  4   3
a73  cuc  3 12   4
a73  cuc 22 41   4
a73  cuc 15 30   4
a73  cuc 32 51   4
a73  cuc  3  7   4
END{};
```

```
        tmt]);
run;

*Model 2;
proc mcmc data=seeds outpost=
   seedsout nmc=5000000 thin=500
   nbi=10000 monitor=(b pi) dic
   seed=1234;
     array b[4];
     array pi[4];
     array e[21];
     parms b:0;
     parms e: 0;
     parms s2 .5;
     prior b: ~ normal(0, var=1);
     prior e: ~ normal(0, var=s2);
     prior s2 ~ uniform(0, 1);
     pi[tmt] = logistic(b[tmt] + e
        [observ]);
     model r ~ binomial(n=n, p=pi[
        tmt]);
run;

ods graphics off;
ods pdf close;
```

CHAPTER 14

POISSON MODEL

When the quantity of interest is the number of occurrences of an event over a given interval, the Poisson distribution is the distribution of choice to model the probability of these rates. The number of occurrences is a discrete count and the interval could be measured in time, distance, area, or volume, among others.

Three such situations where the Poisson distribution is appropriate are the number of pumps that fail at time t, the number of customers to arrive at a checkout stand at time t, or the number of bombs that hit in an area a. Figure 14.1 plots the data set for example 12.

The basic Poisson model shown below can be expanded to account for more complicated situations as needed to accommodate the design of the experiment and accompanying research questions. Five models will be presented in this chapter to demonstrate this flexibility.

$$\begin{aligned} y_i &\sim \text{Poisson}(\lambda_i) \\ \lambda_i &= \theta\lambda_i \\ \theta &\sim \text{Gamma}(\alpha, \beta) \end{aligned}$$

Equations for the likelihood, prior, and posterior distributions are omitted here where they were provided in Chapter 3 because the MCMC algorithms do not require finding the functional form of the posterior distribution. All that is required is the likelihood function and the distribution for all parameters in the model. The MCMC algorithms calculate the posterior distribution from there.

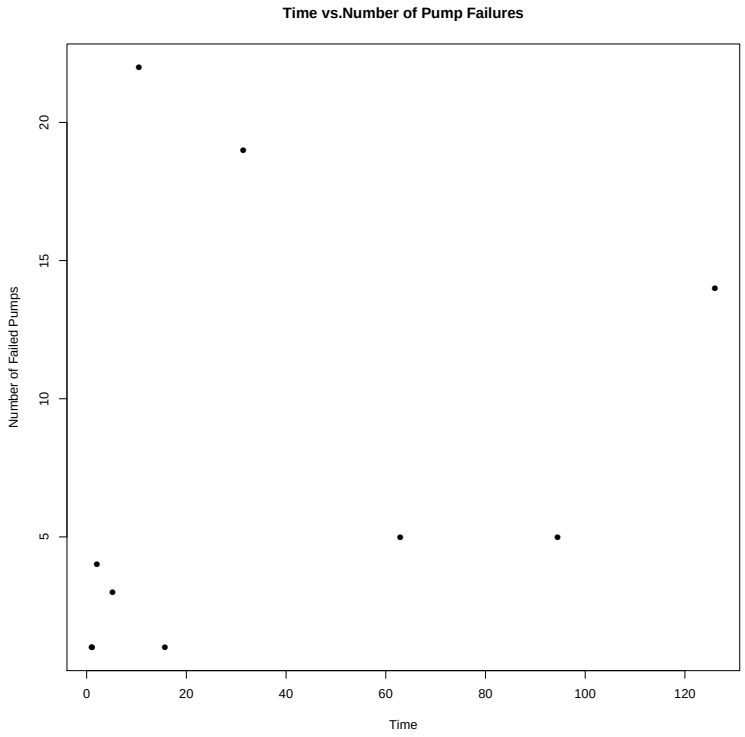


Figure 14.1: Graph of pump failure data.

14.1 WinBUGS

The data come from an experiment monitoring the number of pumps that fail at time t. The Poisson model is appropriate because the data are counts. Figure 14.1 shows the relationship between these two variables. Five different models are given below with their accompanying deviance information criteria (DIC) values displayed in table 14.1 which will be used to make a decision about model selection.

```
#Model 1 is basic Poisson model
model{
   # prior for theta
   theta ~ dgamma(1.5, 1);
   for(i in 1:10){
   # likelihood
   fail[i] ~ dpois(lambda[i]);
   # link function relating lambda, theta, and the covariate time
   lambda[i] <- theta*time[i];
   }
   }
```

```
#Model 2 puts a hierarchy on the parameters of alpha and beta
model{
   # prior for theta
   theta ~ dgamma(alpha, beta);
   for(i in 1:10){
   # likelihood
   fail[i] ~ dpois(lambda[i]);
   # link function relating lambda, theta, and the covariate time
   lambda[i] <- theta*time[i];
   }
   # hyperpriors for theta
   alpha ~ dexp(.1);
   beta ~ dgamma(5, .5);
   }

#Model 3 allows theta to vary with each i
model{
   for(i in 1:10){
   # prior for theta
   theta[i] ~ dgamma(alpha, beta);
   # likelihood
   fail[i] ~ dpois(lambda[i]);
   # link function relating lambda, theta, and the covariate time
   lambda[i] <- theta[i]*time[i];
   }
   # hyperpriors for theta
   alpha ~ dexp(.1);
   beta ~ dgamma(5, .5);
   }

#Model 4 adds an error term as for mixed models
      model{
      for(i in 1:10){
      # prior for theta
      theta[i] ~ dgamma(alpha, beta);
      # likelihood
      fail[i] ~ dpois(lambda[i]);
      # link function relating lambda, theta, the covariate time, and
        random effect
      lambda[i] <- theta[i]*time[i] + u[i];
      # prior for random effect
      u[i] ~ dexp(1);
      }
      # hyperpriors for theta
      alpha ~ dexp(.1);
      beta ~ dgamma(5, .5);
```

