Bayesian Modeling Using Winbugs
Ioannis Ntzoufras
Wiley, 2009

in the noninformative case and in the informative case, to

$$\theta^s|\boldsymbol{y} \sim \text{gamma}(8.168, 7.618) \text{ and } \theta^c|\boldsymbol{y} \sim \text{gamma}(5.001, 8.618).$$

In such problems, we might also be interested in inferences about the difference of the means $\theta^s - \theta^c$ as well as making predictions based on the probability of winning a game. The first issue can easily be solved by simply sampling from each gamma distribution, calculating the difference, and estimating the posterior distribution. This is a simple implementation of simulation and Monte Carlo techniques, presented in the following sections. To describe the second issue, we first need to define the predictive distribution, which is presented in Chapter 10.

Example 1.3. Estimating the prevalence of a disease (Bernoulli data). Suppose that we are interested in the estimation of the prevalence of a disease, for example, coronary heart disease, for men aged 30–40. Data from a prospective study of sample size equal to 1250 men have indicated 5 men experiencing at least one incident. We are interested on estimating the prevalence rate of the disease for this specific age group.

Following (1.2), using a beta$(0.01, 0.01)$ low-information prior, we have

$$\pi|\boldsymbol{y} \sim \text{beta}(5.01, 1245.01),$$

with mean equal to 4 incidents over 1000 men and standard deviation equal to 1.89 incidents over 1000 men; see Figure 1.3 for graphical representation of the posterior distribution of prevalence. Alternatively, we may use the uniform $U(0, 1)$ distribution as a noninformative prior. Under this prior setup, the posterior is $\pi|\boldsymbol{y} \sim \text{beta}(6, 1246)$, with posterior mean of the prevalence rate equal to 4.8 incidents per 1000 males and standard deviation equal to 1.95 incidents per 1000 males.

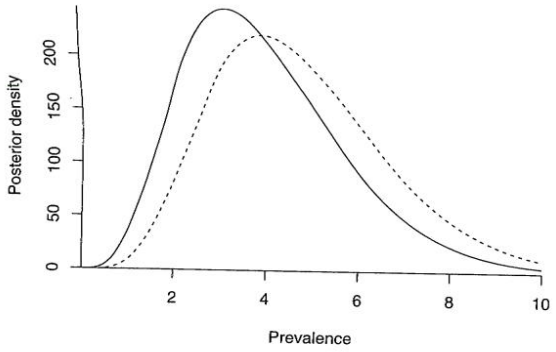


Figure 1.3 *Posterior distribution of prevalence rate (per 1000 males) for coronary heart disease: solid line — beta(0.01, 0.01) prior; dashed line — uniform $U(0, 1)$ prior.*

A simple credible interval can be given by the 2.5% and 97.5% quantiles of the beta distribution. In our case, the prevalence will be in the interval $(1.3, 8.2)$ with probability 95% for the beta$(0.01, 0.01)$ and $(1.76, 9.3)$ for the uniform prior.

Example 1.4. Kobe Bryant's field goals in NBA (binomial data). In this example we consider the field goals and the corresponding attempts of a successful NBA player, Kobe Bryant. Data were downloaded from the Yahoo sports page and are presented in Table 1.3.
Having observed the data of Table 1.3, we are interested in comparing the performance of the player in terms of the success probability (or percentage) of field goals across different seasons.

Table 1.3 Kobe Bryant's field goals for seasons 1999–2007

		Field goals		
Season	Games	Successes	Attempts	Percent (%)
1999/00	66	554	1183	46.8
2000/01	68	701	1510	46.4
2001/02	80	749	1597	46.9
2002/03	82	868	1924	45.1
2003/04	65	516	1178	43.8
2004/05	66	573	1324	43.3
2005/06	80	978	2173	45.0
2006/07	42	399	845	47.2
Total	749	5338	11,734	45.5

Source: Yahoo sports.

In Table 1.4, we present the posterior details, including the parameters of the beta distribution, the means and standard deviations, as well as the 95% posterior intervals. In all calculations a beta prior distribution with parameters equal to 0.01 has been used; 95% posterior intervals (based on the 2.5% and 97.5% posterior percentiles) are also depicted using error bars in Figure 1.4 in order to clearly compare the Kobe Bryant's performance across different seasons.

Table 1.4 Posterior summaries for Kobe Bryant's field goals for seasons 1999–2007

				Posterior parameters		Posterior summaries of field goal success percentage			
Year	Games	Successes	Attempts	$\widetilde{a}$	$\widetilde{b}$	Mean	SD[a]	$Q_{0.025}$	$Q_{0.975}$
1999/00	66	554	1183	554.01	629.01	46.8	1.5	44.0	49.7
2000/01	68	701	1510	701.01	809.01	46.4	1.3	43.9	48.9
2001/02	80	749	1597	749.01	848.01	46.9	1.2	44.5	49.4
2002/03	82	868	1924	868.01	1056.01	45.1	1.1	42.9	47.3
2003/04	65	516	1178	516.01	662.01	43.8	1.4	41.0	46.6
2004/05	66	573	1324	573.01	751.01	43.3	1.4	40.6	46.0
2005/06	80	978	2173	978.01	1195.01	45.0	1.1	42.9	47.1
2006/07	42	399	845	399.01	446.01	47.2	1.7	43.9	50.6
Total	749	5338	11734	5338.01	6396.01	45.5	0.5	44.6	46.4

[a]Standard deviation.
Source: Yahoo sports.

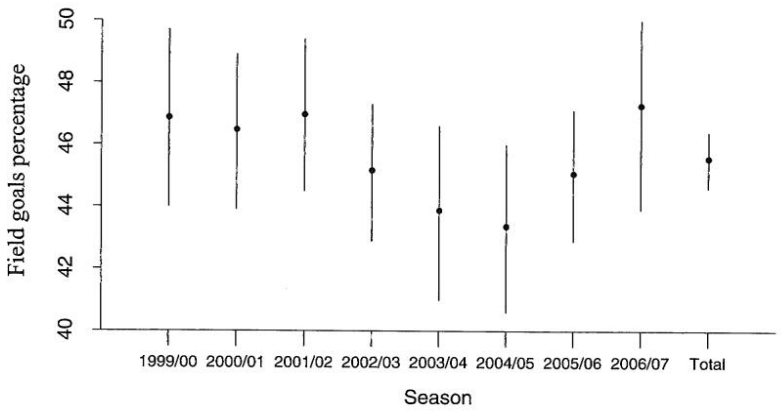


Figure 1.4 95% posterior credible intervals for Kobe Bryant's field goals for seasons 1999–2007; prior =beta(0.01, 0.01).

Updating information over the seasons. In the analysis above we compared the performance of the player across different seasons. In Bayesian analysis, we may also update our posterior after the end of each season. This will result in improving our confidence for the player's performance (although we do not consider other factors that may be important, such as the age of the player).

In Table 1.5 and Figure 1.5 we can observe how posterior distribution is updated after incorporating sequentially data information of each year. As expected, the final posterior distribution coincides with the corresponding posterior, which combines all seasons' data.

Table 1.5 Summaries of the updated posterior distributions for Kobe Bryant's field goals for seasons 1999–2007

				Posterior parameters		Posterior summaries of field goal success percentage			
Year	Games	Successes	Attempts	$\widetilde{a}$	$\widetilde{b}$	Mean	SD[a]	$Q_{0.025}$	$Q_{0.975}$
1999/00	66	554	1183	554	629	46.8	1.5	44.0	49.7
2000/01	68	701	1510	1255	1438	46.6	1.0	44.7	48.5
2001/02	80	749	1597	2004	2286	46.7	0.8	45.2	48.2
2002/03	82	868	1924	2872	3342	46.2	0.6	45.0	47.5
2003/04	65	516	1178	3388	4004	45.8	0.6	44.7	47.0
2004/05	66	573	1324	3961	4755	45.4	0.5	44.4	46.5
2005/06	80	978	2173	4939	5950	45.4	0.5	44.4	46.3
2006/07	42	399	845	5338	6396	45.5	0.5	44.6	46.4

[a]Standard deviation.
Source: Yahoo sports.

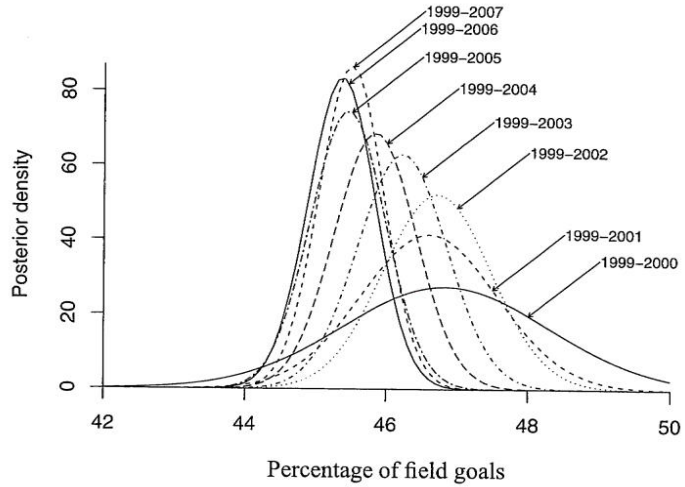


Figure 1.5 Updated posterior densities for Kobe Bryant's percentage of field goals for seasons 1999–2007; prior=beta(0.01, 0.01).

Example 1.5. Body temperature data (normal data). In this example we consider data of Mackowiak et al. (1992), who examined whether the true mean body temperature is 98.6 °F. The same data were reanalyzed and presented by Shoemaker (1996). In the present example, we consider the normal model for inferences about both the mean and the variance of the body temperature.

We adopt the normal model with $y_i \sim N(\mu, \sigma^2)$ and a normal–gamma prior with parameters $\mu = 98.6$, $c = 100$, $a = 0.001$ and $b = 0.001$. This is a low-information prior (since the variance is large) with the mean temperature centered around 98.6 °F which is the widely accepted value.

