Chapter 1

Introduction

Statistics has been developed since the past two centuries. Generally speaking, 19th century is Bayesian statistics and 20th century is frequentist statistics (Efron 2004). Frequentist approaches have dominated statistical theory and practice for most of the past century. Thanks to the fast development of computing facilities and new sampling techniques, in particular, Markov Chain Monte Carlo (MCMC) in the last two decades, Bayesian approach has become feasible and attracts scientists more and more attention in various applications.

1.1 Baye's rule

Bayesian statistical conclusions about parameters $\boldsymbol{\theta}$, or unobserved data $\boldsymbol{y}$, are made in terms of *probability* statements. These probability statements are conditional on the observed values of $\boldsymbol{y}$, which is denoted as $p(\boldsymbol{\theta}|\boldsymbol{y})$, called posterior distributions of parameters $\boldsymbol{\theta}$. Bayesian analysis is a practical method for making inferences from data and prior beliefs using probability models for quantities we observe and for quantities which we wish to learn. Below are three general steps for Bayesian data analysis:

1. Set up a full probability model $p(\boldsymbol{y}, \boldsymbol{\theta})$, i.e., a joint probability distribution for all observable and unobservable quantities;

2. Condition on observed data, calculate and interpret the posterior distributions (i.e., the conditional probability distribution of unobserved quantities of interest, giving the observed data $p(\boldsymbol{\theta}|\boldsymbol{y})$);

3. Evaluate the fit of the model and the implications of the resulting posterior distribution.

In order to make probability statements about $\boldsymbol{\theta}$ given $\boldsymbol{y}$, we must begin with a model providing a *joint probability distribution* for $\boldsymbol{\theta}$ and $\boldsymbol{y}$. This joint probability can be written as the product of $p(\boldsymbol{\theta})$ (*prior distribution*) and $p(\boldsymbol{y}|\boldsymbol{\theta})$ (*sampling distribution*),

$$p(\boldsymbol{\theta}, \boldsymbol{y}) = p(\boldsymbol{\theta})p(\boldsymbol{y}|\boldsymbol{\theta})$$

Conditional probability $p(\boldsymbol{\theta}|\boldsymbol{y})$ can be obtained by dividing both sides by $p(\boldsymbol{y})$.

$$p(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{p(\boldsymbol{\theta})p(\boldsymbol{y}|\boldsymbol{\theta})}{p(\boldsymbol{y})} \propto p(\boldsymbol{\theta})p(\boldsymbol{y}|\boldsymbol{\theta}) \propto \text{prior} \times \text{data information} \tag{1.1}$$

The primary task of any specific application is to develop model $p(\boldsymbol{\theta}, \boldsymbol{y})$ and perform necessary computations to summarize $p(\boldsymbol{\theta}|\boldsymbol{y})$ in appropriate ways.

1.2 Non-informative prior

Bayesian analysis requires prior information (see Section 1.1), however sometimes there is no particularly useful information before data are collected. In these situations, priors with "no information" are expected. Such priors are called *non-informative priors* or *vague priors*. In recent Bayesian literature, *reference priors* are more popularly used for fidelity reason, because any priors do have information. Anyway, *non-informative prior* is so called in the sense that it does not favor one value over another on the parameter space of the parameter(s) $\boldsymbol{\theta}$. Another reason to use non-informative priors is that one can connect the Bayesian modeling results with frequentist analysis.

The following presents some ways to construct non-informative priors.

1. Intuitively, flat over the parameter space. For example:
   (a) $X_i \sim N(\mu, \sigma^2), iid$ with σ^2 known. Then $p(\mu) \propto 1$.
2. Almost flat over the parameter space. In the last example, $p(\mu) \sim N(0, 10^6)$.
3. Due to distribution change through parameter transformation, flat distribution over one parameter may not be flat over its transformed parameter, e.g., if $\sigma^2 \sim$ uniform on $(0, 100)$, then $p(\sigma) \propto \sigma$, not uniform. Jeffrey's prior, which is invariant under transformation, $p(\boldsymbol{\theta}) \propto [I(\boldsymbol{\theta})]^{1/2}$ where $I(\boldsymbol{\theta})$ is the expected Fisher information in the model. For example,
   (a) $X \sim N(\mu, \sigma^2)$ with μ known. The Jeffrey's prior is $p(\sigma^2) \propto 1/\sigma^2$.
   (b) $X \sim Bin(n, \theta)$ with n known. The Jeffrey's prior is $p(\theta) \propto \theta^{-1/2}(1-\theta)^{-1/2}$, which is $Beta(\frac{1}{2}, \frac{1}{2})$.

