Bayesian Inference

Michail Tsikerdekis

Abstract Bayesian inference has a long standing history in the world of statistics and this chapter aims to serve as an introduction to anyone who has not been formally introduced to the topic before. First, Bayesian inference is introduced using a simple and analytical example. Then, computational methods are introduced. Examples are provided with common HCI problems such as comparing two group rates based on a binary variable, numeric variable, as well as building a regression model.

1 Introduction

Dennis V. Lindley, arguably one of the leading advocates of Bayesian statistics has said that "Inside every nonBayesian there is a Bayesian struggling to get out" (Jaynes, 2003). To someone that has never heard of Bayesian statistics, this statement could sound a bit condescending. It could be interpreted as an attack towards "classical" statistics taught in most colleges as the main introductory statistics course. Such an attack is not without precedence. A "war" between Bayesians and Frequentists (a term often reserved for non-Bayesians), has been ongoing for the majority of the 20th century. Ronald A. Fisher, one of the leading contributors to frequentist statistics has referred to Bayesian statistics as "fallacious rubbish" (Aldrich, 2008). Others have followed in his example and a campaign to devalue Bayesian statistics has been going strong ever since. Yet, Bayesian statistics are still strong and often used in many scientific fields especially Computer Science. This is of no surprise since Alan Turing, seen by many as the father of Computer Science, has used Bayesian logic in his infamous Enigma machine meant to decipher German encrypted messages (McGrayne, 2011). Since then, Bayesian logic has been utilized in various problems such as artificial intelligence, machine learning, pattern

Michail Tsikerdekis
College of Communication and Information, University of Kentucky, Lexington KY, e-mail: tsikerdekis@uky.edu

recognition and even email spam classification. Why utilize Bayesian statistics to solve such problems? Looking back at Lindley's statement one may find the answer. Doing Bayesian statistics is in many ways how we intuitively perceive the world as humans; having a prior belief about a statement, looking at the evidence and adjusting our prior belief based on the evidence.

Human-Computer Interaction (HCI) has largely been a field of frequentism when it comes to quantitative research. This can in part be attributed to the lack of proper introductory curriculum for Bayesian statistics in HCI but also a lack of software that can accompany research. Just a few decades ago the computing power was simply non-existent for the complex models and calculations that are required to conduct Bayesian inference (Robert and Casella, 2011). Fast-forward a few decades and today Bayesian statistics are not only popular in a number of scientific fields but one can claim that they are not any more difficult to use than frequentist statistics. Bayesian inference is arguably more powerful and more informative due to its robustness for comparing hypotheses including the null hypothesis as well as making use of more information that is available to a researcher through the use of priors (Wagenmakers et al., 2008). This chapter serves as an introduction to Bayesian inference by presenting examples of typical HCI problems and Bayesian solutions to them.

2 Introduction to Bayes' Theorem

We will consider a computer science adaptation of the popular sunrise problem (Chung and AitSahlia, 2003) in order to understand how Bayesian inference works. Imagine a child receiving their first technology device (e.g., computer, laptop, tablet, etc.) and turning it on for the first time. After spending some time using the device, the child turns it off and goes to sleep. What would the probability be that the device will turn on again when the child wakes up? Frequentist solutions may just assign 100% probability to the event that the device will turn on or may express that if the device fails it would be a 1:1 odds for that event to happen. The difficulty of frequentist statistics for this particular problem is that they are not equipped to provide answers for statements requiring an expression of probabilities from an observer's perspective. They work well for cases such as survey research, where a phenomenon is standardized and repeatable, but, they fail when it comes to answering questions when an infinite number of repetitions (even hypothetical) may not be possible.

This introduces the first major difference between Bayesian inference and frequentist statistics. In the latter, the data (D) are random while the rate that the device turns on (θ) is an unknown yet it is a fixed value. In Bayesian inference, we are concerned with the present without involving hypothetical multiple future attempts. The data (D) are fixed, objective and known while the rate that the device will turn on (θ) is unknown and random. As pointed out by Jackman (2009), this does not mean that the rate θ (the rate the device turns on) keeps changing but rather that our belief about it changes as we observe the digital device turning on each time.

Hence, while a frequentist sees the probability of a device turning on as a characteristic of the device, a Bayesian sees the probability of a device turning on as a degree of one's belief given the observations. While both incorporate uncertainty, the frequentist approach fails in some problems since uncertainty is defined as a measurement error of finding the characteristic of a device. In contrast, a Bayesian's uncertainty is expressed in weaker probabilities that represent a belief due to limited data at hand or data that comes in conflict with prior beliefs.

This perspective gives a Bayesian statistician the power to assign probabilities to statements or beliefs. In our case that would be assigning probabilities to the rate θ. We can say that there are two different outcomes for the child's digital device: $\theta = 0$ the device does not turn on, and, $\theta = 1$ the device turns on. Or better yet, we may stipulate that the possibilities of the rate θ should be expressed in terms of a likelihood scale. We can say that the possibilities (or possible values) for our parameter θ will represent that a device will: Not Turn On, Not Very Probable to Turn On, Not Probable to Turn On, Probable to Turn On, Very Probable to Turn On, Turn On. Having six possibilities for our θ and assuming a range between 0 to 1, we can give θ six different numeric possibilities that correspond to our likelihood scale. So, θ is denoted by $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$.

Before we see any data (D) and even begin evaluating a problem, we have certain preconceptions or *prior* beliefs. These are prior probabilities ($p(\theta)$) that are assigned to each possible value (or outcome) for θ and should always sum to 1. For those fancying formal expressions that would be:

$$\sum_{i=0}^{n} p(\theta_i) = 1 \tag{1}$$

where n is all the discrete possible values for theta based on our made up likelihood scale.

If we believe prior to seeing any data that digital devices usually turn on, we can assign more weight on the $\theta = 1$ which corresponds to the *will turn on* belief in our likelihood scale and gradually decrease our assigned probabilities. Our *probability mass* for our prior beliefs for $p(\theta)$ will be 0.1, 0.15, 0.15, 0.20, 0.20, 0.4. Such a prior is considered to be a *subjective* prior. However, one can also decide that there is no apriori knowledge before one observes the data (D) and assign uniform probabilities to the prior $p(\theta)$ such as 0.2, 0.2, 0.2, 0.2, 0.2, 0.2. This is often called an *objective* prior since all possible outcomes for θ have equal probabilities ($p(\theta)$). However, even the uniform prior is not the least informative prior which can be selected[1] but it is considered sufficiently uninformative for many problems. Figure 1 shows the probability mass for the two examples of priors.

For our example we will assume that a friend informed the child that devices usually turn on and it is rare that they would not turn on. The child has a prior knowledge on the likelihood that a device will turn on, therefore we assume a prior belief that the device is likely to turn on. In R this will look like:

```
> Theta = c(0,0.2,0.4,0.6,0.8,1)
```

[1] Priors with higher variance can be considered less informative in this setting.

