Contents

Chapter 7
Graphical Models for Complex Stochastic Systems

7.1 Introduction

In this chapter we describe of the use of graphical models in a Bayesian setting, in which parameters are treated as random quantities on equal footing with the random variables, enabling analysis of stochastic systems based on. As this has become one of the most successful applications of graphical models, we shall give a brief treatment here and refer to **?** for a more comprehensive discussion.

The paradigm used in Chapters **??**, **??** and **??** was that of identifying a joint distribution of a number of variables based on independent and identically distributed samples, with parameters unknown apart from restrictions determined by a log–linear, Gaussian, or mixed graphical model.

In contrast, Chap. **??** illustrated how a joint distribution for a Bayesian network may be constructed from a collection of conditional distributions; the network could be subsequently used to infer values of interesting unobserved quantities, given evidence, i.e. observations of other quantitites. As parameters and random variables are on an equal footing in the Bayesian paradigm, we may think of the interesting unobserved quantitites as the parameters and the evidence as data.

In the present chapter we take this idea to its consequence in a general statistical setting. We mainly focus on constructing full joint distributions of parameters, unobserved, and observed random variables by specifying a collection of conditional distributions for a graphical model determined by a directed acyclic graph with nodes representing all these quantities. Bayes' Theorem is then invoked to perform the necessary inference.

7.2 Bayesian Graphical Models

7.2.1 Simple Repeated Sampling

In the simplest possible setting we specify the joint distribution of a parameter θ and data x through a *prior distribution* $\pi(\theta)$ for θ and a conditional distribution $p(x \mid \theta)$ of data x for fixed value of θ, leading to the joint distribution

$$p(x, \theta) = p(x \mid \theta)\pi(\theta).$$

The prior distribution represents our knowledge (or rather uncertainty) about θ before the data have been observed. After observing that $X = x$ our *posterior distribution* $\pi^*(\theta)$ of θ is obtained by conditioning with the data x to obtain

$$\pi^*(\theta) = p(\theta|x) = \frac{p(x|\theta)\pi(\theta)}{p(x)} \propto L(\theta)\pi(\theta),$$

where $L(\theta) = p(x \mid \theta)$ is the *likelihood* . Thus *the posterior is proportional to the likelihood times the prior* and the normalizing constant is the marginal density $p(x) = \int p(x|\theta)\pi(\theta)d\theta$.

If the data is a sample $x = (x^1, x^2, x^3, x^4, x^5)$ we can represent this process by a small Bayesian network as shown to the left in Fig. 7.1. This network represents the model

$$p(x^1, \ldots, x^5, \theta) = \pi(\theta) \prod_{\nu=1}^{5} p(x^\nu \mid \theta).$$

reflecting that the individual observations are conditionally independent and identically distributed given θ. We can make a more compact representation of the network by introducing a *plate* which indicates repeated observations, such as shown to the right in Fig. 7.1.

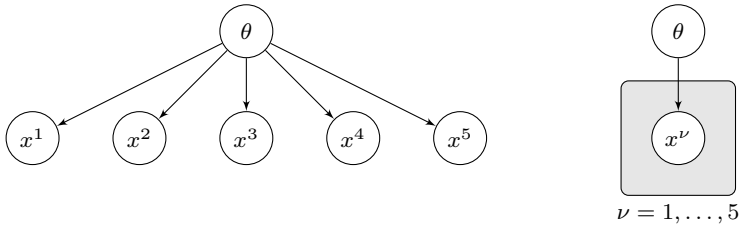
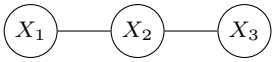


Fig. 7.1 Representation of a Bayesian model for simple sampling. The graph to the left indicates that observations are conditionally independent given θ; the picture to the right represents the same, but the plate allows a more compact representation.

For a more sophisticated example, consider a graphical Gaussian model given by the conditional independence $X_1 \perp\!\!\!\perp X_3 \,|\, X_2$ for fixed value of the concentration matrix K. In previous chapters we would have represented this model with its dependence graph:



However, in the Bayesian setting we need to include the parameters explicitly into the model, and could for example do that by the graph in Fig. 7.2.

