There is clear evidence that $\sigma_e$ 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 4.5. Comparison of inferential conclusions about the multiplicative effect parameter $e^{\gamma_1}$ from the fixed- and random-effects Poisson regression models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Posterior Mean</th>
<th>Posterior SD</th>
<th>Central 95% Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poisson</td>
<td>0.816</td>
<td>0.0735</td>
<td>(0.683, 0.969)</td>
</tr>
<tr>
<td>REPR</td>
<td>0.830</td>
<td>0.0921</td>
<td>(0.665, 1.02)</td>
</tr>
</tbody>
</table>

Table 4.5 compares the REPR model inferential results with those from model (4), which could also be called a fixed-effects Poisson regression model.

The "error" SD $\sigma_e$ has posterior mean 0.68, give or take about 0.07 (on the log($\lambda$) scale), corresponding to substantial extra-Poisson variability, which translates into increased uncertainty about the multiplicative effect parameter $e^{\gamma_1}$.

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 $100(1 - 0.83)\% = \boxed{17\%}$ give or take about 9% (ironically this conclusion is similar to that from the Gaussian model, but this is coincidence).
4.2 Predictive Diagnostics

Model selection: A Bayesian perspective.
An interest in diagnostics implies a willingness to evaluate and criticize a statistical model, i.e., to discover what (if anything) is deficient about the current "working" model and (if possible) to find a better model.

Thus the real topic here is model selection.

This is an active area of Bayesian research, both for hierarchical and other models (e.g., Leamer 1978, Spiegelhalter and Smith 1982, McCulloch and Rossi 1992, Key et al. 1998).

It's clearly a confusing topic, to judge from the volume of Bayesian papers that descend into ad-hockery (e.g., Aitkin 1991, O'Hagan 1995, Gelman et al. 1996, Berger and Pericci 1996, Bayarri and Berger 1998), which is nature's way of telling you as a Bayesian to think harder about problem formulation.

One of the nice things about Bayes is that there's only one legitimate way forward in any given situation involving a decision that needs to be made: maximization of expected utility based on probabilities updated with Bayes' Theorem.
4.2.1 Model Selection as a Decision Problem

In the case of model selection it would seem self-evident that to choose a model you have to say to what purpose the model will be put, for how else will you know whether your model is good enough?

Specifying this purpose demands a decision-theoretic basis for model choice (e.g., Draper 1996).

To take two examples,

(1) If you’re going to choose which of several ways to behave in the future, then the model has to be good enough to reliably aid you in choosing the best behavior; or

(2) If you wish to make a scientific summary of what’s known, then—remembering that the hallmark of good science is good prediction—the model has to be good enough to make sufficiently accurate predictions of observable outcomes (in which the dimensions along which accuracy is to be monitored are driven by what’s scientifically relevant).
Utility-Based Variable Selection

Draper and Fouskakis (2000) (also see Fouskakis and Draper 2002) give one example of this theory in action, demonstrating that variable selection in regression models should often be governed by the principle that the final model should only contain variables that predict well enough given how much they cost to collect (see Figure 4.4, which compares $2^{14} = 16,384$ models).

![Diagram showing estimated expected utility as a function of number of predictor variables.](image)

Figure 4.4. Estimated expected utility as a function of number of predictor variables, in a problem involving the construction of a cost-effective scale to measure the sickness at hospital admission of elderly pneumonia patients. The best models only have 4–6 sickness indicators out of the 14 possible predictors.
Choosing the Utility Function

Any reasonable utility function in the admission sickness problem will have two components, one quantifying data collection costs associated with the construction of a given sickness scale, the other rewarding and penalizing the scale’s predictive successes and failures.

This requires an intimate knowledge of the real-world consequences of correct choices and mistakes, a level of knowledge that is always desirable but is frequently costly in time and money to acquire.

In practice people often try to get away with a more generic utility function, e.g., one that simply rewards predictive accuracy.

