Applied Bayesian Methods

Michael Escobar
University of Toronto
m.escobar@utoronto.ca
www.utstat.utoronto.ca\escobar

Example 11: One Week Mortality

An investigator is interested in looking at the effect of a type of beta-blocker versus a placebo on the week mortality of subjects following myocardial infarction.[a]. The next slide contains the data for the study:

[a]Data from Makuch, Stephens, Escobar, 1989, *The Statistician*, 61-70.

Beta Blocker	Placebo
3/26	4/23
1/47	6/48
3/46	1/35
1/33	1/15
2/35	4/71
3/73	6/187
29/238	24/242
18/698	

Number dead at 1 week/Number of patients entered on study.

Model for Binomial Outcomes

This model is an extension of the beta-binomial model.

- First, consider the following notation:
  - Let Y_{ij} be the number of deaths for the j^{th} treatment at site i.
  - Let n_{ij} be the total number of patients enrolled for the j^{th} treatment at site i.
- Following the discussion in lecture 1 , we assume that Y_{ij} has a binomial distribution with parameters p_{ij} and n_{ij}, where p_{ij} is the probability of death for treatment j and site i.
- Using the beta conjugate prior, let p_{ij} have a beta prior distribution with parameters a_j and b_j.

Model for Binomial Outcomes

(continued)

- Since we are interested in the difference between the treatments, we are then interested in the differences between the distributions for p_{i1} versus the distribution for p_{i2}. Also, since we have several sites for each treatment, we are in a position to estimate something about the distributions for p_{i1} and p_{i2}. Therefore, we cannot fix the parameters a_1, b_1, a_2, and b_2. We need to put a prior on these parameters.

- Let the two a_j's have a gamma prior with parameters d_a and σ and let the two b_j's have a gamma prior with parameters d_a and σ.

Model for Binomial Outcomes

(continued)

This results in the following model:

$$\begin{aligned} y_{ij}|n_{ij}, p_{ij} &\sim \text{Bin}(y_{ij}|p_{ij}, n_{ij}) \\ p_{ij}|a_j, b_j &\sim \text{Beta}(p_{ij}|a_j, b_j) \\ a_j|d_a, \sigma &\sim \text{Gam}(a_j|d_a, \sigma) \\ b_j|d_b, \sigma &\sim \text{Gam}(b_j|d_b, \sigma) \end{aligned}$$

Where y_{ij} and n_{ij} are data value and the parameters d_a, d_b, and σ are given fixed values to finalize the prior.

Model for Binomial Outcomes

Comments:

- Note that we can define the mean μ_j of the p_{ij}'s as:

  $$\mu_j = \frac{a_j}{a_j + b_j}$$

  and the precision θ_j as $\theta_j = a_j + b_j$.

- In making inferences about the treatment, one is probably interested in looking at the difference between the μ_j's for inferences about the treatment effect. That is, one is probably interested in $\mu_1 - \mu_2$.

Alternative Parametrization

- A new addition to this model is the use of the gamma distribution as a prior to the a_j and b_j parameters. If someone is to use this model, then it is important to understand the what these gamma distribution are doing if one is going to claim that these gamma distributions are at least approximating ones prior belief.
- To understand the effect of these gamma distributions, it is helpful to see what these distributions imply about the distribution of the parameters μ_j and θ_j. Under this model, the parameter μ_j has a beta distribution with parameters d_a and d_b and the parameter θ_j has a gamma distribution with parameters $d_a + d_b$ and σ.

Alternative Parametrization

(continued)

- Given the elicitation information in the first lecture this morning, one could probably derive meaningful values for d_a, d_b, and σ. In the worked examples in this talk, the parameters d_a and d_b are chosen to be 1 and σ is chosen to be .001.

Code for Beta-Binomial Model

The following is the BUGS syntax for the model that we are discussing:

```
model{
for(j in 1:T){
  for(i in 1:N){
    y[i,j] ~ dbin( p[i,j], n[i,j])
    p[i,j] ~ dbeta(a[j], b[j])
    }
  mu[j] <- a[j]/(a[j]+b[j])
  theta[j] <- a[j] + b[j]
  a[j] ~ dgamma( 1, .001)
  b[j] ~ dgamma( 1, .001)
  }
}
```

Alternative Beta-Binomial Model

- Note that in the above BUGS code, the nodes `a[j]` and `b[j]` are stochastic nodes. That is, BUGS recognizes them as random variables in BUGS and the nodes `mu[j]` and `theta[j]` are deterministic nodes. That is, they are simply a function of stochastic nodes and other deterministic nodes. The MCMC sampler operates on the stochastic nodes and then redefines the values of the deterministic nodes based on the values of the stochastic nodes.

Alternative Beta-Binomial Model

- It is possible to reparametrize this model and switch which are stochastic nodes and which are the deterministic nodes. That is, we can make `mu[j]` and `theta[j]` stochastic nodes and make and the `a[j]` and `b[j]` deterministic nodes. Then, the MCMC algorithm will sample new values for `mu[j]` and `theta[j]` and then redefine `a[j]` and `b[j]` according to the value of the stochastic nodes.
- The next slide contains the code for the alternative parametrization of this model. Please note that both codes represent the same theoretical model. However, they are sampled differently.

Code for Alternative Parametrization

The following is the BUGS syntax for the model with the alternative parametrization:

```
model{
for(j in 1:T){
  for(i in 1:N){
    y[i,j] ~ dbin( p[i,j], n[i,j])
    p[i,j] ~ dbeta(a[j], b[j])
    }
  mu[j] ~ dbeta(1,1)
  theta[j] ~ dgamma(2,.001)
  a[j] <- mu[j]*theta[j]
  b[j] <- (1-mu[j])*theta[j]
  }
}
```

Third model for Mortality Example

- Again, assume that Y_{ij} is from a binomial distribution with parameters p_{ij} and n_{ij}.
- In the last model, instead of putting priors on the parameters of the two beta distributions (that is, the parameters a_j and b_j), the parameters are first transformed and then the prior is put on the parameters μ_j and θ_j.
- Here, consider simply putting a different prior on the parameters p_{ij}. So, first, transform the parameters using the logit transform (this is $\log[x/(1-x)]$). So, define $\lambda_{ij} = \log(p_{ij}/(1-p_{ij}))$. Then put a prior on λ_{ij}. For simplicity, let us put a normal prior. That is, let λ_{ij} have a normal prior distribution with mean δ_j and precision τ.
- Now, let us put a the usual type of priors on δ_j and τ. That is, let the prior for δ_j be normal with mean δ_0 and precision τ_0.

Let the prior for both τ and τ_0 be a gamma distribution and the prior for δ_0 be normal.

- Note, in the BUGS code, the parameters for the priors for τ and τ_0 are given the value .001 and .001, and the prior for δ_0 is given a mean of 0 and precision of .001.

Third Model of Mortality

So, with the assumption from the previous slide, here is the third model for the mortality:

$$
\begin{aligned}
y_{ij}|n_{ij},p_{ij} &\sim \text{Bin}(y_{ij}|p_{ij},n_{ij}) \\
\lambda_{ij} &= \log\left[\frac{p_{ij}}{1-p_{ij}}\right] \\
\lambda_{ij}|\delta_j,\tau &\sim N(\lambda_{ij}|\delta_j,\tau) \\
\delta_j|\delta_0,\tau_0 &\sim N(\delta_j|\delta_0,\tau_0) \\
\delta_0|\delta_{00},\tau_{00} &\sim N(\delta_0|\delta_{00},\tau_{00}) \\
\tau|A,B &\sim \text{Gam}(\tau|A,B) \\
\tau_0|A_0,B_0 &\sim \text{Gam}(\tau_0|A_0,B_0)
\end{aligned}
$$

Code for Third Model

The following is the BUGS syntax for the third model:

```
model{
for(j in 1:T){
  for(i in 1:N){
    y[i,j] ~ dbin( p[i,j], n[i,j])
    logit(p[i,j]) <- lambda[i,j]
    lambda[i,j] ~ dnorm(delta[j],tau)
    }
  delta[j] ~ dnorm(delta0, tau0)
  }
delta0~dnorm(0,.001)
tau ~ dgamma(.001,.001)
tau0~dgamma(.001,.001)
}
```

Diagnosing Convergence

- An overview of the methods
  - Main problems
  - Simple methods
  - Complex methods
  - Standard error, burn-in, and thinning
  - Some advice
- An illustration of these methods on the Beta-Binomial data.

Main Problems

Some bad things:

- Slow mixing: High autocorrelation, cross correlation, "ridges" of posterior mass.
- Multimodes/ multiple clusters. The chain does not easily move between different pockets of mass.

Other issues:

- How long to "burn in".
- Have I run the chain long enough? How accurate are the estimates.
- Should I "thin" the chain?

Some Simple Methods

- Trace plots
- Autocorrelation, plots of lag-correlations
- Cross correlation
- Compare estimates from parallel chains.

Some Complex Methods

- Gelman and Rubin: Based on ANOVA methods. Involves looking at several parallel chains. Look at the between chain and within chain variances to see if the chain has stabilized.
- Geweke: Based on time series methods. Look at one long chain and see if different sections of the chain have the same estimated values. The estimated values are compared with a t-statistic. The variance used in the t-statistics is based on the spectral density (there by controlling for the non-indepedence of the sample).

Standard error, burn-in, and thinning

- Different ways of getting standard error of estimates: a) batch means, b) spectral density, c) from the autocorrelation function.
- Burn-in refers to the practice of “throwing away” the beginning of the chain. This is done because it takes awhile for the chain to settle down and for it to be sampling from the approximate stationary distribution.
- Thinning is the practice of keeping every m-th value. This is done to reduce the autocorrelation of the values that are kept. Therefore, the “effective sample size” is near the number of samples that are kept. Might be an issue if one is storing the chain, etc.

Model Diagnostics

Advice from Carlin and Louis (1996) and Cowles and Carlin (JASA, 1996):

- Use a variety of diagnostic tools. They are designed to catch different problems.
- Run a few parallel chains (perhaps 3 or 5) with different starts.
- Visually inspect these chains.
- Use Gelman-Rubin diagnostic and also look for autocorrelation.
- Investigate cross correlation. Look for collinearity (like regression diagnostics).
- Also: use different algorithms for particular models. Look at different models for same data.
- Understand Your Data.

Software for Diagnostics

- BOA is a package that runs in both Splus or R.
  - It has a very good manual with a very useful summary of the various procedures and how to interpret the various diagnostics. Manual available at: `http://www.public-health.uiowa.edu/boa`.
  - A bit "mousey". That is, best to run with a menu, and that can be tedious.
- Also, for R, there is the R package "coda" which is available through the R project. Easier to run in command line and call up the functions you need once chains converted to "mcmc" class or "mcmc.list" class.

Diagnostics for Mortality Data

Okay, now let us look at the mortality example again and first perform some simple diagnostics on this data.

- As a reminder, This data contained the number of mortalities one week from a heart attack compared to the number of people enrolled in the study. There are several sites in the study. Each site is given one of two treatments.
- The data is modelled as a binomial. The parameter of the binomial for each study, p_{ij} has a beta prior with parameters a_j and b_j. The parameters a_j and b_j are of interest and there is a prior distribution on these values. So, one wishes to find the posterior distribution of (a_j, b_j).

The Two Models

Also, in the way of reminder, note that there are two different parametrization.

- In one (called the ‘regular” model here), the prior distribution on (a_j, b_j) takes the form of independent gamma distributions on the parameters.
- The other parametrization (the alternative parametrization), uses a prior distribution on two functions of (a_j, b_j). Defining $\mu_j = a_j/(a_j + b_j)$ and $\theta_j = a_j + b_j$, the parameter μ_j has a beta prior and θ_j has a gamma prior.

First, let us look at some preliminary results from the regular model. BUGS is run and there are three chains produced.

Simple Diagnostics for Regular Model

- One of the first things to do is to simply get a look at the raw data over time.
- One can get this in a one dimensional way with a trace plot (which is the values of a MCMC chain, $X^{(m)}$, versus m the iteration number).
- Alternatively, one can get a small 2 dimensional view with a path plot (where two sampled parameters are plot against each other and having the points connected in sequential order).

The next few slides show these plots for treatment group 1.

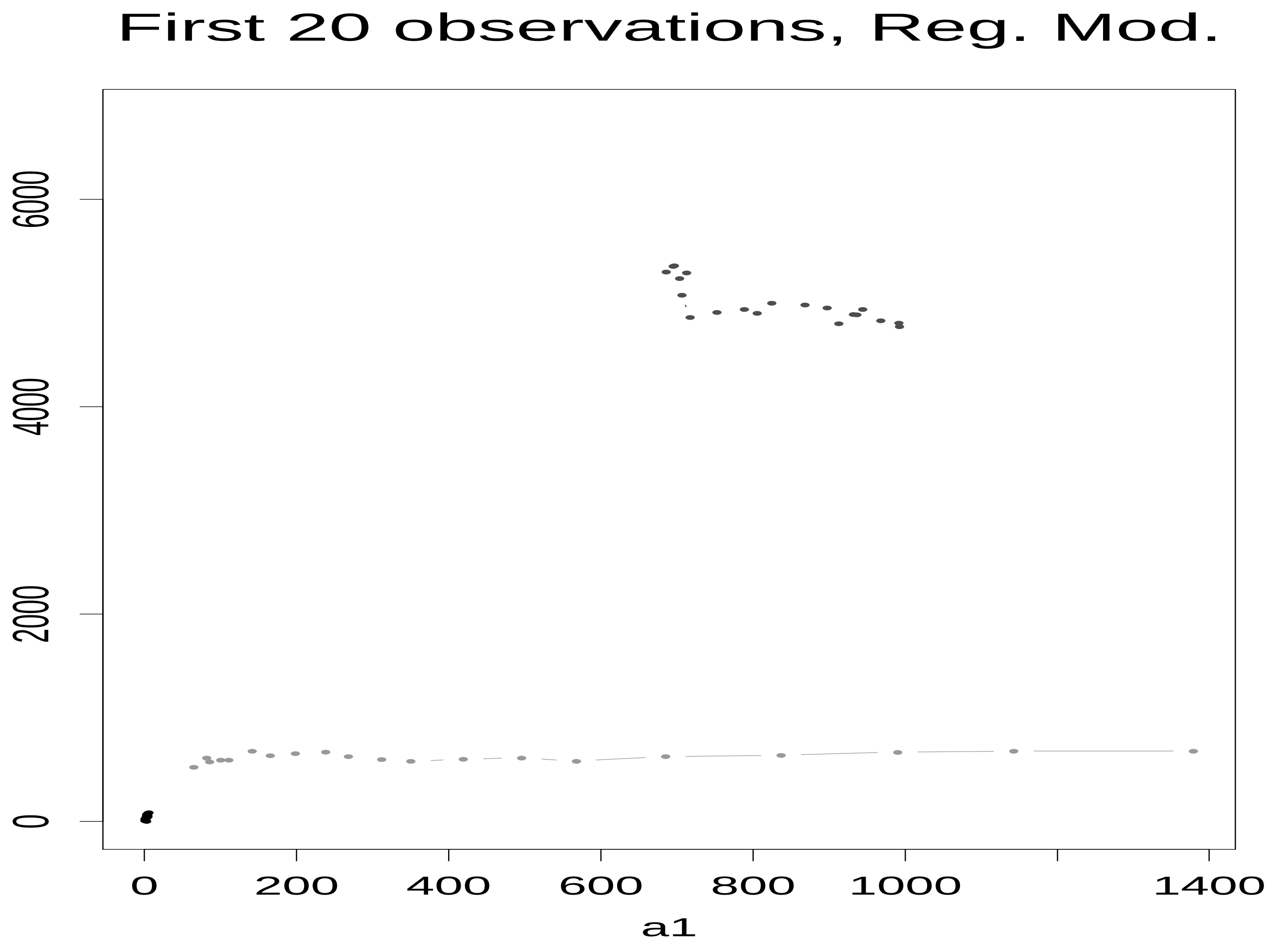

Figure 1: Path plot for a_1 and b_1 under the regular model.

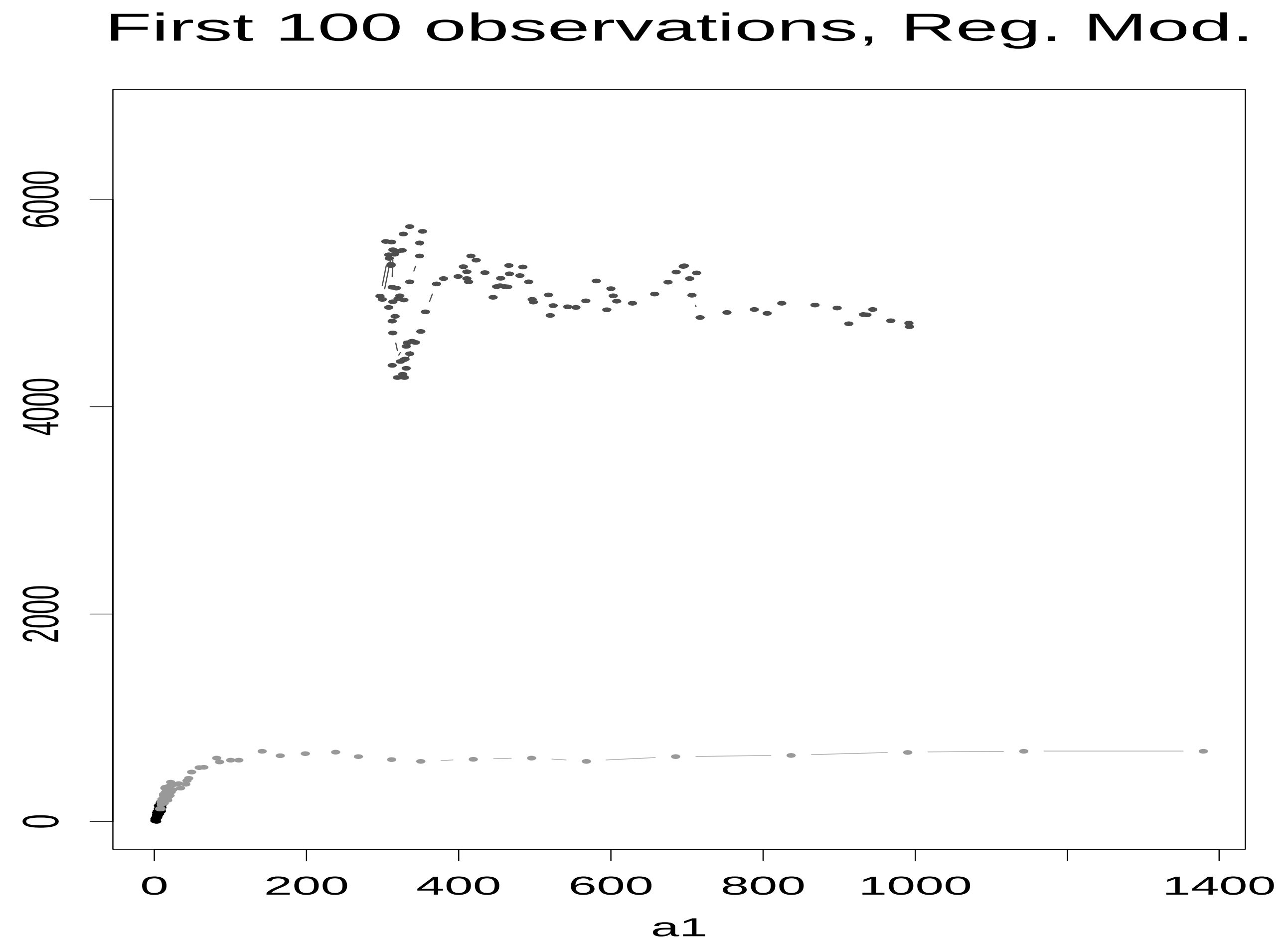

Figure 2: Path plot for a_1 and b_1 under the regular model.

