MT4531: Bayesian Inference
MT5731: Advanced Bayesian Inference
Course Notes - part 2

Semester 1, 2021

Lecturer and Module co-ordinator: Dr Nicolò Margaritella

Notes compiled on: November 16, 2021

Rev. Thomas Bayes
(Probably)

University of St. Andrews

School of Mathematics and Statistics

CONTENTS

2. Bayesian Computation and Model Comparison 1
2.1 The BUGS language 1
2.2 Gibbs Sampler 7
2.2.1 Example 7
2.3 Metropolis-Hastings Algorithm 9
2.3.1 The Independence Sampler 12
2.3.2 Single-updates and the Gibbs Sampler 12
2.4 Comparison of Gibbs sampler and Metropolis-Hastings algorithm 14
2.5 Data Augmentation 15
2.5.1 Hierarchical Models 17
2.6 Bayes Factors - Choice of hypothesis 19
2.6.1 Simple hypotheses 21
2.6.2 Composite hypotheses 21
2.7 Model Discrimination 23
2.7.1 Bayes Factors 24
2.7.2 Deviance Information Criterion 28
2.7.3 The Watanabe - Akaike (WAIC) criterion 31
2.8 Direct Sampling 32
2.8.1 Method of Inversion 32
2.8.2 Rejection sampling 33
2.8.3 Importance sampling 37

A. Probability Distributions 41
A.1 Discrete distributions 41
A.2 Continuous distributions 42

B. Nimble 43
B.1 NIMBLE: Numerical Inference for Statistical Models for Bayesian and Likelihood Estimation 43
B.2 What is required for using the R package Nimble 43

Chapter 2

BAYESIAN COMPUTATION AND MODEL COMPARISON

In the last chapter we described the ideas behind Bayesian statistics and saw a number of (simple) examples. However, real statistical problems are typically significantly more complex, most notably in terms of the number of unknown parameters to be estimated, resulting in high dimensional posterior distributions of complex and often of non-standard form. This was essentially the reason why the classical approach dominated statistics until the 1990s – Bayesian statistics were fine in theory, but intractable in practice. However, the increase in computational power, coupled with computational algorithms introduced to the statistical literature has made Bayesian analyses feasible (and often easier than alternative classical approaches!). In this half of the course we describe these algorithms and the computer language `BUGS` (Bayesian inference Using Gibbs Sampling). BUGS is implemented in the free package `Nimble`. Nimble is an `R` statistical package that allows users to fit complex statistical models using the BUGS language (and more, beyond the scope of this module).

We will demonstrate in lectures how to utilise the BUGS language to fit Bayesian models using `Nimble`. This is the scope of the next section, along with the lecture demonstrations and code provided in Moodle. For more details...

... a guide to using `Nimble` is available on Moodle. Appendix B contains instructions on what is required for running `Nimble` within R, other than R studio of course. One of the appeals of using Nimble through R is that it is compatible with operating systems other than Windows. Further materials and examples can be found on the website *https://r-nimble.org/*.

2.1 The `BUGS` language

It is perhaps easiest to demonstrate the BUGS language via the following example.

Example - Rats

The BUGS code is composed of 2 or 3 separate components:

1. Model specification - containing the likelihood and prior specification in distribution form;
2. Data - input data and model constants; and
3. Initial starting value of the Markov chain, i.e. an initial starting value for each parameter (optional - but it can be necessary for some prior distributions).

We consider an example relating to the weights of rats. Details can be found via the `Examples menu` (http://www.openbugs.net/Manuals/Contents.html), and `Examples Vol I`, `Rats`. However,

note that we use a simpler model than the one presented in `OpenBUGS`[1]. The data and model specification are presented before the associated BUGS code.

Data

The data correspond to the weight of 30 rats on 5 different days. We can display the data in the form given in Table 2.1.

	Day (x_j)				
Rat	8	15	22	29	36
1	151	199	246	283	320
2	145	199	249	293	354
3	147	214	263	312	328
4	155	200	237	272	297
⋮					
30	153	200	244	286	324

Tab. 2.1: The weights of the rats at each time.

Model

Let Y_{ij} denote the weight of rat $i = 1, \ldots, 30$ at time $j = 1, \ldots, 5$. We assume a very simple linear model, where weight is linearly regressed on time, and

$$Y_{ij} \sim N(\alpha + \beta z_j, \sigma^2),$$

where α, β and σ^2 are parameters to be estimated; and z_j denotes the jth (normalised) time given by

$$z_j = \frac{x_j - \text{mean}(x)}{\text{sd(x)}}.$$

(This is not a very realistic model!).

Note: Normalising the data corresponding to the explanatory variable(s) can be useful for comparability, interpretability and MCMC convergence. Normalisation requires that we take each set of explanatory variables, subtract the sample mean from each value and divide by the sample standard deviation.

Priors

We assume that we have no prior information on the parameters, and specify:

$$\alpha \sim N(0, 10^5); \qquad \beta \sim N(0, 10^5); \qquad \sigma^2 \sim \Gamma^{-1}(0.001, 0.001).$$

Note that in BUGS the Normal distribution is parametrised via the precision ($\tau = 1/\sigma^2$).

[1] `OpenBUGS` and `WinBUGS` were the first and most widely used software for implementing Bayesian analysis. More recent tools for Bayesian analysis using MCMC include `Nimble`, `JAGS`, `STAN`, and many others.

BUGS code

The BUGS code contains the model, data and initial values. The code below can be found in Moodle. The model component provides the likelihood and priors specified in distributional form.

```
# Specify the likelihood:
for(i in 1 : N) {
  for(j in 1 : T) {
    Y[i,j] ~ dnorm(mu[i,j],tau)
    mu[i,j] <- alpha + beta*(x[j] - mean(x[]))/sd(x[])
  }
}

# Prior specification:
alpha ~ dnorm(0,0.00001)
beta ~ dnorm(0,0.00001)
tau ~ dgamma(0.001,0.001)
sigma2 <- 1 / tau
```

(# denotes a comment in BUGS code).

Data

The data component simply provides the data, including, in this case, values for $\boldsymbol{Y}$ (responses), $\boldsymbol{x}$ (explanatory variable), N (number of rats) and T (number of times). The data are input using a `list()` function. For example, we can read in the data using the following:

```
# Data:
list(x = c(8.0, 15.0, 22.0, 29.0, 36.0),
Y = t(matrix(  c(151, 199, 246, 283, 320,
                          145, 199, 249, 293, 354,
                          147, 214, 263, 312, 328,
                          ......................
                          153, 200, 244, 286, 324),
                          5,30)))
```

Initial parameter values

Finally we need to provide the initial parameter values for all the parameters. In this example, we specify initial values for α, β, and τ (recall $\tau = 1/\sigma^2$ since BUGS works on precision). Note that we need to specify initial values on the parameters that have a stochastic node in BUGS (i.e. for each parameter which we specify a prior distribution for). For example, possible initial values could be specified as follows:

```
# Input initial parameter values:
list(alpha = 0, beta = 0, tau = 1)
```

Note: Initial parameter values can, in general, be generated automatically within `Nimble`, by sampling from the prior distributions specified on the parameters (but user-specified initial values can give more control on the detection of convergence problems).

Running BUGS code within `Nimble`

Running the BUGS code is simple with `Nimble`, as you do it by giving a series of R commands. See the lecture demonstration and the code uploaded in Moodle for a standard set of commands. See also the `Nimble` manual uploaded in Moodle.

Convergence diagnostics and results

The simulations are run for 5000 iterations. The corresponding trace and density plots of the parameters α, β, σ^2 and τ are provided in Figure 2.1. Trace plots suggest that convergence of the Markov chains to the stationary distribution is very rapid. Marginal density plots show unimodal posterior distributions.

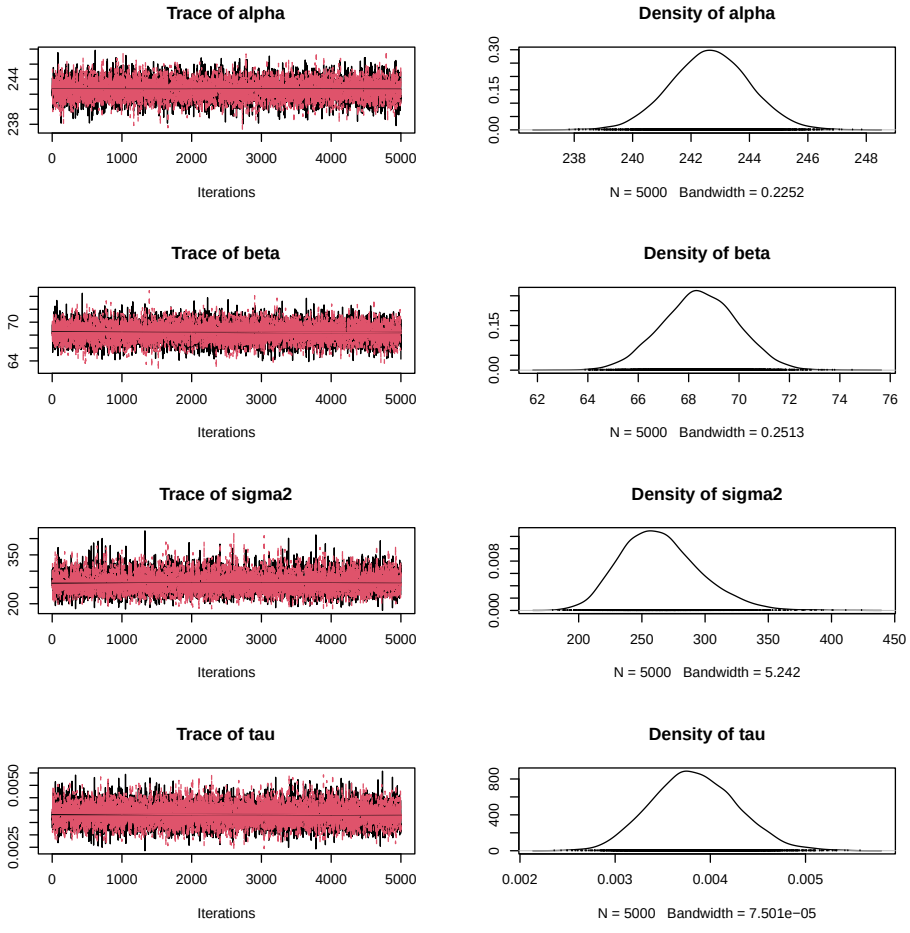


Fig. 2.1: Raw trace plots for the parameters α, β, σ^2 and τ.

We run multiple chains from different starting points ($\alpha = 1, \beta = 1, \sigma^2 = 10$, $\alpha = -10, \beta = -10, \sigma^2 = 100$ and $\alpha = 100, \beta = -100, \sigma^2 = 0.1$). The corresponding BGR trace plots are provided

in Figure 2.2. The BGR statistic is plotted in black and the 97.5% CI upper bound in red. The BGR statistics should converge to 1 with a narrow credible interval. Convergence is very rapid; however, we select a burn-in of 1,000 iterations which is clearly very conservative in this scenario.

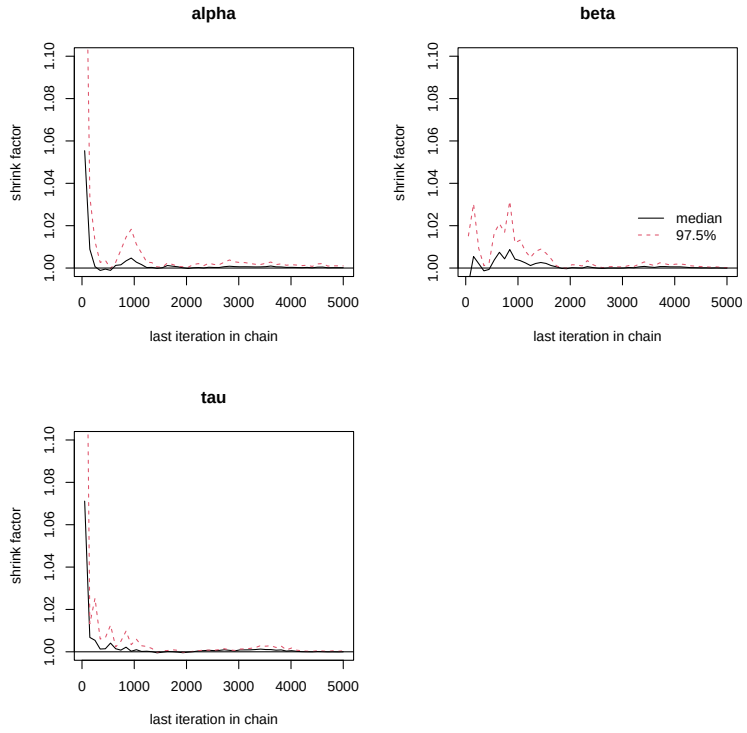


Fig. 2.2: BGR convergence diagnostic plots for the parameters α, β and σ^2.

The autocorrelation plots (not shown) show correlation plunging to almost zero already by lag 2; this is mirrored in the number of effective samples which is almost equal to that of the samples collected after burn-in in the three chains.

```
> effectiveSize(combinedchains)
alpha     beta   sigma2      tau
12000.00 12000.00 11520.65 11581.87
```

The (default) summary statistics are given by,

```
> posterior$summary$all.chains
           Mean   Median St.Dev. 95%CI_low 95%CI_upp
alpha  242.6465 242.6557  1.3410  240.0528  245.2919
beta    68.4461  68.4460  1.4842   65.5503   71.3565
sigma2 264.4997 262.0922 31.2660  211.3259  332.7252
```

The first column provides the name of the parameters with the 2nd, 3rd and 4th columns the corresponding posterior (marginal) mean, median and standard deviation, respectively. The next 2 columns provide the 2.5%, and 97.5% posterior quantiles, providing the posterior symmetric 95% credible interval. We can obtain the Monte Carlo error (`Time-series SE`) by using `summary()` on the combined chains (see R code on Moodle). In this case, the values (0.0122, 0.0135 and 0.292 for

alpha, beta and sigma2, respectively) all appear to be very small compared to the posterior mean (so that if the simulations are replicated we would expect only a very small deviation from the current values) and posterior SD.

We note that α can be interpreted as the intercept term (after the weights have been normalised) and β the slope of the linear regression. In particular, since β is clearly positive, there is strong evidence that there is a positive relationship between the weight of the rat and time, so that the weight of the rat increases with time (which we assume is linear in nature). In addition we have that the 95% symmetric credible interval for β does not contain the value of 0 which would correspond to no (linear) relationship between weight and time. Thus an *ad-hoc* interpretation of this output might conclude that there is evidence that time is important in this regression (note we return to this issue in §2.7 where we conduct a formal model selection procedure or hypothesis test).

Prior Sensitivity Analysis

We conduct a prior sensitivity analysis by rerunning the MCMC iterations in `Nimble` using different priors on each of the parameters. In particular we specify the following priors,

$$\alpha \sim N(0, 10^3); \qquad \beta \sim N(0, 10^3); \qquad \sigma \sim U[0, 100].$$

In `Nimble` these could be specified as follows,

```
# Priors:
    alpha ~ dnorm(0,0.001)
    beta ~ dnorm(0,0.001)
    tau <- 1/(sigma^2)
    sigma ~ dunif(0,100)
```

Note that with this change in prior we need to also change the inputting of the initial parameter values as we need to specify initial values for the parameters that are defined to have a prior distribution (α, β and σ instead of τ).

```
# Input initial parameter values:
list(alpha = 0, beta = 0, sigma = 1)
```

In addition if we want to monitor and record σ^2 we also need to include within the model specification `sigma2 <- sigma*sigma`.

Once more convergence is extremely swift and we obtain the corresponding posterior summary statistics:

```
         Mean    Median   St.Dev. 95%CI_low  95%CI_upp
alpha  242.2281 242.2514  1.3357  239.6435  244.8147
beta    68.3180  68.3080  1.4937   65.4284   71.2769
sigma2 265.8298 263.3831 31.4475  210.9933  335.7229
```

These are again very similar to the previous posterior summary statistics. Thus we would typically conclude that the posterior distribution is data-driven.

We now describe the most common algorithms for simulating a Markov chain with a given stationary distribution.

2.2 Gibbs Sampler

The Gibbs sampler works as follows. Given a vector variable $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p) \in \mathbb{R}^p$ with distribution $\pi(\boldsymbol{\theta})$, the Gibbs sampler uses the set of full conditionals of π to sample indirectly from the full posterior distribution. (Within our Bayesian context, π is the posterior distribution of interest but we have dropped the conditioning on the data, $\boldsymbol{x}$, for notational simplicity).

Let $\pi(\theta_i|\boldsymbol{\theta}_{(i)})$ denote the induced full conditional of θ_i, given the values of the other components $\boldsymbol{\theta}_{(i)} = (\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_p)$, (and the data). Then, we initially set arbitrary starting values $\boldsymbol{\theta}^0 = (\theta_1^0, \ldots, \theta_p^0)$. Given the Markov chain is in state $\boldsymbol{\theta}^t$ at iteration t of the Markov chain, the Gibbs sampler successively makes random drawings from the full conditional distributions $\pi(\theta_i|\boldsymbol{\theta}_{(i)})$, $i = 1, \ldots, p$ as follows:

$$
\begin{array}{lll}
\theta_1^{t+1} & \text{is sampled from} & \pi(\theta_1|\theta_2^t, \ldots, \theta_p^t) \\
\theta_2^{t+1} & \text{is sampled from} & \pi(\theta_2|\theta_1^{t+1}, \theta_3^t, \ldots, \theta_p^t) \\
\quad . & \quad . & \quad . \\
\quad . & \quad . & \quad . \\
\quad . & \quad . & \quad . \\
\theta_i^{t+1} & \text{is sampled from} & \pi(\theta_i|\theta_j^{t+1},\ j < i \text{ and } \theta_j^t, j > i) \\
\quad . & \quad . & \quad . \\
\quad . & \quad . & \quad . \\
\quad . & \quad . & \quad . \\
\theta_k^{t+1} & \text{is sampled from} & \pi(\theta_p|\theta_1^{t+1}, \ldots, \theta_{p-1}^{t+1}).
\end{array}
$$

This completes a transition from $\boldsymbol{\theta}^t$ to $\boldsymbol{\theta}^{t+1}$. Iteration of the full cycle of random variate generations from each of the full conditionals in turn, produces a sequence $\boldsymbol{\theta}^0, \boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^t, \ldots, \boldsymbol{\theta}^T$. The values $\boldsymbol{\theta}^0, \ldots, \boldsymbol{\theta}^N$ are discarded as burn-in, for some suitable value of N (see Section 1.12.2), and the values $\boldsymbol{\theta}^{N+1}, \ldots, \boldsymbol{\theta}^T$ can be used to obtain Monte Carlo estimates of interest.

