Introduction to Bayesian Data Analysis

4: Bayesian Random-Effects Hierarchical Modeling

David Draper

Department of Applied Mathematics and Statistics
University of California, Santa Cruz

draper@ams.ucsc.edu
http://www.ams.ucsc.edu/∼draper

Centers for Disease Control and Prevention
(Atlanta GA)

16–17 June 2008

4.1 Formulating hierarchical models for quantitative outcomes from scientific context

Meta-analysis of effects of aspirin on heart attacks. Table 2.1 (Draper et al., 1993a) gives the number of patients and mortality rate from all causes, for six randomized controlled experiments comparing the use of aspirin and placebo by patients following a heart attack.

Table 2.1. Aspirin meta-analysis data.

	Aspirin		Placebo	
Study (i)	# of Patients	Mortality Rate (%)	# of Patients	Mortality Rate (%)
UK-1	615	7.97	624	10.74
CDPA	758	5.80	771	8.30
GAMS	317	8.52	309	10.36
UK-2	832	12.26	850	14.82
PARIS	810	10.49	406	12.81
AMIS	2267	10.85	2257	9.70
Total	5599	9.88	5217	10.73

	Comparison			
Study (i)	$y_i =$ Diff (%)	$\sqrt{V_i} =$ SE of Diff (%)	$Z_i^{\ddagger}$	$p_i^{\S}$
UK-1	2.77	1.65	1.68	.047
CDPA	2.50	1.31	1.91	.028
GAMS	1.84	2.34	0.79	.216
UK-2	2.56	1.67	1.54	.062
PARIS	2.31	1.98	1.17	.129
AMIS	−1.15	0.90	−1.27	.898
Total	0.86	0.59	1.47	.072

$^{\ddagger}Z_i$ denotes the ratio of the difference in mortality rates over its standard error, assuming a binomial distribution. $^{\S}p_i$ is the one-sided p value associated with Z_i, using the normal approximation.

Meta-Analysis

The first five trials are reasonably consistent in showing a (weighted average) mortality decline for aspirin patients of 2.3 percentage points, a 20% drop from the (weighted average) placebo mortality of 11.5% (this difference is **highly clinically significant**). However, the sixth and largest trial, AMIS, went the other way: an *increase* of 1.2 percentage points in aspirin mortality (a 12% rise from the placebo baseline of 9.7%).

Some relevant questions (Draper, 1995): $\boxed{\mathbf{Q}_1}$ Why did AMIS get such different results? $\boxed{\mathbf{Q}_2}$ What should be done next to reduce the uncertainty about Q_1? $\boxed{\mathbf{Q}_3}$ If I were a doctor treating a patient like those eligible for the trials in Table 2.1, what therapy should I employ while answers to Q_1 and Q_2 are sought? One possible paraphrase of Q_3: $\boxed{\mathbf{Q}_4}$ How should the information from these six experiments be **combined** to produce a more informative summary than those obtained from each experiment by itself?

The discipline of $\boxed{\textbf{meta-analysis}}$ is devoted to answering questions like Q_4. One leading school of frequentist meta-analysis (e.g., Hedges and Olkin, 1985) looks for methods for combining the Z and p values in Table 2.1, an approach that often leads only to an overall p value. A more satisfying form of meta-analysis (which has both frequentist and Bayesian versions) builds a **hierarchical model (HM)** that indicates how to combine information from the mortality differences in the Table. A Gaussian meta-analysis model for the aspirin data, for example (Draper et al., 1993a), might look like

$$\begin{array}{rcll} (\theta, \sigma^2) & \sim & p(\theta, \sigma^2) & \text{(prior)} \\ (\theta_i \mid \theta, \sigma^2) & \overset{\text{IID}}{\sim} & N\big(\theta, \sigma^2\big) & \text{(underlying effects)} \qquad (1) \\ (y_i \mid \theta_i) & \overset{\text{indep}}{\sim} & N(\theta_i, V_i) & \text{(data)} \, . \end{array}$$

A Gaussian HM

The bottom level of (10), the **data** level of the HM, says that—because of relevant differences in patient cohorts and treatment protocols—each study has its own underlying treatment effect θ_i, and the observed mortality differences y_i are like random draws from a normal distribution with mean θ_i and variance V_i (the normality is reasonable because of the CLT, given the large numbers of patients). In meta-analyses of data like those in Table 2.1 the V_i are typically taken to be known (again because the patient sample sizes are so big), $V_i = SE_i^2$, where SE_i is the standard error of the mortality difference for study i in Table 2.1.

The middle level of the HM is where I would bring in the $\boxed{\textbf{study-level covariates}}$, if I have any, to try to explain why the studies differ in their underlying effects. Here there are no study-level covariates, so the middle level of (10) is equivalent to a Gaussian regression with no predictor variables. Why the "error" distribution should be Gaussian at this level of the HM is not clear—it's a conventional choice, not a choice that is automatically scientifically reasonable (in fact I will challenge it later). σ^2 in this model represents **study-level heterogeneity**.

The top level of (10) is where the **prior** distribution on the regression parameters from the middle level is specified. Here, with only an intercept term in the regression model, a popular conventional choice is a normal/**scaled-inverse**-χ^2 prior (more about this later; see Gelman et al. 2003).

$\boxed{\textbf{Fixed effects and random effects.}}$ If σ^2 were known somehow to be 0, all of the θ_i would have to be equal deterministically to a common value θ, yielding a simpler model: $(y_i|\theta) \overset{\text{indep}}{\sim} N(\theta, V_i), \theta \sim p(\theta)$. Meta-analysts call this a **fixed-effects** model, and refer to model (10) as a **random-effects** model. When σ^2 is not assumed to be 0, with this terminology the θ_i are called **random effects**.

4.2 Approximate Fitting of Gaussian Hierarchical Models: Maximum Likelihood and Empirical Bayes

Fitting HM (10). Some algebra based on model (10) yields that the conditional distributions of the study-level effects θ_i given the data and the parameters (θ, σ^2), have a simple and revealing form:

$$(\theta_i | y_i, \theta, \sigma^2) \overset{\text{indep}}{\sim} N(\theta_i^*, V_i(1 - B_i)) \qquad (2)$$

$$\text{with} \quad \theta_i^* = (1 - B_i)\, y_i + B_i\, \theta \quad \text{and} \quad B_i = \frac{V_i}{V_i + \sigma^2}. \qquad (3)$$

In other words, the conditional mean of the effect for study i given y_i, θ, and σ^2 is a $\boxed{\textbf{weighted average}}$ of the sample mean for that study, y_i, and the overall mean θ.

The weights are given by the so-called **shrinkage factors** B_i (e.g., Draper et al., 1993a), which in turn depend on how the variability V_i within study i compares to the between-study variability σ^2: the more accurately y_i estimates θ_i, the more weight the "local" estimate y_i gets in the weighted average. The term *shrinkage* refers to the fact that, with this approach, unusually high or low individual studies are drawn back or "shrunken" toward the overall mean in the calculation $(1 - B_i)\, y_i + B_i\, \theta$. Note that θ_i^* uses data from all the studies to estimate the effect for study i—this is referred to as **borrowing strength** in the estimation process.

Closed-form expressions for $p(\theta|y)$ and $p(\theta_i|y)$ with $y = (y_1, \ldots, y_k), k = 6$ are not available even with a normal/scaled-inverse-χ^2 prior for (θ, σ^2); a full Bayesian analysis of these data is most readily accomplished with **Markov chain Monte Carlo** methods, to which we will soon turn.

Maximum Likelihood and Empirical Bayes

In the meantime $\boxed{\textbf{maximum likelihood}}$ calculations provide some idea of what to expect: the likelihood function based on model (10) is

$$l(\theta, \sigma^2 | y) = c \prod_{i=1}^{k} \frac{1}{\sqrt{V_i + \sigma^2}} \exp\left[-\frac{1}{2} \sum_{i=1}^{k} \frac{(y_i - \theta)^2}{V_i + \sigma^2} \right]. \qquad (4)$$

The maximum likelihood estimates (MLEs) $(\hat{\theta}, \hat{\sigma}^2)$ then turn out to be the **iterative** solutions to the following equations:

$$\hat{\theta} = \frac{\sum_{i=1}^{k} \hat{W}_i \, y_i}{\sum_{i=1}^{k} \hat{W}_i} \quad \text{and} \quad \hat{\sigma}^2 = \frac{\sum_{i=1}^{k} \hat{W}_i^2 \left[(y_i - \hat{\theta})^2 - V_i \right]}{\sum_{i=1}^{k} \hat{W}_i^2}, \qquad (5)$$

$$\text{where} \quad \hat{W}_i = \frac{1}{V_i + \hat{\sigma}^2}. \qquad (6)$$

Start with $\hat{\sigma}^2 = 0$ and iterate (14–15) to convergence (if $\hat{\sigma}^2$ converges to a negative value, $\hat{\sigma}^2 = 0$ is the MLE). The MLEs of the θ_i are then given by

$$\hat{\theta}_i = (1 - \hat{B}_i) \, y_i + \hat{B}_i \, \theta \quad \text{where} \quad \hat{B}_i = \frac{V_i}{V_i + \hat{\sigma}^2}. \qquad (7)$$

These are called $\boxed{\textbf{empirical Bayes}}$ (EB) estimates of the study-level effects, because it turns out that this analysis approximates a fully Bayesian solution by (in effect) using the data to estimate the prior specifications for θ and σ^2.

Large-sample (mainly meaning large k) approximations to the (frequentist) distributions of the MLEs are given by

$$\hat{\theta} \sim N\left(\theta, \left[\sum_{i=1}^{k} \frac{1}{V_i + \hat{\sigma}^2} \right]^{-1} \right) \quad \text{and} \quad \hat{\theta}_i \sim N\big(\theta_i, V_i (1 - \hat{B}_i) \big). \quad (8)$$

MLEB (continued)

NB The variances in (17) *do not account fully for the uncertainty in* σ^2 and therefore underestimate the actual sampling variances for small k (adjustments are available; see, e.g., Morris (1983, 1988)).

MLEB estimation can be implemented simply in about 15 lines of `S+` or `R` code (Table 2.2).

Table 2.2. `S+` or `R` program to perform MLEB calculations.

```
mleb <- function( y, V, m ) {
  sigma.2 <- 0.0
  for ( i in 1:m ) {
    W <- 1.0 / ( V + sigma.2 )
    theta <- sum( W * y ) / sum( W )
    sigma.2 <- sum( W^2 * ( ( y - theta )^2 - V ) ) / sum( W^2 )
    B <- V / ( V + sigma.2 )
    effects <- ( 1 - B ) * y + B * theta
    se.theta <- 1.0 / sqrt( sum( 1.0 / ( V + sigma.2 ) ) )
    se.effects <- sqrt( V * ( 1.0 - B ) )
    print( c( i, theta, se.theta, sigma.2 ) )
    print( cbind( W, ( W / sum( W ) ), B, y, effects, se.effects ) )
  }
}
```

With the aspirin data it takes 18 iterations (less than 0.1 second on a 400MHz `UltraSPARC` Unix machine) to get 4-digit convergence to the summaries in Table 2.3 and the following estimates (standard errors in parentheses):

$$\widehat{\theta} = 1.45 \ (0.809), \quad \widehat{\sigma} = 1.24.$$

Table 2.3. Maximum likelihood empirical Bayes meta-analysis of the aspirin data.

study(i)	$\widehat{W}_i$	normalized $\widehat{W}_i$	$\widehat{B}_i$	y_i	$\widehat{\theta}_i$	$\widehat{SE}(\widehat{\theta}_i)$
1	0.235	0.154	0.640	2.77	1.92	0.990
2	0.308	0.202	0.529	2.50	1.94	0.899
3	0.143	0.0934	0.782	1.84	1.53	1.09
4	0.232	0.151	0.646	2.56	1.84	0.994
5	0.183	0.120	0.719	2.31	1.69	1.05
6	0.427	0.280	0.346	−1.15	−0.252	0.728

Aspirin Meta-Analysis: Conclusions

Note that (1) AMIS gets much less weight (normalized $\widehat{W}_i$) than would have been expected given its small V_i; (2) the shrinkage factors ($\widehat{B}_i$) are considerable, with AMIS shrunk almost all the way into positive territory (see Figure 2.1); (3) there is **considerable study-level heterogeneity** ($\widehat{\sigma} \doteq 1.24$ percentage points of mortality); and (4) the standard errors of the effects are by and large smaller than the $\sqrt{V_i}$ (from the borrowing of strength) but are still considerable.

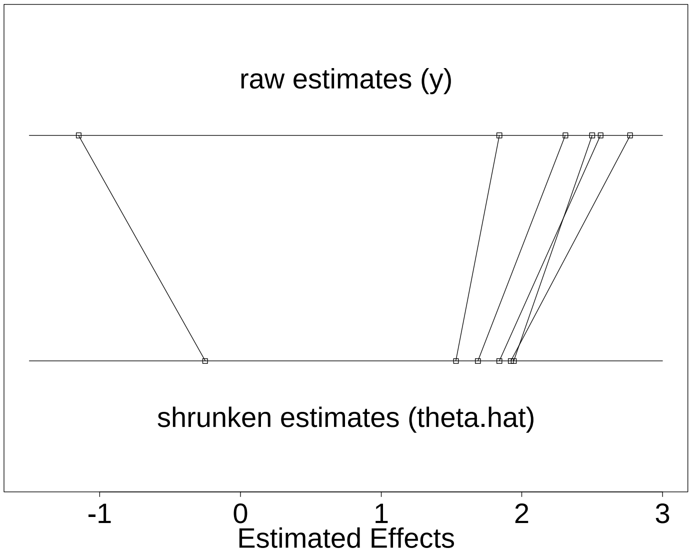


Figure 2.1. **Shrinkage plot** for the aspirin MLEB meta-analysis.

The 95% interval estimate of θ, the overall underlying effect of aspirin on mortality, from this approach comes out

$$\widehat{\theta} \pm 1.96 \cdot \widehat{SE}(\widehat{\theta}) \doteq (-0.140, 3.03),$$

which if interpreted Bayesianly gives

$$\boxed{P(\text{aspirin reduces mortality}|\text{data}) \doteq 1 - \Phi\left(\frac{0-1.45}{0.809}\right) = \mathbf{0.96}},$$

where Φ is the standard normal CDF. Thus although the interval includes 0, so that in a frequentist sense the effect is not statistically significant, **in fact from a Bayesian point of view the evidence is running strongly in favor of aspirin's usefulness**.

WinBUGS Analysis of Aspirin Data

Aspirin meta-analysis revisited. I create three files for WinBUGS: a **model** file, a **data** file, and an **initial values** file (I'm using the most recent release, 1.4.1, of WinBUGS).

The (first) **model** file for the aspirin data:

```
{

  mu ~ dnorm( 0.0, 1.0E-6 )
  tau.theta ~ dgamma( 1.0E-3, 1.0E-3 )

  for ( i in 1:k ) {

    theta[ i ] ~ dnorm( mu, tau.theta )
    y[ i ] ~ dnorm( theta[ i ], tau.y[ i ] )

  }

  sigma.theta <- 1.0 / sqrt( tau.theta )

}
```

`WinBUGS` Analysis of Aspirin Data

Here μ plays the role of θ in model (10) above to avoid using the name `theta` twice for two different purposes in the `WinBUGS` program.

In specifying a normal distribution `WinBUGS` works not with a **standard deviation** (SD) or a **variance** but with a **precision**—the **reciprocal** of the variance—so that the $N(\mu, \sigma^2)$ distribution is specified by `dnorm( mu, tau )` with $\tau = \frac{1}{\sigma^2}$.

Then the **SD** has to be computed as a derived quantity ($\sigma = \frac{1}{\sqrt{\tau}}$) which is written above as

```
sigma.theta <- 1.0 / sqrt( tau.theta )
```

If—before the aspirin experiments were performed—I'm relatively **ignorant** about the quantities θ (μ) and σ in model (10), or equivalently μ and $\tau = \frac{1}{\sigma^2}$, I can build a **diffuse** or `flat` prior for both quantities that expresses this relative ignorance.

Since μ lives on $(-\infty, \infty)$ a convenient choice for a flat prior for it is a **normal** distribution with mean (say) 0 and very small precision: `mu` $\sim$ `dnorm( 0.0, 1.0E-6 )`

For `tau.theta`, which lives on $(0, \infty)$, I want something that's flat throughout (almost) all of that range; a convenient choice (to get an **initial idea** of where the posterior distribution for `sigma.theta` is **concentrated**) is a **gamma** distribution with small positive values of both of its parameters.

This is the $\boxed{\Gamma(\epsilon, \epsilon)}$ distribution for some **small** $\epsilon > 0$ like 0.001: `tau.theta` $\sim$ `dgamma( 1.0E-3, 1.0E-3 )`

`WinBUGS` Aspirin Analysis (continued)

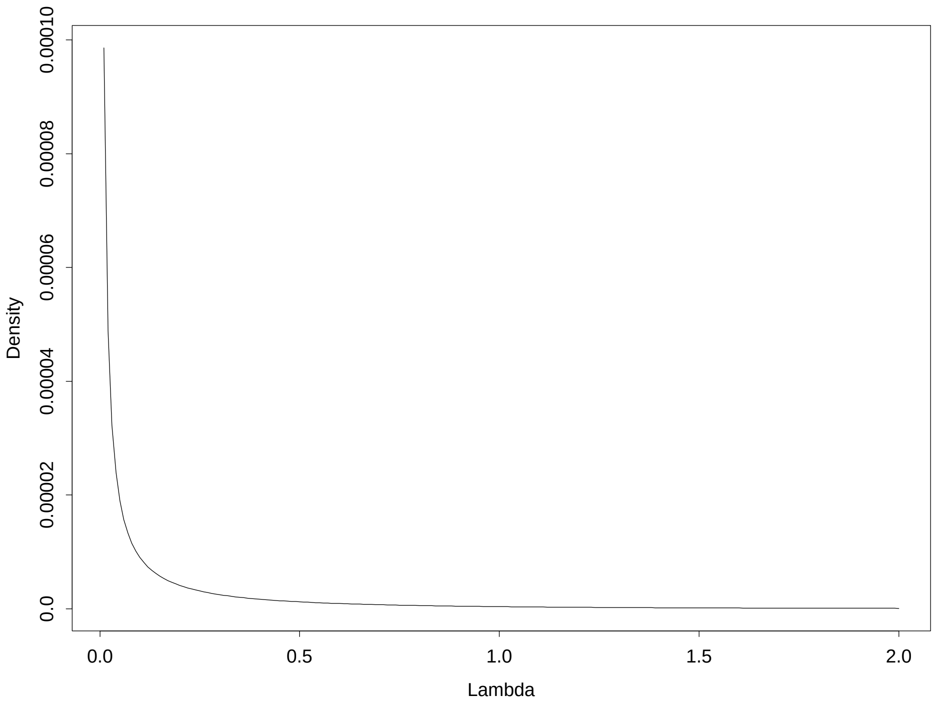


Figure 3.1. The $\Gamma(0.001, 0.001)$ distribution.