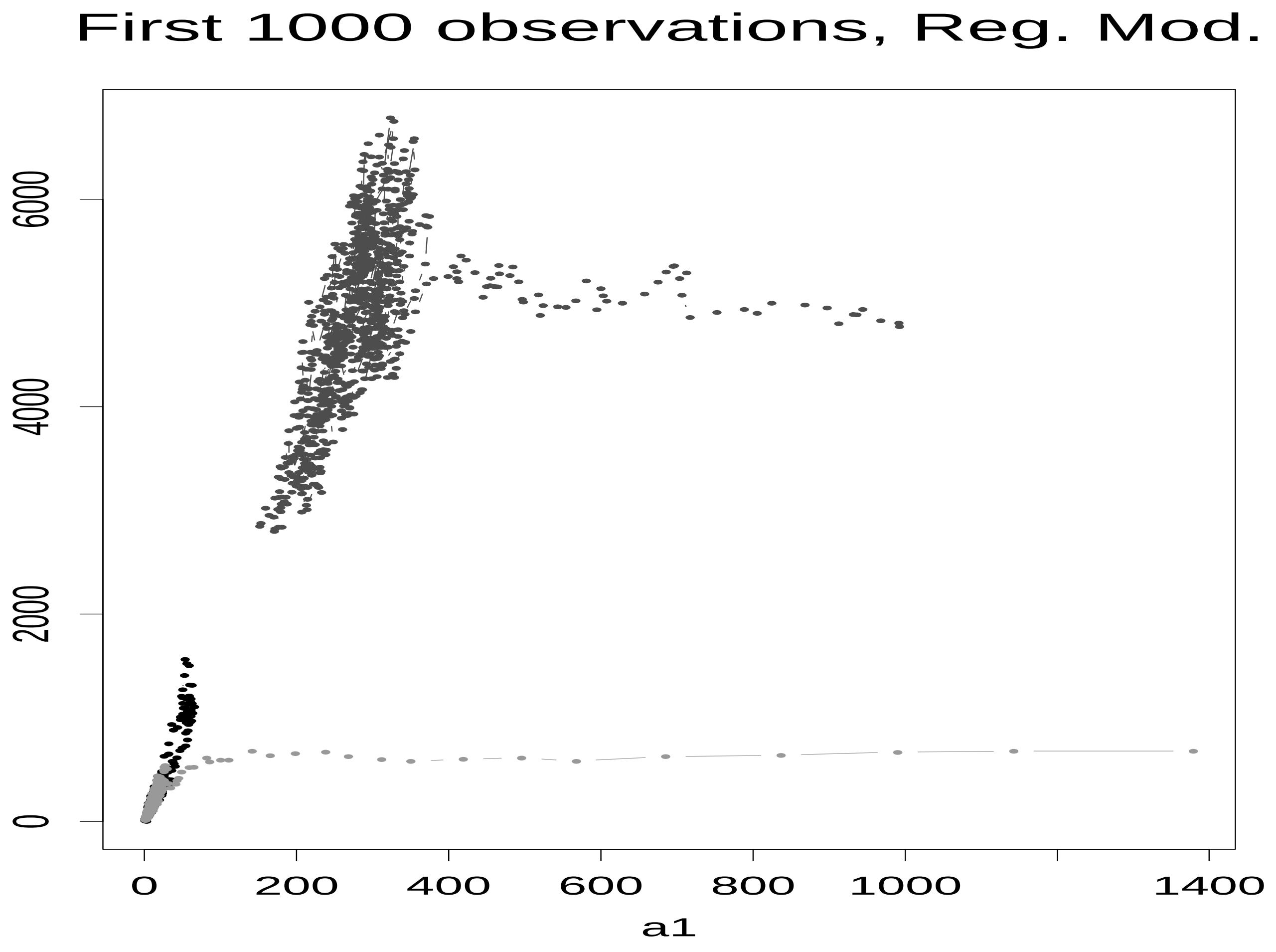

Figure 3: Path plot for a_1 and b_1 under the regular model.

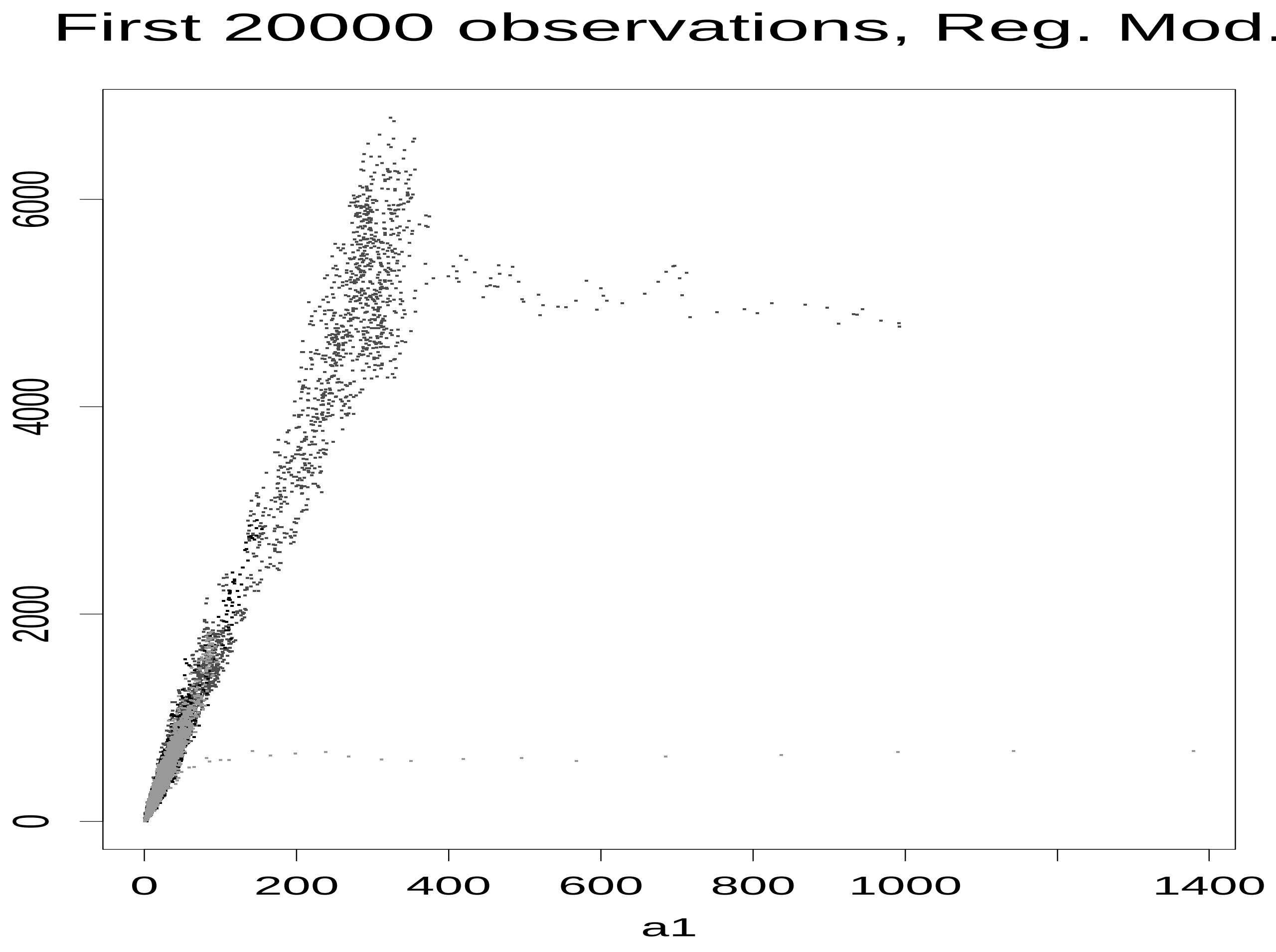

Figure 4: Path plot for a_1 and b_1 under the regular model.

Comments of Path Plots

- When we look at the first 20 samples from the Regular model, we see the three chains going to very different places. Even for the plots showing the first 100 and 1000 samples, the three chains have not joined up yet. It takes 20,000 observations before the regions of the three chains seem to intersect.
- Also, from the plot of 20,000 samples, it appears that the mass is bunched up towards the lower values and there are only a few scatter plots out in the extreme end. This “comet” looking plot usually suggest a log transformation of both variables.

So, next, let us try these same plots with the log transformations.

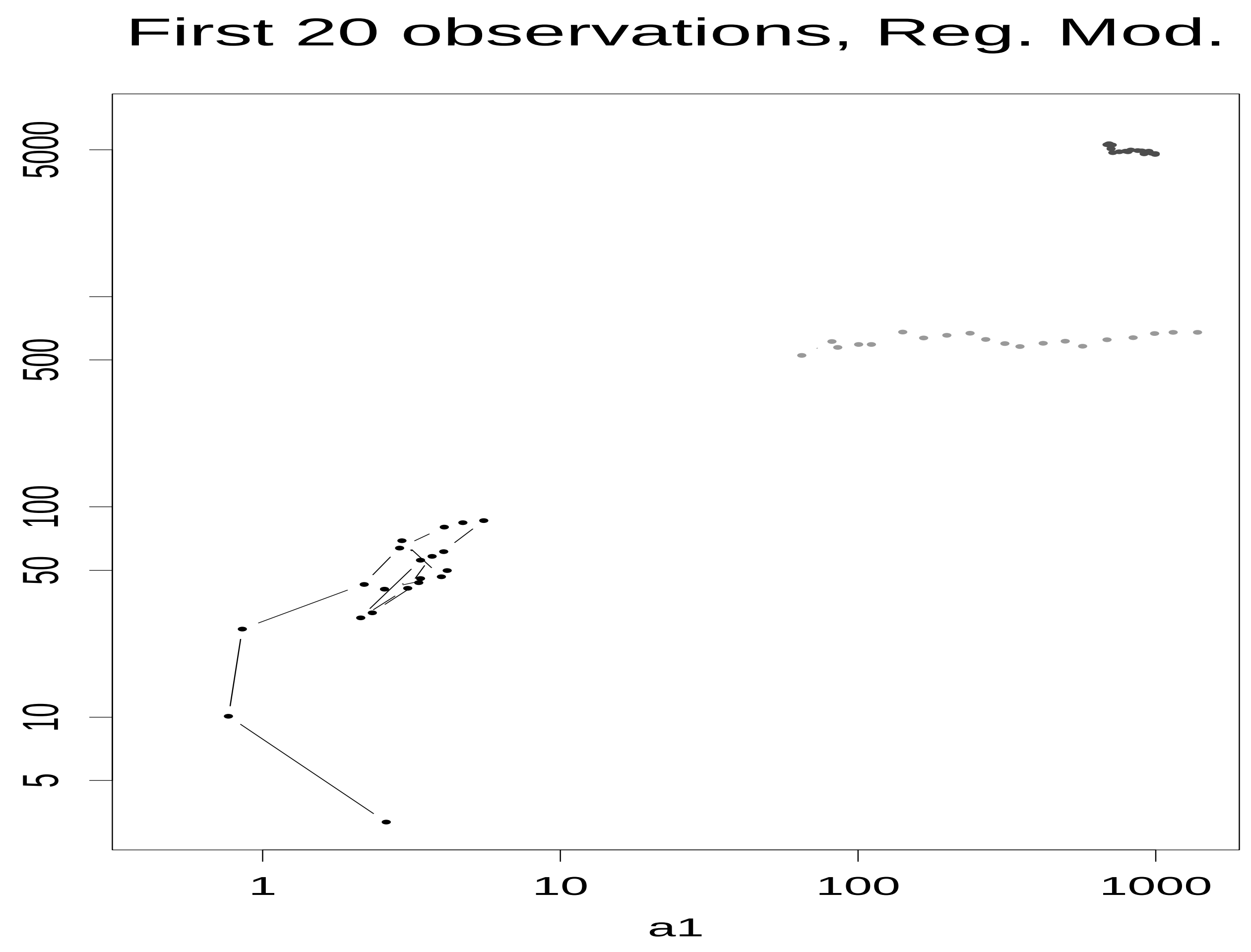

Figure 5: Path plot for a_1 and b_1 under the regular model.

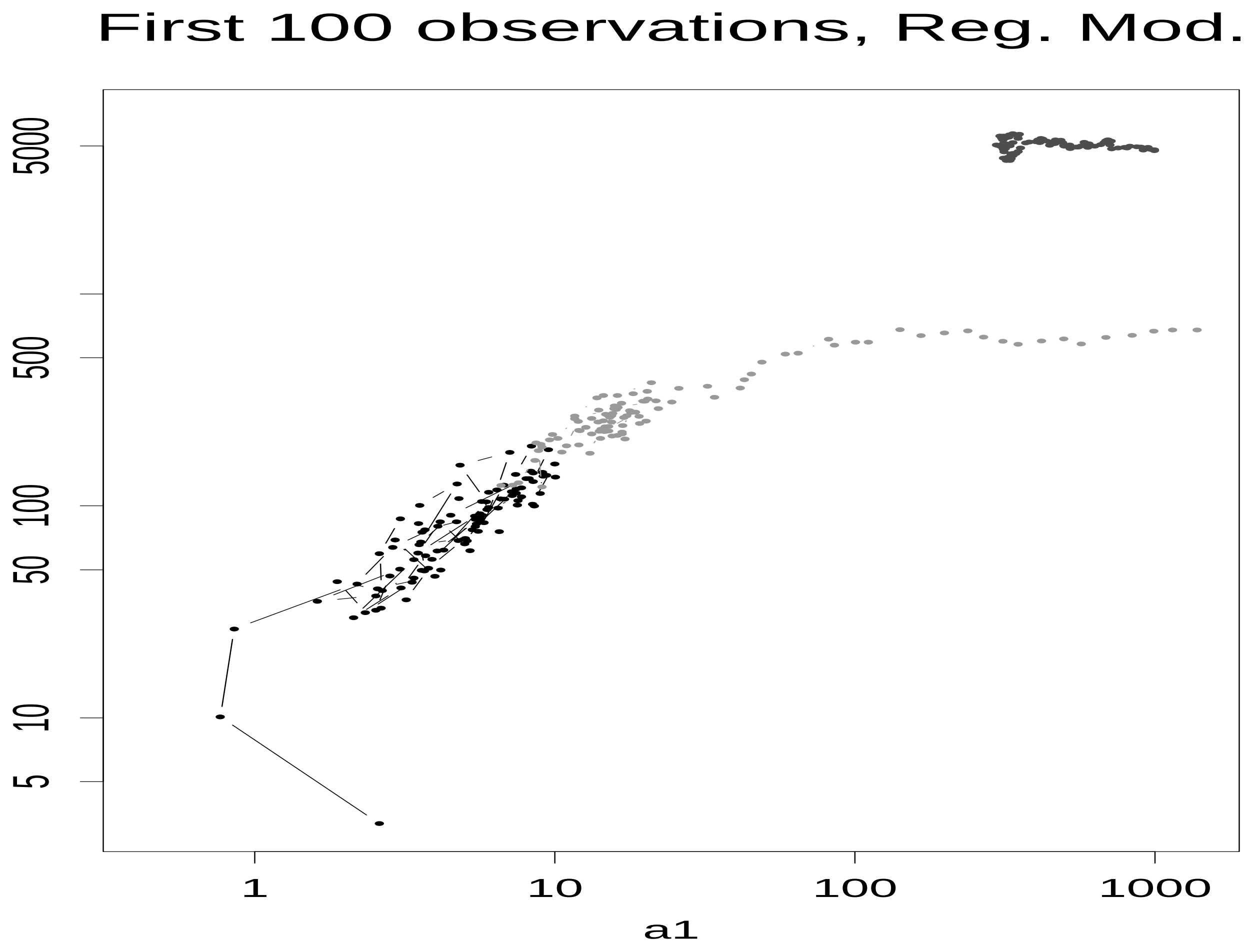

Figure 6: Path plot for a_1 and b_1 under the regular model.

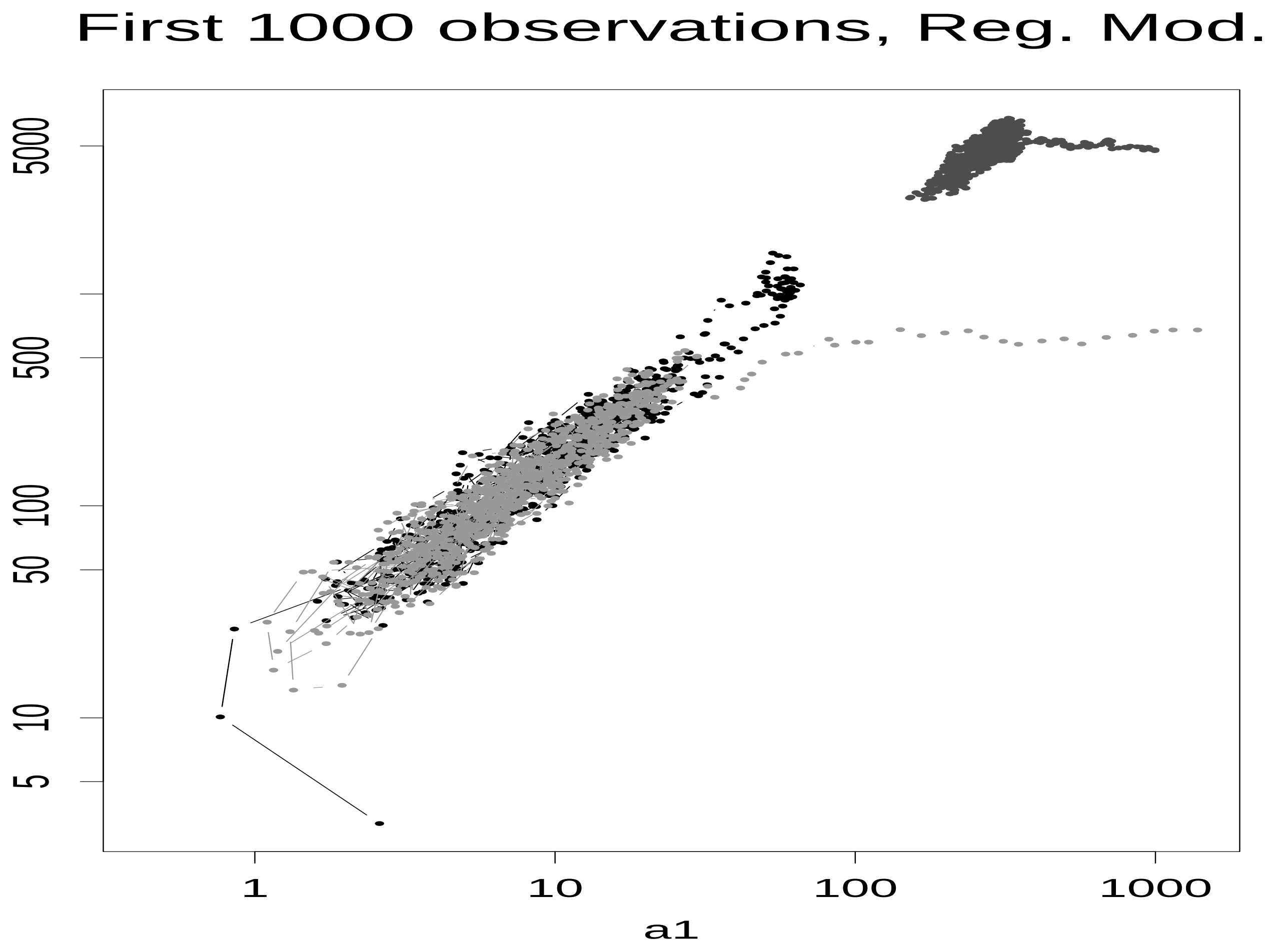

Figure 7: Path plot for a_1 and b_1 under the regular model.

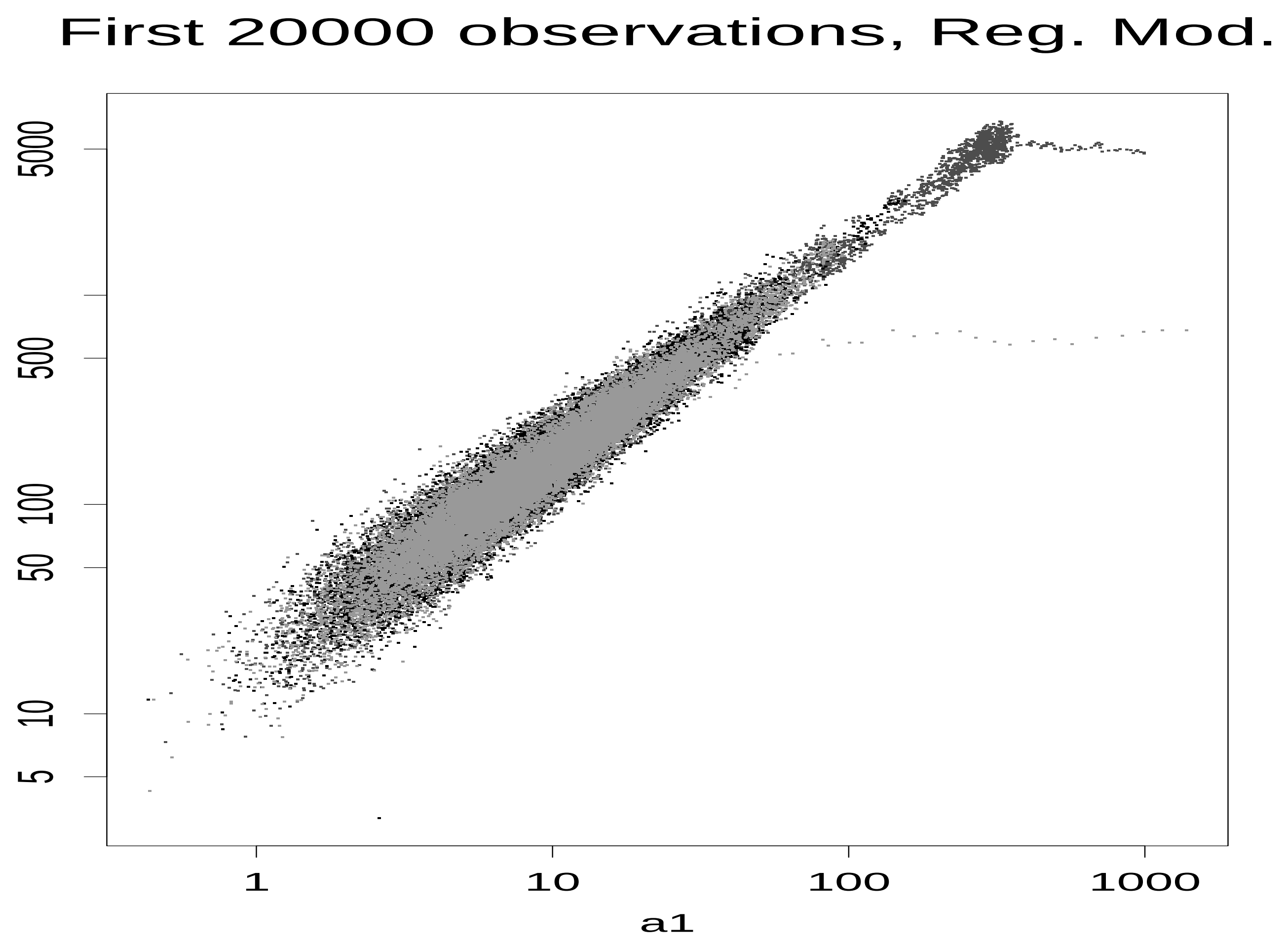

Figure 8: Path plot for a_1 and b_1 under the regular model.