The transition kernel for going from $\boldsymbol{\theta}^t$ to $\boldsymbol{\theta}^{t+1}$ is given by

$$
\mathcal{K}_G(\boldsymbol{\theta}^t, \boldsymbol{\theta}^{t+1}) = \prod_{i=1}^{k} \pi(\theta_i^{t+1}|\theta_j^{t+1},\ j < i \text{ and } \theta_j^t,\ j > i), \tag{2.1}
$$

and has stationary distribution π.

Thus, in two dimensions a typical trajectory of the Gibbs sampler may look something like that given in Figure 2.3.

2.2.1 Example

Consider observed data $x_1, \ldots, x_n$ such that, given μ and σ, each $X_i \overset{iid}{\sim} N(\mu, \sigma^2)$, where both μ and σ^2 are unknown. We specify independent priors:

$$
\mu \sim N(\phi, \tau^2); \qquad \text{and} \qquad \sigma^2 \sim \Gamma^{-1}(\alpha, \beta).
$$

Then we have that,

$$
\begin{aligned}
\mu|\boldsymbol{x}, \sigma^2 &\sim N\left(\frac{\tau^2 n\bar{x} + \sigma^2\phi}{\tau^2 n + \sigma^2}, \frac{\sigma^2\tau^2}{\tau^2 n + \sigma^2}\right); \\
\sigma^2|\boldsymbol{x}, \mu &\sim \Gamma^{-1}\left(\frac{n}{2} + \alpha, \frac{1}{2}\sum_{i=1}^{n}(x_i - \mu)^2 + \beta\right),
\end{aligned}
$$

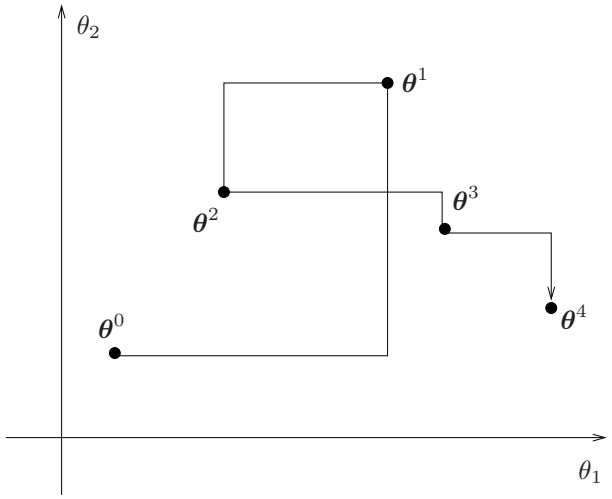


Fig. 2.3: Typical Gibbs sampler sample path.

(see §1.11 and Chapter 1 for each parameter - recall that for the posterior conditional distributions we condition on all other parameters, and so treat them as if they were fixed). Thus, setting initial parameter values for μ^0 and $(\sigma^2)^0$, we would implement the Gibbs sampler by simulating,

$$\begin{aligned}
\mu^{t+1}|\boldsymbol{x}, (\sigma^2)^t &\sim N\left(\frac{\tau^2 n\bar{x} + (\sigma^2)^t\phi}{\tau^2 n + (\sigma^2)^t}, \frac{(\sigma^2)^t\tau^2}{\tau^2 n + (\sigma^2)^t}\right); \\
(\sigma^2)^{t+1}|\boldsymbol{x}, \mu^{t+1} &\sim \Gamma^{-1}\left(\frac{n}{2} + \alpha, \frac{1}{2}\sum_{i=1}^{n}(x_i - \mu^{t+1})^2 + \beta\right).
\end{aligned}$$

In other words:

Step 1. Set initial parameter value for μ and σ^2 denoted by $(\mu, \sigma^2)^0 = \{\mu^0, (\sigma^2)^0\}$.

Step 2. Conditional on the current parameter value, $(\sigma^2)^t$, generate a new value for μ, from the posterior conditional distribution,

$$\mu^{t+1}|\boldsymbol{x}, (\sigma^2)^t \sim N\left(\frac{\tau^2 n\bar{x} + (\sigma^2)^t\phi}{\tau^2 n + (\sigma^2)^t}, \frac{(\sigma^2)^t\tau^2}{\tau^2 n + (\sigma^2)^t}\right).$$

Step 3 Conditional on the newly updated parameter value, μ_{t+1}, generate a new value for σ^2, from the posterior conditional distribution,

$$(\sigma^2)^{t+1}|\mu^{t+1}, \boldsymbol{x} \sim \Gamma^{-1}\left(\frac{n}{2} + \alpha, \frac{1}{2}\sum_{i=1}^{n}(x_i - \mu^{t+1})^2 + \beta\right).$$

Step 4. Increase t by one and return to step 2, until T iterations have been performed.

Note that the ordering of the parameters is unimportant so we could reverse the order of updating

each parameters and use the updating scheme,

$$\begin{aligned}(\sigma^2)^{t+1}|\boldsymbol{x}, \mu^t &\sim \Gamma^{-1}\left(\frac{n}{2}+\alpha, \frac{1}{2}\sum_{i=1}^{n}(x_i-\mu^t)^2+\beta\right)\\ \mu^{t+1}|\boldsymbol{x}, (\sigma^2)^{t+1} &\sim N\left(\frac{\tau^2 n\bar{x}+(\sigma^2)^{t+1}\phi}{\tau^2 n+(\sigma^2)^{t+1}}, \frac{(\sigma^2)^{t+1}\tau^2}{\tau^2 n+(\sigma^2)^{t+1}}\right).\end{aligned}$$

Remember that within an iteration, we always use the "current" value of the other parameters - this corresponds to the updated value (i.e. the $(t+1)$th value) if we have already updated the parameter within the iteration of the Markov chain OR the value of the parameter in the previous iteration (i.e. the tth value).

For example, setting prior parameters $\phi = 0$, $\tau = 1$, $\alpha = 0.1$ and $\beta = 0.01$ and simulating $x_1, \ldots, x_{10}$ from a standard normal distribution (so the true values are $\mu = 0$ and $\sigma^2 = 1$), Figure 2.4 provides the corresponding (marginal) trace plots of a Gibbs sampler for parameters μ and σ^2 for 100 iterations.

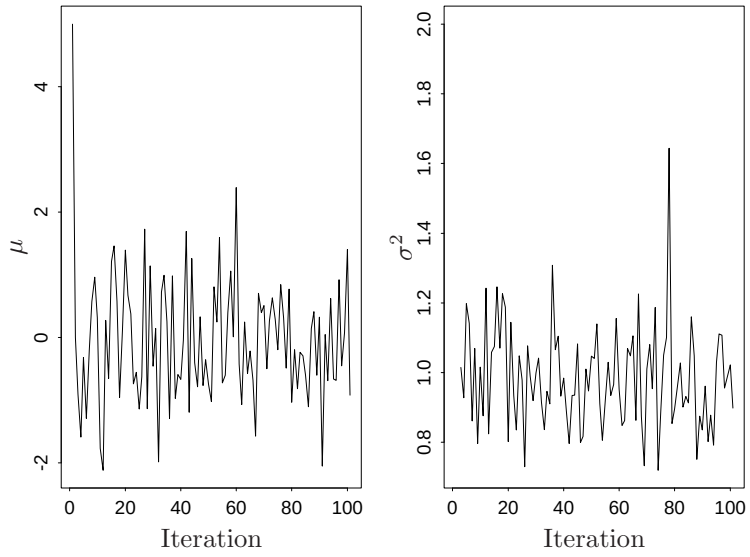


Fig. 2.4: Output from the Gibbs sampler, showing the sample paths for the variables μ and σ^2.

Conceptually, the Gibbs sampler appears to be a rather straightforward algorithmic procedure. Ideally, each of the conditionals will be of the form of a standard distribution and suitable prior specification often ensures that this is the case (for example, use of conjugate priors). However, in the cases where one or more of the conditionals is non-standard there are many ways to sample from univariate conditionals (e.g. direct sampling methods) however many of these algorithms are generally computationally intensive and inefficient. An alternative (and standard) approach is to use the Metropolis-Hastings algorithm for non-standard posterior conditionals.

2.3 Metropolis-Hastings Algorithm

A general way to construct MCMC samplers is as a form of generalised rejection sampling, where values are drawn from approximate/proposal distributions and "corrected" in order that, asymptotically, they are random observations from the target distribution. This is the motivation for methods

such as the Metropolis-Hastings algorithm which sequentially draws candidate observations from a distribution, conditional only upon the last observation, thus inducing a Markov chain. The most important aspect of such algorithms is not the Markov property, but the fact that the approximating candidate distributions can be improved at each step in the simulation.

The method commonly known as the Metropolis-Hastings algorithm, is based upon the observation that given a Markov chain with transition kernel $\mathcal{K}(\boldsymbol{\theta}, \boldsymbol{\phi})$ (intuitively, the function that shows how likely it is that the chain moves from $\boldsymbol{\theta}$ to $\boldsymbol{\phi}$) and exhibiting detailed balance for π i.e.,

$$\pi(\boldsymbol{\theta})\mathcal{K}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \pi(\boldsymbol{\phi})\mathcal{K}(\boldsymbol{\phi}, \boldsymbol{\theta}), \tag{2.2}$$

the chain has stationary density, $\pi(\cdot)$.

The candidate generating density (or proposal density) typically depends upon the current state of the chain, and we denote it by $q(\boldsymbol{\phi}|\boldsymbol{\theta}^t)$. The choice of proposal density is essentially arbitrary. For the induced chain to satisfy the detailed balance condition of (2.2), we introduce an acceptance function $\alpha(\boldsymbol{\theta}^t, \boldsymbol{\phi})$. (See Question 4 in Tutorial 4 for a proof.)

We accept the candidate observation, and set $\boldsymbol{\theta}^{t+1} = \boldsymbol{\phi}$, with probability $\alpha(\boldsymbol{\theta}^t, \boldsymbol{\phi})$; else if the candidate observation is rejected, the chain remains at $\boldsymbol{\theta}^t$, so that $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t$.

It can be shown (this is not part of this module) that the optimal form for the acceptance function, in the sense that suitable candidates are rejected least often and computational efficiency is maximised, is given by

$$\alpha(\boldsymbol{\theta}^t, \boldsymbol{\phi}) = \min\left(1, \frac{\pi(\boldsymbol{\phi})q(\boldsymbol{\theta}^t|\boldsymbol{\phi})}{\pi(\boldsymbol{\theta}^t)q(\boldsymbol{\phi}|\boldsymbol{\theta}^t)}\right).$$

Then,

$$\mathcal{K}(\boldsymbol{\theta}^t, \boldsymbol{\phi}) = q(\boldsymbol{\phi}|\boldsymbol{\theta}^t)\alpha(\boldsymbol{\theta}^t, \boldsymbol{\phi}).$$

The notation above can be simplified using $\boldsymbol{\theta}$ instead of $\boldsymbol{\theta}^t$, as $\boldsymbol{\theta}^t$ is the value we have currently assigned to $\boldsymbol{\theta}$. The Metropolis-Hastings method can be written algorithmically as follows.

Step 1. Set initial parameter value for $\boldsymbol{\theta}$ denoted by $\boldsymbol{\theta}^0$.

Step 2. Given the current position, $\boldsymbol{\theta} = \boldsymbol{\theta}^t$, generate a new value, $\boldsymbol{\phi}$, from the distribution $q(\boldsymbol{\phi}|\boldsymbol{\theta})$.

Step 3. Calculate

$$\alpha(\boldsymbol{\theta}, \boldsymbol{\phi}) = \min\left(1, \frac{\pi(\boldsymbol{\phi})q(\boldsymbol{\theta}|\boldsymbol{\phi})}{\pi(\boldsymbol{\theta})q(\boldsymbol{\phi}|\boldsymbol{\theta})}\right).$$

Step 4. With probability $\alpha(\boldsymbol{\theta}, \boldsymbol{\phi})$, set $\boldsymbol{\theta}^{t+1} = \boldsymbol{\phi}$, else set $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}(= \boldsymbol{\theta}^t)$.

Step 5. Increase t by one and return to step 1 until T iterations have been performed. To obtain a set of sampled values from π, discard $\boldsymbol{\theta}^0, \ldots, \boldsymbol{\theta}^N$ as burn-in, for some suitable value of N, and consider the values $\boldsymbol{\theta}^{N+1}, \ldots, \boldsymbol{\theta}^T$ as a (dependent) sample from π which can be used to obtain Monte Carlo estimates of parameters of interest.

Important notes:

1. We only need to know π up to proportionality, since any constants of proportionality cancel in the numerator and denominator of the calculation of α.

2. The performance of the MCMC algorithm is dependent on the choice of the proposal distribution q. If q is chosen poorly, then the number of rejections may be high, so that the efficiency of the procedure can be low; conversely if q is chosen such that only very small moves are proposed,

it may take a very long time for the Markov chain to traverse the set of plausible posterior parameter values (see example below).

(Simple) Example

Suppose that we are interested in sampling from the standard normal distribution, and that we choose to use a proposal distribution of the form

$$q(\phi|\theta) \sim N(\theta, \sigma^2),$$

where σ^2 is to be specified. Then the acceptance probability is given by $\alpha(\theta, \phi) = \min\{1, A\}$ where,

$$A = \frac{\exp\left(-\frac{1}{2}\phi^2\right)\exp\left(-\frac{1}{2\sigma^2}(\theta-\phi)^2\right)}{\exp\left(-\frac{1}{2}\theta^2\right)\exp\left(-\frac{1}{2\sigma^2}(\phi-\theta)^2\right)} = \exp\left(-\frac{1}{2}(\phi^2-\theta^2)\right). \tag{2.3}$$

Figure 2.5 plots the resulting output for $\sigma^2 = 0.1$, 1 and 100.

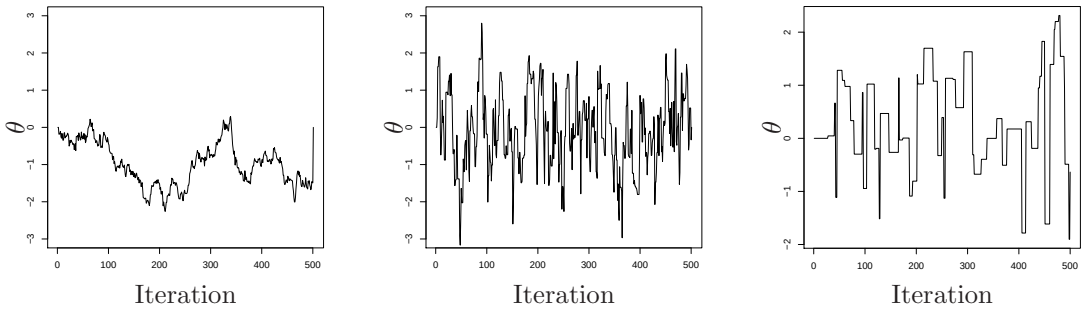


Fig. 2.5: Sample paths for Metropolis-Hastings algorithms with (a) $\sigma^2 = 0.1$, (b) $\sigma^2 = 1$ and (c) $\sigma^2 = 100$.

Notice that, in this case, the acceptance function is independent of σ, but that the value of σ has a significant impact upon the acceptance rate of the chain. This is because the proposal with $\sigma^2 = 1$ is much closer to the target distribution, so that more sensible candidates are generated and subsequently accepted. However, the proposal with $\sigma^2 = 100$, generates candidate observations too far out in the tail to come from the target distribution and these are subsequently rejected. Conversely, the proposal with $\sigma^2 = 0.1$ generates candidates very similar to the current value that are typically accepted, but means that it takes a long time to move over the parameter space. The movement around the parameter space is often referred to as "mixing" (see also Improving Performance)

Trace plots are a useful graphical tool for (informally) assessing the missing of the Markov chain. In addition to trace plots ACF plots (see §**??**) are often used in conjunction with trace plots to assess mixing. For the above trace plots, Figure 2.6. provides the corresponding ACF plots for both $\sigma^2 = 0.1$ and $\sigma^2 = 100$ appear fairly similar. Thus, ACF plots on their own do not provide enough information to explain why a Markov chain may be experiencing poor mixing - this could be a result of (at least) a high rejection probability (as for $\sigma^2 = 100$) or always very small "step" sizes (as for $\sigma^2 = 0.1$).

Special Cases

There are a number of special cases (or "flavours") of the Metropolis-Hastings algorithm. We consider a number of these next.