The sample size, mean, and variance are equal to

$$n = 130, \ \ \overline{y} = 98.24923$$

and

$$s^2 = 0.5375575 \Leftrightarrow \text{SS} = (130 - 1) \times 0.5375575 = 69.34492,$$

respectively, resulting in an NIG posterior distribution for the mean and the variance of temperature with parameters

$$\begin{aligned}
\widetilde{\mu} &= 0.999923 \times 98.24923 + (1 - 0.999923) \times 98.6 = 98.249 \\
\widetilde{c} &= \frac{w}{n} = \frac{0.999923}{153} = 0.007692 \\
\widetilde{a} &= \frac{150}{2} + 0.001 = 65.001 \\
\widetilde{b} &= \frac{69.34492}{2} + 0.001 = 34.6735 .
\end{aligned}$$

CHAPTER 4

WinBUGS SOFTWARE: ILLUSTRATION, RESULTS, AND FURTHER ANALYSIS

In this chapter we provide a full example into WinBUGS and additional, more specialized, functions, and tools in WinBUGS . The chapter is divided in five sections. In Section 4.1 we analyze the data of Example 1.4 (Kobe Bryant's's data) using the basic functions of WinBUGS introduced in the previous chapter. In Section 4.2 we proceed to describe the remaining functions of the inference menu, which offers a variety of tools for the analysis of the MCMC output. Generation of multiple chains is described in Section 4.3. Finally, control of figure properties and the remaining tools, and menus are presented in the last two sections of the chapter. All functions, tools and menus are illustrated using Example 1.4 (Kobe Bryant's's data) and the model fitted in Section 4.1.

4.1 A COMPLETE EXAMPLE OF RUNNING MCMC IN WinBUGS FOR A SIMPLE MODEL

4.1.1 The model

In this section we will consider Example 1.4. Let us examine again the Kobe Bryant success probabilities but now, instead of the original parameterization, alternatively consider the following expression of the success probabilities

$$\pi_k = \pi_{k-1} \times R_k \tag{4.1}$$

for $k = 2, \ldots, 8$ (corresponding to seasons 2000/01, ..., 2006/07), while π_1 will be estimated individually and will denote the success probability for season 1999. Quantities R_k

Bayesian Modeling Using WinBUGS, by Ioannis Ntzoufras

denote the performance of the current season (in terms of success probabilities) in comparison to the previous season. We use a beta prior for the success probability of the first season and gamma priors for the relative success measures R_k. In both cases we use low parameter values to express prior ignorance. Thus, we use

$$\pi_1 \sim \text{beta}(0.01,\ 0.01) \quad \text{and} \quad R_k \sim \text{gamma}(0.01,\ 0.01) \quad \text{for } k = 2, \ldots, 8.$$

This prior setup can be simply defined in WinBUGS code using the commands

```
pi[1] ~ dbeta( 0.01, 0.01)
R[1]  <- 1
for (k in 2:YEARS){
    R[k] ~ dgamma( 0.01, 0.01)
}
```

Note that if we use a vector to specify R_k, then we also need to define R_1. For this reason, we assign it a dummy value equal to one.

The likelihood for this model is given by

$$f(\boldsymbol{y}|\pi_1, \ldots, \pi_8) = \prod_{i=1}^{8} \pi_i^{y_i}(1-\pi)^{N_i - y_i},$$

with π_k given by (4.1). This likelihood will be written in WinBUGS code as

```
for (k in 1:YEARS){ y[k]  ~ dbinom( pi[k], N[k] ) }
```

while Eq. (4.1) will be expressed by the following syntax:

```
for (k in 2:YEARS){ pi[k]<-pi[k-1]*R[k] }
```

Hence the complete model code is given as follows:

```
model {
    # stochastic part of the likelihood
    for (k in 1:YEARS){ y[k]  ~ dbin( pi[k], N[k] ) }
    # deterministic part of the likelihood
    for (k in 2:YEARS){ pi[k]<-pi[k-1]*R[k] }
    # prior for pi[1]
    pi[1] ~ dbeta( 0.01, 0.01)
    # dummy value for R[1]
    R[1]  <- 1
    # Prior for R[k]
    for (k in 2:YEARS){R[k] ~ dgamma( 0.01, 0.01) }
}
```

In this code we can avoid multiple loops by including all commands related to the model in a single loop. Thus, the following code will have exactly the same effect as the previous code:

```
model {
    # stochastic likelihood part for the first observation
    y[1]  ~ dbin( pi[1], N[1] )
    for (k in 2:YEARS){
        # stochastic part of the likelihood
        y[k]  ~ dbin( pi[k], N[k] )
        # deterministic part of the likelihood
        pi[k]<-pi[k-1]*R[k]
        # Prior for R[k]
        # dummy value for R[1]
        R[1]  <- 1
        # prior for pi[1]
        pi[1] ~ dbeta( 0.01, 0.01)
}
```


Although the final result will be the same in terms of WinBUGS , the model code cannot be followed as easily as the first one. We recommend writing model codes that are easy to follow, including detailed and descriptive comments. A clear, well-written model code accompanied by meaningful comments facilitates future reuse by its author and smooth implementation by other users.

4.1.2 Data and initial values

For illustration, we use a mixed data format: a combination of rectangular and list formatted data. In the list format, only the number of available seasons (i.e., the number of binomial experiments) is specified. The successes and the total attempts (y_i and N_i) for each season are defined as vectors in the following rectangular data format:

```
list(YEARS=8)
y[]  N[]
554  1183
701  1510
749  1597
868  1924
516  1178
573  1324
978  2173
399   845
END
```

Alternatively, a single list format can be used:

```
list(YEARS=8, y=c(554,701,749,868,516,573,978,399),
              n=c(1183,1510,1597,1924,1178,1324,2173, 845) )
```

Initial values values must be specified for π_1 and R_k ($k = 2, \ldots, 8$) using the following syntax:

```
list(pi=c(0.5, NA, NA, NA, NA, NA, NA, NA), R=c(NA,1,1,1,1,1,1,1) )
```

In this last syntax, initial values have been specified only for the first component of vector $\boldsymbol{\pi}$ and for all components of $\boldsymbol{R}$ except the first one. No initial values can be specified for $\pi_2, \ldots, \pi_8$ and R_1 since the they are deterministic and constant nodes, respectively. The value `NA` represents missing (unavailable) data.

4.1.3 Compiling and running the model

To compile and run the MCMC algorithm for 1000 burnin and 2000 iterations finally kept, we follow the steps below.

1. Highlight the word `model` in the model code window.
2. Open the `model specification` tool following the path `Model> Specification`.
3. Click on `check model` box.

4. If the message `model is syntactically correct` appears in the left bottom bar of the WinBUGS window, then proceed to step 5 (data load). Otherwise, correct model code according to the indication given by WinBUGS (the cursor will indicate the position of the false code).

5. Load the list format data by clicking on the `list` command and then press the `load data` button at the model specification tool. The message `data loaded` must appear if the data syntax is correct.

6. Load the rectangular formatted data by clicking on the header row (i.e., the names of the columns — vectors) and then press the `load data` button at the model specification tool. The message `data loaded` must appear (again) if the data syntax is correct.

7. Click on the `compile` button of the model specification tool. If no error is found, then the message `model compiled` will appear in the bottom left of the window. Otherwise correct the error indicated by WinBUGS and rerun everything (from the beginning).

8. Load initial values by highlighting the word `list` and then press the button `load inits` at the model specification tool.

9. If the message `model initialized` appears in the bottom left of the screen, then the algorithm is ready to run.

10. Open the `update` tool by following the path `Model>Update`.

11. Select the number of burnin iterations (updates), the refresh lag and thin, and press the `update` button (for this example we use updates=1000, refresh=100, thin=1). The index `iteration` will begin counting the number of iterations. It will stop when the number of iterations specified in the `updates` text box is reached.

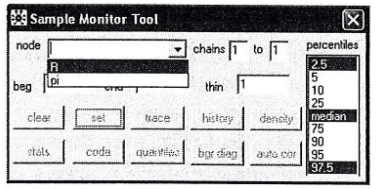
12. Open the `sample monitor` tool in the path `Inference>Samples`.

13. Set the parameters that we wish to monitor (vector π and components 2–8 of vector $\boldsymbol{R}$). In the sample monitor tool, write in the `node` text box `pi` (the name of π) and press the `set` button. Repeat the same for components 2–8 of vector $\boldsymbol{R}$ by inserting the name `R[2:8]` in the `node` text box. Repeat the same procedure for all additional nodes and parameters that you wish to monitor.

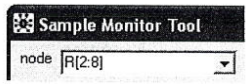
14. Open the online trace plots. In the sample monitor tool, type the star character (asterisk, `*`) in the `node` text box and press the `trace` button. A window with the online plots will appear.

15. Return to the update tool and set the number of iterations equal to 2000. Then press the `update` button. The index `iteration` will start (again), increasing until the desired number iterations is achieved. During the update the online plot will depict the generation of the monitored values.

16. We can now use the `sample monitor` tool to obtain some initial output analysis or extract data in other statistical software.
In order to get analysis for a single monitored parameter, we need to type the name of the corresponding WinBUGS node in the `Node` text box of the `sample monitor` tool and then press the option or button we wish to implement. The names of the monitored parameters can be also selected from the list of monitored nodes that is available at the right of the `node` text box.



If we wish to select specific components of a matrix or array, we can extract them using the `I:J` syntax. For example, `R[2:8]` will select components 2–8 of vector `R`.

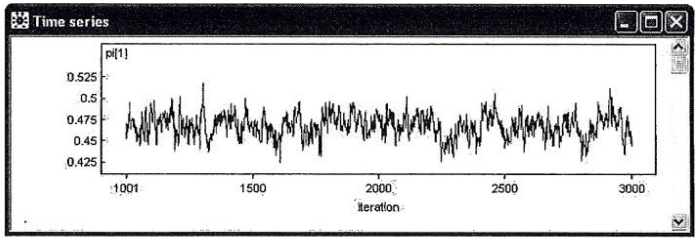


If we wish to implement the same analysis for all monitored parameters, then we type the asterisk (`*`) character in the `node` text box instead of the name of a specific node.