```
    }

#Model 5 keeps the error term but models one theta
model{
   # prior for theta
   theta ~ dgamma(alpha, beta);
   for(i in 1:10){
   # likelihood
   fail[i] ~ dpois(lambda[i]);
   # link function relating lambda, theta, the covariate time, and
      random effect
   lambda[i] <- theta*time[i] + u[i];
   # prior for random effect
   u[i] ~ dexp(1);
   }
   # hyperpriors for theta
   alpha ~ dexp(.1);
   beta ~ dgamma(5, .5);
   }
```

Model one is the basic Poisson model. Model two puts hyperpriors on the parameters of α and β. Model three keeps the hierarchical structure of model two while also allowing θ to vary with each time t. Model four builds on model three by adding a term to the λ equation in an effort to account for additional variability that may be present as was done for mixed models. Model five takes model four and changes θ to one occurrence. As can be seen in table 14.1, models three and four are the best fitting models because they have the two lowest DIC values. There is a rather large drop in DIC from models one and two to models three and four. Although allowing θ to vary with each time t appears to be the right way to model this parameter, is the extra variability term adding information to the model?

This answer is a judgement call by the researcher. We would conclude that since the two models' DIC values are so close, the slightly better fit from the extra variability term does not add enough information to balance the fact that this model has an additional ten error parameters. Thus, model three appears to be the model of choice because it is simpler than model four. This same conclusion is reached when looking at the DIC values as calculated by SAS® 9.2.

The summary statistics are shown in table 14.2, giving the posterior summaries for model three. Figure 14.2 gives a sample of the posterior summary plots, showing the posterior distribution of the α parameter in model 3. The plots indicate that convergence was reached and that there were no problems with autocorrelation.

Table 14.1: Table of DIC for each model.

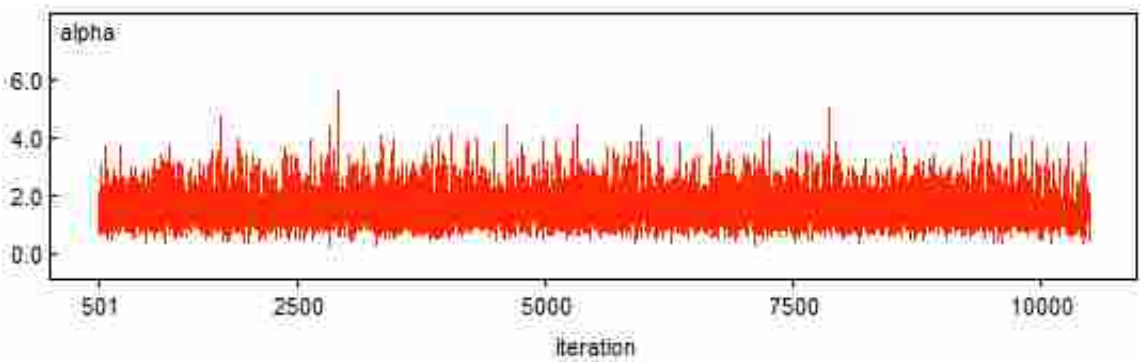
	Model 1	Model 2	Model 3	Model 4	Model 5
WinBUGS	160.06	160.00	52.96	51.50	63.61
SAS	160.04	159.95	53.08	51.53	63.12

Table 14.2: Summary statistics from model 3 in WinBUGS.

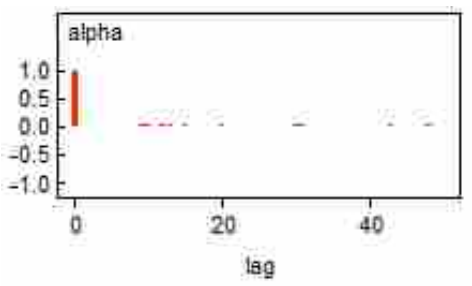
	mean	sd	2.5%	25%	50%	75%	97.5%
theta[1]	0.07	0.03	0.03	0.05	0.06	0.08	0.13
theta[2]	0.13	0.09	0.02	0.07	0.12	0.18	0.34
theta[3]	0.10	0.04	0.04	0.07	0.09	0.12	0.19
theta[4]	0.12	0.03	0.07	0.10	0.12	0.14	0.19
theta[5]	0.52	0.25	0.15	0.34	0.48	0.66	1.13
theta[6]	0.59	0.13	0.36	0.49	0.58	0.67	0.87
theta[7]	0.57	0.39	0.08	0.29	0.49	0.76	1.53
theta[8]	0.57	0.39	0.09	0.29	0.48	0.74	1.52
theta[9]	1.01	0.49	0.32	0.66	0.92	1.26	2.19
theta[10]	1.68	0.38	1.02	1.41	1.65	1.92	2.49
alpha	1.62	0.59	0.73	1.20	1.54	1.95	3.02
beta	3.76	1.56	1.49	2.62	3.50	4.60	7.53
deviance	45.31	5.23	37.33	41.47	44.56	48.30	57.61

14.2 PROC MCMC

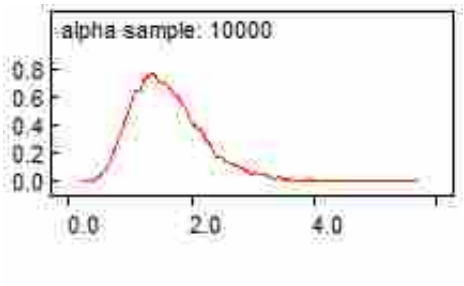
The same five models are presented in SAS code as was done for WinBUGS above. As can be seen in table 14.1, DIC dropped significantly from models one and two to models three and four. Thus models three and four are the better fitting models because they have the two lowest DIC values. Again, allowing θ to vary with each time t appears to be the right way to model this parameter, and we will select model three as the model of choice here because it is the simpler model between models three and four.



(a) Trace plot



(b) Autocorrelation



(c) Posterior density

Figure 14.2: WinBUGS summary plots for the posterior distribution of the α parameter in model 3.

Lines one through four direct SAS to read in the data. However, notice that line three references an additional column in the data set. It is necessary to add a column to the data set prior to reading it in to SAS that indicates the observation number in order to subscript the θ_i's and u_i's in models three, four, and five. This column consists of a sequence from 1 to 10. It is good practice to look over a print out of the data after SAS has read it in, which is what lines six and seven accomplish. Lines nine, ten, and seventy-six are a useful tool for a researcher to capture the output in *.pdf format, but are not necessary to run the analysis.

Lines eleven and seventy-five initialize and close the graphics windows where the plots are sent. Model one is coded in lines thirteen through eighteen; model two is coded in lines twenty-one through thirty; model three is coded in lines thirty-three through forty-three; model four is coded in lines forty-six through fifty-nine; model five is coded in lines sixty-two

through seventy-four. The first line in each model where PROC MCMC is initialized and various options are called, increase the number of MCMC iterations, thinning, and number of burn-in iterations from models two to three and again from models three to four. This increase has the affect of decreasing autocorrelation and aids in the reaching of convergence.

Notice that all five models have the same likelihood statement found in lines seventeen, twenty-nine, forty-two, fifty-eight, and seventy-three. Models three, four, and five need arrays to hold the ten θ_i's and/or the ten u_i's, and consequently, the parms and prior statements for these variables in these models include the use of a colon to indicate that the respective values be applied to each entry in the array. The indicator column in the data set is referenced in models three, four and five to correctly subscript θ_i and/or u_i as can be seen in lines forty-one, fifty-seven, and seventy-two.

As a word of caution to the researcher, it is imperative that one become familiar with the distributional parameterizations in both WinBUGS and SAS® 9.2. The reference manuals for both programs are invaluable in this regard. These two programs do not define the distributions exactly the same way. If the researcher is unaware of the definitions, problems may arise from carelessness.

```
* read in the data file;
1  data pumps;
2       infile ``     '' firstobs=2;
* create indicator column for data file;
3       input time fail ind;
4  run;
5
* print the data file for inspection;
6  proc print;
7  run;
8
* initializes saving of output as a pdf file;
9  ods pdf
10      file = ``     '';
* turn on graphics device;
11 ods graphics on;
12
* Model 1;
```

```
13 proc mcmc data=pumps outpost=pumpsout nmc=10000 thin=1 nbi=1000
   monitor=(_parms_) dic seed=1234;
* set parameter and initial value;
14      parms theta 1.5;
* define prior;
15      prior theta ~ gamma(1.5, iscale=1);
* link function relating lambda, theta, and the covariate time;
16      lambda = theta*time;
* likelihood;
17      model fail ~ poisson(lambda);
18 run;
19
20
*Model 2;
21 proc mcmc data=pumps outpost=pumpsout nmc=10000 thin=1 nbi=1000
   monitor=(_parms_) dic seed=1234;
* set parameters and initial values;
22      parms theta 1.5;
23      parms alpha 1;
24      parms beta 1.5;
* define priors;
25      prior theta ~ gamma(alpha,iscale=beta);
26      prior alpha ~ expon(iscale=.1);
27      prior beta ~ gamma(5, iscale=.5);
* link function relating lambda, theta, and the covariate time;
28      lambda = theta*time;
* likelihood;
29      model fail ~ poisson(lambda);
30 run;
31
32
*Model 3;
33 proc mcmc data=pumps outpost=pumpsout nmc=500000 thin=50 nbi=10000
   monitor=(_parms_) dic seed=1234;
* define array of length 4;
34      array theta[10];
* set parameters and initial values;
* the colon on theta indicates that the initial value be applied to all
    array entries;
35      parms theta: 1.5;
36      parms alpha 2;
37      parms beta 5;
* define priors;
* the colon on theta indicates that the distribution be applied to all
   array entries;
38      prior theta: ~ gamma(alpha, iscale=beta);
39      prior alpha ~ expon(iscale=.1);
```

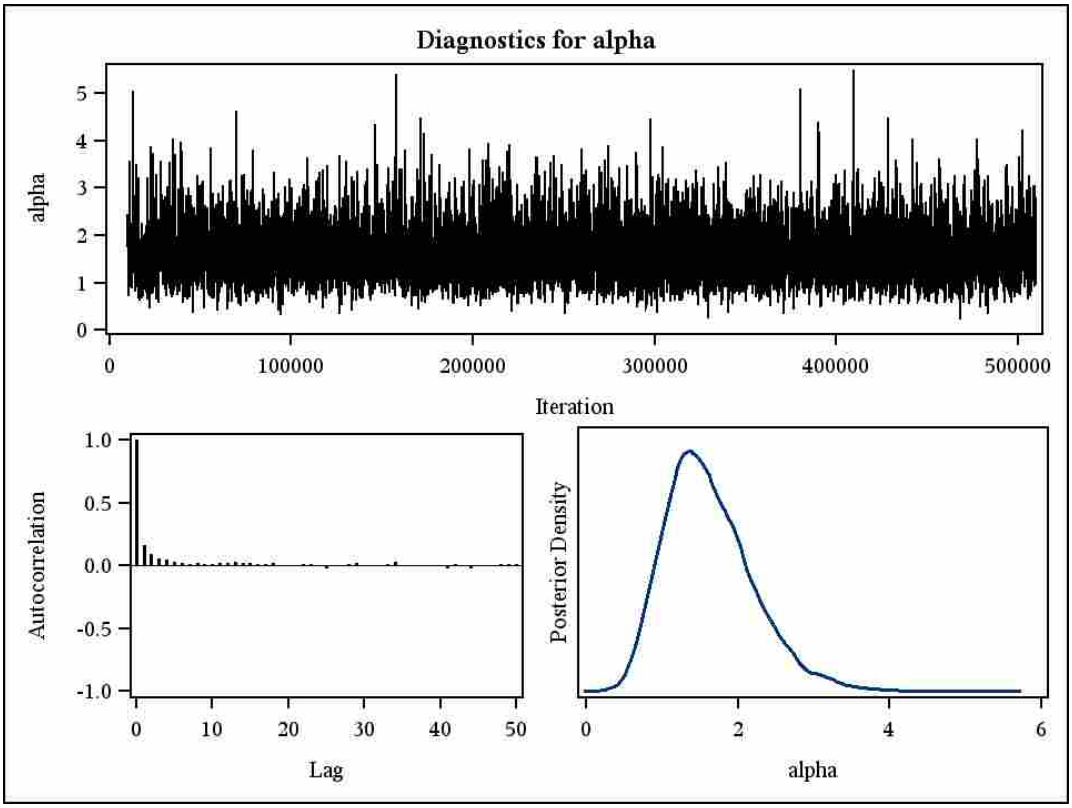
```
40      prior beta ~ gamma(5, iscale=.5);
* link function relating lambda, theta, and the covariate time;
41      lambda = theta[ind]*time;
* likelihood;
42      model fail ~ poisson(lambda);
43 run;
44
45
*Model 4;
46 proc mcmc data=pumps outpost=pumpsout nmc=1000000 thin=100 nbi=10000
    monitor=(theta alpha beta u lambda) dic seed=1234;
* define arrays for theta and random effect;
47      array theta[10];
48      array u[10];
* set parameters and initial values;
* the colon on theta and u indicate that the initial value be applied
   to all array entries;
49      parms theta: 1.5;
50      parms u: 0;
51      parms alpha 2;
52      parms beta 5;
* define priors;
* the colon on theta and u indicate that the distribution be applied to
    all array entries;
53      prior theta: ~ gamma(alpha, iscale=beta);
54      prior alpha ~ expon(iscale=.1);
55      prior beta ~ gamma(5, iscale=.5);
56      prior u: ~ expon(iscale=1);
* link function relating lambda, theta, the random effect and the
   covariate time;
57      lambda = theta[ind]*time + u[ind];
* likelihood;
58      model fail ~ poisson(lambda);
59 run;
60
61
*Model 5;
62 proc mcmc data=pumps outpost=pumpsout nmc=1000000 thin=100 nbi=10000
    monitor=(theta alpha beta u lambda) dic seed=1234;
* define array for random effect;
63      array u[10];
* set parameters and initial values;
* the colon on u indicates that the initial value be applied to all
   array entries;
64      parms theta 1.5;
65      parms u: 0;
66      parms alpha 2;
```