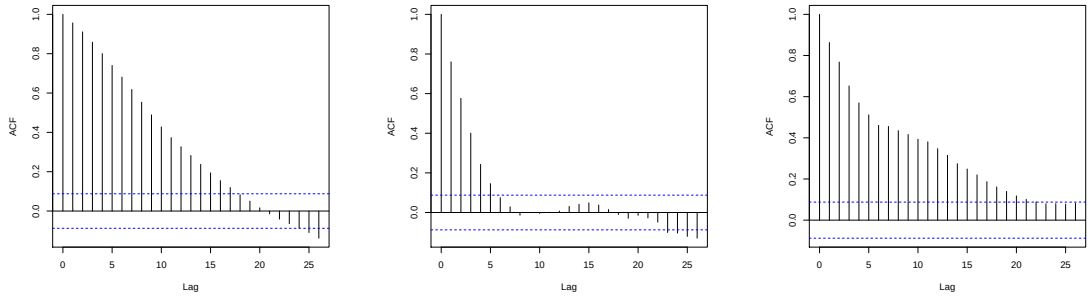


Fig. 2.6: ACF plots for Metropolis-Hastings algorithms with (a) $\sigma^2 = 0.1$, (b) $\sigma^2 = 1$ and (c) $\sigma^2 = 100$.

Random Walk Metropolis

This special case specifies the candidate generating function as a symmetric function so that,

$$q(\boldsymbol{\phi}|\boldsymbol{\theta}) = q(\boldsymbol{\theta} - \boldsymbol{\phi}) = q(\boldsymbol{\phi} - \boldsymbol{\theta}) = q(\boldsymbol{\theta}|\boldsymbol{\phi}).$$

Then, the acceptance function reduces to

$$\alpha(\boldsymbol{\theta}, \boldsymbol{\phi}) = \min\left(1, \frac{\pi(\boldsymbol{\phi})}{\pi(\boldsymbol{\theta})}\right). \tag{2.4}$$

If $q(\boldsymbol{\phi}|\boldsymbol{\theta}) = q(\boldsymbol{\theta} - \boldsymbol{\phi}) = q(\boldsymbol{\phi} - \boldsymbol{\theta}) = q(\boldsymbol{\theta}|\boldsymbol{\phi})$, then the kernel driving the chain is a random walk, since the candidate observation is of the form $\boldsymbol{\phi} = \boldsymbol{\theta}^t + \boldsymbol{z}$, where $\boldsymbol{z} \sim f$, where f is symmetric about zero. There are many common choices for f, including the uniform distribution on the unit disk, or a multivariate normal or t-distribution. In the univariate case, a standard choice is the Normal distribution $N(0, \sigma^2)$.

2.3.1 The Independence Sampler

If $q(\boldsymbol{\phi}|\boldsymbol{\theta}) = f(\boldsymbol{\phi})$, then the candidate observation is drawn *independently* of the current state of the chain. In this case, the acceptance probability can be written as

$$\alpha(\boldsymbol{\theta}, \boldsymbol{\phi}) = \min\left(1, \frac{\pi(\boldsymbol{\phi})f(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta})f(\boldsymbol{\phi})}\right) = \min\left(1, \frac{w(\boldsymbol{\phi})}{w(\boldsymbol{\theta})}\right),$$

where $w(\boldsymbol{\theta}) = \pi(\boldsymbol{\theta})/f(\boldsymbol{\theta})$. ($w$ is the importance weight function that would be used in importance sampling given samples generated from f; see Section 2.8.3 later.)

2.3.2 Single-updates and the Gibbs Sampler

The Metropolis-Hastings algorithm need not update all variables in the sample space simultaneously, it can be used in stages, updating variables one-at-a-time, much like the Gibbs Sampler. This is commonly called the single-update Metropolis-Hastings algorithm. This algorithm can be described as follows. Let the initial state be denoted by $\boldsymbol{\theta}^0 = (\theta_1^0, \ldots, \theta_k^0)$. At iteration t we cycle through each of the $\theta_1, \ldots, \theta_k$ parameters in turn and propose to update the parameter value. Consider parameter θ_p and set $\boldsymbol{\theta}_p^t = (\theta_1^{t+1}, \ldots, \theta_{p-1}^{t+1}, \theta_p^t, \theta_{p+1}^t, \ldots, \theta_k^t)$.

Step 1 Propose new candidate value $\phi_p \sim q(\phi_p|\boldsymbol{\theta}_p^t)$, and set $\boldsymbol{\phi}_p = (\theta_1^{t+1}, \ldots, \theta_{p-1}^{t+1}, \phi_p, \theta_{p+1}^t, \ldots, \theta_k^t)$ (i.e. $\boldsymbol{\phi}_p$ denotes the vector of parameter values replacing parameter θ_p^t with the candidate value for ϕ_p). Intuitively, it is as if for the other parameters the proposal is that $\boldsymbol{\phi}_{(p)} = \boldsymbol{\theta}_{(p)}^t$ with probability 1.

Step 2 Accept the candidate value with probability $\min(1, A)$, where,

$$A = \frac{\pi(\boldsymbol{\phi}_p)q(\theta_p^t|\boldsymbol{\phi}_p)}{\pi(\boldsymbol{\theta}_p^t)q(\phi_p|\boldsymbol{\theta}_p^t)} = \frac{\pi(\phi_p|\boldsymbol{\theta}_{(p)}^t)\pi(\boldsymbol{\theta}_{(p)}^t)q(\theta_p^t|\boldsymbol{\phi}_p)}{\pi(\theta_p^t|\boldsymbol{\theta}_{(p)}^t)\pi(\boldsymbol{\theta}_{(p)}^t)q(\phi_p|\boldsymbol{\theta}_p^t)} = \frac{\pi(\phi_p|\boldsymbol{\theta}_{(p)}^t)q(\theta_p^t|\boldsymbol{\phi}_p)}{\pi(\theta_p^t|\boldsymbol{\theta}_{(p)}^t)q(\phi_p|\boldsymbol{\theta}_p^t)}.$$

where $\boldsymbol{\theta}_{(p)}^t = \boldsymbol{\phi}_{(p)} = (\theta_1^{t+1}, \ldots, \theta_{p-1}^{t+1}, \theta_{p+1}^t, \ldots, \theta_k^t)$. This simplification is obtained by noting that $\pi(\boldsymbol{\phi}_p) = \pi(\phi_p|\boldsymbol{\theta}_{(p)}^t)\pi(\boldsymbol{\theta}_{(p)}^t)$ and similarly for $\pi(\boldsymbol{\theta}_p^t)$.

If the candidate value is accepted, set $\theta_p^{t+1} = \phi_p$; else if the move is rejected set $\theta_p^{t+1} = \theta_p^t$.

We note that, in fact, the Gibbs Sampler is a special case of the single-update Metropolis-Hastings algorithm. Suppose that in the single-update Metropolis-Hastings algorithm, we break each iteration of the algorithm into k steps, and let q_j denote a proposal for candidates in the jth co-ordinate direction, so that

$$q_j(\boldsymbol{\phi}|\boldsymbol{\theta}) = \begin{cases} \pi(\phi_j|\boldsymbol{\theta}_{(j)}) & \boldsymbol{\phi}_{(j)} = \boldsymbol{\theta}_{(j)}, \quad j = 1, \ldots, k. \\ 0 & \text{else} \end{cases}$$

With this proposal, the acceptance probability at the jth step is given by

$$\alpha_j(\boldsymbol{\theta}, \boldsymbol{\phi}) = \min(1, A),$$

where,

$$\begin{aligned} A &= \frac{\pi(\boldsymbol{\phi}_j)\pi(\theta_j^t|\boldsymbol{\phi}_{(j)})}{\pi(\boldsymbol{\theta}_j^t)\pi_j(\phi_j|\boldsymbol{\theta}_{(j)}^t)} \\ &= \frac{\pi(\boldsymbol{\phi}_j)/\pi(\phi_j|\boldsymbol{\theta}_{(j)}^t)}{\pi(\boldsymbol{\theta}_j^t)/\pi(\theta_j^t|\boldsymbol{\phi}_{(j)})} \\ &= \frac{\pi(\boldsymbol{\phi}_j)/\pi(\phi_j|\boldsymbol{\phi}_{(j)})}{\pi(\boldsymbol{\theta}_j^t)/\pi(\theta_j^t|\boldsymbol{\theta}_{(j)}^t)}, \quad \text{since } \boldsymbol{\phi}_{(j)} = \boldsymbol{\theta}_{(j)}^t \\ &= \frac{\pi(\boldsymbol{\phi}_{(j)})}{\pi(\boldsymbol{\theta}_{(j)}^t)}, \\ &\quad \text{by definition of conditional probability for } \boldsymbol{\theta} = (\theta_j, \boldsymbol{\theta}_{(j)}) \\ &= 1, \text{ since } \boldsymbol{\phi}_{(j)} = \boldsymbol{\theta}_{(j)}^t. \end{aligned}$$

Thus, at each step, the only possible jumps are to parameter vectors $\boldsymbol{\phi}$, that match $\boldsymbol{\theta}$ on all components other than the jth, and these are automatically accepted.

In other words we have the proposal density given by,

$$q(\boldsymbol{\phi}|\boldsymbol{\theta}) = \prod_{j=1}^{p} q_j([\boldsymbol{\phi}_{<j}, \boldsymbol{\theta}_{\geq j}], [\boldsymbol{\phi}_{\leq j}, \boldsymbol{\theta}_{>j}]),$$

where

$$\boldsymbol{\theta}_{>j} = (\boldsymbol{\theta}_i, \ \ i > j),$$

$$\boldsymbol{\phi}_{<j} = (\boldsymbol{\phi}_i, \ \ i < j)$$

and, since $\alpha_j(\boldsymbol{\theta}, \boldsymbol{\phi}) = 1 \quad \forall j$, the transition distribution can, by simple manipulation, be re-written in the same form as the Gibbs transition density given in (2.1).

Improving performance

The performance of the MCMC algorithm depends jointly on the target distribution of interest and the updating algorithm used. The target distribution is typically fixed (e.g. posterior distribution of interest), so that the performance of the algorithm can be changed only through the proposals used in the updating algorithm. With the exception of the Gibbs sampler, most MCMC updates require a degree of *pilot-tuning* to obtain a chain with good mixing properties. In practice, this often involves adjusting the relevant proposal variances to obtain a Metropolis-Hastings acceptance rate of 20-40% (Gelman *et al*, 1996). This can often be achieved by implementing a pilot run of the MCMC algorithm, for 1000 iterations, say, calculating the mean acceptance rate for each parameter and adjusting the proposal variance accordingly to obtain a mean acceptance rate in the given interval.

In addition, if parameters are highly correlated with each other (this is usually easy to assess from an initial pilot-run - for example calculating the correlation of the parameters), multi-parameter updates can be used, proposing to update a number of parameters simultaneously. This is often referred to as *blocking*, since parameters are updated in "blocks". In practice this is most commonly done via the Metropolis-Hastings algorithm, since the joint posterior conditional distribution of the parameters is typically non-standard.

2.4 Comparison of Gibbs sampler and Metropolis-Hastings algorithm

The Gibbs sampler is usually the "default" updating algorithm when the posterior conditional distribution is of standard form. This is because it can be seen as "efficient" since the parameter is always updated (i.e. with probability 1) and there are many computer programs (including `R`) that includes efficient intrinsic functions for simulating from a wide range of standard distributions. In addition, since the Gibbs sampler is conditional on the other parameters, it can be viewed as an "adaptive" updating algorithm in that the parameters of the conditional distribution are usually a function of the other parameters in the model and so change as the other parameter values change within the Markov chain. In other words the conditional distribution of the parameter will change dependent on the current state of the Markov chain.

However, the Metropolis-Hastings algorithm has the advantage over the Gibbs Sampler, in that it is not necessary to know (or recognise) all of the conditional distributions, we need only simulate from q, which we can choose arbitrarily. However, if q is poorly chosen, then the mixing of the Markov chain can be slow, so that the efficiency of the procedure can be low. Thus, the choice of q typically involves some *pilot-tuning* for the parameters within the proposal distribution.

Typically, Gibbs updates are more computationally intensive than Metropolis-Hastings updates. Gibbs updates require simulating from (possibly) complex distributions, whereas the Metropolis-Hatings algorithm involves simulating from simpler distributions which is computationally faster (the Metropolis-Hastings only needs to simulate from the proposal distribution q and a $U[0,1]$ distribution to perform the accept/reject step). However, typically time is taken when implementing the Metropolis-Hastings algorithm to perform pilot-tuning to make the algorithm efficient, whereas no such pilot-tuning is required for the Gibbs step. If parameters are highly correlated *a posteriori*, using a single-update MCMC algorithm (either Gibbs or Metropolis-Hastings) will often perform poorly. This can be solved by block updates (i.e. updating more than one parameter within a single step). Block-updates are typically easier to perform within the Metropolis-Hastings algorithm, due to the arbitrary nature of the proposal distribution q (we can define q to be any multivariate distribution,

such as the multivariate Normal distribution).

In general the "default" MCMC algorithm would update parameters with standard posterior conditional distributions via the Gibbs sampler; whereas parameters with non-standard posterior conditionals are updated using the Metropolis-Hastings algorithm. The real exception comes when parameters are highly correlated, in which case (unless the joint posterior conditional distribution of these parameters are of standard form, particularly multivariate Normal) a block Metropolis-Hastings algorithm would typically be used.

2.5 Data Augmentation

In general a data augmentation (or auxiliary variable) approach can be applied when data are missing, or parameters that do not have to exist are included in the analysis. Specifically:

1. When data not observed are included in the analysis. In theory, posterior distributions can be obtained by integrating out the missing data. However, typically this is difficult (or impossible) to do analytically. Note that prediction can be put into a missing data framework.

2. The likelihood of the data is intractable (or difficult to calculate), but, conditional on a collection of unobserved data or parameters (auxiliary quantities), likelihood and posterior calculations become tractable (or simpler and quicker).

To implement an auxiliary variable approach, we adopt the following technique, by essentially extending the standard Bayes' Theorem. Let $\boldsymbol{x}$ denote the observed data and $\boldsymbol{y}$ the unobserved (or missing) data. We treat the missing data (or auxiliary variables) as additional parameters to be estimated, and form the joint posterior distribution over both these auxiliary variables and model parameters $\boldsymbol{\theta}$. Bayes' Theorem states that,

$$\pi(\boldsymbol{\theta}, \boldsymbol{y}|\boldsymbol{x}) \propto f(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}).$$

We use a standard MCMC algorithm to sample from the joint posterior distribution $\pi(\boldsymbol{\theta}, \boldsymbol{y}|\boldsymbol{x})$. However, we are interested in the posterior distribution $\pi(\boldsymbol{\theta}|\boldsymbol{x})$. This is simply the marginal posterior distribution,

$$\pi(\boldsymbol{\theta}|\boldsymbol{x}) = \int \pi(\boldsymbol{\theta}, \boldsymbol{y}|\boldsymbol{x})d\boldsymbol{y}.$$

Thus, we simply obtain a sample from the joint posterior distribution and use the realisations of $\boldsymbol{\theta}$ to obtain posterior summary statistics of interest.

Example: Genetic Linkage

We illustrate the missing data approach with a simple example relating to the genetic linkage of 197 animals each allocated to one of four categories, according to some genetic characteristic:

$$\boldsymbol{y} = (y_1, y_2, y_3, y_4) = (125, 18, 20, 34)$$

with cell probabilities

$$\left(\frac{1}{2} + \frac{\theta}{4}, \frac{1}{4}(1-\theta), \frac{1}{4}(1-\theta), \frac{\theta}{4}\right).$$

Though it is possible to sample the posterior distribution of θ directly by the Metropolis-Hastings algorithm we use a data augmentation approach that allows us to simplify the likelihood and use the Gibbs sampler. Specifically, we augment the observed data $\boldsymbol{y}$ by dividing the first cell into two with

respective cell probabilities $1/2$ and $\theta/4$, giving an augmented data set $\boldsymbol{x} = (x_1, x_2, x_3, x_4, x_5)$ where $x_1 + x_2 = y_1$, $x_3 = y_2$, etc. Now specifying the prior,

$$\theta \sim U[0, 1],$$

we have

$$\pi(\theta|\boldsymbol{y}) \propto \left(\frac{1}{2} + \frac{\theta}{4}\right)^{y_1} \left(\frac{1}{4}(1-\theta)\right)^{y_2+y_3} \left(\frac{\theta}{4}\right)^{y_4}.$$

In this case, writing $\boldsymbol{x} = (\boldsymbol{y}, z)$ for "missing" data z (i.e., $x_1 = y_1 - z$, $x_2 = z$, $x_3 = y_2$, $x_4 = y_3$, $x_5 = y_4$) we have

$$\begin{aligned} \pi(\theta|\boldsymbol{x}) &\propto \left(\frac{1}{2}\right)^{x_1} \left(\frac{\theta}{4}\right)^{x_2+x_5} \left(\frac{1}{4}(1-\theta)\right)^{x_3+x_4} \\ &= \left(\frac{1}{2}\right)^{y_1-z} \left(\frac{\theta}{4}\right)^{z+y_4} \left(\frac{1}{4}(1-\theta)\right)^{y_2+y_3}. \end{aligned}$$

Thus we have,

$$\theta|z, \boldsymbol{y} \sim Beta(z + y_4 + 1, y_2 + y_3 + 1)$$

and

$$Z|\theta, \boldsymbol{y} \sim Bin\left(y_1, \frac{\theta}{2+\theta}\right).$$

Thus, we can update the parameter θ and auxiliary variables z using the Gibbs sampler, with both posterior conditional distributions of standard form.

Note: in the situation where the introduction of auxiliary variables simplifies the likelihood (as in the above genetic linkage example) there is typically a computational trade-off between the simplification of the likelihood and the additional computational expense of updating the auxiliary variables. The trade-off will be problem dependent. Clearly, if the likelihood is analytically intractable without the use of auxiliary variables, the trade-off is clear!