4.1.4 MCMC output analysis and results

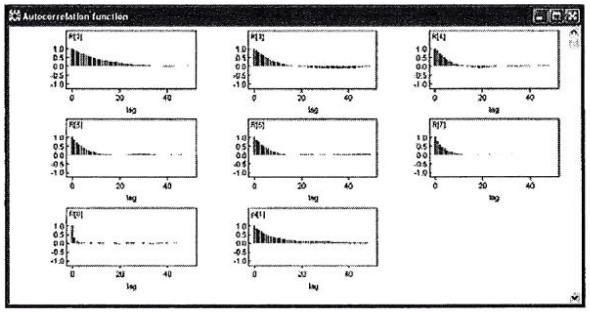
4.1.4.1 Checking convergence. The following analysis can be obtained using `sample monitor` tool in WinBUGS.

We first produce trace plots for all monitored parameters by typing the asterisk (`*`) in the `node` text box of the monitor tool. We then press the `History` button. The trace plot of π_1 follows for illustration.

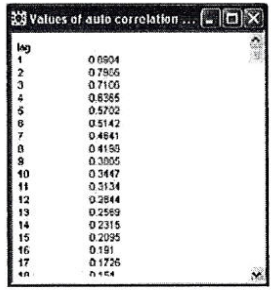


No patterns or irregularities are observed, and therefore convergence can be assumed. (see sections 2.2.2.4 and 3.6 for details).

We then monitor autocorrelations by pressing the `auto cor` button.

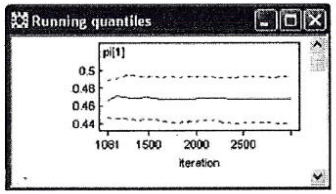


From this window, we observe that autocorrelations for all parameters become low only after considering a lag equal to 30. Thus, an independent sample can be obtained by re-running the algorithm with `thin` set equal to 30 at the update tool.



Numerical values of the calculated autocorrelations can be obtained by selecting the autocorrelation plot (by double-clicking on its picture) and then pressing the keyboard Control key and the left mouse button. A window with all autocorrelation values of the selected parameter will appear on the screen.

Then we monitor the evolution of selected quantiles using the `quantiles` button. This plot indicates that the requested quantiles have been stabilized, implying that the algorithm has converged in terms of π_1. Note that in this example we cannot use the option `bgr diag` since only a single chain was generated.



Using the `stats` option, Monte Carlo errors can be calculated. They measure the variation of the mean of the parameter of interest due to the simulation. If MC errors are low in comparison to the corresponding estimated posterior standard deviations, then the estimated the posterior mean was estimated with high precision. Increasing the number of iterations will decrease MC error. Here for parameter π_1 we get MC error= 0.00116, while the posterior standard deviation is 0.0135 (for a sample of 2000 iterations after discarding an additional 1000 iterations as burnin), which is equal to the 8.6% of the posterior standard deviation.

Node statistics

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
pi[1]	0.4674	0.01348	0.001164	0.4401	0.4679	0.4931	1001	2000

Increasing the number of iterations to 10,000 will considerably decrease MC error to 0.00056 (48% decrease), while the posterior standard deviation will be relatively stable and equal to 0.0150. Note that the Monte Carlo variability of the remaining posterior summaries is minimal.

Node statistics

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
pi[1]	0.4687	0.01449	5.648E-4	0.441	0.4688	0.4972	1001	10000

Finally, the `CODA` button creates two windows for each chain. The first one is the output window (or file) with all the generated values that are stored sequentially for each monitored parameter with the corresponding iteration number. Hence the values generated for the first node will be stored first followed by the second ones and so on. This window must be stored as a text file with extension `out` to enable the user to import it directly in `CODA`.

CODA for chain 1

1001	0.9602
1002	0.979
1003	0.9673
1004	0.9572
1005	0.9585
1006	0.9592
1007	0.949
1008	0.9309
1009	0.9007
1010	0.8936

The second window refers to an index file with details related to the position of each node stored in the `out` file. Thus, the name of each node and the number of the starting and ending lines of the corresponding output are given in this window which must be stored as a text file with extension `ind` for direct use with the `CODA` package.

Generally, transforming `CODA`-type data to usual `R` or `Splus` data frames is not difficult for a user of medium experience. Nevertheless, functions and routines are available on the Web for converting `CODA`-type files in `R/Splus` data frames directly.

CODA index

R[2]	1	2000
R[3]	2001	4000
R[4]	4001	6000
R[5]	6001	8000
R[6]	8001	10000
R[7]	10001	12000
R[8]	12001	14000
pi[1]	14001	16000
pi[2]	16001	18000
pi[3]	18001	20000
pi[4]	20001	22000
pi[5]	22001	24000
pi[6]	24001	26000
pi[7]	26001	28000
pi[8]	28001	30000

4.1.4.2 *Calculation of posterior summaries.*

The main tool for the calculation of the posterior summaries in WinBUGS is the `stats` option, which provides estimates of the posterior mean, standard deviation, and quantiles (including the median) for the given generated sample. The total number of iterations (generated sample size) and the number of iterations that the generated sample started (hence the burnin period) are also provided.

Node statistics

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
R[2]	0.994	0.03967	0.003786	0.923	0.9918	1.074	1001	2000
R[3]	1.011	0.0359	0.002278	0.9383	1.01	1.079	1001	2000
R[4]	0.9615	0.03426	0.001879	0.8925	0.9618	1.026	1001	2000
R[5]	0.9725	0.03851	0.002836	0.9014	0.9723	1.047	1001	2000
R[6]	0.9908	0.04484	0.003078	0.9026	0.9902	1.079	1001	2000
R[7]	1.038	0.04279	0.002523	0.9617	1.036	1.126	1001	2000
R[8]	1.051	0.04557	0.001521	0.9676	1.049	1.147	1001	2000
pi[1]	0.4674	0.01348	0.001164	0.4401	0.4679	0.4931	1001	2000

From these results we can infer that Kobe Bryant's expected success field rate was equal to 46.7% for season 1999–2000. For the next two seasons (2000/01, 2001/02), his success rate was about the same (posterior means for $R_2 = 0.994$ and $R_3 = 1.011$). For seasons

2002/03 and 2003/04, his success rate was decreased by 4% and 3%, respectively (posterior expectations for R_4 and R_5 equal to 0.9615 and 0.9725). For the next season, the success rate was constant (posterior mean for $R_6 = 0.99$), and finally for the last two seasons we observe increase of the success rate by 4% and 5%, respectively (posterior means of R_7 and R_8 equal to 1.038 and 1.051).

We may also report the 95% posterior credible intervals for all R_i that quantify the relative performance of a current season in comparison to the previous one. All intervals lie around the reference value of one, which indicates a constant and stabilized performance of the player over the seasons studied.

The `density` option provides a graphical representation of the posterior density estimate for each node. For parameter π_1 in the Kobe Bryant's data, see Figure 4.1.

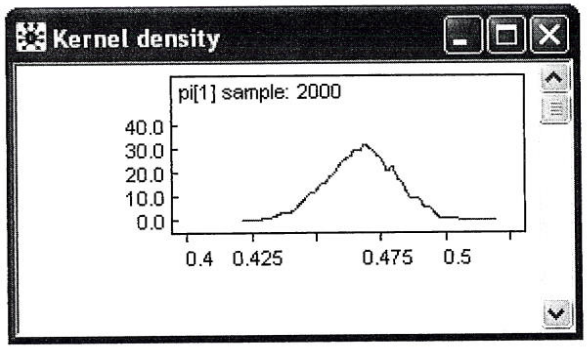


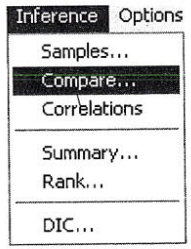
Figure 4.1 Density plot of Kobe Bryant's success rate of field goals for the first season (π_1).

Additional analysis can be performed using the remaining available options of the `inference` menu. These menus are described in detail in Section 4.2. Moreover, using the `CODA` option, the generated MCMC output can be exported in a text format and then imported in any statistical software for a more detailed output analysis.

4.2 FURTHER OUTPUT ANALYSIS USING THE INFERENCE MENU

In this section, details are provided concerning further output analysis within WinBUGS. We focus on the tools that are available in the `inference` menu while generating multiple chains, and the remaining operations available in WinBUGS are described in the sections that follow.

All procedures are described using the example of Kobe Bryant's field goal success rates presented in Section 4.1.



The `inference` menu offers a variety of tools for a more detailed output analysis. In summary, the following tools are available:

`Compare`: This tool is used to compare graphically the elements of a monitored vector node.

`Correlations`: This tool enables the user to obtain the correlation values and the scatterplots between two or more nodes.

`Summary`: This tool provides summary statistics for a node. The results are similar to those obtained with `stats` option of the `sample monitor` tool.

`Rank`: This tool provides useful analysis of the posterior distribution of ranks of the elements of a vector node. This tool is convenient when focus is on the evaluation of ranking of experimental units (organizations, teams, or subjects).

`DIC`: This tool offers the possibility of calculating the deviance information criterion (DIC) (Spiegelhalter et al., 2002) used to compare competitive models.

4.2.1 Comparison of nodes

In order to compare the elements of a vector node, we need to open the `comparison` tool following the path `Inference>Compare`.

We can plot the elements of a monitored vector by typing its name at the `node` text box and then pressing the `box plot` or the `caterpillar` buttons. The first will draw a boxplot for each node; the second one will draw horizontal lines or bars representing the 95% credible intervals of each node. For graphical representation of the procedure, see Figures 4.2 and 4.3.