```
67      parms beta 5;
* define priors;
* the colon on u indicates that the distribution be applied to all
   array entries;
68      prior theta ~ gamma(alpha, iscale=beta);
69      prior alpha ~ expon(iscale=.1);
70      prior beta ~ gamma(5, iscale=.5);
71      prior u: ~ expon(iscale=1);
* link function relating lambda, theta, the random effect and the
   covariate time;
72      lambda = theta*time + u[ind];
* likelihood;
73      model fail ~ poisson(lambda);
74 run;
* turn off graphics device;
75 ods graphics off;
* stop saving output file;
76 ods pdf close;
```

Table 14.3: Summary Statistics of Model 3 from PROC MCMC.

Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
theta1	10000	0.0674	0.0268	0.0478	0.0640	0.0829
theta2	10000	0.1346	0.0851	0.0712	0.1178	0.1814
theta3	10000	0.0980	0.0391	0.0696	0.0925	0.1212
theta4	10000	0.1209	0.0314	0.0987	0.1182	0.1407
theta5	10000	0.5188	0.2494	0.3374	0.4783	0.6554
theta6	10000	0.5851	0.1291	0.4932	0.5744	0.6677
theta7	10000	0.5583	0.3800	0.2915	0.4789	0.7304
theta8	10000	0.5714	0.3894	0.2926	0.4823	0.7555
theta9	10000	0.9940	0.4814	0.6587	0.9031	1.2315
theta10	10000	1.6644	0.3758	1.3956	1.6323	1.8976
alpha	10000	1.6273	0.5817	1.2122	1.5501	1.9657
beta	10000	3.8204	1.5617	2.6867	3.5764	4.6706

Figure 14.3: Summary plots for the posterior distribution of the α parameter from model 3.



The summary statistics are given in table 14.3, showing posterior summaries from model three. Figure 14.3 gives the posterior plots for the distribution of the α parameter from model three. These plots indicate that convergence was reached, no autocorrelation problems exist and the density of the posterior is drawn.

14.3 Side by Side Computer Code

WinBUGS Code:

```
#Model 1
model{
    theta ~ dgamma(1.5, 1);
    for(i in 1:10){
    fail[i] ~ dpois(lambda[i]);
    lambda[i] <- theta*time[i];
    }
    }

#Model 2 puts a hierarchy on the
   parameters of alpha and beta
model{
    theta ~ dgamma(alpha, beta);
    for(i in 1:10){
    fail[i] ~ dpois(lambda[i]);
    lambda[i] <- theta*time[i];
    }
    alpha ~ dexp(.1);
    beta ~ dgamma(5, .5);
    }

#Model 3 allows theta to vary
   with each i
```

SAS Code:

```
data pumps;
    infile ''    '' firstobs=2;
    input time fail ind;
run;

proc print;
run;

ods pdf
    file =  ''    '';
ods graphics on;
* Model 1;
proc mcmc data=pumps outpost=
   pumpsout nmc=10000 thin=1 nbi
   =1000 monitor=(_parms_) dic
   seed=1234;
     parms theta 1.5;
     prior theta ~ gamma(1.5,
        iscale=1);
     lambda = theta*time;
     model fail ~ poisson(lambda);
run;
*Model 2;
```

```
model{
    for(i in 1:10){
    theta[i] ~ dgamma(alpha, beta)
       ;
    fail[i] ~ dpois(lambda[i]);
    lambda[i] <- theta[i]*time[i];
    }
    alpha ~ dexp(.1);
    beta ~ dgamma(5, .5);
    }

#Model 4 adds an error term as
   for mixed models
       model{
       for(i in 1:10){
       theta[i] ~ dgamma(alpha,
          beta);
       fail[i] ~ dpois(lambda[i]);
       lambda[i] <- theta[i]*time[
          i] + u[i];
       u[i] ~ dexp(1);
       }
       alpha ~ dexp(.1);
       beta ~ dgamma(5, .5);
       }

#Model 5 keeps the error term but
```

```
proc mcmc data=pumps outpost=
   pumpsout nmc=10000 thin=1 nbi
   =1000 monitor=(_parms_) dic
   seed=1234;
     parms theta 1.5;
     parms alpha 1;
     parms beta 1.5;
     prior theta ~ gamma(alpha,
        iscale=beta);
     prior alpha ~ expon(iscale
        =.1);
     prior beta ~ gamma(5, iscale
        =.5);
     lambda = theta*time;
     model fail ~ poisson(lambda);
run;
*Model 3;
proc mcmc data=pumps outpost=
   pumpsout nmc=500000 thin=50 nbi
   =10000 monitor=(_parms_) dic
   seed=1234;
     array theta[10];
     parms theta: 1.5;
     parms alpha 2;
     parms beta 5;
     prior theta: ~ gamma(alpha,
        iscale=beta);
```

```
    models one theta
model{
    theta ~ dgamma(alpha, beta);
    for(i in 1:10){
    fail[i] ~ dpois(lambda[i]);
    lambda[i] <- theta*time[i] + u
       [i];
    u[i] ~ dexp(1);
    }
    alpha ~ dexp(.1);
    beta ~ dgamma(5, .5);
    }

#The data set:
time[] fail[]
94.5 5
15.7 1
62.9 5
126 14
5.24 3
31.4 19
1.05 1
1.05 1
2.1 4
10.5 22
END{};
```

```
    prior alpha ~ expon(iscale
       =.1);
    prior beta ~ gamma(5, iscale
       =.5);
    lambda = theta[ind]*time;
    model fail ~ poisson(lambda);
run;
*Model 4;
proc mcmc data=pumps outpost=
   pumpsout nmc=1000000 thin=100
   nbi=10000 monitor=(theta alpha
   beta u lambda) dic seed=1234;
    array theta[10];
    array u[10];
    parms theta: 1.5;
    parms u: 0;
    parms alpha 2;
    parms beta 5;
    prior theta: ~ gamma(alpha,
       iscale=beta);
    prior alpha ~ expon(iscale
       =.1);
    prior beta ~ gamma(5, iscale
       =.5);
    prior u: ~ expon(iscale=1);
    lambda = theta[ind]*time + u[
       ind];
```

