Applied Bayesian Methods

Michael Escobar

University of Toronto

m.escobar@utoronto.ca

Lecture 3

Outline:

- Review some R code for the simple normal example.
- Normal Models
- WinBugs: a package for Bayesian data analysis
- Example of Growth curve: RATS

Introduction

- In the previous weeks, we learned the basic material. The first lecture covered the basic Bayesian model. The second lecture discussed the basics of Markov Chain Monte Carlo methods which include a basic example involving a simple normal model.
- This week, we are going to demonstrate how to apply these methods to a number of different types of models. The goal is not to provide a complete overview of the different models that can be fit. Rather, the purpose is to show how to use these methods in some basic models. Once you have seen these methods in operation on these models, then you can apply these methods to your own models.

Introduction

(Continued)

- There are a number of different sources which demonstrate these methods to different problems. This includes the BUGS manual (reference: Spiegelhalter, Thomas, Best, Lunn, *WinBUGS User Manual*, http://www.mrc-bsu.cam.ac.uk/bugs) and also a recent book by Congdon (2001, *Bayesian Statistical Modelling*, Wiley.).
- While demonstrating these MCMC applications, diagnostic procedures are shown.

Review R code: see hand out.

See the code in either the file `SimpMCMC.r` or `SimpMCMCJpeg.r`. They contain the code for running the simple Heights example.

Basic MCMC code

The R codes in the files `SimpMCMC.r` or `SimpMCMCJpeg.r` are very simple codes. Maybe there are better ways to do it, but the code gets the job done. The difference between the two files is that one produces `pdf`-files and the other `jpeg`-files.

The basic structure:

1. Assign the beginning values to the model
2. Initialize the vector of values for the chain. ("Zero them out")
3. Loop over the chain
4. Use the sampled values

1 Assign the beginning values

- We have the following model for heights:

$$\begin{aligned} X_i|\mu,\tau &\sim N(X_i|\mu,\tau) \\ \mu|\mu_0,\tau,\theta &\sim N(\mu|\mu_0,\tau\theta) \\ \tau|\alpha,\beta &\sim \text{Gam}(\tau|\alpha,\beta) \end{aligned}$$

- Priors: $\mu_0 = 66$, $\theta = 4$, $\alpha = 1$, $\beta = 25$.
- Data: $X = (64, 73, 64, 63, 69, 71)$, $n = 6$, $\bar{X} = 67.333$, and $\sum(X_i - \bar{X})^2 = 89.33$.

1 Assign the beginning values

The following code will do that:

```
# prior
mu0<-66; theta<-4; alp<-1; bet<-25

# data
x<-c(64, 73, 64, 63, 69, 71)
xbar<-mean(x)
n<-length(x)
```

2 Initialize the values

Now we have to assign values for item before we go into the loop.

This includes doing the following:

- "zero" out the values that we will sample in the chain. R will feel uncomfortable if you keep "adding" elements to a vector. So, it is better to first make the vector (with zero's) and then reassign them values are you sample them.
- Set up the starting point of the chain.
- Calculate values that will not change inside the loop so that you don't have to recalculate inside the loop.

2 Initialize the values

The following code will do that:

```
# starting values
muStart<-20;tauStart<-0.0025;

#  MCMC parameters
nbig<-50000;
set.seed(333); # The purpose of set.seed is to get the same random
```

(cont)

2 Initialize the values

The following code will do that:

```
# Initialize chain
mu<-rep(nbig, 0)
tau<-rep(nbig,0)

mu[1]<-muStart; tau[1]<-tauStart

mu0p<- (theta*mu0+n*xbar)/(theta+n)  # does not change in loop
alpp<-alp+(0.5)*(n+1)                # does not change in loop

tau0p<-rep(nbig,0)    # values that we will we track in the loop
betp<-rep(nbig,0)      # values that we will we track in the loop
```

3 Loop over the chain

Here, we do the actual looping over the chain.

3 Loop over the chain

The following code will do that:

```
for(i in 2:nbig){
# sample mu
tau0p[i]<-tau[i-1]*(n+theta)
mu[i]<-rnorm(1,mean=mu0p,sd=1/sqrt(tau0p[i]))

# sample tau
betp[i]<- bet+ 0.5*( sum( (x-mu[i])^2) +
                     theta*(mu[i]-mu0p)^2)
tau[i]<-rgamma(1,alpp,betp[i])
}
```

4 Use the sampled values

Now the vectors `mu` and `tau` contain the sample from the MCMC. So, one can treat them like samples from the posterior distribution.

For example if you wanted to plot the density of μ without the first 5 values, you could type:

```
plot(density(mu[-(1:5)]))
```

Normal Models

Now that we have reviewed the simple normal model, let us look at some of the other basic normal models.

- One-way ANOVA model. These are one of the simplest models. This model shows some of the strengths of using MCMC methods.
- Multilevel models. Here we see that once we understand the ANOVA model it is quite easy to extend the ANOVA model to multilevel models with MCMC methods.

Normal Models

(Continued)

- Linear regression an introduction to Bayesian software. Here Bayesian software package BUGS is introduced. With this software, we do not have to worry about deriving the conditional distributions. All one has to do is specify the model (the likelihood and the prior) and the program constructs the required conditional distributions.