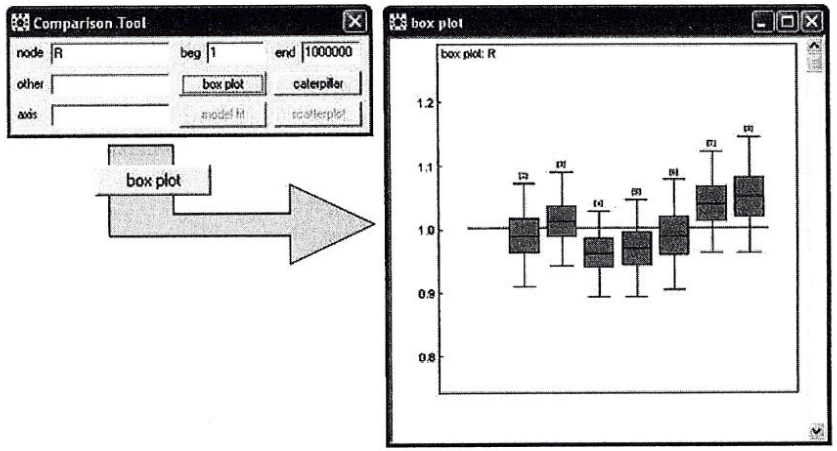


Figure 4.2 Comparison of posterior distributions of elements of vector $\boldsymbol{R}$ (relative performance in field goals) using boxplots.

Boxplots produced in WinBUGS are slightly different from the traditional one. The limits of each box represent the posterior quartiles, while the middle bar the posterior mean (by default, there is an option that allows the user to depict the posterior medians instead). The ending of the whisker lines represent the 2.5% and 97.5% posterior percentiles. The WinBUGS caterpillar plot essentially differs from the boxplot in two ways: (1) in the caterpillar plot, the box (referring to quartiles) is omitted, and (2) the bars are horizontal (parallel to the x axis). In both plots, the horizontal reference line represents the posterior mean estimated from all nodes depicted in the plot.

The `axis` box can be used to define which values will be used in the x axis. The node used in this box must be of the same length as the corresponding node that defined the `node` box. It can be either stochastic or constant (e.g., the values of an explanatory variable). In cases where the `axis` node is stochastic, the posterior means are used as values for the x axis.

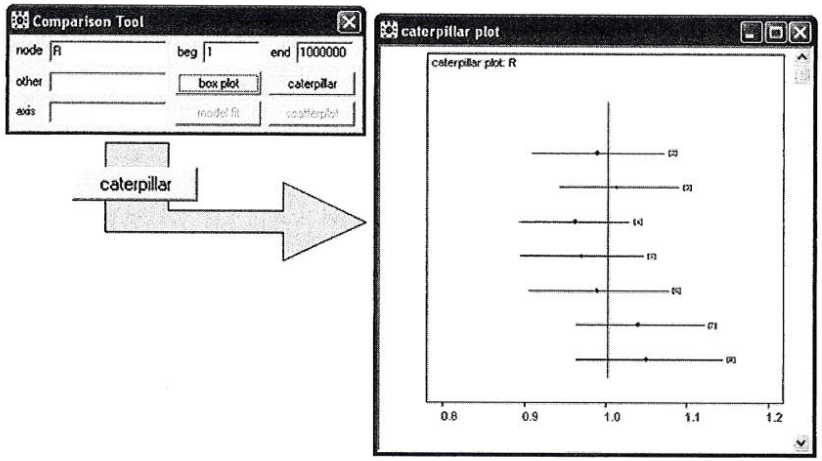
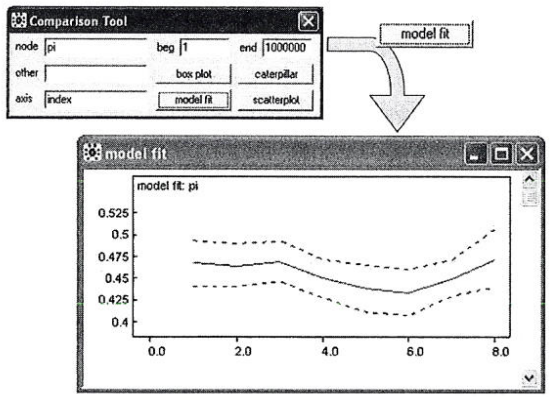


Figure 4.3 Comparison of posterior 95% credible intervals of elements of vector $\boldsymbol{R}$ (relative performance in field goals).

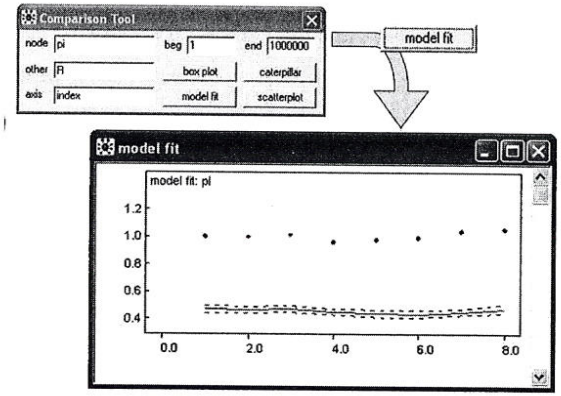
To illustrate the last two options, we have also defined a new deterministic node called `index` simply indicating the year sequence (taking values from one to eight) using the command

```
for (k in 1:YEARS){index[k] <-k }
```

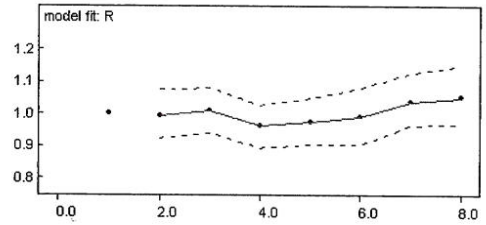
After compiling and running the new model, we activate the `comparison` tool. We can now plot the parameter `pi` (Kobe Bryant's success rate) versus the time `index` by typing the corresponding names at the text boxes `node` and `axis`. Boxplot and caterpillar plots will not be affected by putting an additional node in the `axis` text box, but now the buttons `model fit` and `scatterplot` will also be activated. The first one creates a plot of the 95% credible interval of `pi` (`node` argument) for each element with the corresponding x axis values taken by `index` (`axis` argument). The 2.5% and 97.5% percentiles and mean points for successive values of `index` will be also connected in a plot that is equivalent to using a piecewise linear model.



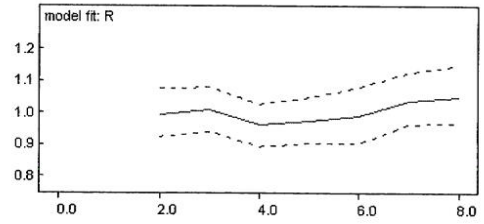
If we additionally type a node name in text box `others`, the values of this vector will be also printed on the plot. For example, using node `R` in the `others` text box will create the following plot



including the posterior means of node `R`. Note that the constant value of one for `R[1]` is also plotted. Generally the three arguments (`node`, `other`, `axis`) must be vectors of equal length. When stochastic nodes are used in `other` and `axis` then the posterior means are used. If we type `R`, `R`, and `index` in the text boxes `node`, `other`, and `axis`, respectively, we obtain the following plot:

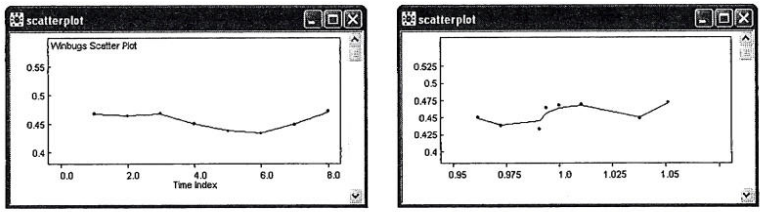


No posterior credible interval is plotted for `R[1]` since it refers to a constant node. If we eliminate node `R[]` by the `others` text box, then the corresponding points will not be plotted in the graph.



The `scatterplot` option provides a similar plot, with the values of `node` plotted against the values of `axis`. When stochastic nodes are used, then posterior means are used. Additionally, an exponentially weighted smoothed lined is fitted and plotted. Medians, bars, linear regression lines, and additional features can be plotted by changing the properties of the graph (click the right mouse button, then select `properties` and then `special`; for

more details, see Section 4.4); for examples of such plots, see Figure 4.4 . Finally, beg and end text boxes are used to specify the the beginning and the ending iterations used for the visual comparisons.

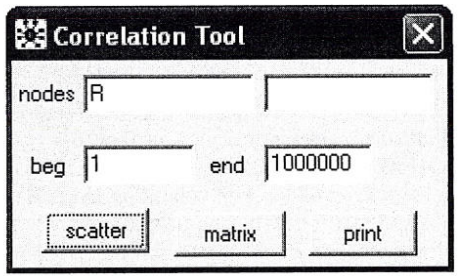


(a) (b)

Figure 4.4 Scatterplots of (a) success probabilities versus time index (`pi` vs. `index`) and (b) success probabilities versus relative performance (`pi` vs. `R`) for Example 1.4 (Kobe Bryant's data).

4.2.2 Calculation of correlations

Correlations between the elements of a monitored vector node or between two monitored nodes can be obtained using the `correlation` tool (follow the path `Inference> Correlations`).