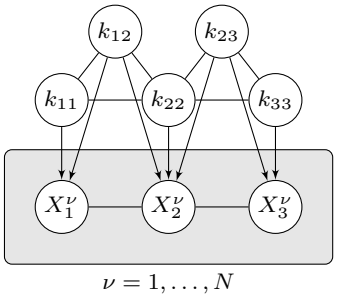


Fig. 7.2 A chain graph representing N independent observations of $X = (X_1, X_2, X_3)$ from a Bayesian graphical Gaussian model in which $X_1^\nu \perp\!\!\!\perp X_3^\nu \,|\, X_2^\nu, K$ and K follows a hyper Markov prior distribution.

The model is now represented by a chain graph, where the first chain component describes the structure of the prior distribution for the parameters in the concentration matrix. We have here assumed a so-called *hyper Markov prior distribution* (**?**): conditionally on k_{22}, the parameters (k_{11}, k_{12}) are independent of (k_{23}, k_{33}). The plate indicates that there are N independent observations of X, so the graph totally has $3N + 5$ nodes. The chain component on the plate reflects the factorization

$$f(x_1, x_2, x_3 \,|\, K) \propto \\ \det(K)^{1/2} \exp\{-(x_1^2 k_{11} + x_2^2 k_{22} + x_3^2 k_{33} + 2x_1 x_2 k_{12} + 2x_2 x_3 k_{23})/2\}$$

for each of the individual observations of $X = (X_1, X_2, X_3)$.

7.2.2 Models based on Directed Acyclic Graphs

A major feature of Bayesian graphical models is that explicitly including parameters and observations themselves in the graphical representation enables much more complex examples of observational patterns to be accommodated. Consider for example a linear regression model

$$Y_i \sim N(\mu_i, \sigma^2) \text{ with } \mu_i = \alpha + \beta x_i \text{ for } i = 1, \ldots, N.$$

To obtain a full probabilistic model we must specify a joint distribution for (α, β, σ) whereas the dependent variables x_i are assumed known (observed). If we specify independent distributions for these quantities, Fig. 7.3 shows a plate- based representation of this model with α, β, and σ being marginally independent and independent of Y_i.

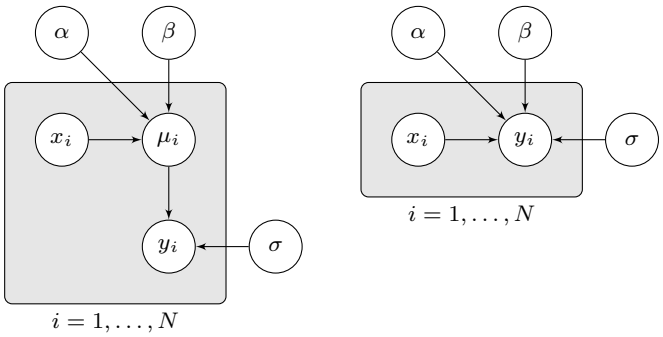


Fig. 7.3 Graphical representations of a traditional linear regression model with unknown intercept α, slope β, and variance σ^2. In the representation to the left, the means μ_i have been represented explicitly.

Note that μ_i are deterministic functions of their parents and the same model can also be represented without explicitly including these nodes. However, there can be specific advantages of representing the means directly in the graph. If the independent variables x_i are not centered, i.e. $\bar{x} \neq 0$, the model would change if x_i were replaced with $x_i - \bar{x}$, as α then would be the conditional mean when $x_i = \bar{x}$ rather than when $x_i = 0$, inducing a different distribution of μ_i.

To get a full understanding of the variety and complexity of models that can easily be described by DAGs with plates, we refer to the manual for `BUGS` (**?**), which also gives the following example.