Log transformed Path Plots

- In these path plots we see that the two lower chains join up first, and then the last chain joins up between 1000 and 20,000 iterations.
- To get a better idea when the third chain joins, a trace plot is useful. Remember a trace plot is a plot of the parameter at each iteration versus the iteration number.
- Below is the trace plots for both a_1 and b_1 with and without a log transform.

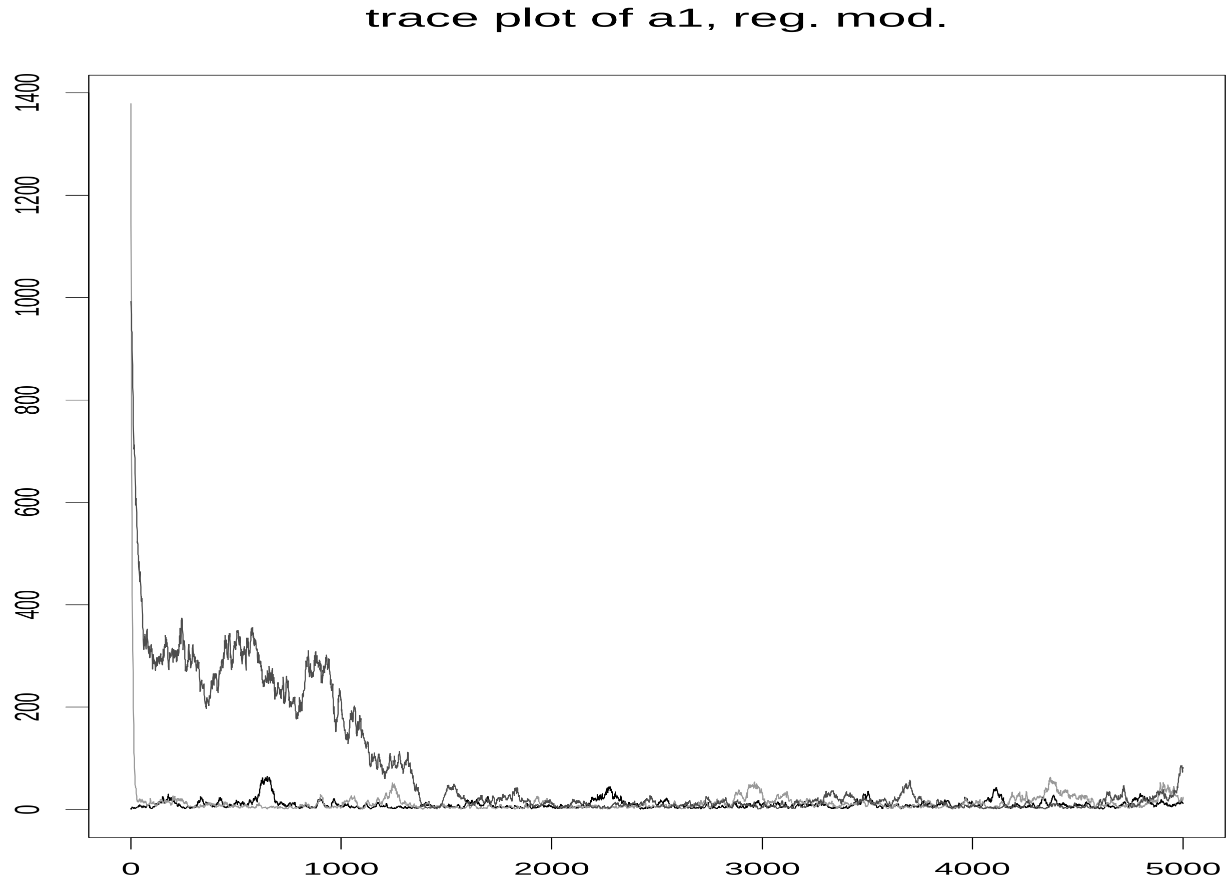

Figure 9: Trace plots for a_1 under the regular model.

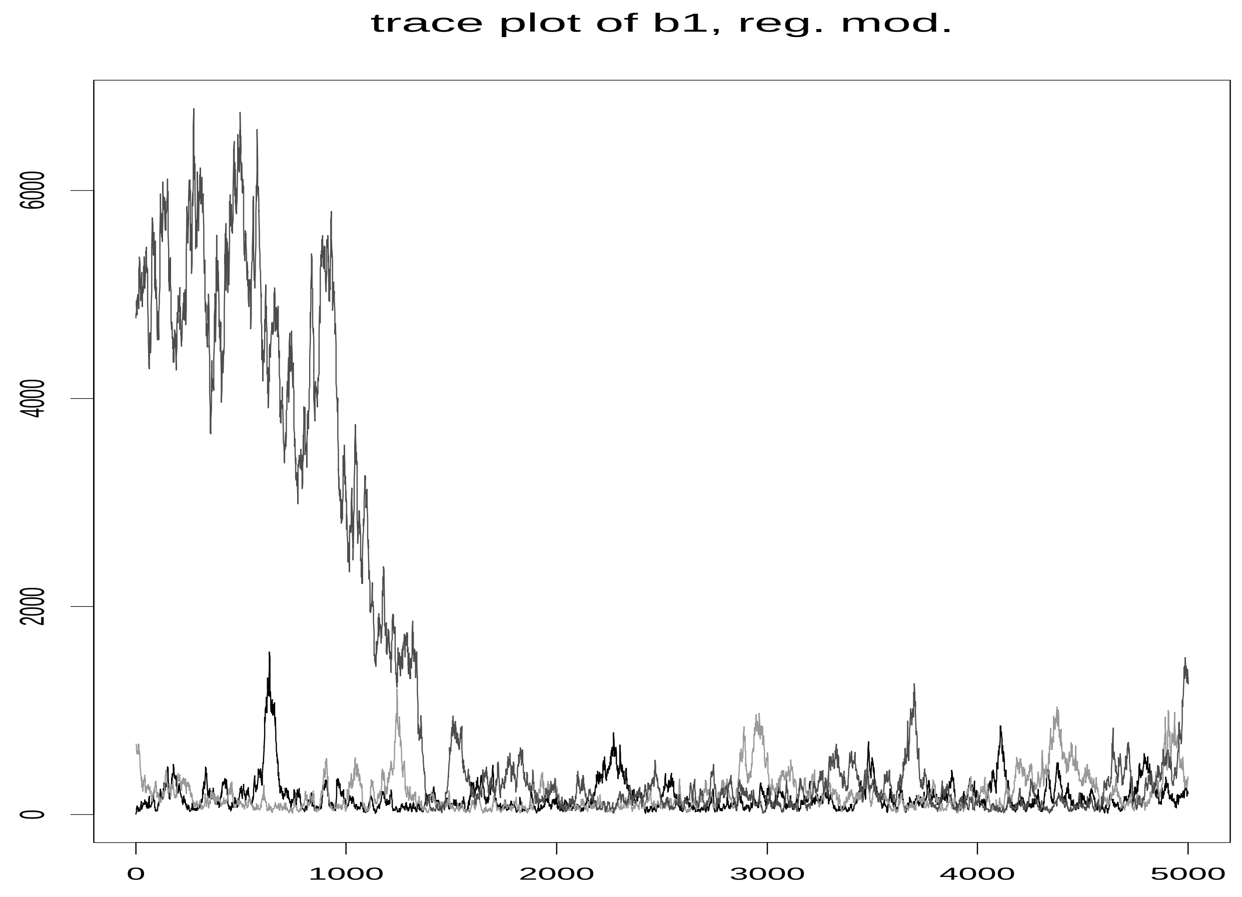

Figure 10: Trace plots for b_1 under the regular model.

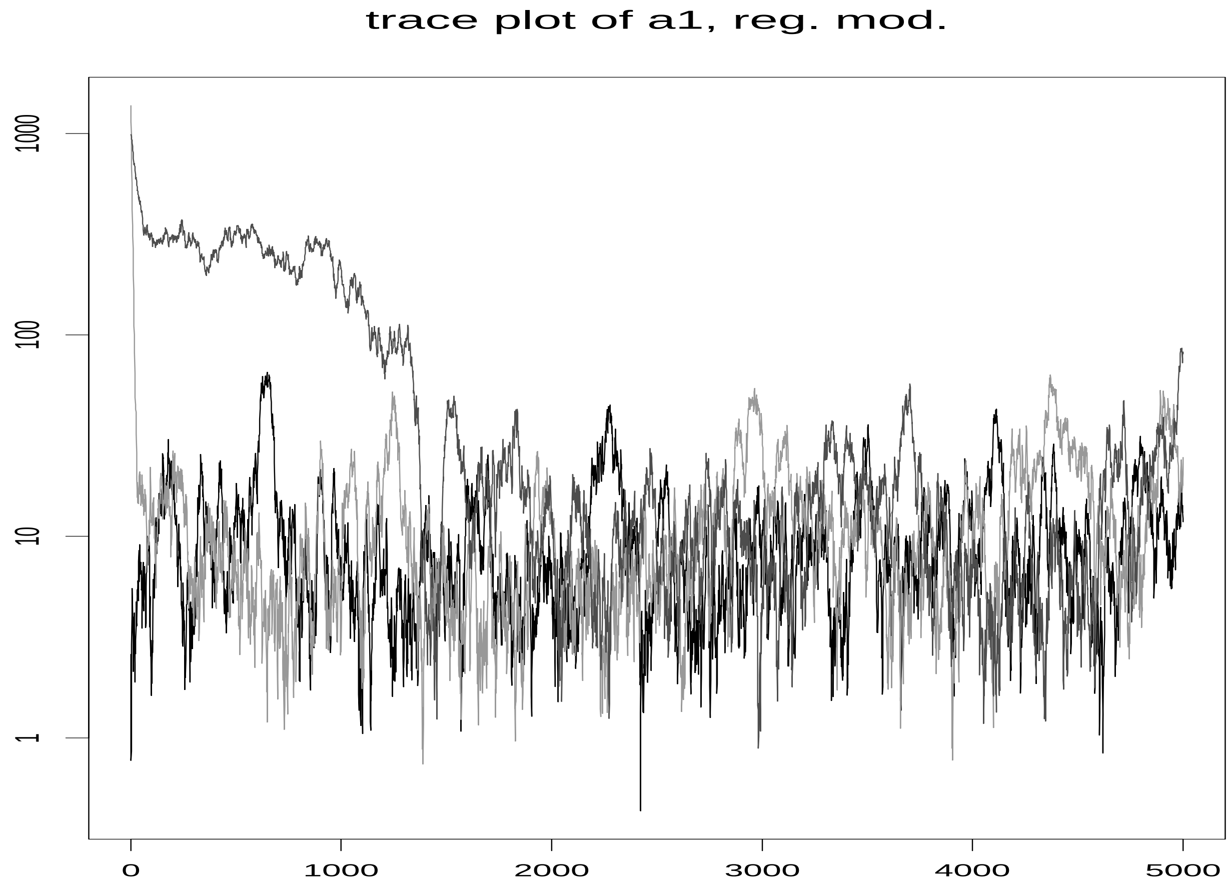

Figure 11: Trace plots for a_1 under the regular model. The parameters are log transformed.

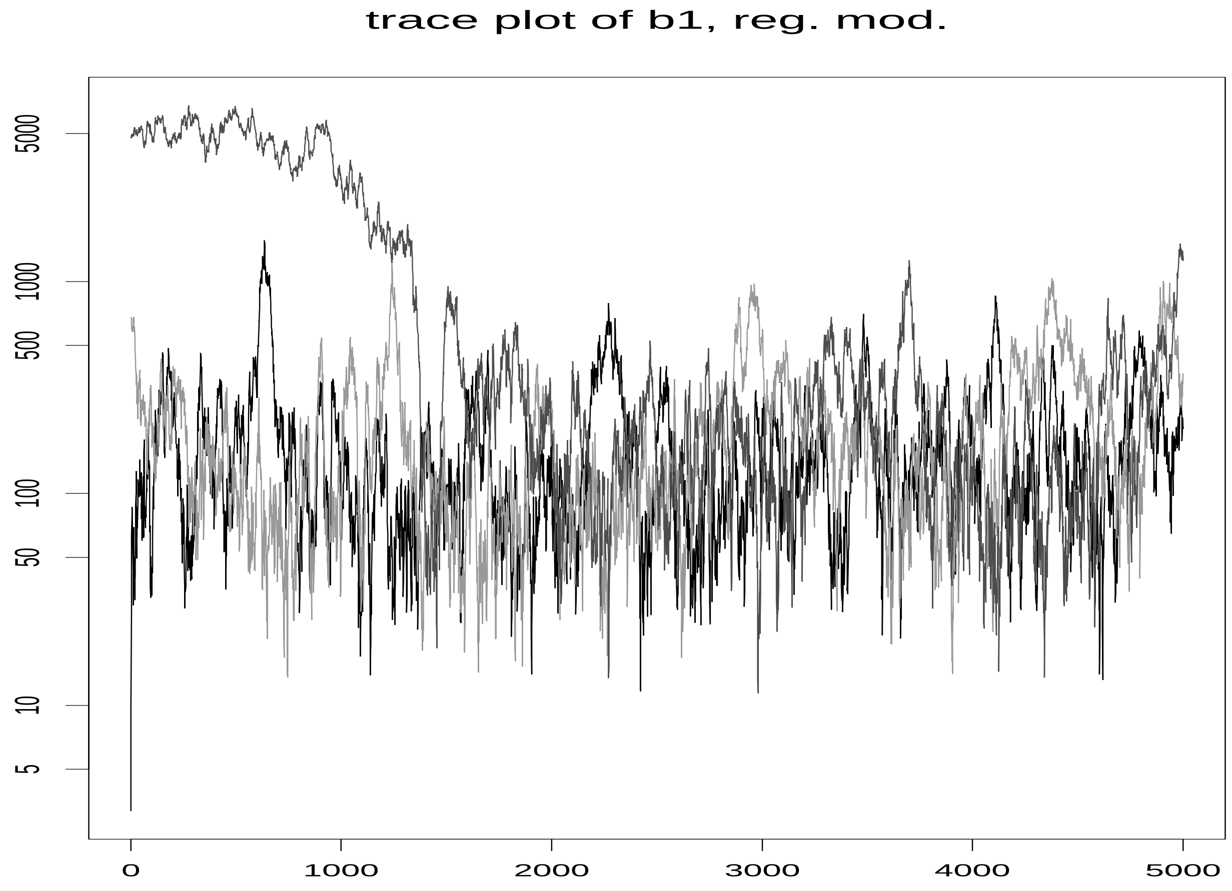

Figure 12: Trace plots for b_1 under the regular model. The parameters are log transformed.

- From the trace plots, we see that the third chain joins the other two chain before the 2,000 iteration. Therefore, we might want to throw away at least the first 2,000 iterations since these samples were taken from the main probability mass of the limiting distribution, but instead these earlier iterations were strongly influenced by the starting the values.
- Please note that here we see the advantage of the trace plot over the path plot. With the trace plot it is easier to see when the different chains hook up. The path plot is better at showing the shape of the joint posterior distribution.

The Autocorrelation of the Samples

- Now that we have a sense that it takes about 2000 samples before the sampler settles in and gives us values which are like the limiting distribution, the next thing to consider is how fast the chain mixes. That is, it is good if the chain cycles through the main mass of probability at a high frequency. If the chain takes a long time to move around, then it means that we should run the sampler a long time to compensate for the slow movement.

- A good measure for how quickly the chain moves is the autocorrelation function. For a sequence of random variables X_i, the correlation function correlation of X_i with its l^{th} lag. That is, the autocorrelation at lag l is the correlation of X_i with X_{i-l}.

The Autocorrelation of the Samples

(Continued)

- If the after a few lags, the autocorrelation is almost zero, then that would desirable. It is true that correlation does not imply independence, but if there is correlation then one does not have independence.

The Autocorrelation of the Samples

(Continued)

- If the samples are correlated, then the estimates derived from such a sample are not as efficient as having a sample from an independent sample. In fact, in many area of statistics, one talks about "effective sample size". If an estimate from a 1000 samples from an dependent sample has about the same standard error as a sample of 600 independent samples, then one would say that the effective sample size of the dependent sample is 600. In the same way, if our sample from the MCMC had high autocorrelations, then our effective sample size from the MCMC might be much smaller than the number of iterations that we had.

The Autocorrelation of the Samples

(Continued)

- The autocorrelation function is usually easy to obtained. Besides the option to have BUGS calculate it for you, most modern statistical packages will have it as part of their time series functions. In Splus and R it is the function `acf`.
- Below are plots of the autocorrelation function for a_1 and b_1 with and without log transformation. In all these analysis, the first 2000 iterations are thrown out as part of the "burn-in".

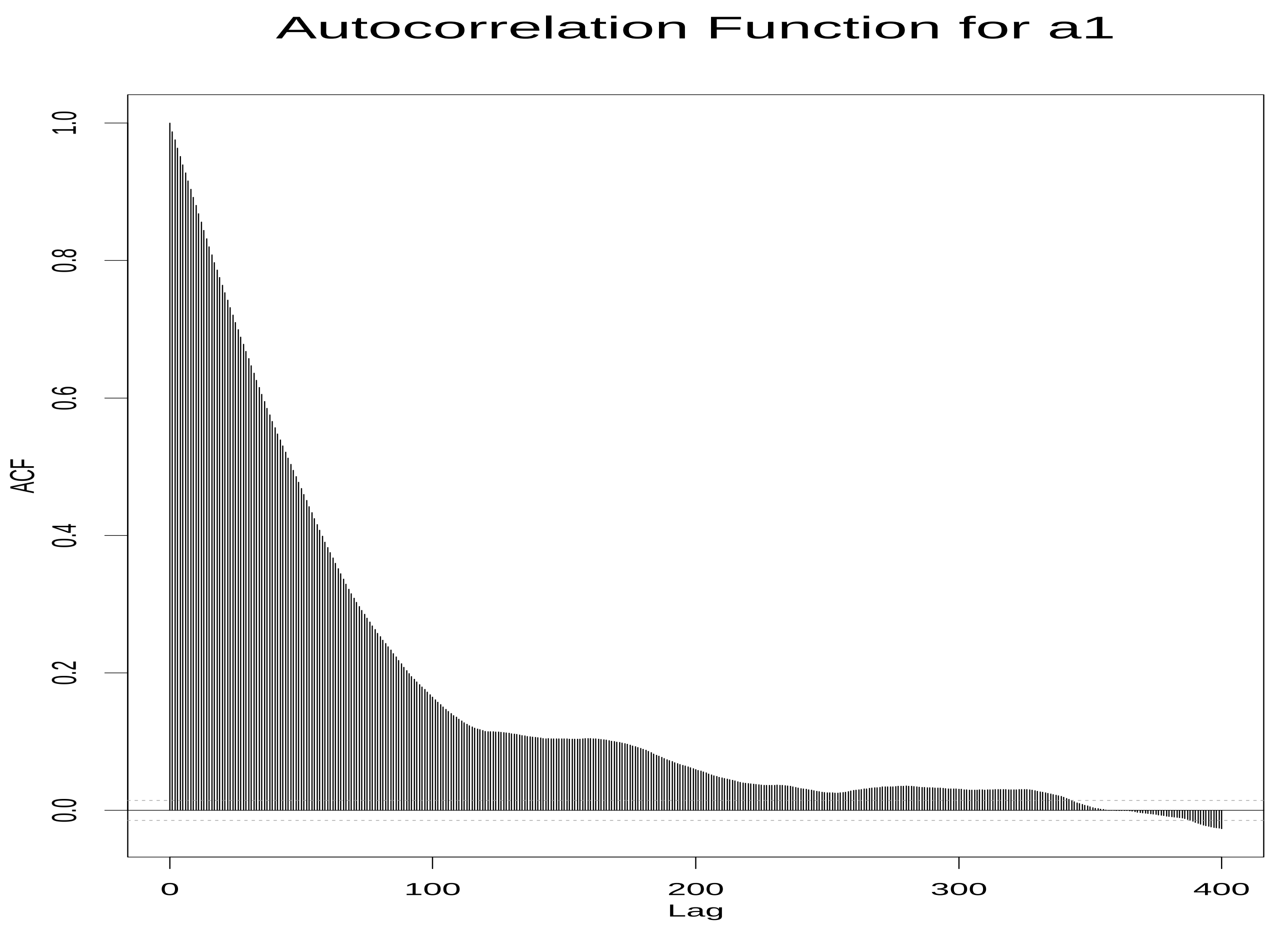

Figure 13: Autocorrelation functions.