The **data** file in the aspirin meta-analysis is

```
list( k = 6, y = c( 2.77, 2.50, 1.84, 2.56, 2.31, -1.15 ),
  tau.y = c( 0.3673, 0.5827, 0.1826, 0.3586, 0.2551, 1.235 ) )
```

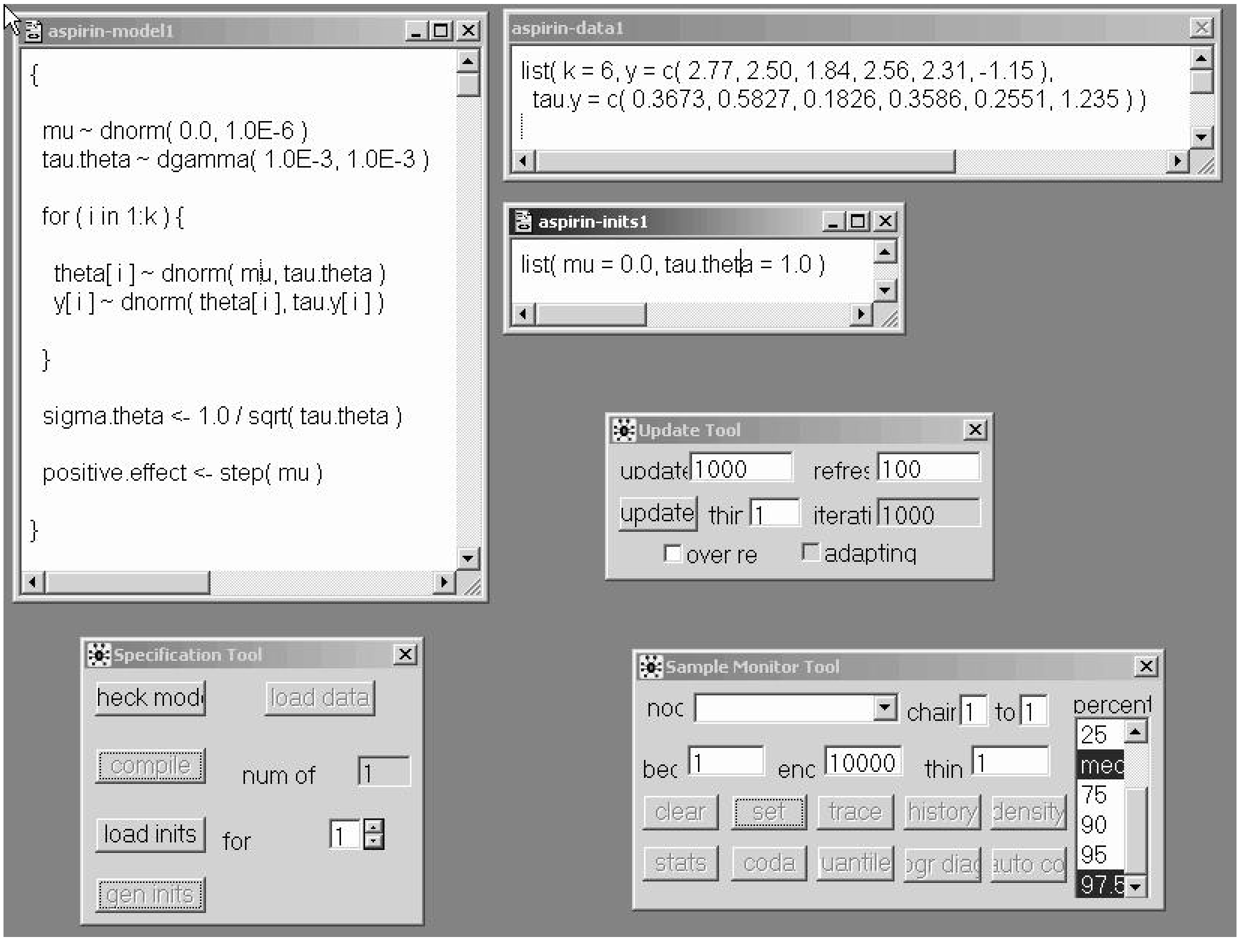
Here, e.g., `tau.y[ 1 ]` $= \frac{1}{1.65^2} \doteq 0.3673$, where 1.65 is the **standard error** of the difference `y[ 1 ]` for experiment 1 in Table 2.1 on p. 20.

Finally, the **initial values** file in the aspirin meta-analysis is

```
list( mu = 0.0, tau.theta = 1.0 )
```

In a simple example like this there's no harm in starting the Markov chain off in a **generic** location: since μ and τ_θ live on $(-\infty, \infty)$ and $(0, \infty)$, convenient generic choices for their starting values are 0 and 1, respectively (more care may be required in models with **more complex random-effects structure**).

`WinBUGS` **Aspirin Analysis (continued)**



aspirin-model1

```
{

  mu ~ dnorm( 0.0, 1.0E-6 )
  tau.theta ~ dgamma( 1.0E-3, 1.0E-3 )

  for ( i in 1:k ) {

    theta[ i ] ~ dnorm( mu, tau.theta )
    y[ i ] ~ dnorm( theta[ i ], tau.y[ i ] )

  }

  sigma.theta <- 1.0 / sqrt( tau.theta )

  positive.effect <- step( mu )

}
```

aspirin-data1

```
list( k = 6, y = c( 2.77, 2.50, 1.84, 2.56, 2.31, -1.15 ),
  tau.y = c( 0.3673, 0.5827, 0.1826, 0.3586, 0.2551, 1.235 ) )
```

aspirin-inits1

```
list( mu = 0.0, tau.theta = 1.0 )
```

Update Tool

updat 1000 refres 100
thin 1 iterati 1000
□ over re □ adapting

Specification Tool

num of 1
for 1

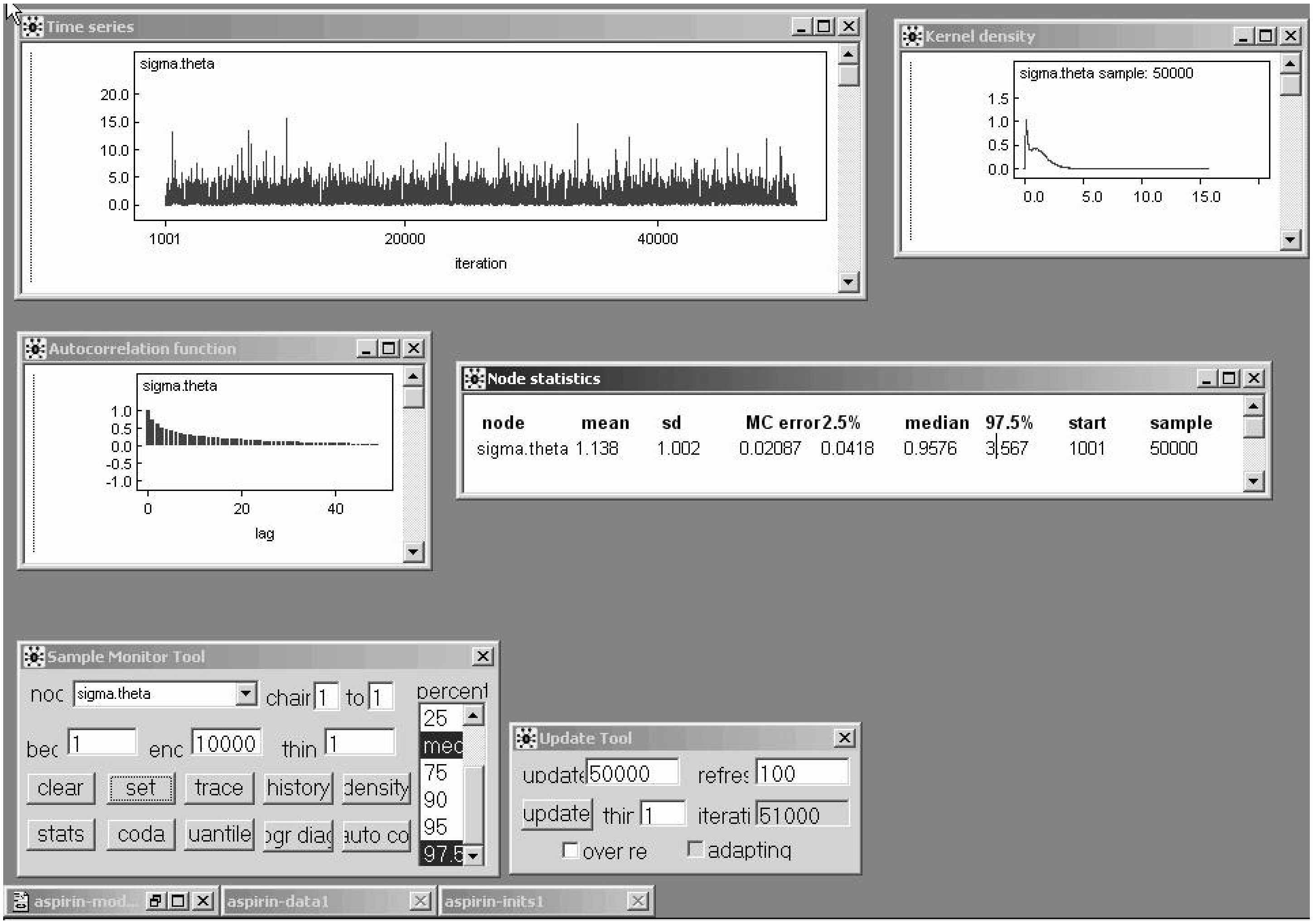
Sample Monitor Tool

noc
chain 1 to 1
beg 1 end 10000 thin 1
percent 25 med 75 90 95 97.5

I (1) get a `Specification Tool` from the `Model` menu, (2) click on the **model** window and click `check model`, (3) click on the **data** window and click `load data` and `compile`, (4) click on the **initial values** window and click `load inits`, and (5) click `gen inits` (because the random effects θ_i were uninitialized in the inits file); I'm now ready to do some MCMC sampling.

I (6) get an `Update Tool` from the `Model` menu, and click `update` to perform a **burn-in** of 1,000 iterations (the default), which takes **0s** at 1.6 Pentium GHz; (7) I then get a `Sample Monitoring Tool` from the `Inference` menu, and type `sigma.theta` and click `set`.

WinBUGS **Aspirin Analysis (continued)**



node	mean	sd	MC error	2.5%	median	97.5%	start	sample
sigma.theta	1.138	1.002	0.02087	0.0418	0.9576	3.567	1001	50000

noc sigma.theta chair 1 to 1

bec 1 enc 10000 thin 1

update 50000 refres 100

thir 1 iterati 51000

(8) I type 50000 in the updates window in the Update Tool and click update to get a **monitoring** run of **50,000** iterations (this took **15s**).

Then (9) selecting sigma.theta in the node window, all 10 buttons from clear through autoC are active, and I click on history (to get a Time Series window), density (to get a Kernel density window), autoC to get an Autocorrelation function window, and stats (to get a Node statistics window), **yielding the screen above**.

The output of an MCMC sampler, when considered as a **time series**, often exhibits **positive autocorrelation**; in fact it often looks like a realization of an **autoregressive** AR_p model of order $p = \mathbf{1}$ $(\theta_t = \alpha + \beta\theta_{t-1} + e_t)$ with **positive first-order autocorrelation** ρ.

WinBUGS **Aspirin Analysis (continued)**

This does not affect the **validity** of Monte Carlo inferences about the unknowns (e.g., the mean of any **stationary stochastic process** is a **consistent** estimator of the underlying process mean), but it does affect the **efficiency** of these inferences: for example, the Monte Carlo variance of the sample mean $\bar{\theta}$ based on M draws from an AR_1 time series is

$$V(\bar{\theta}) = \frac{\sigma_\theta^2}{M} \left(\frac{1+\rho}{1-\rho} \right), \tag{9}$$

and the **sample size inflation factor** $\frac{1+\rho}{1-\rho} \rightarrow \infty$ as $\rho \rightarrow +1$.

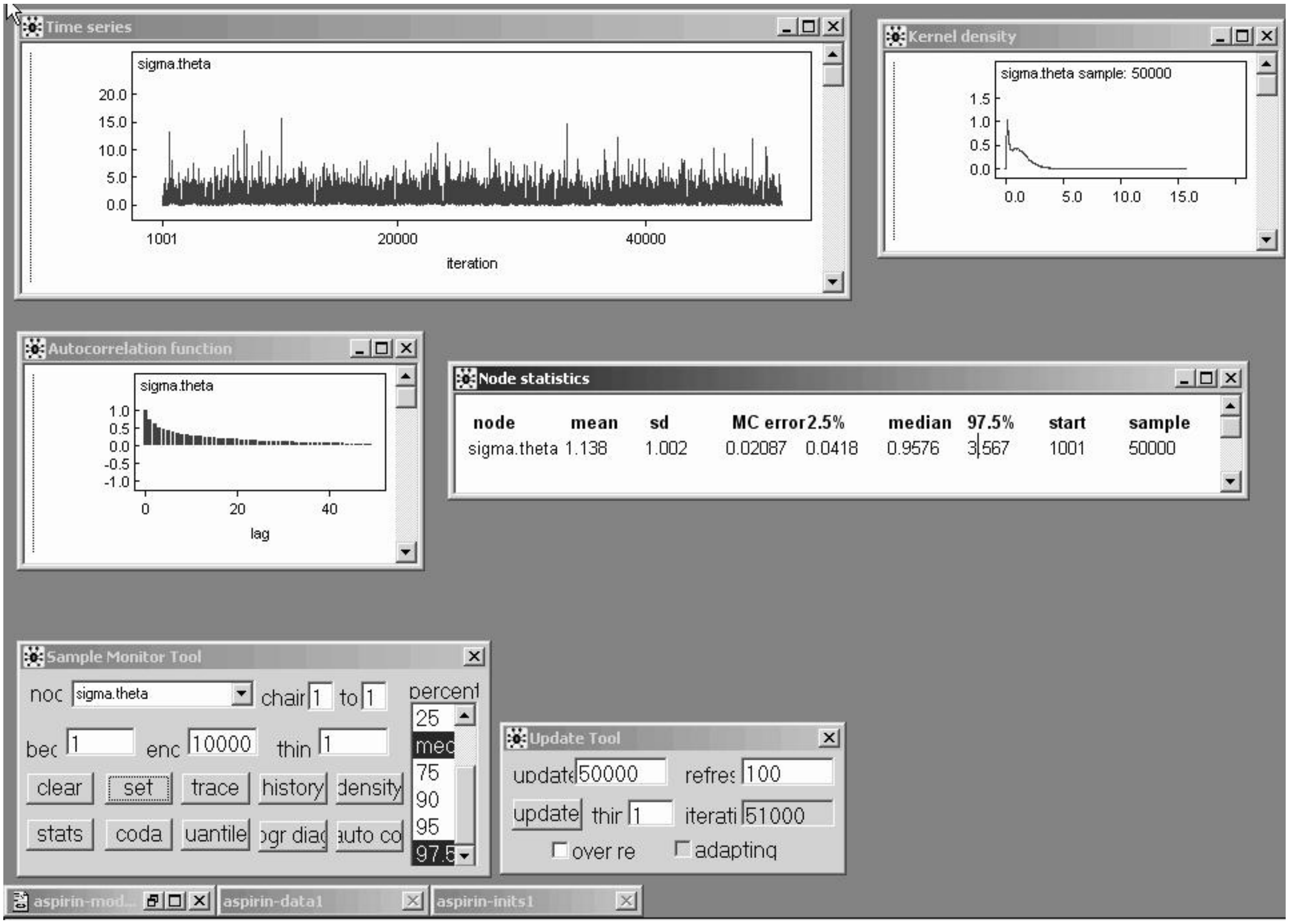
An MCMC sampler which produces output for any given unknown θ with ρ near 0 (if $\rho = 0$ the output is white noise, i.e., equivalent to IID draws from the posterior) is said to be **mixing well** in that unknown.

The time series trace for σ_θ above is only mixing **moderately well**: the autocorrelation function has the familiar ski-slope shape of an AR_1 series with $\rho \doteq 0.7$ (the height of the bar at lag 1).

The **marginal posterior distribution** for σ_θ (from the `Kernel density` window) looks heavily skewed to the right, which makes sense for a scale parameter.

The **posterior mean** and **SD** of σ_θ (using the $\Gamma(\epsilon, \epsilon)$ prior for τ_θ) are estimated to be 1.14 and 1.00, respectively; the **Monte Carlo standard error** of the posterior mean estimate is 0.021 (so that with 50,000 monitoring iterations I don't yet have **3 significant figures** of accuracy for the posterior mean); the **posterior median** is estimated to be 0.96; and a **95% central interval** for σ_θ with this prior is estimated to run from 0.042 to 3.57.

WinBUGS Aspirin Analysis (continued)



```
node         mean    sd     MC error 2.5%    median  97.5%  start  sample
sigma.theta  1.138   1.002  0.02087  0.0418  0.9576  3.567  1001   50000
```

```
noc sigma.theta   chair 1  to 1   percent
bec 1   enc 10000   thin 1        25  med  75  90  95  97.5
```

```
updat 50000   refre 100
thir 1        iterati 51000
□ over re     □ adapting
```

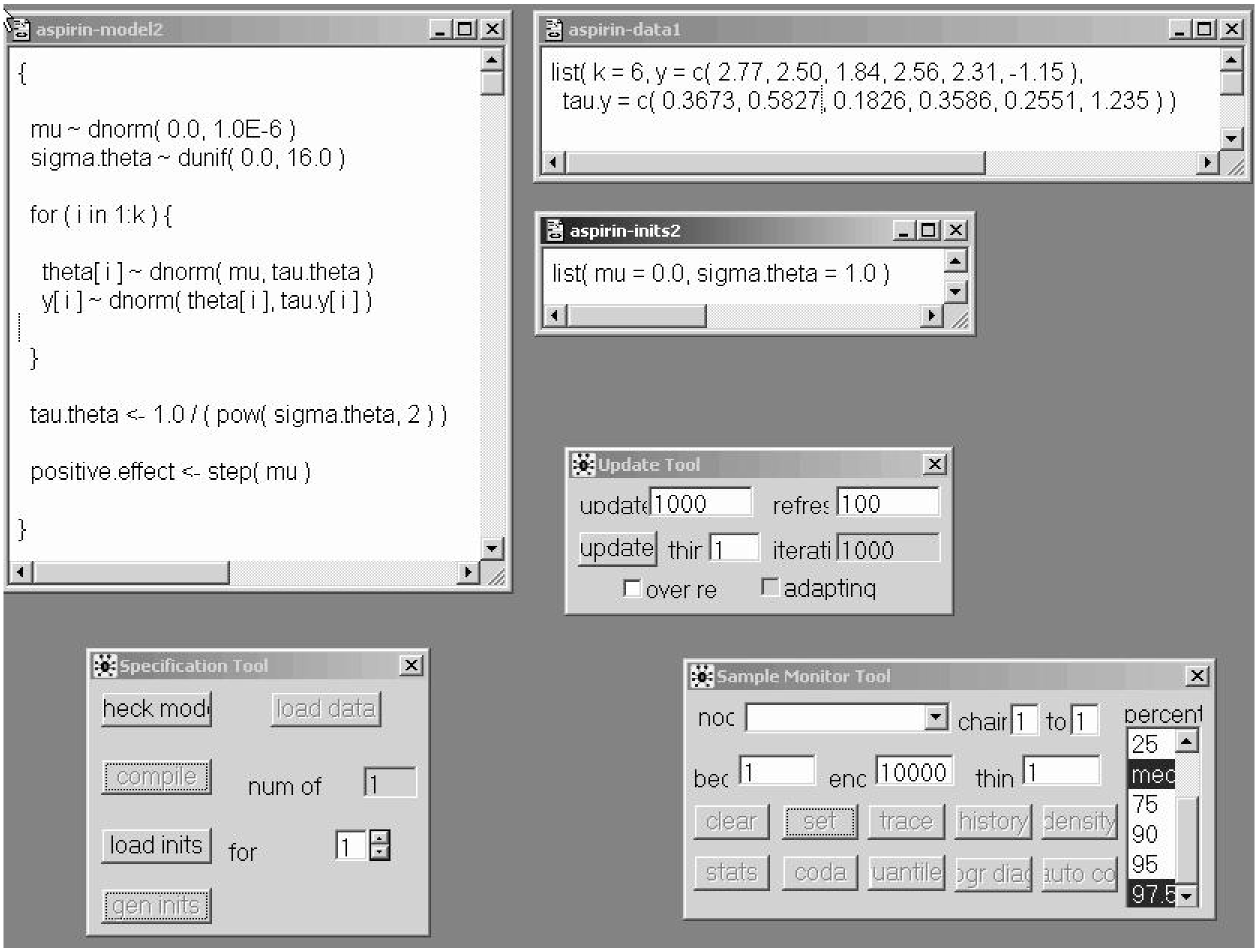
The main thing to notice, however, is that the **range of plausible values** for `sigma.theta` in its posterior is approximately from **0 to 16**.