One Way ANOVA

- The one way ANOVA is one of the simplest normal models. However, getting the mathematical expression for the joint posterior distribution can be quite involved. See Box and Tiao for the details. (Box, GEP, and Tiao,GC, *Bayesian Inference in Statistical Analysis*, 1973, reprinted in 1992). Although there are good approximations that can be done.
- However, it is fairly easy to apply MCMC methods to this model. As we will see below, one simply needs the basic conditional distribution used for the normal model which was discussed in the morning.

Example 8: Oxygen Measurements

Now lets consider the problem of a laboratory science who is investigating oxygen affinity in hemoglobin.

- In the experiment, the scientist measure the amount of oxygen in several subjects under three different laboratory conditions.
- Let Y_{ij} be the oxygen measurement of the jth subject under laboratory condition i.
- Let n_i be the number of samples for each condition. Also, let $\bar{Y_{i}}.$ be the mean of the values under condition i.

Example 8: Oxygen Measurements

Here is the data:

Condition		
1	2	3
7	15	11
9	11	10
5	15	15
5	10	17
10	17	4
11	15	8
7	11	
12		
5		

Model for ANOVA

Assume the following model for this data:

$$
\begin{aligned}
Y_{ij}|\mu_i, \tau &\sim \text{Normal}(\mu_i, \tau) \\
\mu_i|\mu_0, \tau_0 &\sim \text{Normal}(\mu_0, \tau_0) \\
\mu_0|\mu_{00}, \tau_{00} &\sim \text{Normal}(\mu_{00} = 10, \tau_{00} = .001) \\
\tau|\alpha, \beta &\sim \text{Gamma}(\alpha = .01, \beta = .01) \\
\tau_0|\alpha_0, \beta_0 &\sim \text{Gamma}(\alpha_0 = .01, \beta_0 = .01).
\end{aligned}
$$

Model for ANOVA

(continued)

- So, by the above notation, the conditional distribution of Y_{ij} is a normal distribution with mean μ_i and precision τ.
- Similarly, the conditional distribution of τ is a gamma distribution with parameters α and β and where both of the parameters values are set equal to .01.
- Note: these are vague priors.

The Gibbs Sampler

To analyze this model, a Gibbs sampler algorithm is used.

- In order to do this, we need to know that conditional distribution of all the parameters in the model conditioning on the other parameters of the model.
- So, when we sample each parameter value separately, we will have a value for the other parameters. These other values will be the values which had previously been sampled.

The Conditional Distribution of μ_i

First, let's look at the conditionally distribution of μ_i which is the theoretical mean for the data for one of the experimental conditions.

- We are also assuming that we know a value for all the other parameters in the model.
- Without loss of generality, lets assume this is the mean for the first condition.
- So, we want to get the conditional distribution:

$$f(\mu_1|Y_{ij}, \mu_2, \mu_3, \mu_0, \mu_{00}, \tau, \tau_0, \tau_{00})$$

The Conditional Distribution of μ_i

(continued)

- The first thing to note, is that if we know μ_0, then the parameters μ_{00} and τ_{00} don't give us any more information. That is, once we observe the value for μ_0 knowing what distribution it came from does not give us any more information about μ_1.
- Similarly, once we know τ_0, then knowing α_0 and β_0 is not very useful. Also, knowing the parameters μ_0 and τ_0 means that the parameters μ_2 and μ_3 are not needed.
- Similarly, all the sampled values from conditions 2 and 3 are no longer needed. Also, since we know the value of τ then we no longer need to know α and β.

- So, we see that:

$$f(\mu_1|Y_{ij}, \mu_2, \mu_3, \mu_0, \mu_{00}, \tau, \tau_0, \tau_{00}) = f(\mu_1|Y_{1j}, \mu_0, \tau, \tau_0)$$

  Therefore, given Y_{1j}, μ_0, τ, and τ_0, μ_1 is conditionally independent of Y_{2j}, Y_{3j}, μ_2, μ_3, μ_{00}, and τ_{00}.

- So, this conditional probability reduces to the previous case of the normal distribution with unknown μ and know τ.

- That is, the conditional posterior distribution of μ_1 is normal with posterior mean μ'_{0i} and precision τ'_{0i}, with τ'_{0i} equal to $\tau_0 + n_1\tau$ and with

$$\mu'_{0i} = \frac{\tau_0\mu_0 + n_1\tau\bar{Y_1}_\cdot}{\tau_0 + n_i\tau}.$$

- Similarly, we have the conditional distribution for μ_2 and μ_3.

Conditional Probability of τ

- In order to get the conditional distribution of τ, please note that since we have a values of the μ_i's then we don't need the parameters μ_0, τ_0, μ_{00}, and τ_{00}. That is, given the mu_i's, then τ is conditionally independent of those parameters.
- So, given the data and the μ_i's, then the posterior distribution of τ is a gamma distribution with parameter α' equal to $\alpha + (n_1 + n_2 + n_3)/2$ and β', where

$$\beta' = \beta + \frac{1}{2}\sum_{i=1}^{3}\sum_{j=1}^{n_i}(Y_{ij} - \mu_i)^2.$$

Conditional Probability of μ_0

- When calculating the conditional probability of the μ_i's and of τ, we found that these parameters were conditionally independent of several of the other parameters in the model. So, only some of the other parameters were needed for the conditional probability.
- In this case, when we know the values of the μ_i's, then we don't need the data value Y_{ij}. That is, for the conditional distribution of μ_0 we only need the μ_i's, τ_0, and the parameters for the prior for μ_0.