Weights have been measured weekly for 30 young rats have weights measured during five weeks. The observations Y_{ij} are the weights of rat i measured at age x_j. The model is essentially a random effects linear growth curve:

$$Y_{ij} \sim \mathcal{N}(\alpha_i + \beta_i(x_j - \bar{x}), \sigma_c^2)$$

and

$$\alpha_i \sim \mathcal{N}(\alpha_c, \sigma_\alpha^2), \quad \beta_i \sim \mathcal{N}(\beta_c, \sigma_\beta^2),$$

where $\bar{x} = 22$. Interest particularly focuses on the intercept at zero time (birth), denoted $\alpha_0 = \alpha_c - \beta_c \bar{x}$. The graphical representation of this model is displayed in Fig. 7.4.

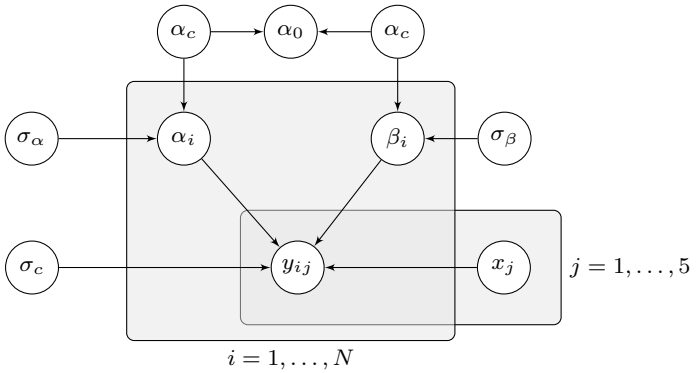


Fig. 7.4 Graphical representation of a random coefficient regression model for the growth of rats.

For a final illustration we consider the chest clinic example in Sect. **??**. Fig. 7.5 shows a directed acyclic graph with plates representing N samples from the chest clinic network.

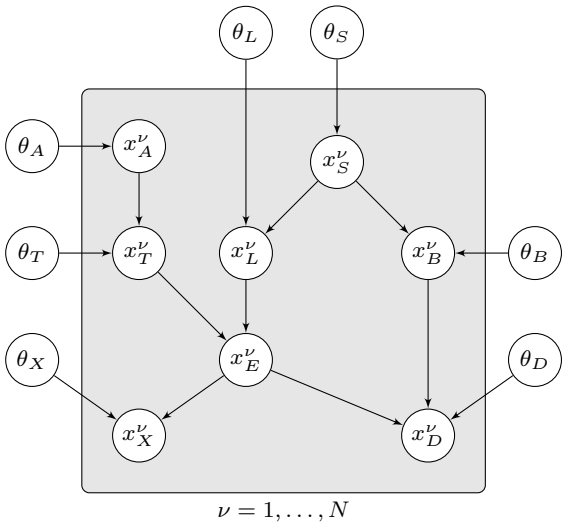


Fig. 7.5 A graphical representation of N samples from the chest clinic network, with parameters unknown and marginally independent for seven of the nodes.

Here we have introduced a parameter node for each of the variables. Each of these nodes may contain parameters for the conditional distribution of a node given any configuration of its parents, so that, following **?**, we would write for the joint model

$$p(x,\theta) = \prod_{v\in V} \pi(\theta_v) \prod_{\nu=1}^{N} p(x_v^\nu \mid x_{\mathrm{pa}(v)}^\nu, \theta_v).$$

7.3 Inference based on Probability Propagation

If the prior distributions of the unknown parameters are concentrated on a finite number of possibilities, i.e. the parameters are all discrete, the marginal posterior distribution of each of these parameters can simply be obtained by probability propagation in a Bayesian network with $7 + 8N$ nodes, inserting the observations as observed evidence. The moral graph of this network is shown in Fig. 7.6. This graph can be triangulated by just adding edges