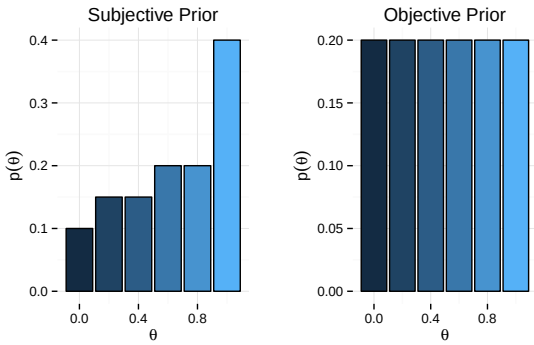

Fig. 1 Probability mass for prior beliefs. Left figure shows an subjective prior while right figure shows an objective prior.

```
> pTheta = c(0.1,0.15,0.15,0.20,0.20,0.4)
```

After declaring our prior beliefs, the next step involves processing the data D that the child has observed. This is usually expressed as the *Likelihood* or $p(D|\theta)$ which translates as the probability of the Data (D) given each θ. In our example, a device can turn on and turn off. Since this is a binary problem, we can define that the device can turn on as θ and device not turning on as $1-\theta$. Given that the possible outcomes for θ belong to the range of decimals between 0 and 1, you can think of the outcome *turning on* (θ) and outcome *not turning on* ($1-\theta$) as "polar" opposites. To avoid any confusion, the possible outcomes $\{\theta, 1-\theta\}$ which are derived from our data, are different from our arbitrarily defined possible values for $\theta = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. For example, we could represent θ in a three point scale such as $\theta = \{1, 2, 3\}$ and perceive the values as a discrete scale that means *Not Likely*, *Neutral* and *Likely*.

The likelihood ($p(D|\theta)$) is calculated using the binary data outcomes and observations (successes and failures) regarding these outcomes for each possible value θ. This is formally defined as:

$$\underbrace{p(D|\theta)}_{\text{likelihood}} = \underbrace{\theta^s}_{\text{succeses}} \underbrace{(1-\theta)^f}_{\text{failures}} \tag{2}$$

where s represents the numbers of times where the digital device turned on and f represents the number of times where the digital device failed to turn on[2]. In R we can write this using the following code:

```
> Data = c(1)
> s = sum( Data == 1 )
```

[2] A more formal version of the likelihood would be $p(\{y_1, ..., y_n\}|\theta) = \prod_i \theta^{y_i}(1-\theta)^{(1-y_i)}$, where the set $D = \{y_1, ..., y_n\}$ represents the outcome for the sequence of attempts to turn on the device (Kruschke, 2013).

```
> f = sum( Data == 0 )
> pDataGivenTheta = Theta^s * (1-Theta)^f
```

This will result in $p(D|\theta)$ with probabilities for each possible θ: 0.0, 0.2, 0.4, 0.6, 0.8, 1.0 which are also shown in Figure 2. Since we had just one success in turning on the device the biggest probability for our θ likelihood scale is placed in $\theta = 1$ with gradually decreasing probabilities on the rest possible values for θ. Notice that $\theta = 0$ that represents the *will not turn on* case has virtually a zero probability of occurring.

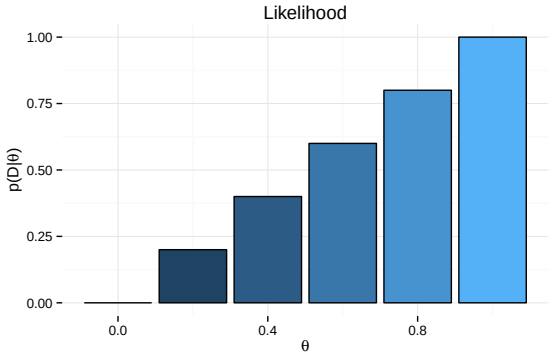

Fig. 2 Probability mass for likelihood $p(D|\theta)$.

Of course, Bayesian inference is all about the transformation of our prior beliefs ($p(\theta)$) to a posterior belief ($p(\theta|D)$) having seen the data through the likelihood ($p(D|\theta)$). The posterior is basically our set of probabilities for θ after we have seen the data. To achieve this we use a mathematical formula called *Bayes' Rule*. It was conceptualized by Reverend Thomas Bayes in 1740s and later was given a formal mathematical form and scientific application by Pierre Simon Laplace. It is formally defined as (Kruschke, 2010):

$$\underbrace{p(\theta|D)}_{\text{posterior}} = \underbrace{p(D|\theta)}_{\text{likelihood}} \underbrace{p(\theta)}_{\text{prior}} / \underbrace{p(D)}_{\text{evidence}} \tag{3}$$

In other words, having a prior belief, $p(\theta)$, times the likelihood, $p(D|\theta)$, divided by the evidence, $p(D)$, we can obtain a posterior belief conditional on the data $p(\theta|D)$. In cases where θ has a discrete set of values, the evidence can be calculated as the sum of the likelihood times the prior or formally defined as:

$$\underbrace{p(D)}_{\text{evidence}} = \sum_{i=1}^{n} \underbrace{p(D|\theta_i)}_{\text{likelihood}} \underbrace{p(\theta_i)}_{\text{prior}} \tag{4}$$

In R, we calculate the evidence $p(D)$ using:

```
> pData = sum( pDataGivenTheta * pTheta )
```

Notice that since $p(D)$ is a sum of all probabilities, the result is a number (in our example $p(D) = 0.77$) and not a probability mass like we had in the case of $p(\theta)$ and $p(D|\theta)$.

Having calculated all the necessary components we can finally calculate our posterior probabilities based on our data using:

```
> pThetaGivenData = pDataGivenTheta * pTheta / pData
```

The posterior probabilities for $p(\theta|D)$ are 0.00, 0.04, 0.08, 0.16, 0.21, 0.52. This is also shown in figure 3. Due to our prior belief favoring the the possible values for θ where a device will most likely turn on, and, the fact that the data through the likelihood also favored the case where a device turned on, our posterior belief is elevated higher towards the possibility that a device will turn on.

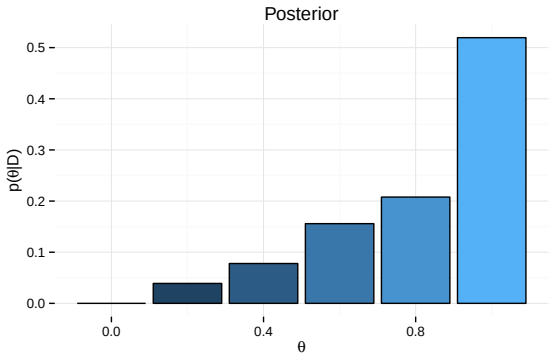

Fig. 3 Probability mass for posterior $p(\theta|D)$.

The answer provided by Bayesian inference may appear to be terminal but this is not the case. Just like in real life, we hold a belief and we update it as new evidence (or data) comes in. This process can be iterative with today's posterior belief becoming tomorrow's prior.

The child may accept her current beliefs about the device turning on. After ten days she can use those same beliefs as a new prior and calculate the new posterior but this time having observed ten successful times where the device turned on. This transformation of beliefs is shown in figure 4 and it is the core concept behind Bayesian inference. Repeated times of following this process and having continuous success in turning the device on will place even more weight in $\theta = 1$ for the posterior $p(\theta|D)$, however, this will never become 100%. There is always uncertainty and this unknown property of probability is included in Bayesian inference.

