MCMC Diagnostics (continued)

```
GEWEKE CONVERGENCE DIAGNOSTIC (Z-score):
========================================

Iterations used = 1002:6001     Fraction in
Thinning interval = 1             1st window = 0.1
Sample size per chain = 5000    Fraction in
                                  2nd window = 0.5
-+---------+-------------+-
  | VARIABLE |   bugs1     |
  | ======== |   =====     |
  |          |             |
  | mu       |   2.39      |
  | nu       |   1.78      |
  | sigma    |   1.14      |
  |          |             |
-+---------+-------------+-
```

Here for run 1 with the NB10 data (the left-hand set of commands in the table on p. 54) there is **some evidence of nonstationarity** with a burn-in of only 1,000 (although a z-value of 2.4 is **not overwhelming**).

Gelman-Rubin (1992) have suggested a diagnostic that looks for **multimodality** of the posterior distribution.

If the posterior has (say) two major modes that are far away from each other in parameter space, and you initialize the chain near one of the modes,
you may never find the other one.

The idea is to run the chain two or more times from widely-dispersed starting points and **see if you always converge to the same place**.

Gelman and Rubin do what amounts to an **analysis of variance** within and between the chains, looking for evidence of **large variability between them**.

Gelman-Rubin Shrink Factors

"This comparison is used to estimate the factor by which the scale parameter of the marginal posterior distribution of each [quantity being monitored] might [**shrink**] if the chain were run to infinity" (Best et al., 1995).

The output is the 50% and 97.5% quantiles of the distributions of **shrink factors**, one for each quantity monitored.

If these quantiles are both close to 1.0 then there is **little evidence of dispersion** between the distributions to which the chains are converging.

```
GELMAN AND RUBIN 50% AND 97.5% SHRINK FACTORS:
==============================================

Iterations used for diagnostic  = 2501:5000
Thinning interval = 1
Sample size per chain = 5000

-+----------+-----------------------------+-
  | VARIABLE |  Point est. 97.5% quantile  |
  | ======== |  ========== ==============  |
  |          |                             |
  | mu       |  1.00       1.00            |
  | nu       |  1.00       1.01            |
  | sigma    |  1.00       1.00            |
  |          |                             |
-+----------+-----------------------------+-
```

Here, with initial values as different as $(\mu, \tau, \nu) = (405.0, 0.1823, 5.0)$ and $(402.0, 0.03, 11.0)$ there is **no evidence of multimodality** at all.

Raftery-Lewis Dependence Factors

(To be really safe I should run a **number of additional chains**—Gelman and Rubin (1992) give advice on how to generate the set of initial values to try—but with even modest sample sizes (like $n = 100$) the posterior in t models is **unimodal** so there would be no point in this case.)

Raftery-Lewis (1992) suggested a **diagnostic** that directly helps to answer question (3)—How do you pick b and m?

The answer to this question depends on how **accurate** you want your posterior summaries to be, so Raftery and Lewis require you to **input** three values:

(a) **Which quantiles** of the marginal posteriors are you most interested in?

Usually the answer is the **2.5%** and **97.5%** points, since they're the basis of a 95% interval estimate.

(b) How close to the **nominal levels** would you like the **estimated quantiles** to be?

The CODA default is **0.5%**, e.g., if the left-hand value of your 95% interval is supposed to be at the **2.5%** point of the distribution, CODA will recommend a length of monitoring run so that the actual level of this quantile will be between **2.0%** and **3.0%**.

(**NB** This is sometimes more, and **often less**, Monte Carlo accuracy than you really need.)

(c) With what minimum **probability** do you want to achieve these accuracy goals? The default is **95%**.

Having input these values, the output is of **five kinds** for each quantity monitored:

(a) A recommended **thinning interval**. When the Gibbs sampler is performing poorly people say the output is not **mixing well**, and what they mean is that the Markovian nature of the time series for each quantity has led to **large positive serial autocorrelations** in time, e.g., μ_{1000} depends highly on μ_{999}, μ_{998}, and so on.

Raftery-Lewis (continued)

This is another way to say that the **random draws** in the simulation process are **not moving around the parameter space quickly**.

When this happens, one way to reduce the autocorrelation is to **run the chain a lot longer and only record every kth iteration**—this is the **thinning interval** (**NB** this only reduces the autocorrelation of the **saved** MCMC data set; the underlying Markov chain is of course **unaffected** by this).