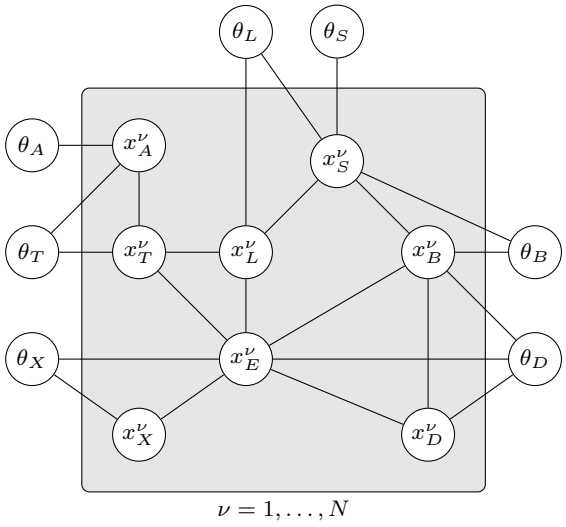


Fig. 7.6 Moral and triangulated graph of N samples from the chest clinic network, with seven unknown parameters.

between x_L^ν and x_B^ν and the associated junction tree would thus have $10N$ cliques of size at most 4. Thus, propagation would be absolutely feasible, even for large N.

We shall illustrate this procedure in the simple case of $N = 3$ where we only introduce unknown parameters for the probability of visiting Asia and the probability of a smoker having lung cancer, each having three possible levels, low, medium and high. We first define the parameter nodes

```
> library(gRain)
> lmh <- c("low","medium","high")
> thA<- cptable(~theta_A, values =c(1,1,1), levels=lmh)
```

```
> thL<- cptable(~theta_L, values =c(1,1,1), levels=lmh)
> param <- list(thA, thL)
```

and then specify a template for probabilities where we notice that A and L have an extra parent

```
> yn <- c("yes","no")
> a    <- cptable(~asia[i]|theta_A, values=c(1,99,2,98,5,95),levels=yn)
> t.a  <- cptable(~tub[i]|asia[i], values=c(5,95,1,99),levels=yn)
> s    <- cptable(~smoke[i], values=c(5,5), levels=yn)
> l.s  <- cptable(~lung[i]|smoke[i]:theta_L,
+                 values=c(5,95,1,99,1,9,1,99,1,4,1,99), levels=yn)
> b.s  <- cptable(~bronc[i]|smoke[i], values=c(6,4,3,7), levels=yn)
> e.lt <- cptable(~either[i]|lung[i]:tub[i],
+                 values=c(1,0,1,0,1,0,0,1),levels=yn)
> x.e  <- cptable(~xray[i]|either[i], values=c(98,2,5,95), levels=yn)
> d.be <- cptable(~dysp[i]|bronc[i]:either[i],
+                 values=c(9,1,7,3,8,2,1,9), levels=yn)
> plist.tmp <- list(a, t.a, s, l.s, b.s, e.lt, x.e, d.be)
```

We create 3 instances of the pattern defined above. In these instance the variable name `asia[i]` is replaced by `asia1`, `asia2` and `asia3` respectively.

```
> plate     <- repeatPattern(plist.tmp, instances=1:3)
```

We then proceed to the specification of the full network which is displayed in Fig. 7.7:

```
> plist   <- compileCPT(c(param, plate))
> chestlearn <-grain(plist)
> plot(chestlearn)
```

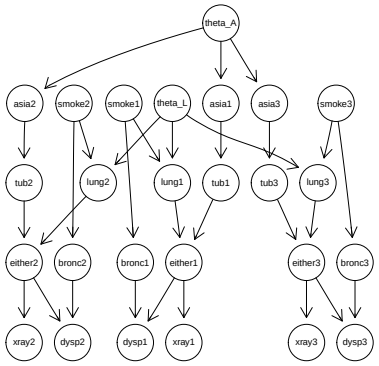


Fig. 7.7 Bayesian network for the chest clinic example with two unknown parameter nodes and two potential observations of the network. Parameters appear as nodes in the graph.

