1 Introduction

1.1 What is BUGS?

BUGS is a program that carries out Bayesian inference on statistical problems using a simulation technique known as Gibbs sampling.

BUGS assumes a Bayesian or full probability model, in which all quantities are treated as random variables. The *model* consists of a defined joint distribution over all unobserved (parameters and missing data) and observed quantities (the data); we then need to condition on the data in order to obtain a posterior distribution over the parameters and unobserved data. Marginalising over this posterior distribution in order to obtain inferences on the main quantities of interest is carried out using a Monte Carlo approach to numerical integration (Gibbs sampling).

There is a small set of BUGS *commands* to control a session in which a (possibly very complex) statistical model expressed using the BUGS *language* is analysed. A *compiler* processes the model and available data into an internal data structure suitable for efficient computation, and a *sampler* operates on this structure to generate appropriate values of the unknown quantities. A suite of *S-plus* (Statistical Science, Inc) functions, CODA (Best *et al.*, 1995), is provided for analysis and plotting of the output files, and diagnosing convergence.

1.2 For what kind of problems is BUGS best suited?

BUGS is intended for complex models in which there may be many unknown quantities but for which substantial conditional independence assumptions are appropriate. Particular structures include generalised linear models with hierarchical or crossed random effects, latent variable or frailty models, measurement errors in responses and covariates, informative censoring, constrained estimation, and missing data. See Gilks *et al.* (1993) for examples of typical applications. BUGS is a general purpose program and so it is inevitable that many types of models, such as spatial smoothing, could be more efficiently implemented in special purpose software. There are currently restrictions on the classes of model that can be fitted using this version of BUGS, and these will be made clear in the manual (see Section 3.2).

1.3 Markov Chain Monte Carlo (MCMC) techniques

BUGS is intended for problems for which there is no exact analytic solution, and for which standard approximation techniques have difficulties. *Markov Chain Monte Carlo* (MCMC) is being increasingly used as an approach for dealing with such problems. The basic philosophy behind MCMC is to take a Bayesian approach and carry out the necessary numerical integrations using simulation: see Gelfand and Smith (1990); Smith and Roberts (1993) for background. Instead of calculating exact or approximate estimates, this computer-intensive technique generates a stream of simulated values for each quantity of interest. To perform MCMC additional tools are required to form samples from the relevant distributions, monitor the stream for convergence, and summarise the accumulated samples. See Gilks *et al.* (1995) for a wide range of articles on all practical aspects of using MCMC.

1.4 A simple example

We introduce a trivial problem for which exact solutions are possible in order to illustrate the nature of the Gibbs approach. This example will then be analysed in stages in order both to check the installation of the software and to illustrate the use of BUGS.

Consider a set of 5 observed (x, y) pairs $(1,1), (2,3), (3,3), (4,3), (5,5)$. We shall fit a simple linear regression of y on x, using the notation

$$Y_i \sim \text{Normal}(\mu_i, \tau) \quad (1)$$

$$\mu_i = \alpha + \beta(x_i - \overline{x}) \quad (2)$$

Note that we have separated out the linear function (2) expressing the dependence on x_i of the expectation of Y_i , from the stochastic link (1) between μ_i and Y_i. This is not strictly necessary but enhances clarity and follows the tradition of generalized linear modelling. The parameterisation of the normal distribution is also slightly non-standard, in that $\tau = 1/\sigma^2 = 1/\text{variance}(Y) =$ the *precision* of Y.

Classical unbiased estimates are $\hat{\alpha} = \overline{y} = 3.00, \hat{\beta} = \sum_i y_i(x_i - \overline{x})/\sum_i (x_i - \overline{x})^2 = .80, \hat{\sigma^2} = \sum(y_i - \hat{y}_i)^2/(n-2) = .533$, with $\hat{\text{Var}}(\hat{\alpha}) = \hat{\sigma}^2/n = .107, \hat{\text{Var}}(\hat{\beta}) = \hat{\sigma}^2/\sum_i (x_i - \overline{x})^2 = .053$. Both frequentist and Bayesian 'noninformative' priors lead to inference being based on the pivotal quantities $(\alpha - \hat{\alpha})/\sqrt{\hat{\text{Var}}(\hat{\alpha})}$ and $(\beta - \hat{\beta})/\sqrt{\hat{\text{Var}}(\hat{\beta})}$ both having t_3 distributions with mean 0 and variance 3, and $\hat{\sigma^2}(n-2)/\sigma^2$ having a χ^2_3 distribution, leading to 95% confidence/credible intervals given below.