The example in this section has been using a discrete θ for instructional purposes. In practice, a continuous θ should be used instead. The exact mathematical approach for solving this problem would require a prior ($p(\theta)$) that would be a

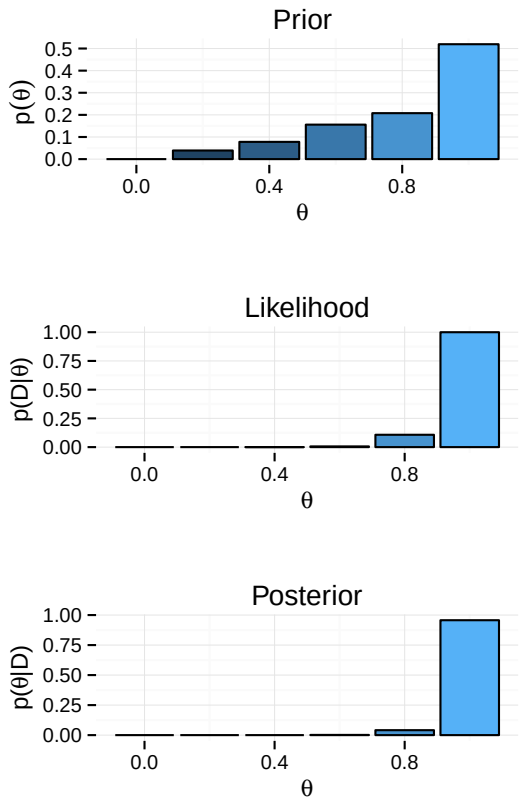

Fig. 4 Overview of prior, likelihood, and posterior.

continuous probability distribution and not discrete. There is also a need for a likelihood function ($p(D|\theta)$) that when combined with the prior, produces a posterior probability distribution of the same form with the prior. The prior of this form is often called a conjugate prior. This binary problem is mathematically solved using a Beta distribution prior and a Binomial likelihood[3]. The computational approach for

[3] In the Beta/Binomial approach, the prior is defined using the Beta distribution's probability density function (PDF). The simplified form of Beta's PDF (for this type of problem), is $p(\theta|\alpha,\beta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$. Assuming that the friend told the child that he/she has seen these devices turn on ten times ($\alpha = 10$) and fail to turn on two times ($\beta = 2$), our prior would be: $p(\theta|\alpha = 10,\beta = 2) \propto \theta^{10-1}(1-\theta)^{2-1}$. The likelihood function is based on the Bernoulli distribution with 1 successes and 0 failures expressed as $p(D|\theta) \propto \theta^1(1-\theta)^0$. Using Bayes' Rule we can combine the likelihood and prior to produce the posterior distribution: $p(\theta|D) \propto p(D|\theta)p(\theta) = \theta^{10-1}(1-\theta)^{2-1}\theta^1(1-\theta)^0 = \theta^{10}(1-\theta)^1$. The posterior density is a beta density that we can easily interpret if we calculate its α and β parameters: $\alpha = 10+1$ and $\beta = 1+1$. As such the mean for θ is $M = \alpha/(\alpha+\beta) = 11/(11+2) = 0.846$ or the child's beliefs that the device will turn on

this problem is described later on in this chapter. An alternative mathematical approach for this type of problem is the use of a Gamma prior and a Poisson likelihood (Lynch, 2007).

As it is the case with many digital devices, at some point in the future the device will not turn on and that may make us question our belief in induction, positivism, and, our ability to deal with an unhappy child. However, this is a matter for another discussion.

3 Computing Bayesian Statistics

It is easy to demonstrate how Bayesian statistics work in a simple problem such as the one described in the previous section, however, nowadays we do not conduct Bayesian inference by hand. The issue is one of complexity as problems are defined in more "detail." For example, the probability mass for our posterior may be determined by a more complex likelihood ($p(D|\theta)$) function that involves more outcomes for θ. Additionally, the possible outcome values for θ may not be discrete (e.g., the probability scale metaphor that we have used in our previous example) but can be instead continuous involving a range of values (e.g., a range from 0 to 1 with all possible decimal points in between). For example, studying user reaction times when replying to emails would require such a continuous θ. The posterior probabilities in this case are not called a probability mass but a *probability density* (Kruschke, 2010). The evidence $p(D)$ which is the sum of the likelihood times the prior for all θ values cannot be calculated as such since there could be an infinite number of values with all of their decimals possibilities. Hence, we calculate instead the integral of the likelihood times the prior for all θ values which in layman's terms produces an approximation of what the sum would be if we could calculate it. Such complex problems require a different approach to the analytical approach that we have used in the previous section.

Monte Carlo Markov Chain algorithms are used as a tool in order to solve complex problems since they can approximate model parameters such as the parameter θ in our previous example (Gilks, 2005). They use random walks (switching between different values of all parameters in a model) based on a model of probabilities derived from our observed data to approximate the point where a probability mass (or probability density) for parameters is reaching a state of "equilibrium." The random walk creates a sequence which is called a *chain* (also known as Markov chain) and the length of a chain (called the *sample*) is important for accurately determining the value of a parameter. Each step in a chain is considered to be "memoryless." In essence, each step transitions between various states for a state space and the probability distribution for every step is only dependent on the previous step. The-

---

is focused at 84.6%. The standard deviation is $SD = \sqrt{\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}} \approx 0.0093$. The probability interval with a 95% probability will be $0.846 \pm 1.96 \times 0.0093$ which places the child's belief in the device turning on between 82.7% and 86.4%.

oretically, an infinite sample would create the most accurate result but in practical terms we usually obtain a large enough sample (Plummer et al., 2006).

Figure 5 demonstrates how the algorithm works step by step to approximate the probability "equilibrium." The top two plots demonstrate a small sample. Our sample is just 10 which creates 10 steps in our chain (also called an *interval*). The algorithm walks randomly between values but still within the constraints of observed data. The *traceplot* demonstrates this walk and chain. The index represents each step of the walk while the y-axis shows the sampled value for that step. If we summarize these ten values we can create a probability *density plot* for our parameter $\bar{X}$. It is evident that a sample of 10 is not large enough to obtain an accurate estimate. However, as we increase the sample (seen in the rest of the plots in figure 5) we slowly achieve higher accuracy and approximate our probability "equilibrium."

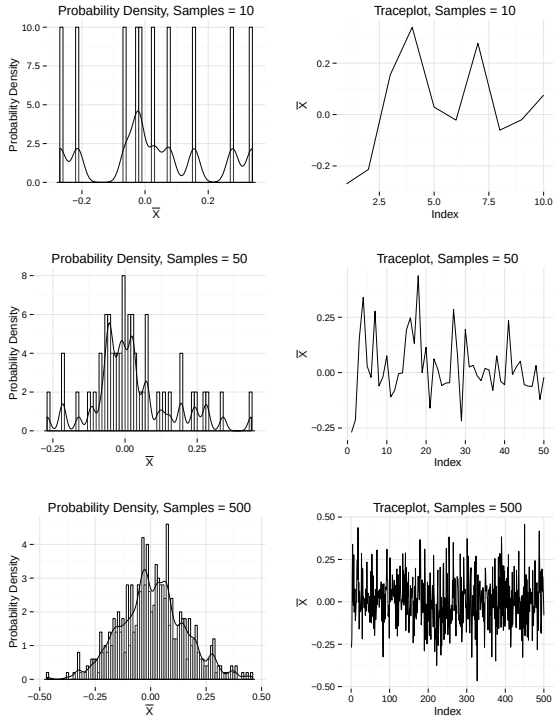

Fig. 5 Example of Probability Density plot and Trace plot of MCMC with varying samples for $\bar{X}$ parameter.