Conditional Probability of μ_0

(continued)

- This is now another case of a normal distribution family with unknown mean and known precision parameter. Therefore, the posterior distribution of μ_0 is normal with precision $(3\tau_0 + \tau_{00})$ and mean

$$\mu'_{00} = \frac{3\tau_0 \bar{\mu}. + \tau_{00}\mu_{00}}{3\tau_0 + \tau_{00}},$$

  where $\bar{\mu}.$ is the mean of the μ_i's.

- Also, note that the number 3 in the above formula appears there because there are three different experimental conditions in this data. If there was a different number of groups, then this number would of course change.

Conditional Probability of τ_0

- When we calculated the conditional distribution of μ_0 we saw that data, Y_{ij} no longer directly came into the calculation. This happens again when calculating the conditional distribution for τ_0.
- When the parameters μ_i and μ_0 are know, then the data is conditionally independent of τ_0.
- The posterior conditional distribution of τ_0 is a gamma distribution with parameters α_0' equal to $\alpha_0 + 3/2$ and β_0' which equals:

$$\beta_0' = \beta_0 + \frac{1}{2}\sum_{i=1}^{3}(\mu_i - \mu_0)^2$$

Conditionally Independent Hierarchical Models

- Please note that although this model is more complicated than the simple height example, the model broke up into pieces which are just as easy to analyze as the height example.
- The important thing to note that when you “know” the other parameters in the model, then complex models often break up into small easy to work with smaller models which are conditionally independent of the other parts of the model.
- In fact, later in this course, we will see that very complex models can often be broken into these simpler, smaller models and we will be able to tackle models which are extremely difficult to handle with other methods.
- This structure is known as a conditionally independent hierarchical model.

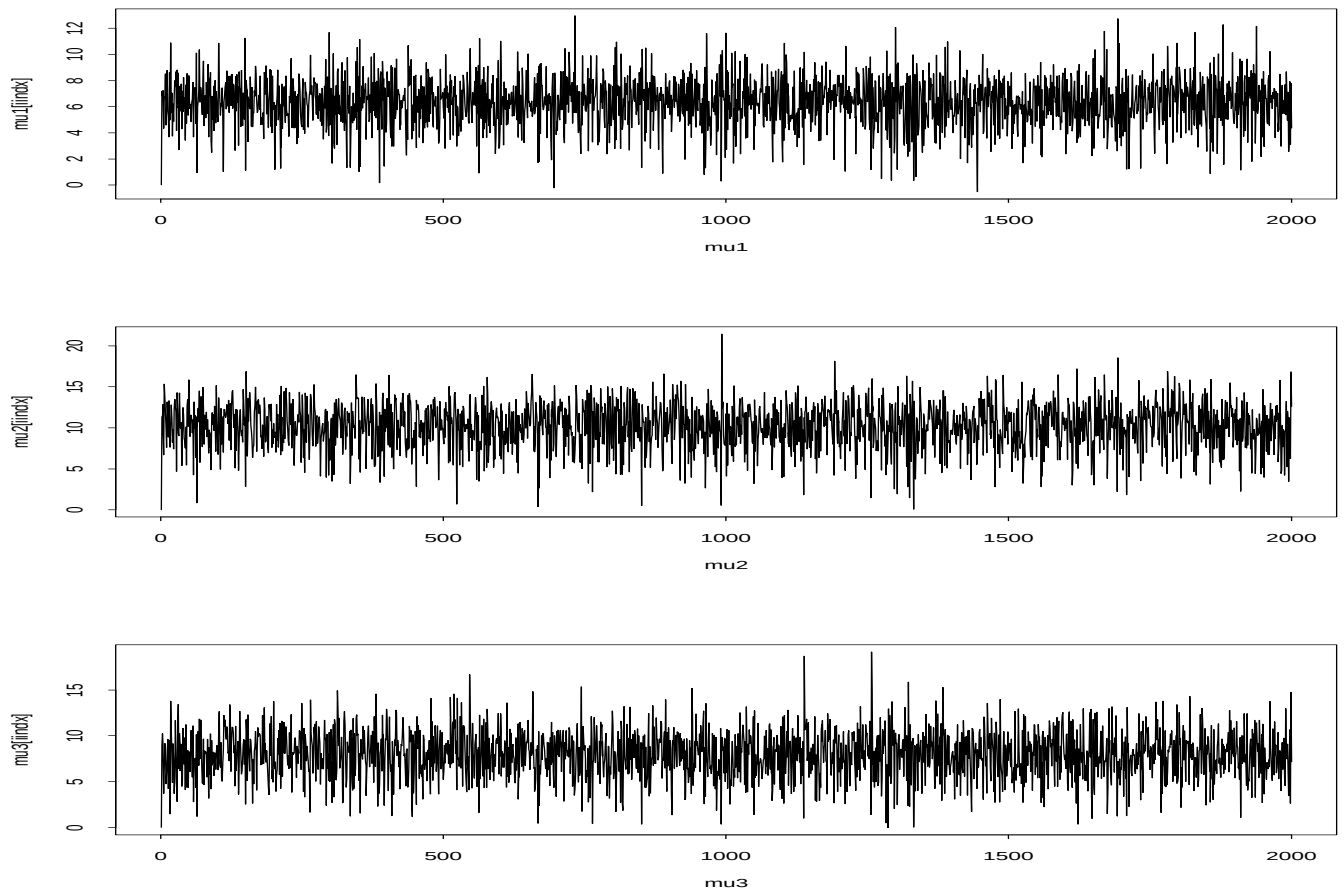

Figure 1: This is the trace plot for the three means.