95% interval for α : $\quad \hat{\alpha} \pm 3.18\sqrt{\hat{\text{Var}}(\hat{\alpha})} \quad = (1.96,\ 4.04\ )$

95% interval for β : $\quad \hat{\beta} \pm 3.18\sqrt{\hat{\text{Var}}(\hat{\beta})} \quad = (.07,\ 1.53)$

95% interval for τ : $\quad (.22, 9.35) \div (3\hat{\sigma}^2) \quad = (.14,\ 5.85)$

The BUGS language allows a concise expression for the model which is contained in the file `line.bug`, with the core relations (1) and (2) described as follows.

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

Simple commands for running BUGS are in the file `line.cmd` and are reproduced below.

```
compile("line.bug")
update(500)
monitor(alpha)
monitor(beta)
monitor(sigma)
monitor(tau)
update(1000)
stats(alpha)
stats(beta)
stats(sigma)
stats(tau)
q()
```

This compiles the model, generates an initial run of 500 iterations as a "burn-in" in order (with luck) to reach convergence, starts monitoring samples of parameters of interest, and then performs 1000 iterations that result in a file containing a series of values simulated from the joint posterior of the unknown quantities. Summary statistics are available directly, or a graphics program may be used to display the whole sample. For example, the CODA suite of functions for *S-Plus* provided with BUGS (see section 7) will give the output shown in Figure 1.

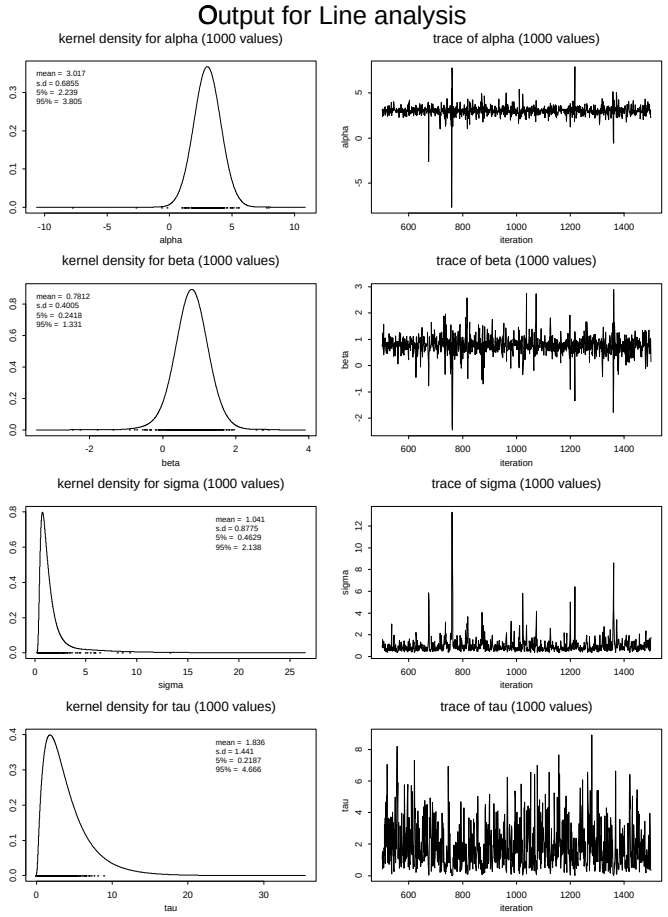

Figure 1: BUGS output for model `line.bug` run for 1000 iterations after a 500 iteration burn-in.

The results may be compared with the exact solutions above. Note the occasional fairly extreme values for the parameters (although these are quite compatible with the posterior distribution). In addition, no checking for convergence has been done, although various diagnostics are available in CODA for that purpose.

1.5 Hardware platforms

Version 0.50: Instructions will be given for SUN Sparc Station, Hewlett-Packard, and PC 386+387/486/586 versions. The default for PCs is that a maths co-processor is available: however there is only a minor reduction in speed without one and we can provide versions that do not need it. Other platforms may soon be available - please contact us.