(b) A recommended length of **burn-in** to use, above and beyond whatever you've already done.

(c) A recommended **total length of run** N (including burn-in) to achieve the desired accuracy.

(d) A **lower bound** N_{min} on run length—what the minimum would have needed to be if the quantity in question had an **IID** time series instead of an autocorrelated series.

(e) And finally, the ratio $I = N/N_{min}$, which Raftery and Lewis call the **dependence factor**—values of I near 1 indicate good mixing.

```
RAFTERY AND LEWIS CONVERGENCE DIAGNOSTIC:
=========================================

Iterations used = 1001:6000
Thinning interval = 1
Sample size per chain = 5000

Quantile = 0.025
Accuracy = +/- 0.005
Probability = 0.95
```

VARIABLE	Thin (k)	Burn-in (M)	Total (N)	Lower bound (Nmin)	Dependence factor (I)
mu	1	3	4533	3746	1.21
nu	3	18	39720	3746	10.6
sigma	3	12	13308	3746	3.55

Heidelberger-Welch Diagnostic

Here μ is mixing well—5,000 iterations are sufficient to achieve the default accuracy goal—but σ **and (especially)** ν **require longer monitoring periods**: the recommendation is to run for about 40,000 iterations and store every third.

Heidelberger-Welch (1983) propose a diagnostic approach that uses the **Cramér-von Mises statistic** to test for stationarity.

If **overall stationarity** fails for a given quantity being monitored, CODA **discards** the first 10% of the series for that quantity and recomputes the **C-vonM** statistic, continuing in this manner until only the final 50% of the data remain.

If stationarity still fails with the last half of the data then CODA reports **overall failure** of the stationarity test.

CODA also computes a **half-width** test, which tries to judge whether the portion of the series that passed the stationarity test is sufficient to estimate the posterior mean with a particular default accuracy (**NB** this default is often not stringent enough for careful numerical work).

Here the table below shows that the first run with the NB10 data **clears the Heidelberger-Welch hurdle** with ease.

Autocorrelations and Cross-correlations. CODA also computes the **autocorrelations** for each monitored quantity at lags from 1 to 50 and the **cross-correlations** between all of the variables.

As mentioned previously, the **autocorrelation** at **lag** k of a time series $\{\theta_t^*, t = 1, \ldots, m\}$ (e.g., Chatfield 1996) measures the extent to which the series at time $(t + k)$ and at time t are **linearly related**, for $k = 1, 2, \ldots$.

MCMC Diagnostics (continued)

```
HEIDELBERGER AND WELCH STATIONARITY AND INTERVAL HALFWIDTH TESTS:
================================================================

Precision of halfwidth test = 0.1

-+----------+-------------------------------------------------------+-
 |          |  Stationarity   # of iters.  # of iters.  C-vonM      |
 | VARIABLE |      test         to keep    to discard    stat.      |
 | ======== |  ============   ==========  ==========  ======      |
 |          |                                                       |
 | mu       |  passed         5000        0           0.126       |
 | nu       |  passed         5000        0           0.349       |
 | sigma    |  passed         5000        0           0.176       |
 |          |                                                       |
-+----------+-------------------------------------------------------+-
 |          |  Halfwidth                          |
 | VARIABLE |    test       Mean    Halfwidth     |
 | ======== |  =========    ====    =========     |
 |          |                                     |
 | mu       |  passed      404.00   0.0160        |
 | nu       |  passed        3.75   0.1500        |
 | sigma    |  passed        3.89   0.0344        |
 |          |                                     |
-+----------+-------------------------------------+-
```

The usual **sample estimate** of this quantity is

$$r_k = \frac{c_k}{c_0}, \text{ where } c_k = \frac{1}{m-k} \sum_{t=1}^{m-k} \left(\theta_t^* - \bar{\theta^*}\right) \left(\theta_{t+k}^* - \bar{\theta^*}\right) \qquad (47)$$

$$\text{and } \bar{\theta^*} = \tfrac{1}{m} \textstyle\sum_{t=1}^{m} \theta_t^*.$$

The **cross-correlation** at **lag** k of two time series $\{\theta_t^*, t = 1, \ldots, m\}$ and $\{\eta_t^*, t = 1, \ldots, m\}$ measures the extent to which the first series at time $(t + k)$ and the second at time t are **linearly related**, for $k = 1, 2, \ldots$.