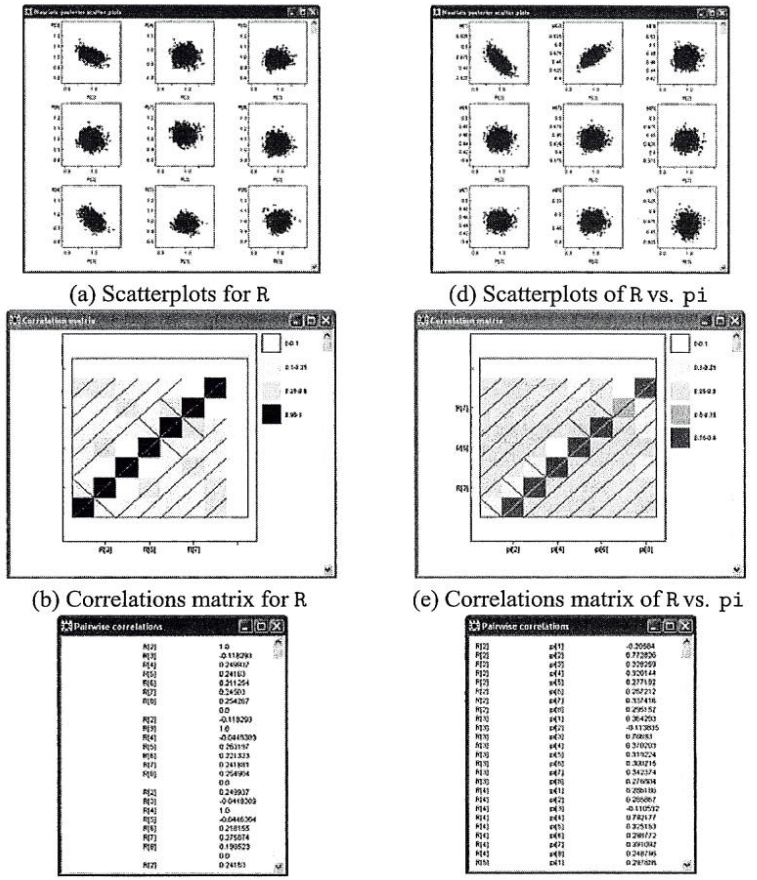


In the `Correlation` tool, the following fields and buttons and options are available:

- `nodes` *text boxes*: Here, the names of the nodes for which we wish to calculate posterior correlations are specified by the user. The second box can be left empty, provided that the node typed in the first one is a vector. In this case, correlations between all elements of the typed vector node are calculated. If two vector nodes are defined (one in each text box), then posterior correlations between the elements of the first and second nodes are calculated.
- `beg` *and* `end` *text boxes*: As usual, the range of the iterations that will be used for calculation of the posterior correlations can be defined here.
- `scatter` *button*: This produces a scatterplot matrix between the elements of the requested nodes.
- `matrix` *button*: Produces a visual representation of the correlation matrix using different shades of black instead of the actual values. This simplifies comparisons of the correlations in multidimensional problems.

- `print` *button*: This provides the numerical estimates of the posterior correlations between the elements of the requested nodes.

In Figure 4.5 we provide results of this tool for a single vector node (`R`; plots a–c) and for two vector nodes (`R` and `pi`; plots d and e).



(a) Scatterplots for R (d) Scatterplots of R vs. pi

(b) Correlations matrix for R (e) Correlations matrix of R vs. pi

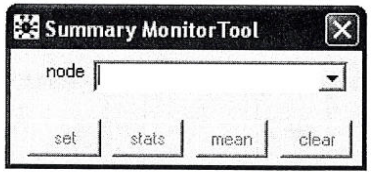
(c) Posterior correlations for R (f) Posterior correlations between R and pi

Figure 4.5 Results of the `correlation` tool using data of Example 1.4 (Kobe Bryant's data).

4.2.3 Using the `summary` tool

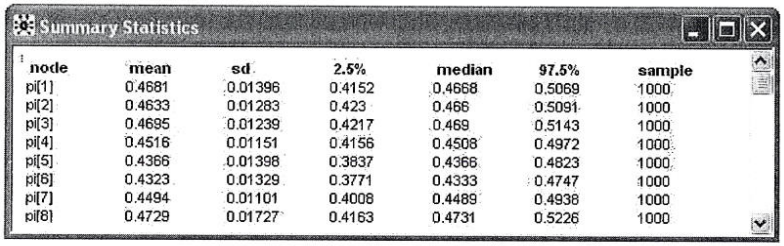
The `summary monitor` tool can be used independently from the `sample monitor` tool. We can open the corresponding dialog box by following the path `Inference>Summary`. Using this tool, summary statistics of the posterior distribution of a node can be obtained. It is equivalent to the `stats` option in `sample monitor` tool. The main difference from the

latter is that no values are stored here and hence less storage is required. This is convenient in high-dimensional problems where only some summary statistics may be needed for specific nodes.



When we open the `summary monitor` tool, all four buttons are inactive. In order to activate them, we must define again which nodes we wish to monitor via this tool and then generate some additional iterations. Implementation of this procedure in Example 1.4 for `pi` can be summarized by the following steps:

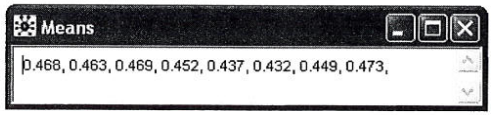
- Type `pi` in the `node` text box and then press the `set` button.
- Return to the `update` tool and generate additional values (e.g., 1000).
- Retype the node's name (`pi`) and press the `stats` button. Then the following results will appear:



node	mean	sd	2.5%	median	97.5%	sample
pi[1]	0.4681	0.01396	0.4152	0.4668	0.5069	1000
pi[2]	0.4633	0.01283	0.423	0.466	0.5091	1000
pi[3]	0.4695	0.01239	0.4217	0.469	0.5143	1000
pi[4]	0.4516	0.01151	0.4156	0.4508	0.4972	1000
pi[5]	0.4366	0.01398	0.3837	0.4366	0.4823	1000
pi[6]	0.4323	0.01329	0.3771	0.4333	0.4747	1000
pi[7]	0.4494	0.01101	0.4008	0.4489	0.4938	1000
pi[8]	0.4729	0.01727	0.4163	0.4731	0.5226	1000

  As it is evident, the resulting output is the same as the corresponding one given by the `stats` option in the `sample monitor` tool.

- For `pi` node, press the `mean` button. Then only the posterior means of this vector node will appear on a separate window.



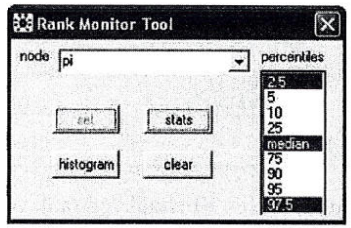
If the second step is skipped (i.e., the generation of additional iterations after `set`), then no action will be performed when pressing either the `stats` or the `means` button.

4.2.4 Evaluation and ranking of individuals

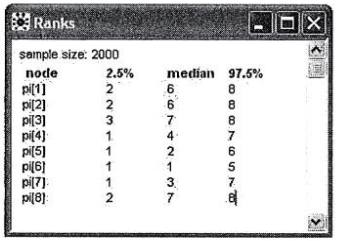
In many problems there is interest in the evaluation and ranking of individual or units under study. Such problems typically arise in the evaluation of healthcare units (hospitals or medical centers), educational institutes (schools, colleges, universities, postgraduate programs), students, or athletic teams

WinBUGS provides the facility to monitor some rank-related statistics on the basis of specific stochastic vector nodes that represent estimated or predicted performance. This can be done automatically using the `rank monitor` tool, which can be invoked following the path `Inference>Rank`. Using this tool, in each iteration of the MCMC algorithm the values of the vector node are rearranged in ascending order and the corresponding ranks are calculated and stored. Therefore, ranks indicate the sequence of the stochastic vector elements in ascending order; for example, rank 1 may indicate that this element had the lowest value. This is also an independent tool, where we need to specify separately which nodes will be monitored in terms of rank. The procedure (for Example 1.4) for evaluating Kobe Bryant's performance using the ranks of node `pi` can be summarized by the following steps.

- Type `pi` in the `node` text box and then press the `set` button (in `rank monitor` tool).



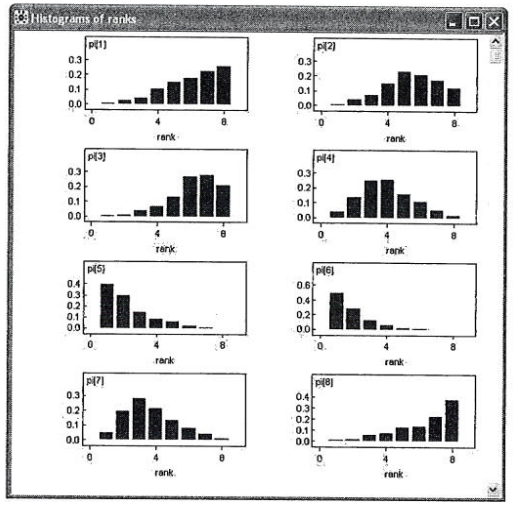
- Return to the `update` tool and generate additional values (e.g., 1000).
- Retype the node's name (`pi`) and press the `stats` button. Then the following results will appear:



node	2.5%	median	97.5%
pi[1]	2	6	8
pi[2]	2	6	8
pi[3]	3	7	8
pi[4]	1	4	7
pi[5]	1	2	6
pi[6]	1	1	5
pi[7]	1	3	7
pi[8]	2	7	8

  As you can see, the median of the rank for each p_i is given within the 2.5–97.5% percentiles. By this window we can infer which seasons were the best in terms of the player's performance. For example, seasons 3 and 8 have the higher median rank (equal to 7). We can further infer that season 3 was the best since the 2.5% percentile is higher than the corresponding one for season 8 (3 vs. 2). In this way we can obtain an informal ordering of the player's performance for each season.

- The button `histogram` will provide a visual representation of the rank distribution for every node. Thus, using the `histogram` button, we obtain the following plots:



- Finally, the `clear` button deletes all simulated rank values for the selected node. If we wish to rerun some new rank values for the same node, then we have to repeat the procedure outlined above.

Note that more detailed analysis of ranks can be obtained if we include corresponding code within the model definition; for details see the surgical example in Spiegelhalter et al. (1996*b*, p. 18). In terms of storage, the rank monitor tool is more efficient since it reserves lower storage space than does the corresponding one, which calculates ranks analytically within the model's code.