It’s a mistake, however, to use the data twice in measuring this sort of thing (once to make predictions, and again with the same data to see how good they are).

**Out-of-sample predictive validation** (e.g., Gelfand et al. 1992) solves this problem: e.g., successively remove each observation $y_i$ one at a time, construct the predictive distribution for $y_j$ based on $y_{-j}$ (the data vector with $y_j$ removed), and see where $y_j$ falls in this distribution.
Log scoring rule

This motivates the log scoring rule (e.g., Bernardo and Smith 1994): with \( n \) data values \( y_j \) and when choosing among \( k \) models \( M_i, i \in I \), find that model \( M_i \) which maximizes

\[
\frac{1}{n} \sum_{j=1}^{n} \log p(y_j|M_i, y_{-j}).
\]  

(10)

As noted earlier, it’s also revealing to compute predictive \( z \)-scores, for observation \( j \) under model \( i \):

\[
z_{ij} = \frac{y_j - E(y_j|M_i, y_{-j})}{\sqrt{V(y_j|M_i, y_{-j})}}.
\]

(11)

A serious Bayesian modeling problem: Cromwell’s Rule. Lindley (1982) reminds us of what he calls Cromwell’s rule: Anything with zero prior probability has to have zero posterior probability, no matter how the data come out.

This gives Bayesians trouble when applied to model selection: if you pick an initial model before seeing the data, thus giving zero prior probability to huge regions of model space, and the data make you wish that you had enlarged the initial model, what can you do without cheating (using the data twice)?
3CV: 3-Fold Cross-Validation

One possibility: Bayesian nonparametrics (if your prior on model space spans all (scientifically relevant) possibilities then Cromwell's Rule is no longer a problem).

Another possibility (three-fold cross-validation (3CV): Draper and Krnjajic 2003): Taking the cross-validation idea one step further,

1. **Divide** data at random into three (non-overlapping) subsets $S_i$.

2. **Fit** tentative \{likelihood + prior\} to $S_1$. **Expand** initial model in all feasible ways suggested by data exploration using $S_1$. **Iterate** until you're happy.

3. Use final model (fit to $S_1$) from (2) to create predictive distributions for all data points in $S_2$. Compare actual outcomes with these distributions, checking for predictive calibration. Go back to (2), change likelihood as necessary, **retune prior** as necessary, to get good calibration. Iterate until you're happy.

4. Announce **final model** (fit to $S_1$ and $S_2$) from (3), and report predictive calibration of this model on data points in $S_3$ as indication of how well it would perform with new data.
IHGA data revisited

<table>
<thead>
<tr>
<th>Experimental Status</th>
<th>1/4 Model</th>
<th>1/4 Validation</th>
<th>1/4 Projection</th>
<th>Total n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Treatment</td>
<td>143</td>
<td>71</td>
<td>71</td>
<td>285</td>
</tr>
<tr>
<td>Control</td>
<td>143</td>
<td>72</td>
<td>72</td>
<td>287</td>
</tr>
<tr>
<td>Total n</td>
<td>286</td>
<td>143</td>
<td>143</td>
<td>572</td>
</tr>
</tbody>
</table>

\[
\text{log.score.summary} \leftarrow \text{function( treatment, control, n3cv.T, n3cv.C, S, N )} \{ \\
\text{n.T} \leftarrow \text{length( treatment )} \\
\text{n.C} \leftarrow \text{length( control )} \\
\text{log.score1.V} \leftarrow \text{rep( 0, S )} \\
\text{log.score2.V} \leftarrow \text{rep( 0, S )} \\
\text{log.score3.V} \leftarrow \text{rep( 0, S )} \\
\text{log.score4.V} \leftarrow \text{rep( 0, S )} \\
\text{z.summary1.V} \leftarrow \text{matrix( 0, S, 6 )} \\
\text{z.summary2.V} \leftarrow \text{matrix( 0, S, 6 )} \\
\text{z.summary3.V} \leftarrow \text{matrix( 0, S, 6 )} \\
\text{z.summary4.V} \leftarrow \text{matrix( 0, S, 6 )} \\
\text{for ( i in 1:S )} \{ \\
\text{# general stuff for all models} \\
\text{rand.T} \leftarrow \text{sample( 1: n.T )} \\
\text{y.TM} \leftarrow \text{treatment[ rand.T[ 1:n3cv.T[1] ] ]} \\
\text{rand.C} \leftarrow \text{sample( 1: n.C )} \\
\text{y.CM} \leftarrow \text{control[ rand.C[ 1:n3cv.C[1] ] ]} \\
\} 
\]