It has recently been shown that the **simplest diffuse prior** on σ_θ that has **good calibration properties** (i.e., such that 95% **nominal** Bayesian interval estimates for all of the parameters in model (10) do in fact have **actual coverage close to 95%**) is

$$\sigma_\theta \sim U(0, c), \qquad (10)$$

where c is chosen to be (roughly) the **smallest value that doesn't truncate the likelihood function** for σ_θ; here it's evident that $c \doteq 16$ will **work well**.

WinBUGS Aspirin Analysis (continued)



aspirin-model2

```
{

  mu ~ dnorm( 0.0, 1.0E-6 )
  sigma.theta ~ dunif( 0.0, 16.0 )

  for ( i in 1:k ) {

    theta[ i ] ~ dnorm( mu, tau.theta )
    y[ i ] ~ dnorm( theta[ i ], tau.y[ i ] )

  }

  tau.theta <- 1.0 / ( pow( sigma.theta, 2 ) )

  positive.effect <- step( mu )

}
```

aspirin-data1

```
list( k = 6, y = c( 2.77, 2.50, 1.84, 2.56, 2.31, -1.15 ),
  tau.y = c( 0.3673, 0.5827, 0.1826, 0.3586, 0.2551, 1.235 ) )
```

aspirin-inits2

```
list( mu = 0.0, sigma.theta = 1.0 )
```

Update Tool: updat 1000, refres 100, thir 1, iterati 1000, □ over re, □ adapting

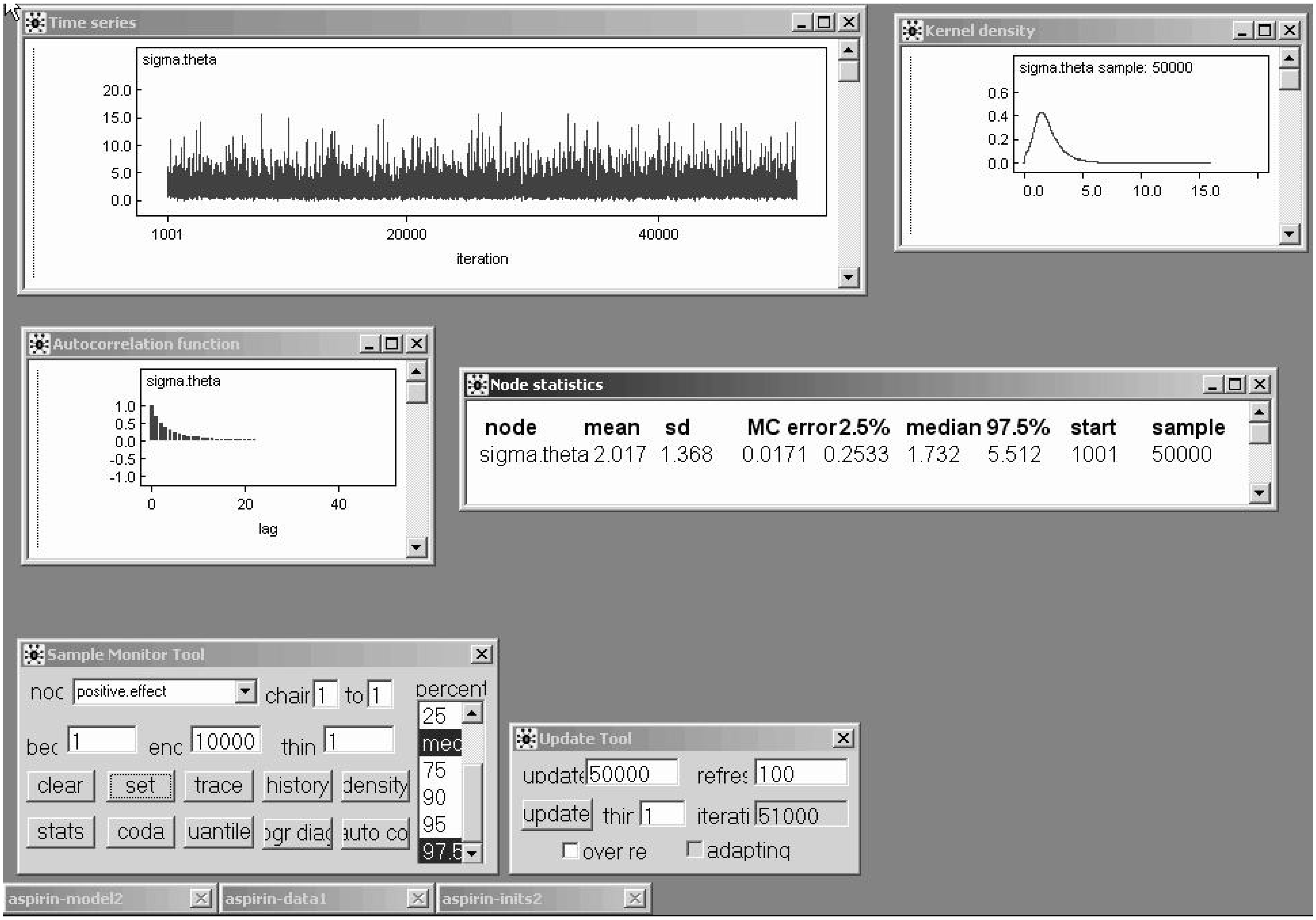
Specification Tool: num of 1, for 1

Sample Monitor Tool: noc, chair 1 to 1, bec 1, enc 10000, thin 1, percent 25, mec, 75, 90, 95, 97.5

So I estimate a **second model** placing a Uniform$(0, c)$ prior on σ_θ (this model also requires a **new initial values file** that initializes `sigma.theta` instead of `tau.theta`).

This time in the `Sample Monitor Tool` I `set` all of the **interesting** quantities: `mu`, `sigma.theta`, `theta`, and `positive.effect`, and I use the same MCMC strategy as before (a **burn-in of 1,000** followed by a **monitoring run of 50,000**).

WinBUGS Aspirin Analysis (continued)



Node statistics

```
 node        mean   sd     MC error 2.5%    median 97.5%  start  sample
 sigma.theta 2.017  1.368  0.0171   0.2533  1.732  5.512  1001   50000
```

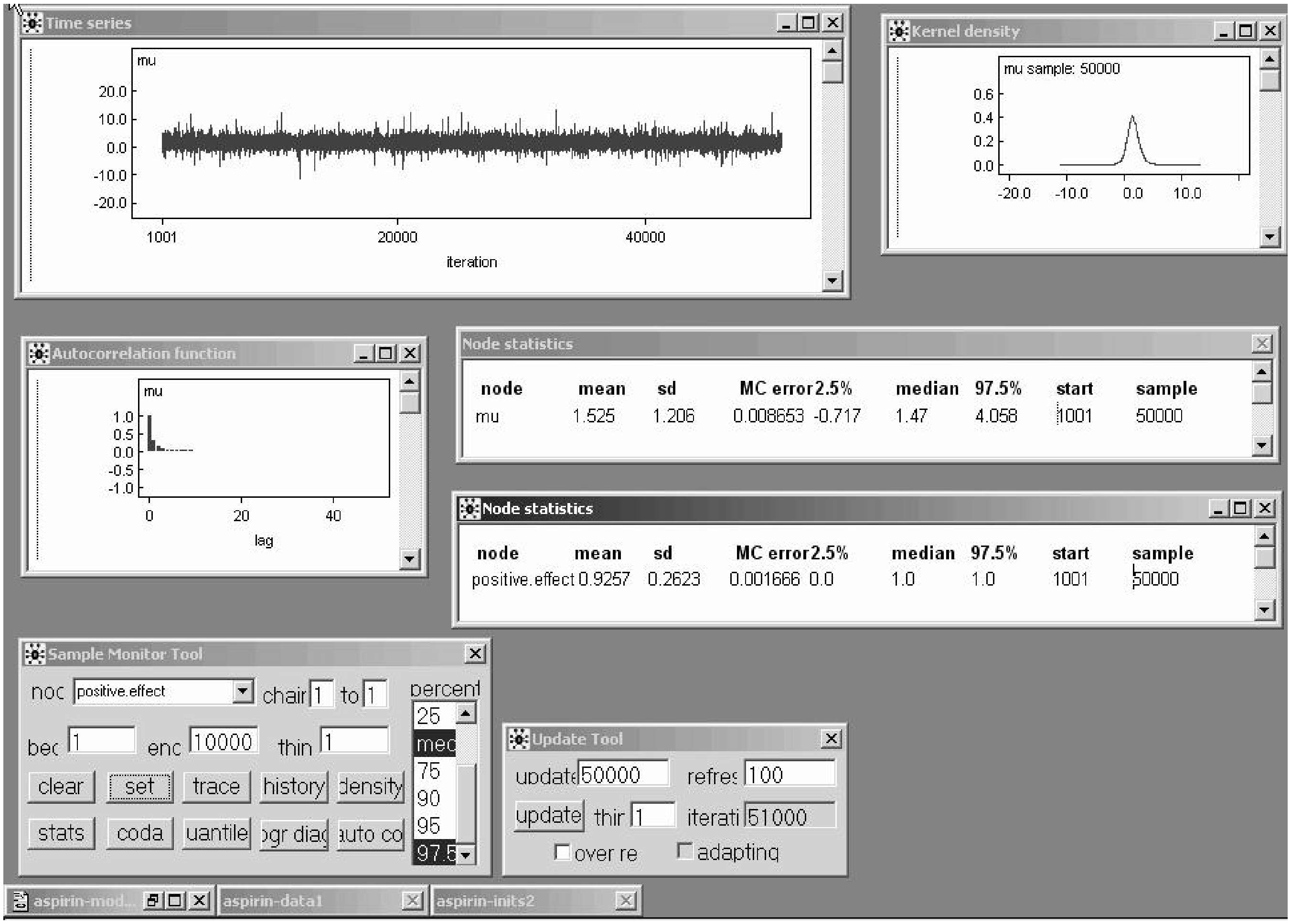
```
noc positive.effect   chair 1   to 1
bec 1   enc 10000   thin 1
```

```
update 50000   refres 100
thir 1   iterati 51000
```

With the Uniform$(0, c)$ prior on σ_θ the posterior mean of σ_θ is now **sharply higher** than before (**2.02** versus the **1.14** value I got with the initial $\Gamma(\epsilon, \epsilon)$ prior (this sort of **discrepancy** will only arise when the number of studies k is **small**; when it does arise I **trust** the results from the Uniform$(0, c)$ prior).

Note that the posterior mean of σ_θ is also **quite a bit bigger** than the value (**1.24**) obtained from **MLEB** back on page 25—this is a reflection of the **tendency of MLEB to understate the between-study heterogeneity** in model (10) with small k.

WinBUGS Aspirin Analysis (continued)



Node statistics

```
node     mean    sd      MC error 2.5%    median  97.5%  start  sample
mu       1.525   1.206   0.008653 -0.717  1.47    4.058  1001   50000
```

Node statistics

```
node              mean    sd      MC error 2.5%   median  97.5%  start  sample
positive.effect   0.9257  0.2623  0.001666 0.0    1.0     1.0    1001   50000
```

Sample Monitor Tool: noc positive.effect, chain 1 to 1, bec 1, enc 10000, thin 1, percent 25, med, 75, 90, 95, 97.5

Update Tool: update 50000, refres 100, thin 1, iterati 51000, □ over re, □ adapting

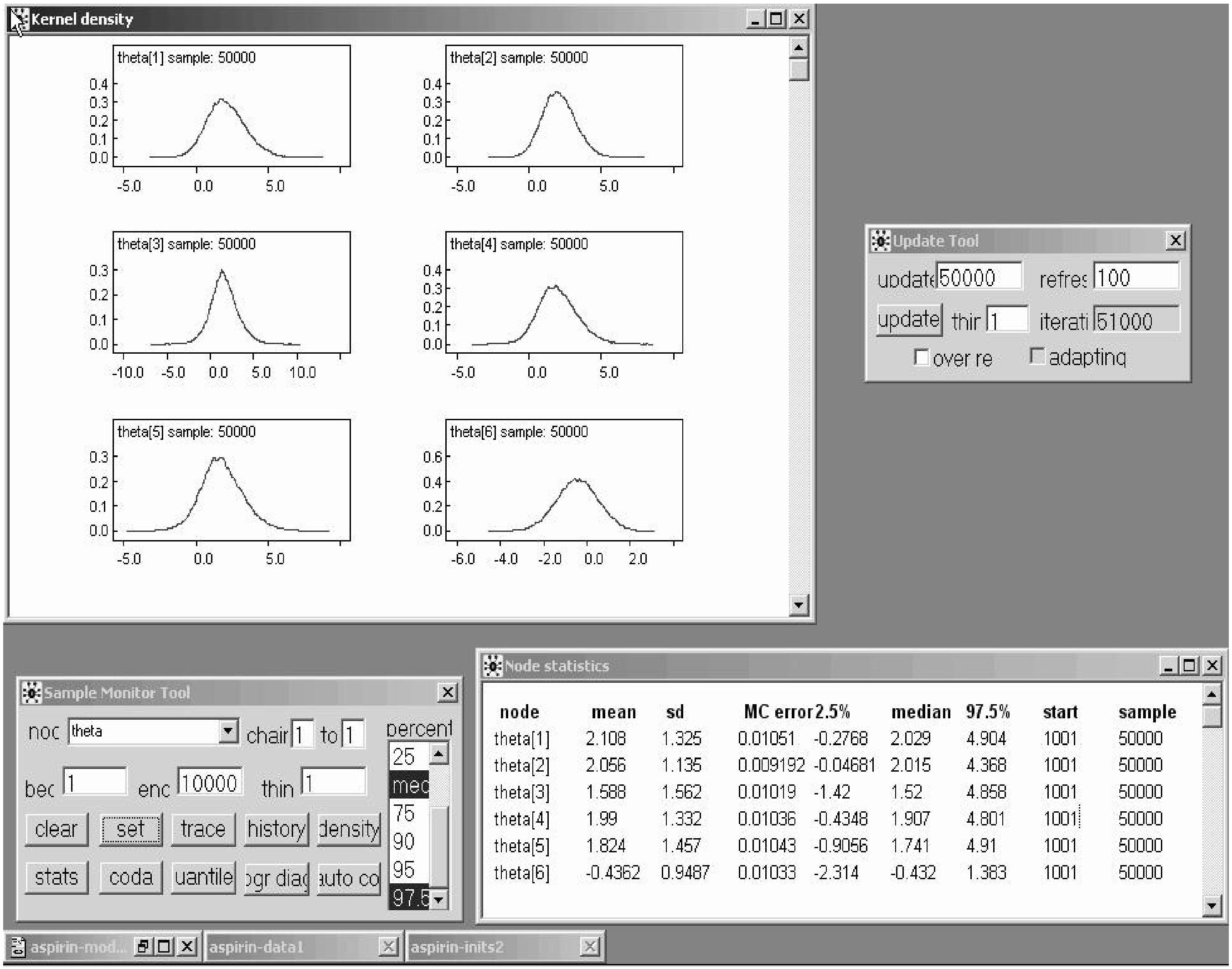
On pp. 25–26 above we saw that the MLEB estimate of μ was **1.45** with an approximate standard error of **0.809**, and an approximate 95% confidence interval for μ ran from -0.14 to $+3.03$.

The corresponding **Bayesian** results are: posterior **mean 1.52**, posterior **SD 1.21**, 95% **interval (–0.72, 4.06)**.

As is often true, the simple MLEB approximations leading to these estimates have **underestimated the actual uncertainty** about μ: the Bayesian 95% interval with the Uniform prior is **50% wider**.

It's easy to monitor the **posterior probability that aspirin is beneficial**, with the built-in `step` function applied to `mu`: $P(\mu > 0 | \text{data, diffuse prior information}) \doteq$ **0.93**, i.e., posterior betting odds of about **12.5 to 1 that aspirin reduces mortality**.

WinBUGS Aspirin Analysis (continued)



updat 50000 refres 100

thin 1 iterati 51000

noc theta chain 1 to 1

bec 1 enc 10000 thin 1

```
node      mean     sd      MC error 2.5%      median  97.5%  start  sample
theta[1]  2.108    1.325   0.01051  -0.2768   2.029   4.904  1001   50000
theta[2]  2.056    1.135   0.009192 -0.04681  2.015   4.368  1001   50000
theta[3]  1.588    1.562   0.01019  -1.42     1.52    4.858  1001   50000
theta[4]  1.99     1.332   0.01036  -0.4348   1.907   4.801  1001   50000
theta[5]  1.824    1.457   0.01043  -0.9056   1.741   4.91   1001   50000
theta[6]  -0.4362  0.9487  0.01033  -2.314    -0.432  1.383  1001   50000
```

The marginal density plots of the θ_i values show **interesting departures from normality**, and the Bayesian estimates (a) exhibit **rather less shrinkage** and (b) have **27–43% larger** uncertainty estimates.

Table 3.1. MLEB and Bayesian (posterior mean) estimates of the θ_i.

	Maximum Likelihood		Bayesian Posterior	
study(i)	$\widehat{\theta}_i$	$\widehat{SE}(\widehat{\theta}_i)$	mean	SD
1	1.92	0.990	2.11	1.33
2	1.94	0.899	2.06	1.14
3	1.53	1.09	1.59	1.56
4	1.84	0.994	1.99	1.33
5	1.69	1.05	1.82	1.46
6	-0.252	0.728	-0.44	0.95

Hierarchical Model Expansion

Looking at the **shrinkage plot** on p. 26 or the **raw data values** themselves, it's evident that a **Gaussian** model for the θ_i may not be appropriate: study 6 is so different than the other 5 that a **heavier-tailed distribution** may be a better choice.

This suggests **expanding** the HM (10), by embedding it in a **richer model class** of which it's a **special case** (this is the main Bayesian approach in practice to **dealing with model inadequacies**).