Finally we insert evidence for three observed cases, none of whom have been to Asia, all being smokers, one of them presenting with dyspnoea, one with a positive X-ray, one with dyspnoea and a negative X-ray; we then query the posterior distribution of the parameters:

```
> chestlearn.ev<- setFinding(chestlearn,
+  nodes = c("asia1","smoke1","xray1"), c("no","yes","yes"))
> chestlearn.ev<- setFinding(chestlearn.ev,
+  nodes = c("asia2","smoke2","dysp2"), c("no","yes","yes"))
> chestlearn.ev<- setFinding(chestlearn.ev,
+  nodes = c("asia3","smoke3","dysp3","xray3"), c("no","yes","yes","no"))
> querygrain(chestlearn.ev,nodes =c("theta_A","theta_L"))

$theta_A
theta_A
   low medium   high
0.3504 0.3399 0.3096

$theta_L
theta_L
   low medium   high
0.2211 0.3099 0.4690
```

We see that the probabilities of visiting Asia is now more likely than before to be low, whereas the probability of having lung cancer for a smoker is more likely to be high.

In the special case where all cases have been completely observed, it is not necessary to form the full network with $7 + 8N$ nodes, but updating can be performed sequentially as follows.

Let $p_n^*(\theta)$ denote the posterior distribution of θ given n observations $x^1, \ldots, x^n$, i.e. $p_n^*(\theta) = p(\theta \,|\, x^1, \ldots, x^n)$. We then have the recursion:

$$\begin{aligned} p_n^*(\theta) \propto p(x^1, \ldots, x^n, \theta) &= \Big\{ \prod_{\nu=1}^{n} p(x^\nu \,|\, \theta) \Big\} p(\theta) \\ &= p(x^n \,|\, \theta) \Big\{ \prod_{\nu=1}^{n-1} p(x^\nu \,|\, \theta) \Big\} p(\theta) \\ &\propto p(x^n \,|\, \theta) p_{n-1}^*(\theta). \end{aligned}$$

Hence we can incorporate evidence from the n-th observation by using the posterior distribution from the $n-1$ first observations as a prior distribution for a network respresenting only a single case. It follows from the moral graph in Fig. 7.6 that if all nodes in the plates are observed, the seven parameters are conditionally independent also in the posterior distribution after n observations. If cases are incomplete, such a sequential scheme can only be used approximately (**?**).

7.4 Computations using Monte Carlo Methods

In most cases the posterior distribution

$$\pi^*(\theta) = p(\theta|x) = \frac{p(x|\theta)\pi(\theta)}{p(x)} \propto p(x|\theta)\pi(\theta) \tag{7.1}$$

of the parameters of interest cannot be calculated or represented in a simple fashion. This would for example be the case if the parameter nodes in Fig. 7.5 had values in a continuum and the observations of a case would be incomplete, such as in the example given in the previous section.

In such models one will often resort to Markov chain Monte Carlo (MCMC) methods: we cannot calculate $\pi^*(\theta)$ analytically but if we can generate samples $\theta^{(1)}, \ldots, \theta^{(M)}$ from the distribution $\pi^*(\theta)$, we can do just as well.

7.4.1 Metropolis–Hastings and the Gibbs Sampler

Such samples can be generated by the Metropolis–Hastings algorithm. In the following we change the notation slightly.

We suppose that we know $p(x)$ only up to a normalizing constant. That is to say, $p(x) = k(x)/c$, where $k(x)$ is known but c is unknown. We partition x into blocks, for example $x = (x_1, x_2, x_3)$.