```
    model fail ~ poisson(lambda);
run;
*Model 5;
proc mcmc data=pumps outpost=
   pumpsout nmc=1000000 thin=100
   nbi=10000 monitor=(theta alpha
   beta u lambda) dic seed=1234;
    array u[10];
    parms theta 1.5;
    parms u: 0;
    parms alpha 2;
    parms beta 5;
    prior theta ~ gamma(alpha,
       iscale=beta);
    prior alpha ~ expon(iscale
       =.1);
    prior beta ~ gamma(5, iscale
       =.5);
    prior u: ~ expon(iscale=1);
    lambda = theta*time + u[ind];
    model fail ~ poisson(lambda);
run;
ods graphics off;
ods pdf close;
```

CHAPTER 15

POISSON REGRESSION

There are some settings when the response variable is a count and the researcher is interested in how this count changes as the explanatory variable increases. One such setting is in pharmaceutical studies of how the response variable changes as the dose is increased. The tool for analyzing this situation is Poisson Regression.

The likelihood for the data is Poisson and the mean outcome, the λ, is considered log-linear in the coefficients. It is typical to transform the response and explanatory variable(s) to the log-scale because this transformation allows the model to work along the real line, while keeping the outcome in its correct space. The log of the mean will be modeled and then exponentiated for interpretability of the results,

$$\begin{aligned} \mathbf{Y} \sim & \text{Poisson}(\lambda) \\ \log(\lambda) = & \mathbf{X}\boldsymbol{\beta} \\ \lambda = & e^{\mathbf{X}\boldsymbol{\beta}}. \end{aligned}$$

The analysis here is designed to model how the outcome changes as the explanatory variable(s) increase. The graph of the data shown in figure 15.1 shows how the log-response relates to the log-dose for this example's data.

The setting here consists of counting the number of colonies that grow on a particular plate that has been exposed to a specific treatment dose. Since the response variable consists of counts, it is reasonable to model these with a Poisson likelihood. The mean response, λ, will undergo a log-linear transformation and the $\boldsymbol{\beta}$'s will be given priors.

An interesting feature of this particular data set, however, is the fact that there are replicates for each of the six doses. The inclusion of these replicates comes from the researcher thoughtfully designing the experiment so that the variability due to measurement

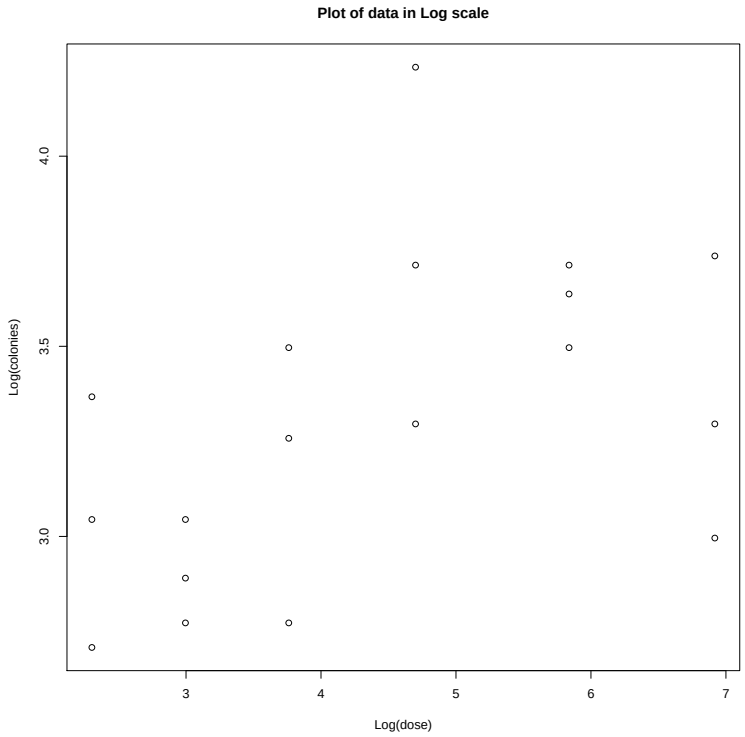


Figure 15.1: Graph of the data on the log scale.

error could be accounted for in the analysis. As such, two models will be presented in the following code, the first model will include a term to model this additional variability and the second model will not. The deviance information criteria, DIC, values will be compared to determine if the Poisson likelihood can model all of the variability here on its own, or if the extra variability term adds to the analysis and improves how the model fits the data.

To answer the question of whether or not the number of colonies that grow on a plate is related to dose amount, the following models will be used to analyze the data. The first

model is

$$\begin{aligned} y_{ij} &\sim \text{Poisson}(\lambda_i) \\ \log(\lambda_i) &= \beta_0 + \beta_1 \log(x+10) + \beta_2 x + u_i \\ \beta_0 &\sim \text{Normal}(0,1) \\ \beta_1 &\sim \text{Normal}(0,1) \\ \beta_2 &\sim \text{Normal}(0,1) \\ u_i &\sim \text{Normal}(0,\sigma^2) \\ \sigma^2 &\sim \text{Uniform}(0,2), \end{aligned}$$

and the second model is

$$\begin{aligned} y_{ij} &\sim \text{Poisson}(\lambda_i) \\ \log(\lambda_i) &= \beta_0 + \beta_1 \log(x+10) + \beta_2 x \\ \beta_0 &\sim \text{Normal}(0,1) \\ \beta_1 &\sim \text{Normal}(0,1) \\ \beta_2 &\sim \text{Normal}(0,1). \end{aligned}$$

The regression coefficients are the β's. The variable x represents the changing dose level; its initial value starts at zero for the control group and then increases. Notice that in $\log(\lambda_i)$'s second term, 10 is added to x inside the log function. This is done mainly because $\log(0)$ is undefined. In an effort to accommodate this limitation of the log function, it is standard practice in this type of pharmaceutical setting to add to x the difference in dose between the control group and the first dose here in this term of the model. The DIC values for each model will be compared to determine which one fits the data better.

Equations for the likelihood, prior, and posterior distributions are omitted here where they were provided in Chapter 3 because the MCMC algorithms do not require finding the

functional form of the posterior distribution. All that is required is the likelihood function and the distribution for all parameters in the model. The MCMC algorithms calculate the posterior distribution from there.

15.1 WinBUGS

The data come from an experiment where different doses of a treatment were applied to plates and the number of colonies that grew as a result were recorded. The question of interest is the relationship between dose amount and number of colonies that grow on a plate. Six different dose amounts (0, 10, 33, 100, 333, and 1000) were chosen and each dose amount was replicated on three plates resulting in eighteen observations. Figure 15.1 shows how the log-response relates to the log-dose in this data set.

Two models are presented in an effort to determine if the Poisson likelihood is able to model all of the variability, or if a term is needed to model the variability from the repeated measurements. The first model includes this random effects term with corresponding prior distribution while the second model leaves these out. Both sets of code begin with a dummy variable for plate because WinBUGS requires that all columns in the data set be referenced and this column is not necessary to run the code.

```
# Model with random effect:
model{
   # dummy variable to use all columns of data set
   dummy1 <- plate[1];
   for(i in 1:18){
   # likelihood
   colonies[i] ~ dpois(lambda[i]);
   # log transformation of mean is linear in the coefficients
   log(lambda[i]) <- a + b*log(dose[i] + 10) + c*dose[i] + u[i];
   # the random effect prior
   u[i] ~ dnorm(0, precd);
   }
   # priors for the beta coefficients
   a ~ dnorm(0, 1);
   b ~ dnorm(0, 1);
   c ~ dnorm(0, 1);
```

```
    # hyper prior for u's variance and adjusting it in terms of
       precision
    s2d ~ dunif(0, 2);
    precd <- 1/s2d;
    }