A **natural choice** would be a t model for the θ_i with **unknown degrees of freedom** ν:

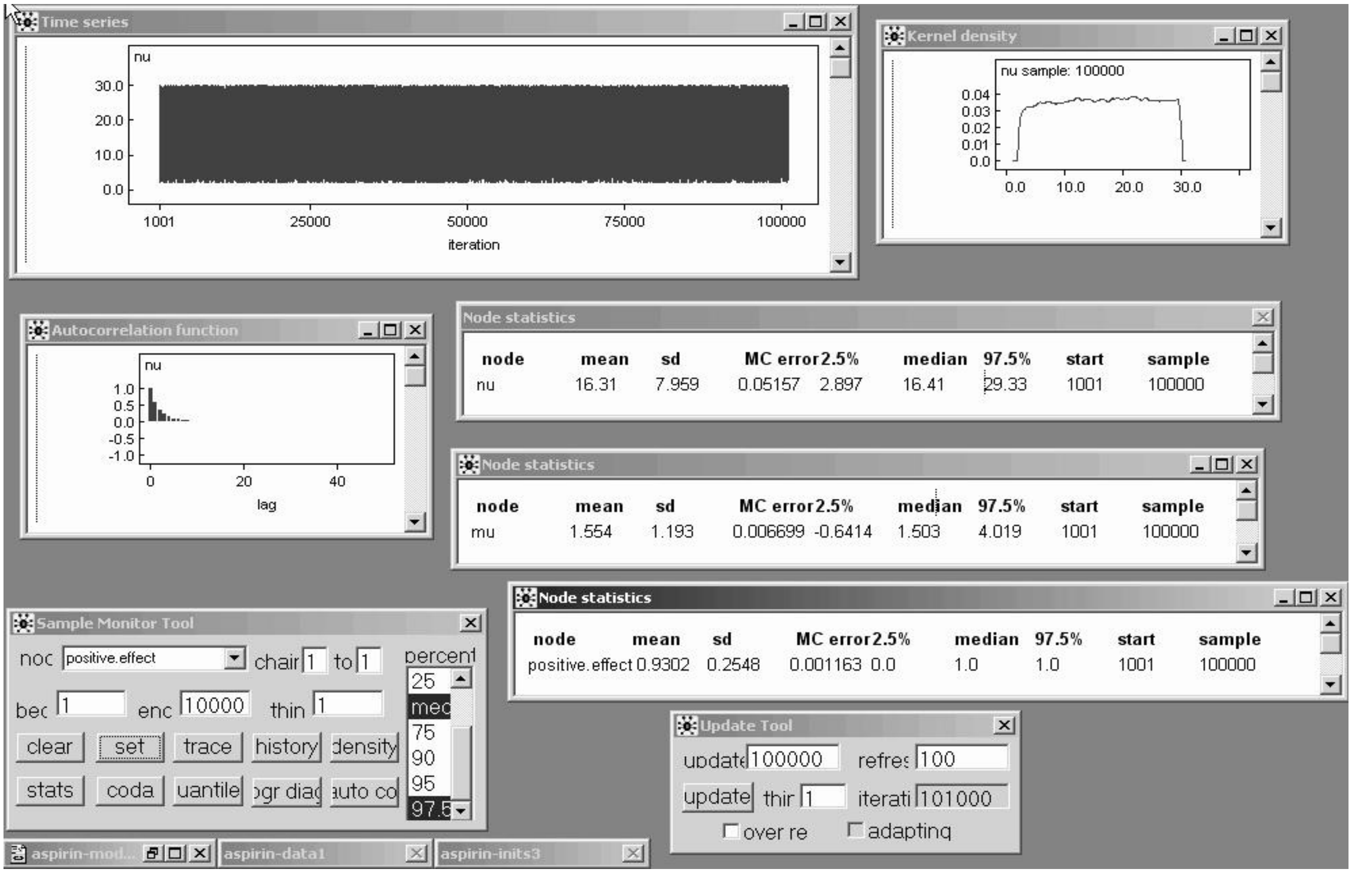
$$\begin{array}{rcll} (\theta, \sigma^2, \nu) & \sim & p(\theta, \sigma^2, \nu) & \text{(prior)} \\ (\theta_i | \theta, \sigma^2, \nu) & \stackrel{\text{IID}}{\sim} & t\big(\theta, \sigma^2, \nu\big) & \text{(underlying effects)} \\ (y_i | \theta_i) & \stackrel{\text{indep}}{\sim} & N(\theta_i, V_i) & \text{(data)} . \end{array} \qquad (11)$$

Here $\eta \sim t\big(\theta, \sigma^2, \nu\big)$ just means that $\left(\frac{\eta - \theta}{\sigma} \right)$ follows a **standard** t **distribution** with ν degrees of freedom. This is **amazingly easy** to implement in `WinBUGS` (it is considerably more difficult to carry out an **analogous ML analysis**).

The new model file is

```
{

  mu ~ dnorm( 0.0, 1.0E-6 )
  sigma.theta ~ dunif( 0.0, 16.0 )
  nu ~ dunif( 3.0, 30.0 )

  for ( i in 1:k ) {

    theta[ i ] ~ dt( mu, tau.theta, nu )
    y[ i ] ~ dnorm( theta[ i ], tau.y[ i ] )

  }

  tau.theta <- 1.0 / pow( sigma.theta, 2 )

}
```

Model Expansion (continued)



```
node      mean    sd      MC error 2.5%     median  97.5%  start  sample
nu        16.31   7.959   0.05157  2.897    16.41   29.33  1001   100000
```

```
node      mean    sd      MC error 2.5%     median  97.5%  start  sample
mu        1.554   1.193   0.006699 -0.6414  1.503   4.019  1001   100000
```

```
node             mean    sd      MC error 2.5%  median  97.5%  start  sample
positive.effect  0.9302  0.2548  0.001163 0.0   1.0     1.0    1001   100000
```

noc positive.effect chain 1 to 1
bec 1 enc 10000 thin 1

update 100000 refres 100
thir 1 iterati 101000

To express comparative prior ignorance about ν I use a **uniform** prior on the interval from 2.0 to 30.0 (below $\nu = 2$ the t distribution has **infinite variance**, and above about 30 it starts to be **indistinguishable** in practice from the Gaussian).

A **burn-in** of 1,000 and a **monitoring run** of 100,000 iterations takes **about twice as long as with 50,000 iterations in the Gaussian model** (i.e., about the **same speed per iteration**) and yields the **posterior summaries** above.

It's clear that there's **little information in the likelihood function** about ν: the prior and posterior for this parameter **virtually coincide**.

The results for μ and the θ_i are **almost unchanged**; this would not necessarily be the case if study 6 had been **more extreme**.

5.3 Hierarchical Model Selection: A Case Study

Case Study: *In-home geriatric assessment (IHGA)*. In an experiment conducted in the 1980s (Hendriksen et al. 1984), 572 elderly people living in a number of villages in Denmark were randomized, 287 to a **control** (C) group (who received standard care) and 285 to an **experimental** (E) group (who received standard care plus IHGA: a kind of **preventive medicine** in which each person's medical and social needs were assessed and acted upon individually).

One important outcome was the number of **hospitalizations** during the two-year life of the study (Table 6.1).

Table 6.1. Distribution of number of hospitalizations in the IHGA study over a two-year period.

	Number of Hospitalizations										
Group	0	1	2	3	4	5	6	7	n	Mean	SD
Control	138	77	46	12	8	4	0	2	287	0.944	1.24
Experimental	147	83	37	13	3	1	1	0	285	0.768	1.01

Evidently IHGA lowered the mean hospitalization rate (for these elderly Danish people, at least) by $(0.944 - 0.768) = \mathbf{0.176}$, which is about a $100\left(\frac{0.768-0.944}{0.944}\right) = \mathbf{19\%}$ reduction from the control level, a difference that's **large in clinical terms**.

Modeling the IHGA Data

An **off-the-shelf** analysis of this experiment might pretend (**Model 0**) that the data are Gaussian,

$$\begin{array}{rcl} \left(C_i | \mu_C, \sigma_C^2 \right) & \overset{\text{IID}}{\sim} & N\left(\mu_C, \sigma_C^2 \right), i = 1, \ldots, n_C, \\ \left(E_j | \mu_E, \sigma_E^2 \right) & \overset{\text{IID}}{\sim} & N\left(\mu_E, \sigma_E^2 \right), j = 1, \ldots, n_E, \end{array} \quad (12)$$

and use the ordinary frequentist **two-independent-samples "z-machinery"**:

```
rosalind 15> R

R : Copyright 2001, The R Development Core Team
Version 1.2.1  (2001-01-15)

> C <- c( rep( 0, 138 ), rep( 1, 77 ), rep( 2, 46 ),
    rep( 3, 12 ), rep( 4, 8 ), rep( 5, 4 ), rep( 7, 2 ) )

> print( n.C <- length( C ) )

[1] 287                # sample size in the control group

> mean( C )

[1] 0.9442509          # control group mean

> sqrt( var( C ) )

[1] 1.239089           # control group
                       # standard deviation (SD)
> table( C )

   0  1  2  3 4 5 7    # control group
 138 77 46 12 8 4 2    # frequency distribution
```

Analysis of Model 0

```
> E <- c( rep( 0, 147 ), rep( 1, 83 ), rep( 2, 37 ),
    rep( 3, 13 ),rep( 4, 3 ), rep( 5, 1 ), rep( 6, 1 ) )

> print( n.E <- length( E ) )

[1] 285                 # sample size in the
                        # experimental group
> mean( E )

[1] 0.7684211           # experimental group mean

> sqrt( var( E ) )

[1] 1.008268            # experimental group SD

> table( E )

   0  1  2  3 4 5 6     # experimental group
 147 83 37 13 3 1 1     # frequency distribution

> print( effect <- mean( E ) - mean( C ) )

[1] -0.1758298          # mean difference ( E - C )

> effect / mean( C )

[1] -0.1862109          # relative difference ( E - C ) / C

> SE.effect <- sqrt( var( C ) / n.C + var( E ) / n.E )

[1] 0.09442807          # standard error of the difference

> print( CI <- c( effect - 1.96 * SE.effect,
    effect + 1.96 * SE.effect ) )

[1] -0.3609 0.009249   # the 95% confidence interval from
                       # model 0 runs from -.36 to +.01
```

Deficiencies of Model 0

The frequentist analysis of Model 0 is equivalent to a Bayesian analysis of the same model with **diffuse priors** on the control and experimental group means and SDs $(\mu_C, \sigma_C, \mu_E, \sigma_E)$, and is summarized in Table 6.2.

Table 6.2. Summary of analysis of Model 0.

	Posterior		
	Mean	SD	95% Interval
Treatment effect $(\mu_E - \mu_C)$	-0.176	0.0944	$(-0.361, 0.009)$

However, both distributions have long right-hand tails; in fact they look rather **Poisson**.

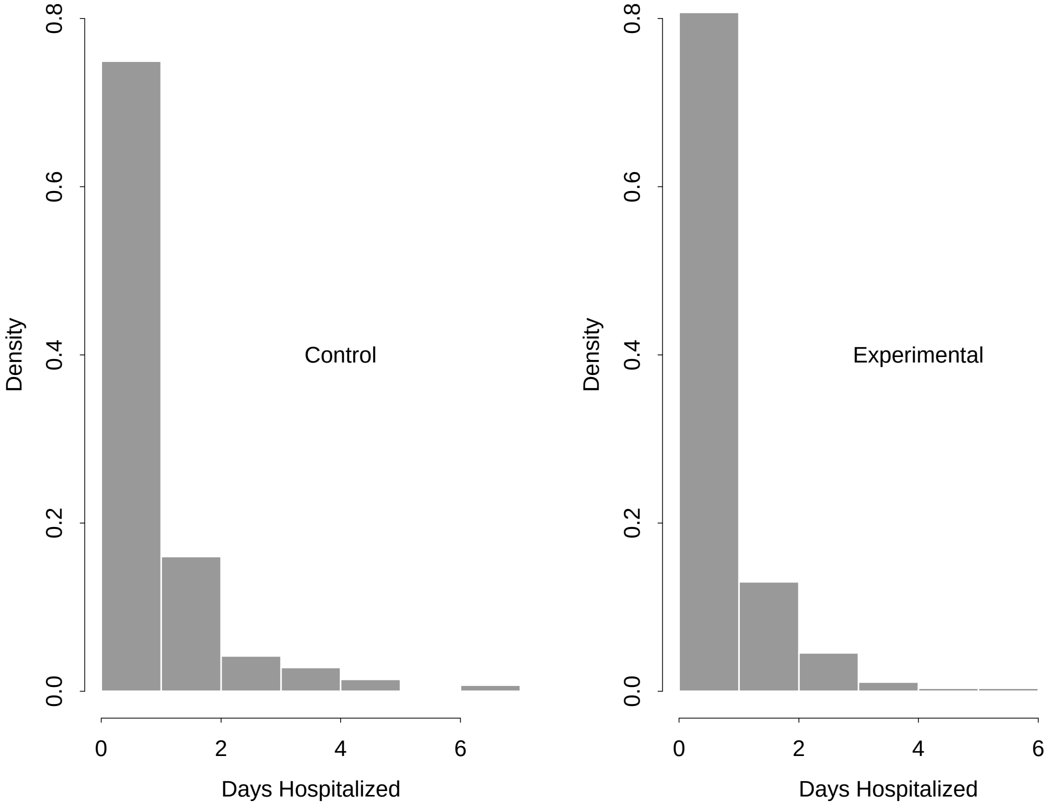


Figure 6.1. Histograms of control and experimental numbers of hospitalizations.

5.4 Poisson Fixed-Effects Modeling

R code to make the histograms:

```
> x11( )                             # to open a
                                     #   graphics window
> par( mfrow = c( 1, 2 ) )           # to plot two histograms

> hist( C, nclass = 8, probability = T,
    xlab = ’Days Hospitalized’, ylab = ’Density’,
    xlim = c( 0, 7 ), ylim = c( 0, 0.8 ) )

> text( 4, 0.4, ’Control’ )

> hist( E, nclass = 8, probability = T,
    xlab = ’Days Hospitalized’, ylab = ’Density’,
    xlim = c( 0, 7 ), ylim = c( 0, 0.8 ) )

> text( 4, 0.4, ’Experimental’ )
```

So I **created** a classicBUGS file called poisson1.bug that looked like this:

```
model poisson1;

const

  n.C = 287, n.E = 285;

var

  lambda.C, lambda.E, C[ n.C ], E[ n.E ], effect;

data C in "poisson-C.dat", E in "poisson-E.dat";

inits in "poisson1.in";
```

Initial Poisson Modeling (continued)

```
{

  lambda.C ~ dgamma( 0.001, 0.001 );
  lambda.E ~ dgamma( 0.001, 0.001 );

  for ( i in 1:n.C ) {

    C[ i ] ~ dpois( lambda.C );

  }

  for ( j in 1:n.E ) {

    E[ j ] ~ dpois( lambda.E );

  }

  effect <- lambda.E - lambda.C;

}
```

`poisson1.in` initializes both λ_C and λ_E to 1.0; the $\Gamma(0.001, 0.001)$ priors for λ_C and λ_E are chosen (as usual to create diffuseness) to be **flat** in the region in which the **likelihood** is **appreciable**:

```
> sqrt( var( C ) / n.C )

[1] 0.07314114

> sqrt( var( E ) / n.E )

[1] 0.05972466

> c( mean( C ) - 3.0 * sqrt( var( C ) / n.C ),
    mean( C ) + 3.0 * sqrt( var( C ) / n.C ) )
```

Initial Poisson Modeling (continued)

```
[1] 0.7248275 1.1636743

> c( mean( E ) - 3.0 * sqrt( var( E ) / n.E ),
    mean( E ) + 3.0 * sqrt( var( E ) / n.E ) )

[1] 0.5892471 0.9475950

> lambda.grid <- seq( 0.01, 2.0, 0.01 )

> plot( lambda.grid, 0.001 * dgamma( lambda.grid, 0.001 ),
    type = 'l', xlab = 'Lambda', ylab = 'Density' )
```

The likelihood under the Gaussian model is **concentrated** for λ_C from about 0.7 to 1.2, and that for λ_E from about 0.6 to 1; you can see from the plot that across those ranges the $\Gamma(0.001, 0.001)$ prior is **essentially constant**.

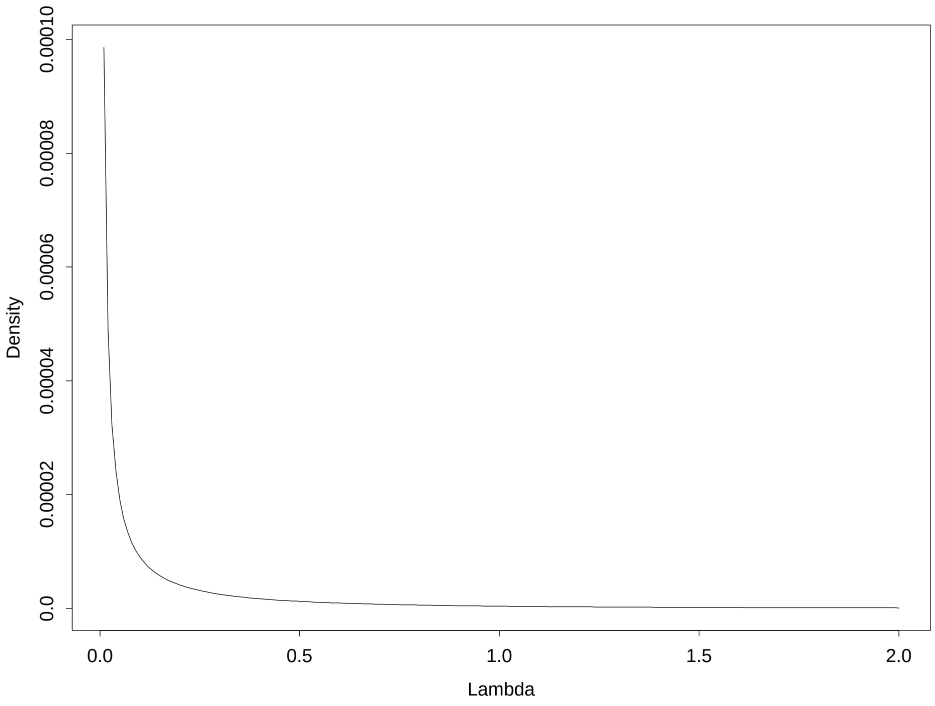
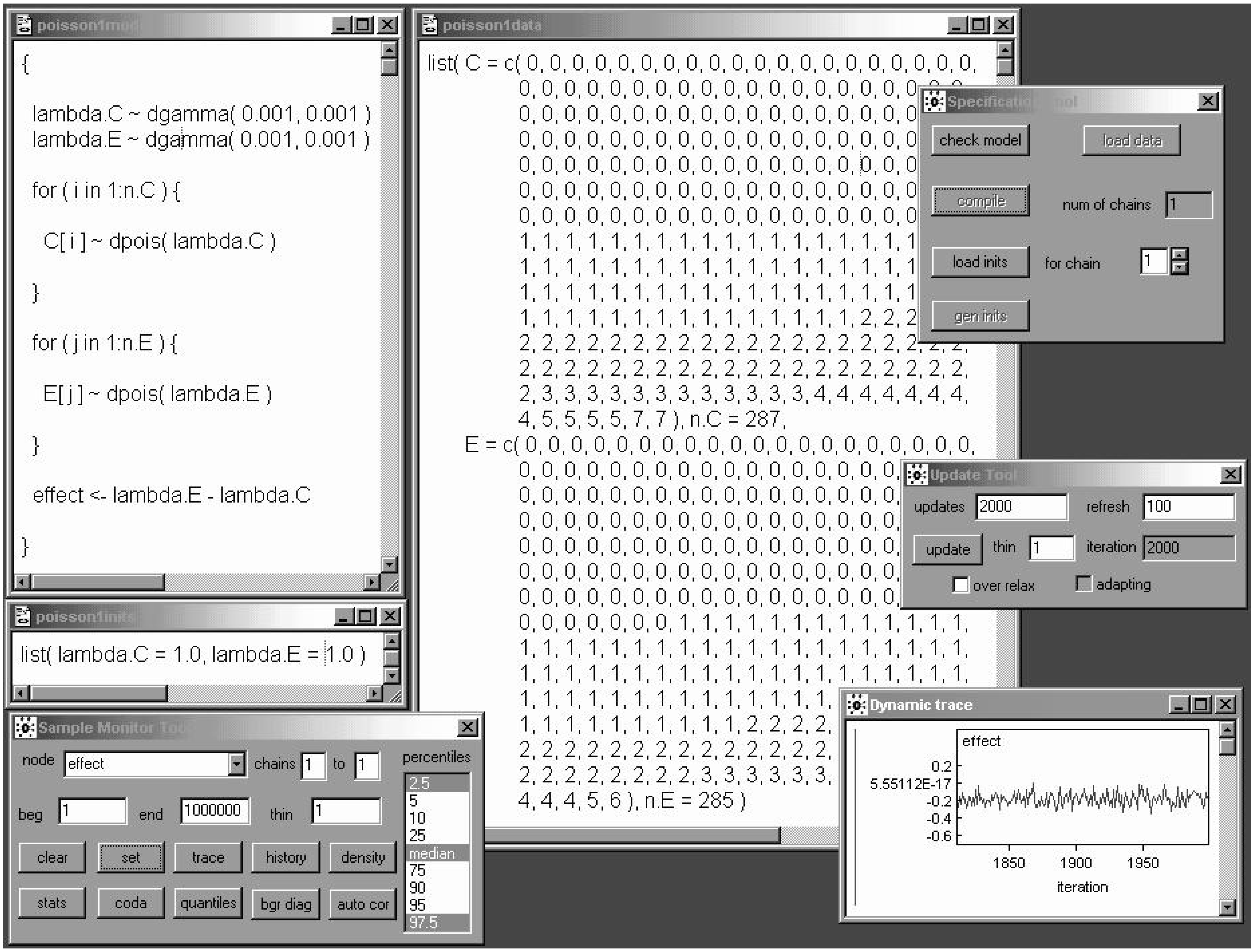


Figure 6.2. The $\Gamma(0.001, 0.001)$ distribution in the region in which the likelihoods for λ_C and λ_E are appreciable.