We wish to generate samples $x^1, \ldots, x^M$ from $p(x)$. Suppose we have a sample $x^{t-1} = (x_1^{t-1}, x_2^{t-1}, x_3^{t-1})$ and also that x_1 has also been updated to x_1^t in the current iteration. The task is to update x_2. To do so we need to specify a proposal distribution h_2 from which we can sample candidate values for x_2. The *single component Metropolis–Hastings* algorithm works as follows:

1. `Draw` $x_2 \sim h_2(\cdot \,|\, x_1^t, x_2^{t-1}, x_3^{t-1})$. `Draw` $u \sim U(0,1)$.
2. `Calculate acceptance probability`

$$\alpha = \min(1, \frac{p(x_2 \,|\, x_1^t, x_3^{t-1}) h_2(x_2^{t-1} \,|\, x_1^t, x_2, x_3^{t-1})}{p(x_2^{t-1} \,|\, x_1^t, x_3^{t-1}) h_2(x_2 \,|\, x_1^t, x_2^{t-1}, x_3^{t-1})}) \tag{7.2}$$

3. `If` $u < \alpha$ `set` $x_2^t = x_2$; `else set` $x_2^t = x_2^{t-1}$.

The samples $x^1, \ldots, x^M$ generated this way will form an ergodic Markov chain that, under certain conditions, has $p(x)$ as its stationary distribution so that the expectation of any function of x can be calculated approximately as

$$\int f(x)p(x)\,dx = \lim_{M\to\infty} \frac{1}{M} \sum_{\nu=1}^{M} f(x^\nu) \approx \frac{1}{M} \sum_{\nu=1}^{M} f(x^\nu).$$

Note that $p(x_2 \,|\, x_1^t, x_3^{t-1}) \propto p(x_1^t, x_2, x_3^{t-1}) \propto k(x_1^t, x_2, x_3^{t-1})$ and therefore the acceptance probability can be calculated even though $p(x)$ may only be known up to proportionality.

A special case of the single component Metropolis–Hastings algorithm is the *Gibbs–sampler* : If as proposal distribution h_2 we choose $p(x_2 \,|\, x_1^t, x_3^{t-1})$ then the acceptance probability becomes 1 because terms cancel in 7.2. The

conditional distribution of a single component X_2 given all other components (X_1, X_3) is known as the *full conditional* distribution.

For a directed graphical model, the density of full conditional distributions can be easily identified:

$$f(x_i \,|\, x_{V\setminus i}) \propto \prod_{v\in V} f(x_v \,|\, x_{\mathrm{pa}(v)}) \\ \propto f(x_i \,|\, x_{\mathrm{pa}(i)}) \prod_{v\in \mathrm{ch}(i)} f(x_v \,|\, x_{\mathrm{pa}(v)}) = f(x_i \,|\, x_{\mathrm{bl}(i)}), \quad (7.3)$$

x where $\mathrm{bl}(i)$ is the *Markov blanket* of node i:

$$\mathrm{bl}(i) = \mathrm{pa}(i) \cup \mathrm{ch}(i) \cup \left\{\cup_{v\in\mathrm{ch}(i)} \mathrm{pa}(v) \setminus \{i\}\right\}$$

or, equivalently, the neighbours of i in the moral graph, see Sec. **??**. Note that (7.3) holds even if some of the nodes involved in the expression corresponds to values that have been observed. To sample from the posterior distribution of the unobserved values given the observed ones, only unobserved variables should be updated in the Gibbs sampling cycle.

In this way, a Markov chain of pseudo-observations from all unobserved variables is generated, and those corresponding to quantities (parameters) of interest can be monitored.

7.4.2 Using WinBUGS via R2WinBUGS

The program `WinBUGS` (**?**) is based on the idea that the user specifies a Bayesian graphical model based on a DAG, including the conditional distribution of every node given its parents. `WinBUGS` then identifies the Markov blanket of every node and using properties of the full conditional distributions in (7.3), a sampler is automatically generated by the program. As the name suggests, `WinBUGS` is available on Windows platforms only. `WinBUGS` can be interfaced from `R` via the **R2WinBUGS** package (**?**) and to do this, `WinBUGS` must be installed. **R2WinBUGS** works by calling `WinBUGS`, doing the computations there, shutting `WinBUGS` down and returning control to `R`.