A **natural sample estimate** of this quantity is

$$r_{\theta\eta}(k) = \frac{c_{\theta\eta}(k)}{\sqrt{c_{\theta\theta}(0) c_{\eta\eta}(0)}}, \quad \text{where}$$

$$c_{\theta\eta}(k) = \frac{1}{m-k} \sum_{t=1}^{m-k} \left(\theta_t^* - \bar{\theta^*}\right) \left(\eta_{t+k}^* - \bar{\eta^*}\right). \qquad (48)$$

MCMC Diagnostics (continued)

LAGS AND AUTOCORRELATIONS WITHIN EACH CHAIN:

Chain	Variable	Lag 1	Lag 10	Lag 50
bugs1	mu	0.29400	0.00118	-0.01010
	nu	0.97200	0.78900	0.32100
	sigma	0.62100	0.30300	0.10800

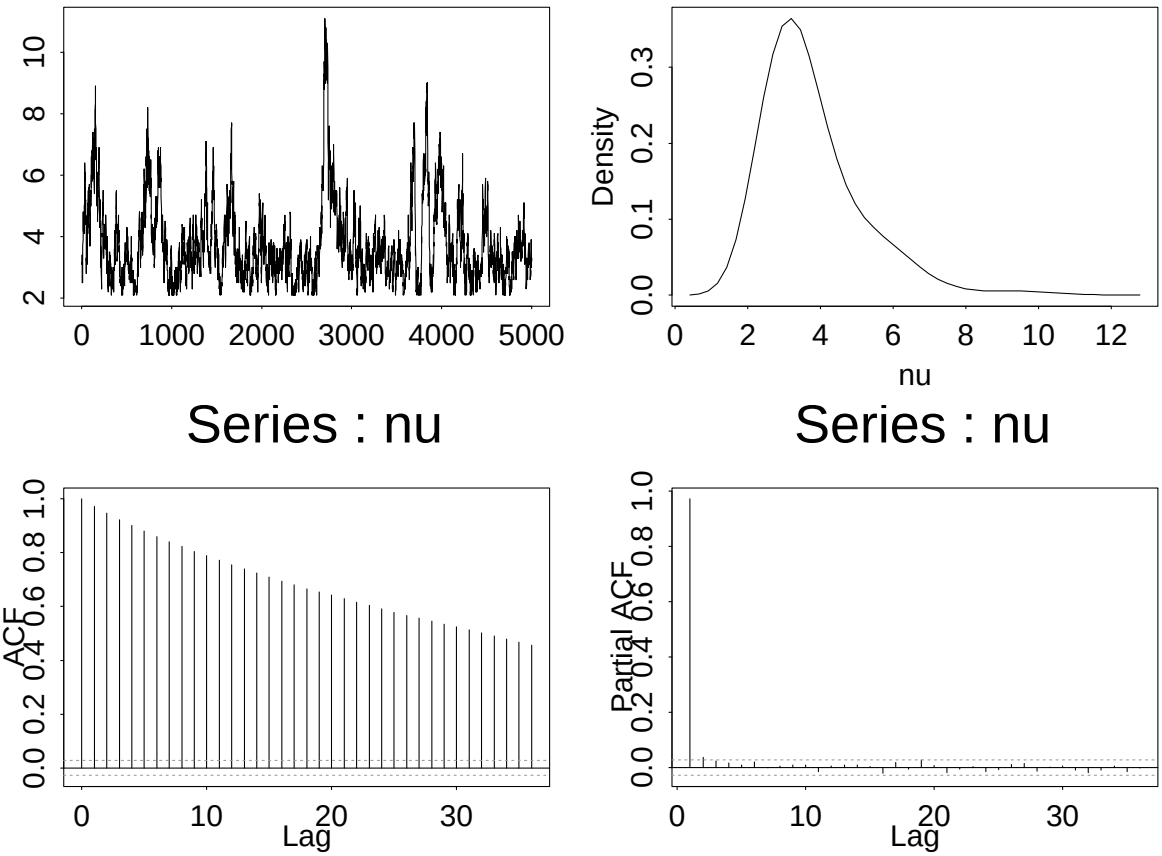
CROSS-CORRELATION MATRIX:

VARIABLE	mu	nu	sigma
mu	1.0000		
nu	0.0946	1.0000	
sigma	0.0534	0.5540	1.0000

You can see (a) that the series for ν is **especially strongly autocorrelated**, and (b) that ν and σ are **fairly strongly positively correlated**, which connects with the observation earlier about confounding of scale and shape in the t family.