39
S+ Code (continued)

y.M <- c( y.CM, y.TM )
y.V <- c( y.CV, y.TV )
y.P <- c( y.CP, y.TP )

n.M <- length( y.M )
n.V <- length( y.V )
n.P <- length( y.P )

n.TM <- length( y.TM )
n.CM <- length( y.CM )
n.TV <- length( y.TV )
n.CV <- length( y.CV )
n.TP <- length( y.TP )
n.CP <- length( y.CP )

Stuff specific to model 1

ybar.TM <- mean( y.TM )
s.TM <- sqrt( var( y.TM ) )

ybar.CM <- mean( y.CM )
s.CM <- sqrt( var( y.CM ) )

log.score1.TV <- sum( log( dnorm( y.TV, ybar.TM, s.TM ) ) )
log.score1.CV <- sum( log( dnorm( y.CV, ybar.CM, s.CM ) ) )

log.score1.V[i] <- ( log.score1.TV + log.score1.CV ) / n.V

z.scores1.TV <- ( y.TV - ybar.TM ) / s.TM
z.scores1.CV <- ( y.CV - ybar.CM ) / s.CM

z.scores1.V <- c( z.scores1.TV, z.scores1.CV )

qq1 <- qqnorm( z.scores1.V, plot = F )

z.summary1.V[i,] <- c( mean( z.scores1.V ),
var( z.scores1.V ), sum( abs( z.scores1.V ) > 1.959964 )
/ n.V, sum( abs( z.scores1.V ) > 2.575829 ) / n.V,
sum( abs( z.scores1.V ) > 3.290527 ) / n.V,
cor( qq1$x, qq1$y ) )
S+ Code (continued)

# stuff specific to model 2

r.M <- sum( y.M )


pred.mean2 <- r.M / n.M

pred.SD2 <- sqrt( r.M * ( 1.0 + 1.0 / n.M ) / n.M )

z.scores2.V <- ( y.V - pred.mean2 ) / pred.SD2

qq2 <- qqnorm( z.scores2.V, plot = F )

z.summary2.V[i,] <- c( mean( z.scores2.V ),
var( z.scores2.V ), sum( abs( z.scores2.V ) > 1.959964 ) / n.V,
sum( abs( z.scores2.V ) > 2.575829 ) / n.V,
sum( abs( z.scores2.V ) > 3.290527 ) / n.V,
cor( qq2$x, qq2$y ) )

# stuff specific to model 3

r.TM <- sum( y.TM )

log.score3.TV <- sum( log( dpg( y.TV, r.TM, n.TM, 1 ) ) )

r.CM <- sum( y.CM )

log.score3.CV <- sum( log( dpg( y.CV, r.CM, n.CM, 1 ) ) )

log.score3.V[i] <- ( log.score3.TV + log.score3.CV ) / n.V

pred.meanT3 <- r.TM / n.TM

pred.SDT3 <- sqrt( r.TM * ( 1.0 + 1.0 / n.TM ) / n.TM )

z.scores3.TV <- ( y.TV - pred.meanT3 ) / pred.SDT3

pred.meanC3 <- r.CM / n.CM

pred.SDC3 <- sqrt( r.CM * ( 1.0 + 1.0 / n.CM ) / n.CM )

z.scores3.CV <- ( y.CV - pred.meanC3 ) / pred.SDC3

z.scores3.V <- c( z.scores3.TV, z.scores3.CV )

qq3 <- qqnorm( z.scores3.V, plot = F )
S+ Code (continued)

z.summary3.V[i,j] <- c(mean(z.scores3.V),
            var(z.scores3.V), sum(abs(z.scores3.V) > 1.959964)
/ n.V, sum(abs(z.scores3.V) > 2.575829) / n.V,
            sum(abs(z.scores3.V) > 3.290527) / n.V,
cor(qq3$x, qq3$y))