A specification of the model described in Fig. 7.3 in the `BUGS` language looks as follows (notice that the dispersion of a normal distribution is parameterized in terms of the concentration τ where $\tau = \sigma^{-2}$):

```
model {
  for (i in 1:N) {
    Y[i] ~ dnorm(mu[i],tau)
    mu[i] <- alpha + beta*(x[i] - x.bar)
  }
  x.bar <- mean(x[])
```

```
  alpha ~ dnorm(0, 1.0E-6)
  beta  ~ dnorm(0, 1.0E-6)
  sigma ~ dunif(0,100)
  tau   <- 1/pow(sigma,2)
}
```

BUGS comes with a Windows interface in the program WinBUGS. To analyse this model in R we can use the package R2WinBUGS. First we save the model specification to a plain text file:

```
> cat(
+ "model {
+   for (i in 1:N) {
+     Y[i] ~ dnorm(mu[i],tau)
+     mu[i] <- alpha + beta*(x[i] - x.bar)
+   }
+   x.bar <- mean(x[])
+   alpha ~ dnorm(0, 1.0E-6)
+   beta  ~ dnorm(0, 1.0E-6)
+   sigma ~ dunif(0,100)
+   tau   <- 1/pow(sigma,2)
+ }",
+ file="linesModel.txt" )
```

We specify data:

```
> Y <- c(1,3,3,3,5)
> x <- c(1,2,3,4,5)
> N <- 5
```

As the sampler must start somewhere, we specify initial values for the unknowns:

```
> p.ini <- list(alpha = 0, beta = 0, sigma = 1)
```

We may now ask WinBUGS for a sample from the model :

```
> library(R2WinBUGS)
> lines.res <-
+   bugs(data = list( Y=Y, x=x, N=N ),
+        inits    = list( p.ini ),
+        param    = c("alpha","beta","sigma"),
+        model    = "linesModel.txt",
+        n.chains = 1,
+        ## Total number of samples, including burn-in:
+        n.iter   = 7000,
+        ## Burn-in values; will be discarded in subsequent analyses:
+        n.burnin = 5000,
+        ## Of the non-discarded samples only every 'n.thin'th will be used.
+        n.thin   = 5,
+        bugs.directory = "c:/Programs/WinBUGS14/",
+        debug    = F,
+        clearWD  = TRUE )
```

The outout lines.res contains the samples. A simple summary of the samples is

```
> print(lines.res)
```

```
Inference for Bugs model at "linesModel.txt", fit using WinBUGS,
 1 chains, each with 7000 iterations (first 5000 discarded), n.thin = 5
 n.sims = 400 iterations saved
         mean  sd 2.5%  25%  50%  75% 97.5%
alpha     3.0 1.0  1.7  2.7  3.0  3.3   4.6
beta      0.9 0.7 -0.1  0.6  0.8  1.0   2.3
sigma     1.5 2.1  0.5  0.7  1.0  1.5   6.2
deviance 14.4 5.3  9.0 10.8 12.8 16.4  28.5

DIC info (using the rule, pD = Dbar-Dhat)
pD = 0.2 and DIC = 14.7
DIC is an estimate of expected predictive error (lower deviance is better).
```

We next convert the output to a format suitable for analysis with the `coda` package:

```
> library(coda)
> lines.coda <- as.mcmc.list(lines.res)
```

An summary of the posterior distrubution of the monitored parameters is as follows:

```
> summary(lines.coda)

Iterations = 5001:6996
Thinning interval = 5
Number of chains = 1
Sample size per chain = 400

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

           Mean    SD Naive SE Time-series SE
alpha     2.980 1.037   0.0518         0.0525
beta      0.887 0.735   0.0367         0.0465
deviance 14.425 5.307   0.2654         0.3996
sigma     1.534 2.139   0.1070         0.1536

2. Quantiles for each variable:

           2.5%    25%    50%   75% 97.5%
alpha     1.708  2.708  3.023  3.28  4.56
beta     -0.065  0.598  0.813  1.04  2.35
deviance  9.046 10.837 12.775 16.41 28.47
sigma     0.459  0.740  1.002  1.49  6.16
```