4.2.5 Calculation of deviance information criterion

The deviance information criterion (DIC) was introduced by Spiegelhalter et al. (2002) as a measure of model comparison and adequacy. It is given by the expression

$$\text{DIC}(m) = 2\overline{D(\boldsymbol{\theta}_m, m)} - D(\overline{\boldsymbol{\theta}}_m, m) = D(\overline{\boldsymbol{\theta}}_m, m) + 2p_m,$$

where $D(\boldsymbol{\theta}_m, m)$ is the usual deviance measure, which is equal to minus twice the log-likelihood

$$D(\boldsymbol{\theta}_m, m) = -2 \log f(\boldsymbol{y}|\boldsymbol{\theta}_m, m)$$

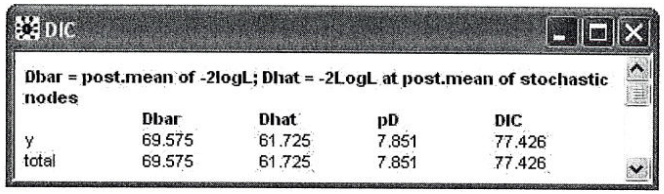
and $\overline{D(\boldsymbol{\theta}_m, m)}$ is its posterior mean, p_m can be interpreted as the number of "effective" parameters for model m given by

$$p_m = \overline{D(\boldsymbol{\theta}_m, m)} - D(\overline{\boldsymbol{\theta}}_m, m),$$

and $\overline{\boldsymbol{\theta}}_m$ is the posterior mean of the parameters involved in model m. Smaller DIC values indicate a better-fitting model. DIC must be used with caution since it assumes that the

are not symmetric or unimodal. Details concerning DIC limitations and possible problem can be found in Spiegelhalter et al. (2003*d*, section entitled "Tricks: Advanced Use the BUGS Language"). A detailed illustration of DIC can be found in Section 6.4.3; f additional details on the use of DIC and other methods for model comparison, see Chapt 11.

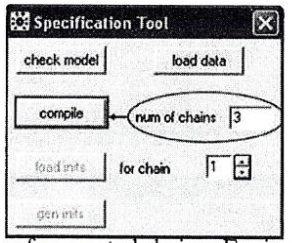
The `DIC` tool can be opened following the path `Inference>DIC`. The procedure f calculating DIC is similar to those for `summary` and `ranks` monitor tools. We first pre the `set` button to start storing DIC values, then update the sampler and finally obtain t results by pressing the `DIC` button.



4.3 MULTIPLE CHAINS

4.3.1 Generation of multiple chains

The procedure described in Section 4.1.3 is followed when we wish to generate a sing chain. If we wish to generate additional samples, then we follow steps 1–6 as in Secti 4.1.3. The remaining of the steps are slightly modified according to the following:



7. Set the number of chains you wish to generate in the text box next to the `compile` button and then click on `compile`. For this example, three chains were used.

8. Load initial values by highlighting the word `list` and then pressing the button `load inits` in the model specification tool. Repeat the procedure as many times as the number of generated chains. For instan for three chains we need to set three different sets of initial values. The `genera` button might be used for some chains but is recommended for specifying differ starting values: with appropriate dispersion in order to ensure convergence. For Kc Bryant's data, we have used the following three sets of initial values.

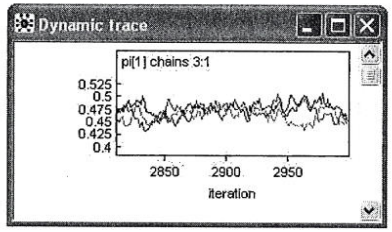
```
list(pi=c(0.5,NA,NA,NA,NA,NA,NA,NA), R=c(NA,1,1,1,1,1,1,1) )
list(pi=c(0.1,NA,NA,NA,NA,NA,NA,NA), R=c(NA,1,1,1,1,1,1,1) )
list(pi=c(0.9,NA,NA,NA,NA,NA,NA,NA), R=c(NA,1,1,1,1,1,1,1) )
```

9–12. If the model is initialized, then update all chains by the given number of iteratic (the same as in Section 4.1.3). Hence 1000 iterations means that each chain will updated by 1000 iterations.

13. Set the parameters to be monitored. By default, observations from all chains

present on the upper right of the `sample monitor` tool. These text boxes can be used to extract posterior summaries and output analysis only for selected chains.

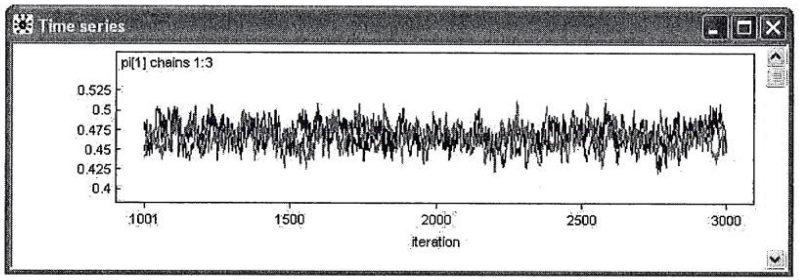
14. Open the online trace plots. Now one line for each chain is outlined on the dynamic trace plot.
15. Update the sampler by 2000 iterations.
16. The `sample monitor` tool can now be used to obtain initial output analysis and export data to other statistical programs.



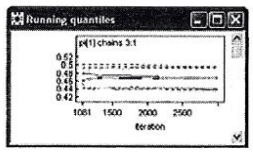
4.3.2 Output analysis

In the output analysis we observe the following differences:

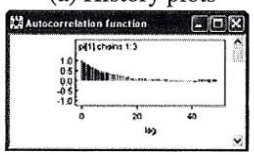
- In all plots (autocorrelations, trace, history, quantiles), results for all chains are visually separated in the same graph. For example, in the `history` plot one line for each chain is printed; see Figure 4.6 for the trace plot resulting from the model of Kobe Bryant's data. Only in the `density` plot, a single estimate of the posterior distribution is calculated (and plotted) using values from all chains.



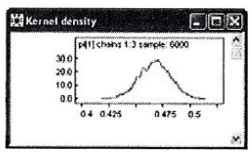
(a) History plots



(b) Quantiles plot



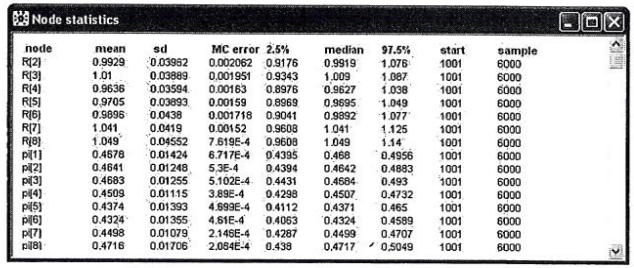
(c) Autocorrelations plot



(d) Density plot

Figure 4.6 WinBUGS plots when generating three chains in model for Kobe Bryant's data.

- In the `stats` option, the posterior summaries using values from all requested chains are calculated. For an update of 2000 iterations using three chains, we obtain the following results.



node	mean	sd	MC error	2.5%	median	97.5%	start	sample
R[2]	0.9929	0.03982	0.002062	0.9176	0.9919	1.076	1001	6000
R[3]	1.01	0.03989	0.001951	0.9343	1.009	1.087	1001	6000
R[4]	0.9636	0.03594	0.00163	0.8976	0.9627	1.038	1001	6000
R[5]	0.9705	0.03893	0.00159	0.8969	0.9695	1.049	1001	6000
R[6]	0.9896	0.0438	0.001718	0.9041	0.9892	1.077	1001	6000
R[7]	1.041	0.0419	0.00152	0.9608	1.041	1.125	1001	6000
R[8]	1.049	0.04552	7.619E-4	0.9608	1.049	1.14	1001	6000
pi[1]	0.4678	0.01424	6.717E-4	0.4395	0.468	0.4956	1001	6000
pi[2]	0.4641	0.01248	5.3E-4	0.4394	0.4642	0.4883	1001	6000
pi[3]	0.4683	0.01255	5.102E-4	0.4431	0.4684	0.493	1001	6000
pi[4]	0.4509	0.01115	3.89E-4	0.4298	0.4507	0.4732	1001	6000
pi[5]	0.4374	0.01393	4.699E-4	0.4112	0.4371	0.465	1001	6000
pi[6]	0.4324	0.01355	4.61E-4	0.4063	0.4324	0.4589	1001	6000
pi[7]	0.4498	0.01079	2.146E-4	0.4287	0.4499	0.4707	1001	6000
pi[8]	0.4716	0.01706	2.084E-4	0.438	0.4717	0.5049	1001	6000

In this output, 6000 iterations in total were used to calculate the posterior summaries (2000 from each of three generated samples).

- In the `coda` option, one output window/file with the generated values of each chain is obtained. The index window/file is common for all chains; see Figure 4.7

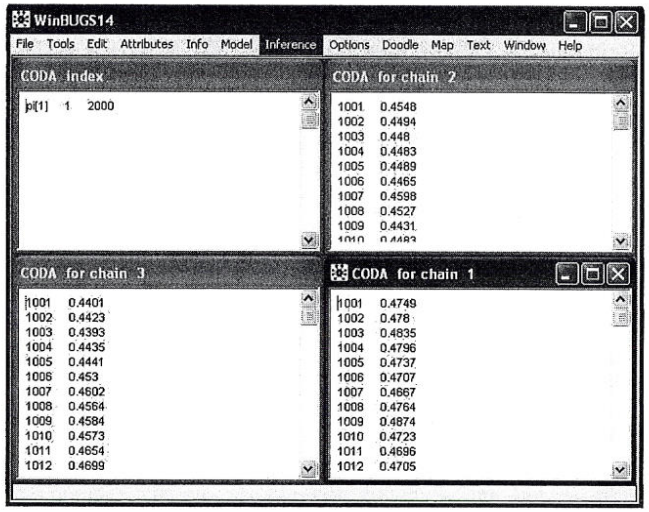


Figure 4.7 CODA output when generating three chains in model for Kobe Braynt's data.

