WinBUGS Implementation

I read in three files—the model, the data, and the initial values—and used the Specification Tool from the Model menu to check the model, load the data, compile the model, load the initial values, and generate additional initial values for uninitialized nodes in the graph.

I then used the Sample Monitor Tool from the Inference menu to set the mu, sigma, nu, and y.new nodes, and clicked on Dynamic Trace plots for mu and nu.

Then choosing the Update Tool from the Model menu, specifying 2000 in the updates box, and clicking update permitted a burn-in of 2,000 iterations to occur with the time series traces of the two parameters displayed in real time.
After **minimizing** the model, data, and inits windows and **killing** the Specification Tool (which are no longer needed until the model is respecified), I typed 10000 in the updates box of the Update Tool and clicked update to generate a **monitoring run** of 10,000 iterations (you can watch the updating of $\mu$ and $\nu$ dynamically to get an idea of the mixing, but this slows down the sampling).

After **killing** the Dynamic Trace window for $\mu$ (to concentrate on $\mu$ for now), in the Sample Monitor Tool I selected $\mu$ from the pull-down menu, set the beg and end boxes to 2001 and 12000, respectively (to summarize only the monitoring part of the run), and clicked on history to get the **time series trace** of the monitoring run, density to get a **kernel density trace** of the 10,000 iterations, stats to get numerical summaries of the monitored iterations, quantiles to get a trace of the **cumulative estimates** of the 2.5%, 50% and 97.5% points in the estimated posterior, and autoc to get the **autocorrelation function**.
You can see that the output for $\mu$ is **mixing fairly well**—the ACF looks like that of an $AR_1$ series with first-order **serial correlation** of only about **0.3**.

$\sigma$ is mixing less well: its ACF looks like that of an $AR_1$ series with first-order **serial correlation** of about **0.6**.

This means that a monitoring run of 10,000 would probably **not be enough** to satisfy **minimal Monte Carlo accuracy goals**—for example, from the Node statistics window the estimated posterior mean is **3.878** with an estimated MC error of **0.0128**, meaning that we've not yet achieved **three-significant-figure accuracy** in this posterior summary.
And $\nu$'s mixing is the worst of the three: its ACF looks like that of an $AR_1$ series with first-order serial correlation of a bit less than $+0.9$.

WinBUGS does not seem to have a provision for printing out the autocorrelations, but you can approximately infer $\hat{\rho}_1$ from an equation like (51) above: assuming that the WinBUGS people are taking the output of any MCMC chain as (at least approximately) $AR_1$ and using the formula

$$\widehat{SE}(\hat{\theta}^*) = \frac{\hat{\sigma}_\theta}{\sqrt{m}} \sqrt{\frac{1 + \hat{\rho}_1}{1 - \hat{\rho}_1}},$$

you can solve this equation for $\hat{\rho}_1$ to get

$$\hat{\rho}_1 = \frac{m \left[ \widehat{SE}(\hat{\theta}^*) \right]^2 - \hat{\sigma}_\theta^2}{m \left[ \widehat{SE}(\hat{\theta}^*) \right]^2 + \hat{\sigma}_\theta^2}.$$
WinBUGS Implementation (continued)

Plugging in the relevant values here gives
\[
\hat{\rho}_1 = \frac{(10,000)(0.04253)^2 - (1.165)^2}{(10,000)(0.04253)^2 + (1.165)^2} = 0.860,
\]  
which is smaller than the corresponding value of 0.972 generated by the classicBUGS sampling method (from CODA, page 67).

To match the classicBUGS strategy outlined above (page 71) I typed 30000 in the updates window in the Update Tool and hit update, yielding a total monitoring run of 40,000.

Remembering to type 42000 in the end box in the Sample Monitoring Tool window before going any further, to get a monitoring run of 40,000 after the initial burn-in of 2,000, the summaries below for \( \mu \) are satisfactory in every way.
WinBUGS Implementation (continued)

A monitoring run of 40,000 also looks good for $\sigma$: on this basis, and conditional on this model and prior, I think $\sigma$ is around 3.87 (posterior mean, with an MCSE of 0.006), give or take about 0.44 (posterior SD), and my 95% central posterior interval for $\sigma$ runs from about 3.09 to about 4.81 (the distribution has a bit of skewness to the right, which makes sense given that $\sigma$ is a scale parameter).
If the real goal were $\nu$ I would use a longer monitoring run, but the main point here is $\mu$, and we saw back on p. 67 that $\mu$ and $\nu$ are close to uncorrelated in the posterior, so this is good enough.

If you wanted to report the posterior mean of $\nu$ with an MCSE of 0.01 (to come close to 3-sigfig accuracy) you’d have to increase the length of the monitoring run by a multiplicative factor of $\left(\frac{0.02213}{0.01}\right)^2 \approx 4.9$, which would yield a recommended length of monitoring run of about 196,000 iterations (the entire monitoring phase would take about 3 minutes at 2.0 (PC) GHz).
The posterior predictive distribution for $y_{n+1}$ given $(y_1, \ldots, y_n)$ is interesting in the $t$ model: the predictive mean and SD of 404.3 and 6.44 are not far from the sample mean and SD (404.6 and 6.5, respectively), but the predictive distribution has very heavy tails, consistent with the degrees of freedom parameter $\nu$ in the $t$ distribution being so small (the time series trace has a few simulated values less than 300 and greater than 500, much farther from the center of the observed data than the most outlying actual observations).
Gaussian Comparison

The posterior SD for $\mu$, the only parameter directly comparable across the Gaussian and $t$ models for the NB10 data, came out $0.47$ from the $t$ modeling, versus $0.65$ with the Gaussian, i.e., the interval estimate for $\mu$ from the (incorrect) Gaussian model is about 40% wider that that from the (much better-fitting) $t$ model.
A Model Uncertainty Anomaly?

**NB** Moving from the Gaussian to the $t$ model involves a net increase in **model uncertainty**, because when you assume the Gaussian you’re in effect saying that you know the $t$ degrees of freedom are $\infty$, whereas with the $t$ model you are treating $\nu$ as unknown. And yet, even though there’s been an increase in model uncertainty, the inferential uncertainty about $\mu$ has gone down.

This is relatively rare—**usually when model uncertainty increases so does inferential uncertainty** (Draper 2002)—and arises in this case because of two things: (a) the $t$ model **fits better** than the Gaussian, and (b) the Gaussian is actually a **conservative** model to assume as far as inferential accuracy for location parameters is concerned.
References