Prediction

Prediction can be seen in the framework of missing data - the values we wish to predict are simply missing values! These missing values are treated as auxiliary variables and the joint posterior distribution formed over the model parameters and auxiliary variables. An MCMC algorithm is constructed to obtain a sample from this distribution, and hence the marginal distribution of the missing values. For example, reconsider the rats model in Lecture 12, $Y_{ij} \sim N(\alpha + \beta z_j, \sigma^2)$. The following prior are specified,

$$\alpha \sim N(0, 10^5); \quad \beta \sim N(0, 10^5); \qquad \sigma^2 \sim \Gamma^{-1}(0.001, 0.001).$$

We wish to predict the weight of a rat at future times, $x_2, \ldots, x_5$, given they weight 150 at time x_1. We can do this using the BUGS language by simply adding in another rat to the dataset. The model specification stays the same! However, the data now reads:

```
# Data:
list(x = c(8.0, 15.0, 22.0, 29.0, 36.0), N = 31, T = 5,
Y = structure(.Data =   c(151, 199, 246, 283, 320,
                          145, 199, 249, 293, 354,
                          .......................
                          153, 200, 244, 286, 324,
                          150, NA, NA, NA, NA),
                          .Dim = c(31,5)))
```

Thus, we increase the number of individuals from $N = 30$ to $N = 31$. We add another rat to the dataset with initial weight of 150, and use `NA` for the weight at the other times, which is interpreted as unknown, to be updated within the algorithm. Note that the missing data values also need initial starting values. These can be read in by specifying `Y` in the initial starting values with `NA` for all the observed data. Alternatively and much easier when possible is to simply let `Nimble` generate these initial starting values. Further, `Y` needs to be monitored in order to obtain posterior estimates of the weight of the rat at the future times. Doing this we obtain posterior summary estimates of:

node	mean	sd	MC error	2.5\%	median	97.5\%	start	sample
Y[31,2]	194.8	10.82	0.08945	173.5	194.8	216.2	1001	19000
Y[31,3]	238.2	10.98	0.08733	216.5	238.2	259.7	1001	19000
Y[31,4]	281.5	10.88	0.0783	260.1	281.5	302.8	1001	19000
Y[31,5]	324.7	11.09	0.08437	303.0	324.7	346.4	1001	19000

For the complete NIMBLE code look at the `Nimble_rats_pred.R` file (Lecture 15); For further (simple) examples of how to set up prediction in NIMBLE, look at the `Nimble_Exp_lik_Gamma_prior.R` file (Lecture 12).

2.5.1 Hierarchical Models

Hierarchical modelling offers a very useful alternative to prior selection based on subjective or objective methods (as seen in the first part of this course). The essential idea of hierarchical models is to learn the prior to use for the data we are analysing, by looking at related data sets. These models can be fitted easily within a Bayesian framework via data augmentation. A detailed treatment of this topic is beyond the scope of the course; in this section we will give an overview of the main ideas behind hierarchical models. We will start with an example.

California's earthquakes

Suppose we are interested in predicting the occurrence of large earthquakes on a particular fault in southern California. On this particular fault, there have only ever been 5 large earthquakes in the last 100 years. This leads to 4 inter-event times which are (in years):

$$Y_0 = (10, 25, 37, 22)$$

We assume the inter-event times are Exponential(λ_0). Therefore, to predict future earthquakes, we need to estimate λ_0, the rate at which earthquakes happen on this fault. We have seen before that the Gamma(α, β) distribution is the conjugate prior in this case and the posterior is then:

$$\lambda_0 | Y_0 \sim \Gamma(\alpha + n, \beta + \sum_i Y_i),$$

where $n = 4$ and $\sum_i Y_i = 92$ in our example. Note that there are several other faults in Southern California which also have data on their respective earthquakes. It would be sensible to use somehow these data to inform our choice of prior for α and β. Such an approach would not be 'subjective' since all prior information is coming from other data sets. But it would not be 'objective' either since we are analysing the current data set using outside information. Therefore,we want to use the data from faults $Y_1, Y_2, \ldots$; they all have few inter-event times each, but they could still help us estimate the parameter λ_0 that generated the event times Y_0 on the original fault we are interested in. We assume that $Y_k \sim \text{Exp}(\lambda_k)$, with λ_k denoting the inter-event times on fault k. For now, suppose that we are

only interested in λ_0. The data from the other faults is additional data which we want to use to get a better estimate of λ_0. We have our original fault we are interested in, and K others. Let n_k denote the number of earthquakes on fault k. We write $Y_{k,i}$ to denote the i^{th} inter-arrival time on fault k.

No pooling

The simplest approach would be to assume that there is no statistical relationship between the different faults and analyse each independently. This is called **no pooling**: we assume that there is no information that we can share across faults. As such, each value of λ_i is estimated using only the earthquakes on that fault. This is the standard way of doing statistical analysis. We analyse only the data that we have in front of us. This leads us to the posterior we have seen before:

$$\lambda_0|Y_0 \sim \Gamma(\alpha + n_0, \beta + \sum_i Y_{0,i}).$$

The problem with this approach is that since we don't have many earthquakes on each fault, our estimation of all λ_k parameters will not be great. The posteriors are going to be very wide, so our predictions will not be accurate. Remember in general that the fewer observations we have available to estimate a parameter, the less accurate our estimate will be. If we have a good subjective prior based on accurate expert knowledge, we can use this to help drive our estimate towards the most likely values. But if we don't, then it is likely that our small data set (only 5 earthquakes in this case) will result in inaccurate estimation.

Complete pooling

An alternative approach is to assume that every fault is exactly the same so that all inter-event times have the same distribution i.e. that $\lambda_0 = \lambda_1 =, \ldots, = \lambda_K$. In this case, we can combine all our data together (**Complete pooling**). We have only a single value λ to estimate now, and $n_0 + n_1 + \cdots + n_K$ observations. If our assumption is correct, then having more available data will lead to more accurate estimation. The posterior is now:

$$\lambda_0|Y_0, Y_1, \ldots, Y_K \sim \Gamma\left(\alpha + \sum_k n_k, \quad \beta + \sum_k \sum_i Y_{k,i}\right).$$

We now have far more data with which to estimate λ_0. As such, our estimate will be more accurate...assuming that we were correct that the inter-event times on all faults were identically distributed. When this assumption is violated, we may do worse than in the no-pooling case, since data from faults with radically different values of λ_k will bias our estimate away from its true value!
In this example, it is probably unreasonable to assume that all faults have the same inter-event time distribution. Faults vary for many reasons - different types of faulting systems, different geometries, etc. Realistically, we expect the different λ_i values to be similar, but not identical.

Between complete and no pooling: hierarchical modelling

The scenario in this example is often observed with real data. We are interested in analysing a particular data set, but we also have access to similar, related data sets. We cannot reasonably assume that these other data sets have been generated by the exact same parameter value as ours. But they should still tell us something about the parameter value for our current data set.

$\Rightarrow$ Basic idea: we use these other data sets to choose an appropriate prior for our analysis.

Why is this reasonable? Remember that the posterior combines likelihood and prior. If the other datasets influence only the prior (not the likelihood), they give us a good 'initial guess' for the parameter value. But as we collect more observations on our fault, the likelihood will start to outweigh the prior so it matters less. Unlike no pooling, we are taking the other faults into account. But unlike complete pooling, we are not assuming our fault is identical (i.e. same parameter value) to any other faults.

- **Idea**: if we have no obvious reason to think our fault is different from the other faults, we can combine the no-pooling and complete-pooling approaches by using the other faults to set the parameters of the prior distribution for our fault.

In other words, we will assume that each of the values of λ_i are independent samples from some distribution p(λ). Our model is hence:

$$\begin{aligned} Y_k &\sim \text{Exp}(\lambda_k) \\ \lambda_k &\sim \Gamma(\alpha, \beta) \\ \alpha, \beta &\sim p(\alpha, \beta) \end{aligned}$$

Since we have access to many values of λ_k, we can essentially use these to estimate the parameters α and β. A key point is that we treat α and β as parameters to be estimated along with the λ_k i.e. we put a prior on α and β and learn them from the data. We will treat them simply as extra unknown parameters. Note that, conditional on α and β, we are estimating each λ_k *independently* of all the others (no pooling). But then we update α and β using *all* the λ_k values (complete pooling). As a consequence, the estimate of λ_0 will *shrink* towards an overall mean α/β, *borrowing strength* from the other estimates λ_k indirectly through α and β.
The parameters λ_k are often called 'random effects'. Depending on the particular analysis, random effects can be either considered as auxiliary variables or parameters of interest (as in our example). In general, suppose we have model parameters $\boldsymbol{\theta}$ and random effects $\boldsymbol{\lambda}$, we can form the joint posterior distribution as

$$\pi(\boldsymbol{\theta}, \boldsymbol{\lambda}|Y) \propto f(Y|\boldsymbol{\theta}, \boldsymbol{\lambda}) p(\boldsymbol{\lambda}|\boldsymbol{\theta}) p(\boldsymbol{\theta})$$

To obtain inference on the model parameters $\boldsymbol{\theta}$ sample from this joint posterior distribution (using MCMC) and marginalise to obtain estimates of the posterior summary statistics of interest. Estimates of the random effect terms are obtained by considering the sampled $\boldsymbol{\lambda}$ values (ignoring the sampled model parameters values). This model might seem complex but we can easily sample from the posterior of all parameters via Gibbs sampling (see lecture slides for further details).

2.6 Bayes Factors - Choice of hypothesis

The underlying principle within classical hypothesis testing is that we wish to reject some hypothesis H_0, in favour of an alternative hypothesis H_1. Bayesian statisticians also sometimes refer to H_0 as the *null hypothesis* and H_1 as the *alternative hypothesis*, by convention, but within the Bayesian framework the two hypotheses are interchangeable. The hypotheses are statements concerning the parameter(s) of interest. Typically, suppose that we are interested in the parameter $\theta \in \Theta$. Then, our hypotheses are of the form,

$$H_0 : \theta \in \Theta_0; \qquad H_1 : \theta \in \Theta_1,$$

where Θ_1 and Θ_2 are disjoint and exhaustive subsets of the parameter space Θ. Note that this can translate to a model comparison setting. For example, to choose between fitting a constant or a simple linear regression,

$$M_0 : \mathrm{E}(Y_i) = \beta_0; \qquad M_1 : \mathrm{E}(Y_i) = \beta_0 + \beta_1 x_i,$$

one has to choose between the hypotheses,

$$H_0 : \beta_1 = 0; \qquad H_1 : \beta_1 \neq 0.$$

Within the Bayesian context, we need to place a prior on the parameter space. Let,

$$p_i = \mathbb{P}(H_i) = \mathbb{P}(\theta \in \Theta_i),$$

for $i = 0, 1$, such that $p_0 + p_1 = 1$. Then, the prior odds for the null hypothesis, H_0, against the alternative hypothesis, H_1 are p_0/p_1. Therefore, the prior odds specify your prior beliefs concerning which of the two hypotheses is more likely, before observing any data. If $p_0 > p_1$ we would favour the null hypothesis; if $p_0 < p_1$, then we would favour the alternative hypothesis; and if $p_0 \approx p_1$, we regard both hypotheses as roughly equally likely within the prior specification.

Once we have specified the prior, we observe data $\boldsymbol{x}$, and wish to update our prior beliefs concerning the hypotheses. We do this by calculating the corresponding *posterior odds*, given by,

$$\frac{\mathbb{P}(\theta \in \Theta_0|\boldsymbol{x})}{\mathbb{P}(\theta \in \Theta_1|\boldsymbol{x})} = \frac{p_0 f(\boldsymbol{x}|\theta \in \Theta_0)/f(\boldsymbol{x})}{p_1 f(\boldsymbol{x}|\theta \in \Theta_1)/f(\boldsymbol{x})} = \frac{p_0 f(\boldsymbol{x}|\theta \in \Theta_0)}{p_1 f(\boldsymbol{x}|\theta \in \Theta_1)}$$

Then, if the posterior odds are greater than one, we would favour the null hypothesis; if the posterior odds are less than one, we favour the alternative hypothesis; if they are equal each hypothesis is equally likely *a posteriori*. Note that the posterior odds give a quantitative comparison between hypotheses H_0 and H_1.

A further statistic that is often used is the *Bayes factor*. This is simply defined to be the ratio of posterior odds to prior odds. The Bayes factor for H_0 against H_1 is denoted by B_{01} and given by,

$$B_{01} = \frac{\mathbb{P}(\theta \in \Theta_0|\boldsymbol{x})/\mathbb{P}(\theta \in \Theta_1|\boldsymbol{x})}{p_0/p_1} = \frac{f(\boldsymbol{x}|\theta \in \Theta_0)}{f(\boldsymbol{x}|\theta \in \Theta_1)}.$$

Considering model selection, some prefer to report Bayes factors rather than posterior model probabilities. This is because the true model may not be in the set of models under consideration. Kass and Raftery (1995) suggested the following 'rule of thumb' for Bayes factors.

Bayes Factor	Interpretation
< 3	No evidence of H_0 over H_1;
> 3	Positive evidence for H_0;
> 20	Strong evidence for H_0;
> 150	Very strong evidence for H_0.

This general guideline is often used by practicing Bayesians in the interpretation of their results.

Note: Under the loss function,

$$L = \begin{cases} 0 & \text{choose } H_0 \text{ when } H_0 \text{ is true} \\ 0 & \text{choose } H_1 \text{ when } H_1 \text{ is true} \\ l_{10} & \text{choose } H_1 \text{ when } H_0 \text{ is true} \\ l_{01} & \text{choose } H_0 \text{ when } H_1 \text{ is true} \end{cases}$$

the optimal decision is to choose H_1 over H_0 if-f $B_{01} < \frac{l_{01}}{l_{10}} \frac{P(H_1)}{P(H_0)}$. (Bernardo and Smith, 1994, p. 392).

2.6.1 Simple hypotheses

Suppose that we wish to test,

$$H_0 : \theta = \theta_0 \quad \text{vs} \quad H_1 : \theta = \theta_1.$$

Then, using Bayes Theorem, the posterior probability that hypothesis H_i is true is given by,

$$\mathbb{P}(\theta = \theta_i|\boldsymbol{x}) = \frac{f(\boldsymbol{x}|\theta_i)p_i}{f(\boldsymbol{x})} \propto f(\boldsymbol{x}|\theta_i)p_i,$$

for $i = 0, 1$. As $\mathbb{P}(\theta = \theta_0|\boldsymbol{x}) + \mathbb{P}(\theta = \theta_1|\boldsymbol{x}) = 1$, the corresponding constant of proportionality is obtained after calculating,

$$f(\boldsymbol{x}) = \sum_{j=0}^{1} f(\boldsymbol{x}|\theta_j)p_j.$$

Then, in the case of simple hypotheses, the Bayes factor is simply equal to the likelihood ratio, i.e.,

$$B_{01} = \frac{f(\boldsymbol{x}|\theta_0)}{f(\boldsymbol{x}|\theta_1)}$$

and hence is based solely on the data.

2.6.2 Composite hypotheses

Suppose that θ is continuous, and that we wish to test,

$$H_0 : \theta \in \Theta_0 \quad \text{vs} \quad H_1 : \theta \in \Theta_1.$$

Let $p_i = p(H_i)$ denote the prior probability of the hypothesis H_i, such that $\theta \in \Theta_i$. Let the corresponding prior for θ be denoted by $p(\theta)$. Then, $p_i = \int_{\theta\in\Theta_i} p(\theta)d\theta$, and,

$$\mathbb{P}(\theta \in \Theta_i|\boldsymbol{x}) \quad = \quad \int_{\theta\in\Theta_i} \pi(\theta|\boldsymbol{x})d\theta.$$

The corresponding Bayes' factor is given by,

$$B_{01} \quad = \quad \frac{\mathbb{P}(\theta \in \Theta_0|\boldsymbol{x})p_1}{\mathbb{P}(\theta \in \Theta_1|\boldsymbol{x})p_0} \left(= \frac{f(\boldsymbol{x}|\theta \in \Theta_0)}{f(\boldsymbol{x}|\theta \in \Theta_1)}\right).$$

Calculating prior and posterior hypothesis probabilities for the expression above (ignoring the ratio within the brackets) is the simplest approach for calculating the Bayes factor. This is done by integrating (analytically or by Monte Carlo integration) the prior or posterior over the required parameter subspace, or by using R for standard distributions. To fully demonstrate the importance of the prior distribution, we also consider the following derivation: let $p_i(\theta)$ denote the prior density restricted to Θ_i, renormalized to give a probability density over Θ_i. Then,

$$p_i(\theta) = p(\theta|H_i) = \frac{p(H_i|\theta)p(\theta)}{p(H_i)} = \begin{cases} \frac{p(\theta)}{p_i} & \text{when } \theta \in \Theta_i \\ 0 & \text{otherwise} \end{cases}$$

Then, the posterior probability for hypothesis $i = 0, 1$, is given by,

$$\begin{aligned} \mathbb{P}(\theta \in \Theta_i|\boldsymbol{x}) &= \int_{\theta\in\Theta_i} \pi(\theta|\boldsymbol{x})d\theta \\ &= \frac{1}{f(\boldsymbol{x})} \int_{\theta\in\Theta_i} f(\boldsymbol{x}|\theta)p(\theta)d\theta \\ &= \frac{1}{f(\boldsymbol{x})} p_i \int_{\theta\in\Theta_i} f(\boldsymbol{x}|\theta)p_i(\theta)d\theta, \end{aligned}$$

So, for a composite hypothesis, the integral $\int_{\theta\in\Theta_i} f(\boldsymbol{x}|\theta)p_i(\theta)d\theta$ is $f(\boldsymbol{x}|\theta\in\Theta_i)$.