For multiple parameters, the non-informative priors can be constructed by assuming "independence" among the parameters. For example, $p(\theta_1, \theta_2) = p(\theta_1)p(\theta_2)$ and each prior on the right hand side is the univariate non-informative prior. We can also use multivariate version of Jeffrey's prior, $p(\boldsymbol{\theta}) \propto |I(\boldsymbol{\theta})|^{1/2}$ where $|\cdot|$ denotes the determinant.

Note that non-informative prior may be improper, in that $\int p(\theta)d\theta = \infty$, but Bayesian inference is still possible as long as it leads to proper posterior.

1.3 Introduction to BUGS

BUGS (Bayesian inference Using Gibbs Sampling) is a piece of computer software for the Bayesian analysis of complex statistical models using Markov Chain Monte Carlo (MCMC) methods. It has been developing and maturing over the years, and is probably best known in its WinBugs incarnations. The latest version can run on Windows and Linux, as well as from inside the R statistical package.

1.3.1 WinBUGS

WinBUGS is part of the BUGS project, which aims to make practical MCMC methods available to applied statisticians. WinBUGS 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, although it can be called from other software. For a version that BUGS (BRugs) that sits within the R statistical package, see the OpenBUGS site. To obtain the software and learn more about it, please visit:
`http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml`

1.3.2 Open Bugs

The Windows version of OpenBUGS contains three seperate exectutable files: winbugs.exe for running the GUI Windows version, the shortcut BackBUGS for running a non interative script in WinBUGS and ClassicBUGS, a non Windows command line version of BUGS. BRugs, a set of R functions which reproduce the functionality of the GUI interface, is also avaliable to Windows users. The Linux version of OpenBUGS consists of a single shell script, LinBUGS, which provides the "ClassicBUGS" interface. At present the BRugs R functions do not work under Linux. To download the software and learn more, please go to:
`http://mathstat.helsinki.fi/openbugs/`

Reference

1. Efron B. (2004) Presidential address in JSM, 2004.

Chapter 2

Hierarchical normal / normal model

This chapter provides a concrete example of Bayesian hierarchical (multilevel) normal / normal model for longitudinal data. From this example, we will get hands-on experience about how to draw MCMC samples from posterior distributions using WinBUGS and BRugs package in R. All the following chapters will incorporate these aspects: (1) a concrete example with data publicly available; (2) mathematical model formulae statisticians are familiar with; (3) brief Bayesian theory if necessary; (4) programs ready to run; (5) interpretation of modeling results; (6) references for further investigation.

Hierarchical normal / normal model is analogous to mixed model, however in Bayesian world, there are no fixed effects because all parameters are treated as random with distributions.

2.1 Data

Data are obtained from WinBUGS (Spielhalter et al. 2002) example volume I (`http://www.mrc-bsu.cam.ac.uk/bugs`), originally from Gelfand et al. (1990). 30 young rats' weights were measured weekly for five weeks (Figure 2.1). For illustration purpose in the later part of this chapter, we add an artificial treatment group variable `trt` and assign the first half (15) rats to the first treatment group and the other half to the second treatment group. Denote Y_{ij} as the weight of the ith rat measured at age x_j. The data is available at the IBC homepage. `http://biostat.mc.vanderbilt.edu/BayesianDataAnalysisWithOpenBUGSAndBRugs` and

2.2 Random effects model

The data suggest a growing pattern with age with a little downward curvature. For now we assume a linear model (2.1) with random effects to account for the subject-specific growth pattern. You may want to model the nonlinear pattern using restricted cubic spline (Harrell 2001). The programming code is provided for the restricted cubic spline model at the end of the chapter.