# stuff specific to model 4

write(c(rep(0, n.CM), rep(1, n.TM)), "poisson-x.dat",
       ncol = 20, append = F)
write(y.M, "poisson-y.dat", ncol = 20, append = F)

junk <- system("backbugs poisson4.cmd")

temp <- matrix(scan("bugs.out"), 2 * N, 2, byrow = T)

pred.dist.x0.data <- temp[1:N, 2]
top.0 <- max(pred.dist.x0.data)
pred.dist.x0 <- hist(pred.dist.x0.data, breaks = (-0.5 +
(0:(top.0 + 1)) ), plot = F)$counts / N
pred.dist.x0 <- pred.dist.x0 + min(pred.dist.x0[
            pred.dist.x0 > 0.0 ])
pred.dist.x0 <- pred.dist.x0 / sum(pred.dist.x0)

pred.dist.x1.data <- temp[(N + 1):(2 * N), 2]
top.1 <- max(pred.dist.x1.data)
pred.dist.x1 <- hist(pred.dist.x1.data, breaks = (-0.5 +
(0:(top.1 + 1)) ), plot = F)$counts / N
pred.dist.x1 <- pred.dist.x1 + min(pred.dist.x1[
            pred.dist.x1 > 0.0 ])
pred.dist.x1 <- pred.dist.x1 / sum(pred.dist.x1)

log.score4.CV <- sum(log(pred.dist.x0[1 + y.CV]))
log.score4.TV <- sum(log(pred.dist.x1[1 + y.TV]))

pred.meanC4 <- mean(pred.dist.x0.data)
pred.SDC4 <- sqrt(var(pred.dist.x0.data))
z.scores4.CV <- (y.CV - pred.meanC4) / pred.SDC4

pred.meanT4 <- mean(pred.dist.x1.data)
pred.SDT4 <- sqrt(var(pred.dist.x1.data))
z.scores4.TV <- (y.TV - pred.meanT4) / pred.SDT4
z.scores4.V <- c( z.scores4.TV, z.scores4.CV )

qq4 <- qqnorm( z.scores4.V, plot = F )

z.summary4.V[i,] <- c( mean( z.scores4.V ),
                      var( z.scores4.V ), sum( abs( z.scores4.V ) > 1.959964 )
                      / n.V, sum( abs( z.scores4.V ) > 2.575829 ) / n.V,
                      sum( abs( z.scores4.V ) > 3.290527 ) / n.V,
                      cor( qq4$x, qq4$y ) )

write( signif( c( i, log.score1.V[i], z.summary1.V[i,], log.score2.V[i], z.summary2.V[i,], log.score3.V[i], z.summary3.V[i,], log.score4.V[i], z.summary4.V[i,] ), 7 ), "poisson4.raw", ncol = 29, append = ( i > 1 ) )

}


}

dpg <- function( y, alpha, beta, n ) {

    return( exp( alpha * log( beta ) + lgamma( alpha + y ) + y * log( n ) - lgamma( alpha ) - lgamma( y + 1 ) - ( alpha + y ) * log( beta + n ) ) )

}
BUGS code

compile( "poisson4.bug" )
update( 500 )
monitor( y.x0.new )    # This is
monitor( y.x1.new )    # poisson4.cmd
update( 10000 )
q()