Diagnostic and Summary Plots. The figure below presents four plots that are useful as **MCMC diagnostics** and for **graphical summaries of posterior distributions**, in the case of the parameter ν with run 1 from the NB10 data.

Diagnostic and Summary Plots



The upper left panel is a **time series trace**, which documents the poor mixing that has been evident from several of the numerical diagnostics.

The lower left panel is a plot of the **autocorrelation** function (ACF) for ν, and the lower right panel plots the **partial autocorrelation** function (PACF).

One of the most common behaviors observed in time series in general, and in the output of MCMC samplers in particular, is that of an **autoregressive process**.

Letting e_t denote an IID (or **white-noise** or **purely random**) process with mean 0 and variance σ_e^2, the time series θ_t^* is said to be an **autoregressive process of order** p (AR_p) if

$$\theta_t^* = \alpha_1 \theta_{t-1}^* + \ldots + \alpha_p \theta_{t-p}^* + e_t. \tag{49}$$

Diagnostic and Summary Plots

Equation (49) is like a **multiple regression model** except that θ_t^* is being regressed on **past values of itself** instead of on other predictor variables; this gives rise to the term **autoregressive**.

The **partial autocorrelation function** (PACF) measures the excess correlation between θ_t^* and θ_{t+k}^* not accounted for by the autocorrelations $r_1, \ldots, r_{k-1}$, and is useful in **diagnosing the order** of an AR_p process: if θ_t^* is AR_p then the PACF at lags $1, \ldots, p$ will be significantly different from 0 and then close to 0 at lags larger than p.

The lower right-hand plot above shows the characteristic **single spike at lag 1** which diagnoses an AR_1 series (the dotted lines in the ACF and PACF plots represent **2 standard error traces around 0**, indicating how big an ACF or PACF value needs to be to be significantly different from 0).

This is reinforced by the ACF plot: if θ_t^* is AR_1 with positive first-order autocorrelation ρ_1 then the autocorrelation function should show a **slow geometric decay** (a **ski-slope** shape), which it clearly does in this case.

We would conclude that the Gibbs sampling output for ν, when thought of as a **time series**, behaves like an AR_1 process with **first-order autocorrelation** roughly $r_1 = 0.972$ (from the table above).

MCMC Accuracy. Suppose that θ_t^* is a **stationary time series** with underlying true mean μ_θ and variance σ_θ^2.

MCMC Accuracy

It can be shown that if $\{\theta_t^*, t = 1, \ldots, m\}$ is AR_1 with first-order autocorrelation ρ_1 then **in repeated sampling** the **uncertainty about** μ_θ on the basis of the sample mean $\bar{\theta}^*$ is quantified by

$$V(\bar{\theta}^*) = \frac{\sigma_\theta^2}{m}\left(\frac{1+\rho_1}{1-\rho_1}\right). \tag{50}$$

Thus if you want to use MCMC to estimate the posterior mean of a given quantity θ with **sufficient accuracy** that the standard error of the Monte Carlo mean estimate $\bar{\theta}^*$ based on a monitoring run of length m is **no larger than a specified tolerance** T, and the MCMC output θ^* behaves like an AR_1 series with first-order autocorrelation ρ_1, you would need m to satisfy

$$\widehat{SE}(\bar{\theta}^*) = \frac{\widehat{\sigma}_\theta}{\sqrt{m}}\sqrt{\frac{1+\widehat{\rho}_1}{1-\widehat{\rho}_1}} \leq T, \tag{51}$$

from which

$$m \geq \frac{\widehat{\sigma}_\theta^2}{T^2}\left(\frac{1+\widehat{\rho}_1}{1-\widehat{\rho}_1}\right). \tag{52}$$

This formula explains why monitoring runs with MCMC often need to be **quite long**: as $\rho_1 \to 1$ the required $m \to \infty$.

For example, we've seen that $\widehat{\rho}_1 = r_1$ for ν in the NB10 t model is +0.972, and we'll see below that the **sample mean and SD** based on the output for ν are roughly 3.628 and 1.161, respectively.