Early in the sample, chains often appear to be random. They can take a while to get into the "sweet" spot of a parameter's probabilities. For this reason, we often decide to ignore the early parts of a chain and retain the latter parts. This is called a *burnin*. For example, we may decide to retrieve a sample of 15,000 steps but have a

burnin of the first 15,000 steps. Figure 6 demonstrates a sample of 15,000 without a burnin and a sample with a burnin of 15,000. The lower two plots basically start from the 15,000th step and end at 30,000th. Notice that for the first sample without the burnin, the chain moves slightly downward. Chains that take longer to converge are often referred to as *slow-mixing* (Lynch, 2007).

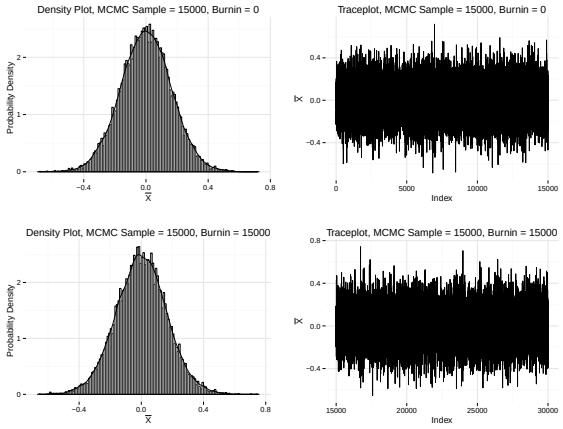

Fig. 6 Example of Probability Density plot and Trace plot of MCMC with the same sample size but different burnin for $\bar{X}$ parameter.

The process of random walking utilized by MCMC algorithms can lead to chains that look different each time. Using multiple chains and aggregating the results into one ensemble can create a more accurate estimate. Figure 7 demonstrates 4 different chains for a small MCMC sample.

An MCMC sample with a large interval and many chains can be a computationally intensive task especially depending on the number of parameters we wish to retain in order to construct their probability density distribution. In order to make this process more memory efficient for computational systems the idea of *thinning* was invented. Thinning retains only every *n*th element in a chain therefore reducing the size of the sample in memory (Albert, 2009).

While we know that MCMC algorithms can produce a desired probability distribution for a parameter, we can never be sure how long it will take for a chain to converge to that distribution. We may set an interval of 100,000 and a chain may still not converge. For this reason, we use tests of *convergence* (Rossi et al., 2005). There are several tests to verify an MCMC sample's convergence. We can test for convergence visually or using algorithms that test for it.

Visual inspection for convergence can be done by viewing traceplots for chains and their *mixing*. If a chain takes longer to move through the whole parameter space then it will take longer to converge. Good mixing appears as a trace that tends to be stable within the values of the parameter (or parameter space). Figure 8 provides an example of a sample that converges and one that does not.

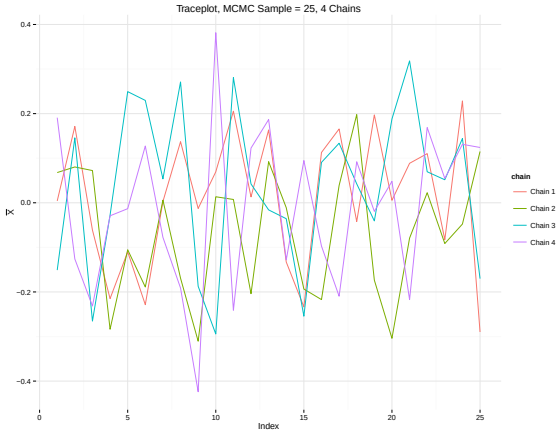

Fig. 7 Example of Trace plot with the same sample size with 4 chains for $\bar{X}$ parameter.

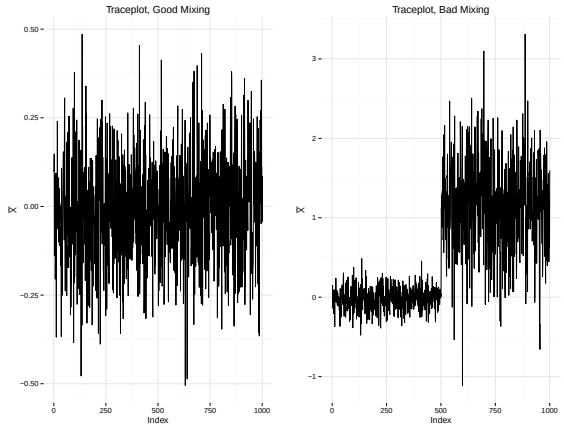

Fig. 8 Example of Trace plot that converges and one that does not for $\bar{X}$ parameter.

The *Gelman and Rubin shrink factor* is another way to verify that the convergence (Gelman and Rubin, 1992). The measure uses within-chain and between-chain variance to produce a value of how well the model is converging over time. If the shrink factor is close to 1 then our model has converged while values beyond 1.2 are indicative of a model that may have not converged and requires a longer chain. However, there may be an occasion where one may receive a value that is smaller than 1.2 by chance. Hence, plotting the statistic over time is considered to be a more accurate approach. Figure 9 shows two *Gelman plots* for two parameters. The shrink factor appears to reach 1 and hover around it after approximately the 2,000th step in the sample. This is the point where the model started to converge. In this case, a 3,000 interval for our MCMC sample would be sufficient.

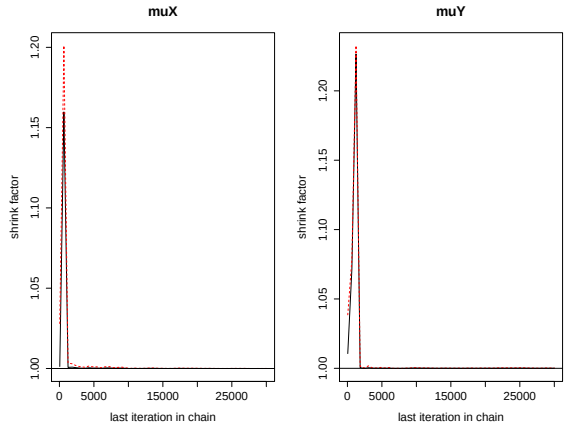

Fig. 9 Example of Gelman and Rubin shrink factor plot for two parameters.

The *Geweke diagnostic* is another popular measure for detecting convergence (Cowles and Carlin, 1996). The diagnostic tests for convergence on a per chain basis. The test retrieves samples from the chain of two non-overlapping samples (by default the first 10% and the last 50% of a chain) and conducts a test of equality of means. If convergence has occurred the means should be virtually the same. The result is a Z-score where scores below 1.98 indicate convergence and greater than 1.98 indicate statistically significant difference for the samples derived from a chain using a significance level .05 (5% probability that we are wrong over repeated samples). One can also produce a plot over time for the statistic which is demonstrated in figure 10. In this case, we can notice that parameter *muX* is having trouble converging. Some of the points are beyond the threshold lines that mark the ± 1.98 limit. One also has to take into account whether an MCMC chain had a burnin which would affect convergence diagnostic.