- The Gelman–Rubin diagnostic (Gelman and Rubin, 1992) is now available via the `bgr diag` option (see Section 4.3.3 for details).

4.3.3 The Gelman–Rubin convergence diagnostic

The Gelman–Rubin diagnostic (Gelman and Rubin, 1992) of is available in WinBUGS via the `bgr diag` option, when multiple chains are generated in parallel, each one starting from different initial values. Then an ANOVA-type diagnostic test is implemented by calculating and comparing the between-sample and the within-sample variability (i.e., intersample and intrasample variability). The statistic R can be estimated by

$$\hat{R} = \frac{\hat{V}}{\text{WSS}} = \frac{T'-1}{T'} + \frac{\text{BSS}/T'}{\text{WSS}} \frac{\kappa+1}{\kappa}$$

where κ is the number of generated samples/chains, T' is the number of iterations kept in each sample/chain, BSS/T' is the variance of the posterior mean values over all generated samples/chains (between-sample variance), WSS is the mean of the variances within each sample (within-sample variability), and

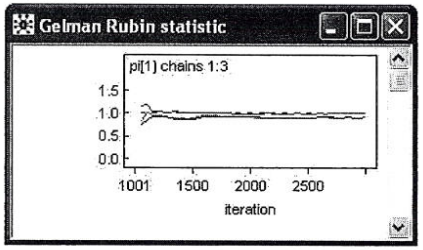
$$\hat{V} = \frac{T'-1}{T'}\text{WSS} + \frac{\text{BSS}}{T'}\frac{\kappa+1}{\kappa}$$

is the pooled posterior variance estimate. When convergence is achieved and the size of the generated data is large, then $\hat{R} \to 1$. Brooks and Gelman (1998) adopted a corrected version of this statistic given by

$$\hat{R}_c = \frac{d+3}{d+1}\hat{R}$$

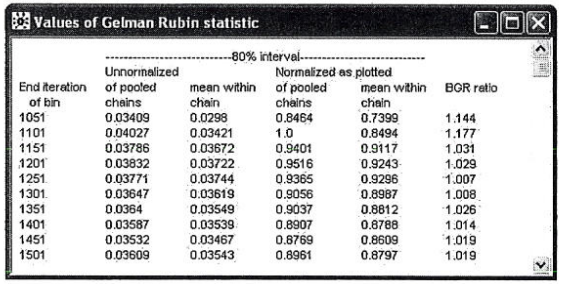
where d is the estimated degrees of freedom for the pooled posterior variance estimate $\hat{V}$; for more details see Brooks and Gelman (1998, sec. 1.3).

Using a line plot, the `bgr diag` option available in the `sample monitor` tool plots the evolution of the pooled posterior variance $\hat{V}$ (in green color), average within-sample variance WSS (in blue), and their ratio $\hat{R}$. The dashed line denotes the reference value of one. Note that in WinBUGS the estimates are based on the ranges of the 80% posterior credible intervals. These quantities are calculated every 50 iterations.



We generally expect $\hat{R}$ to be higher than one at the initial stage of the algorithm, provided that the starting points are suitably scattered over the parameter space. As the number of iterations increase, $\hat{R}$ tends to one and $\hat{V}$ and WSS will stabilize, indicating the convergence of the algorithm.

Numerical estimates can be obtained by double clicking the left mouse button on the plot (opening the figure) and then pressing the left mouse and the keyboard Control buttons to open the window with the associated information.



	80% interval				
	Unnormalized		Normalized as plotted		
End iteration of bin	of pooled chains	mean within chain	of pooled chains	mean within chain	BGR ratio
1051	0.03409	0.0298	0.8464	0.7399	1.144
1101	0.04027	0.03421	1.0	0.8494	1.177
1151	0.03786	0.03672	0.9401	0.9117	1.031
1201	0.03832	0.03722	0.9516	0.9243	1.029
1251	0.03771	0.03744	0.9365	0.9296	1.007
1301	0.03647	0.03619	0.9056	0.8987	1.008
1351	0.0384	0.03549	0.9037	0.8812	1.026
1401	0.03587	0.03539	0.8907	0.8788	1.014
1451	0.03532	0.03467	0.8769	0.8609	1.019
1501	0.03609	0.03543	0.8961	0.8797	1.019

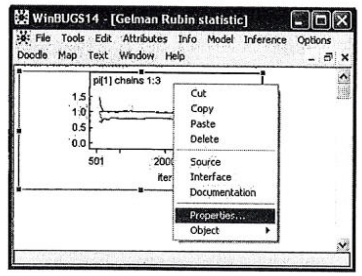
Normalized values are used so that all lines can be plotted in the same figure. These normalized values are calculated by dividing the original $\hat{V}$ and WSS by their maximum value (i.e., $\max\{\hat{V}, \text{WSS}\}$).

4.4 CHANGING THE PROPERTIES OF A FIGURE

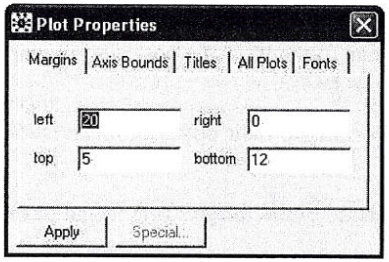
4.4.1 General graphical options

Options of each graph can be changed by following the steps below:

1. Select the graph (left mouse click).
2. Right mouse click to open the drop-down menu.
3. Select properties in the menu.



After this procedure, the `plot Properties` menu opens with five different tabs: `margins`, `axis bounds`, `titles`, `all plots`, and `fonts`.



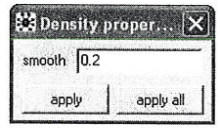
The `margins`, `axis bounds`, and `titles` tabs are straightforward to use. The `fonts` tab can be used to change the font family and size in titles and in the axes. Finally, by pressing the `all plots` tab, the user can apply the settings of the current plot (all of them or specific ones) to all graphs in the WinBUGS working window.

Two buttons are also available in the `plot properties` menu: `apply` and `special`. The first one executes a requested change on the plot properties while the latter generates special figure properties that vary according to the type of the plot. In simple plots, this option is inactive since no special options are available. Special options of specific plots are illustrated below.

4.4.2 Special graphical options

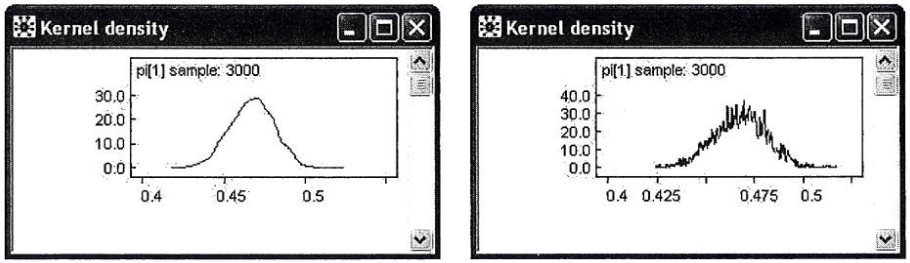
Special graphical options are available for the density plot of the `inference` tool, and for boxplots, caterpillar, model fit, and scatterplot in the `compare` tool.

For the density plot, the only available special option is the smoothing parameter of the plot (default value = 0.2). Increasing the value of this parameter decreases the smoothing producing a higher number of spikes on the plot, while decreasing this value increases the smoothing; see example in Figure 4.8



The available special properties in boxplots allow for several modifications in the graph. From this menu, we can

- Eliminate the reference line or change its value (the default is the mean over all components).



(a) Smoothing parameter = 0.1 (b) Smoothing parameter = 0.4

Figure 4.8 Density plots using different smoothing parameters.

- Eliminate the boxplot labels.
- Depict the posterior mean (default) or the posterior median using the middle line of the boxplot.
- Order the boxplots according to their posterior mean/median from the lowest (left) to the highest (right).
- Plot the boxplots parallel to either the y axis (vertical plot) or the x axis (horizontal plot).
- Use the logarithmic scale.
- Change the color of the boxes.

See Figure 4.9, for a graphical description of the menu.

Special properties in caterpillar plots are similar to the boxplot special properties.

The only available `model fit` special options are related to the use of logarithmic scale on x and/or y axis.

Finally the scatterplot special properties allow us to

- Plot the posterior means or medians as points on the graph.
- Change the color of the points.
- Draw or eliminate the fitted line.
- Select the color of the fitted line.
- Select the type of fitted line (linear or smoothed) and their corresponding parameter values.
- Use logarithmic scale on x and/or y axis.
- Plot credible intervals (for means of the variable plotted on y axis) as bars around the points.

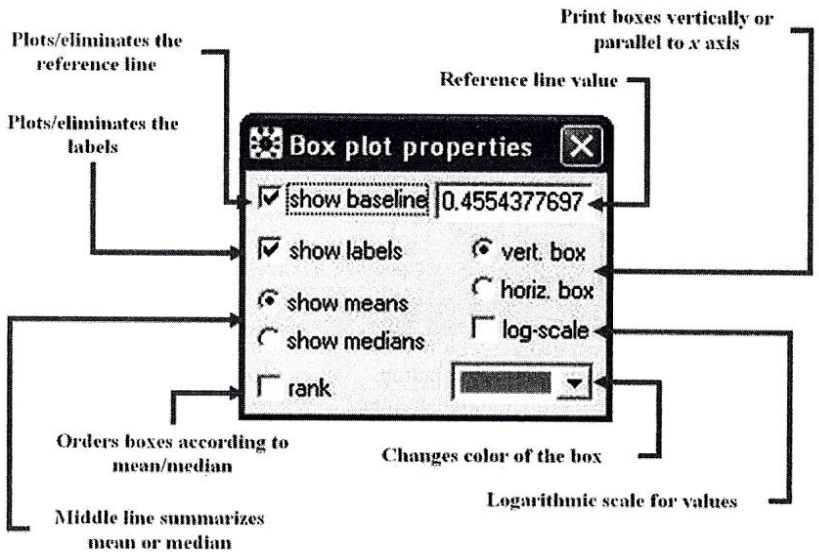


Figure 4.9 Special properties of WinBUGS boxplots.