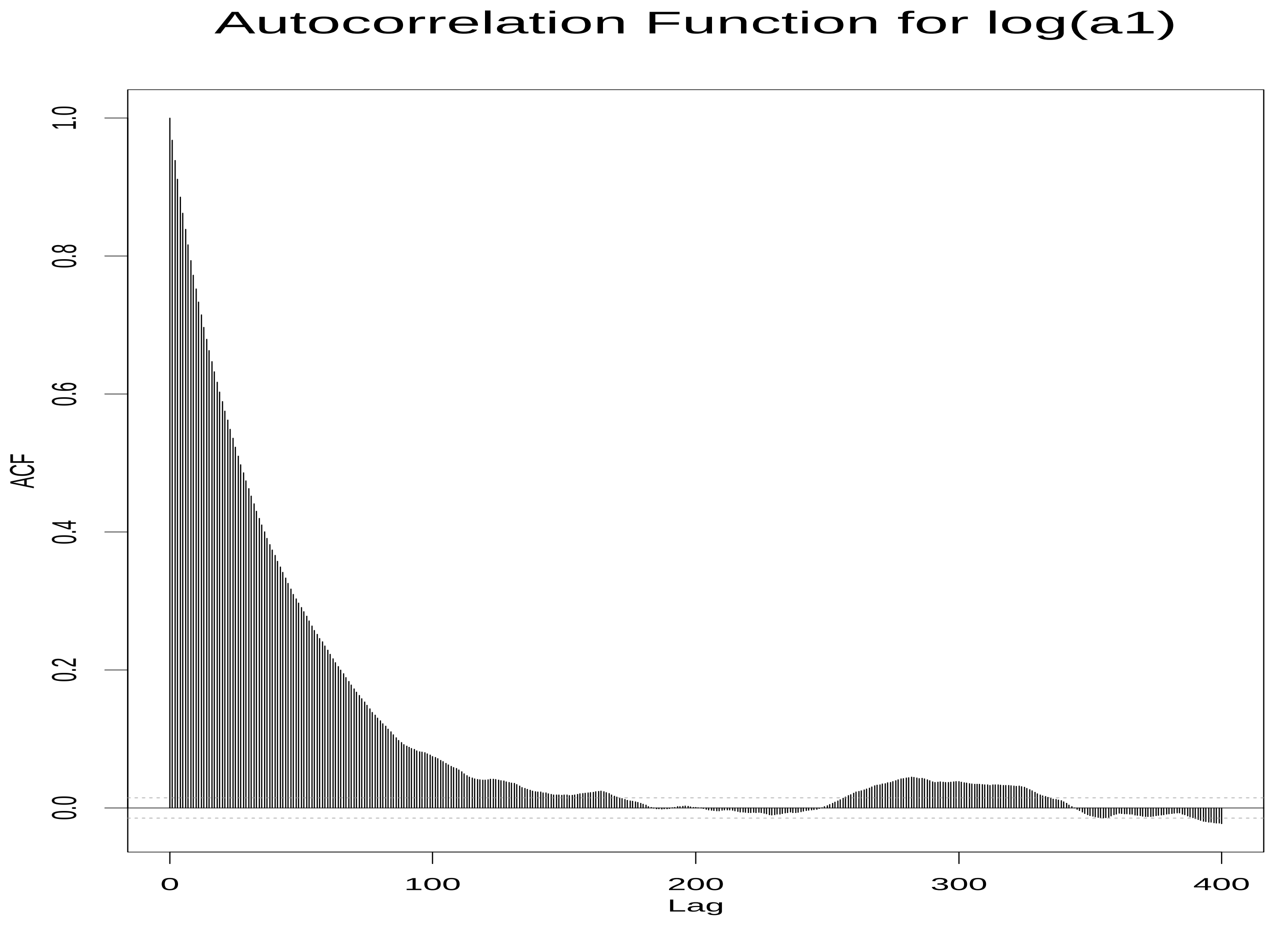

Figure 14: Autocorrelation functions.

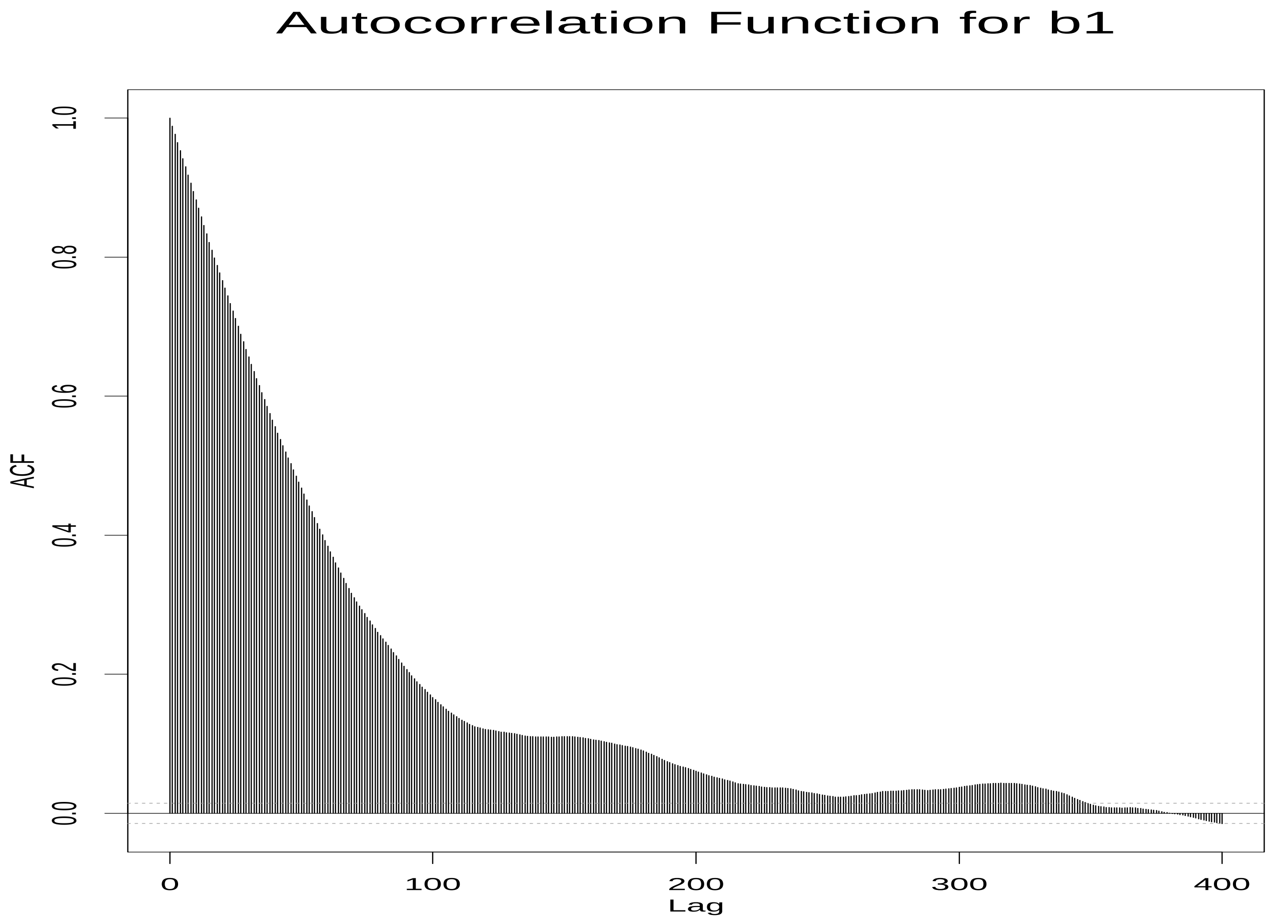

Figure 15: Autocorrelation functions.

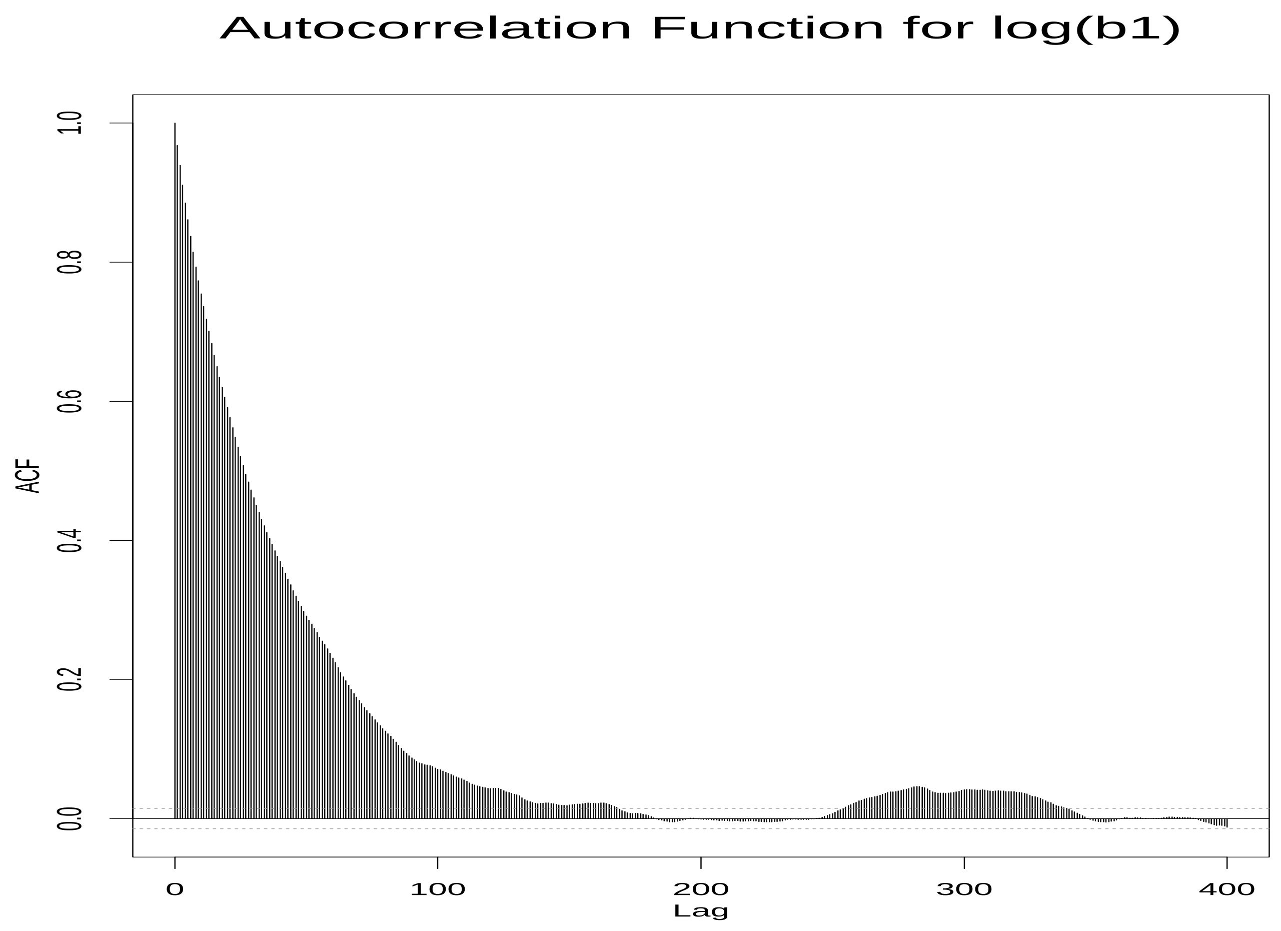

Figure 16: Autocorrelation functions.

Autocorrelation: Comments

- From the autocorrelation plots for the log transformed parameters, we see that the autocorrelation does not get small until about lag 100 and does not in the range of about zero until about lag 200.
- The autocorrelation is even larger for the none transformed parameters, however, because the none transformed parameters are so highly skewed, those estimates are more suspect.

Autocorrelation: Comments

(Continued)

Now, one could get samples from this chain which have near zero autocorrelation at lag one.

- To do this, one could “thin” the chain. That is, one could systematically only take every k^{th} sample. This would result in a shorter chain but a chain which is less internal dependence.
- Since you are throwing away some information, you are not going to be as accurate.

Autocorrelation: Comments

(Continued)

- If there are difficulties with computer storage or processing time, there could be large benefits in that area with minimal loss of information. For example, in the present chain, one might consider sampling a million values. There might be storage difficulties and there might be computational difficulties with a million values. Instead, if one took only every 200th value, then one would have a sample of 5,000 values which were not autocorrelated.
- The next slide is the autocorrelation function log(a1) of the chained that has been thinned by taking every 200th value.

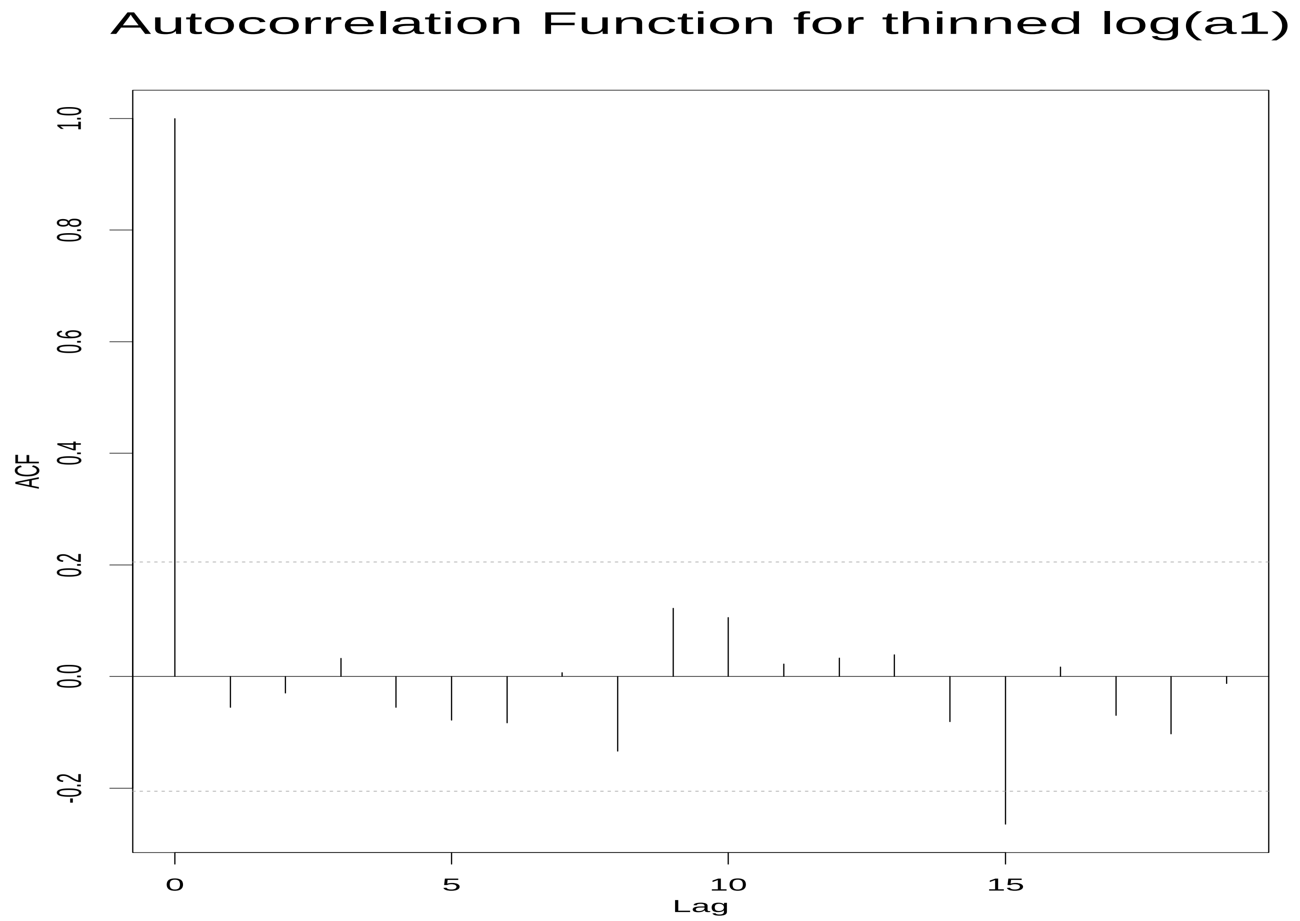

Figure 17: Autocorrelation functions for the thinned chain. Note that the first lag is lag one for the thinned chain but is lag 200 for the originally chain. (Hence the x-legend states: "lag*200").

3 Chain Estimates

To end our tour of the simple methods, lets try and estimate some values of interest using the three sampled chains.

- Some of the interesting values which we might like to know the posterior distribution might be a_1, μ_1, and maybe $\mu_1 - \mu_2$. So, let us estimate the posterior distribution of each of these parameters.
- We have been collected value of a_1, b_1, a_2, and b_2. So, it is easy to get the density of a_1. (We simple get a density estimate of the sampled a_1 values, after throwing away the "burn-in" values.) The rest are easy if we remember that: $\mu_1 = a_1/(a_1 + b_1)$ and $\mu_1 - \mu_2 = a_1/(a_1 + b_1) - a_2/(a_2 + b_2)$.
- The next set of slides contains an estimate of these values.

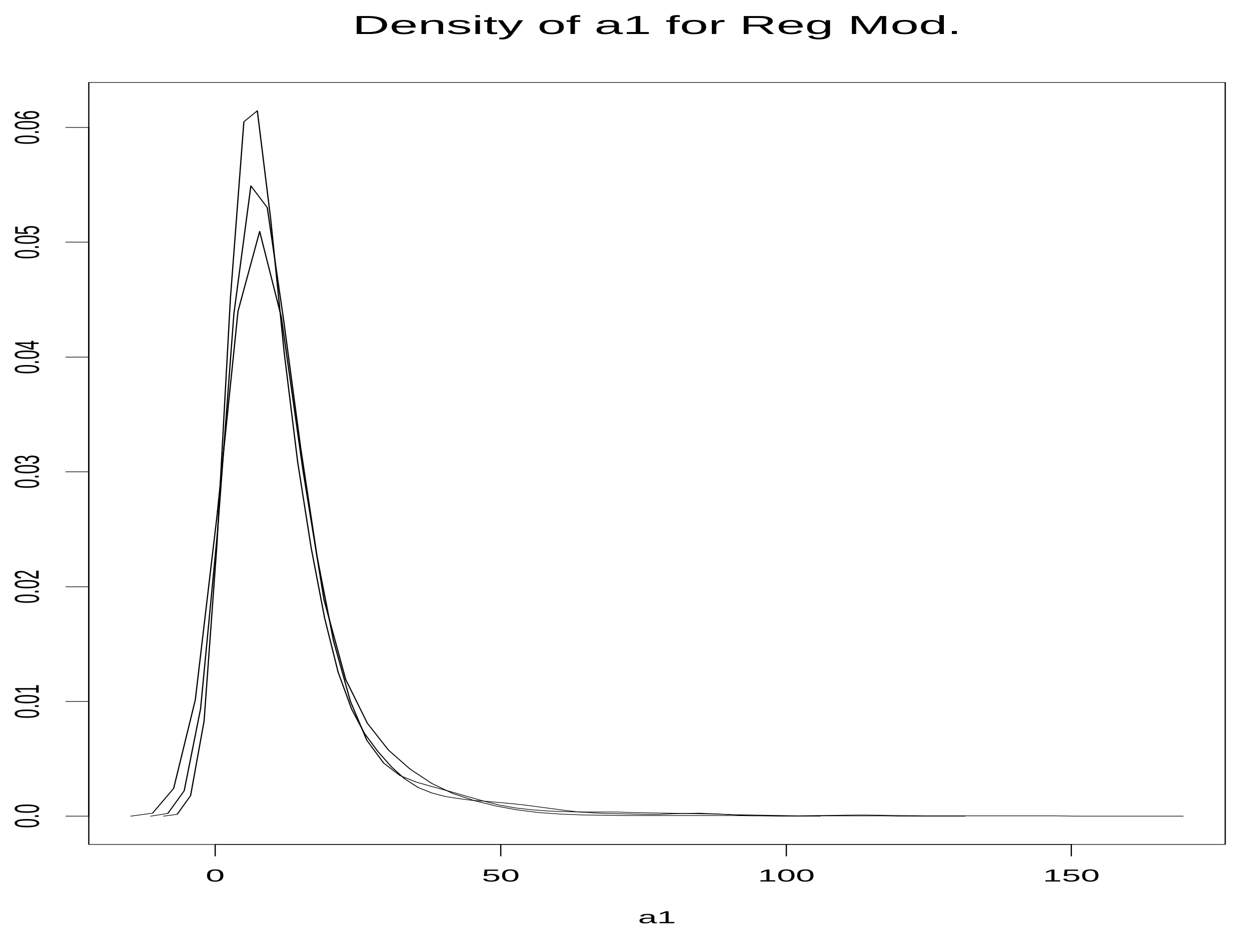

Figure 18: The estimate of the posterior density of a_1 from three independent chains for the "regular" model.