If you wanted to be able to report the posterior mean of ν to **3–significant-figure accuracy** (3.63) with reasonably high Monte Carlo probability, you would want T to be **on the order of 0.01**, giving an enormous monitoring run:

Diagnostic Plots (continued)

$$m \geq \left(\frac{1.161}{0.01}\right)^2 \left(\frac{1+0.972}{1-0.972}\right) \doteq (13,479)(70.4) \doteq 949,322 \tag{53}$$

This is much larger than the **Raftery-Lewis default recommendation** above (there is no conflict in this fact; the two diagnostics are focusing on **different posterior summaries**).

Note from (52) that if you could figure out how to sample **in an IID manner** from the posterior for θ you would only need $m_{\text{IID}} \geq \frac{\widehat{\sigma}_\theta^2}{T^2}$, which in this case is about 13,500 draws.

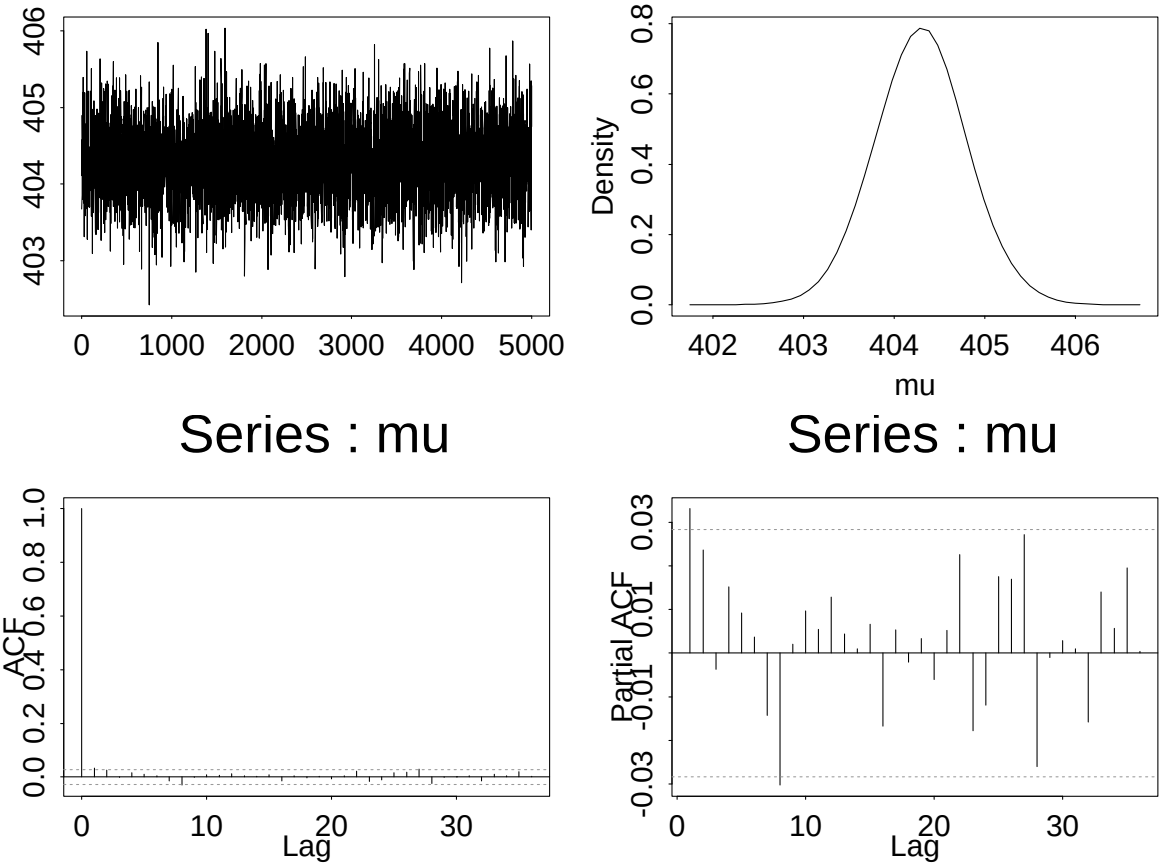
The term $\left(\frac{1+\widehat{\rho}_1}{1-\widehat{\rho}_1}\right)$ in (52) represents the amount by which m_{IID} would need to be multiplied to get the same accuracy from MCMC output—it's natural to call this the **sample size inflation factor** (SSIF), which for ν comes out a whopping 70.4.

The upper right panel in the diagnostic plots above gives a **density trace** for ν, which shows a mode at about 3 degrees of freedom and a long right-hand tail.