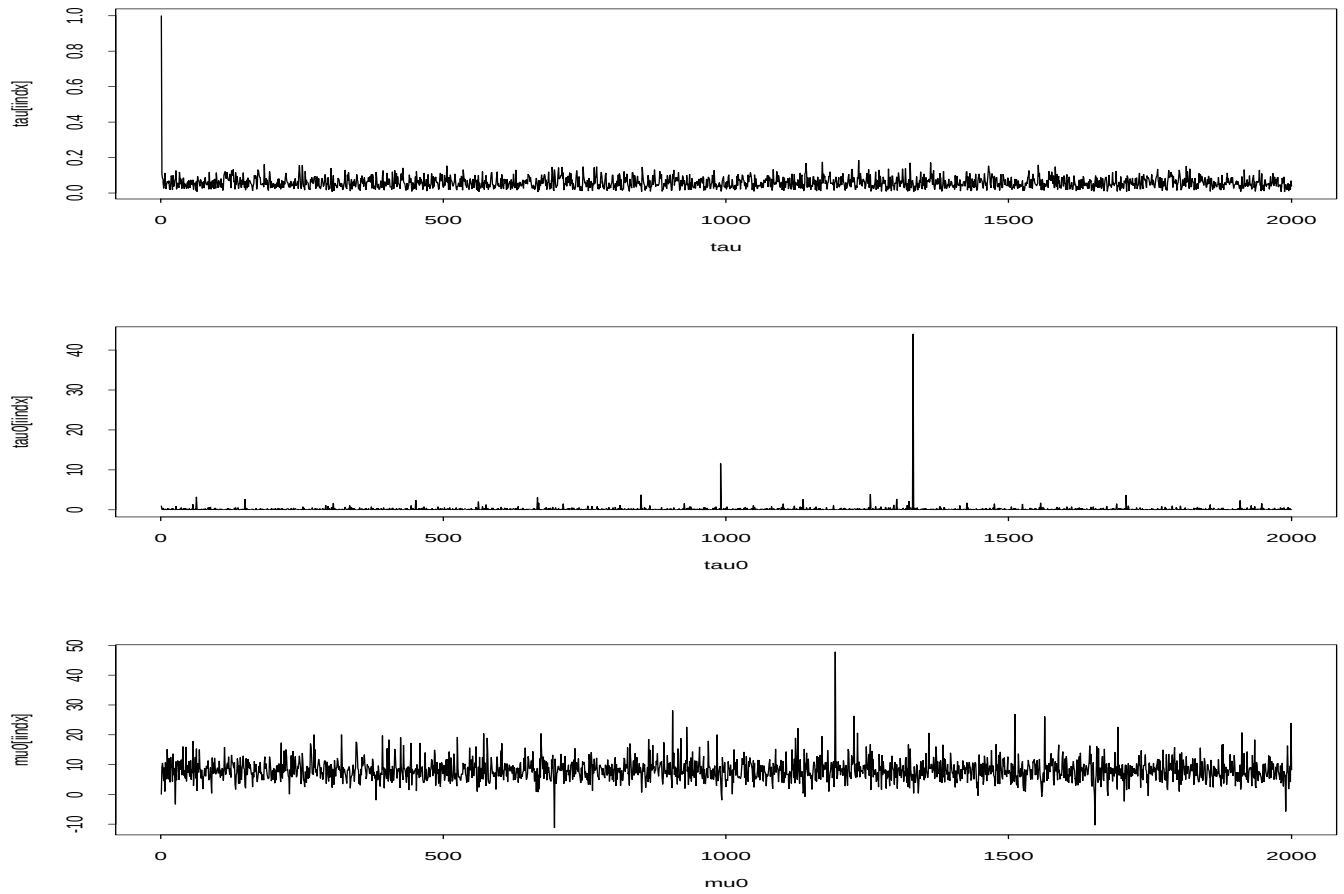

Figure 2: This is the trace plot for the τ, τ_0, and μ_0.

- From the trace plots, we see that the samples quickly cycle through the main probability mass. So, we are running enough samples to be "close" to the posterior distribution.
- The large spikes present in the trace plots of τ_0 and μ_0 are discussed a little later in this example.
- To further see if things are good, plot the estimated densities from three different sampled chains.
- This gives a sense of the accuracy of the estimates since the three chains are independent.