############################################################################

model poisson4;

const    # And this is
          # poisson4.bug
    n = 286;

var

    x[n], y[n], gamma.0, gamma.1, tau.e, x.bar, e[n],
    lambda[n], lambda.C, lambda.E, mult.effect, sd.e,
    e.x0.new, lambda.x0.new, y.x0.new, e.x1.new,
    lambda.x1.new, y.x1.new;

data x in "poisson-x.dat", y in "poisson-y.dat";
inits in "poisson4.in";

{

    gamma.0 ~ dnorm( 0.0, 0.0001 );
    gamma.1 ~ dnorm( 0.0, 0.0001 );
    tau.e ~ dgamma( 0.0001, 0.0001 );

    x.bar <- mean( x[] );

    for (i in 1:n) {

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

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

e.x0.new ~ dnorm( 0.0, tau.e );
log( lambda.x0.new ) <- gamma.0 + gamma.1 * ( 0.0 - x.bar )
   + e.x0.new;
y.x0.new ~ dpois( lambda.x0.new );

e.x1.new ~ dnorm( 0.0, tau.e );
log( lambda.x1.new ) <- gamma.0 + gamma.1 * ( 1.0 - x.bar )
   + e.x1.new;
y.x1.new ~ dpois( lambda.x1.new );

}
## Results

Simulation SEs in parenthesis

### model 1: Gaussian

<table>
<thead>
<tr>
<th>log score</th>
<th>mean</th>
<th>var</th>
<th>z(0.05)</th>
<th>z(0.01)</th>
<th>z(0.001)</th>
<th>corr</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td>-1.552</td>
<td>0.005086</td>
<td>1.050</td>
<td>0.05637</td>
<td>0.02647</td>
<td>0.01358</td>
</tr>
<tr>
<td>(0.001)</td>
<td>(0.001469)</td>
<td>(0.004)</td>
<td>(0.00027)</td>
<td>(0.00024)</td>
<td>(0.00015)</td>
<td>(0.0003)</td>
</tr>
</tbody>
</table>

This is OK on mean and variance grounds but (from z(0.01) and z(0.001)) does not have heavy enough tails.

### model 2: Poisson with common lambda

<table>
<thead>
<tr>
<th>log score</th>
<th>mean</th>
<th>var</th>
<th>z(0.05)</th>
<th>z(0.01)</th>
<th>z(0.001)</th>
<th>corr</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td>-1.316</td>
<td>0.002293</td>
<td>1.507</td>
<td>0.07700</td>
<td>0.03417</td>
<td>0.02951</td>
</tr>
<tr>
<td>(0.001)</td>
<td>(0.001755)</td>
<td>(0.004)</td>
<td>(0.00027)</td>
<td>(0.00021)</td>
<td>(0.00022)</td>
<td>(0.0003)</td>
</tr>
</tbody>
</table>

Better log score but MUCH worse var z-score and % unusual z-values

### model 3: Poisson with different lamdas

<table>
<thead>
<tr>
<th>log score</th>
<th>mean</th>
<th>var</th>
<th>z(0.05)</th>
<th>z(0.01)</th>
<th>z(0.001)</th>
<th>corr</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td>-1.314</td>
<td>0.004297</td>
<td>1.493</td>
<td>0.07438</td>
<td>0.04352</td>
<td>0.02318</td>
</tr>
<tr>
<td>(0.001)</td>
<td>(0.001750)</td>
<td>(0.004)</td>
<td>(0.00030)</td>
<td>(0.00028)</td>
<td>(0.00020)</td>
<td>(0.0003)</td>
</tr>
</tbody>
</table>

Evidently (from the z-score variances) the Poisson is underdispersed

### model 4: random-effects Poisson regression

<table>
<thead>
<tr>
<th>log score</th>
<th>mean</th>
<th>var</th>
<th>z(0.05)</th>
<th>z(0.01)</th>
<th>z(0.001)</th>
<th>corr</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td>-1.18</td>
<td>0.0013</td>
<td>1.02</td>
<td>0.052</td>
<td>0.011</td>
<td>0.002</td>
</tr>
<tr>
<td>(0.001)</td>
<td>(0.0016)</td>
<td>(.004)</td>
<td>(.003)</td>
<td>(.002)</td>
<td>(.001)</td>
<td>(.001)</td>
</tr>
</tbody>
</table>