$$\begin{aligned} Y_{ij} &\sim N(a_i + \beta trt_i + b_i(x_j - \bar{x}), \tau_0^{-1}) \\ a_i &\sim N(\mu_a, \tau_a^{-1}) \end{aligned}$$

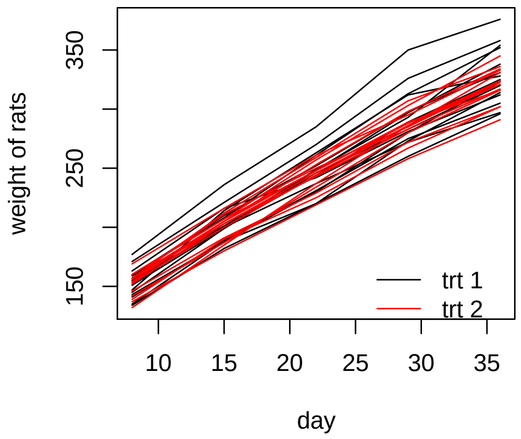

Figure 2.1: Rats data in hierarchical normal model

$$b_i \quad \sim \quad N(\mu_b, \tau_b^{-1}) \tag{2.1}$$

where $\bar{x} = 22$, the average of x, trt_i is the group assignment for rat i, and τ_0, τ_a, τ_b are precisions (1/variance) for the corresponding normal distributions. For now, we standardize the x_j's around their mean to reduce dependence between two random effects a_i and b_i in their likelihood. This model suggests that for each subject (i.e., fix random effects a_i and b_i, and group trt_i), the growth curve is linear with noise precision τ_0. The group effect can be captured by β.

A little about Bayesian notation: In Bayesian models, precisions or precision matrices are more commonly used than variances or covariance matrices. In BUGS language, `Normal(0, tau)` means τ is precision NOT variance, which is different from the common textbooks.

2.3 Prior and hyperprior

The above model is not a fully Bayesian model, because it can be treated as a typical mixed model with fixed effects `intercept`, `day`, `trt` and random effects `intercept`, `day`. This mixed model can be fitted using popular statistical software, e.g., SAS (`Mixed` procedure) and R (`nlme` library). A fully Bayesian model needs additional equipments, priors and / or hyperpriors. Bayesian inference is from the posterior distribution based on both prior beliefs $p(\boldsymbol{\theta})$ and data-based likelihood $p(\boldsymbol{y}|\boldsymbol{\theta})$. Now let's look at model (2.1) in Bayesian way. The first equation in model (2.1) specifies the likelihood and the other two specify priors for $\boldsymbol{a}$ and $\boldsymbol{b}$ through another level of parameters μ_a, μ_b, τ_a, and τ_b. The other priors need to specify are for the error precision τ_0 and β. Because we do not have informative belief about them, vague priors are desired. One type of vague prior is Gamma(ϵ, ϵ) (mean 1, variance $1/\epsilon$) (Gelman's book) for ϵ fairly small, e.g., 0.001, and $\beta \sim N(0, 10^{-6})$.

After we specify all priors for parameters, we may also need to further specify the priors for the parameters in the priors, e.g., μ_a, μ_b, τ_a, and τ_b in model (2.1), which are called hyperpriors. In most cases, the hyperpriors are vague. In this model, the vague hyperpriors are specified as follows $\mu_a, \mu_b \sim N(0, 10^{-6})$ and $\tau_a, \tau_b \sim \text{Gamma}(\epsilon, \epsilon)$. As a summary the fully Bayesian model (2.1) consists of three levels: data-based likelihood level $p(\boldsymbol{y}|\boldsymbol{\theta})$, prior level $p(\boldsymbol{\theta}|\boldsymbol{\psi})$, and hyperprior level $p(\boldsymbol{\psi})$. Complex models may involve more levels, but models with more than four levels are unusual and unhelpful. The higher the level is, the more contribution to the posterior inference, so the likelihood provides the most information, then the prior, then the hyperprior. In clinical trials, as data cumulates during the trial, the prior's effect on the posterior becomes less.