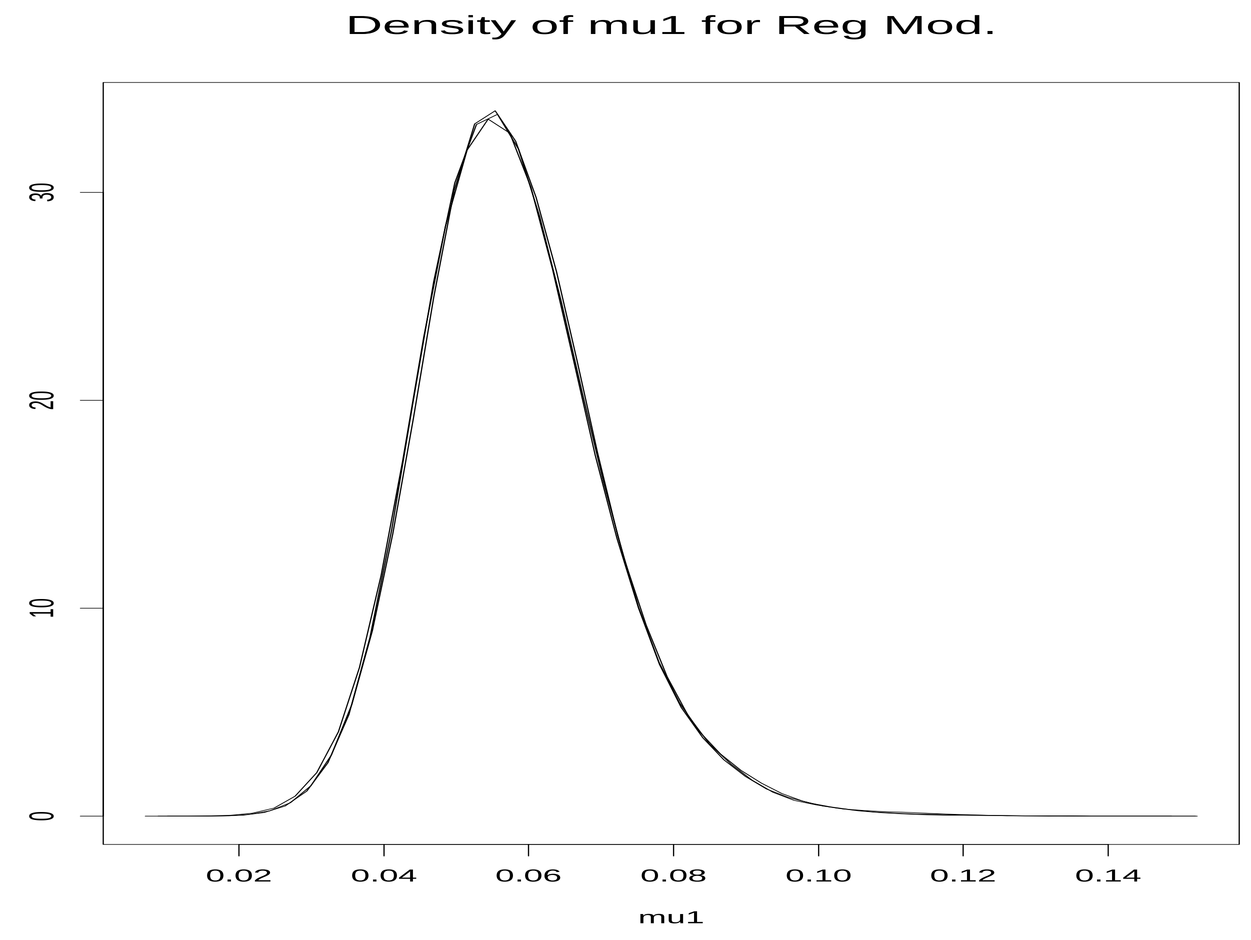

Figure 19: The estimate of the posterior density of μ_1 from three independent chains for the "regular" model.

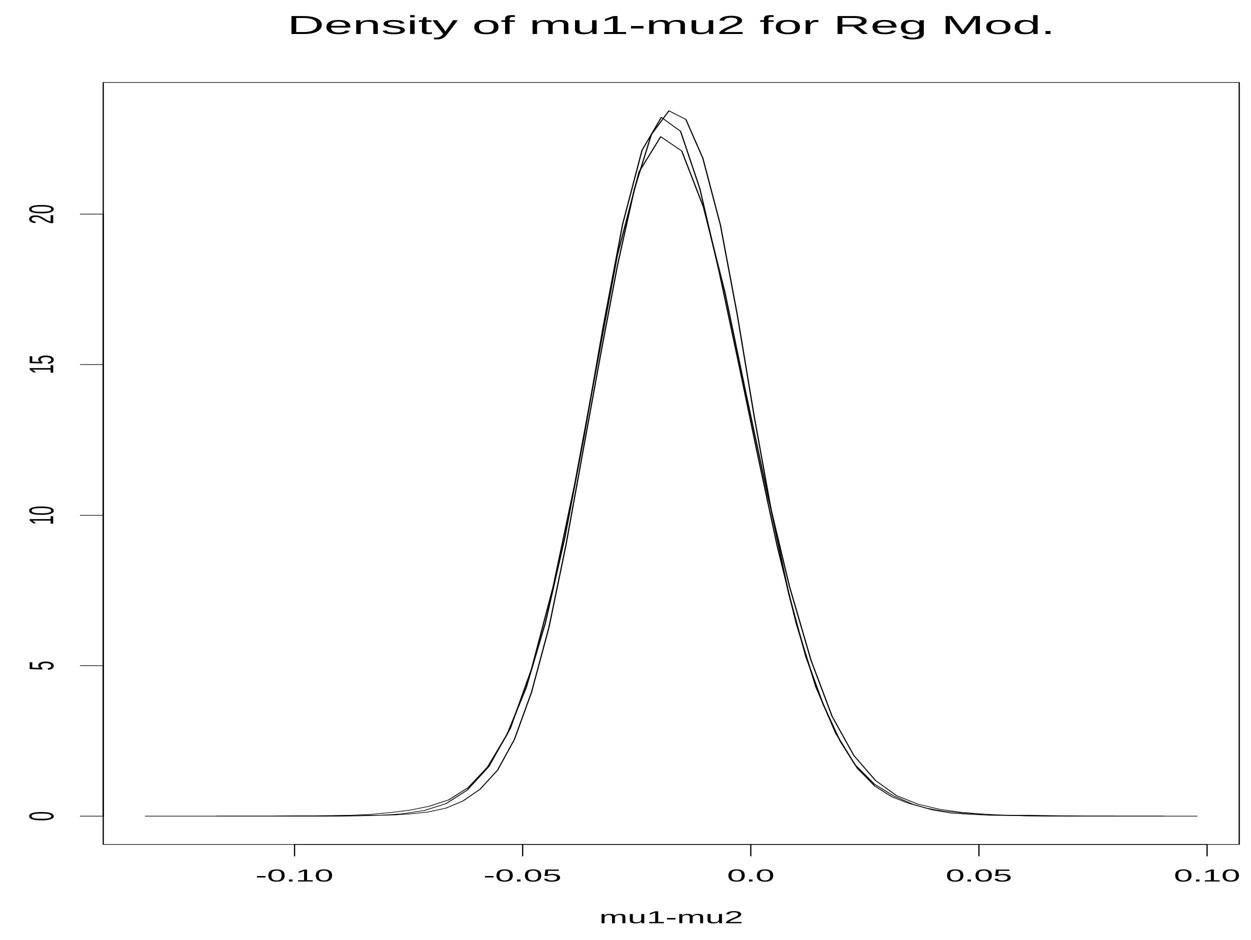

Figure 20: The estimate of the posterior density of $\mu_1 - \mu_2$ from three independent chains for the "regular" model.

- It would appear that for each of the posterior densities, the three independent chains have provide fairly close estimates. This does give us some idea how good our estimate is.
- The only estimate which seems a little off maybe is the estimate of the posterior density of a_1. The peak and the tail areas are in some disagreement. However, the samples for this parameter value were very skewed. The next slide shows the estimate of the posterior distribution for $\log(a_1)$. There still seems to be a fair amount of variation in the estimate of this density.
- On the other hand, the main question of interest was the difference $\mu_1 - \mu_2$. Here the estimates seem fairly close. Also, each chain suggest that there is some evidence that μ_1 has lower mortality, but the evidence does not seem very strong.

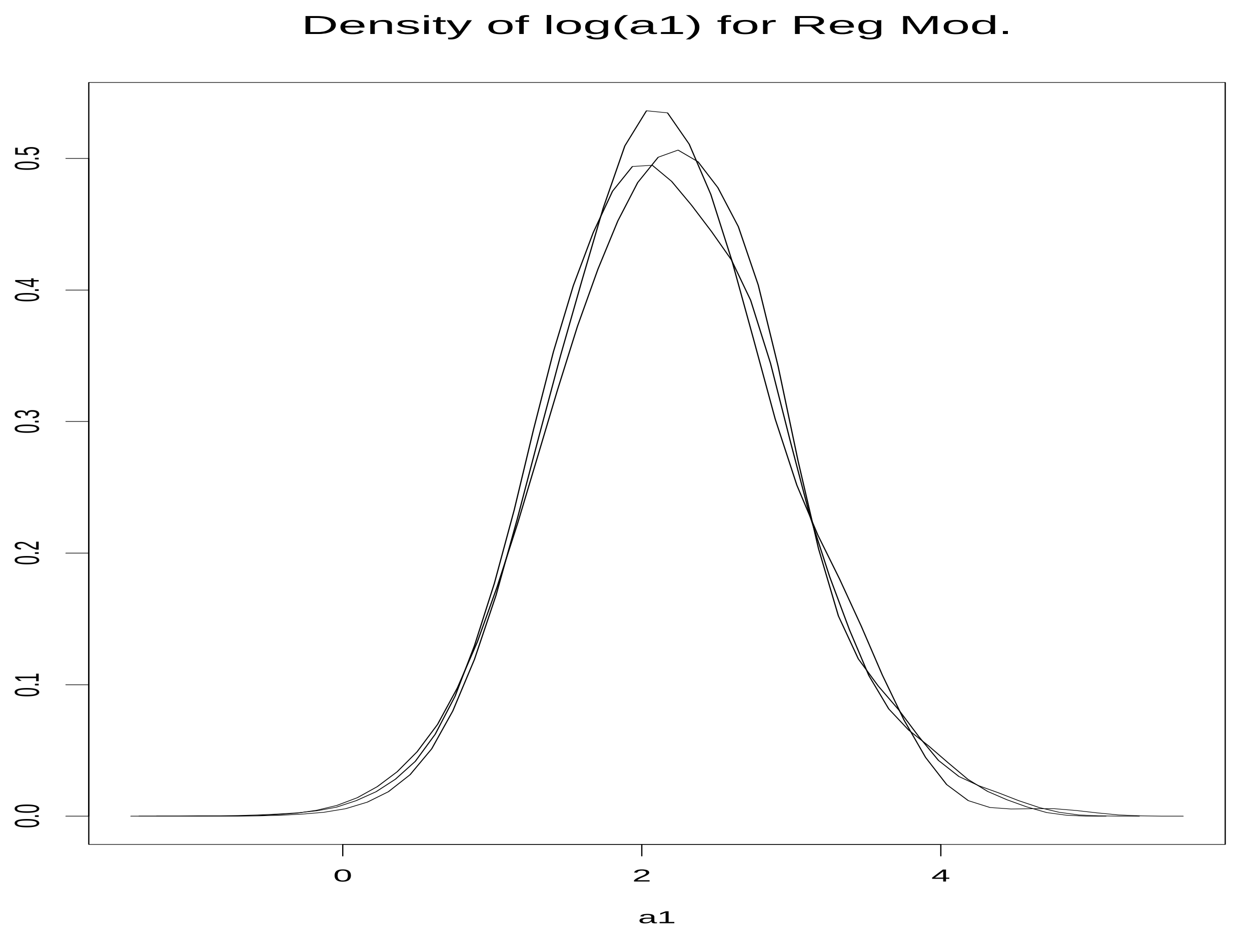

Figure 21: The estimate of the posterior density of $\log(a_1)$ from three independent chains for the “regular” model.

Using Simple Methods with the Alt. Model

- Now let us compare the alternative model with the regular model. The regular model used the parameters a_j and b_j in the MCMC sampler while the alternative model (the Alt Mod) used a different parametrization.
- First, let us use the simple graphic tools that were just used to look at the regular model.

Simple Diagnostics for Regular Model

When looking at the path plots and the trace plot, it was discovered that it would better to look at the log transform of the parameters a_1 and b_1. So, let us look at the path plot for the alternative model. These plots are on the next few slides.

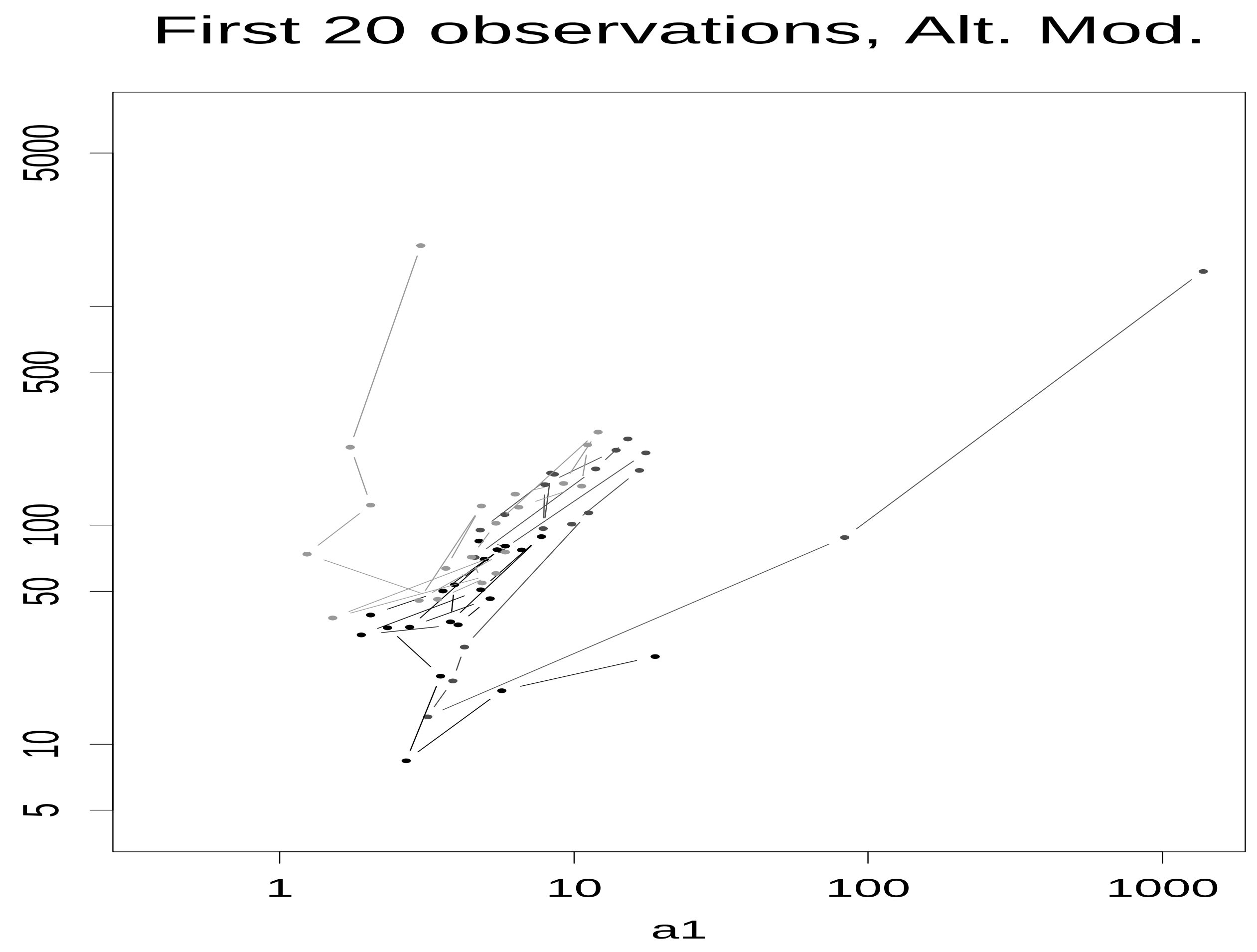

Figure 22: The path plot for the alternative parametrized model. Note both axis are on a log scale.

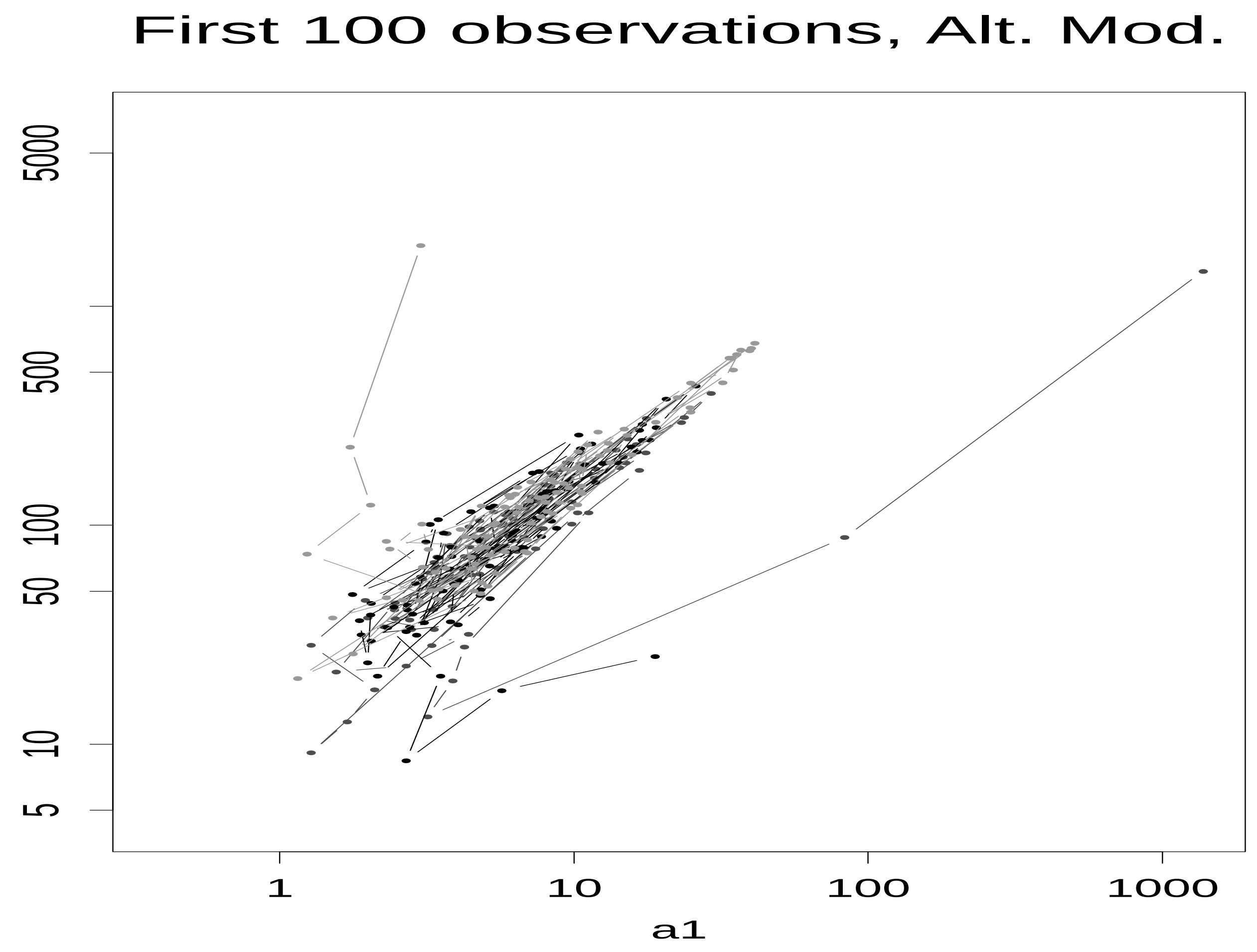

Figure 23: The path plot for the alternative parametrized model. Note both axis are on a log scale.