Finally, there other diagnostics that can verify convergence for MCMC algorithms such as measuring autocorrelation lag, Raferty and Lewis diagnostic, and, Heidelberg and Welch diagnostic (Cowles and Carlin, 1996).

4 Bayesian Two Group Comparison for Binary Variables

We can consider a practical example in order to demonstrate how MCMC algorithms assist in Bayesian inference. The example below can also be solved using a mathematical approach with a combination of a Beta prior and a Bernoulli likelihood. MCMC is used in this section for instructional purposes.

A company wishes to improve employees email practices by investing in Mango smart watches. The employees use mainly their desktop computers but need to carry their tablets whenever they are away from their workspace. Tablets however are

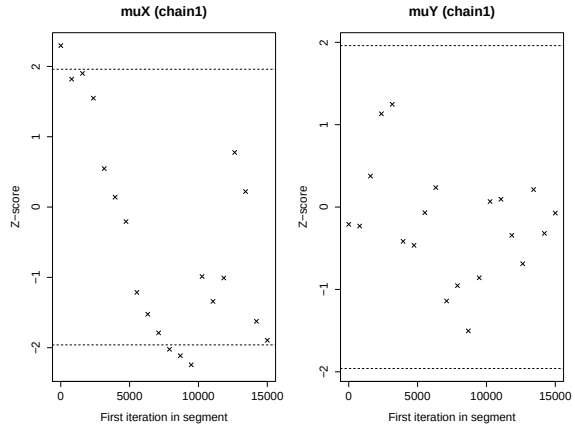

Fig. 10 Example of Geweke plot for two parameters based on one chain.

bulkier and so many employees prefer to carry their personal cell phones with them. The use of personal cell phones for work-related matters is a security risk according to company policy and so replacing them with smart watches is considered to be a safer alternative by the company. The company conducts an experiment where group A gets smart watches while group B does not. They receive back answers on whether users have utilized their cell phones during the study. The results were measured in a binary scale for individuals that used their cell phones during the period of the study and individuals that did not.

We need to compare the rates between the two groups in order to determine if the smart watches are a good choice for reducing the likelihood for cell phone use [4]. The problem involves a dichotomous variable (exactly a yes or no question). We start by defining our data in R:

```
> source("generate.R")
> email$usedcellphone = 0 #generating random data
> email$usedcellphone[email$team == 0] = sample(c(0,1),
+ nrow(email[email$team == 0,]),2,prob=c(.80,.20))
> email$usedcellphone[email$team == 1] = sample(c(0,1),
+ nrow(email[email$team == 1,]),2,prob=c(.20,.80))
> s1 = sum(email$usedcellphone[email$team == 0])
> s2 = sum(email$usedcellphone[email$team == 1])
> n1 = nrow(email[email$team == 0,])
> n2 = nrow(email[email$team == 1,])
```

where $s1$ and $s2$ represent the number of people who end up using their cellphones for each group respectively, and, $n1$ and $n2$ represent the total number of

[4] An alternative approach to solving the problem would be to use Bayesian Probit Regression (Jackman, 2009).

people on each group. In our example, the total number of people varies between the two groups.

MCMC algorithms work based on the process of model building. The aim is to structure probability distributions for our parameters in order to simulate them based on our observed data. Understanding probability distributions beforehand is essential for building models. For example, normal distributions are reserved usually for numeric variables and take as parameters the mean of a variable and the standard deviation. In the case of binary variables, beta distributions are more appropriate because they are continuous, bound between 0 and 1 and take two parameters α and β, which define the slope of the distribution. Figure 11 shows examples of beta distributions for a θ parameter. In our previous example with the digital device, we decided that θ values will represent a likelihood scale. In this problem, we can decide that θ values will denote the likelihood of cell phone use with 1 representing an absolute probability for using a cell phone while 0 representing no probability of using a cell phone. As such, α and β become representations of using and not using a cell phone for our beta distribution.

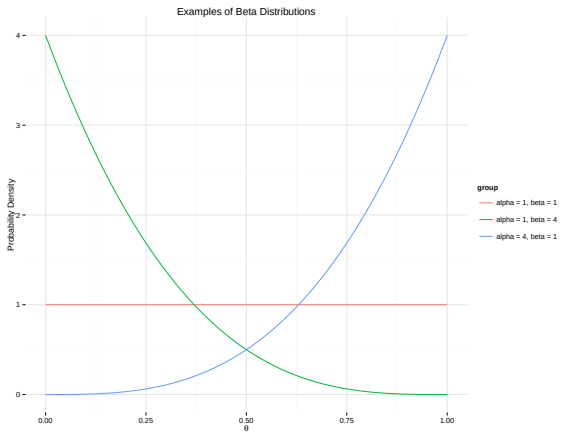

Fig. 11 Example Beta distributions based on different α and β.

Just like our digital device problem turning on and off, we can define a parameter denoting cell phone use for each group, $\theta 1$ for group 1 and $\theta 2$ for group 2. These rates will also need to be assigned priors based on our prior belief. In this case, we decide that we do not have prior knowledge on what the outcome may be for our experiment and we define equal prior "results":

```
> s1prior = 1
> f1prior = 1
> s2prior = 1
> f2prior = 1
```

where $s1prior$ and $s2prior$ are the prior rates of people that use cell phones for the two groups and $f1prior$ and $f2prior$ are the prior rates for people that do not use cell phones for each group respectively. In this case the values for all variables are 1 but they could have been any number as long as high and low values are equal so that we can produce a uniform set of prior probabilities (see figure 11 first example).

MCMC modelling in R is represented by a set of distributions and their parameters (e.g., α and β for a Beta distribution) as well as functions. All of these are enclosed within an R function that contains the model. Models unlike programming code do not have to be written sequentially as they are not executed as such. Additionally, in programming we often declare functions as y = x + z where y is the unknown while x and z are variables with known values. In MCMC models, we could also declare the same function but with x and y as variables with a known value and z being the unknown variable.

```
> jags.bin <- function() {
+     theta1 ~ dbeta(s1prior,f1prior)
+     theta2 ~ dbeta(s2prior,f2prior)
+     s1 ~ dbin(theta1,n1)
+     s2 ~ dbin(theta2,n2)
+     delta <- theta1 - theta2
+ }
```

Our two distributions for $\theta1$ and $\theta2$ are declared within the JAGS model. JAGS stands for Just Another Gibbs Sampler and it is a program for analyzing Bayesian models using MCMC sampling. We first supply parameters and their assigned prior beliefs (which are uniform in this case) to our model. We also need to define the likelihood. Binomial distribution is our choice for this problem which is a discrete distribution using parameters as the probability rate (probability of using a cell phone) and the total number of sequences. From a programming perspective this may appear a bit peculiar, however, models do not have to operate sequentially or have functions and distributions being sequentially defined. As long as all variables, distributions and parameters are all accounted for, the model will produce results. In this case, $s1$, $s2$, $n1$, $n2$ are known discrete variables and $\theta1$ and $\theta2$ are unknown albeit with priors defined. Finally, we can also calculate the difference between $\theta1$ and $\theta2$ called δ (in the R code defined as delta).