As the observations are very informative, the posterior distributions of the regression parameters α and β are similar to the sampling distributions obtained from a standard linear regression analysis:

```
> summary(lm(Y~I(x-mean(x))))

Call:
lm(formula = Y ~ I(x - mean(x)))

Residuals:
        1         2         3         4         5
-4.00e-01  8.00e-01  4.84e-17 -8.00e-01  4.00e-01

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)       3.000      0.327    9.19   0.0027 **
I(x - mean(x))    0.800      0.231    3.46   0.0405 *
```

```
---
Signif. codes:  0 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1

Residual standard error: 0.73 on 3 degrees of freedom
Multiple R-squared:  0.8,        Adjusted R-squared: 0.733
F-statistic:   12 on 1 and 3 DF,  p-value: 0.0405
```

A *traceplot* (see Fig. 7.8) of the samples is useful for visual inspection of indications that the sampler has not converged. There appears to be no problem here:

```
> library(coda)
> par(mfrow=c(2,2))
> traceplot(lines.coda)
```

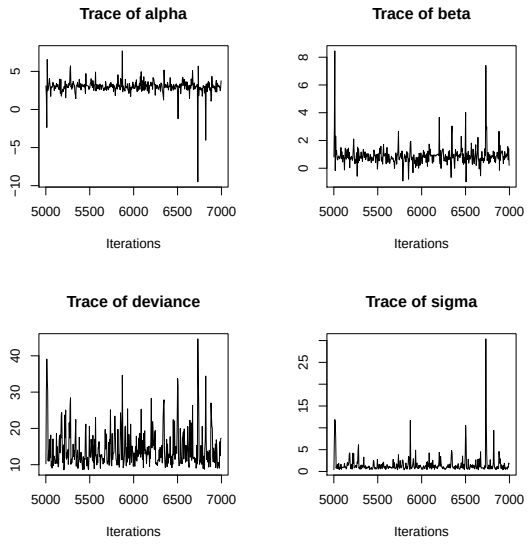


Fig. 7.8 A traceplot of the samples produced by `BUGS` is a useful tool for visual inspection of indications of that the sampler has not converged.

A plot of the marginal posterior densities (see Fig. 7.9) provides a supplement to the numeric summaries shown above:

```
> par(mfrow=c(2,2))
> densplot(lines.coda)
```

7.5 Various

An alternative to `WinBUGS` is `OpenBUGS` (**?**). The two programs have the same genesis and the model specification languages are very similar. `OpenBUGS` can

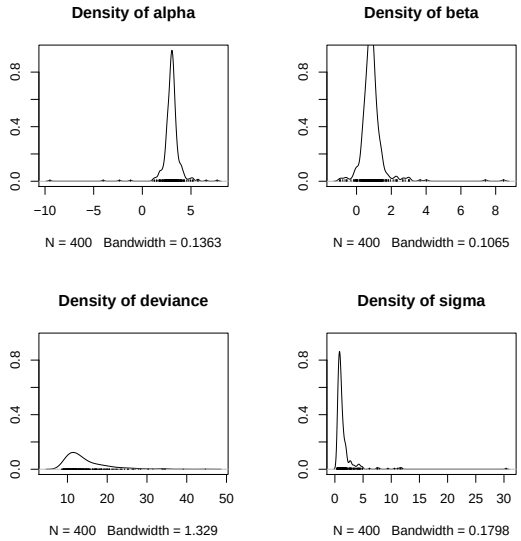


Fig. 7.9 A plot of each posterior marginal distribution provides a provides a supplement to the numeric summary statistics.

be interfaced from `R` via the **BRugs** package and `OpenBUGS` / **BRugs** is available for all platforms. The modus operandi of **BRugs** is fundamentally different from that of **WinBUGS**: A sampler created using **BRugs** remains alive in the sense that one may call the sampler repeatedly from within `R`. Yet another alternative is package **rjags** which interfaces the `JAGS` program; this must be installed separately and is available for all platforms.

List of Corrections