Round 2. From all of this I decided to run the chain again with the BUGS commands in the right-hand part of the table on page 54: a **burn-in** of 2,000 and a **monitoring run** of 40,000, thinning the output by writing out to disk only every **8th draw** (thus ending up with 5,000 stored values).

The MCMC diagnostics were **much better**: Raftery-Lewis total N recommendations all less than 5,000, all other summaries fine.

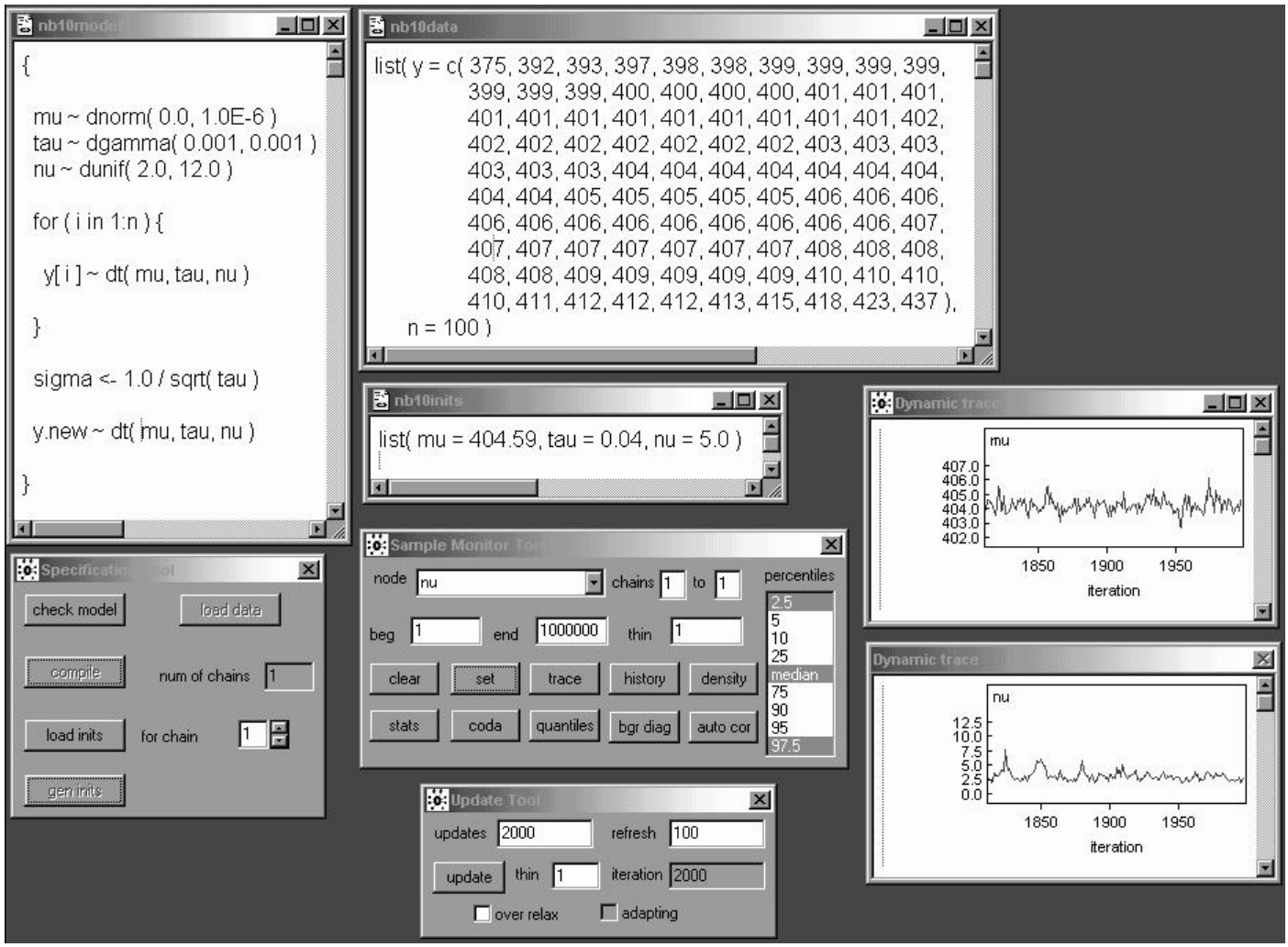
Numerical Summaries



All the parameters are **mixing well now**, so numerical posterior summaries are worth making, as in the table below.