# Model without random effect:
model{
    # dummy variable to use all columns of data set
    dummy1 <- plate[1];
    for(i in 1:18){
    # likelihood
    colonies[i] ~ dpois(lambda[i]);
    # log transformation of mean is linear in the coefficients
    log(lambda[i]) <- a + b*log(dose[i] + 10) + c*dose[i];
    }
    # priors for the beta coefficients
    a ~ dnorm(0, 1);
    b ~ dnorm(0, 1);
    c ~ dnorm(0, 1);
    }
```

Deviance information criteria (DIC) values were calculated from both models and can be used to determine which model fits the data better. Since the lower DIC indicates better fit and model one's DIC of 124.211 is lower than model two's of 152.814, we conclude that the added term to model the extra variability should be included in the analysis. Therefore, model one is the model of choice for this data set.

The summary statistics are shown in table 15.1, giving the posterior summaries for model one. Figure 15.2 gives a sample of the posterior summary plots, showing the posterior distribution of the intercept from model one. The plots indicate that convergence was reached and that there were no problems with autocorrelation.

15.2 PROC MCMC

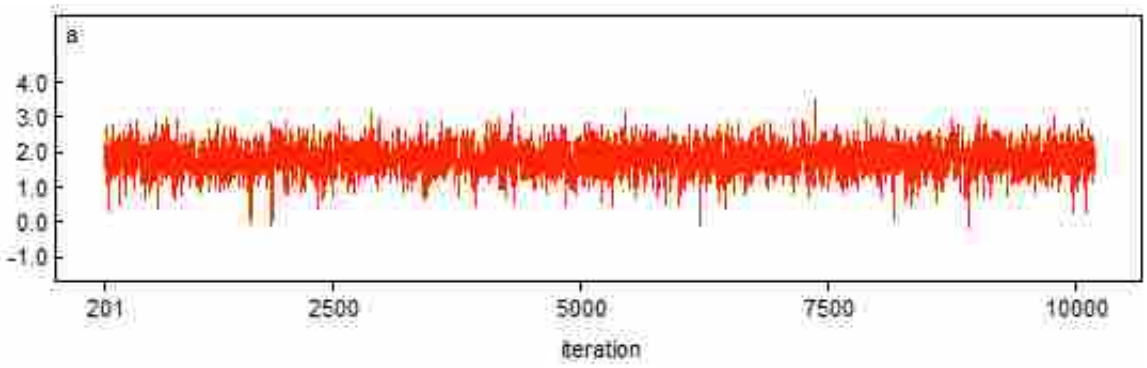
The same two models are presented in SAS code as was done for WinBUGS above. Lines one through four direct SAS to read in the data, and the use of lines six through seven invoke the good practice of looking over a print out of the data after SAS has read it in to verify

Table 15.1: Summary statistics of the first model from WinBUGS.

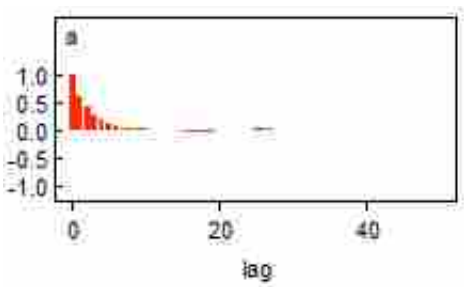
	mean	sd	2.5%	25%	50%	75%	97.5%
a	1.81	0.40	0.98	1.56	1.82	2.08	2.56
b	0.41	0.11	0.20	0.33	0.40	0.48	0.63
c	-0.00	0.00	-0.00	-0.00	-0.00	-0.00	-0.00
s2d	0.13	0.08	0.04	0.08	0.11	0.16	0.33
u[1]	-0.04	0.23	-0.51	-0.19	-0.04	0.11	0.42
u[2]	0.20	0.23	-0.24	0.04	0.19	0.35	0.66
u[3]	0.45	0.23	0.03	0.29	0.44	0.60	0.93
u[4]	-0.17	0.21	-0.61	-0.30	-0.16	-0.02	0.24
u[5]	-0.10	0.21	-0.51	-0.23	-0.09	0.04	0.31
u[6]	0.01	0.21	-0.40	-0.13	0.01	0.14	0.42
u[7]	-0.36	0.21	-0.80	-0.50	-0.35	-0.22	0.04
u[8]	-0.04	0.19	-0.42	-0.17	-0.04	0.08	0.34
u[9]	0.14	0.18	-0.21	0.02	0.14	0.26	0.51
u[10]	-0.23	0.20	-0.65	-0.36	-0.22	-0.10	0.14
u[11]	0.09	0.18	-0.28	-0.03	0.09	0.21	0.45
u[12]	0.55	0.17	0.21	0.43	0.55	0.66	0.89
u[13]	-0.21	0.20	-0.62	-0.34	-0.20	-0.07	0.16
u[14]	-0.10	0.19	-0.49	-0.23	-0.10	0.03	0.26
u[15]	-0.04	0.19	-0.42	-0.16	-0.04	0.09	0.33
u[16]	-0.23	0.26	-0.75	-0.39	-0.23	-0.07	0.26
u[17]	-0.02	0.25	-0.51	-0.18	-0.03	0.13	0.46
u[18]	0.33	0.25	-0.12	0.17	0.32	0.49	0.84
deviance	109.92	5.72	100.60	105.80	109.30	113.40	122.70

this was as expected. Lines nine, ten and thirty-eight are a useful tool for a researcher to capture the output in *.pdf format, but are not necessary to run the analysis.

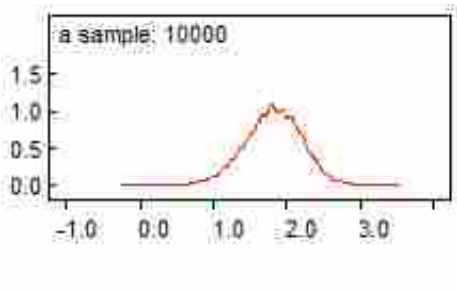
Lines eleven and thirty-seven initialize and close the graphics windows where the plots are sent. Model one is coded in lines thirteen through twenty-five and model two is coded in lines twenty-eight through thirty-six. Notice the large number of MCMC iterations, thinning, and number of burn-in iterations that are called for in lines thirteen and twenty-eight. These values were increased in an effort to reduce autocorrelation and aid in the reaching of convergence. However, the posterior plots indicate some autocorrelation still exists as can be seen in figure 15.3; the researcher should be mindful of this characteristic when working with the output.



(a) Trace plot



(b) Autocorrelation



(c) Posterior density

Figure 15.2: WinBUGS summary plots for the posterior distribution of the intercept from the first model.

The likelihood statements are given in lines twenty-four and thirty-five. The array statement in lines fourteen through sixteen and lines twenty-nine through thirty initialize arrays of length eighteen where SAS will keep track of values as the analysis progresses. When the parms and prior statements refer to arrays, a colon is included to indicate that the initial values and prior distributions need to be applied to all entries in the array.

```
* read in the data file;
1  data dose;
2       infile ''     '' firstobs=2;
3       input  dose plate colonies;
4  run;
5
* print the data file for inspection;
6  proc print;
7  run;
8
* initializes saving of output as a pdf file;
```

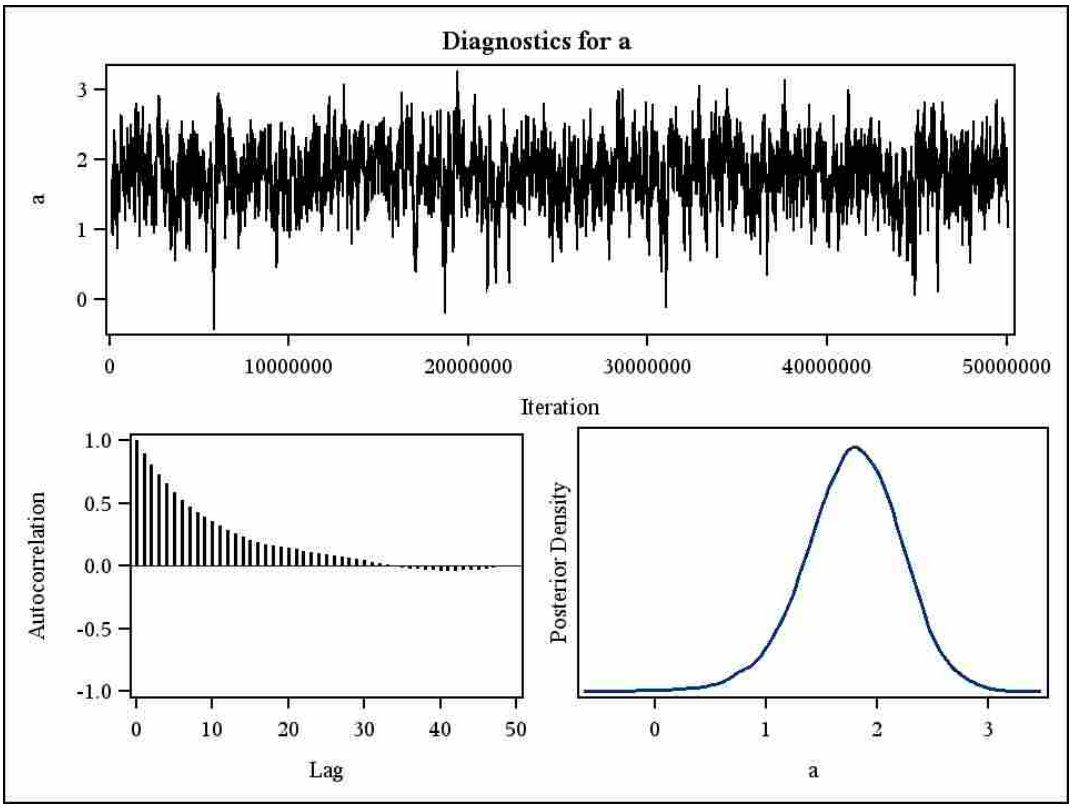
```
9  ods pdf
10       file=''     '';
* turn on graphics device;
11  ods graphics on;
12
* Model with random effect;
13  proc mcmc data=dose outpost=doseout nmc=50000000 thin=5000 nbi
   =100000 monitor=(a b c s2 u) dic seed=1234;
* define arrays for random effect, loglambda, and lambda;
14     array u[18];
15     array llambda[18];
16     array lambda[18];
* set parameters and initial values;
* the colon indicates that the initial value should be applied to all
   array entries;
17     parms u: 0;
18     parms a 0 b 0 c 0 s2 1;
* define priors;
* the colon indicates that the distribution be applied to all array
   entries;
19     prior u: ~ normal(0, var=s2);
20     prior a b c ~ normal(0, var=1);
21     prior s2 ~ uniform(0, 2);
* log transform of the mean is linear in the coefficients;
22     llambda[plate] = a + b*log(dose + 10) + c*dose + u[plate];
* exponentiating will back transform to give the mean;
23     lambda[plate] = exp(llambda[plate]);
* likelihood;
24     model colonies ~ poisson(lambda[plate]);
25  run;
26
27 * Model without random effect;
28 proc mcmc data=dose outpost=doseout nmc=50000000 thin=5000 nbi
   =100000 monitor=(a b c) dic seed=1234;
* define arrays for loglambda and lambda;
29     array llambda[18];
30     array lambda[18];
* set parameters and initial values;
31     parms a 0 b 0 c 0;
define priors;
32     prior a b c ~ normal(0, var=1);
* log transform of the mean is linear in the coefficieints;
33     llambda[plate] = a + b*log(dose + 10) + c*dose;
* exponentiating will back transform to give the mean;
34     lambda[plate] = exp(llambda[plate]);
* likelihood;
35     model colonies ~ poisson(lambda[plate]);
```