After setting up our model we can proceed by setting the parameters for MCMC using the command *jag.model* and then utilize *coda.samples* to generate posterior samples based on the parameters of interest. The process simulates all variables for our model however the sampled chains that are returned are only those that interest us. These are declared as a list.

```
> n.simu <- 50000
> n.burnin <- n.simu/2
> par <- c("theta1","theta2","delta","deltaprior",
+     "theta1prior","theta2prior")
> D <- list(s1 = s1, s2 = s2, n1 = n1, n2 = n2,
```

```
+     s1prior = s1prior, s2prior = s2prior,
+     f1prior = f1prior, f2prior = f2prior)
> m.jags <- jags.model("jags.txt", data = D,
+     n.adapt = n.burnin, quiet = TRUE, n.chains = 4)
> s <- coda.samples(m.jags, par,
+     n.iter = n.simu - n.burnin, quiet = TRUE)
```

The object produced by *coda.samples* contains all the variables requested using the list *par*. The object can be converted to a data frame containing all chains for easier post-processing. The downside to this approach is that the data frame can be quite large unless thinning is applied. Using the data frame we can then obtain the mean value of the posterior sample for $\theta 1$, $\theta 2$, and δ.

```
> df = as.data.frame(as.matrix( s ))
> mean(df$theta1)
[1] 0.2335576
> mean(df$theta2)
[1] 0.7144396
> mean(df$delta)
[1] -0.480882
```

In this case, we can see that group B has a higher cell phone use rate than group A. The difference δ is almost half (0.481) for our values of θ that range between 0 and 1. Just like traditional confidence intervals in frequentist statistics we can also calculate 95% probability intervals. The commonly used probability interval for Bayesian statistics is called *High Density Interval* (HDI) included in R's *BEST* package.

```
> c(hdi(df$theta1,.95)[1],hdi(df$theta1,.95)[2])
    lower     upper
0.0930076 0.3841601
> c(hdi(df$theta2,.95)[1],hdi(df$theta2,.95)[2])
    lower     upper
0.4879375 0.9260797
> c(hdi(df$delta,.95)[1],hdi(df$delta,.95)[2])
     lower      upper
-0.7445349 -0.2098999
```

We can observe that the probability interval for the difference is quite broad which is likely a result of a small sample. However, we can still be confident based on the results that group B exhibited higher cell phone use. In other words, the implementation of devices in group A had a substantial improvement in reducing cell phone use and therefore improving security in compliance with what the company wanted to achieve. But, how about hypothesis testing?

Bayesian inference also provides paths to perform hypothesis testing. A popular formula is *Bayes Factor* which allows us to test the odds ratio between two hypotheses (e.g., $H0$: $\delta = 0$ and $H1$: $\delta \neq 0$). This is covered in chapter 9 of this book.

An alternative to the Bayes Factor is provided by Kruschke (2010), which is similar to equivalence testing used in biomedical sciences. One can define a *Region of Practical Equivalence* (ROPE) around a point of non-difference between rates and determine on whether a practical difference exists between two rates. In our case, we could suggest that for the point of equivalence ($\delta = 0$) we define a ROPE in a ± 0.20. The definition is dependent on the context and experimental design. For example, in our binary problem of cell phone use, we may assume that if the difference between two groups is less than 0.2 then we may decide that the smart watches are not a worthwhile investment. Different companies may be willing to invest on the smart watches with differences as low as 0.1. After a ROPE is defined, we identify whether we can accept the hypothesis of $\delta = 0$ or whether our 95% HDI falls within the ROPE or falls completely outside the ROPE. As our sample size grows, 95% HDI tends to accumulate more around the mean of the posterior sample and as such probability around the posterior mean value increases.

To calculate the percentage of our posterior sample that falls within the ROPE we need to determine if any of the sample values fall within our ROPE.

```
> ROPE = c(-0.2,0.2)
> (pcInROPE =  sum( df$delta > ROPE[1] & df$delta <
+ ROPE[2] ) / length( df$delta ) )
[1] 0.02984
```

Determining the amount of HDI that falls within the ROPE requires us to calculate what is the 95% of our posterior sample for δ and then use a similar approach to determine what amount falls within the ROPE. The code for this is provided in the supplementary materials of the book.

For this particular example, 2.984% of our posterior sample falls within the ROPE, while 0% of our HDI falls within the ROPE. We can therefore reject the null hypothesis and accept that there is a substantial difference between groups.

Similar to our previous example, the advantage of using Bayesian inference to determine the improvement of processes due to the implementation of a product or user interface is that we can relaunch an experiment for future device implementations. The second experiment can then utilize the posterior results from the first experiment as priors.

5 Bayesian Two Group Comparison for Numeric Variables

In HCI, we often want to evaluate the difference for a numeric variable between two groups. In our smart watch example, we have measured the email response times between group A that had the smart watch and group B that did not have the smart watch.

The process of obtaining posterior samples for each group and hypothesis testing is similar to testing binary rates. The main difference involves establishing a model that can reflect the nature of the numeric variable.

In this case, distributions of numeric variables are usually normal. A normal distribution is a continuous probability distribution that accumulates around a single point and gradually dissipates (see Fig. 6). Comparing two rates between normal distributions is establishing the difference between their two means. Hence, our likelihood portion for our model will include a loop between our data for each group using a normal distribution.

```
> jags.bin <- function() {
+     for (i in 1:n1) {
+         group1[i] ~ dnorm(muX, tauX)
+     }
+     for (i in 1:n2) {
+         group2[i] ~ dnorm(muY, tauY)
+     }
+     muX ~ dnorm(0, 0.001)
+     muY ~ dnorm(0, 0.001)
+     sigmaX ~ dunif(0, 1000)
+     sigmaY ~ dunif(0, 1000)
+     tauX <- 1 / (sigmaX * sigmaX)
+     tauY <- 1 / (sigmaY * sigmaY)
+     delta <- muY - muX
+ }
```

Notice that in this case the known part is the list of values for $group1$ and $group2$ that are indexed by i within the loop. The means, μ_X and μ_Y (in the code typed as muX and muY), as well as the standard deviations, τ_X and τ_Y (in the code typed as tauX and tauY), are the unknown components. Normal distributions in JAGS use standard deviations using τ and not σ which is more common in statistics. We need to take this into account when modeling.

Since the means μ_X and μ_Y are our rates of interest (just like θ previously), we need to represent them in the form of a probability distribution not a single point estimate. As such we can set their priors in a form of a normal distribution. We also set the standard deviations, τ_X and τ_Y, derived from σ_X and σ_Y (in the code typed as sigmaX and sigmaY) in a form of a uniform distribution where all probabilities are the same for all possible values.

Finally, we add to the model any final calculated variables such as the difference between the means, δ (in the code typed as delta).