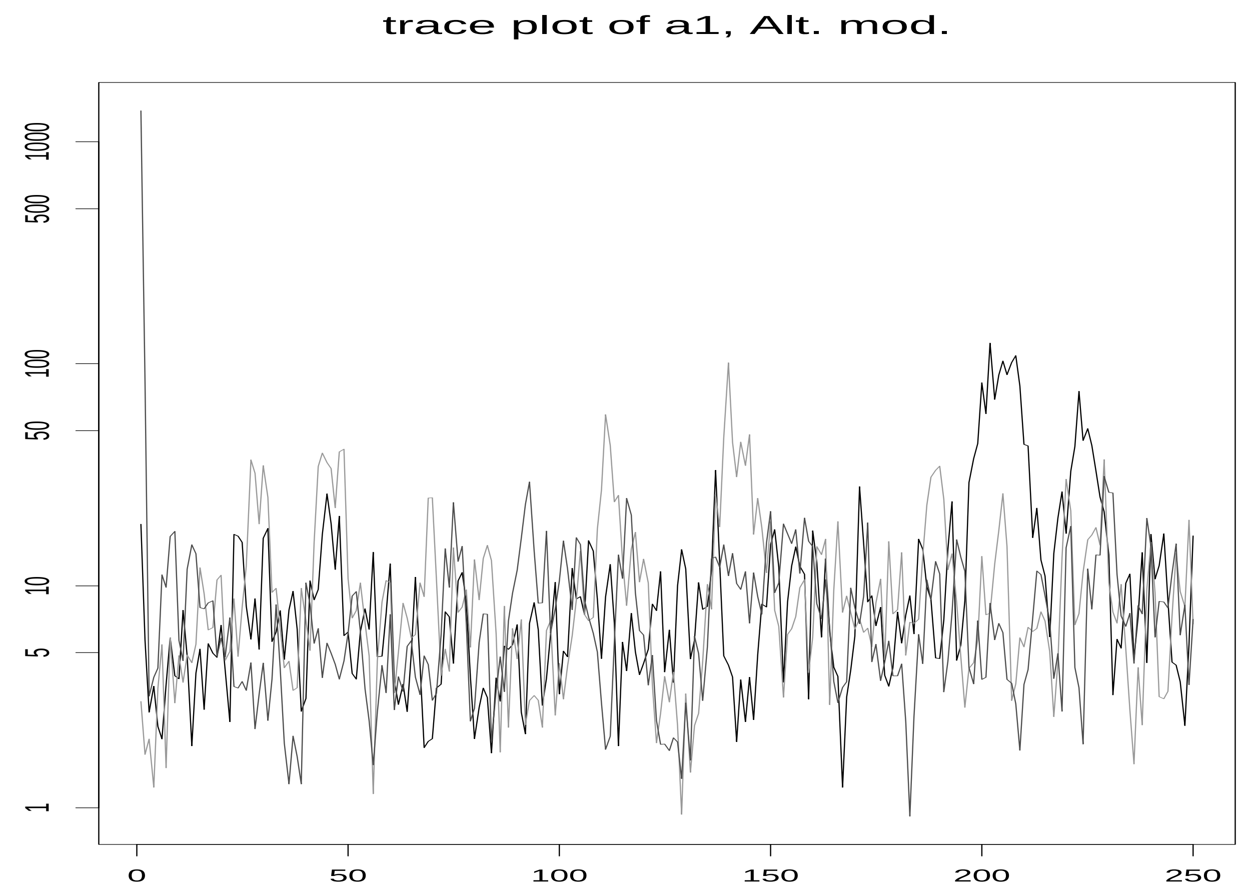

Figure 24: The path plot for the alternative parametrized model. Note both axis are on a log scale.

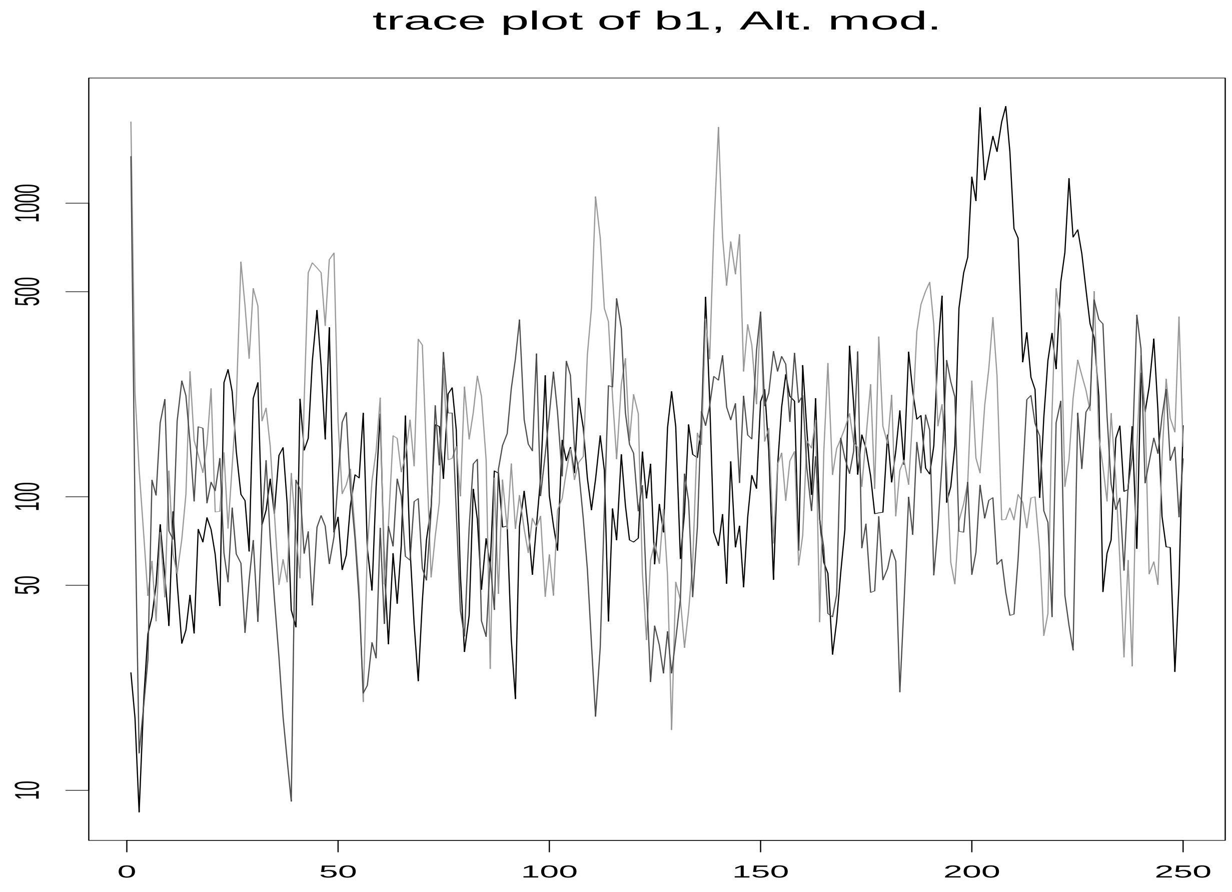

Figure 25: The path plot for the alternative parametrized model. Note both axis are on a log scale.

Path Plots and Trace Plots for Alt. Mod.

- Note that from the path plot, it appears that within 20 iterations, all three chains have merged together.
- The same can be seen from the trace plots. In the above figures, it appears that all three chains have found the main mass of probability.
- Remember, that this model has a different parametrization. Instead of basing the simulation on the parameters a_1 and b_1, this chain is based on the parameters μ_1 and θ_1. So, let us look at the estimated shape of the posterior distribution. Below is the contour plot of the sampled points. (Note, this plot models $\log(\theta_1)$.)

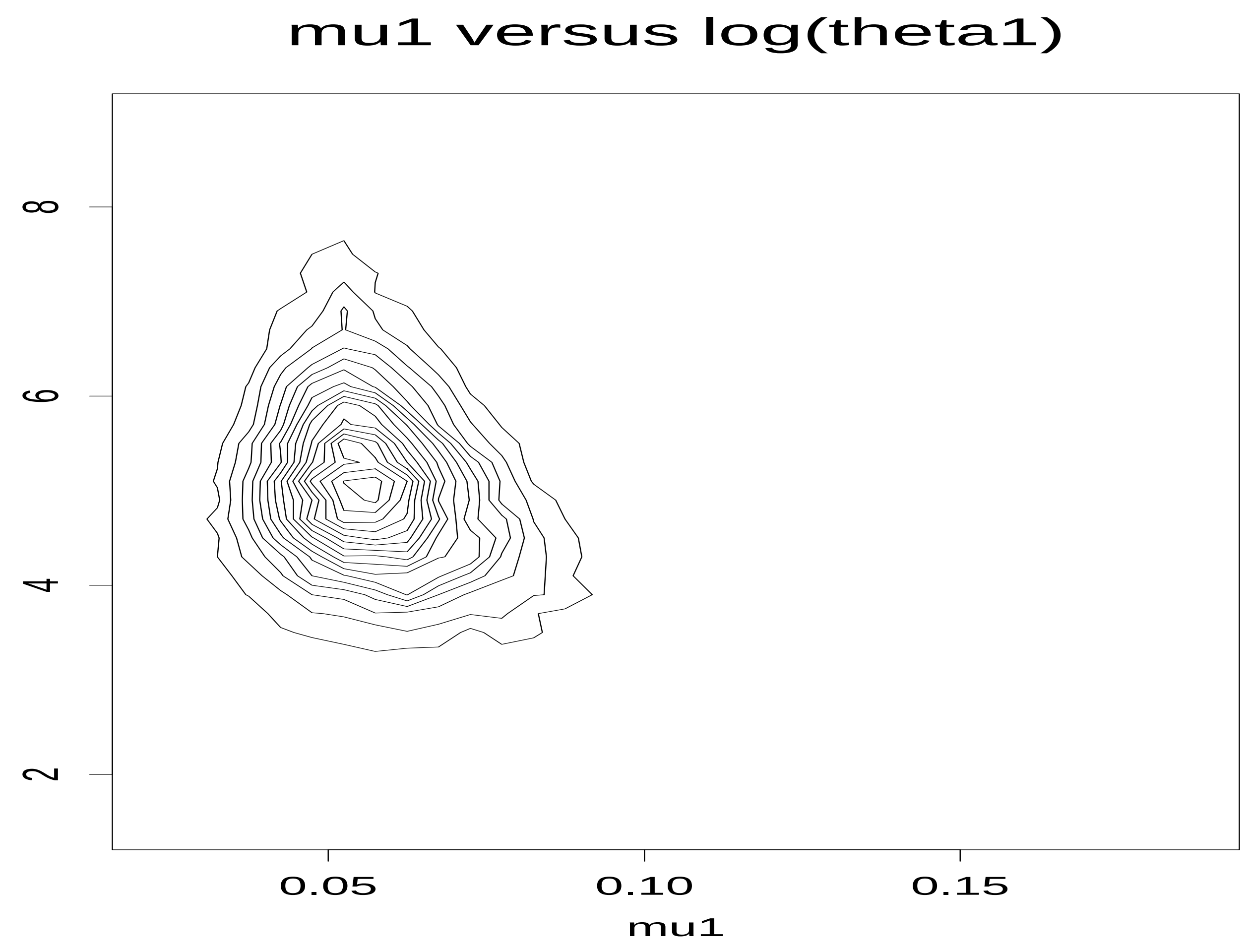

Figure 26: This is an estimate of the joint posterior for $\log(\theta_1)$ and μ_1.

- In comparing the joint distribution of a_1 and b_1 which drives the regular model to the joint distribution of μ_1 and θ_1 which drives the alternative model, we see that the the parameters in the regular model are highly correlated while the parameters in the alternative model are not correlated with each other. This allows the sampler in the alternative model to quickly move over the joint posterior distribution.
- An examination of the autocorrelation functions further supports this observations. This plots follow.

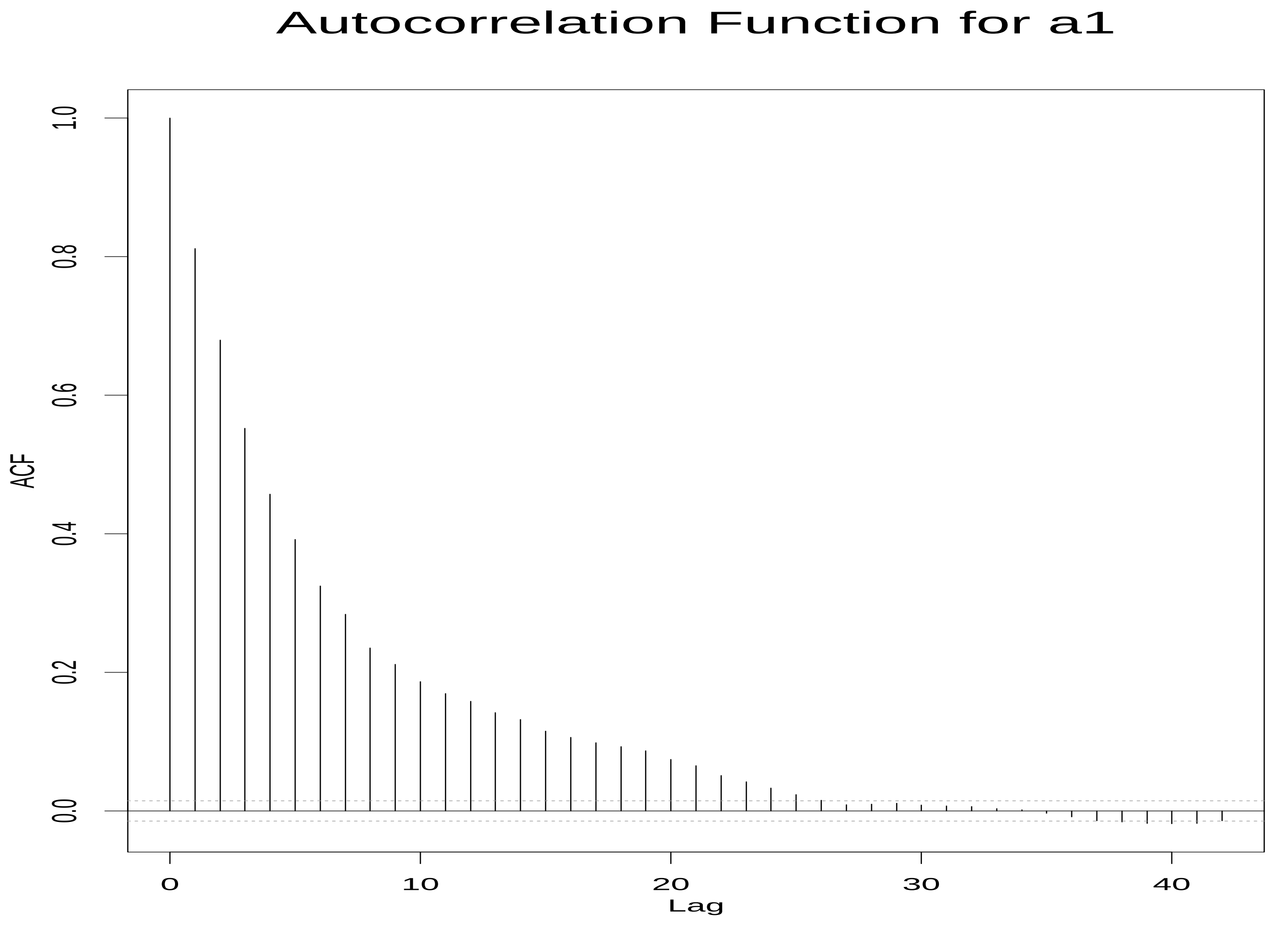

Figure 27: The autocorrelation function.

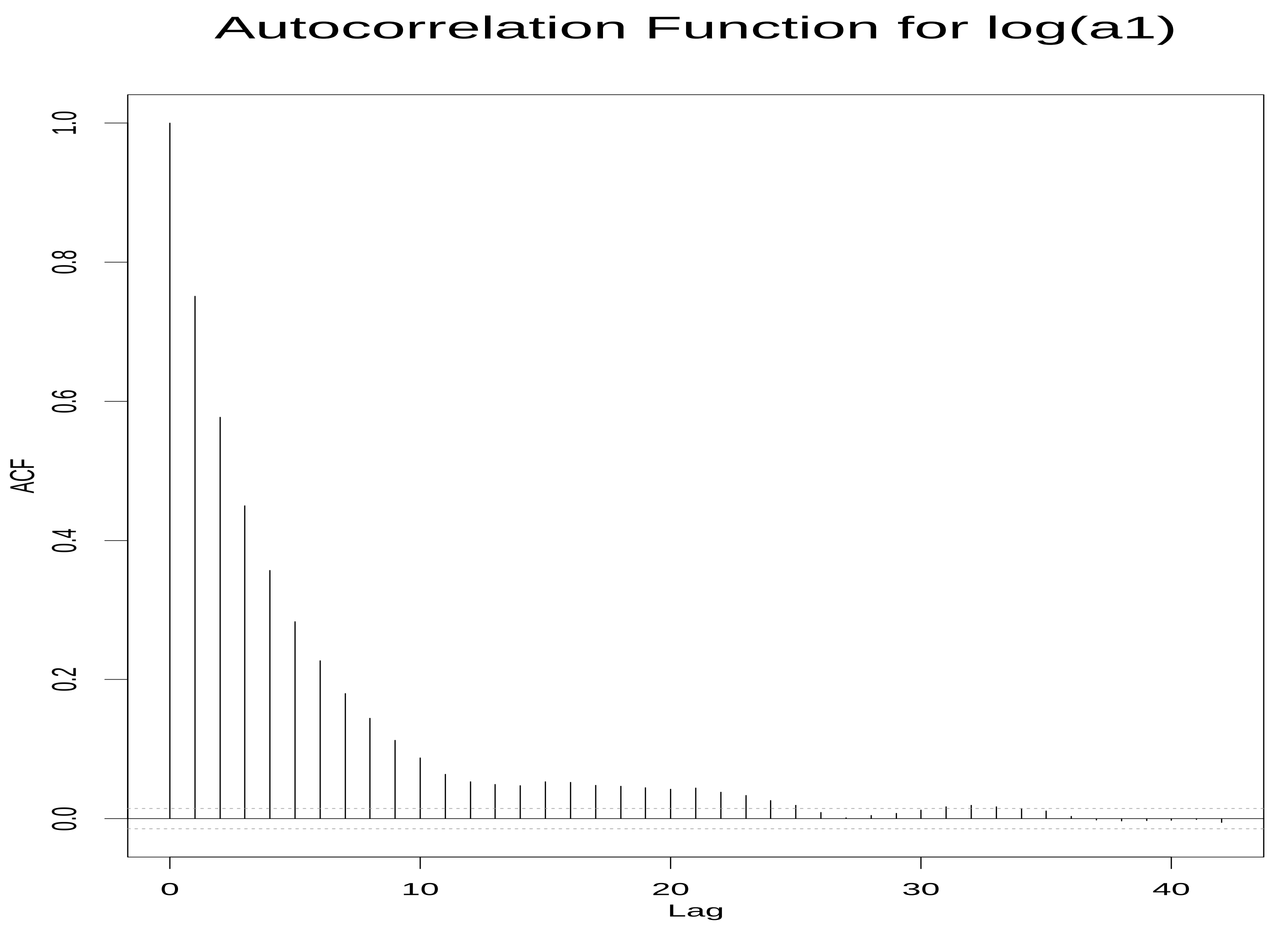

Figure 28: The autocorrelation function.

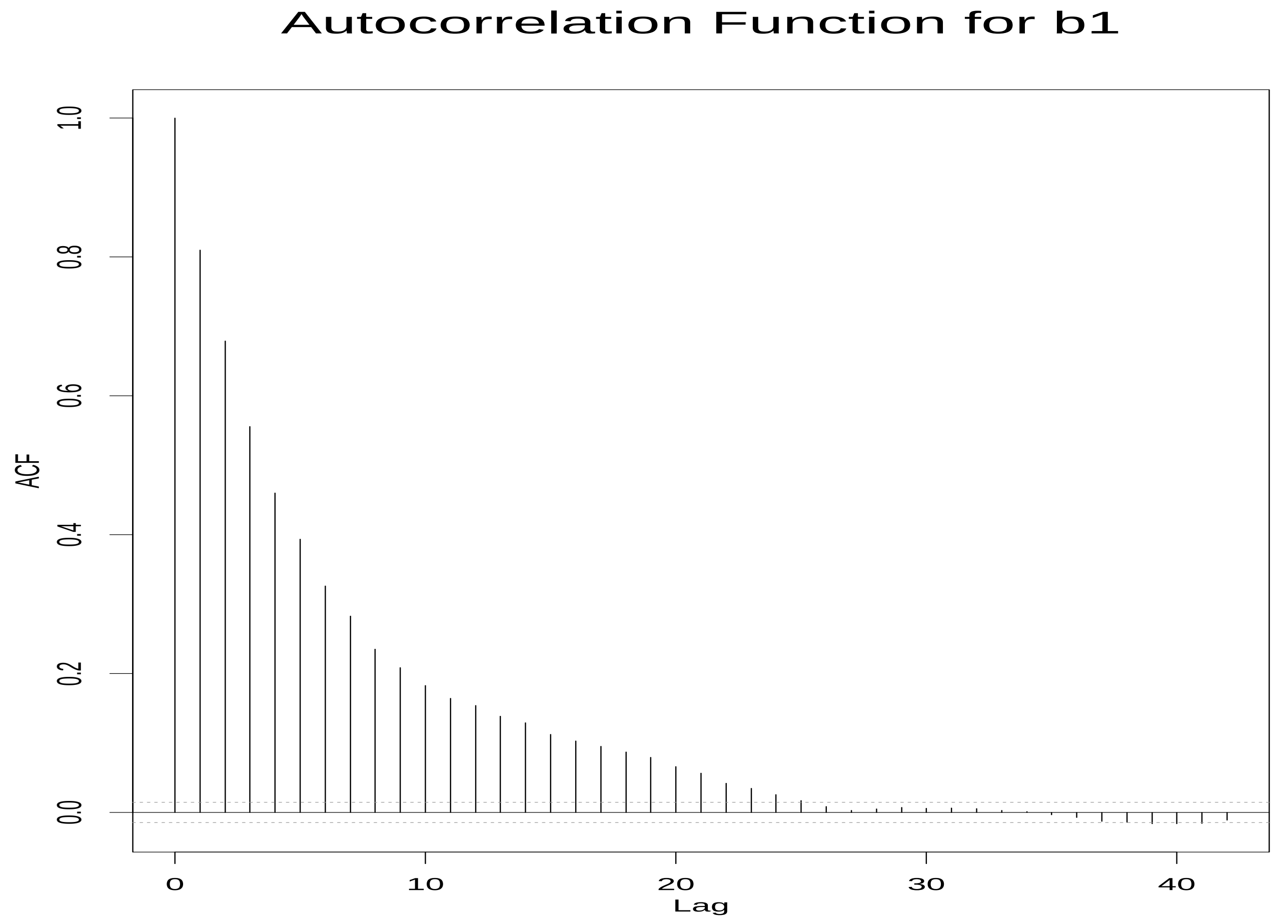

Figure 29: The autocorrelation function.