This model fits really well.
4.3 References


4.3 References (continued)


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 \( \theta' \) of the parameter vector \( \theta \), if the output of your MCMC sampler for \( \theta' \) behaves like an AR\(_1\) series with first-order **autocorrelation** \( \rho_1 \), 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 \( \hat{\sigma}_{\theta'} \), and first-order **autocorrelation** \( \hat{\rho}_1 \) as **estimates** of the corresponding summaries for \( \theta' \).

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

\[
2\Phi^{-1}\left(1 - \frac{\alpha}{2}\right) \frac{SE(\bar{\theta'})}{\hat{\sigma}_{\theta'}} \leq 10^{b-k+1}, \tag{12}
\]

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

\[
SE(\bar{\theta'}) = \frac{\hat{\sigma}_{\theta'}}{\sqrt{m}} \sqrt{\frac{1 + \hat{\rho}_1}{1 - \hat{\rho}_1}}, \tag{13}
\]

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

\[
m \geq 4 \left[ \Phi^{-1}\left(1 - \frac{\alpha}{2}\right) \right]^2 \left(\frac{\hat{\sigma}_\theta}{10^{b-k+1}}\right)^2 \left(\frac{1 + \hat{\rho}_1}{1 - \hat{\rho}_1}\right). \tag{14}
\]

This is referred to in the **MLwiN** documentation as the **Brooks-Draper** diagnostic (Brooks and Draper 2002).

**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 \( \mu \), on the micrograms below 10g scale, was \( \tilde{\mu} = 404.3 \) (to 4 sigfis); the other relevant quantities for \( \mu \) were as follows: posterior SD \( \tilde{\sigma}_\mu = 0.464 \) and first-order autocorrelation \( \tilde{\rho}_1 = 0.294 \) (NB the MCSE for \( \mu \) is already down to 0.009 with 5,000 iterations, so I already have a bit more than 4 sigfis 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 sigfis with at least 95% Monte Carlo probability for the posterior mean—write \( \tilde{\mu} = 4.043 \cdot 10^2 \), so that \( b = 2 \), take \( \alpha = 0.05 \) and substitute into (14) to yield

\[
m \geq 4(1.96)^2 \left( \frac{0.464}{10^{2-5+1}} \right)^2 \left( \frac{1 + 0.294}{1 - 0.294} \right) \approx 60,600.
\] (15)

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 \( \mu \) 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 sigfis of accuracy you would write \( \tilde{\mu} = 3.0 \cdot 10^{-1} \) and obtain

\[
m \geq 4(1.96)^2 \left( \frac{0.464}{10^{-1-2+1}} \right)^2 \left( \frac{1 + 0.294}{1 - 0.294} \right) \approx 60,600.
\] (16)

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 sigfis—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 \((10^1)^2 = 100\).
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 $\theta'$ of the parameter vector $\theta$, and that the output of your MCMC sampler for $\theta'$ behaves like an $AR_1$ series with first-order autocorrelation $\rho_1$; 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 $\tilde{\sigma}_{\theta'}$, and first-order autocorrelation $\hat{\rho}_1$ as estimates of the corresponding summaries for $\theta'$.

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

$$\sqrt{SE(\tilde{\sigma}_{\theta'})} = \frac{\tilde{\sigma}_{\theta'}}{\sqrt{2m}} \sqrt{\frac{1 + \hat{\rho}_1^2}{1 - \hat{\rho}_1^2}}.$$  \hfill (17)

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

$$\frac{\sqrt{SE(\bar{\theta}')}}{\sqrt{SE(\tilde{\sigma}_{\theta'})}} = \sqrt{\frac{2(1 + \hat{\rho}_1)^2}{1 + \hat{\rho}_1^2}},$$  \hfill (18)

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 tail is).