```
36  run;
* turn off graphics device;
37  ods graphics off;
* stop saving output file;
38  ods pdf close;
```

Table 15.2: Summary Statistics of the first model in Example 13 from PROC MCMC.

Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
a	10000	1.7754	0.4366	1.5020	1.7927	2.0693
b	10000	0.4177	0.1191	0.3372	0.4129	0.4919
c	10000	-0.00136	0.000539	-0.00169	-0.00133	-0.00100
s2	10000	0.1329	0.0813	0.0800	0.1134	0.1614
u1	10000	-0.0279	0.2420	-0.1869	-0.0279	0.1233
u2	10000	0.2004	0.2398	0.0418	0.1925	0.3489
u3	10000	0.4624	0.2414	0.2992	0.4478	0.6169
u4	10000	-0.1631	0.2175	-0.3026	-0.1579	-0.0204
u5	10000	-0.0891	0.2127	-0.2301	-0.0873	0.0529
u6	10000	0.00838	0.2094	-0.1269	0.00581	0.1465
u7	10000	-0.3632	0.2153	-0.4996	-0.3548	-0.2158
u8	10000	-0.0398	0.1925	-0.1650	-0.0382	0.0863
u9	10000	0.1417	0.1833	0.0208	0.1396	0.2616
u10	10000	-0.2425	0.2025	-0.3708	-0.2339	-0.1064
u11	10000	0.0823	0.1858	-0.0384	0.0809	0.2056
u12	10000	0.5433	0.1755	0.4258	0.5384	0.6591
u13	10000	-0.2179	0.2045	-0.3439	-0.2098	-0.0789
u14	10000	-0.1101	0.1960	-0.2364	-0.1052	0.0225
u15	10000	-0.0499	0.1934	-0.1736	-0.0437	0.0795
u16	10000	-0.2236	0.2525	-0.3811	-0.2145	-0.0606
u17	10000	-0.0166	0.2456	-0.1739	-0.0205	0.1417
u18	10000	0.3377	0.2464	0.1735	0.3289	0.4918

Figure 15.3: Summary plots for the posterior distribution of the intercept from the first model.



SAS utilizes the plate column in the data set where WinBUGS did not. This column is used as an indicator for observation number in lines twenty-two through twenty-four and lines thirty-three through thirty-five. It should be noted that SAS does not require the use of all columns in the data set as WinBUGS does.

The summary statistics are given in table 15.2, showing the posterior summaries from model one. Figure 15.3 gives the posterior plots for the distribution of the intercept from model one. These plots indicate that convergence was reached, but some autocorrelation still exists in the draws. The researcher should be aware of such autocorrelation when using the posterior draws from this analysis. It is interesting to note that WinBUGS' plots indicate

no autocorrelation concerns in its draws even though fewer iterations were required to reach convergence.

15.3 Side by Side Computer Code

WinBUGS code:

```
# Model with random effect:
model{
   dummy1 <- plate[1];
   for(i in 1:18){
   colonies[i] ~ dpois(lambda[i])
      ;
   log(lambda[i]) <- a + b*log(
      dose[i] + 10) + c*dose[i] +
      u[i];
   u[i] ~ dnorm(0, precd);
   }
   a ~ dnorm(0, 1);
   b ~ dnorm(0, 1);
   c ~ dnorm(0, 1);
   s2d ~ dunif(0, 2);
   precd <- 1/s2d;
   }
# Model without random effect:
model{
   dummy1 <- plate[1];
   for(i in 1:18){
```

SAS code:

```
data dose;
    infile ''    '' firstobs=2;
    input  dose plate colonies;
run;

proc print;
run;

ods pdf
    file='';
ods graphics on;

proc mcmc data=dose outpost=
   doseout nmc=50000000 thin=5000
   nbi=100000 monitor=(a b c s2 u)
    dic seed=1234;
  array u[18];
  array llambda[18];
  array lambda[18];
  parms u: 0;
  parms a 0 b 0 c 0 s2 1;
```

```
    colonies[i] ~ dpois(lambda[i])
       ;
    log(lambda[i]) <- a + b*log(
       dose[i] + 10) + c*dose[i];
    }
    a ~ dnorm(0, 1);
    b ~ dnorm(0, 1);
    c ~ dnorm(0, 1);
    }

#The data set:
dose[] plate[] colonies[]
0  1 15
0 2 21
0 3 29
10 4 16
10 5 18
10 6 21
33 7 16
33 8 26
33 9 33
100 10 27
100 11 41
100 12 69
333 13 33
333 14 38
333 15 41
```

```
    prior u: ~ normal(0, var=s2);
    prior a b c ~ normal(0, var=1);
    prior s2 ~ uniform(0, 2);
    llambda[plate] = a + b*log(dose
       + 10) + c*dose + u[plate];
    lambda[plate] = exp(llambda[
       plate]);
    model colonies ~ poisson(lambda
       [plate]);
run;

* Model without random effect;
proc mcmc data=dose outpost=
    doseout nmc=50000000 thin=5000
    nbi=100000 monitor=(a b c) dic
    seed=1234;
    array llambda[18];
    array lambda[18];
    parms a 0 b 0 c 0;
    prior a b c ~ normal(0, var=1);
    llambda[plate] = a + b*log(dose
       + 10) + c*dose;
    lambda[plate] = exp(llambda[
       plate]);
    model colonies ~ poisson(lambda
       [plate]);
run;
```

```
1000 16 20
1000 17 27
1000 18 42
END{};
```

```
ods graphics off;
ods pdf close;
```

CHAPTER 16

SURVIVAL MODEL WITH CENSORING

Some experiments are concluded before every experimental unit has experienced the response, as in a study of the effect of a treatment on survival time of subjects. Not all subjects will live for the duration of the experiment, and not all subjects will have died at the conclusion of the experiment. This type of setting calls for a survival model.

The survival model is interested in the time until a subject experiences the event of interest, i.e., death or failure. However, there are situations where a subject fails to participate through to the conclusion of a study and their response is not able to be observed. Another concern is when a subject has not experienced the event by the conclusion of the study. These subjects should not just be removed from the data set because their responses were not able to be observed. Such observations are said to be censored and they contain valuable information that needs to be considered in the analysis. This characteristic is the main feature of survival analysis, and as such, typical statistical methods do not adequately model these situations. (Collett 2003)

The survival function is defined as the probability that the survival time is greater than or equal to some time t,

$$S(t) = P(T \geq t).$$

This function can be used to represent the probability that a subject will survive from the time of origin to some time beyond t. Typically survival data is modeled with a Weibull or Exponential distribution. However, other distributions may be used; the reader is referred to survival analysis literature for further study on other appropriate distributional models for survival data.

The following analysis in WinBUGS and SAS® 9.2 will demonstrate different models. This is because WinBUGS is able to handle censoring of observations directly, while SAS® 9.2 is not. The resulting posterior distributions are similar, however, despite the different approaches shown below.

16.1 WinBUGS

WinBUGS allows for left, right, and interval censoring of the time to event.

- Right censored data:
  - $y \sim dweib(a, b)I(\text{lower bound},)$.
- Left censored data:
  - $y \sim dweib(a, b)I(, \text{upper bound})$.
- Interval censored data:
  - $y \sim dweib(a, b)I(\text{lower bound, upper bound})$.

This model for WinBUGS is appropriate because our data consist of both uncensored and right censored observations.

$$\begin{aligned} y_{ij} &\sim \text{Weibull}(r, \mu_i)I(c,) \\ \mu_i &= e^{\beta_i} \\ \beta_i &\sim \text{Normal}(0, 100) \\ r &\sim \text{Exponential}(0.1) \end{aligned}$$

The Weibull distribution is used to model the survival function here in WinBUGS because using the Weibull is typical practice for a parametric analysis and obtaining appropriate

summary statistics is not difficult. Here, r is the scale parameter and μ is the shape parameter. A linking function is used to connect the scale parameter with a function of e. It is reasonable to model the scale parameter's β with a normal prior and the shape parameter with an exponential prior.

Equations for the likelihood, prior, and posterior distributions are omitted here where they were provided in Chapter 3 because the MCMC algorithms do not require finding the functional form of the posterior distribution. All that is required is the likelihood function and the distribution for all parameters in the model. The MCMC algorithms calculate the posterior distribution from there.

WinBUGS parameterizes the Weibull as

$$x \sim \text{Weibull}(v, \lambda) = v\lambda x^{(v-1)} e^{-\lambda x^v}, \quad x > 0.$$

In survival analysis, a summary statistic of great interest is median survival time because survival times are typically heavily right skewed. With the above parameterization, the median may be calculated as

$$\left(\frac{ln(2)}{\lambda}\right)^{\frac{1}{v}}.$$

The data for this analysis come from an experiment where mice were placed into four treatment groups and each group was exposed to a different treatment. Their survival time in days was recorded. Not every mouse had died at the conclusion of the study (40 days), however, so these observations were censored. The data set contains four columns: a mouse ID column, an indicator for treatment membership, the observed time to event, and a censoring indicator that is 0 if the observation was not censored and 40 if it was censored.

The code below begins with a dummy variable for mouse ID. It is necessary to utilize every column of the data set in the WinBUGS code to avoid errors. The use of the indicator for censoring in the likelihood tells WinBUGS how to handle those observations that experienced censoring. The rest of the code follows the typical structure of previous models.