Parameter	Posterior Mean	Posterior SD	95% Interval
μ	404.3	0.4641	(403.4, 405.2)
ν	3.63	1.16	(2.2, 6.6)
σ	3.873	0.4341	(3.100, 4.778)

WinBUGS Implementation

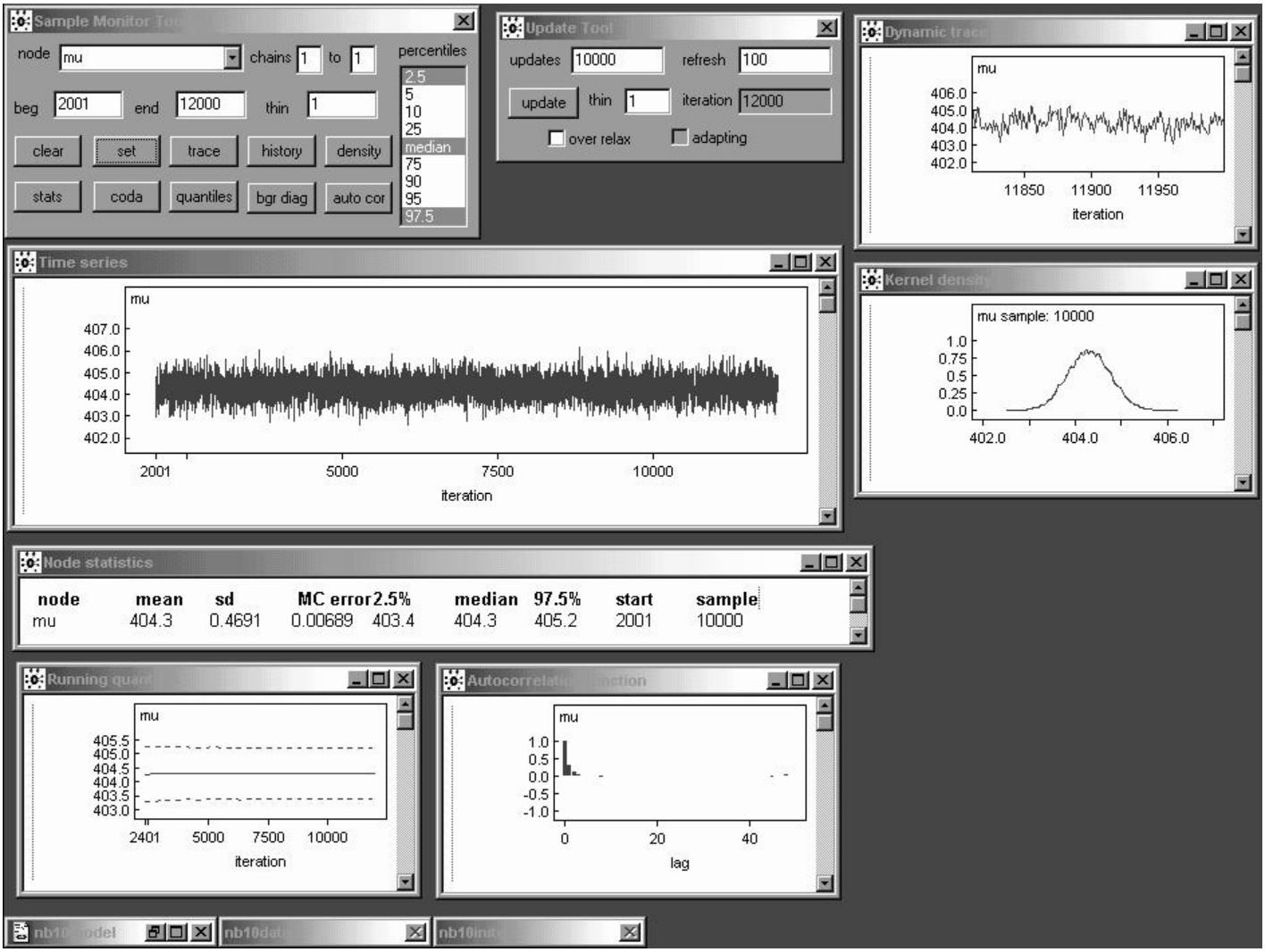


I read in three files—the **model**, the **data**, and the **initial values**—and used the `Specification Tool` from the `Model` menu to `check` the model, `load` the data, `compile` the model, `load` the initial values, and `generate` additional initial values for uninitialized nodes in the graph.