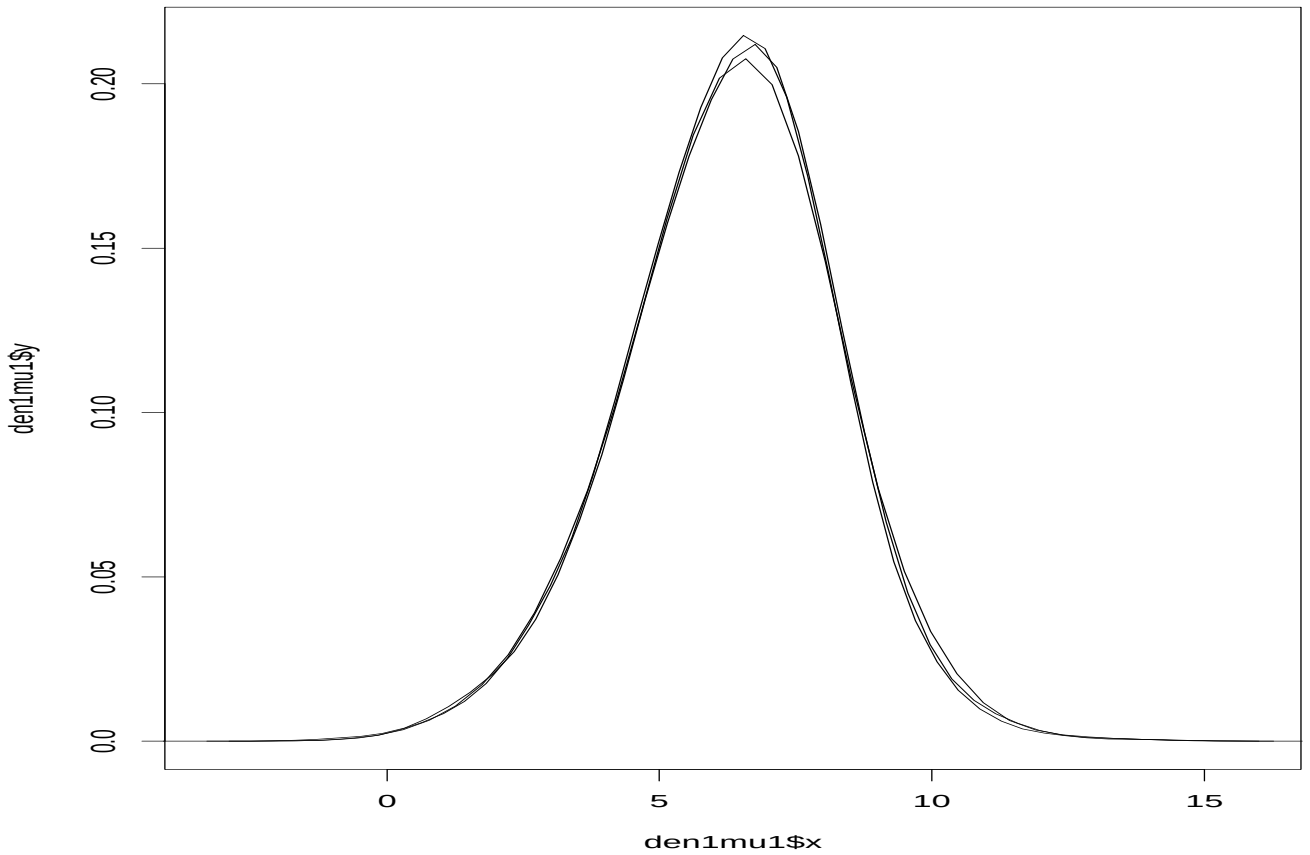

Figure 3: This is an estimate of the posterior density of μ_1 using three different sampled chains.

- Since the density estimates are basically the same given estimates from independent chains, then we can be fairly confident of the estimates.
- Now, the scientist wanted to know if one of the processes was better than any of the others.
- To look at this, let's start by looking at the posterior distributions of the three means.