WinBUGS Implementation



poisson1model

```
{

 lambda.C ~ dgamma( 0.001, 0.001 )
 lambda.E ~ dgamma( 0.001, 0.001 )

 for ( i in 1:n.C ) {

  C[ i ] ~ dpois( lambda.C )

 }

 for ( j in 1:n.E ) {

  E[ j ] ~ dpois( lambda.E )

 }

 effect <- lambda.E - lambda.C

}
```

poisson1data

```
list( C = c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2
             2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
             2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
             2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4,
             4, 5, 5, 5, 5, 7, 7 ), n.C = 287,
    E = c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2,
             2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
             2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3,
             4, 4, 4, 5, 6 ), n.E = 285 )
```

poisson1inits

```
list( lambda.C = 1.0, lambda.E = 1.0 )
```

Specification Tool: num of chains 1; for chain 1

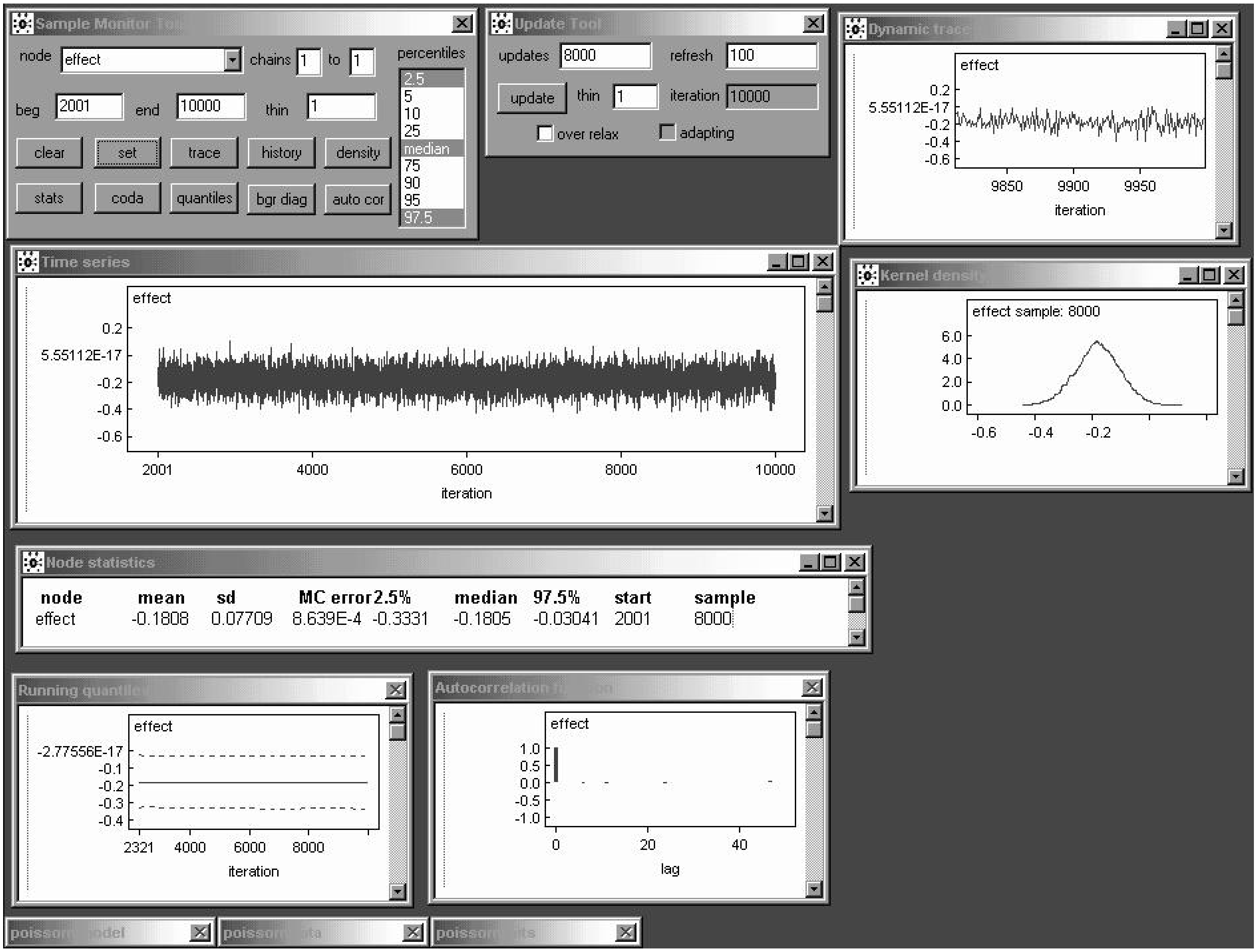
Update Tool: updates 2000; refresh 100; thin 1; iteration 2000; □ over relax; □ adapting

Sample Monitor Tool: node effect; chains 1 to 1; beg 1; end 1000000; thin 1

The **screendump** above presents part of the results of fitting the **2-independent-samples Poisson model** at the top of page 7 in WinBUGS.

A **burn-in of 2,000** was almost **instantaneous at 2.0 PC GHz** and revealed **good mixing** for the three main quantities of interest.

WinBUGS Implementation (continued)



node effect; chains 1 to 1; beg 2001; end 10000; thin 1

percentiles: 2.5, 5, 10, 25, median, 75, 90, 95, 97.5

updates 8000; refresh 100; thin 1; iteration 10000

□ over relax □ adapting

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
effect	-0.1808	0.07709	8.639E-4	-0.3331	-0.1805	-0.03041	2001	8000

A monitoring run of 8,000 reveals that the effect parameter in the **2-independent-samples Poisson model** is behaving like **white noise**, so that already with only 8,000 iterations the posterior mean has a Monte Carlo standard error of **less than 0.001**.

Initial Poisson Modeling (continued)

Thus a burn-in of 2,000 and a monitoring run of 8,000 yields **good MCMC diagnostics** and permits a comparison between model 0 (Gaussian) and model 1 (Poisson), as in Table 6.3.

Table 6.3. Comparison of inferential conclusions from models 0 and 1.

$\boxed{\lambda_C}$ Model	Posterior Mean	Posterior SD	Central 95% Interval
Gaussian	0.944	0.0731	$(0.801, 1.09)$
Poisson	0.943	0.0577	$(0.832, 1.06)$

$\boxed{\lambda_E}$ Model	Posterior Mean	Posterior SD	Central 95% Interval
Gaussian	0.768	0.0597	$(0.651, 0.885)$
Poisson	0.769	0.0521	$(0.671, 0.875)$

$\boxed{\Delta = \lambda_E - \lambda_C}$ Model	Posterior Mean	Posterior SD	Central 95% Interval
Gaussian	-0.176	0.0944	$(-0.361, 0.009)$
Poisson	-0.174	0.0774	$(-0.325, -0.024)$

The two models produce **almost identical point estimates**, but the Poisson model leads to **sharper inferences** (e.g., the posterior SD for the treatment effect $\Delta = \lambda_E - \lambda_C$ is **22%** larger in model 0 than in model 1).

5.5 Additive and Multiplicative Treatment Effects

This is the same point we noticed with the NB10 data—when a location parameter is the only thing at issue, the Gaussian is a **conservative** modeling choice (intuitively, the Poisson gains its "extra accuracy" from the variance and the mean being equal, which permits **second-moment** information to help in estimating the λ values along with the usual first-moment information).

Both the Gaussian and Poisson models so far implicitly assume that the treatment effect is **additive**:

$$E \overset{\text{st}}{=} C + \text{effect}, \tag{13}$$

where $\overset{\text{st}}{=}$ means *is stochastically equal to*; in other words, apart from random variation the effect of the IHGA is to **add or subtract a constant** to or from each person's underlying rate of hospitalization.

However, since the outcome variable is non-negative, it is plausible that a **better model** for the data is

$$E \overset{\text{st}}{=} (1 + \text{effect})\, C. \tag{14}$$

Additive vs. Multiplicative Effect

Here the treatment effect is **multiplicative**—in other words, apart from random variation the effect of the IHGA is to **multiply** each person's underlying rate of hospitalization by a constant above or below 1.

A **qqplot** of the control and experimental outcome values can in some cases be helpful in choosing between additive and multiplicative models:

```
> CEqq <- qqplot( C, E, plot = F )

> table( CEqq$y, CEqq$x )

               Interpolated C values
      0 0.965  1 1.5  2 2.82 3 3.91 4 4.96 5 6.99 7
  0 137     1  9   0  0    0 0    0 0    0 0    0 0
  1   0     0 66   1 16    0 0    0 0    0 0    0 0
  2   0     0  0   0 29    1 7    0 0    0 0    0 0
E 3   0     0  0   0  0    0 4    1 7    1 0    0 0
  4   0     0  0   0  0    0 0    0 0    0 3    0 0
  5   0     0  0   0  0    0 0    0 0    0 0    1 0
  6   0     0  0   0  0    0 0    0 0    0 0    0 1

> symbols( c( 0, 0.964798, 1, 1, 1.5, 2, 2, 2.823944, 3, 3,
    3.908447, 4, 4.964813, 5, 6.985962, 7 ), c( rep( 0, 3 ),
    rep( 1, 3 ), rep( 2, 3 ), rep( 3, 4 ), 4, 5, 6 ),
    circles = c( 137, 1, 9, 66, 1, 16, 29, 1, 7, 4, 1, 7, 1,
    3, 1, 1 ), xlab = 'C', ylab = 'E' )
```

Additive vs. Multiplicative Effect

```
> abline( 0, 1 )                    # E = C (no effect)

> abline( 0, 0.793, lty = 2 )       # E = 0.816 C
                                    #   (multiplicative)

> abline( -0.174, 1, lty = 3 )      # E = C - 0.174 (additive)
```

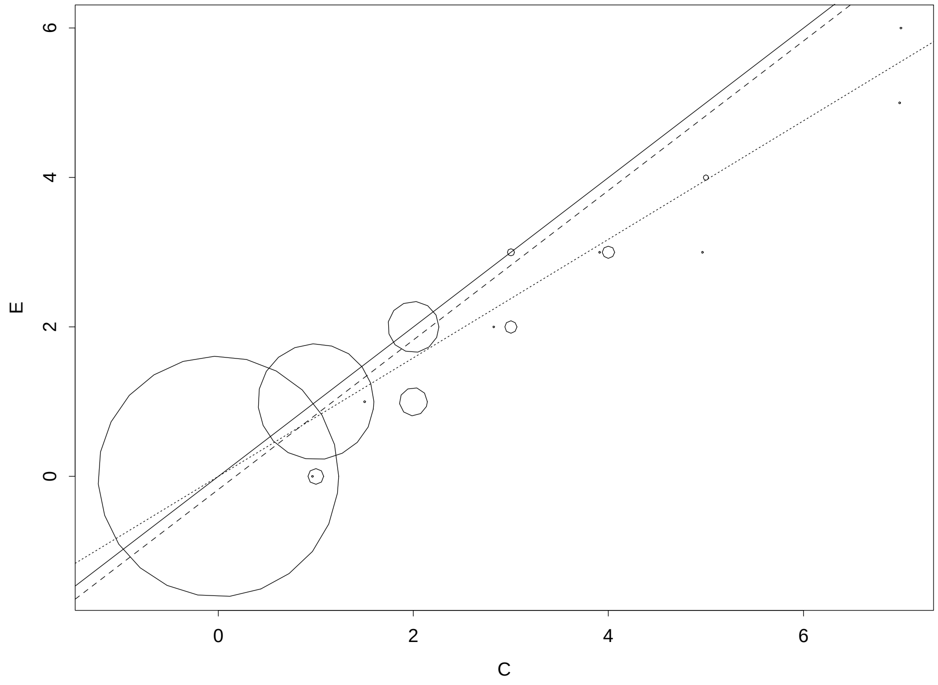


Figure 6.3. QQplot of E versus C values, with the radii of the plotted circles proportional to the number of observations at the indicated point. The solid line corresponds to no treatment effect, the small dotted line to the best-fitting multiplicative model ($E \overset{\text{st}}{=} 0.816\,C$), and the large dotted line to the best-fitting additive model ($E \overset{\text{st}}{=} C - 0.174$).

Here, because the Poisson model has only **one parameter** for both location and scale, the multiplicative and additive formulations **fit equally well**, but the multiplicative model **generalizes** more readily (see below).

A Multiplicative Poisson Model

A simple way to write the multiplicative model is to re-express the data in the form of a **regression** of the outcome y on a **dummy variable** x which is 1 if the person was in the experimental group and 0 if he/she was in the control group:

i	1	2	$\cdots$	287	288	289	$\cdots$	572
x_i	0	0	$\cdots$	0	1	1	$\cdots$	1
y_i	1	0	$\cdots$	2	0	3	$\cdots$	1

Then for $i = 1, \ldots, n = 572$ the **multiplicative** model can be written

$$\begin{aligned} (y_i \,|\, \lambda_i) & \overset{\text{indep}}{\sim} \text{Poisson}(\lambda_i) \\ \log(\lambda_i) & = \gamma_0 + \gamma_1 x_i \\ (\gamma_0, \gamma_1) & \sim \text{diffuse} \end{aligned} \qquad (15)$$

In this model the **control** people have

$$\log(\lambda_i) = \gamma_0 + \gamma_1(0) = \gamma_0, \quad \text{i.e.,} \quad \lambda_C = e^{\gamma_0}, \qquad (16)$$

and the **experimental** people have

$$\begin{aligned} \log(\lambda_i) & = \gamma_0 + \gamma_1(1) = \gamma_0 + \gamma_1, \quad \text{i.e.,} \\ \lambda_E & = e^{\gamma_0 + \gamma_1} = e^{\gamma_0} e^{\gamma_1} = \lambda_C e^{\gamma_1}. \end{aligned} \qquad (17)$$

Now you may remember from basic **Taylor series** that for γ_1 not too far from 0

$$e^{\gamma_1} \doteq 1 + \gamma_1, \qquad (18)$$

A Multiplicative Poisson Model

so that finally (for γ_1 **fairly near 0**)

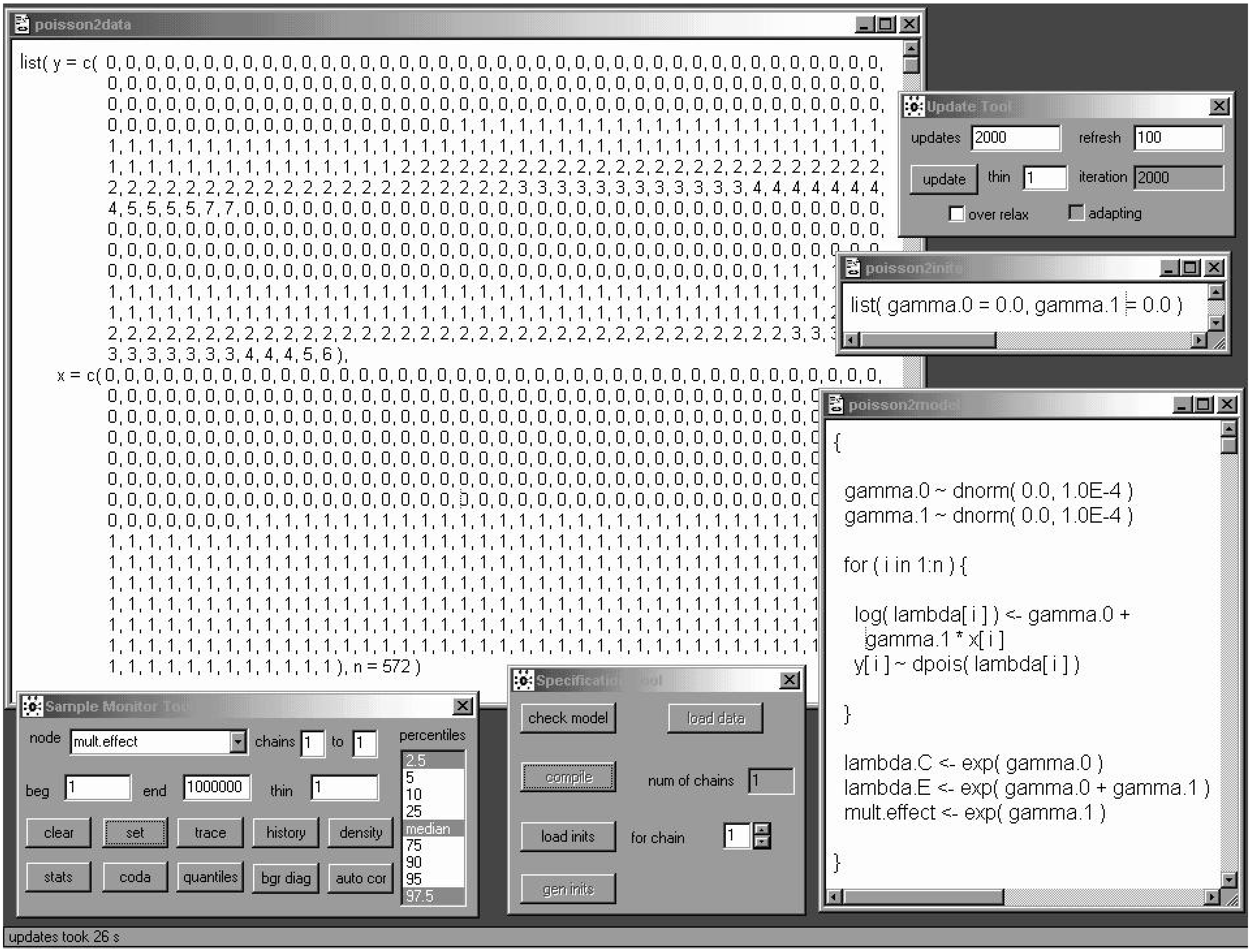
$$\lambda_E \doteq (1 + \gamma_1)\,\lambda_C, \tag{19}$$

which is a way of expressing equation (3) in **Poisson language**.