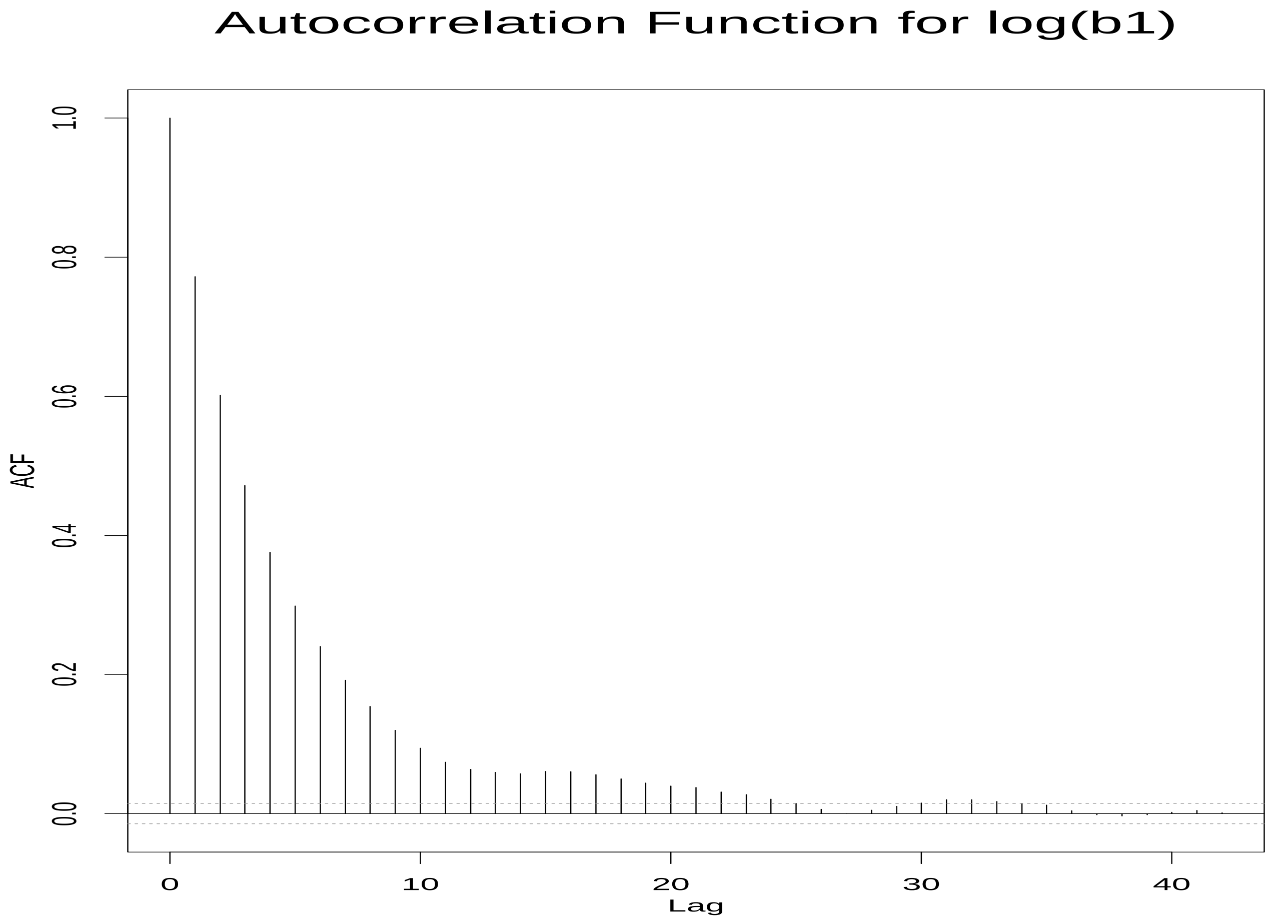

Figure 30: The autocorrelation function.

Estimates of Some Posterior Distributions

- Using the samples from the regular model, four posterior densities were estimated: the posteriors for a_1, μ_1, $\mu_1 - \mu_2$ and $\log(a_1)$.
- In the next sequence of slides, these densities are estimated with the samples from the alternative model.

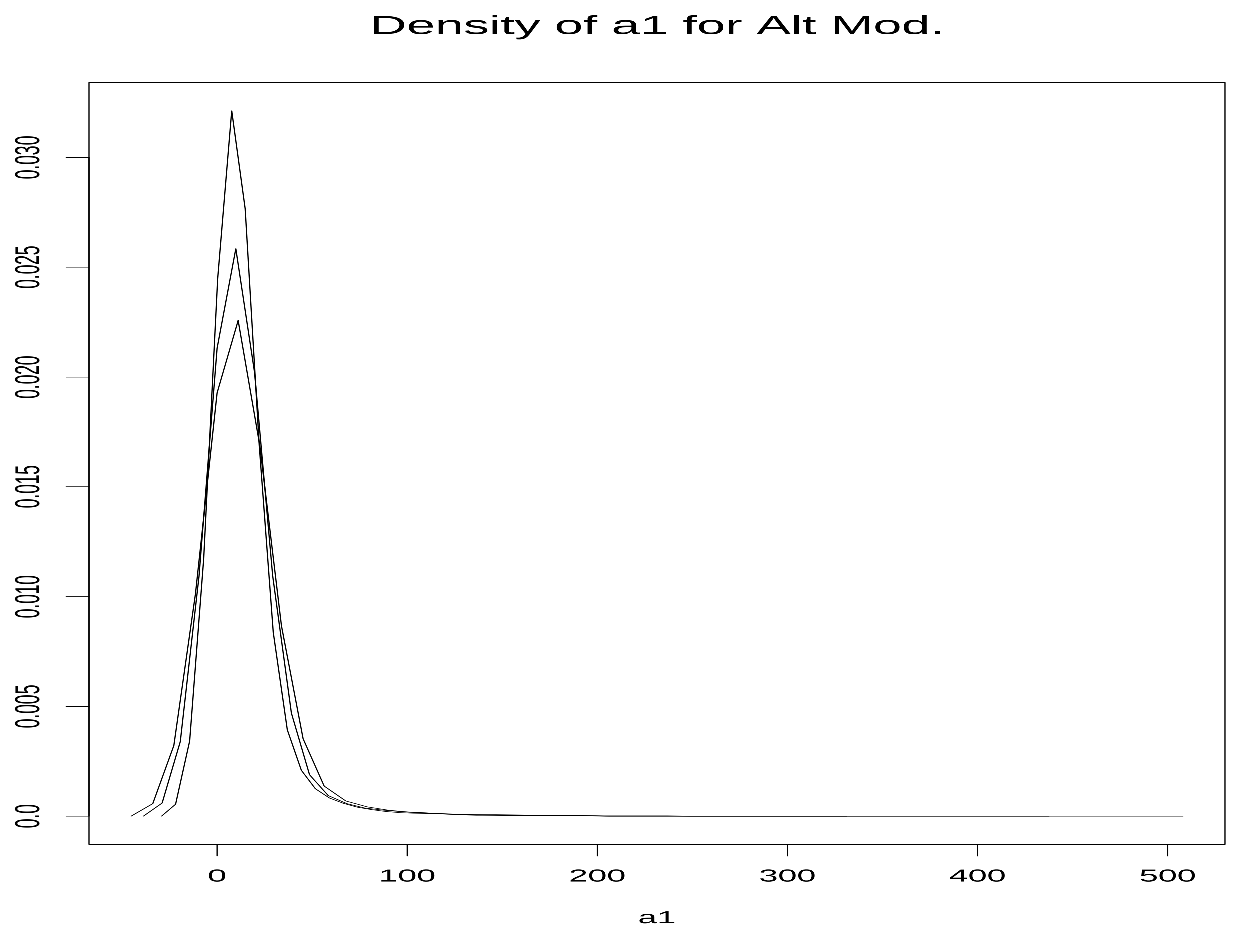

Figure 31: The estimate of the posterior density of a_1 from three independent chains from the alternative model.

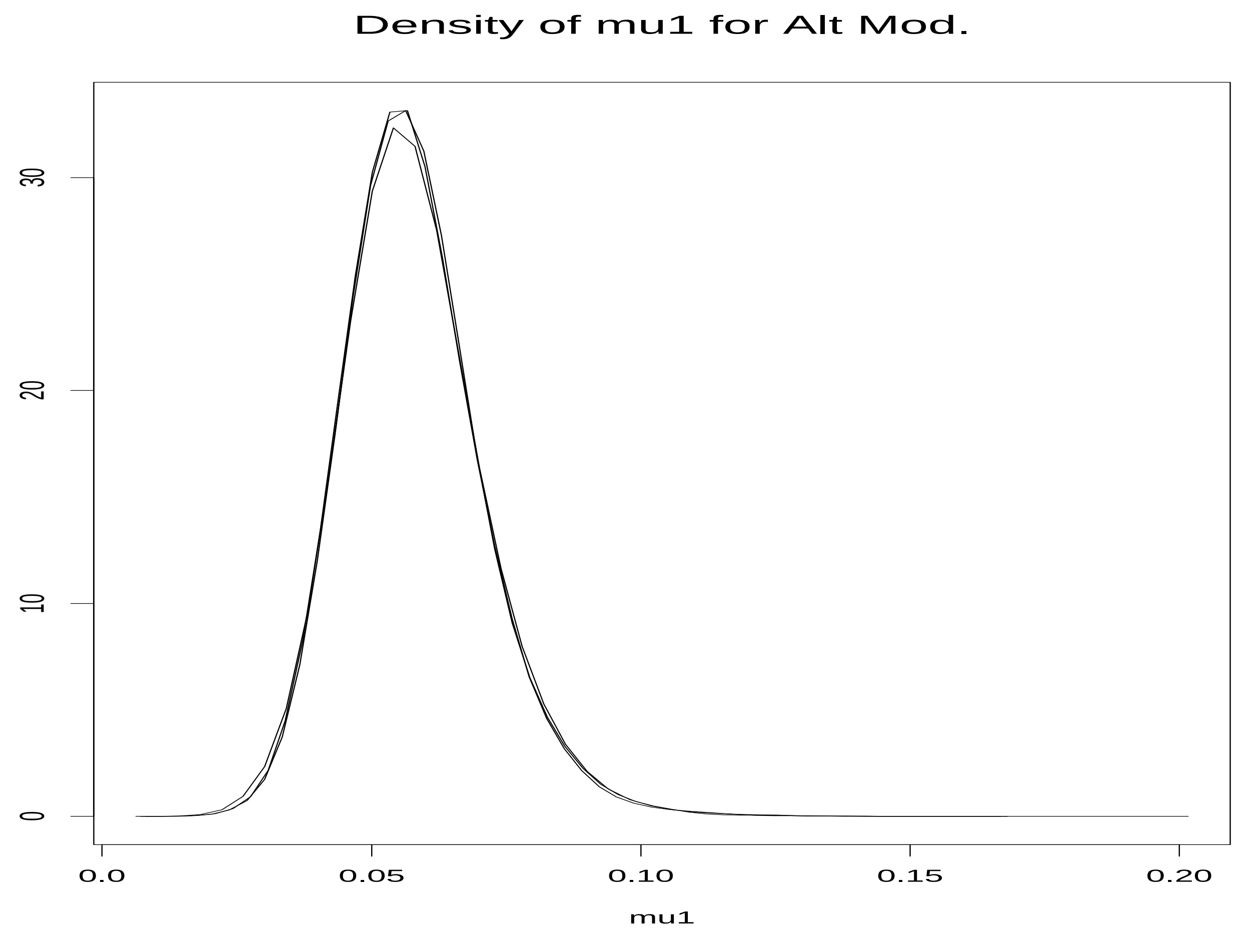

Figure 32: The estimate of the posterior density of μ_1 from three independent chains from the alternative model.

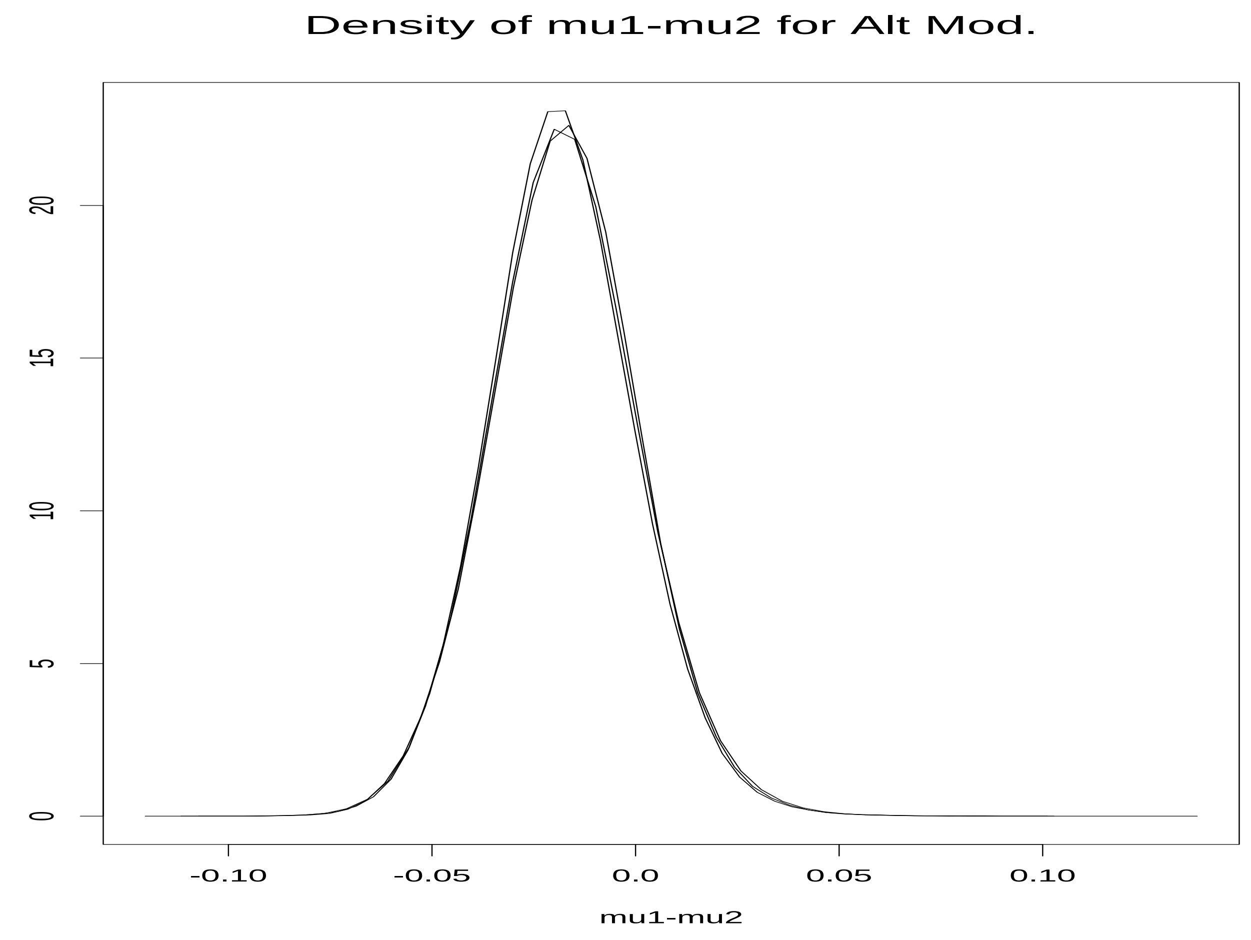

Figure 33: The estimate of the posterior density of $\mu_1 - \mu_2$ from three independent chains from the alternative model.

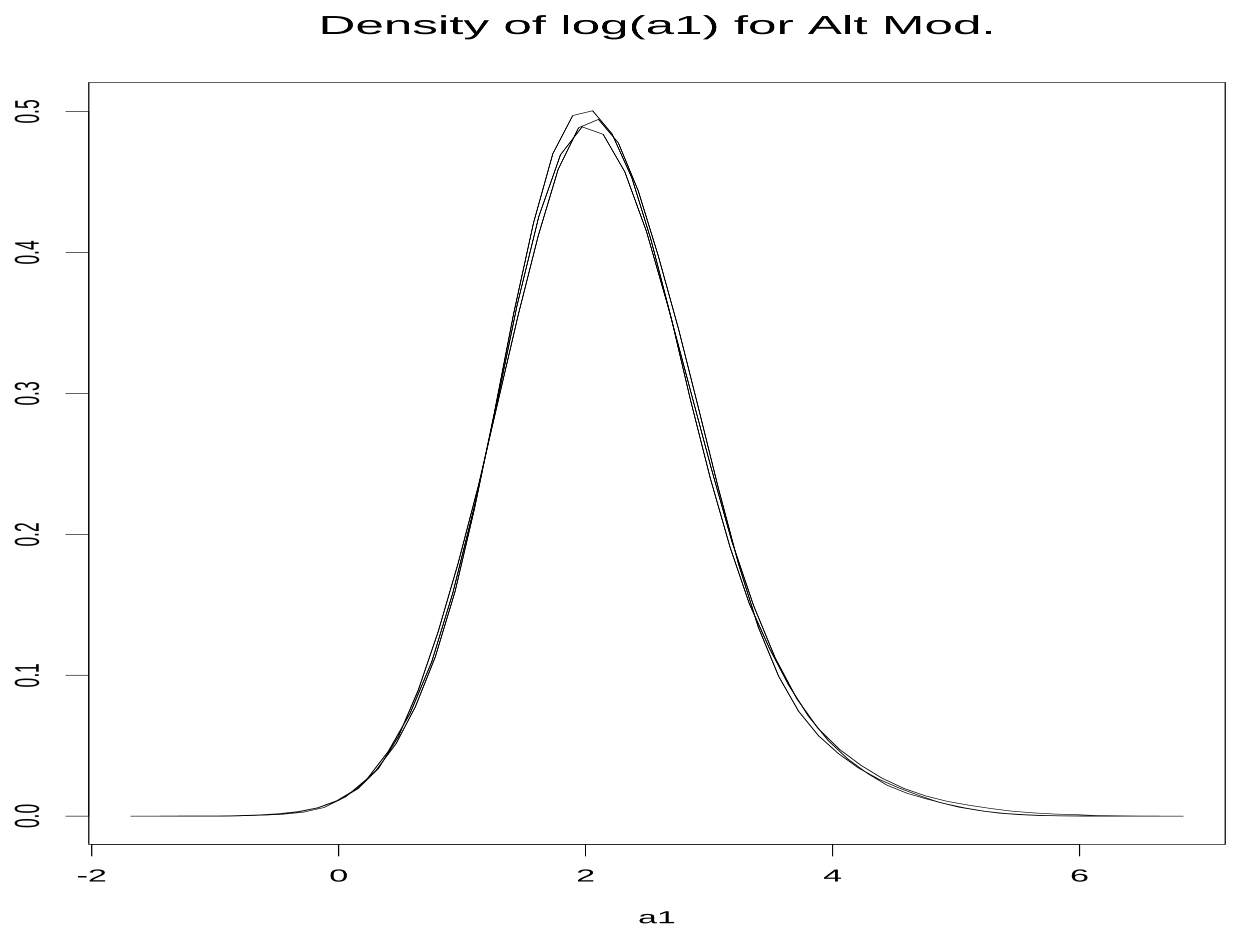

Figure 34: The estimate of the posterior density of $\log(a_1)$ from three independent chains from the alternative model.

Comparing Posterior Densities

- The three different estimate provided by the alternative model for each of the parameters is fairly good. Like the regular model, there seemed to be a fair bit of variation in the estimate of the distribution for a_1. However, the alternative model seemed to do a much better job of estimating the distribution for $\log(a_1)$.
- The alternative model seem to provide a much tighter estimate of the distribution of $\mu_1 - \mu_2$. However, the inference does not change. There is some evidence that the first treatment is better, but it is not very strong.

Model Comparison with Simple Diagnostics

- There exist a step ridge in the posterior when the models are parametrized by (a_j, b_j) as oppose to the (μ_1, θ_1) parametrization.
- Because of this step ridge, the markov chain takes longer to find the main probability mass, the samples have a much higher autocorrelation, and the estimated values are not as accurate for the same number of samples.