By declaring the model using JAGS, we can obtain posterior samples and determine the means for the two groups. These can be obtained as a point estimate (e.g., based on the mean of posterior sample) or as a posterior probability distributions for the two groups which can be plotted. These are shown on figure 12.

It is evident that the implementation of smart watches for group A had a dramatic effect in reducing email response time. Notice that the two probability distributions do not overlap and that we are more certain about a point estimate for group A.

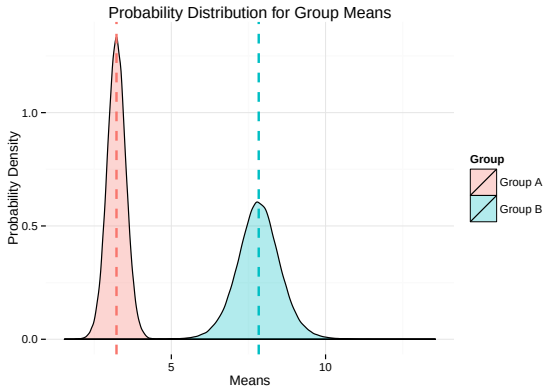

Fig. 12 Probability distributions between Group A and Group B for the smart watch experiment.

We can further calculate the point estimate based on the mean for the posterior samples of the two means, their 95% HDI, and further use the posterior sample for δ to determine based on a ROPE whether to reject or accept the null hypothesis.

The results for this particular experiment suggest that there is a considerable difference between group A (M=3.222, 95% HDI [2.597, 3.822]) and group B (M=7.835, 95% HDI [6.372, 9.239]) since the 95% HDI for δ falls outside the ROPE of $\pm$0.2. The mean difference is 4.6 minutes which can have a considerable effect on productivity. Just like before, whether smart watches are a worthwhile investment depends on the company and the threshold used as a ROPE. For example, in our case the managers have decided on using a really small ROPE, which is a 0.2 minutes difference. On the other hand, if managers were to decide that an investment was worth it only if email response time was beyond three minutes then the ROPE would have to be set to 3. Even for this case, the HDI falls outside the ROPE so the smart watches are a good investment. An alternative model to the one offered in this chapter for a Bayesian t-test can be found by Lee and Wagenmakers (2014).

6 Bayesian Regression with Numeric Predicted Variable

Aside from determining how an experimental design may influence the outcome of email response time, we can also attempt to determine information such as how technical efficacy of people may affect email response time. We can start by generating an artificially correlated variable for illustrative purposes.

```
> source("generate.r")
> email$technicalEfficacy = email$responseTime *
+     runif(length(email$responseTime), 0.0, 5.0)
```

Regression models that involve numeric predictors are implemented using the regular regression formula ($y = b_0 + b_1 * x$) with a slight modification:

```
> jags.bin <- function() {
+     for(i in 1:N) {
+         y[i] ~ dnorm(f[i], tau)
+         f[i] <- b0 + b1 * x[i]
+     }
+
+     # Priors
+     tau <- 1/pow(sigma,2)
+     sigma ~ dunif(0, 1000)
+     b0 ~ dnorm(0, 0.001)
+     b1 ~ dnorm(0, 0.001)
+
+     # R-squared calculation
+     y.mean <- mean(y[])
+     for (i in 1:N) {
+         ss.res.temp[i] <- y[i] - f[i]
+         ss.res[i] <- pow(ss.res.temp[i], 2)
+         ss.reg.temp[i] <- f[i] - y.mean
+         ss.reg[i] <- pow(ss.reg.temp[i], 2)
+         ss.tot.temp[i] <- y[i] - y.mean
+         ss.tot[i] <- pow(ss.tot.temp[i], 2)
+     }
+     r.squared <- (sum(ss.reg[])) / (sum(ss.tot[]))
+ }
```

In this case, the known variables are y (representing response time), x (representing technical efficacy) and N which is the total number of cases that are used for our loop. Our modeled variable is f which forms the typical regression formula with b_0 being the intercept and b_1 the coefficient for our predictor variable.

Priors are defined in our model for τ, σ, b_0 and b_1. The priors in this case are objective since there is no prior knowledge for these variables. Priors for the coefficients have a mean of 0 and a small standard deviation while sigma is a uniform distribution, which is a distribution with equal probability for all outcomes.

Using the coefficients b_0 and b_1, we can obtain all relevant data for our regression model similar to a frequentist linear regression. We can also implement within the model other calculated statistics. An important measure for regression is calculating R^2 (the amount of explained variance by our model) and in Bayesian regression we can obtain a probability distribution for it. In this example, the implementation of R^2 uses the exact same formula used in frequentist regression.

We can then obtain our posterior samples the same way we did before using *JAGS* and *coda.samples*. Figure 13 demonstrates the probability distribution for R^2 which indicates a substantial amount of accounted variance.

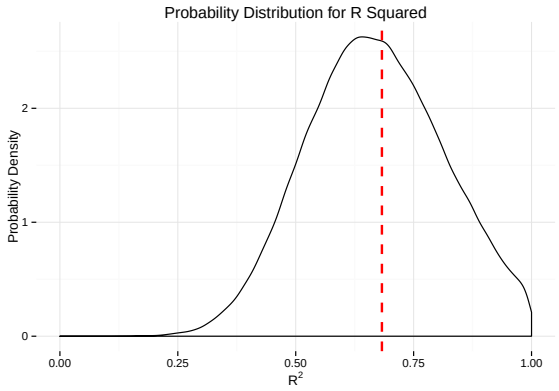

Fig. 13 Probability distribution for R^2 when modeling email response time and technical efficacy. Red dotted line indicates the mean for the posterior sample. The more narrower the curve around a single point, the more likely the R^2 is around that single point. Narrower curves also produce tighter HDIs for R^2 which increases are certainty for the accounted variance of a model around a specific point.

We can further obtain point estimates for our coefficients as well as their 95% HDIs.

```
> mean(df$b0)
[1] 2.230924
> c(hdi(df$b0,.95)[1],hdi(df$b0,.95)[2])
   lower    upper
1.463655 3.008316
> mean(df$b1)
[1] 0.1877244
> c(hdi(df$b1,.95)[1],hdi(df$b1,.95)[2])
    lower     upper
0.1433942 0.2326414
> mean(df$r.squared)
[1] 0.6821717
> c(hdi(df$r.squared,.95)[1],hdi(df$r.squared,.95)[2])
    lower     upper
0.3720231 1.0010375
```

Notice that the upper bounds for the R^2 95% HDI exceed 1 which is the otherwise analytical upper limit. However, since in Bayesian statistics we simulate calculated variables, these could go beyond the limits for metrics such as R^2. It is a consequence caused by the "noise" created by our simulation. As the sample size increases this behavior will dissipate and R^2 will be bounded between 0 and 1 as it is expected to be.

The accounted variance (R^2) informs us that technical efficacy is an important factor that affects email response time. The coefficient for technical efficacy ($b1$) has a mean of 0.18 according to its posterior probability distribution. The arbitrary scale for technical efficacy used in this example varies between 0 and 51 with lowest scores representing higher efficacy. Assuming that email response time is measured in minutes, this translates to about a minute of slower response time for every five-point increase (less technical expertise) in the technical efficacy scale. The result suggests that training individuals to have better technical skills will result in a substantial increase in email response time. On the other hand, if the email response time was measured in seconds, the increase in response time may have been negligible even though our model will still suggest a large accounted variance between technical efficacy and email response time.