Fitting this model in `classicBUGS` is easy:

```
model poisson2;

const

  n = 572;

var

  gamma.0, gamma.1, lambda[ n ], x[ n ], y[ n ], lambda.C,
  lambda.E, mult.effect;

data x in "poisson-x.dat", y in "poisson-y.dat";
inits in "poisson2.in";

{
  gamma.0 ~ dnorm( 0.0, 1.0E-4 );    # flat priors for
  gamma.1 ~ dnorm( 0.0, 1.0E-4 );    # gamma.0 and gamma.1

  for ( i in 1:n ) {

    log( lambda[ i ] ) <- gamma.0 + gamma.1 * x[ i ];
    y[ i ] ~ dpois( lambda[ i ] );

  }

  lambda.C <- exp( gamma.0 );
  lambda.E <- exp( gamma.0 + gamma.1 );
  mult.effect <- exp( gamma.1 );
}
```

WinBUGS Implementation (continued)



```
list( y = c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
             2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4,
             4, 5, 5, 5, 5, 7, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2
             2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3
             3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 5, 6 ),
      x = c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
             0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ), n = 572 )
```

updates 2000 refresh 100

thin 1 iteration 2000

□ over relax □ adapting

```
list( gamma.0 = 0.0, gamma.1 = 0.0 )
```

```
{

  gamma.0 ~ dnorm( 0.0, 1.0E-4 )
  gamma.1 ~ dnorm( 0.0, 1.0E-4 )

  for ( i in 1:n ) {

    log( lambda[ i ] ) <- gamma.0 +
      gamma.1 * x[ i ]
    y[ i ] ~ dpois( lambda[ i ] )

  }

  lambda.C <- exp( gamma.0 )
  lambda.E <- exp( gamma.0 + gamma.1 )
  mult.effect <- exp( gamma.1 )

}
```

node mult.effect chains 1 to 1

beg 1 end 1000000 thin 1

percentiles: 2.5, 5, 10, 25, median, 75, 90, 95, 97.5

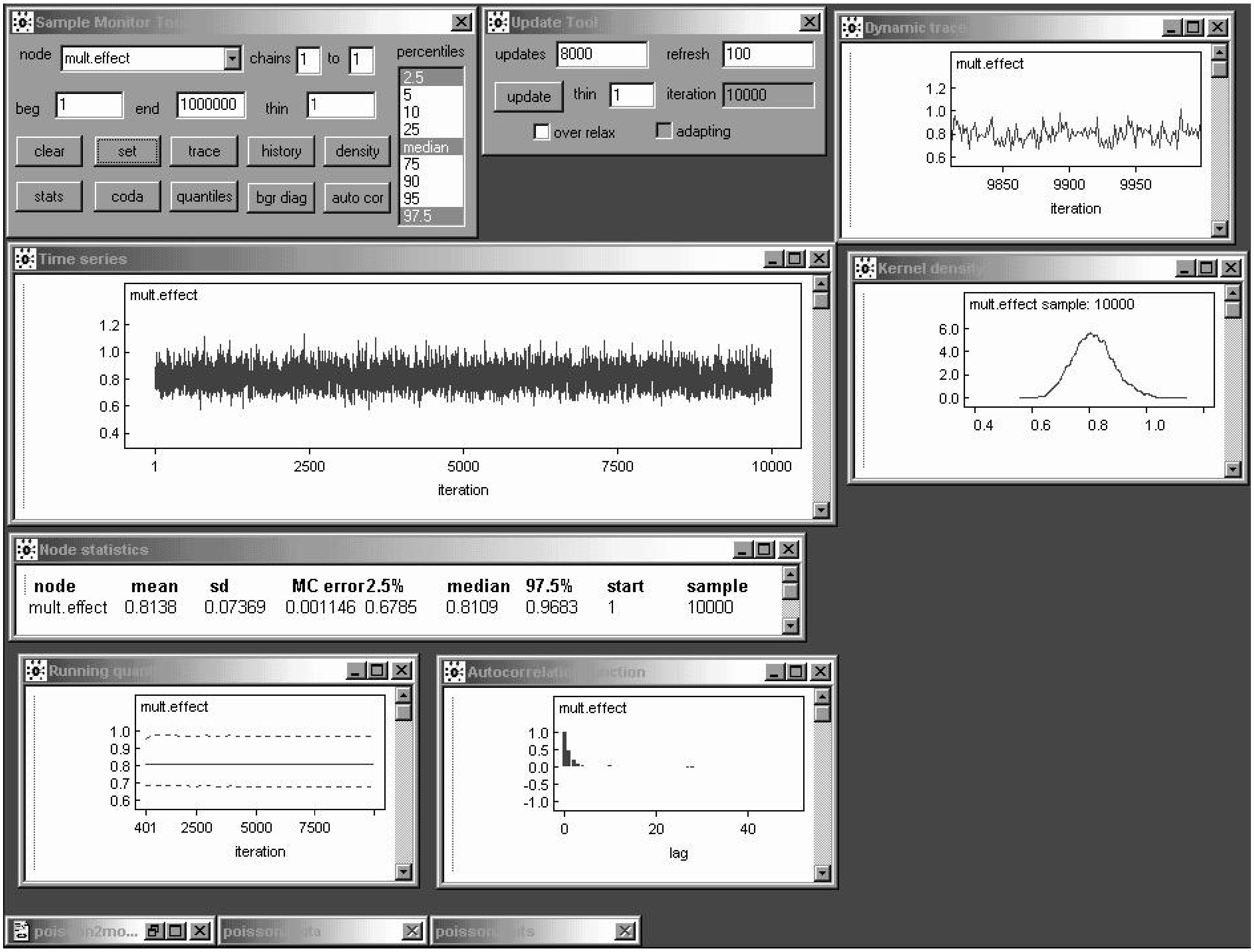
num of chains 1

for chain 1

updates took 26 s

The **multiplicative Poisson model** (4) takes longer to run—2,000 burn-in iterations now take about **4 seconds at 2.0 PC GHz**—but still exhibits **fairly good mixing**, as we'll see below.

WinBUGS Implementation (continued)



Sample Monitor Tool: node mult.effect, chains 1 to 1, beg 1, end 1000000, thin 1

Update Tool: updates 8000, refresh 100, thin 1, iteration 10000, ☐ over relax, ☐ adapting

```
node          mean     sd        MC error  2.5%     median   97.5%    start   sample
mult.effect   0.8138   0.07369   0.001146  0.6785   0.8109   0.9683   1       10000
```

A total of **10,000 iterations** (the chain started essentially in equilibrium, so the burn-in can be absorbed into the monitoring run) reveals that the **multiplicative effect parameter** e^{γ_1} in model (4) behaves like an AR_1 series with $\hat{\rho}_1 \doteq 0.5$, but the Monte Carlo standard error for the posterior mean is still only about **0.001** with a run of this length.

Additive versus Multiplicative Fit

A burn-in of 2,000 and a monitoring run of 8,000 again yields **good MCMC diagnostics** and permits a comparison between the additive and multiplicative Poisson models, as in Table 6.4.

Table 6.4. Comparison of inferential conclusions from the additive and multiplicative Poisson models.

$\boxed{\lambda_C}$ Model	Posterior Mean	Posterior SD	Central 95% Interval
additive	0.943	0.0577	$(0.832, 1.06)$
multiplicative	0.945	0.0574	$(0.837, 1.06)$

$\boxed{\lambda_E}$ Model	Posterior Mean	Posterior SD	Central 95% Interval
additive	0.769	0.0521	$(0.671, 0.875)$
multiplicative	0.768	0.0518	$(0.671, 0.872)$

$\boxed{\text{effect}}$ Model	Posterior Mean	Posterior SD	Central 95% Interval
additive	-0.174	0.0774	$(-0.325, -0.024)$
multiplicative	-0.184	0.0743	$(-0.324, -0.033)$

With this model it is as if the experimental people’s average underlying rates of hospitalization have been **multiplied by 0.82**, give or take about 0.07.

The additive and multiplicative effects are **similar** here, because both are not too far from zero.

Extra-Poisson Variability

However, none of this has verified that the **Poisson model is reasonable** for these data—the histograms show that the Gaussian model is clearly unreasonable, but the diagnostic plots in `WinBUGS` and `CODA` only check on the adequacy of the **MCMC** sampling, not the model.

In fact we had a good clue that the data are **not** Poisson back on page 2: as noted in part 2, the Poisson(λ) distribution has mean λ and also variance λ—in other words, the **variance-to-mean-ratio** (VTMR) for the Poisson is 1. But

```
> var( C ) / mean( C )
[1] 1.62599
> var( E ) / mean( E )
[1] 1.322979
```

i.e., the data exhibit **extra-Poisson variability** (VTMR$\,> 1$).

This actually **makes good sense** if you think about it, as follows.

The Poisson model assumes that everybody in the control group has the **same underlying rate** λ_C of hospitalization, and similarly everybody in the experimental group has the **same rate** λ_E.

Unobserved Predictor Variables

In reality it's far more reasonable to think that each person has his/her **own** underlying rate of hospitalization that depends on **baseline health status**, **age**, and various other things.

Now Hendriksen **forgot to measure** (or at least to report on) these other variables (he may have hoped that the randomization would balance them between C and E)—the only predictor we have is x, the **experimental status dummy variable**—so the best we can do is to lump all of these other **unobserved** predictor variables together into a kind of **"error" term** e.

This amounts to **expanding** the second Poisson model (4) above: for $i = 1, \ldots, n = 572$ the new model is

$$\begin{array}{rcl} (y_i \,|\lambda_i) & \overset{\text{indep}}{\sim} & \text{Poisson}(\lambda_i) \\ \log(\lambda_i) & = & \gamma_0 + \gamma_1 x_i + e_i \\ e_i & \overset{\text{IID}}{\sim} & N\Big(0, \sigma_e^2\Big) \\ \Big(\gamma_0, \gamma_1, \sigma_e^2\Big) & \sim & \text{diffuse.} \end{array} \qquad (20)$$

5.6 Random-Effects Poisson Regression

The **Gaussian** choice for the error distribution is **conventional**, not dictated by the science of the problem (although if there were a lot of unobserved predictors hidden inside the e_i their weighted sum would be close to normal by the **Central Limit Theorem**).

Model (9) is an **expansion** of the earlier model (4) because you can obtain model (4) from (9) by setting $\sigma_e^2 = 0$, whereas with (9) we're letting σ_e^2 vary and **learning about it from the data**.

The addition of the $\boxed{\textbf{random effects}}$ e_i to the model is one way to address the extra-Poisson variability: this model would be called a **lognormal mixture of Poisson distributions** (or a **random effects Poisson regression** (REPR) model) because it's as if each person's λ is drawn from a lognormal distribution and then his/her number of hospitalizations y is drawn from a Poisson distribution with his/her λ, and this mixing process will make the variance of y **bigger than its mean**.

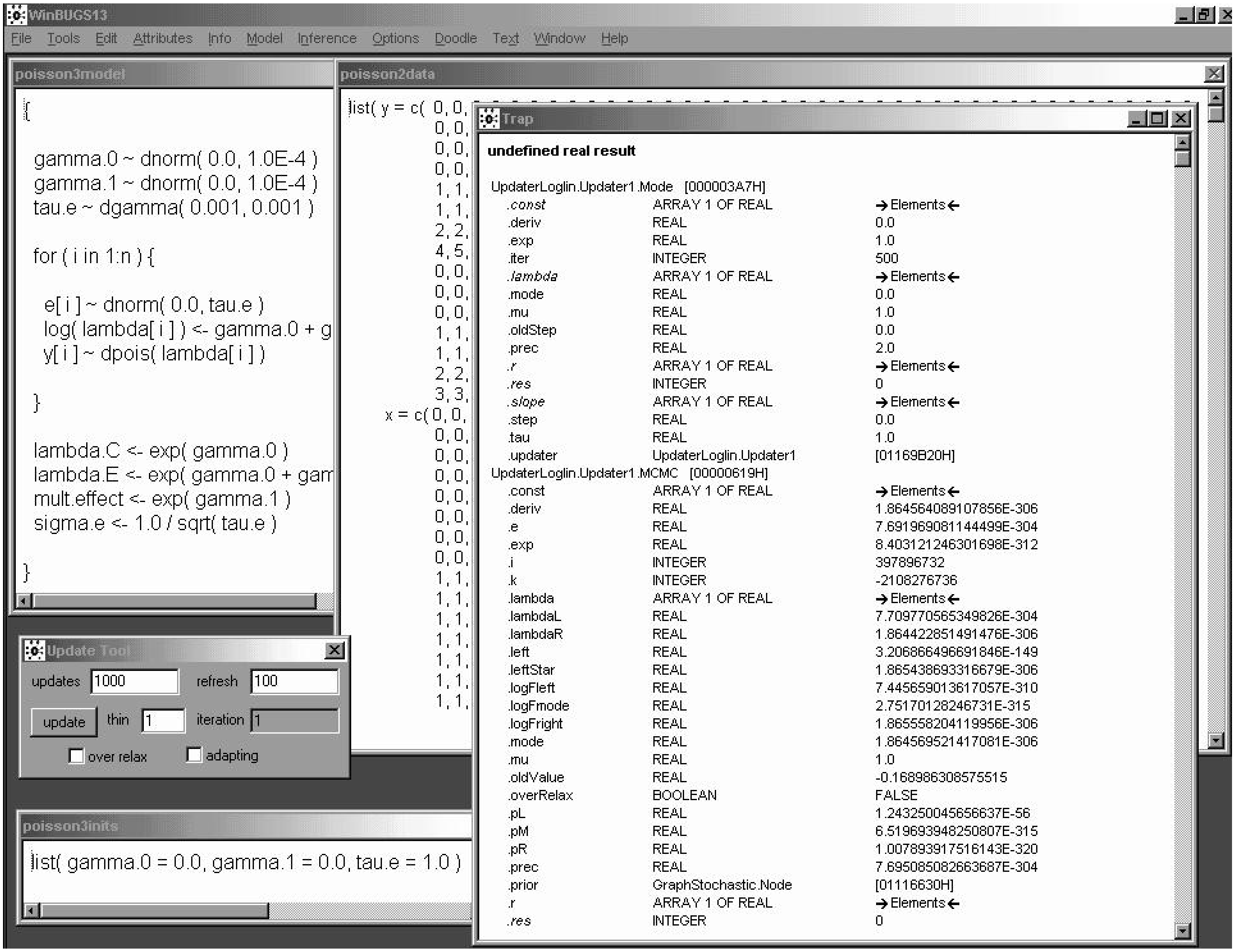
WinBUGS Implementation

The new WinBUGS model is

```
{

  gamma.0 ~ dnorm( 0.0, 1.0E-4 )
  gamma.1 ~ dnorm( 0.0, 1.0E-4 )
  tau.e ~ dgamma( 0.001, 0.001 )

  for ( i in 1:n ) {

    e[ i ] ~ dnorm( 0.0, tau.e )
    log( lambda[ i ] ) <- gamma.0 + gamma.1 * x[ i ] +
      e[ i ]
    y[ i ] ~ dpois( lambda[ i ] )

  }

  lambda.C <- exp( gamma.0 )
  lambda.E <- exp( gamma.0 + gamma.1 )
  mult.effect <- exp( gamma.1 )
  sigma.e <- 1.0 / sqrt( tau.e )

}
```

I again use a **diffuse** $\Gamma(\epsilon, \epsilon)$ prior (with $\epsilon = 0.001$) for the **precision** τ_e of the random effects.

WinBUGS Implementation (continued)


WinBUGS13
File Tools Edit Attributes Info Model Inference Options Doodle Text Window Help
poisson3model
poisson2data
Trap
Update Tool
poisson3inits
update

```
{

  gamma.0 ~ dnorm( 0.0, 1.0E-4 )
  gamma.1 ~ dnorm( 0.0, 1.0E-4 )
  tau.e ~ dgamma( 0.001, 0.001 )

  for ( i in 1:n ) {

    e[ i ] ~ dnorm( 0.0, tau.e )
    log( lambda[ i ] ) <- gamma.0 + g
    y[ i ] ~ dpois( lambda[ i ] )

  }

  lambda.C <- exp( gamma.0 )
  lambda.E <- exp( gamma.0 + gam
  mult.effect <- exp( gamma.1 )
  sigma.e <- 1.0 / sqrt( tau.e )

}
```

```
list( y = c( 0,0,
            0,0,
            0,0,
            0,0,
            1,1,
            1,1,
            2,2,
            4,5,
            0,0,
            0,0,
            0,0,
            1,1,
            1,1,
            2,2,
            3,3,
       x = c( 0,0,
            0,0,
            0,0,
            0,0,
            0,0,
            0,0,
            0,0,
            0,0,
            1,1,
            1,1,
            1,1,
            1,1,
            1,1,
            1,1,
            1,1,
```

updates 1000 refresh 100
thin 1 iteration 1
□ over relax □ adapting

```
list( gamma.0 = 0.0, gamma.1 = 0.0, tau.e = 1.0 )
```

```
undefined real result

UpdaterLoglin.Updater1.Mode   [000003A7H]
   .const         ARRAY 1 OF REAL          →Elements←
   .deriv         REAL                     0.0
   .exp           REAL                     1.0
   .iter          INTEGER                  500
   .lambda        ARRAY 1 OF REAL          →Elements←
   .mode          REAL                     0.0
   .mu            REAL                     1.0
   .oldStep       REAL                     0.0
   .prec          REAL                     2.0
   .r             ARRAY 1 OF REAL          →Elements←
   .res           INTEGER                  0
   .slope         ARRAY 1 OF REAL          →Elements←
   .step          REAL                     0.0
   .tau           REAL                     1.0
   .updater       UpdaterLoglin.Updater1   [01169B20H]
UpdaterLoglin.Updater1.MCMC   [00000619H]
   .const         ARRAY 1 OF REAL          →Elements←
   .deriv         REAL                     1.864564089107856E-306
   .e             REAL                     7.691969081144499E-304
   .exp           REAL                     8.403121246301698E-312
   .i             INTEGER                  397896732
   .k             INTEGER                  -2108276736
   .lambda        ARRAY 1 OF REAL          →Elements←
   .lambdaL       REAL                     7.709770565349826E-304
   .lambdaR       REAL                     1.864422851491476E-306
   .left          REAL                     3.206866496691846E-149
   .leftStar      REAL                     1.865438693316679E-306
   .logFleft      REAL                     7.445659013617057E-310
   .logFmode      REAL                     2.75170128246731E-315
   .logFright     REAL                     1.865558204119956E-306
   .mode          REAL                     1.864569521417081E-306
   .mu            REAL                     1.0
   .oldValue      REAL                     -0.168986308575515
   .overRelax     BOOLEAN                  FALSE
   .pL            REAL                     1.243250045656637E-56
   .pM            REAL                     6.519693948250807E-315
   .pR            REAL                     1.007893917516143E-320
   .prec          REAL                     7.695085082663687E-304
   .prior         GraphStochastic.Node     [01116630H]
   .r             ARRAY 1 OF REAL          →Elements←
   .res           INTEGER                  0
```

With a model like that in equation (9), there are n **random effects** e_i that need to be sampled as nodes in the graph (the e_i play the role of **auxiliary variables** in the MCMC) along with the fixed effects (γ_0, γ_1) and the variance parameter σ_e^2.

In earlier releases of the software, at least, this made it more crucial to give WinBUGS **good starting values**.

Here WinBUGS release **1.3** has figured out that random draws like $\boxed{1.66 \cdot 10^{-316}}$ result from the **generic** (and quite poor) initial values $(\gamma_0, \gamma_1, \tau_e) = (0.0, 0.0, 1.0)$ and has **refused to continue sampling**.