2.4 BUGS program

Throughout this course, we only focus on BUGS language for it is very convenient and easy to program. We recommend use it whenever possible. BUGS is a highly-structured language and users do not have a lot control unlike R and C. Both WinBUGS standalone and BRugs in R share the same code, however WinBUGS will be used first because of its user-friendly interface.

```
model
{
  #likelihood p(Y|theta)
for( i in 1 : N ) {
for( j in 1 : T ) {
Y[i , j] ~ dnorm(mu[i , j],tau.0)
mu[i , j] <- a[i] + beta * trt[i] + b[i] * (x[j] - xbar)
}
#Prior p(theta|Psi)
a[i] ~ dnorm(mu.a, tau.a)
b[i] ~ dnorm(mu.b, tau.b)
}
#prior
tau.0 ~ dgamma(0.001,0.001)
beta ~ dnorm(0.0,1.0E-6)

#hyper-priors
mu.a ~ dnorm(0.0,1.0E-6)
mu.b ~ dnorm(0.0,1.0E-6)
tau.a ~ dgamma(0.001,0.001)
tau.b ~ dgamma(0.001,0.001)

#parameters of interest
sigma <- 1 / sqrt(tau.0) #error sd
w0[1] <- mu.a - xbar * mu.b #weight at birth for 1st group
w0[2] <- mu.a + beta - xbar * mu.b #weight at birth for 2nd group
}
```

After write the model structure, the next step is to provide data. The data can be written in two formats, a list or a table. The list format can be created from R using `dput` command. The data for this program is as follows.

```
list(x = c(8.0, 15.0, 22.0, 29.0, 36.0), xbar = 22, N = 30, T = 5,
Y = structure(
.Data =   c(151, 199, 246, 283, 320,
 145, 199, 249, 293, 354,
 147, 214, 263, 312, 328,
 155, 200, 237, 272, 297,
 135, 188, 230, 280, 323,
 159, 210, 252, 298, 331,
 141, 189, 231, 275, 305,
 159, 201, 248, 297, 338,
 177, 236, 285, 350, 376,
 134, 182, 220, 260, 296,
 160, 208, 261, 313, 352,
 143, 188, 220, 273, 314,
 154, 200, 244, 289, 325,
 171, 221, 270, 326, 358,
 163, 216, 242, 281, 312,
 160, 207, 248, 288, 324,
 142, 187, 234, 280, 316,
 156, 203, 243, 283, 317,
 157, 212, 259, 307, 336,
 152, 203, 246, 286, 321,
 154, 205, 253, 298, 334,
 139, 190, 225, 267, 302,
 146, 191, 229, 272, 302,
 157, 211, 250, 285, 323,
 132, 185, 237, 286, 331,
 160, 207, 257, 303, 345,
 169, 216, 261, 295, 333,
 157, 205, 248, 289, 316,
 137, 180, 219, 258, 291,
 153, 200, 244, 286, 324),
.Dim = c(30,5)))
```

It's very convenient to create data from R, but be careful about two issues: (1) list data obtained from R do not have the required `.Data` keyword for BUGS. Add this keyword for BUGS. (2) BUGS reads matrix in a different way from R. For example, there is a matrix $M : 5 \times 3$ in R. In order to use it in BUGS, follow this procedure: (a) transpose M: `M <- t(M)`; (b) dump M: `dput(M, "M.dat")`; (c) open `M.dat`, add `.Data` keyword and change `.Dim = c(3,5)` to `.Dim = c(5,3)`.