[Note: This calculation can also be done as,

$$P(H_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|H_i)P(H_i)}{f(\boldsymbol{x})} = \frac{P(H_i)}{f(\boldsymbol{x})}\int_{\Theta_i} f(\boldsymbol{x}|\theta,H_i)p(\theta|H_i)d\theta = \frac{P(H_i)}{f(\boldsymbol{x})}\int_{\Theta_i} f(\boldsymbol{x}|\theta,H_i)p_i(\theta)d\theta$$

]

Then, the corresponding posterior odds are given by,

$$\frac{\mathbb{P}(\theta\in\Theta_0|\boldsymbol{x})}{\mathbb{P}(\theta\in\Theta_1|\boldsymbol{x})} = \frac{p_0\int_{\theta\in\Theta_0} f(\boldsymbol{x}|\theta)p_0(\theta)d\theta}{p_1\int_{\theta\in\Theta_1} f(\boldsymbol{x}|\theta)p_1(\theta)d\theta}.$$

The corresponding Bayes' factor is given by,

$$B_{01} = \frac{\int_{\theta\in\Theta_0} f(\boldsymbol{x}|\theta)p_0(\theta)d\theta}{\int_{\theta\in\Theta_1} f(\boldsymbol{x}|\theta)p_1(\theta)d\theta},$$

which is the ratio of "weighted" likelihoods by the densities $p_i(\theta)$. Thus, the Bayes factor is not solely dependent on the data, but also on the corresponding prior placed on the parameter.

Note that when calculating the Bayes factor, only proper prior distributions should be used for the model parameters, otherwise the Bayes factor becomes arbitrary.If improper priors were allowed, one could set $p_0(\theta) = c_0h_0(\theta)$, where the integral of $h_0(\theta)$ diverges (say, $h_0(\theta) = 1$) and c_0 is any arbitrary constant. Similarly, set $p_1(\theta) = c_1h_1(\theta)$, where the integral of $h_1(\theta)$ diverges (say, $h_1(\theta) = 1$) and c_1 is any arbitrary constant. By assigning equal prior probabilites for the two hypotheses, one could then set c_0/c_1 to be arbitrarily large or small, and thus control the derived Bayes factor. This is even more obvious in the case where $H_0 : \beta = 0$, $H_1 : \beta \neq 0$. Then,

$$B_{01} = \frac{f(\boldsymbol{x}|\beta=0)}{\int_{\beta\neq0} f(\boldsymbol{x}|\beta)p_1(\beta)d\beta},$$

and one could set that, $p(\beta|H_1) \propto c_1$, where c_1 is arbitrarily large or small.

Example

Suppose that $\boldsymbol{X} = \{X_1,\ldots,X_n\}$ where, given μ each $X_i \overset{iid}{\sim} N(\mu,1)$. For $n = 10$, we observe data $\boldsymbol{x}$:

$$3.4,\ 2.9,\ 3.0,\ 3.5,\ 3.3,\ 3.7,\ 2.7,\ 3.9,\ 2.7,\ 2.9,$$

so that $\bar{x} = 3.2$, and wish to test the simple hypothesis:

$$H_0 : \mu = 3, \quad \text{vs} \quad H_1 : \mu = 3.5.$$

What is the corresponding Bayes factor of H_0 against H_1? We have that,

$$\mathbb{P}(\mu = 3|\boldsymbol{x}) = \frac{p_0 f(\boldsymbol{x}|\mu=3)}{f(\boldsymbol{x})}.$$

Now,

$$\begin{aligned} f(\boldsymbol{x}|\mu=3) &= \prod_{i=1}^{10}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x_i-3)^2}{2}\right) \\ &= \frac{1}{(2\pi)^5}\exp\left(-\frac{2}{2}\right) \\ &= \frac{\exp(-1)}{(2\pi)^5}. \end{aligned}$$

Similarly, we have that,

$$f(\boldsymbol{x}|\mu = 3.5) = \frac{\exp(-1.25)}{(2\pi)^5}$$

Thus, we have that,

$$\begin{aligned} B_{01} &= \frac{f(\boldsymbol{x}|\mu = 3)}{f(\boldsymbol{x}|\mu = 3.5)} \\ &= 1.28. \end{aligned}$$

Thus there is only slight evidence to support model H_0 over H_1.

Example

Revisit the Example in Section 1.2.2, where, given λ, $x_i \sim Exp(\lambda)$, and $1/\lambda$ denotes the average lifetime of a laptop in years. Assume, as in Section 1.2.2, that prior beliefs are described by $\lambda \sim \Gamma(0.2, 0.6)$, so that $\mathrm{E}(\lambda) = 1/3 = 0.33$ (the median is 0.03). 20 laptops are tested, and their average lifetime turns out to be $\bar{x} = 5$. Then, as $\lambda|\boldsymbol{x} \sim \Gamma(n + \alpha, n\bar{x} + \beta)$,

$$\lambda|\boldsymbol{x} \sim \Gamma(20 + 0.2, 20 \times 5 + 0.6) = \Gamma(20.2, 100.6),$$

so that $\mathrm{E}(\lambda|\boldsymbol{x}) = 20.2/100.6 = 0.2007$.

Assume some investigator wants to test,

$$H_0 : \lambda > 1/4 \quad \text{vs} \quad H_1 : \lambda \leq 1/4.$$

Now, $P(H_0|\boldsymbol{x}) = \int_{1/4}^{\infty} \pi(\lambda|\boldsymbol{x})d\lambda = 0.14$, using 1-pgamma(0.25,20.2,100.6) in R. Also, $P(H_1|\boldsymbol{x}) = \int_0^{1/4} \pi(\lambda|\boldsymbol{x})d\lambda = 0.86$, using pgamma(0.25,20.2,100.6) in R.

Similarly, $p_0 = \int_{1/4}^{\infty} p(\lambda)d\lambda = 0.27$, and $p_1 = \int_0^{1/4} p(\lambda)d\lambda = 0.73$. Then,

$$B_{01} = \frac{P(H_0|\boldsymbol{x})/P(H_1|\boldsymbol{x})}{p_0/p_1} = 0.4,$$

with $B_{10} = 2.5$. The posterior odds are,

$$\frac{P(H_0|\boldsymbol{x})}{P(H_1|\boldsymbol{x})} = 0.15, \qquad \frac{P(H_1|\boldsymbol{x})}{P(H_0|\boldsymbol{x})} = 6.6.$$

This is how the data updated the prior odds,

$$\frac{p_0}{p_1} = 0.36, \qquad \frac{p_1}{p_0} = 2.7.$$

2.7 Model Discrimination

We now consider the issue of model discrimination in some detail. Within statistics the issue of model selection is often of particular interest and we discuss how this is addressed within a Bayesian framework. To illustrate the ideas we will make particular use of linear multiple regression, where we wish to assess which explanatory variables the response variable is a function of. In the simplest case, as for the previous rats example, there may be a single explanatory variable and we wish to assess whether a response variable is a function of a given explanatory variable (or covariate). Mathematically, we let y_i denote the set of observed response values with associated covariate value z_i, for $i = 1, \ldots, n$. Assuming a linear relationship (if it exists), there are two possible models we could consider:

$$\text{Model 0: } y_i = \alpha + \epsilon_i; \qquad \text{Model 1: } y_i = \alpha + \beta z_i + \epsilon_i,$$

where for each model, we assume, $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$. Thus, we could express the models in the form:

$$\text{Model 0: } y_i \sim N(\alpha, \sigma^2); \qquad \text{Model 1: } y_i \sim N(\alpha + \beta z_i, \sigma^2).$$

The two models clearly have different interpretations. Model 0 can be interpreted as the responses all having a common, constant, mean (α), independently of the explanatory variable; whereas Model 1 has the interpretation that the responses are linearly regressed on the given covariate. Note that in this case we say that the models are *nested*, since we can regard model 0 as a special case of model 1, where $\beta = 0$.

For multiple regression, the extension follows immediately. Let there be a total of J explanatory variables, such that the values of the explanatory variables associated with response y_i are given by $\boldsymbol{z}_j = \{z_{1i}, \ldots, z_{iJ}\}$. The "full" model is given by,

$$y_i = \alpha + \sum_{j=1}^{J} \beta_j z_{ij} + \epsilon_i,$$

where $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$. Equivalently we can write,

$$y_i \sim N\left(\alpha + \sum_{j=1}^{J} \beta_j z_{ij}, \sigma^2\right).$$

Sub-models can be specified by setting $\beta_j = 0$ for some of the j subscripts. In general, there are a total of 2^J models corresponding to each possible combination of explanatory variables present in the model. Models can be compared by considering posterior model probabilities, or pairwise Bayes factors that generate a ranking of the possible models. Note that for interpretability, each of the explanatory variables should all be measured on the same scale. Most commonly, each set of explanatory variables and response variable are normalised (by taking the raw values, subtracting their mean and dividing by the sample standard deviation so that the normalised values have a sample mean of 0 and sample standard deviation of 1) - this is very important for calculating Bayes factors (and posterior model probabilities).

2.7.1 Bayes Factors

We have already met the idea of Bayes factors. For example, for a simple linear regression, under model 0 (i.e. hypothesis H_0 or model M_0) the set of model parameters can be $\boldsymbol{\theta}_0 = \{\alpha, \sigma^2\}$; alternatively, under model 1 (i.e. hypothesis H_1 or model M_1), the model parameters may be $\boldsymbol{\theta}_1 = \{\alpha, \beta, \sigma^2\}$. We can represent the model selection process in the form of a hypothesis test, where we have:

$$H_0 \text{ or } M_0 : \beta = 0 \qquad \text{vs} \qquad H_1 \text{ or } M_1 : \beta \neq 0.$$

Thus, we have a simple null hypothesis versus a composite alternative hypothesis. (Note that a one-tailed hypothesis test follows similarly). The Bayes factor for choosing one of the 2 models is defined in exactly the same manner as previously by,

$$B_{01} = \frac{\pi(M = 0|\boldsymbol{x})/\pi(M = 1|\boldsymbol{x})}{p(M = 0)/p(M = 1)},$$

where $p(M = m)$ denotes the probability that m is the true model, for $m = 0, 1$. The prior is specified in the form of the prior probability for each model $p(M = m)$ and the prior for the parameters, $p(\boldsymbol{\theta}_m|M = m)$ conditional on each hypothesis. For this example, we need to specify $p(\alpha, \sigma^2|M = 0)$

and $p(\alpha, \beta, \sigma^2|M = 1)$ as the prior distribution of the parameters, conditional on each model. Given a data vector $\boldsymbol{x}$, the posterior probability of model m is

$$\pi(M = m|\boldsymbol{x}) \propto f(\boldsymbol{x}|M = m)p(M = m) = \int f(\boldsymbol{x}|M = m, \boldsymbol{\theta}_m)p(\boldsymbol{\theta}_m|M = m)\, d\boldsymbol{\theta}_m p(M = m).$$

The constant of proportionality is simply,

$$\left[f(\boldsymbol{x}|M = 0)p(M = 0) + f(\boldsymbol{x}|M = 1)p(M = 1)\right]^{-1}.$$

Thus, in theory, we can calculate the posterior model probabilities (and hence Bayes factors) by performing the above integration. This provides a *quantitative* discrimination between competing models. However, in general the necessary integration will be analytically intractable..... (problem number 1!)

In addition, often we may not want to compare only a limited number of possible models (2 in the above hypothesis test description or 4 in the DIC rats example below), but may have many competing models that we wish to discriminate between (problem number 2!). For example, suppose that instead of only 1 covariate, there are k covariates that the response variable may be linearly regressed upon. Then, in general there are a total of 2^k different possible models, corresponding to the inclusion/exclusion of each covariate in the regression equation.

In general, in the presence of model uncertainty, we can extend the standard Bayesian theorem, by considering the model to be a discrete variable M to be estimated (in addition to the parameters in the model). The set of possible values that this parameter can take is simply the set of models deemed to be plausible. By applying Bayes' Theorem, we form the joint posterior distribution over both the model and the parameters. Formally, we have a posterior distribution (over parameter and model space) defined upto proportionality.

$$\pi(\boldsymbol{\theta}_m, M = m|\boldsymbol{x}) = \frac{f(\boldsymbol{x}|\boldsymbol{\theta}_m, M = m)p(\boldsymbol{\theta}_m|M = m)p(M = m)}{f(\boldsymbol{x})} \propto f(\boldsymbol{x}|\boldsymbol{\theta}_m, M = m)p(\boldsymbol{\theta}_m|M = m)p(M = m). \tag{2.5}$$

The posterior model probability of model m is then simply the marginal posterior distribution:

$$\begin{aligned}\pi(M = m|\boldsymbol{x}) &= \int \pi(\boldsymbol{\theta}_m, M = m|\boldsymbol{x})d\boldsymbol{\theta}_m = \frac{1}{f(\boldsymbol{x})}\int f(\boldsymbol{x}|\boldsymbol{\theta}_m, M = m)p(\boldsymbol{\theta}_m|M = m)p(M = m)d\boldsymbol{\theta}_m \\ &= \frac{p(M = m)}{f(\boldsymbol{x})}\int f(\boldsymbol{x}, \boldsymbol{\theta}_m|M = m)d\boldsymbol{\theta}_m = \frac{p(M = m)}{f(\boldsymbol{x})}f(\boldsymbol{x}|M = m).\end{aligned}$$

i.e. we integrate out over the parameter space. Therefore,

$$\pi(M = m|\boldsymbol{x}) \quad \propto \quad f(\boldsymbol{x}|M = m)p(M = m), \tag{2.6}$$

where the likelihood of the data given the model is given by,

$$f(\boldsymbol{x}|M = m) = \int f(\boldsymbol{x}|\boldsymbol{\theta}_m, M = m)p(\boldsymbol{\theta}_m|M = m)d\boldsymbol{\theta}_m. \tag{2.7}$$

Thus, this is again simply the "marginal" likelihood, integrating out over the parameter space. The constant of proportionality is,

$$1/f(\boldsymbol{x}) = \left[\sum_m f(\boldsymbol{x}|M = m)p(M = m)\right]^{-1}.$$

Typically a parameter may be present in more than one model (for example, the "intercept" parameter α in models 0 and 1 above). In general, we can specify a different set of prior distributions

for the parameters in each of the different models. However, it is acceptable to specify (independent) priors on the parameters such that the same prior is specified on the parameter for each model containing that parameter. Caution is required when using a proper prior for the model parameters with a large variance. For example, when comparing nested (linear) models, adopting, say, a $N(0, 10^6)$ or $U(-100, 100)$ prior for the additional parameters may give different results compared to adopting, say, a $N(0, 10^{10})$ or $U(-1000, 1000)$ prior; see O'Hagan (2004, Bayesian Inference, pages 174-180). To overcome this problem, investigators usually adopt proper priors for the model parameters that carry as much information as a single data observation; see Unit Information priors, g-priors, and some additional non-examinable material on Moodle.

The Bayes Factor is a model comparison tool that is consistent. If the true model belongs to the set of analysed models, the Bayes Factor will select the true model with probability one as $n \to \infty$. (The same is true for the BIC, a frequentist information criterion that is an approximation of the Bayes factor; see Raftery, 1995). Later you will also use the Deviance Information Criterion. This is another tool for Bayesian model comparison. It is designed to find the best model in terms of predictive performance, in the same spirit as the frequentist model selection criterion AIC. The DIC is not consistent. This is not considered to be a problem by some investigators, arguing that the true model is rarely in the set of analysed models, as nature is too complex to allow for that.

We will consider two different methods for obtaining an estimate of posterior model probabilities, both essentially using a Monte Carlo type approach. We initially consider a simple Monte Carlo estimate (which, although intuitive and relatively easy to program, typically performs poorly), before (very briefly) presenting the ideas associated with extending the MCMC algorithm in the presence of model uncertainty (but omitting the computational details).

Simple Monte Carlo approach

The Monte Carlo approach involves obtaining an estimate of the marginal likelihood (2.7),

$$f(\boldsymbol{x}|M = m) = \int f(\boldsymbol{x}|\boldsymbol{\theta}_m, M = m)p(\boldsymbol{\theta}_m|M = m)d\boldsymbol{\theta}_m,$$

for each of the different possible models. These can then be combined with the prior model probabilities in order to obtain an estimate of the posterior model probabilities, given in equation (2.6)

$$\hat{\pi}(M = m|\boldsymbol{x}) \propto f(\boldsymbol{x}|M = m)p(M = m).$$

However, obtaining an estimate of the expression $f(\boldsymbol{x}|M = m)$ is non-trivial. We consider a number of different approaches all essentially based on the following principle. We can express equation (2.7) in the form,

$$f(\boldsymbol{x}|M = m) = \mathbb{E}_p[f(\boldsymbol{x}|\boldsymbol{\theta}_m, M = m)],$$

i.e., the expectation of the likelihood, $f(\boldsymbol{x}|\boldsymbol{\theta}_m, M = m)$, with respect to the prior distribution $p(\boldsymbol{\theta}_m)$. We can estimate this expectation by drawing K observations from the prior distribution of the parameters in model m, denoted by $\boldsymbol{\theta}^1, \cdots, \boldsymbol{\theta}^K$ and then use the Monte Carlo estimate,

$$\widehat{f}(\boldsymbol{x}|M = m) = \frac{1}{K}\sum_{j=1}^{K} f(\boldsymbol{x}|\boldsymbol{\theta}^j, M = m).$$

In other words we estimate the mean of the likelihood with respect to the prior distribution by the sample mean where the parameter values are drawn from the prior distribution.