Sensitivity to Initial Values

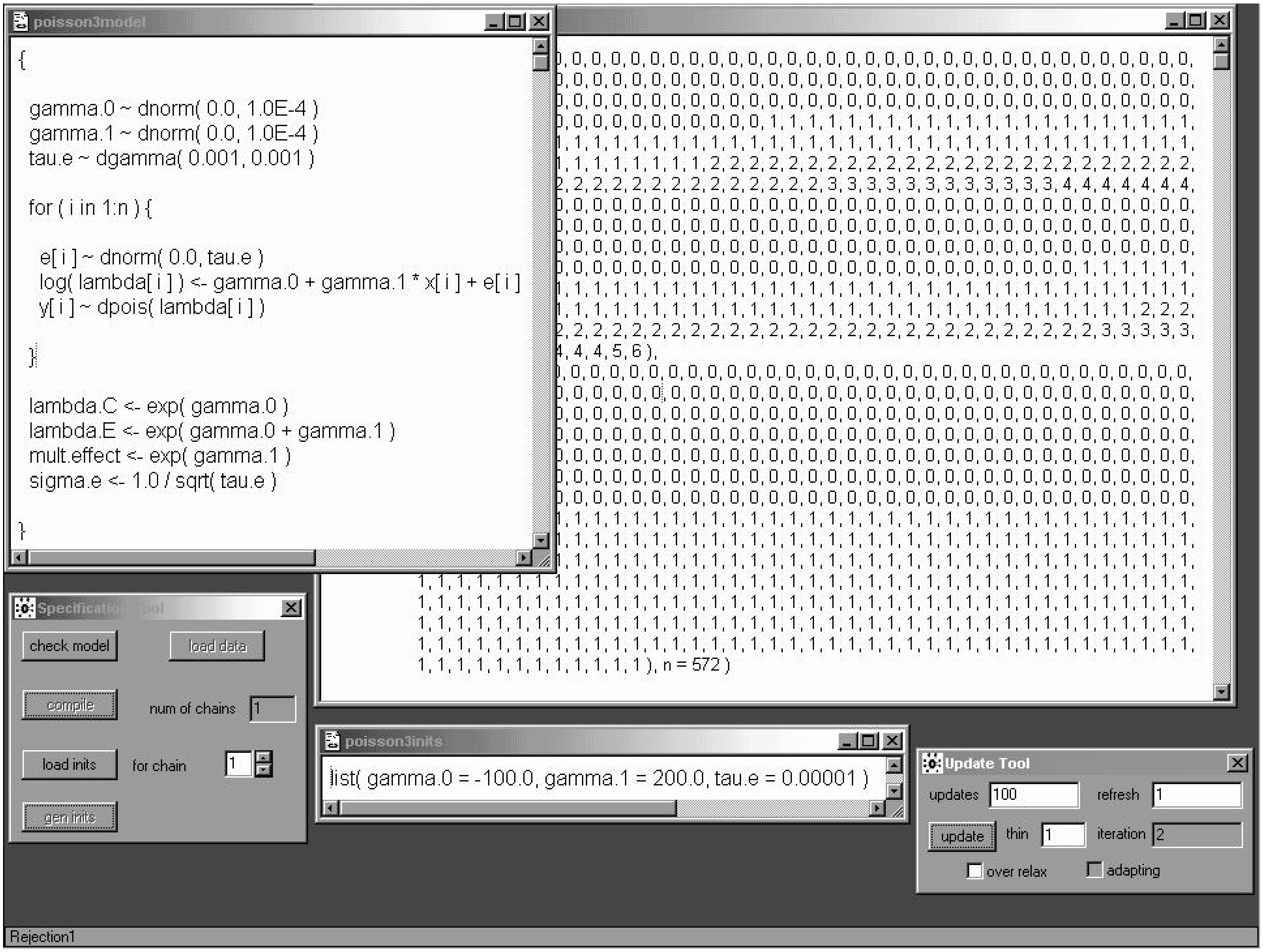
$\boxed{\textbf{Warning}}$ `WinBUGS` can fail, particularly in random-effects models, when you give it initial values that are not very close to the final posterior means; an example in release 1.3 is the REPR model (9) on the IHGA data with the **"generic"** starting values $(\gamma_0, \gamma_1, \tau_e) = (0.0, 0.0, 1.0)$.

When this problem arises there are two ways out in `WinBUGS`: **trial and error**, or a calculation (see below).

NB `MLwiN` does not have this problem—it gets its starting values from **maximum likelihood** (the mode of the likelihood function is often a decent approximation to the mean or mode of the posterior).

Technical note. To get a decent starting value for τ_e in model (9) you can calculate as follows: renaming the random effects η_i to avoid confusion with the number e, (1) $V(y_i) = V[E(y_i \,|\eta_i)] + E[V(y_i \,|\eta_i)]$, where (2) $(y_i \,|\eta_i) \sim$ Poisson $\left(e^{\gamma_0+\gamma_1 x_i+\eta_i}\right)$, so $E(y_i \,|\eta_i) = V(y_i \,|\eta_i) = e^{\gamma_0+\gamma_1 x_i+\eta_i}$. Then (3) $V[E(y_i \,|\eta_i)] = V\!\left(e^{\gamma_0+\gamma_1 x_i+\eta_i}\right) = e^{2(\gamma_0+\gamma_1 x_i)} V(e^{\eta_i})$ and $E[V(y_i \,|\eta_i)] = E\!\left(e^{\gamma_0+\gamma_1 x_i+\eta_i}\right) = e^{\gamma_0+\gamma_1 x_i} E(e^{\eta_i})$. Now (4) e^{η_i} is lognormal with mean 0 and variance σ_e^2 on the log scale, so $E(e^{\eta_i}) = e^{\frac{1}{2}\sigma_e^2}$ and $V(e^{\eta_i}) = e^{\sigma_e^2}\left(e^{\sigma_e^2} - 1\right)$, yielding finally $V(y_i) = e^{2(\gamma_0+\gamma_1 x_i)+\frac{1}{2}\sigma_e^2} + e^{\gamma_0+\gamma_1 x_i+\sigma_e^2}\left(e^{\sigma_e^2} - 1\right)$. (5) Plugging in $x_i = 0$ for the C group, whose sample variance is 1.54, and using the value $\gamma_0 = -0.29$ from runs with previous models, gives an equation for σ_e^2 that can be solved numerically, yielding $\sigma_e^2 \doteq 0.5$ and $\tau_e \doteq 2$.

WinBUGS Implementation (continued)



```
{

  gamma.0 ~ dnorm( 0.0, 1.0E-4 )
  gamma.1 ~ dnorm( 0.0, 1.0E-4 )
  tau.e ~ dgamma( 0.001, 0.001 )

  for ( i in 1:n ) {

    e[ i ] ~ dnorm( 0.0, tau.e )
    log( lambda[ i ] ) <- gamma.0 + gamma.1 * x[ i ] + e[ i ]
    y[ i ] ~ dpois( lambda[ i ] )

  }

  lambda.C <- exp( gamma.0 )
  lambda.E <- exp( gamma.0 + gamma.1 )
  mult.effect <- exp( gamma.1 )
  sigma.e <- 1.0 / sqrt( tau.e )

}
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3,
4, 4, 4, 5, 6 ),
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ), n = 572 )
```

```
list( gamma.0 = -100.0, gamma.1 = 200.0, tau.e = 0.00001 )
```

num of chains 1

for chain 1

updates 100 refresh 1

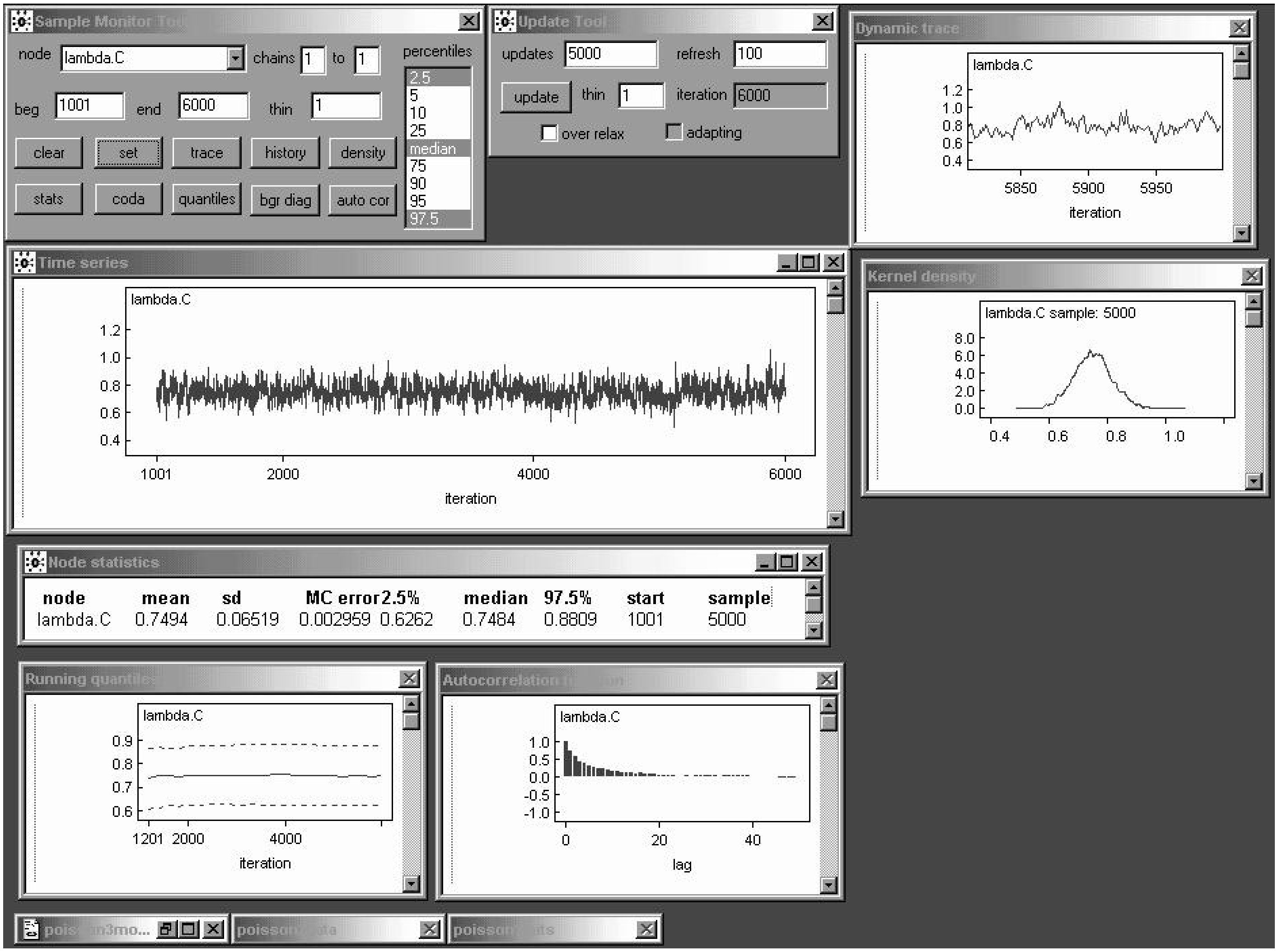
thin 1 iteration 2

□ over relax □ adapting

Rejection1

Interestingly, WinBUGS release 1.4 is able to **sample successfully** with the generic starting values $(\gamma_0, \gamma_1, \tau_e) = (0.0, 0.0, 1.0)$, although of course a **longer burn-in period** would be needed when they're used; you have to try **truly absurd** initial values to get it to fall over, and when it does so the **error message** ("Rejection1") in the lower left corner is **more discreet**.

WinBUGS Implementation (continued)



node lambda.C chains 1 to 1

beg 1001 end 6000 thin 1

updates 5000 refresh 100

thin 1 iteration 6000

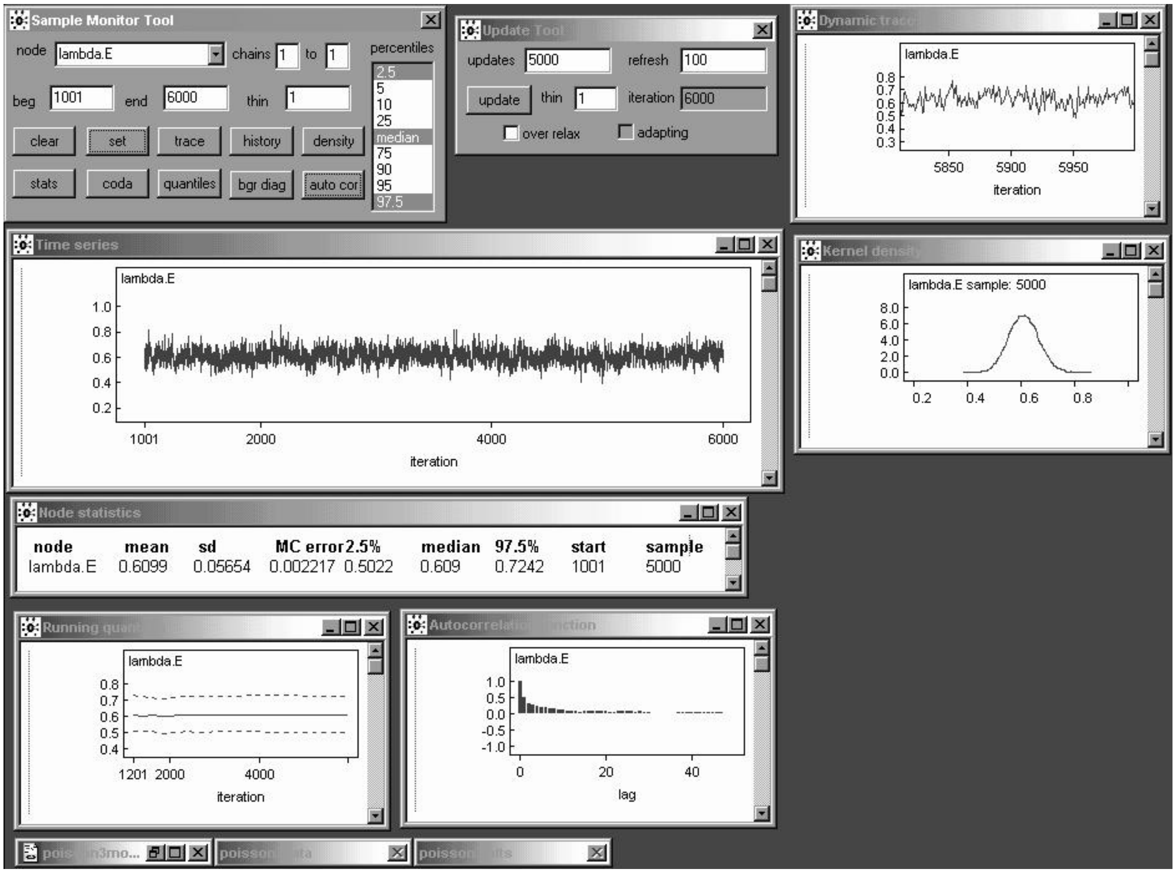
□ over relax □ adapting

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
lambda.C	0.7494	0.06519	0.002959	0.6262	0.7484	0.8809	1001	5000

With a **better set of initial values**—$(\gamma_0, \gamma_1, \tau_e) = (-0.058, -0.21, 2.0)$, obtained from (a) the earlier Poisson models (in the case of the regression parameters γ_j) and (b) either a calculation like the one on the bottom of page 25 or trial and error—WinBUGS is able to make progress, although this model takes **a fairly long time to fit** in release 1.4: a burn-in of 1,000 takes 5.5 seconds at 2.0 PC GHz (for some reason the code runs about twice as fast in release 1.3).

A monitoring run of **5,000** iterations reveals that the random effects make everything **mix more slowly**: λ_C (this page) and λ_E and the multiplicative effect (next page) all behave like AR_1 series with $\hat{\rho}_1 \doteq 0.7$, 0.5, and 0.6, respectively.

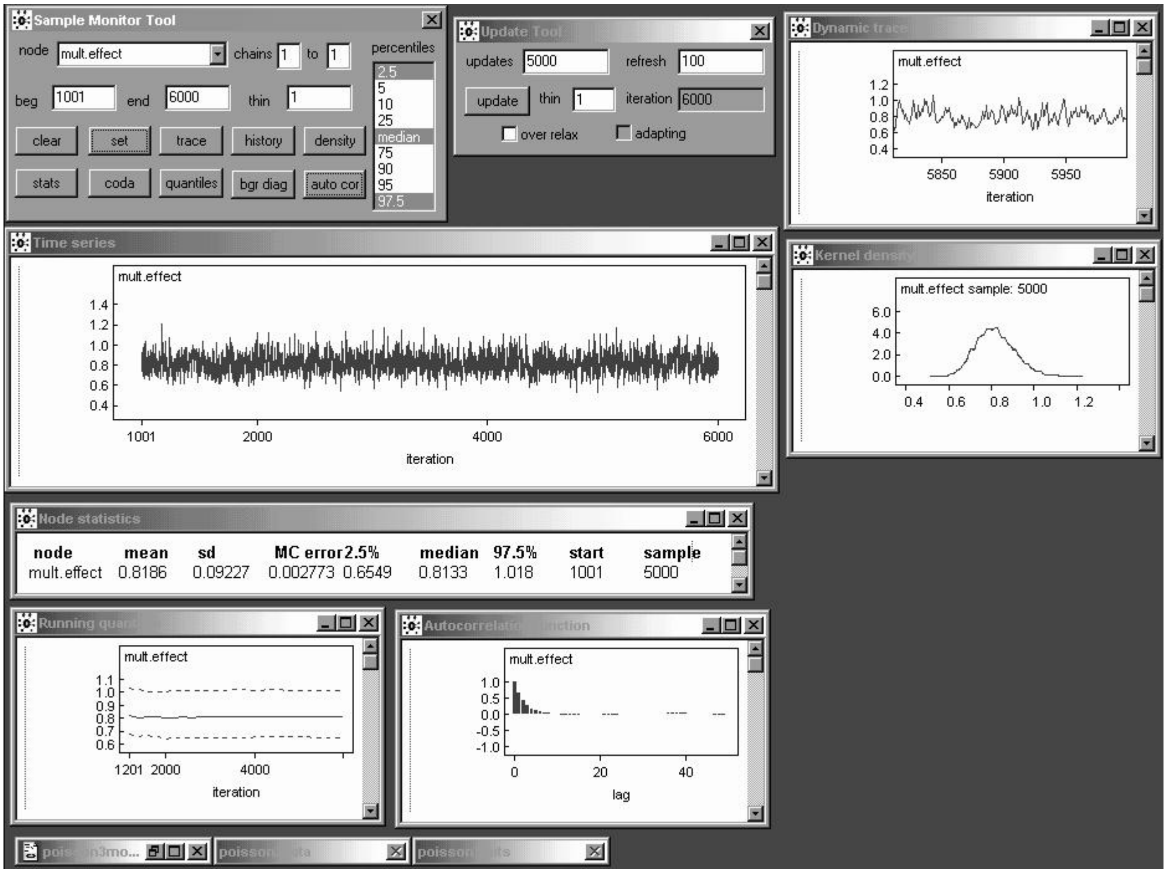
WinBUGS Implementation (continued)



Sample Monitor Tool: node lambda.E, chains 1 to 1, beg 1001, end 6000, thin 1

Update Tool: updates 5000, refresh 100, thin 1, iteration 6000, □ over relax, □ adapting

```
node      mean    sd       MC error2.5%      median  97.5%   start  sample
lambda.E  0.6099  0.05654  0.002217  0.5022  0.609   0.7242  1001   5000
```