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**.

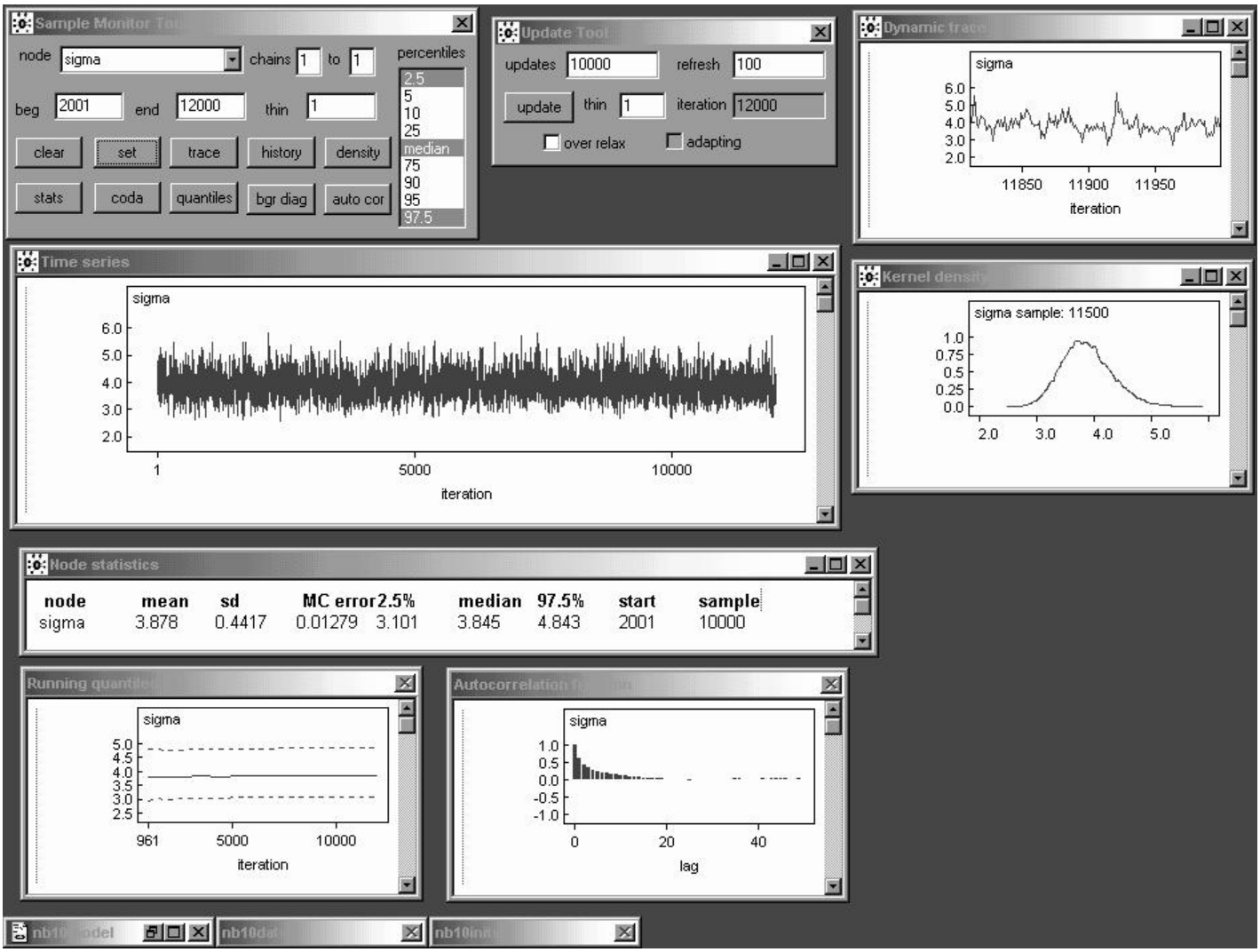
WinBUGS Implementation (continued)



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 `nu` (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**.

WinBUGS Implementation (continued)



You can see that the output for μ is **mixing fairly well**—the ACF looks like that of an AR_1 series with first-order **serial correlation** of only about **0.3**.

σ 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.