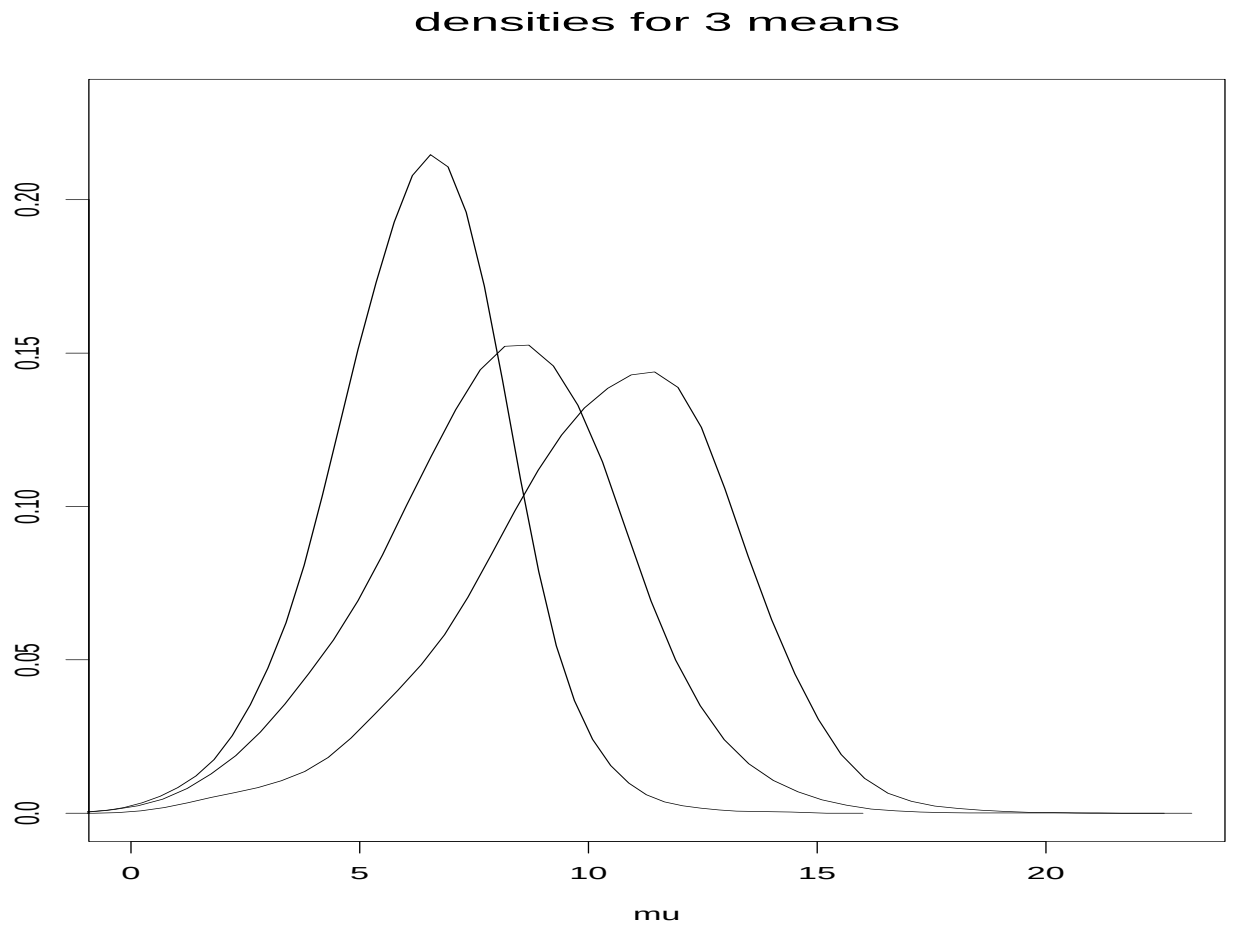

Figure 4: This is an estimate of the posterior density of μ_1 using three different sampled chains. From left to right: densities for μ_1, μ_3, and μ_2.

Posterior Distribution for μ_i's

Here is a table of the summary of the posterior distributions for the μ_i's.

	Mean	SD	Median	95% CR
μ_1	6.28	1.86	6.39	(2.40,9.77)
μ_2	10.25	2.77	10.52	(4.14,9.77)
μ_3	8.05	2.58	8.19	(2.69, 12.87)

Differences

- The scientist is primarily interested in looking at the difference between the different means. In order to do this, we look at the distribution of the difference between any two of the means.
- So, define the following random variables:

$$\begin{aligned} \delta_{12} &= \mu_1 - \mu 2 \\ \delta_{13} &= \mu_1 - \mu_3 \\ \delta_{23} &= \mu_2 - \mu_3 \end{aligned}$$

- The posterior distribution of these random variables can be estimated from the sampled values: $\delta_{ij}^{(m)} = \mu_i^{(m)} - \mu_j^{(m)}$.