An estimate of the posterior probability of model m, up to proportionality, is,

$$\hat{\pi}(M = m|\boldsymbol{x}) \propto p(M = m)\widehat{f}(\boldsymbol{x}|M = m).$$

This estimate converges to the posterior model probability (up to proportionality), as $K \to \infty$. We repeat this process for each model and renormalise the estimates obtained for each model $m = 1, ..., m^*$, to obtain an estimate of the corresponding posterior model probabilities. In other words, we set,

$$\hat{\pi}(M = m|\boldsymbol{x}) = \frac{p(M = m)\widehat{f}(\boldsymbol{x}|M = m)}{\sum_{q=1}^{m^*} p(M = q)\widehat{f}(\boldsymbol{x}|M = q)}.$$

Example: Consider 3 possible models, $m = 1, 2, 3$. Then,

$$f(\boldsymbol{x}|M = m) = \int f(\boldsymbol{x}|\boldsymbol{\theta}_m, M = m)p(\boldsymbol{\theta}_m|M = m)d\boldsymbol{\theta}_m = E_{prior}(f(\boldsymbol{x}|\boldsymbol{\theta}_m, M = m)).$$

After calculating Monte Carlo estimates $\widehat{f}(\boldsymbol{x}|M = 1)$, $\widehat{f}(\boldsymbol{x}|M = 2)$, $\widehat{f}(\boldsymbol{x}|M = 3)$, consider that,

$$\pi(M = m|\boldsymbol{x}) = \frac{p(M = m)}{f(\boldsymbol{x})} f(\boldsymbol{x}|M = m),$$

and that $\pi(M = 1|\boldsymbol{x}) + \pi(M = 2|\boldsymbol{x}) + \pi(M = 3|\boldsymbol{x}) = 1$. Therefore,

$$\widehat{f}(\boldsymbol{x}) = p(M = 1)\widehat{f}(\boldsymbol{x}|M = 1) + p(M = 2)\widehat{f}(\boldsymbol{x}|M = 2) + p(M = 3)\widehat{f}(\boldsymbol{x}|M = 3).$$

Now, for $m = 1, 2, 3$,

$$\widehat{\pi}(M = m|\boldsymbol{x}) = \frac{p(M = m)\widehat{f}(\boldsymbol{x}|M = m)}{p(M = 1)\widehat{f}(\boldsymbol{x}|M = 1) + p(M = 2)\widehat{f}(\boldsymbol{x}|M = 2) + p(M = 3)\widehat{f}(\boldsymbol{x}|M = 3)}.$$

However, the above estimates of the marginal likelihood are generally inefficient and very unstable, as a result of the parameters being drawn from the prior distribution: the corresponding likelihood values evaluated at the sampled parameter values from the prior distribution are often very small, as the prior, in general, is far more dispersed over the parameter space than the likelihood. Thus, the expectation is heavily dominated by only a few sampled values, even for large values of K, resulting in a very large variance, and the instability of the estimate.

Alternative more stable Monte Carlo approach

An alternative Monte Carlo estimate for the likelihood can be derived using importance sampling (see §2.8.3). The idea here is to sample from a distribution for which the values are in high-likelihood areas, and then re-weight the estimate. Formally, if $\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^K$ are drawn from an importance sampling distribution $q(\boldsymbol{\theta})$, define importance sampling weights,

$$w^j = \frac{p(\boldsymbol{\theta}^j)}{q(\boldsymbol{\theta}^j)},$$

for $j = 1, \ldots, K$. The corresponding new estimate of the expression of equation (2.7) is given by,

$$\widehat{f}_2(\boldsymbol{x}|M = m) = \frac{\sum_{j=1}^{K} f(\boldsymbol{x}|\boldsymbol{\theta}^j, M = m)w^j}{\sum_{j=1}^{K} w^j}.$$

In order to avoid the instability problems, as for estimate $\widehat{f}(\boldsymbol{x}|M = m)$, with samples drawn from the prior distribution, we need to specify an importance sampling distribution, where draws from this distribution lie in high likelihood areas. An obvious choice for the importance sampling distribution is the posterior distribution, which we can sample from using the MCMC algorithm, in order to obtain the parameters $\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^K$ from the posterior distribution. Setting the importance sampling

distribution above to be the posterior distribution, (i.e. $q(\boldsymbol{\theta}^j) = \pi(\boldsymbol{\theta}^j|\boldsymbol{x}, M = m)$), the estimate of the likelihood in (2.7) simplifies to the harmonic mean, i.e.

$$\widehat{f_3}(\boldsymbol{x}|M = m) = \left[\frac{1}{K}\sum_{j=1}^{K}\frac{1}{f(\boldsymbol{x}|\boldsymbol{\theta}^j, M = m)}\right]^{-1}.$$

However, this estimate is very sensitive to small likelihood values and can once again be very unstable.

The Monte Carlo estimates are the easiest to program and conceptualise. However, they are often very inefficient and do not always converge within a feasible number of iterations. Note that even if, for a given problem, we can obtain reliable estimates of $f(\boldsymbol{x}|m_i)$, we need to consider each possible model individually. This can quickly become infeasible, even for moderate numbers of models, and essentially impossible for large numbers of possible models. For example, back to our variable selection problem within linear regression, with k possible covariates there are a total of 2^k possible models. So if there are 10 possible covariates (this is quite realistic in practice) there are a total of 1024 possible models! In such instances search algorithms can be implemented to reduce the number of possible models considered, but these are not guaranteed to identify the models with largest posterior support.

MCMC-type approach

An alternative approach which is to consider a "simple" extension of the Metropolis-Hastings algorithm, and most commonly referred to as (RJ) reversible jump MCMC. This approach does not encounter the problems of unstable estimates in the same way as the previous Monte Carlo estimates, and also naturally deals with the issue of large numbers of possible models. The basic idea is to construct a Markov chain with stationary distribution equal to the joint posterior distribution over both parameter and model space. The Metropolis-Hastings algorithm cannot be used since, in general, the competing models are of different dimensions (i.e., have a different number of parameters). The RJMCMC algorithm extends the Metropolis-Hastings algorithm by allowing the constructed Markov chain to move between different dimensions, and hence between different models. Posterior model probabilities are then estimated as the proportion of time that the Markov chain spends in each model.

2.7.2 Deviance Information Criterion

Within a classical framework, model selection is often conducted via the use of Akaike's information criterion (AIC). The model with the smallest AIC is chosen. The AIC is a function of the deviance (i.e. -2 log likelihood evaluated at the MLE of the parameters) and number of parameters, and can be viewed as a trade-off between the fit of the model to the data and its corresponding complexity.

$$AIC : -2\log f_m(\boldsymbol{x}|\hat{\boldsymbol{\theta}}) + 2(\text{number of parameters}).$$

Within the Bayesian framework the parameters no longer have a fixed value but a distribution. Consequently, an alternative criterion has been suggested - the Deviance information criterion (DIC). This is similar to the AIC, in that it discriminates between models by considering both the fit and complexity of the models, but was developed specifically for use within the Bayesian framework. The DIC for model m is given by,

$$DIC_m \quad = \quad -2\mathbb{E}_\pi(\log f(\boldsymbol{x}|\boldsymbol{\theta}_m, M = m)) + p_D(m),$$

where $p_D(m)$ denotes the *effective number of parameters* and is given by

$$p_D(m) = -2\mathbb{E}_\pi(\log f(\boldsymbol{x}|\boldsymbol{\theta}_m, M = m)) + 2\log f(\boldsymbol{x}|\hat{\boldsymbol{\theta}}_m, M = m),$$

where $\hat{\boldsymbol{\theta}}_m$ is a posterior point estimate. $p_D(m)$ is a measure of how constrained the parameters are. For example, for a multiple regression with unconstrained parameters, $p_D(m)$ is the number of parameters. In the rats example, for $\alpha_i \sim N(\alpha, k)$, the smaller k is the more constrained the parameters are and the smaller $p_D(m)$ is. Note that the DIC can also be written in the form,

$$-2\log f(\boldsymbol{x}|\hat{\boldsymbol{\theta}}_m, M = m) + 2p_D(m).$$

The model with the **lowest** DIC statistic is the preferred model.

The most common point estimate to use is the posterior mean, $\hat{\boldsymbol{\theta}} = \mathbb{E}_\pi(\boldsymbol{\theta})$ and we will use this definition from here on. In this case, the effective number of parameters is the difference between the posterior expectation of the deviance and the deviance evaluated at the posterior mean of the parameters. Note that the effective number of parameters is a function of the posterior mean of the parameters, $\mathbb{E}_\pi(\boldsymbol{\theta}|\boldsymbol{x})$. This can be problematic when this is a poor estimate of the location of the posterior distribution (for example, for a bimodal distribution where the mean is generally a very poor point estimate in an area with very low posterior support). In some cases, a negative effective number of parameters can be obtained. Finally we note that this form of the DIC cannot be used where the parameters are discrete-valued, since $\mathbb{E}_\pi(\boldsymbol{\theta}|\boldsymbol{x})$ will typically not be equal to one of these discrete values.

The DIC can be easily calculated within an MCMC algorithm, and is available in `OpenBUGS` (see example below). NIMBLE computes the Watanabe - Akaike (WAIC) criterion which is discussed in the next subsection.

Example: Rats

We return to the rats example. We let $Y_{i,j} \sim N(\mu_{ij}, \sigma^2)$. We consider 4 different models for the mean:

$$\begin{aligned}
\text{Model 0:} \quad Y_{i,j} &= \alpha + \epsilon_{i,j};\\
\text{Model 1:} \quad Y_{i,j} &= \alpha + \beta_1 z_j + \epsilon_{i,j};\\
\text{Model 2:} \quad Y_{i,j} &= \alpha + \beta_2 z_{2j} + \epsilon_{i,j};\\
\text{Model 3:} \quad Y_{i,j} &= \alpha + \beta_1 z_j + \beta_2 z_{2j} + \epsilon_{i,j}.
\end{aligned}$$

where $\epsilon_{i,j} \sim N(0, \sigma^2)$ and z_j denotes the normalised time at time j, and z_{2j} denotes the normalised squared time. Thus, these models correspond to the constant model (model 0), linear regression on time (model 1), linear regression on the time squared (model 2) and quadratic regression on time (model 3). For each parameter, if it is present in the model we specify independent priors,

$$\alpha \sim N(0, 10^5); \qquad \beta_1 \sim N(0, 10^5); \qquad \beta_2 \sim N(0, 10^5); \qquad \sigma^2 \sim \Gamma^{-1}(0.001, 0.001).$$

We initially run the full model (i.e. model 3) for 30,000 iterations with a conservative burn-in of 1,000 iterations and obtain the following summary statistics:

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
alpha	242.6	1.3	0.007311	240.1	242.6	245.2	1001	29000
beta	87.9	7.873	0.04272	72.47	87.96	103.3	1001	29000
beta2	-19.79	7.867	0.04295	-35.06	-19.81	-4.345	1001	29000
sigma2	254.9	30.18	0.1864	202.5	252.5	320.4	1001	29000

The names parameters correspond to `alpha` $= \alpha$; `beta` $= \beta_1$; `beta2` $= \beta_2$; $\sigma 2 = \sigma^2$. Independent replications provided essentially identical results and the convergence diagnostics did not suggest any

lack of convergence. By looking at these posterior estimates it once more appears that there is a strong linear relationship with a 95% symmetric credible interval of $(72.47, 103.3)$ for β_1, which clearly does not contain the value of 0. The 95% symmetric credible interval for β_2 is $(-35.06, -4.345)$. Although this does not contain the value of 0, it is significantly smaller in magnitude than the linear term. We need to perform a more rigorous model discrimination technique to formally compare different competing models.

We calculate the DIC for each of the four possible models. `OpenBUGS` provides the estimated DIC statistic (using the posterior mean as the point estimate in the calculation of the effective number of parameters). The quantity $-2\mathbb{E}_\pi(\log f(\boldsymbol{x}|\boldsymbol{\theta}_m, M = m))$ is shown as Dbar. The quantity $-2\log f(\boldsymbol{x}|\hat{\boldsymbol{\theta}}_m, M = m)$ as Dhat. The following output is obtained in `OpenBUGS` for each model for the posterior estimates and the DIC. Note that we run the MCMC iterations each for 30,000 iterations with 1,000 burn-in for each (each checked for convergence).

Model 0:

```
node      mean      sd      MC error    2.5%     median     97.5%   start  sample
alpha    242.6     5.197    0.03244    232.5     242.6      252.8    1001    29000
sigma2   4091.0    482.6     2.698     3252.0    4055.0     5136.0   1001    29000

Dbar = post.mean of -2logL; Dhat = -2LogL at post.mean of stochastic nodes
          Dbar       Dhat        pD       DIC
Y     1672.050   1670.040     2.009   1674.060
total 1672.050   1670.040     2.009   1674.060
```

Model 1:

```
node      mean      sd      MC error    2.5%     median     97.5%   start  sample
alpha    242.6     1.322    0.007959    240.1     242.6      245.2    1001    29000
beta     68.45     1.487    0.008534    65.53     68.45      71.37    1001    29000
sigma2   263.9     31.14    0.1942      209.9     261.5      331.4    1001    29000

Dbar = post.mean of -2logL; Dhat = -2LogL at post.mean of stochastic nodes
         Dbar       Dhat        pD       DIC
Y     1260.940   1257.930     3.017   1263.960
total 1260.940   1257.930     3.017   1263.960
```

Model 2:

```
node      mean      sd      MC error    2.5%     median     97.5%   start  sample
alpha    242.6     1.765    0.01073    239.2     242.6      246.1    1001    29000
beta2    66.59     1.985    0.01151    62.69     66.59      70.48    1001    29000
sigma2   470.3     55.5     0.3483     374.1     466.1      590.6    1001    29000

Dbar = post.mean of -2logL; Dhat = -2LogL at post.mean of stochastic nodes
           Dbar       Dhat        pD       DIC
Y     1347.630   1344.610     3.016   1350.640
total 1347.630   1344.610     3.016   1350.640
```

Model 3:

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
alpha	242.6	1.3	0.007311	240.1	242.6	245.2	1001	29000
beta	87.9	7.873	0.04272	72.47	87.96	103.3	1001	29000
beta2	-19.79	7.867	0.04295	-35.06	-19.81	-4.345	1001	29000
sigma2	254.9	30.18	0.1864	202.5	252.5	320.4	1001	29000

Dbar = post.mean of -2logL; Dhat = -2LogL at post.mean of stochastic nodes

	Dbar	Dhat	pD	DIC
Y	1255.560	1251.540	4.017	1259.580
total	1255.560	1251.540	4.017	1259.580

Thus model 3 is the preferred model using the DIC statistic. We note that if only a linear component or a quadratic component is present, the linear component is preferred (i.e. model 1 has a lower DIC than model 2). The (relative) fit of the model to the data can be seen by the `Dbar` statistic (the posterior mean deviance) but also the estimated σ^2. Better fitting models have smaller σ^2 values.

Fitting each model to the data may be infeasible for a large number of possible models. In addition, the DIC does not provide a quantitative comparison that is readily interpretable. A more natural approach to Bayesian model discrimination is via Bayes Factors or posterior model probabilities. These have the advantage that they are a quantitative comparison of competing models and permit additional model averaging ideas.

2.7.3 The Watanabe - Akaike (WAIC) criterion

We discussed earlier some of the problems associated with the DIC. For a more detailed discussion, describing additional problems associated with the DIC, see Spiegelhalter et al.(2014). (Uploaded on Moodle in the 'Interesting material' section.)

These problems led to the proposal of a different model selection criterion, the WatanabeAkaike criterion, or WAIC. For the WAIC, for data $\boldsymbol{x} = (x_1, ..., x_n)$, and a replicate dataset $\boldsymbol{x}$ to be predicted, the measure of goodness of fit is

$$-2\log\prod_{i=1}^{n} f(x_i|\boldsymbol{x}, M = m)$$

$$= -2\sum_{i=1}^{n}\log\int_{\Theta} f(x_i|\boldsymbol{\theta}_m, M = m)\pi(\boldsymbol{\theta}_m|\boldsymbol{x}, M = m)d\boldsymbol{\theta}_m,$$

assuming independent observations x_i given $\boldsymbol{\theta}_m$. We use the notion of the replicated dataset, because when we predict each x_i given all the observed data, and we condition on $\boldsymbol{x}$, we want x_i to still be considered a random variable with probability function $f(x_i|\boldsymbol{x}, M = m)$.

There are 2 possible complexity penalties,

$$p_{WAIC1} = \sum_i E_\pi\left(-2\log f(x_i|\boldsymbol{\theta}_m, M = m)\right) + 2\sum_i \log f(x_i|\boldsymbol{x}),$$

and

$$p_{WAIC2} = \frac{1}{4}\sum_i V_\pi\left(-2\log f(x_i|\boldsymbol{\theta}_m, M = m)\right).$$

. For p_{WAIC2}, $V_\pi()$ represents the sample variance of the function's argument, evaluated using samples from the posterior distribution of $\boldsymbol{\theta}_m$. Discussing p_{WAIC1} and p_{WAIC2} in detail is beyond the aims

of this module. For more information, see Gelman et al. (2014) in the 'Interesting Material' Moodle Section.

The WAIC does not depend on point estimates for the model parameters, avoiding some of the pitfalls of the DIC. Also, it is a more reliable criterion for a certain class of models called 'mixture models', in contrast to the DIC. But, keep in mind that, for relatively simple models such as GLMs, the DIC can also be reliable. `Nimble` allows for the calculation of the WAIC for Bayesian model comparison, using p_{WAIC2}. See the lecture demonstration on WAIC calculations, using the 'rats' data. See also the relevant code uploaded on Moodle.

2.8 Direct Sampling

It is sometimes possible that we can sample from some distribution directly. This is typically the case for standard or simple/univariate distributions. We describe a number of sampling approaches for obtaining independent samples from some distribution.

2.8.1 Method of Inversion

This is the simplest of all procedures, as long as one can calculate the inverse cumulative distribution function for the target distribution. If $X \sim f$, with F the corresponding cumulative distribution function, then $F(X) \sim U[0,1]$. Suppose that we wish to simulate a continuous random variable X with cumulative distribution function

$$F(x) = \mathbb{P}(X \leq x).$$

Suppose also that the inverse function $F^{-1}(u)$ is well defined for $0 \leq u \leq 1$. Then we can use the following algorithm to sample from f.