```
model{
   # dummy variable to use all columns of data set
   dummy <- mid[i];
   for(i in 1:80){
   # likelihood
   time[i] ~ dweib(r, mu[tmt[i]])I(censored[i],)
   }
   for(i in 1:4){
   # equation to model mu
   mu[i] <- exp(beta[i]);
   # prior for beta
   beta[i] ~ dnorm(0, 0.01);
   }
   # prior for r
   r ~ dexp(0.1);
   }
```

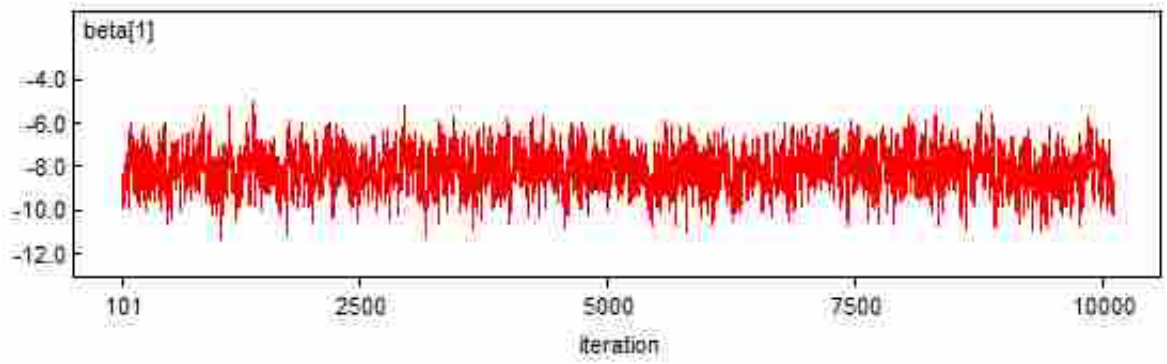
The summary statistics are shown in table 16.1, giving posterior summaries of the four μ's and β's along with the shape parameter r. These values, however, are not very meaningful to a researcher because they are not in the same metric as the data. The transformation of these values into median survival time as described above, however, gives the posterior values in a meaningful metric. These values are shown in table 16.1. Figure 16.1 gives a sample of the posterior summary plots, showing the posterior distribution of β_1. The plots indicate that convergence was reached and that there might be some problems with autocorrelation. Running the analysis again with increased number of burn-in iterations and thinning of the draws will decrease autocorrelation.

16.2 PROC MCMC

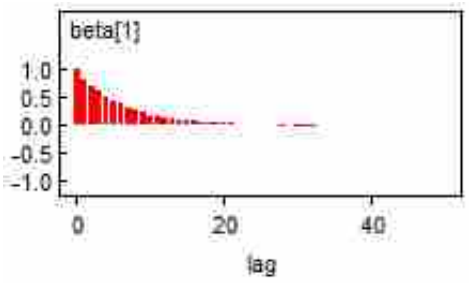
SAS® 9.2 is not able to directly model censored data in a survival model. As such, it is necessary to construct the density function using a combination of the functions LOGPDF, LOGCDF, and LOGSDF depending on how the data is censored. The reader is referred to SAS® 9.2 documentation on PROC MCMC for further study.

Table 16.1: Summary statistics from WinBUGS.

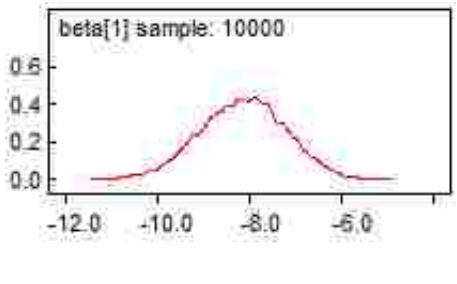
	mean	sd	2.5%	25%	50%	75%	97.5%
mu[1]	0.0004	0.0004	0.0000	0.0002	0.0003	0.0005	0.0016
mu[2]	0.0002	0.0002	0.0000	0.0001	0.0001	0.0002	0.0006
mu[3]	0.0003	0.0003	0.0000	0.0001	0.0002	0.0003	0.0011
mu[4]	0.0004	0.0004	0.0000	0.0001	0.0002	0.0005	0.0014
r	2.4924	0.2617	1.9909	2.3100	2.4860	2.6690	3.0140
beta[1]	-8.1569	0.9117	-9.9680	-8.7770	-8.1340	-7.5290	-6.4360
beta[2]	-9.2129	0.9734	-11.1800	-9.8692	-9.1890	-8.5350	-7.3680
beta[3]	-8.6718	0.9511	-10.5703	-9.3212	-8.6430	-8.0120	-6.8540
beta[4]	-8.3355	0.9238	-10.1800	-8.9552	-8.3240	-7.6967	-6.5610
deviance	528.8494	3.2331	524.5000	526.4000	528.2000	530.6000	536.6000



(a) Trace plot



(b) Autocorrelation



(c) Posterior density

Figure 16.1: WinBUGS summary plots for the posterior distribution of β_1.

Table 16.2: Table showing the posterior mean of each treatment's median survival time as calculated from WinBUGS' analysis.

Tmt1	Tmt2	Tmt3	Tmt4
22.82	35.03	28.11	24.54

The model in SAS® 9.2 is

$$y_{ij} \sim \begin{cases} \text{Normal}(\mu_i, \sigma_i^2) \text{ if uncensored} \\ S(\mu_i) \text{ if right censored} \end{cases}$$

$$\mu_i \sim \text{Normal}(0, 100000)$$

$$\sigma_i^2 \sim \text{Gamma}(2, 50),$$

where $S(\cdot)$ is the survival function, $S(t) = P(T > t)$.

It is necessary that the data file contain a column of lower bound times and a column of upper bound times. The column of left bound times will include all of the observed time to event values and the censored value; for this example these are taken from the time column in the data file with the NA entries replaced by the censored value of 40. The column of right bound times includes the time to event value with NA entries for those observations that were censored; for this example these are equivalent to the time column in the data file.

The MCMC procedure begins on line thirteen. Notice that the number of MCMC iterations is 500,000, thin is 50 and the number of burn-in iterations is 1,000. These values were selected to reduce autocorrelation and aid in convergence to the posterior distribution. A new option that is utilized in this model is the missing=AC option. This must be included in the code so SAS knows that it needs to work with the missing data values instead of ignoring them. This option allows for the modeling of missing values, which is necessary for censoring. An array of length four is initialized for μ and σ^2 in lines fourteen and fifteen with their initial values set in lines sixteen and seventeen and their prior distributions defined in lines eighteen and nineteen. The use of the colon on these last four lines asks that these initial values and prior distributions be applied to all array entries. Lines twenty through twenty-three instruct SAS on the appropriate log-likelihood for uncensored and censored data. The likelihood is defined with the general likelihood in line twenty-four. The reader is referred to the PROC MCMC manual for further study on the use of the general likelihood.

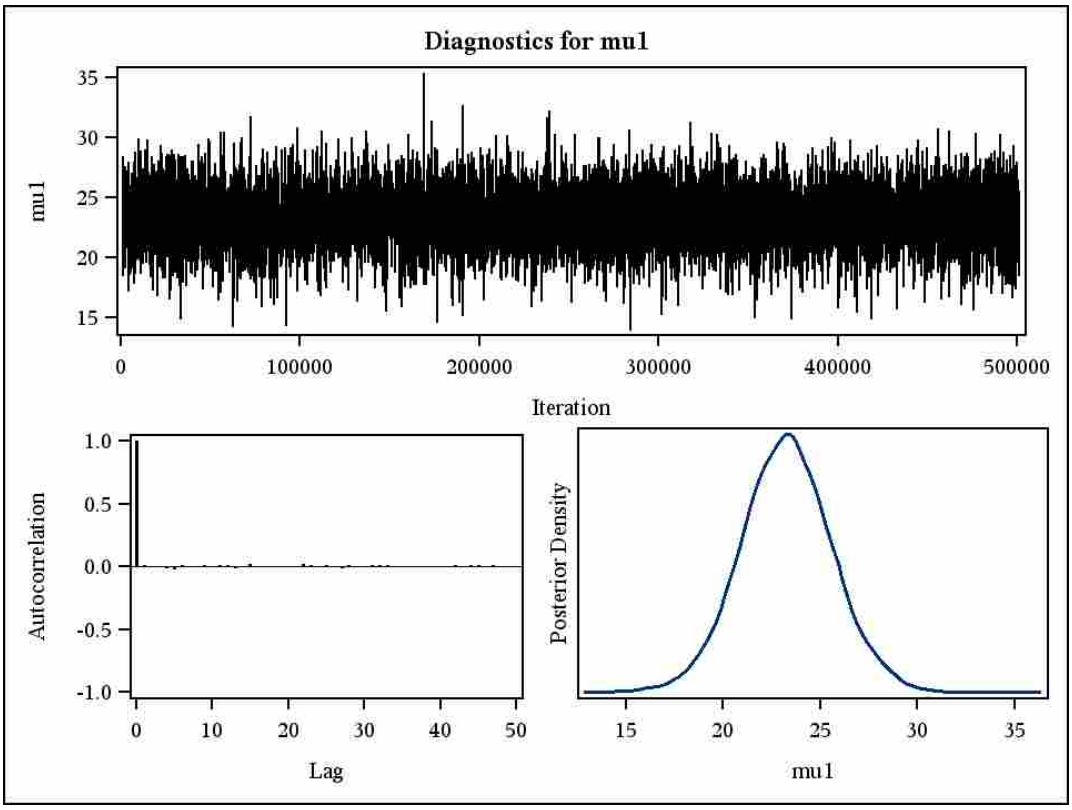
```
* read in the data file;
1  data micetwo;
2       infile "     " firstobs=2;
3       input mid tmt time censored timeleft;
4  run;
5
* print the data file for inspection;
6  proc print;
7  run;
8
* initializes saving of output as a pdf file;
9  ods pdf
10      file='     ';
* turn on graphics device;
11 ods graphics on;
12
13 proc mcmc data=micetwo outpost=miceout nmc=500000 thin=50 nbi=1000
   dic seed=1234 missing=AC monitor=(_parms_);
* define arrays of length 4;
14      array mu[4];
15      array sig2[4];
* set parameters and initial values;
* the colon indicates that the initial values should be applied to all
   array entries;
16      parms mu: 30;
17      parms sig2: 50;
* define priors;
* the colon indicates that the distribution be applied to all array
   entries;
18      prior mu: ~ normal(0, var=100000);
19      prior sig2: ~ gamma(2, iscale=0.02);
* if-else statements to determine appropriate handling of censored and
   uncensored observations;
20      if (timeleft ^= . and time ^= . and timeleft=time) then
21        llike=logpdf('normal',time,mu[tmt],sqrt(sig2[tmt]));
22      else if (timeleft ^= . and time = .) then
23        llike=logsdf('normal',timeleft,mu[tmt],sqrt(sig2[tmt]));
* likelihood;
24      model general(llike);
25 run;
26
* turn off graphics device;
27 ods graphics off;
* stop saving output file;
28 ods pdf close;
```

Table 16.3: Summary Statistics for Example 14 from PROC MCMC.

Posterior Summaries						
				Percentiles		
Parameter	N	Mean	Standard Deviation	25%	50%	75%
mu1	10000	23.2515	2.3414	21.7142	23.2541	24.7700
mu2	10000	34.6878	2.8419	32.7632	34.5624	36.4889
mu3	10000	28.0441	2.8065	26.1966	27.9809	29.8007
mu4	10000	24.8898	2.3718	23.3191	24.8451	26.4276
sig21	10000	110.0	34.9938	85.0655	104.0	128.2
sig22	10000	139.7	49.5930	104.8	131.0	164.6
sig23	10000	150.4	47.9291	115.9	143.4	175.9
sig24	10000	108.1	37.1378	81.5419	101.4	127.3

The summary statistics are given in table 16.3, showing the posterior summaries of the treatment means and associated variances. Figure 16.2 gives the posterior plots for the distribution of treatment one's mean survival time. These plots indicate that convergence was reached, no autocorrelation problems were encountered and the density of the posterior is drawn. Of interest, however, when looking at tables 16.1 and 16.3, one can see that the predicted survival times are similar despite the very different models utilized by the respective computer programs. It appears that treatment two yields the longest survival times. Even so, the researcher should conduct further analysis to determine the statistical significance of such an observation.

Figure 16.2: Summary plots for the posterior distribution of the mean survival time for treatment one.



16.3 Side by Side Computer Code

WinBUGS code:

```
model{
dummy <- mid[i];
    for(i in 1:80){
    time[i] ~ dweib(r, mu[tmt[i]])
       I(censored[i],)
    }
    for(i in 1:4){
    mu[i] <- exp(beta[i]);
    beta[i] ~ dnorm(0, 0.01);
    }
    r ~ dexp(0.1);
    }

#The data set:
mid[] tmt[] time[] censored[]
1 1 12 0
2 1 1 0
3 1 21 0
4 1 25 0
5 1 11 0
6 1 26 0
7 1 27 0
8 1 30 0
```

SAS code:

```
data micetwo;
    infile "  " firstobs=2;
    input mid tmt time censored
       timeleft;
run;

proc print;
run;

ods pdf
    file=``  '';
ods graphics on;

proc mcmc data=micetwo outpost=
   miceout nmc=500000 thin=50 nbi
   =1000 dic seed=1234 missing=AC
   monitor=(_parms_);
     array mu[4];
     array sig2[4];
     parms mu: 30;
     parms sig2: 50;
     prior mu: ~ normal(0, var
        =100000);
```

```
9 1 13 0
10 1 12 0
11 1 21 0
12 1 20 0
13 1 23 0
14 1 25 0
15 1 23 0
16 1 29 0
17 1 35 0
18 1 NA 40
19 1 31 0
20 1 36 0
21 2 32 0
22 2 27 0
23 2 23 0
24 2 12 0
25 2 18 0
26 2 NA 40
27 2 NA 40
28 2 38 0
29 2 29 0
30 2 30 0
31 2 NA 40
32 2 32 0
33 2 NA 40
34 2 NA 40
35 2 NA 40
```

```
        prior sig2: ~ gamma(2, iscale
          =0.02);
        if (timeleft ^= . and time ^=
            . and timeleft=time) then
            llike = logpdf('normal',
              time , mu[tmt], sqrt(
              sig2[tmt]));
        else if (timeleft ^= . and
          time = .) then
            llike = logsdf('normal',
              timeleft , mu[tmt], sqrt
              (sig2[tmt]));
        model general(llike);
    run;

    ods graphics off;
    ods pdf close;
```

36 2 NA 40

37 2 25 0

38 2 30 0

39 2 37 0

40 2 27 0

41 3 22 0

42 3 26 0

43 3 NA 40

44 3 28 0

45 3 19 0

46 3 15 0

47 3 12 0

48 3 35 0

49 3 35 0

50 3 10 0

51 3 22 0

52 3 18 0

53 3 NA 40

54 3 12 0

55 3 NA 40

56 3 NA 40

57 3 31 0

58 3 24 0

59 3 37 0

60 3 29 0

61 4 27 0

62 4 18 0

63 4 22 0

64 4 13 0

65 4 18 0

66 4 29 0

67 4 28 0

68 4 NA 40

69 4 16 0

70 4 22 0

71 4 26 0

72 4 19 0

73 4 NA 40

74 4 NA 40

75 4 17 0

76 4 28 0

77 4 26 0

78 4 12 0

79 4 17 0

80 4 26 0

END{};

BIBLIOGRAPHY

Box, G., and Tiao, G. (1973), *Bayesian Inference in Statistical Analysis*, Wiley Interscience.

BUGS (1996-2008), "The BUGS Project," *MRC Biostatistics Unit, Cambridge, UK*, Retrieved December 9, 2010, http://www.buffalostate.edu/library/docs/asa.pdf.

Carlin, B., and Louis, T. (2009), *Bayesian Methods for Data Analysis* (3rd ed.), CRC Press.

Casella, G., and Berger, R. (2002), *Statistical Inference* (2nd ed.), Duxbury.

Collett, D. (2003), *Modelling Survival Data in Medical Research* (2nd ed.), Chapman and Hall.

Gelfand, A., and Smith, A. (1990), "Sampling-Based Approaches to Calculating Marginal Densities," *Journal of the American Statistical Association*, 85, 398–409.

Geman, S., and Geman, D. (1984), "Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6, 721–741.

Hastings, W. (1970), "Monte Carlo Sampling Methods Using Markov Chains and Their Applications," *Biometrika*, 57, 97–109.

Lunn, D., Thomas, A., Best, N., and Spiegelhalter, S. (2000), "WinBUGS–A Bayesian modelling framework: Concepts, structure, and extensibility," *Statistics and Computing*, 10, 325–337.

Metropolis, N., Rosenbluth, A., M., R., Teller, A., and Teller, E. (1953), "Equation of State Calculations by Fast Computing Machines," *The Journal of Chemical Physics*, 21, 1087–1092.

OpenBUGS (2004), "OpenBUGS," *University of Helsinki, Finland*, Retrieved April 26, 2011, http://www.openbugs.info/w/.

Price, R. (1763), "A letter from the late Reverend Mr. Thomas Bayes, F. R. S. to John Canton, M. A. and F. R. S." *Philosophical Transations of the Royal Society of London*, 53, 269–271.

SAS (1976), "History: Stewardship for today, preservation for tomorrow," *SAS® Institute Inc.*, Retrieved December 29, 2010, http://www.sas.com/company/about/history.html.

SAS Institute Inc. (2008), "SAS/STAT® 9.2," *User's Guide*, NC: SAS Institute Inc.