7 Do not reinvent the wheel

The examples demonstrated in this chapter are not meant to cover a complete view of Bayesian inference but rather serve as an introduction. At this point, it may seem that Bayesian inference may involve a lot of work for HCI professionals, however, this is not the case. Bayesian statistics were prevented from appearing in mainstream curriculum due to the computational inefficiency that existed for MCMC algorithms over the past decades. This fact has also restricted many to develop software for Bayesian statistics that requires the same effort comparable to building a traditional t-test or linear regression model. At the moment, software for Bayesian statistics is not as flexible for building complex models and *JAGS* or similar modeling software is necessary. However, many methods frequently used by HCI professionals such as a variety of regression models as well as various statistical tests (e.g., t-test) are available. For example, the package *BEST* in R Kruschke (based on 2013) provides a way for testing two group means. Also, packages such as *Zelig* (Imai et al., 2008) include an ensemble of many Bayesian methods such as Bayesian Logistic Regression, Multinomial Logistic Regression, Linear Regression and Ordered Probit Regression. As a brief example, the regression model that we tested previously can be built using Zelig with just two lines of code.

```
> z.out <- zelig(output ~ predictor,
+     model = "normal.bayes", data=df,
+     mcmc = n.simu - n.burnin, burnin = n.burnin)
> summary(z.out)

Call: zelig(formula = output ~ predictor,
model = "normal.bayes", data = df,
mcmc = n.simu - n.burnin, burnin = n.burnin)

Iterations = 25001:50000
Thinning interval = 1
```

```
Number of chains = 1
Sample size per chain = 25000

 Mean, standard deviation, and quantiles
  for marginal posterior distributions.

              Mean     SD   2.5%    50%  97.5%
(Intercept) 2.2323 0.3848 1.4725 2.2316 2.9955
predictor   0.1876 0.0223 0.1440 0.1875 0.2316
sigma2      2.7399 0.6624 1.7388 2.6451 4.3284
```

Research papers in HCI as well as other fields have utilized Bayesian analysis as the main analytical method or as a supplementary method (Tsikerdekis, 2013; Triantafyllopoulos and Pikoulas, 2002; Muchnik et al., 2013; Volf et al., 2014; Trusov et al., 2010). When utilizing Bayesian methods the degree of introduction can vary. Some authors choose to provide a bit more background information on the methods used while others prefer to publish the result and refer readers to textbooks for more information on the methods. The same degree of variance can be found in the use of probability distribution charts. At times, authors choose to display point estimates even though Bayesian analyses offer probability distributions for the parameters of interest. The language can also vary when it comes to reporting results. For example, consideration should be given when one needs to report MCMC sample, burnin, thinning and additional measures that may be essential for replicating the same results or approximating them.

Bayesian inference has arrived and it is not just easier to perform but much more powerful compared to frequentist statistics (Wagenmakers et al., 2008). It adds more diversity to HCI research and thus produces more intuitive results that can be directly interpreted based on current and past knowledge. It is time for all of us to listen to our inner Bayesian that is struggling to get out!

References

Albert, J. (2009). *Bayesian Computation with R*. Number 3 in Use R! Springer, New York, NY.

Aldrich, J. (2008). R. A. Fisher on Bayes and Bayes Theorem. *Bayesian Analysis*, 3(1):161–170.

Chung, K. L. and AitSahlia, F. (2003). *Elementary Probability Theory: With Stochastic Processes and an Introduction to Mathematical Finance*. Springer Undergraduate Texts in Mathematics and Technology. Springer.

Cowles, M. K. and Carlin, B. P. (1996). Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review. *Journal of the American Statistical Association*, 91(434):883–904.

Gelman, A. and Rubin, D. B. (1992). Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 7(4):457–472.

Gilks, W. R. (2005). Markov Chain Monte Carlo. In *Encyclopedia of Biostatistics*. John Wiley & Sons, Ltd.

Imai, K., King, G., and Lau, O. (2008). Toward a Common Framework for Statistical Analysis and Development. *Journal of Computational and Graphical Statistics*, 17(4):892–913.

Jackman, S. (2009). *Bayesian Analysis for the Social Sciences*. Wiley, Hoboken, NJ.

Jaynes, E. T. (2003). *Probability Theory: The Logic of Science*. Cambridge University Press, Cambridge.

Kruschke, J. K. (2010). *Doing Bayesian Data Analysis: A Tutorial with R and BUGS*, volume 1. Academic Press.

Kruschke, J. K. (2013). Bayesian estimation supersedes the t test. *Journal of Experimental Psychology: General*, 142(2):573–603.

Lee, M. D. and Wagenmakers, E. J. (2014). *Bayesian Cognitive Modeling: A Practical Course*. Cambridge University Press, Cambridge.

Lynch, S. M. (2007). *Introduction to Applied Bayesian Statistics and Estimation for Social Scientists*. Springer.

McGrayne, S. B. (2011). *The Theory that Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunted Down Russian Submarines, & Emerged Triumphant from Two Centuries of Controversy*. Yale University Press.

Muchnik, L., Aral, S., and Taylor, S. J. (2013). Social Influence Bias: A Randomized Experiment. *Science*, 341(6146):647–651.

Plummer, M., Best, N., Cowles, K., and Vines, K. (2006). CODA: convergence diagnosis and output analysis for MCMC. *R News*, 6(1):7–11.

Robert, C. and Casella, G. (2011). A Short History of Markov Chain Monte Carlo: Subjective Recollections from Incomplete Data. *Statistical Science*, 26(1):102–115.

Rossi, P. E., Allenby, G. M., and McCulloch, R. (2005). *Bayesian Statistics and Marketing*. Wiley.

Triantafyllopoulos, K. and Pikoulas, J. (2002). Multivariate Bayesian regression applied to the problem of network security. *Journal of Forecasting*, 21(8):579–594.

Trusov, M., Bodapati, A. V., and Bucklin, R. E. (2010). Determining Influential Users. *Journal of Marketing Research*, XLVII:643–658.

Tsikerdekis, M. (2013). Dynamic Voting Interface in Social Media: Does it Affect Individual Votes? In Boas, P. v. E., Groen, F. C. A., Italiano, G. F., Nawrocki, J., and Sack, H., editors, *SOFSEM 2013: Theory and Practice of Computer Science*, pages 552–563. Springer Berlin Heidelberg.

Volf, P., Jakubuv, J., Koranda, L., Sislak, D., Pechoucek, M., Mereu, S., Hilburn, B., and Nguyen, D. N. (2014). Validation of an Air-Traffic Controller behavioral model for fast time simulation. In *2014 Integrated Communications, Navigation and Surveillance Conference (ICNS) Conference Proceedings*, pages T1–1–T1–9. IEEE.

Wagenmakers, E.-J., Lee, M. D., Lodewyckx, T., and Iverson, G. (2008). Bayesian versus frequentist inference. In Hoijtink, H., Klugkist, I., and Boelen, P. A., ed-

itors, *Bayesian evaluation of informative hypotheses in psychology*, pages 181–207. Springer, New York.