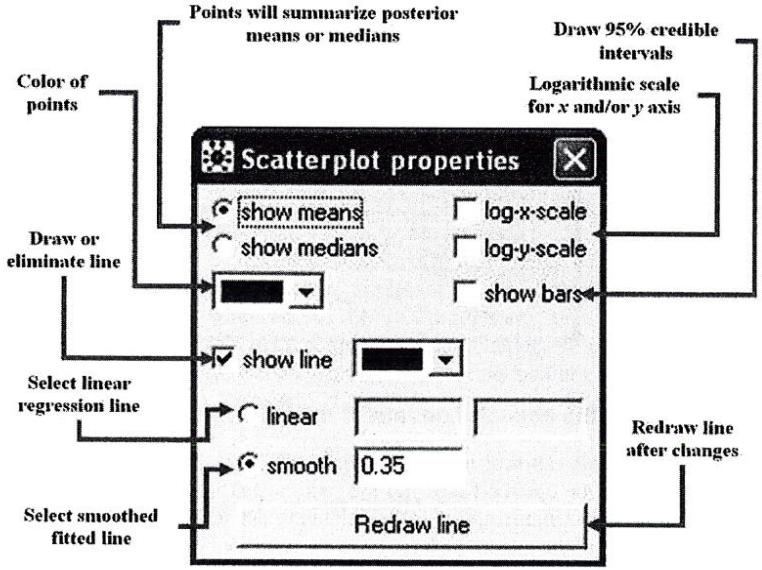
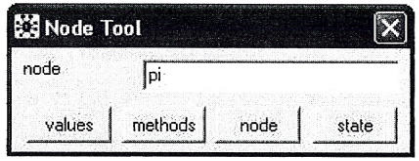


Figure 4.10 Special properties of WinBUGS scatterplots.

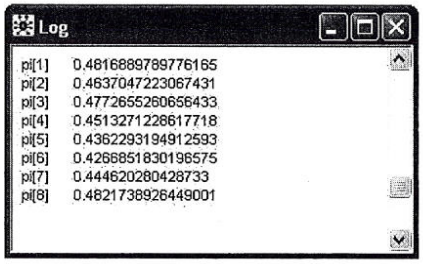
4.5 OTHER TOOLS AND MENUS

4.5.1 The `node info` tool

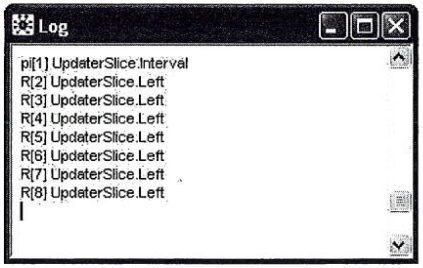
Information concerning the current state of a node can be extracted using the `node info` tool (follow the path `Info>Node info`).



Using this tool, the current `values` of a node and the update `method` used for stochastic nodes is provided by the corresponding buttons. For example, for Kobe Bryant's model, the current state for node `pi` (after 3000 iterations) as follows:



while the slice Gibbs method is used to update π_1 and $R_2, \ldots, R_8$ nodes according to the information we obtain using the `method` option.

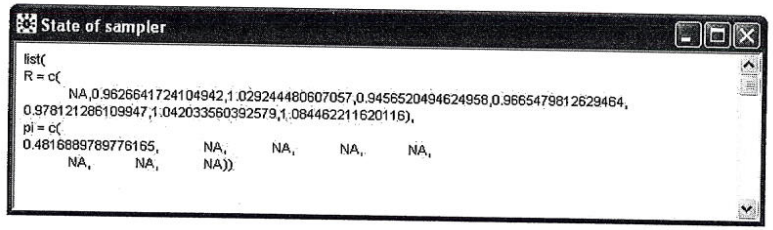


4.5.2 Monitoring the acceptance rate of the Metropolis–Hastings algorithm

The `monitor Metropolis` tool is activated only when the Metropolis method is used by WinBUGS. It can be opened following the path `Model>Monitor Met`. It calculates statistics (minimum, maximum, and average) related to the acceptance rate of the random-walk Metropolis during the adaptive phase used by WinBUGS to tune the variance of the symmetric normal proposal. The adaptive phase is set by default equal to 4000 iterations, and all acceptance rates are calculated for batches of 100 iterations. The aim of the adaptive phase is to achieve an acceptance rate between 20% and 40%; for details, see Spiegelhalter et al. (2003*d*, pp. 6, 25).

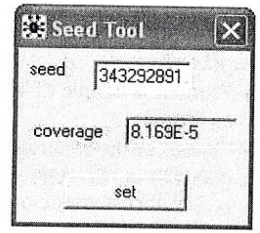
4.5.3 Saving the current state of the chain

Frequently, we get an initial sample from the MCMC algorithm and check whether it has converged. If there is an indication of nonconvergence, then we run the algorithm for an additional number of iterations. This is equivalent to starting the algorithm using as initial values the ones from the last iteration of the previous run. On some occasions, it is convenient to save the current state of the chain in case we wish to generate additional iterations in the future. This can be achieved in WinBUGS by the `Model>Save State` path. The output is given in a list format and hence can be used directly as initial values for the next run of the model. For the example of Koby Bryant's data, we get the following output:



4.5.4 Setting the starting seed number

All simulation algorithms that generate a sequence of pseudorandom numbers are called *random generators*. They require a starting number that is used to generate a sequence of pseudorandom numbers. Every starting seed generates a single sequence of random numbers. This property is useful if we wish to reproduce exactly the same results multiple times. This starting seed can be set in WinBUGS following the path `Model>Seed`.



4.5.5 Running the model as a script

As we have already seen, in order to run a model, we need to repeat a number of actions (check model, define data, compile model, etc.). This procedure is laborious and time-consuming, especially when various different models are fitted. For this reason, WinBUGS provides the facility to write a script code that generates all the actions required to simulate and monitor posterior values. This is equivalent to the `backbugs` command in the classic version of BUGS for DOS and UNIX. More details on this issue are provided in Appendix B.

4.6 SUMMARY AND CONCLUDING REMARKS

In this chapter, we have illustrated both basic and more specialized functions of WinBUGS using a simple example based on Kobe Bryant's success rate in field goals. After this chapter, the reader must be able to construct simple models in WinBUGS, produce more specialized analysis (including boxplots of the posterior densities, analysis of ranks,

correlations, and calculation of DIC) and generate multiple chains and must be able to change the technical details and options of WinBUGS plots. Finally, brief details of the remaining tools and menus are also provided.

Details about the construction of models using directed acyclic graphs and illustration of running WinBUGS models using scripts (instead of menus) are briefly discussed in this chapter. Further details can be found in Appendices A and B, respectively.

In the chapter that follows, the normal linear regression model is introduced and illustrated in detail.

Problems

4.1 Following the guidelines presented in Section 4.1, perform the same analysis in WinBUGS for the following survival times

```
51 3 17 13 5 4 17 1 5 3 8 22 0 1 13 8 15 3 1 13
```

assuming

a) Weibull distribution

b) Gamma distribution

c) Log-normal distribution

4.2 Compare the models of Problem 4.1 using the DIC. Which model is preferable for those data?

4.3 For each model of Problem 4.1, run five chains simultaneously and calculate the Gelman–Rubin convergence diagnostic.

4.4 From WinBUGS examples, volume 1 (Spiegelhalter et al., 2003*a*), run the example `Surgical`.

a) Produce a sample of 2000 iterations after discarding an additional 500 iterations as burnin.

b) Check for the convergence of the chain.

c) Analyze the MCMC output and make inferences concerning the model parameters.

d) Produce an a posteriori analysis of the hospital ranking (using the `rank` tool).

e) Compare the posterior distributions of the mortality rates of all hospitals, using the `compare` tool.

4.5 For the data and implemented models of Problem 3.5

a) Use the `compare` tool to produce scatterplots of the sampled values and estimate the posterior correlations between the parameters of interest.

b) Run the MCMC chain for 5000 iterations plus 1000 burnin iterations. Save the current state of the algorithm and use these values to rerun the sampler using these values as initial values (no burnin is needed in the new sample).

c) Follow the instructions in the WinBUGS manual (Spiegelhalter et al., 2003*d*) and in Appendix B and rerun your MCMC in the script/background mode.

CHAPTER 5

INTRODUCTION TO BAYESIAN MODELS: NORMAL MODELS

5.1 GENERAL MODELING PRINCIPLES

Statistical models are nowadays used to describe parsimoniously real life problems observed under uncertainty. A *statistical model* is a collection of probabilistic statements (and equations) that describe and interpret present behavior or predict future performance. It consists of three important components: the response variable (or variables) Y, the explanatory variables $X_1, X_2, \ldots, X_p$, and a linking mechanism between the two sets of variables.

The response variables Y are the main study variables, and they represent the stochastic part of the model. By the term `stochastic` we refer to random variables whose outcome is uncertain before it is observed. Concerning these variables, we are frequently interested in describing the mechanism underlying or leading to the appearance of a certain outcome of Y and predict a future outcome of Y. Since the response variable is the stochastic component of the model, we can write

$$Y|X_1, X_2, \ldots, X_p \sim \mathcal{D}(\boldsymbol{\theta})$$

where $\mathcal{D}(\boldsymbol{\theta})$ is a distribution with parameter vector $\boldsymbol{\theta}$. For example, for normal regression models, the response (stochastic component of the model) is written as

$$Y|X_1, X_2, \ldots, X_p \sim N(\mu, \sigma^2),$$

where $N(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2. Models with one response variable are called *univariate*, while models with more than one response variables are called *multivariate*. In this book we focus on univariate models.