Step 1. Generate u as a realisation from $U \sim U[0,1]$.
Step 2. Then, $x = F^{-1}(u)$ is a sample from $X \sim f$

Proof:

If $X = F^{-1}(U)$, then what is the distribution of X?

$$\mathbb{P}(X \leq x) = \mathbb{P}(F^{-1}(U) \leq x),$$

but since F is the cumulative distribution function of a continuous random variable, F is a strictly monotonic, increasing and continuous function of x. Hence,

$$\mathbb{P}(F^{-1}(U) \leq x) = \mathbb{P}(U \leq F(x)).$$

But, as U is a $U[0,1]$ random variable,

$$\mathbb{P}(U \leq F(x)) = F(x),$$

i.e.,

$$\mathbb{P}(X \leq x) = F(x).$$

Therefore, $X \sim f(x)$. □

Example

Let $X \sim Exp(\lambda)$, $(\lambda > 0)$. How can we sample from this distribution? Remember that,

$$f(x) = \lambda e^{-\lambda x} \qquad \text{for } x \geq 0,$$

and

$$F(x) = \int_0^x \lambda e^{-\lambda u} du = 1 - e^{-\lambda x}, \qquad x \geq 0.$$

We need to find the function F^{-1}. Let

$$F(x) = u = 1 - e^{-\lambda x},$$

then

$$x = -\frac{1}{\lambda} \ln(1-u).$$

So, to sample from the exponential distribution (if 'rexp()' in R was not available!) we can set

$$x = F^{-1}(u) = -\frac{1}{\lambda} \ln(1-u), \text{ where } U \sim U[0,1].$$

where u is a realisation (sample) from $U \sim U[0,1]$. Note that if $U \sim U[0,1]$ then $(1-U) \sim U[0,1]$ so that we can write, $x = -\frac{1}{\lambda} \ln u$.

2.8.2 Rejection sampling

Suppose that we wish to generate observations $\theta^1, \theta^2, \ldots, \theta^n$ from the posterior distribution $\pi(\theta|\boldsymbol{x})$. One way to do this is to enclose the density function within a rectangular box and generate points uniformly at random over this region - see Figure 2.7.

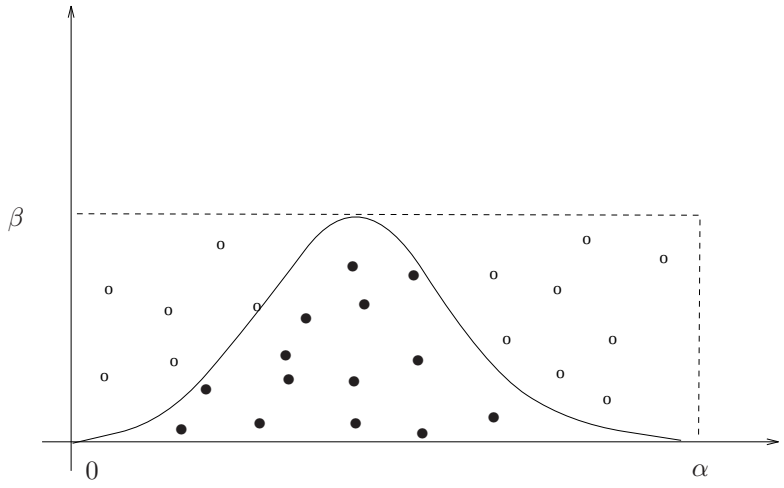


Fig. 2.7: Simulation by rejection sampling using a rectangular envelope box.

Any points above the density function are rejected and any underneath are accepted. We take the abscissa (or x-coordinate) of the accepted points to be our required random number from the given distribution.

Thus, we use the following algorithm. For $0 \leq \theta \leq \alpha$ and $0 \leq \pi(\theta|\boldsymbol{x}) \leq \beta$.

Step 1. Generate θ_* as a realisation from $\Theta_* \sim U[0, \alpha]$.

STEP 2. GENERATE y AS A REALISATION FROM $Y \sim U[0, \beta]$.
STEP 3. ACCEPT θ_* IF $y \leq \pi(\theta_*|\boldsymbol{x})$, ELSE IF $y > \pi(\theta_*|\boldsymbol{x})$ GO BACK TO STEP 1 AND REPEAT.

Notes:

1. Simulating $\theta_* \sim U[0, \alpha]$ is equivalent to simulating $V \sim U[0, 1]$ and setting $\theta_* = V\alpha$.
2. We can also replace steps 2 and 3 above by simulating w from $W \sim U[0, 1]$, and accepting the simulated value of θ_* if $w \leq \frac{\pi(\theta_*|\boldsymbol{x})}{\beta}$.

There are a number of problems associated with this method.

1. A rectangle cannot be used when $\pi(\theta|\boldsymbol{x})$ has an infinite range, and
2. The probability of rejection can become quite large.

A way of overcoming these problems is to *envelope* the posterior density function, $\pi(\theta|\boldsymbol{x})$ *not* by a rectangle, but some some other (more general) curve, $g(\theta)$. $g(\theta)$ is often some multiple of a second p.d.f. $h(\theta)$ i.e., $g(\theta) = Mh(\theta)$, from which it easy to sample.

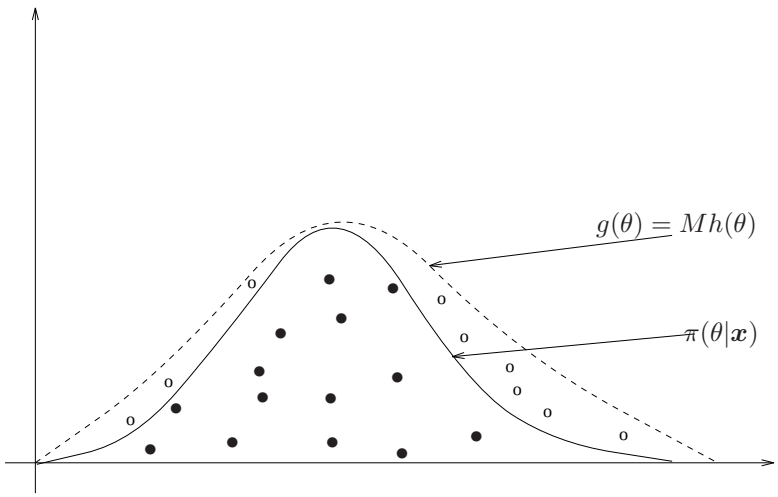


Fig. 2.8: Simulation by rejection sampling, with general envelope function g.

If we let $g(\theta) = Mh(\theta)$, where $M \geq 1$, then we can use the following algorithm.

STEP 1. SIMULATE θ_* AS A REALISATION FROM $\Theta_* \sim h(\theta)$.
STEP 2. GENERATE y FROM $Y \sim U[0, g(\theta_*)]$.
STEP 3. ACCEPT θ_* AS A REALISATION FROM $\pi(\theta|\boldsymbol{x})$ IF AND ONLY IF $y \leq \pi(\theta_*|\boldsymbol{x})$.

Notes:

1. We can again replace steps 2 and 3 above by simulating$u$ from $U \sim U[0, 1]$, and accepting the simulated value of θ_* if $g(\theta_*)u \leq \pi(\theta_*|\boldsymbol{x}) \Rightarrow u \leq \frac{\pi(\theta_*|\boldsymbol{x})}{g(\theta_*)}$.
2. For a specific θ_*, $P(\theta_* \text{ is accepted }) = \pi(\theta_*|\boldsymbol{x})/g(\theta_*) = \pi(\theta_*|\boldsymbol{x})/[Mh(\theta_*)]$.

3. Why does rejection sampling work? To prove this, more rigorous probabilistic notation is required. Denote by Θ^* the random variable that describes the θ derived/accepted by this algorithm. Denote by Θ_* the random variable so that $\Theta_* \sim h(\theta)$. We now just need to show that $P(\Theta^* < z)$ is the c.d.f. of $\pi(\theta|\boldsymbol{x})$.

$$P(\Theta^* < z) = P\left(\Theta_* < z | U \leq \frac{\pi(\Theta_*|\boldsymbol{x})}{Mh(\Theta_*)}\right)$$

$$= \frac{P\left(\Theta_* < z, U \leq \frac{\pi(\Theta_*|\boldsymbol{x})}{Mh(\Theta_*)}\right)}{P\left(U \leq \frac{\pi(\Theta_*|\boldsymbol{x})}{Mh(\Theta_*)}\right)}.$$

Using the Law of Total Probability,

$$P(\Theta^* < z) = \frac{\int_{-\infty}^{\infty} P\left(\Theta_* < z, U \leq \frac{\pi(\Theta_*|\boldsymbol{x})}{Mh(\Theta_*)} | \Theta_* = \theta\right) h(\theta) d\theta}{\int_{-\infty}^{\infty} P\left(U \leq \frac{\pi(\Theta_*|\boldsymbol{x})}{Mh(\Theta_*)} | \Theta_* = \theta\right) h(\theta) d\theta}$$

$$= \frac{\int_{-\infty}^{z} P\left(U \leq \frac{\pi(\Theta_*|\boldsymbol{x})}{Mh(\Theta_*)} | \Theta_* = \theta\right) h(\theta) d\theta}{\int_{-\infty}^{\infty} P\left(U \leq \frac{\pi(\Theta_*|\boldsymbol{x})}{Mh(\Theta_*)} | \Theta_* = \theta\right) h(\theta) d\theta} = \frac{\int_{-\infty}^{z} \frac{\pi(\theta|\boldsymbol{x})}{Mh(\theta)} h(\theta) d\theta}{\int_{-\infty}^{\infty} \frac{\pi(\theta|\boldsymbol{x})}{Mh(\theta)} h(\theta) d\theta} = \int_{-\infty}^{z} \pi(\theta|\boldsymbol{x}) d\theta.$$

This completes the proof.

4. Note that the probability of acceptance (in general, and not for a specific θ_*), is given by,

$$\begin{aligned}
\mathbb{P}(\text{accept}) &= P\left(U \leq \frac{\pi(\Theta_*|\boldsymbol{x})}{Mh(\Theta_*)}\right) \\
&= \int_{-\infty}^{\infty} P\left(U \leq \frac{\pi(\Theta_*|\boldsymbol{x})}{Mh(\Theta_*)} | \Theta_* = \theta_*\right) h(\theta_*) d\theta_* \\
&= \int_{-\infty}^{\infty} \frac{\pi(\theta_*|\boldsymbol{x})}{Mh(\theta_*)} h(\theta_*) d\theta_* \\
&= \frac{1}{M} \times 1 \\
&= \frac{1}{M}
\end{aligned}$$

5. Therefore, we would like M to be as close to 1 as possible, subject to $M \geq 1$. But, we also need that $g(\theta) = Mh(\theta) \geq \pi(\theta|\boldsymbol{x})$, for all θ, since g must "envelope" π. Therefore,

$$M \geq \frac{\pi(\theta|\boldsymbol{x})}{h(\theta)} \; \forall \theta,$$

so that the optimal M is simply

$$M^* = \sup_{\theta} \left(\frac{\pi(\theta|\boldsymbol{x})}{h(\theta)}\right),$$

where this maximum is finite.

Example

Suppose that we wish to sample from a $Beta(3, 2)$ distribution. The support for the distribution is $[0, 1]$ and so we can consider a rectangular sampling distribution (i.e. $h \equiv U[0, 1]$). Given the sampling distribution, we calculate the optimal value of M, to be the smallest value of M such that,

$$M \geq \frac{f(\theta)}{h(\theta)} = \frac{\Gamma(5)}{\Gamma(3)\Gamma(2)} \theta^2 (1 - \theta).$$

The maximum is obtained when $\theta = \frac{2}{3}$ giving the value of $M = \frac{16}{9} = 1.78$. (Check!). The rejection sampling algorithm is:

STEP 1. SIMULATE θ_* FROM $\sim U[0,1]$.
STEP 2. GENERATE y FROM $Y \sim U[0, 1.78]$.
STEP 3. ACCEPT θ_* AS A REALISATION FROM $Beta(3,2)$ IF AND ONLY IF $y \leq \frac{\Gamma(5)}{\Gamma(3)\Gamma(2)}\theta_*^2(1-\theta_*)$.

In a simulation of 100 points, (shown in Figure 2.9) 58 points are accepted, with a sample average of 0.633. The acceptance probability is $1/M = 0.56$ and the expectation of the distribution is known to be 0.6.

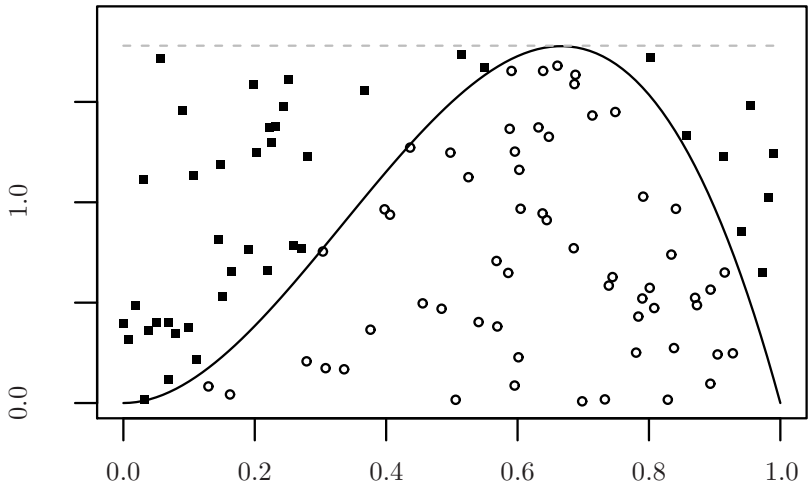


Fig. 2.9: Illustration of a rejection method for the $Beta(3,2)$ distribution. The $\circ$'s are accepted. The x-coordinates of the $\circ$'s are realisations of the random variable with the $Beta(3,2)$ density function.

To improve this algorithm, we can modify the 'envelope' $g(\theta)$, as a step function so that $g(\theta) = 16/9$ for $0.2 < \theta < 1$, as before, but now $g(\theta) = 0.384$ for $0 < \theta \leq 0.2$. Note that $f(\theta)$ is a strictly increasing function in $0 < \theta \leq 0.2$, and that $f(0.2) = 0.384$, so $g(\theta)$ still envelopes $f(\theta)$. Now,

$$\int_0^1 g(\theta)d\theta = 0.2 \times 0.384 + 0.8 \times (16/9) = 1.49902.$$

As $h(\theta)$ has to be a distribution that integrates to one,

$$h(\theta) = \frac{1}{1.49902} g(\theta),$$

and $g(\theta) = Mh(\theta) \Rightarrow M = 1.49902$. The rejection sampling algorithm is:

STEP 1. SIMULATE u FROM $\sim U[0,1]$.
STEP 2. IF $u \leq 0.2 \times 0.384/1.49902$, SIMULATE θ_* FROM $\sim U[0,0.2]$.
STEP 3. IF $u \leq 0.2 \times 0.384/1.49902$, GENERATE y FROM $Y \sim U[0,0.384]$.
STEP 4. IF $u > 0.2 \times 0.384/1.49902$, SIMULATE θ_* FROM $\sim U[0.2,1]$.
STEP 5. IF $u > 0.2 \times 0.384/1.49902$, GENERATE y FROM $Y \sim U[0,16/9]$.
STEP 6. ACCEPT θ_* AS A REALISATION FROM $Beta(3,2)$ IF AND ONLY IF $y \leq \frac{\Gamma(5)}{\Gamma(3)\Gamma(2)}\theta_*^2(1-\theta_*)$.

In a simulation of 100 points, 69 points are accepted, with a sample average of 0.59. The theoretical acceptance probability is $1/M = 0.68$ and the expectation of the distribution is 0.6.

Distribution known up to proportionality

Note that rejection sampling can be extended to the case where the normalisation constant is unknown (as for a posterior distribution). Suppose that we write the posterior distribution as,

$$\pi(\theta|\boldsymbol{x}) = \frac{f_1(\theta|\boldsymbol{x})}{f(\boldsymbol{x})},$$

where $f_1(\theta|\boldsymbol{x}) = f(\boldsymbol{x}|\theta)\pi(\theta)$. Let $g(\theta) = Mh(\theta)$ so that $g(\theta)$ envelopes $f_1(\theta|\boldsymbol{x})$. Then use the algorithm,

STEP 1. SIMULATE θ_* AS A REALISATION FROM $\Theta_* \sim h(\theta)$.
STEP 2. GENERATE y FROM $Y \sim U[0, g(\theta_*)]$.
STEP 3. ACCEPT θ_* AS A REALISATION FROM $\pi(\theta|\boldsymbol{x})$ IF AND ONLY IF $y \leq f_1(\theta_*|\boldsymbol{x})$.

(Note - this works because for $u \leq \frac{f_1(\theta|\boldsymbol{x})}{g(\theta)}$, multiplying both the numerator and denominator by $1/f(\boldsymbol{x})$ does not change the algorithm.) In general, rejection sampling can be very wasteful and is only really feasible in one or two dimensions. In addition, it can be difficult to identify a suitable "g" function (as we ideally want this to be of similar shape to π for efficiency).

2.8.3 Importance sampling

We once more wish to obtain a sample from the posterior distribution $\pi(\theta|\boldsymbol{x})$, which we assume is difficult to do directly. However, suppose that we can easily sample from some other distribution $g(\theta)$ (such that if $\pi(\theta|\boldsymbol{x}) > 0$ then $g(\theta) > 0$ - in other words g has the same support as π). We initially consider the case where both π and g are normalised densities (i.e. we know the constants of proportionality).