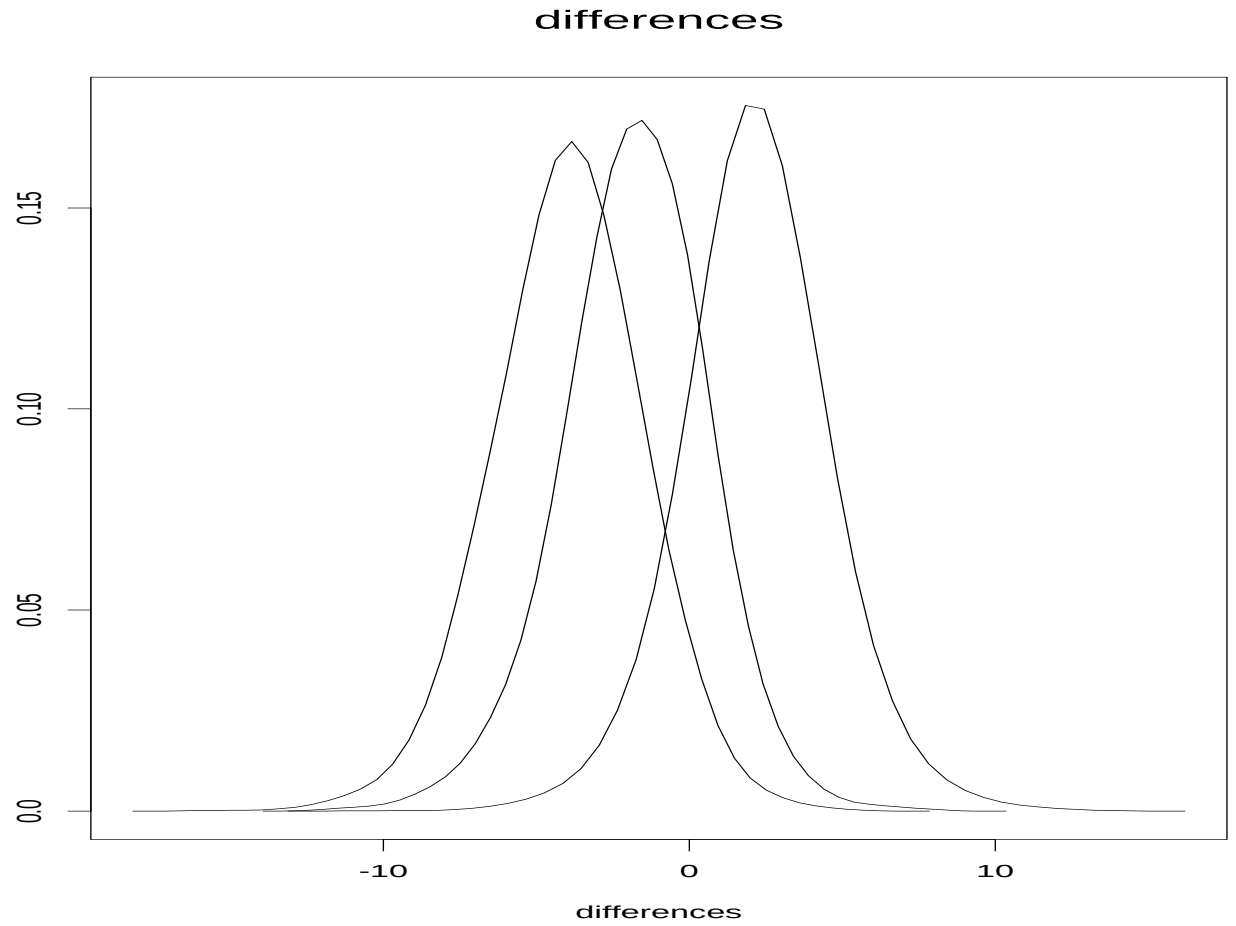

Figure 5: This is an estimate of the posterior density of differences between the means. From left to right: δ_{12}, δ_{13}, and δ_{23}

Here is a table of the summary of the posterior distributions for the differences:

	Mean	SD	Median	95% CR	Prob. of Diff.
δ_{12}	-3.97	2.41	-3.92	(-8.89, 0.60)	P($\mu_1 > \mu_2$)=.046
δ_{13}	-1.77	2.34	-1.71	(-6.72, 2.62)	P($\mu_1 > \mu_3$)=.221
δ_{23}	2.20	2.34	2.16	(-2.48, 6.95)	P($\mu_2 > \mu_3$)=.849

From the above, there appears to be some evidence that condition 1 is smaller than condition 2.

- The posterior probability that μ_1 is smaller than μ_2 is about 95%.
- The 95% credible region for the difference, δ_{12} just barely contains zero.
- There isn't any good evidence here that condition 3 is different than either conditions 1 or 2.

Posterior for other parameters

- Now let's look at the posterior distribution of the parameters τ, τ_0, and μ_0. Below is a table of the summary statistics for the posterior distribution:

	Mean	SD	Median	95% CR
τ	.059	.031	.054	(.013,.132)
τ_0	.173	.338	.089	(.005, .818)
μ_0	8.27	3.80	8.01	(2.05, 16.52)

- The following plots show the posterior distribution of τ, τ_0, and μ_0.