Table data have the format:

```
n[] x[]
47   0
148 18
119  8
END
```

MCMC algorithm needs to be initialized. The last step for programming is to initialize the model. BUGS may automatically generate initial values, but it is highly recommended to provide initial values for fixed effects. Good initial values potentially improve convergence. For this model, the fixed effects are $\mu_a, \mu_b, \beta, \tau_0, \tau_a$, and τ_b. So it is recommended to initialize at least these parameters. All the other parameters can be initialized by BUGS, in this model they are $\boldsymbol{a}$ and $\boldsymbol{b}$. The BUGS code and data are available at the IBC homepage.

```
list(mu.a = 150, mu.b = 10, beta=0, tau.0 = 1, tau.a = 1, tau.b = 1)
```

2.5 Procedure to run BUGS code

2.5.1 WinBUGS

Launch WinBUGS. The icon, which resembles a spider, is in the directory where WinBUGS was installed. After create a shortcut and place it on the desktop, double click the spider icon to launch WinBUGS. Read the license agreement and close it. Open a new file and save it as WinBUGS document (`.odc`).

Check code. Download the above data and code from IBC homepage, then copy-paste or type them on your new WinBUGS document. Pull down the `Model` menu, then select `Specification`. Highlight or double click `list` in the model code, and click `check model` button on the `specification` tool. If the model is correct, "`model is syntactically correct`" will appear at the bottom line of your document. If not correct, the cursor is positioned after the symbol that caused the error. Then highlight or double click `list` in the data code, and click `load data`. If load correctly, "`data loaded`" will appear. Then compile your model by click `compile` button and select the number of MCMC chains. The last step is to initialize the model by click `initialize` button. If only part of parameters are initialized, WinBUGS can generate the other required initial values by clicking `gen inits` button.

Run the code. MCMC needs burn-in period, i.e., samples before convergence. Pull down `Model` menu and click `Update`. On a small pop-up window, click `update` button. Choose the number of burn-in samples. The default number is 1000. Then pull down `specification` menu, click `Samples`. Type parameters of interest and click `set` button. These parameters can be monitored during the program run to check convergence. The commonly-used statistical inference for all parameters in the model is available on the `samples` menu.

2.5.2 OpenBUGS

`OpenBUGS` can be run both in Windows and in Unix systems, however the current version of `BRugs` package only work on Windows. The running procedure of using `BRugs` in R is pretty much the same as in WinBUGS, except that `BRugs` only read text files. Following's the steps to run the above BUGS code.

1. Create three text files namely `ratsmodel.txt`, `ratsdata.txt`, `ratsinits.txt` and save the three pieces of code in these files, respectively.

2. loading BRugs: `library(BRugs)`

3. Check code: `modelCheck("ratsmodel.txt")`

4. Load data: `modelData("ratsdata.txt")`

5. Compile: `modelCompile(numChains=2)`

6. Initialize model: `modelInits(rep("ratsinits.txt", 2))`

7. Burn-in: `modelUpdate(1000)`

8. Monitor samples: `samplesSet(c("w0", "beta"))`
9. More samples: `modelUpdate(1000)`
10. Statistical inference and plots are also available (see BRugs package information).

2.6 Results and interpretation

Suppose we are particularly interested in two aspects in this data. One is treatment effect β and another is the average birth weight $\boldsymbol{w}_0$ for two groups. In order to get inference about these two quantities, they need to be available in the BUGS code. The posterior densities of these parameters can be estimated by the MCMC samples after convergence. The statistical inference may be drawn from the posterior 95% credible intervals (CI). Since 95%CI of β covers 0, there is no significant difference between these two groups at .05 level. As a conclusion, once we have the distribution of a parameter of interest, we completely know that parameter in statistical sense, so we can do whatever inference from it.

Reference

1. Gelfand, A.E., Hills, S., Racine-Poon, A., and Smith, A.F.M. (1990) Illustration of Bayesian Inference in Normal Data Models Using Gibbs Sampling. *Journal Amer. Stat. Assoc.*, **85**:972-985.
2. Spielhalter, D., Thomas, A., Best, N., and Lunn, D. (2002) *WinBUGS User Manual Version 1.4*, Cambridge, UK: MRC Biostatistics Unit.
3. Harrell, F.E. (2001) *Regression modeling Strategies With Applications to Linear Models, Logistic Regression, and Survival Analysis.* Springer.
4. Gelman A., (2003) *Bayesian Data Analysis* CRC press.