Complex Methods

- The complex methods use mathematical statistical theory to develop a test to see if the MCMC is doing what it is suppose to be doing.
- Two good references on these methods are Brooks and Roberts (1998, Statistical Computing, 319-335) and Cowles and Carlin (1996, JASA, 883-904).
- Below, two of these methods are discussed and applied to the beta binomial model. The methods that are used below are the Geweke method and Gelman and Rubin method.

Geweke Method

- This method checks to see if the chain is producing a consistent estimate over the course of the Markov chain.
- Suppose one wants to find the path average of a sequence X_i. If the chain is too short, then the path average may be inaccurate because it takes a while for the chain to cycle over the area of high probability. A chain that is too short might only give the average over some local neighborhood that the X_i happens to be at the time. If one takes a longer chain, the X_i would be visiting all the neighborhoods would be the average of the limiting distribution.

Geweke Method

(Continued)

- So, to check this, Geweke looks at the average over some block of sequences in the beginning of the chain and then over an average of blocks at the end. This is assuming that the chain is already well burned in and those starting values have been discarded.

Geweke Method

(Continued)

- So, now the problem is one of comparing two averages. This would normally be a simple t-test, but the problem is that the sequence of X_i's is correlated. So, one can not get the variance of the estimate of the mean in the usual way. So, Geweke uses basic time series methodology to get the correct variance. (Which is called the spectral density.) Now, with the correct variance estimate, one can form a Z-score.

Geweke Method

(Continued)

- So, if we take the first n_A values in the series and the n_B last observations, we can define the Geweke statistics as:

$$G = \frac{\bar{X}_A - \bar{X}_B}{\sqrt{\frac{s_A^2(0)}{n_A} + \frac{s_B^2(0)}{n_B}}}$$

  where $\bar{X}_A$, and $\bar{X}_B$ are the averages over the first n_A value and the last n_B values and $s_A^2(0)$ and $s_B^2(0)$ are the variance estimated via the spectral densities over those same values.

- So, if these Z-scores are large, (2 or more) then one worries about the estimate being produced.

Geweke applied to Beta Binomial

- Based on the results of the simple procedure, the estimates for the differences in the μ_j's is examined as well as the $\log(a_1)$ and two means: μ_1 and μ_2. The three chains which sampled these values were inputted into the coda functions for the Geweke test.

Geweke applied to Beta Binomial

```
> geweke.diag(mcmc(altmod1))

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5

  logA1  MuDiff     mu1     mu2
-1.0527  0.7260  0.5400 -0.5694
```

Geweke applied to Beta Binomial

- For the alternative model, the Geweke's Z-scores for $\mu_1 - \mu_2$ were 0.72, 0.35, and -1.42. For the $\log(a_1)$ values, the Geweke's Z-scores were -1.05, -0.42, and -0.22. So this does not raise any warning flags.
- For the regular model, the Geweke's Z-scores for $\mu_1 - mu_2$ were -1.87, 0.87, and 3.01, and for the $\log(a_1)$ values, the Geweke's Z-scores were -3.13, 0.26, and 3.44. Many of these values are fairly high for Z-scores. From the earlier analysis, we already know that there is ridge in the surface. In light of these test, we should probably run the chain much longer. *(Aside: these values were run via the BOA function...)*

Gelman-Rubin Methods

- One of the purposes of this test is to see if the chain has found the main posterior mass or are parts of this chain still drifting about. Also, this test can be sensitive to multimodes. It is similar to analysis of variance.
- To run this test, one runs several parallel chains. Ideally, these chains should be started out far from the know center of probability mass. (It is like throwing a big net over the limiting distribution.)
- Like in the Geweke test, one test to see if some scalar estimate is good. One then calculates B, the between chain sum of squares around the estimated value and also W the within chain sum of squares.

Gelman-Rubin Methods

(Continued)

- Then, one estimates the variance of the estimate of interest by a function of W and B. If the chains are still wandering around and have not quite fallen into the sphere of influence of the limiting distribution, then the estimate base on B will be too large. This is estimate of the variance is then divided by an estimate of the variance only based on W. Because of the lack of independence in the sampled value within a chain, the denominator estimate of variance will be too small. This test ratio is called the “Potential Scale Reduction Factor” or the PSRF. If the chains are under the influence of the limiting distribution, then the PSRF will be about 1. The suggested critical value is about 1.2.

Gelman-Rubin Methods

(Details)

The basic idea is from an ANOVA prospective

Consider sampling item X_{ij} for the j-th sample of the i-th chain.

$$\begin{array}{rl} \text{chain 1:} & X_{11}, X_{12}, \ldots, X_{1n} \\ \text{chain 2:} & X_{21}, X_{22}, \ldots, X_{2n} \\ \vdots & \vdots \\ \text{chain m:} & X_{m1}, X_{m2}, \ldots, X_{mn} \end{array}$$

Naively, the idea is that the m chains should not about the same estimated value. If the chains produce difference estimates, then there is still uncertainty in the estimated value.

Gelman-Rubin Methods

(Details – continue)

- However, there will be differences between the estimates.
- Part of this will be due to the variation within a chain. This is the within chain error. One can note that this is the $\text{Var}(X|\text{chain})$.
- Also, there is a variation between the different estimates. So this is the between chain variation. So, in an ANOVA, if there are no difference between the
- If there is no variation between the chains, then the overall variation will be about the same as the variation within chain.

So, the overall variation, can be represented as:

$$\text{Var}(X) = \text{Var}(\text{E}(X|\text{chain})) + \text{E}(\text{Var}(X|\text{chain}))$$

Gelman-Rubin Methods

(Details – continue)

- One can estimate Var(E(X|chain)) by:

$$\frac{B}{n} = \frac{1}{m-1}\sum_{i=1}^{m}(\bar{X}_i - \bar{\bar{X}}_{\cdot})^2$$

- One can estimate E(Var(X|chain)) by:

$$W = \frac{1}{m}\sum_{i=1}^{m}(s_i^2)$$

.

Where,

$$\bar{X}_i = \frac{1}{n}\sum_{j=1}^{n} X_{ij}, \quad \bar{\bar{X}}_{\cdot} = \frac{1}{m}\sum_{i=1}^{m}\bar{X}_i, \text{ and } s_i^2 = \frac{1}{n-1}\sum_{j=1}^{n}(X_{ij} - \bar{X}_i)^2.$$

Gelman-Rubin Methods

(Details – continue)

To estimate the overall variation, Gelman and Rubin (199x) use the following estimate:

$$V = \frac{n-1}{n} W + \left(1 + \frac{1}{m}\right) \frac{B}{n}$$

Note:

- I'm not going to explain the fine details of the different terms with n and m in the above equation. When n and m are large, they don't matter.
- The correct of the m term is similar to what is done in multiple imputation as a small sample size correction.

Gelman-Rubin Methods

(Details – continue)

- If all the chains are now sampling from the same distribution (so things have now "converged" in a certain sense), then the overall variation will be the same as the variation within a chain.

Gelman-Rubin Methods

(Details – continue)

- So, Gelman and Rubin (later modified by Brooks and Gelman) then use a statistics which is based on a ratio of V and W. Their statistics, which is called R is defined in difference sources as either:

$$\begin{aligned} R &= \sqrt{\left(\frac{V}{W}\right)}, \text{ or} \\ &= \left(\frac{V}{W}\right), \text{ or} \\ &= \left(\frac{(d+3)V}{(d+1)W}\right) \end{aligned}$$

If there is convergence, then R should be approximately 1. If the value is over 1.2, then one should worry.

Gelman-Rubin Methods

(Beta-Binomial model.)

The chais for the regular model where analysed via the boa program with the following output:

```
Iterations used = 9001:18000


Potential Scale Reduction Factors
---------------------------------


 "MuDiffr" "logA1r"
  1.001294 1.000756

Multivariate Potential Scale Reduction Factor =
1.0012929
```

```
Corrected Scale Reduction Factors
---------------------------------

          Estimate    0.975
"MuDiffr" 1.003126 1.006786
 "logA1r" 1.001501 1.003689
```

Gelman-Rubin Methods

(Beta-Binomial model.)

- The PSRF for the regular model was 1.00 for the differences in μ_j's and was 1.00 for the $\log(a_1)$. For the alternative model, PSRF numbers were 1.001 and 1.0002 respectively. So, this test does not trip any warnings about our estimates.

Standard error, burn-in, and thinning

- Different ways of getting standard error of estimates: a) batch means, b) spectral density, c) from the autocorrelation function.
- Burn-in refers to the practice of "throwing away" the beginning of the chain. This is done because it takes awhile for the chain to settle down and for it to be sampling from the approximate stationary distribution.
- Thinning is the practice of keeping every m-th value. This is done to reduce the autocorrelation of the values that are kept. Therefore, the "effective sample size" is near the number of samples that are kept. Might be an issue if one is storing the chain, etc.

Standard error

(Continue)

- Batch means: One breaks the chain into batches so the means of the batches is approximately uncorrelated.
- There is a method which can be developed by summing the autocorrelations.
- Spectral density: This is based on methods developed in econometrics. One fits a time series model to the data. These methods allow one to estimate the standard error of the estimate of the mean.

Note: the spectral density estimate will not be discussed here. The other two methods are easy to estimate.

Standard error: via Batch Means

- Batch means: Break the chains in batches. A batch is a consecutive run of observations. Then look at the variance of the batch means.
  - The chain is divided up into K batches. The number of batches, K, is usually 30 or 50 or so.
  - The can actually be overlapping. (Then they are called overlapping batch means.)
  - The means of the batch will have a lower autocorrelation. Actually, one hopes that the autocorrelation is very small.
  - This method was mostly developed in the area of "operation research" or "management science".

Standard error: via Batch Means

- First, one calculates the mean of each batch. Lets call the K means, BM_k. Then, one calculates the "standard error" of the K batch means.
- Note that the mean of the BM_k's is the overall all mean. Also, if the BM_k's are (approximately) uncorrelated, then the standard error of the BM_k's is the standard error of the estimate of the overall mean.

Standard error: via Batch Means

The following code calculates the batch means estimate of the standard error of the mean when one inputs a vector of MCMC values:

```
CalcBatchMeans=function(x,Batn=50){
   BigN=length(x)
   BatInc=ceiling( (1:BigN)/(BigN/Batn) )
   BM=tapply(x,BatInc,mean)
   list(MCE=(sd(BM)/sqrt(length(BM))), BM=BM)}
```

Standard error: via Batch Means

Here is some results for the alternative model:

```
> BatchSE=apply(altmod1,2,function(x)(CalcBatchMeans(x))$MCE)
> BatchSE
  logA1    MuDiff   mu1      mu2
0.01826 0.00054 0.00017 0.00048
```

Note: in the above, `altmod1` is a matrix with the MCMC chains stored as column vectors.

Standard error: via Batch Mean

Note: the above can also be run via a coda command:

```
>batchSE(mcmc(altmod1))
logA1    MuDiff      mu1       mu2
0.01799  0.00045 0.00021  0.00040
```

Standard error: via Autocorrelation Function

- Remember the basic methods for estimating variance of correlated variables. Consider the variables X_1 and X_2 If we assume that they all have the same variance, σ^2 and the correlation between X_1 and X_2 is ρ, then $\text{Var}(X_1 + X_2) = \sigma^2(2 + 2\rho)$.

Note: The right hand side is the sum of all the elements in the covariance matrix.

Standard error: via Autocorrelation Function

- Using a similar arguement, consider a large number of X_i with $i = 1, \ldots, n$ with common variance σ^2 Also, let the correlations between X_i and X_{i+j} be $\rho(j)$. Assume that for some w, all for $j > w$ then $\rho(j)$ is about zero. Also, assume that w is much smaller than n. Therefore,

$$\mathrm{Var}\left(\sum_{i=1}^{n} X_i\right) \approx n\sigma^2 \left(1 + 2\sum_{j=1}^{w} \rho(j)\right)$$

Note: One can see the above if one imagines a banded correlation matrix. The right hand side is the sum of all the elements in the variance matrix. The approximation is due to ignoring some "edge effects".

Standard error: via Autocorrelation Function

- Aside: For the above, consider the i-th row of the correlation matrix. It looks like:

$$(0, \ldots, 0, \rho(w), \ldots, \rho(1), 1, \rho(1), \ldots, \rho(w), 0, \ldots, 0)$$

Again, ignoring edge effects. That is, where $i < w$ or $i > n - w$.

Standard error: via Autocorrelation Function

- To get the estimate of the MC standard error, one uses the usual variance estimate to estimate σ^2 and the sample autocorrelation function to estimate $\rho()$.
- Then, give $X_1, X_2, \ldots, X_n$ as sampled values from an MCMC sampler, one can estimate the standard error of the mean by:

$$(MCE) \approx \frac{S}{\sqrt{n}} \sqrt{1 + 2 \sum_{j=1}^{w} r(j)},$$

where $S = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})^2}$, r is the sample autocorrelation function, and w is such that the autocorrelations greater than w are small.

Standard error: via Autocorrelation Function

The following R function could calculate the MC standard error for a vector of sampled MCMC values:

```
CalcAcSe=function(x,lag.max=50){
   autoc=(acf(x,lag.max=lag.max,plot=FALSE))$acf
   sd(x)/sqrt(length(x))*sqrt(-1+2*sum(autoc))}
```

Standard error: via Autocorrelation Function

Aside: In the last slide in the last line, the term "`sqrt(-1 +2*sum(autoc))`" is correct. The `acf` function includes the "zero" lag which is one.

Standard error: via Autocorrelation Function

Here is some results for the alternative model:

```
>
> AutocSE=apply(altmod1,2,CalcAcSe)
> AutocSE
       logA1        MuDiff           mu1           mu2
0.0184916156 0.0004701510 0.0001934147 0.0004268169
>
```

Note: in the above, `altmod1` is a matrix with the MCMC chains stored as column vectors.

Standard error: via Spectrum Density

The R package `coda` will calculate MC standard error using the spectrum density when the `summary` function is used on a `mcmc` object or a `mcmc.list` object. For example:

```
> summary(mcmc(altmod1))
            Mean      SD  Naive SE Time-series SE
logA1    2.17705 0.83942 6.257e-03      0.0176875
MuDiff -0.01727 0.01717 1.280e-04      0.0005409
mu1      0.05788 0.01239 9.237e-05      0.0001902
mu2      0.07515 0.01165 8.686e-05      0.0004513
```

Parameter values for MC SE

- For the batch mean, it is an open question as to how many batches one should use.
- When using the autocorrelation function, one needs to choose how many lags to use.
- In the next figure, this issue is examined. In that plot, one can see that there is general agreement between the methods when the chain is long enough and for many typical values of the parameters.

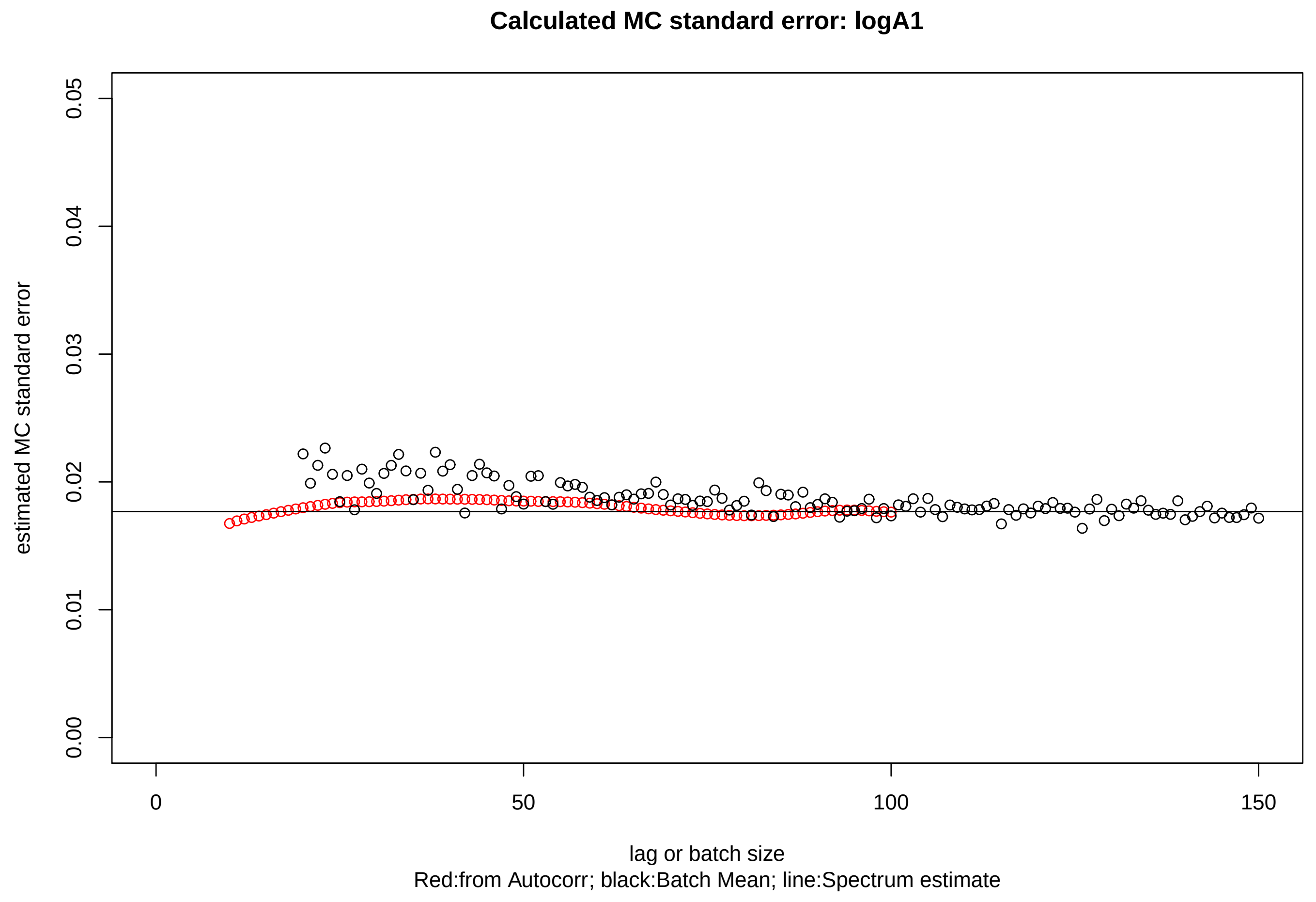

Figure 35: Different estimates of the MC standard error using different methods and different parameters for those methods.

Effective Sample Size

The effective sample size calculated from the MC standard error. It has the following value:

$$\frac{(\text{standard error if chain assumed independent})^2}{(\text{standard error accounting for correlation})^2}$$

$$\times(\text{number of actually samples})$$

Effective Sample Size

Here are the results of calculating the effective sample size using different estimates of the MC standard error:

```
> # if it they were independent samples:
> length(altmod1[,1])
[1] 18000
>
> #  Using Spectrum density estimate:
> effectiveSize(mcmc(altmod1))
    logA1    MuDiff        mu1        mu2
2263.1069 1382.3191 3999.4738  713.7738
>
```

Effective Sample Size

```
> # Using just the autocorrelation function:
> autoc=apply(altmod1,2,function(x){(acf(x,lag.max=50,plot=FALSE)$acf
> apply(autoc,2,function(x){length(altmod1[,1])/(-1+2*sum(x))})
   logA1   MuDiff      mu1      mu2
2060.678 1334.217 4105.697  745.462
>
> # Using Batch means estimate:
> apply(altmod1,2,var)/(BatchSE)^2
    logA1    MuDiff        mu1        mu2
2111.8549  997.9157 5603.7392  601.2635
>
```

Comment about the Beta-Binomial model

- Please note that the likelihood surface of the beta-binomial model also has a sever ridge in it. I discovered this when writing a Newton-Raphson algorithm to fit that model. The (a_j, b_j) parametrization is extremely difficult to fit.
- Users who are use to bullet proof, big corporate software packages might find BUGS a little more fickle then they are use to. However, part of the great flexibility with BUGS is somewhat the cause of the difficulty. Large packages will trap many problems for a user, but they can also limit the flexibility.

Closing Comments

- In 1990, there was a computational break through for Bayesian statisticians. Problems which seemed impossible were now all of a sudden very doable.
- BUGS represents a software break through for Bayesian statisticians. In order to implement MCMC methods, one still needed to calculate a series of conditional distributions and do a fair bit of tedious programming. However, now one simply needs to specify their model and BUGS takes it from their.
- For the first time, applied Bayesian statistician have the tools to practice the art of data analysis.

Summary of Today

Today we looked at:

- Ways of quantifying uncertainty using mathematics.
  - That was done by expressing uncertainty as a probability function.
- Looked at simple models with conjugate priors.
  - This allowed us to work out the mathematical properties in these simple models.

Summary of Today

(Continued)

- Looked at ways to use simulation techniques to get samples from the posterior distribution.
  - This is usually done by sampling from a Markov Chain.
  - Conditional independence allows one to model complex models.
  - Became familiar with the basic software to analyze these Bayesian models.
  - Looked at ways to learn about complex joint posterior using these methods.

Summary

(Continued)

- Looked at ways to perform basic checks to see if the sampling chain is working correctly.
- Note: By learning the above, one now has the basic building blocks to fit Bayesian models. There are many resources available which shows one how to code different statistical models. With the correct code for a model, one can fit and check the model in a similar manner as discussed here.