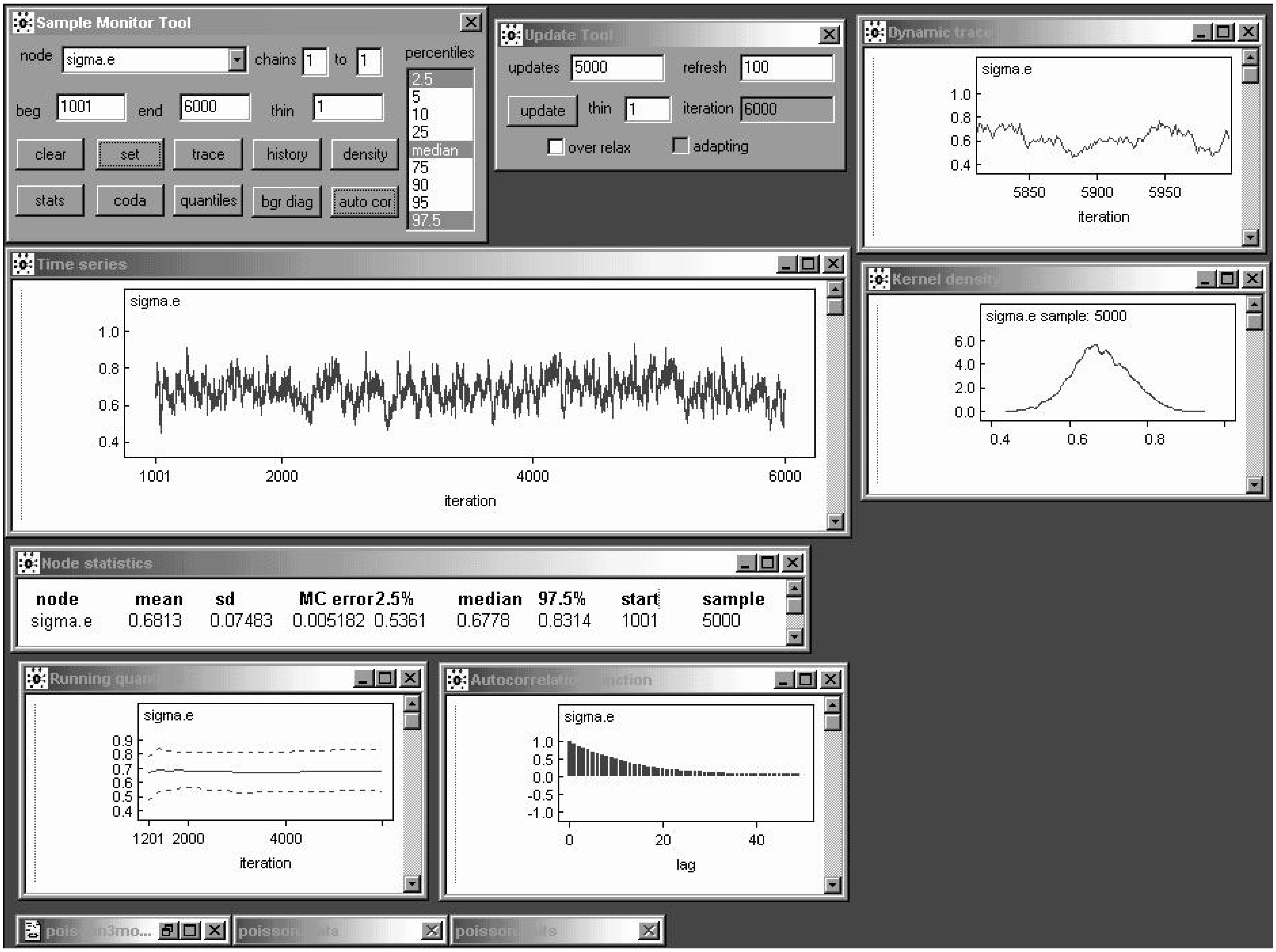


Sample Monitor Tool: node mult.effect, chains 1 to 1, beg 1001, end 6000, thin 1

Update Tool: updates 5000, refresh 100, thin 1, iteration 6000, □ over relax, □ adapting

```
node         mean    sd       MC error2.5%      median  97.5%  start  sample
mult.effect  0.8186  0.09227  0.002773  0.6549  0.8133  1.018  1001   5000
```

`WinBUGS` Implementation (continued)



node sigma.e chains 1 to 1

beg 1001 end 6000 thin 1

updates 5000 refresh 100

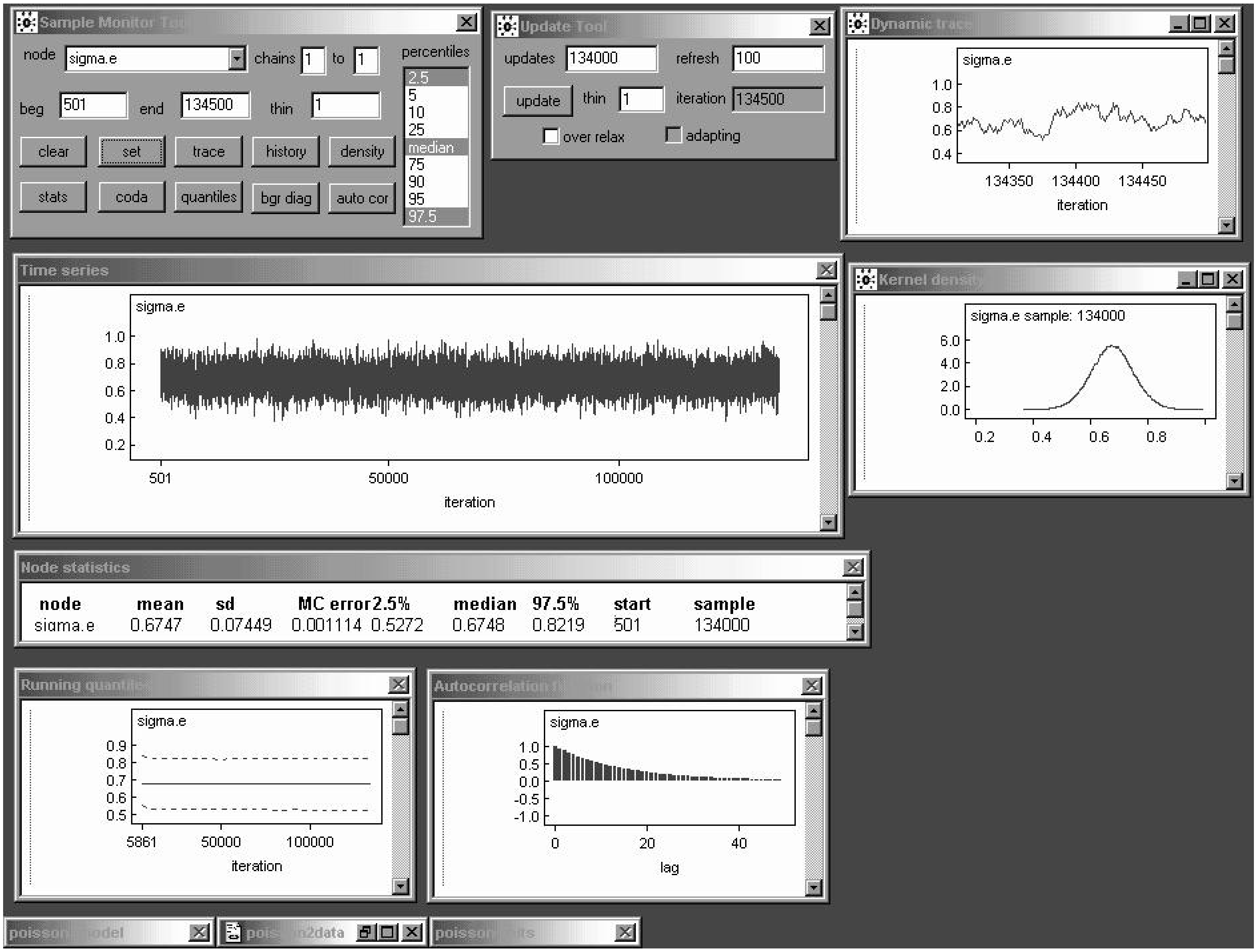
thin 1 iteration 6000

node	mean	sd	MC error	2.5%	median	97.5%	start	sample
sigma.e	0.6813	0.07483	0.005182	0.5361	0.6778	0.8314	1001	5000

Learning about σ_e in this model is **slow**: its autocorrelation function is that of an AR_1 with a **high value** of $\hat{\rho}_1$ (equation (55) on page 75 of part 3 gives $\hat{\rho}_1 \doteq \mathbf{0.92}$).

The MCSE of the posterior mean for σ_e based on 5,000 draws is **0.005182**—to get this down to (say) **0.001** I need to increase the length of the monitoring run by a factor of $\left(\frac{0.005182}{0.001}\right)^2 \doteq \mathbf{26.9}$, meaning a total run of about $(26.9)(5,000) \doteq 134,000$ iterations (this takes about **15 minutes** at 2.0 PC GHz).

WinBUGS Implementation (continued)



Sample Monitor Tool

node sigma.e chains 1 to 1

beg 501 end 134500 thin 1

Update Tool

updates 134000 refresh 100

thin 1 iteration 134500

□ over relax □ adapting

Node statistics

```
node      mean     sd       MC error2.5%     median   97.5%    start   sample
sigma.e   0.6747   0.07449  0.001114 0.5272  0.6748   0.8219   501     134000
```

There is **clear evidence** that σ_e is **far from 0**—its posterior mean and SD are estimated as **0.675** (with an MCSE of about **0.001** after 134,000 iterations) and **0.074**, respectively—meaning that the **model expansion** from (4) to (9) was **amply justified**.

REPR Model Results

(Another way to achieve the goal of describing the extra-Poisson variability would be to fit different **negative binomial** distributions to the observed counts in the C and E groups—the negative binomial is a **gamma mixture of Poissons**, and the gamma and lognormal distributions often fit long-tailed data about equally well, so you would not be surprised to find that the two approaches give **similar results**.)

Table 6.5. Comparison of inferential conclusions about the multiplicative effect parameter e^{γ_1} from the fixed- and random-effects Poisson regression models.

Model	Posterior Mean	Posterior SD	Central 95% Interval
FEPR	0.816	0.0735	$(0.683, 0.969)$
REPR	0.830	0.0921	$(0.665, 1.02)$

Table 6.5 compares the REPR model inferential results with those from model (4), which could also be called a **fixed-effects Poisson regression** (FEPR) model.

The "error" SD σ_e has posterior mean **0.68**, give or take about 0.07 (on the log(λ) scale), corresponding to substantial extra-Poisson variability, which translates into **increased uncertainty** about the multiplicative effect parameter e^{γ_1}.

I'll argue later that the REPR model **fits the data well**, so the conclusion I'd publish from these data is that IHGA reduces the average number of hospitalizations per two years by about $100\,(1 - 0.083)\% = \boxed{\textbf{17\%}}$ give or take about 9% (ironically this conclusion is similar to that from the Gaussian model, but this is **coincidence**).

Two More Items on MCMC Accuracy

(1) A **stringent** but potentially **useful** diagnostic for deciding how long the **monitoring run** should be for a given component θ' of the parameter vector θ, if the output of your MCMC sampler for θ' behaves like an AR_1 series with first-order **autocorrelation** ρ_1, can be derived as follows.

Suppose, after a **burn-in** that's long enough to reach **stationarity**, you've done a **preliminary** monitoring run, obtaining mean $\bar{\theta}'$, SD $\widehat{\sigma}_{\theta'}$, and first-order **autocorrelation** $\widehat{\rho}_1$ as **estimates** of the corresponding summaries for θ'.

Writing $\theta' = a \cdot 10^b$ for $1 \le a < 10$, if you want at least k **significant figures** (sigfigs) of accuracy for the posterior mean summary for θ' with **Monte Carlo probability** of at least $100(1-\alpha)$, you can check that you'll need

$$2\Phi^{-1}\left(1-\frac{\alpha}{2}\right)\widehat{SE}\left(\bar{\theta}'\right) \le 10^{b-k+1}; \tag{21}$$

then **substituting** in the relevant expression from equation (51) in part 3,

$$\widehat{SE}\left(\bar{\theta}'\right) = \frac{\widehat{\sigma}_{\theta'}}{\sqrt{m}}\sqrt{\frac{1+\widehat{\rho}_1}{1-\widehat{\rho}_1}}, \tag{22}$$

and **solving** (12) for m yields

$$m \ge 4\left[\Phi^{-1}\left(1-\frac{\alpha}{2}\right)\right]^2\left(\frac{\widehat{\sigma}_{\theta'}}{10^{b-k+1}}\right)^2\left(\frac{1+\widehat{\rho}_1}{1-\widehat{\rho}_1}\right). \tag{23}$$

This is referred to in the `MLwiN` documentation as the **Brooks-Draper** diagnostic (Brooks and Draper 2004).

Comments. (a) This diagnostic is **sensitive** to the **scale** chosen by the user for reporting results, as far as choosing the **target** number of sigfigs is concerned.

MCMC Accuracy (continued)

Example. In my initial monitoring run of 5,000 iterations in the **NB10** case study, the **posterior mean** of μ, on the micrograms below 10g scale, was $\bar{\theta}' = 404.3$ (to 4 sigfigs); the other relevant quantities for μ were as follows: **posterior SD** $\widehat{\sigma}_{\theta'} \doteq 0.464$ and **first-order autocorrelation** $\widehat{\rho}_1 \doteq 0.294$ (**NB** the MCSE for μ is already down to **0.009** with 5,000 iterations, so I already have a bit more than **4 sigfigs** of accuracy).

Suppose (just for the sake of **illustration**; it's hard to imagine setting an accuracy goal this stringent in practice) that I want to ensure **5 sigfigs** with at least 95% Monte Carlo probability for the posterior mean—write $\bar{\theta}' = 4.043 \cdot 10^2$, so that $b = 2$, take $\alpha = 0.05$ and **substitute** into (14) to yield

$$m \geq 4(1.96)^2 \left(\frac{0.464}{10^{2-5+1}} \right)^2 \left(\frac{1+0.294}{1-0.294} \right) \doteq 60,600. \qquad (24)$$

Now, if you instead **subtracted 404** from all of the data values (on the micrograms below 10g scale) and made a similar MCMC run, everything would be the same as above except that your current posterior mean for μ would be **0.3** to 1 sigfig, and (with the same MCSE of 0.009) you would regard yourself as already having a bit more than **1 sigfig** of accuracy from the initial monitoring run of 5,000.

Then to apply (14) to get **2 sigfigs** of accuracy you would write $\bar{\theta}' = 3.0 \cdot 10^{-1}$ and obtain

$$m \geq 4(1.96)^2 \left(\frac{0.464}{10^{(-1)-2+1}} \right)^2 \left(\frac{1+0.294}{1-0.294} \right) \doteq 60,600. \qquad (25)$$

These two sets of results from (14) are **consistent**—by subtracting 404 from all of the data values you (at least temporarily) **threw away 3 sigfigs**—but you can see that **care** needs to be taken in thinking about how much **accuracy** you want, and this question is closely tied to the **scale of measurement**.

(b) Note from (14) that every time you want to add **1 new sigfig** of accuracy in the posterior mean the required length of monitoring run **goes up multiplicatively** by $(10^1)^2 = \mathbf{100}$.

MCMC Accuracy (continued)

(2) I've concentrated so far on the **MCMC accuracy** of the **posterior mean**—what about other posterior summaries like the **SD**?

Suppose as above that you're interested in a given component θ' of the parameter vector θ, and that the output of your MCMC sampler for θ' behaves like an AR_1 series with first-order **autocorrelation** ρ_1; and suppose as above that after a **burn-in** that's long enough to reach **stationarity**, you've done a **preliminary** monitoring run, obtaining mean $\bar{\theta}'$, SD $\hat{\sigma}_{\theta'}$, and first-order **autocorrelation** $\hat{\rho}_1$ as **estimates** of the corresponding summaries for θ'.

Then it can be shown, in an expression **analogous** to (13), that if the **marginal posterior** for θ' is approximately **Gaussian**

$$\widehat{SE}(\hat{\sigma}_{\theta'}) = \frac{\hat{\sigma}_{\theta'}}{\sqrt{2m}} \sqrt{\frac{1+\hat{\rho}_1^2}{1-\hat{\rho}_1^2}}. \tag{26}$$

Note that with a **parameter** with MCMC output that's approximately AR_1 and **roughly Gaussian** this implies that

$$\frac{\widehat{SE}(\bar{\theta}')}{\widehat{SE}(\hat{\sigma}_{\theta'})} \doteq \sqrt{\frac{2(1+\hat{\rho}_1)^2}{1+\hat{\rho}_1^2}}, \tag{27}$$

which goes from $\sqrt{2}$ to 2 as $\hat{\rho}_1$ ranges from 0 to +1, i.e., the **mean is harder to pin down than the SD with Gaussian data** (a reflection of how **light** the tails are).

5.7 References

Aitkin M (1991). Posterior Bayes factors (with discussion). *Journal of the Royal Statistical Society, Series B*, **53**, 111–142.

Bayarri MJ, Berger JO (1998). Quantifying surprise in the data and model verification (with discussion). In *Bayesian Statistics 6*, Bernardo JM, Berger JO, Dawid AP, Smith AFM (editors). Oxford University Press, 53–82.

Berger JO, Pericchi LR (1996). The intrinsic Bayes factor for model selection and prediction. *Journal of the American Statistical Association*, **91**, 109–122.

Bernardo JM, Smith AFM (1994). *Bayesian Theory*. New York: Wiley.

Brooks SP, Draper D (2004). Comparing the efficiency of MCMC samplers. In preparation.

Draper D (1996). Utility, sensitivity analysis, and cross-validation in Bayesian model-checking. Comment on "Posterior predictive assessment of model fitness via realized discrepancies," by Gelman A, Meng XL, Stern H, *Statistica Sinica*, **6**, 760–767.

Draper D, Fouskakis D (2000). A case study of stochastic optimization in health policy: problem formulation and preliminary results. *Journal of Global Optimization*, **18**, 399–416.

Draper D, Krnjajic M (2003). 3-fold cross-validation as an approach to Bayesian model selection. In preparation.

Fouskakis D, Draper D (2002). Stochastic optimization: a review. *International Statistical Review*, **70**, 315–349.

Gelfand AE, Dey DK, Chang H (1992). Model determination using predictive distributions, with implementation via sampling-based methods (with discussion). In *Bayesian Statistics 4*, Bernardo JM, Berger JO, Dawid AP, Smith AFM (editors). Oxford University Press, 147–167.

Gelman A, Meng XL, Stern H (1996). Posterior predictive assessment of model fitness via realized discrepancies (with discussion). *Statistica Sinica*, **6**, 733–760.

Hendriksen C, Lund E, Stromgard E (1984). Consequences of assessment and intervention among elderly people: a three year randomised controlled trial. *British Medical Journal*, **289**, 1522–1524.

References (continued)

Key JT, Pericchi LR, Smith AFM (1998). Bayesian model choice: what and why? (with discussion). In *Bayesian Statistics 6*, Bernardo JM, Berger JO, Dawid AP, Smith AFM (editors). Oxford University Press, 343–370.

Leamer EE (1978). *Specification Searches: Ad Hoc Inference with Nonexperimental Data*. New York: Wiley.

Lindley DV (1982). The Bayesian approach to statistics. In *Some Recent Advances in Statistics*, Tiago de Olivera J (editor). London: Academic Press, 65–87.

McCulloch RE, Rossi PE (1992). Bayes factors for nonlinear hypotheses and likelihood distributions. *Biometrika*, **79**, 663–676.

O'Hagan A (1995). Fractional Bayes factors for model comparison. *Journal of the Royal Statistical Society, Series B*, **57**, 99–138.

Spiegelhalter DJ, Smith AFM (1982). Bayes factors for linear and log-linear models with vague prior information. *Journal of the Royal Statistical Society, Series B*, **44**, 377–387.