Suppose further that we are interested in obtaining an estimate of $\mathbb{E}_\pi(f(\theta))$. We would normally estimate $\mathbb{E}_\pi(f(\theta))$ by the usual Monte Carlo estimate,

$$\hat{E}_\pi(f(\theta)) = \frac{1}{n}\sum_{i=1}^{n} f(\theta^i),$$

where θ^i would be samples from $\pi(\theta|\boldsymbol{x})$. But sampling from $\pi(\theta|\boldsymbol{x})$ is not easy! Note that,

$$\mathbb{E}_\pi(f(\theta)) = \int f(\theta)\pi(\theta|\boldsymbol{x})d\theta = \int \frac{f(\theta)\pi(\theta|\boldsymbol{x})}{g(\theta)}g(\theta)d\theta.$$

So, the expectation with respect to $\pi(\theta|\boldsymbol{x})$ can also be seen as an expectation with respect to $g(\theta)$, which is easy to sample from. Let $\theta^1, \theta^2, \ldots, \theta^n$ be a sample from $g(\theta)$. We can estimate $\mathbb{E}_\pi(f(\theta))$ by

$$\hat{E}_\pi(f(\theta)) = \hat{E}_g(\frac{f(\theta)\pi(\theta|\boldsymbol{x})}{g(\theta)}) = \frac{1}{n}\sum_{i=1}^{n}\frac{\pi(\theta^i|\boldsymbol{x})}{g(\theta^i)}f(\theta^i) = \frac{1}{n}\sum_{i=1}^{n} w(\theta^i)f(\theta^i).$$

where we have now defined "importance" weights, for the θ^i sampled from $g(\theta)$,

$$w(\theta^i) = \frac{\pi(\theta^i|\boldsymbol{x})}{g(\theta^i)}.$$

The advantage of this method is that we can use it for any densities provided that they are continuous and have the same support. In addition, it can be used even when the constant of proportionality for π is unknown. Assume that $\pi^*(\theta|\boldsymbol{x})$ is the known expression for the posterior, up to proportionality. Then we estimate

$$\hat{E}_\pi(f(\theta)) = \frac{\sum_{i=1}^{n} w^*(\theta^i)f(\theta^i)/n}{\sum_{i=1}^{n} w^*(\theta^i)/n}.$$

where,

$$w^*(\theta^i) = \frac{\pi^*(\theta^i|\boldsymbol{x})}{g(\theta^i)}.$$

This works because the denominator in $\hat{E}_\pi(f(\theta))$ is an importance sampling estimator of $\int \pi^*(\theta|\boldsymbol{x})d\theta$, and when this divides $\pi^*(\theta^i|\boldsymbol{x})$ we obtain an estimate of $\pi(\theta^i|\boldsymbol{x})$.

However there are several disadvantages. The variance of the estimator can be very large, when $g$is not suitable for the problem at hand, leading to estimates that are not reliable. Without the constant of proportionality for π, the variance of the estimator can be even larger.

Note that importance sampling can still be very efficient, and reduce the variance of Monte Carlo estimates. However, the choice of the importance sampling density g is crucial, and the form of an appropriate distribution g depends on $\pi(\theta|\boldsymbol{x})$ and the different functions of interest to be estimated.

Example

Suppose that we wish to estimate the probability $\mathbb{P}(\theta > 2)$, where θ follows a Cauchy distribution, with known density

$$\pi(\theta) = \frac{1}{\pi(1+\theta^2)}, \qquad \theta \in \mathbb{R}$$

so we require

$$\int_2^\infty \pi(\theta)d\theta = \int_{-\infty}^{\infty} I(\theta > 2)\pi(\theta)d\theta,$$

where I denotes the indicator function. We could simulate from the Cauchy distribution directly, but the variance of the ergodic average in this case, is very large. Alternatively, we observe that, for large θ, $\pi(\theta)$ is similar in behaviour to the density

$$g(\theta) = 2/\theta^2 \quad \theta > 2.$$

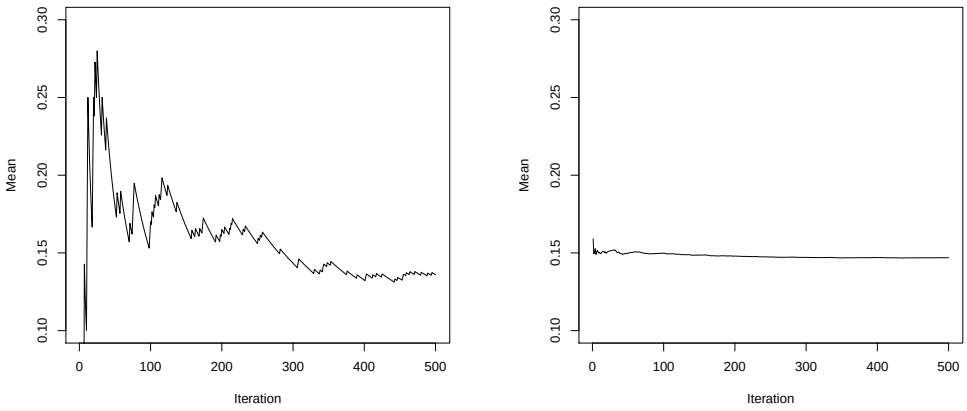
We can simulate from this distribution directly using the method of inversion. Let $U^i \sim U(0,1)$ and set $\theta^i = 2/U^i$ for $i = 1, \ldots, n$ (you should check this!). Note that g does not have the same support as π, but g does have the same support as $I(\theta > 2)\pi(\theta)$ and so we can still use importance sampling. Suppose that we sample $\theta^1, \ldots, \theta^n$ from g. We define importance sampling weights,

$$w_i = \frac{\pi(\theta^i)}{g(\theta^i)} = \frac{(\theta^i)^2}{2\pi(1+(\theta^i)^2)}.$$

Then, since each $\theta^i > 2$ we have that $f(\theta^i) = I(\theta^i > 2) = 1$ for all i. Thus, our estimator becomes:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{(\theta^i)^2}{2\pi(1+(\theta^i)^2)},$$

where $\theta^i = 2/U^i$, for $U^i \sim U[0,1]$ which can be easily coded in, for example, `R`.



(a) (b)

Fig. 2.10: Estimate of $\mathbb{P}(\theta > 2)$ (a) simulating directly from the Cauchy distribution and (b) using importance sampling.

The exact value of $\mathbb{P}(\theta > 2)$ is $0.5 - \pi^{-1}\tan 2 = 0.1476$. Figure 2.10(a) plots the estimated value of $\mathbb{P}(\theta > 2)$ obtained by using the `rcauchy()` command in `R` as a function of n. Figure 2.10(b) provides the corresponding plot of estimated values of $\mathbb{P}(\theta > 2)$ using the above importance sampling algorithm. Clearly, the reduction in variability is substantial!

Sampling importance resampling (SIR)

Sampling importance resampling in its simplest form is a simple extension of the importance sampling algorithm above. In particular, simulate random variables, $\theta^1, \ldots, \theta^N$, from some arbitrary probability density function $g(\theta)$, with (at least) the same support as the posterior distribution of interest, π. For each simulated random deviate from g, calculate the weight,

$$\omega_i = \frac{\pi(\theta^i|\boldsymbol{x})}{g(\theta^i)}$$

for $i = 1, \ldots, n$. (This is the importance sampling part). These weights are normalised using,

$$w_i = \frac{\omega_i}{\sum_{j=1}^{n}\omega_j}.$$

We independently resample with replacement the n simulated θ values, where the probability of simulating θ^i is given by w_i (the resampling part). Let the set of resampled parameter values be denoted by $\phi^1, \ldots, \phi^n$, which can then be used to obtain Monte Carlo estimates of summary statistics of interest. For example, to calculate an estimate of the posterior mean use,

$$\frac{1}{n}\sum_{i=1}^{n}\phi^i.$$

This approach of sampling importance resampling is a special form of particle filtering, which is very commonly used within time series data.

One problem with this sampling importance resampling type of approach can be that of *particle depletion*, where only a few simulated θ values contribute the majority of the weights. In other words suppose that we order the simulated θ values in decreasing order of their weights, and denote these by $\theta_{(1)}, \ldots, \theta_{(n)}$ where $\theta_{(j)}$ denotes the θ value with jth largest weight, with corresponding weight $w_{(j)}$. Then particle depletion occurs when $\sum_{j=1}^{n} w_{(j)}$ is close to one for n relatively small, so that only a relatively few θ values are resampled. Particle depletion leads to poor estimates of summary statistics with poor precision (i.e. large Monte Carlo error). The performance of the algorithm can be improved (as for standard importance sampling and rejection sampling) by a good choice of proposal distribution g.

All direct sampling algorithms suffer from the problem of dimensionality. These methods can be generally implemented to obtain posterior estimates of summary statistics of interest in one dimension (without too many problems) but become significantly more difficult (and generally impossible) to implement efficiently in higher dimensions. This is why we considered earlier Markov chain Monte Carlo, the most common approach for implementing Bayesian analyses and obtaining inference on the parameters of interest.

Appendix A

PROBABILITY DISTRIBUTIONS

This appendix gives the form of the pmf/pdf and summary statistics for common distributions, which are frequently used within these Bayesian Inference notes.

A.1 Discrete distributions

Distribution	Parameters	Mass function	Mean and variance
Binomial $\theta \sim Bin(n,p)$	sample size $n \in \mathbb{N}$ $p \in [0,1]$	$f(\theta) = \left(\begin{array}{c} n \\ \theta \end{array} \right) p^\theta (1-p)^{n-\theta}$ $\theta = 0, 1, \ldots, n$	$\mathbb{E}(\theta) = np$ $Var(\theta) = np(1-p)$
Poisson $\theta \sim Poisson(\lambda)$	rate $\lambda > 0$	$f(\theta) = \lambda^\theta \exp(-\lambda)(\theta!)^{-1}$ $\theta = 0, 1, 2, \ldots$	$\mathbb{E}(\theta) = \lambda$ $Var(\theta) = \lambda$
Geometric $\theta \sim Geom(p)$	$p \in [0,1]$	$f(\theta) = p(1-p)^{\theta-1}$ $\theta = 0, 1, \ldots$	$\mathbb{E}(\theta) = 1/p$ $Var(\theta) = (1-p)/p^2$
Negative Binomial $\theta \sim Neg\text{-}Bin(\alpha, \beta)$	shape $\alpha > 0$ inverse scale $\beta > 0$	$f(\theta) = \left(\begin{array}{c} \theta+\alpha-1 \\ \theta \end{array} \right) \left(\frac{\beta}{\beta+1}\right)^\alpha \left(\frac{1}{\beta+1}\right)^\theta$ $\theta = 0, 1, 2, \ldots$	$\mathbb{E}(\theta) = \alpha/\beta$ $Var(\theta) = \frac{\alpha}{\beta^2}(\beta+1)$
Multinomial $\boldsymbol{\theta} \sim MN(n, \boldsymbol{p})$	sample size $n \in \mathbb{N}$ $p_i \in [0,1]; \sum_{i=1}^k p_i = 1$	$f(\boldsymbol{\theta}) = \frac{n!}{\prod_{i=1}^k \theta_i!} \prod_{i=1}^k p_i^{\theta_i}$ $\theta_i = 0, 1, \ldots, n; \sum_{i=1}^k \theta_i = n$	$\mathbb{E}(\theta_j) = np_j$ $Var(\theta_i) = np_i(1-p_i)$

A.2 Continuous distributions

Distribution	Parameters	Density function	Mean and variance
Uniform $\theta \sim U[a,b]$	$b > a$	$f(\theta) = 1/(b-a)$ $\theta \in [a,b]$	$\mathbb{E}(\theta) = (a+b)/2$ $Var(\theta) = (b-a)^2/12$
Normal $\theta \sim N(\mu, \sigma^2)$	location μ scale $\sigma > 0$	$f(\theta) = \frac{\exp\left(-(\theta-\mu)^2/(2\sigma^2)\right)}{\sqrt{2\pi\sigma^2}}$ $\infty < \theta < \infty$	$\mathbb{E}(\theta) = \mu$ $Var(\theta) = \sigma^2$
log Normal $\theta \sim \log N(\mu, \sigma^2)$	μ $\sigma > 0$	$f(\theta) = \frac{\exp\left(-(\log\theta-\mu)^2/(2\sigma^2)\right)}{\sqrt{2\pi\sigma^2}\theta}$ $0 \leq \theta < \infty$	$\mathbb{E}(\theta) = \exp\left(\mu + \frac{\sigma^2}{2}\right)$ $Var(\theta) = \exp(2\mu+\sigma^2)(\exp(\sigma^2)-1)$
Beta $\theta \sim Beta(\alpha, \beta)$	$\alpha > 0$ $\beta > 0$	$f(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}$ $\theta \in [0,1]$	$\mathbb{E}(\theta) = \frac{\alpha}{\alpha+\beta}$ $Var(\theta) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$
Exponential $\theta \sim Exp(\lambda)$	$\lambda > 0$	$f(\theta) = \lambda\exp(-\lambda\theta)$ $\theta > 0$	$\mathbb{E}(\theta) = 1/\lambda$ $Var(\theta) = 1/\lambda^2$
Gamma $\theta \sim \Gamma(\alpha, \beta)$	shape $\alpha > 0$ rate $\beta > 0$	$f(\theta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\theta^{\alpha-1}\exp(-\beta\theta)$ $\theta > 0$	$\mathbb{E}(\theta) = \alpha/\beta$ $Var(\theta) = \alpha/\beta^2$
Inverse Gamma $\theta \sim \Gamma^{-1}(\alpha, \beta)$	shape $\alpha > 0$ rate $\beta > 0$	$f(\theta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\theta^{-(\alpha+1)}\exp(-\beta/\theta)$ $\theta > 0$	$\mathbb{E}(\theta) = \beta/(\alpha-1)$, for $\alpha > 1$ $Var(\theta) = \frac{\beta^2}{(\alpha-1)^2(\alpha-2)}$, $\alpha > 2$
Chi-squared $\theta \sim \chi^2_\nu$	df $\nu > 0$ (deg. of freedom)	$f(\theta) = \frac{2^{-\nu/2}}{\Gamma(\nu/2)}\theta^{\frac{\nu}{2}-1}\exp(-\theta/2)$ $\theta > 0$ (same as $\Gamma\left(\alpha = \frac{\nu}{2}, \beta = \frac{1}{2}\right)$)	$\mathbb{E}(\theta) = \nu$ $Var(\theta) = 2\nu$
Inverse Chi-squared $\theta \sim \chi^{-2}_\nu$	df $\nu > 0$ (deg. of freedom)	$f(\theta) = \frac{2^{-\nu/2}}{\Gamma(\nu/2)}\theta^{-\left(\frac{\nu}{2}+1\right)}\exp(-1/2\theta)$ $\theta > 0$ (same as $\Gamma^{-1}\left(\alpha = \frac{\nu}{2}, \beta = \frac{1}{2}\right)$)	$\mathbb{E}(\theta) = \frac{1}{\nu-2}$ $Var(\theta) = \frac{2}{(\nu-2)^2(\nu-4)}$
Dirichlet $\theta \sim Dir(\alpha_1, \ldots, \alpha_k)$	$\alpha_i > 0$; $\alpha_0 \equiv \sum_{i=1}^k \alpha_i$	$f(\theta) = \frac{\Gamma\left(\sum_{i=1}^k \alpha_i\right)}{\prod_{i=1}^k \Gamma(\alpha_i)}\prod_{i=1}^k \theta_i^{\alpha_i-1}$ $\theta_i > 0$; $\sum_{i=1}^k \theta_i = 1$	$\mathbb{E}(\theta_i) = \frac{\alpha_i}{\alpha_0}$ $Var(\theta_i) = \frac{\alpha_i(\alpha_0-\alpha_i)}{\alpha_0^2(\alpha_0+1)}$

Appendix B

NIMBLE

B.1 NIMBLE: Numerical Inference for Statistical Models for Bayesian and Likelihood Estimation

NIMBLE is a system for building and sharing analysis methods for statistical models, especially for hierarchical models and computationally-intensive methods. NIMBLE is built in R but compiles your models and algorithms using C++ for speed. It includes three components:

- *A system for using models written in BUGS model language as programmable objects in R.*
- *An initial library of algorithms for models written in BUGS, including basic MCMC, which can be used directly or can be customized from R before being compiled and run.*
- *A language embedded in R for programming algorithms for models, both of which are compiled through C++ code and loaded into R (from r-nimble.org).*

We will learn how to use `Nimble` in RStudio for Bayesian analysis practically, through examples presented and discussed in lectures and tutorials and relevant `R` code uploaded in Moodle. You can find further information and examples at https://r-nimble.org/. The Nimble manual and a Nimble cheatsheet are available also from Moodle. The very first step to start learning `Nimble` is to correctly install it on our computer! (see next section).

B.2 What is required for using the `R` package `Nimble`

Installing a C++ compiler

First, you need to install a C++ compiler on your computer for Nimble to work. Specifically, for the different operating systems, the Nimble Manual states:

MacOS

On MacOS, you should install Xcode. The command-line tools, which are available as a smaller installation, should be sufficient. This is freely available from the Apple developer site and the App Store. In the somewhat unlikely event you want to install from the source package rather than the CRAN binary package, the easiest approach is to use the source package provided at R-nimble.org. If you do want to install from the source package provided by CRAN, you'll need to install this gfortran package.

Linux

On Linux, you can install the GNU compiler suite (gcc/g++). You can use the package manager to

install pre-built binaries. On Ubuntu, the following command will install or update make, gcc and libc: *sudo apt-get install build-essential*

Windows

On Windows, download and install Rtools.exe available at https://cran.r-project.org/bin/windows/Rtools/. Select the appropriate executable corresponding to your version of R (and update your version of R if you notice it is not the most recent).

Putting Rtools on the PATH

You need to put the location of the Rtools 'make' utilities (bash, make, etc.) on the PATH. This can be done by including the following command in your R code:

```
writeLines('PATH="${RTOOLS40_HOME}\\usr\\bin;${PATH}"', con = "~/.Renviron")
```

Installing required R packages

Finally, you need to install the following R packages:

```
install.packages("nimble")
install.packages("igraph")
install.packages("coda")
install.packages("R6")
```

It is easier to use the R studio drop down commands to install the packages, rather than the commands above. Also, remember to 'load' the packages:

```
library(nimble)
library(igraph)
library(coda)
library(R6)
```