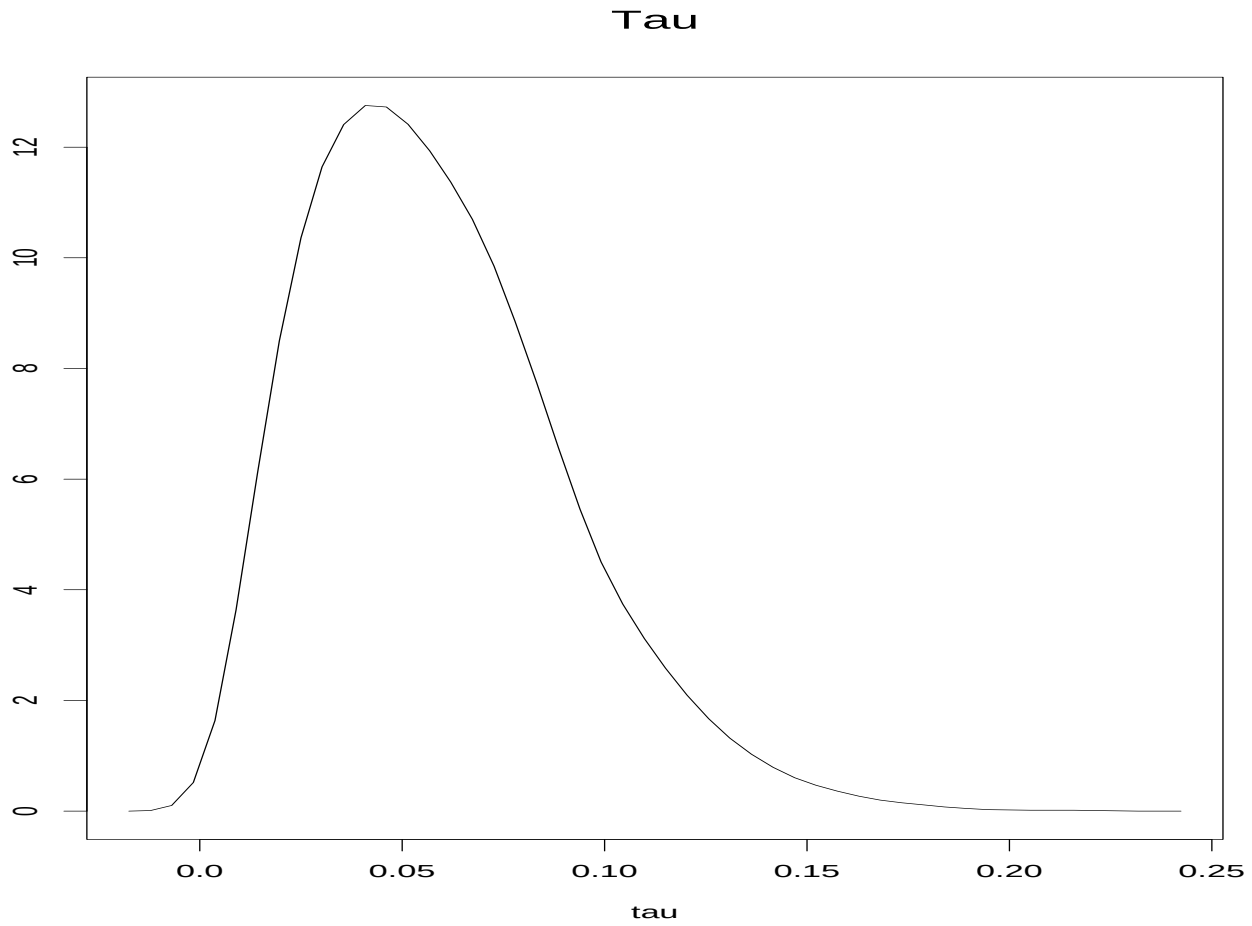

Figure 6: This is an estimate of the posterior density of τ.

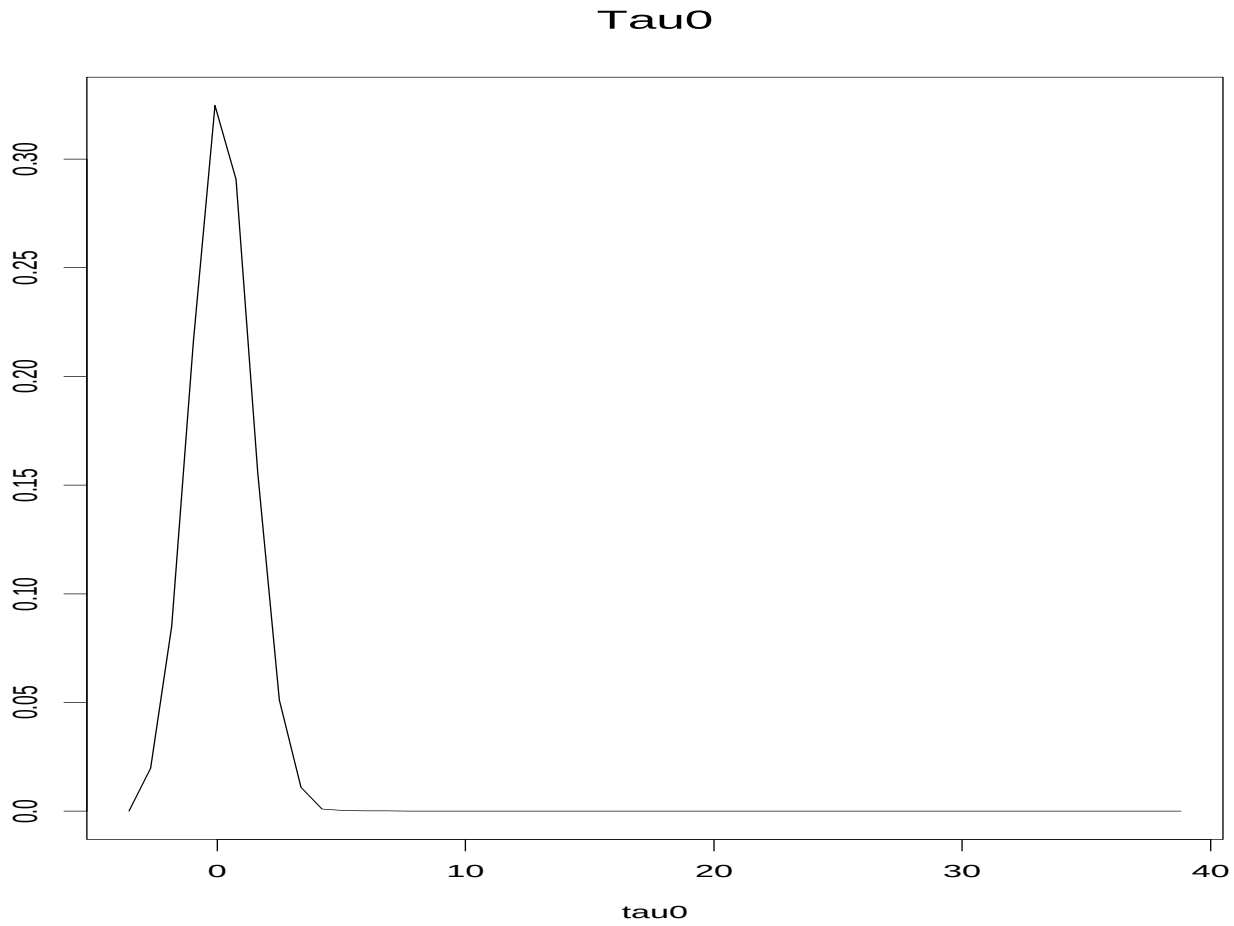

Figure 7: This is an estimate of the posterior density of τ_0.

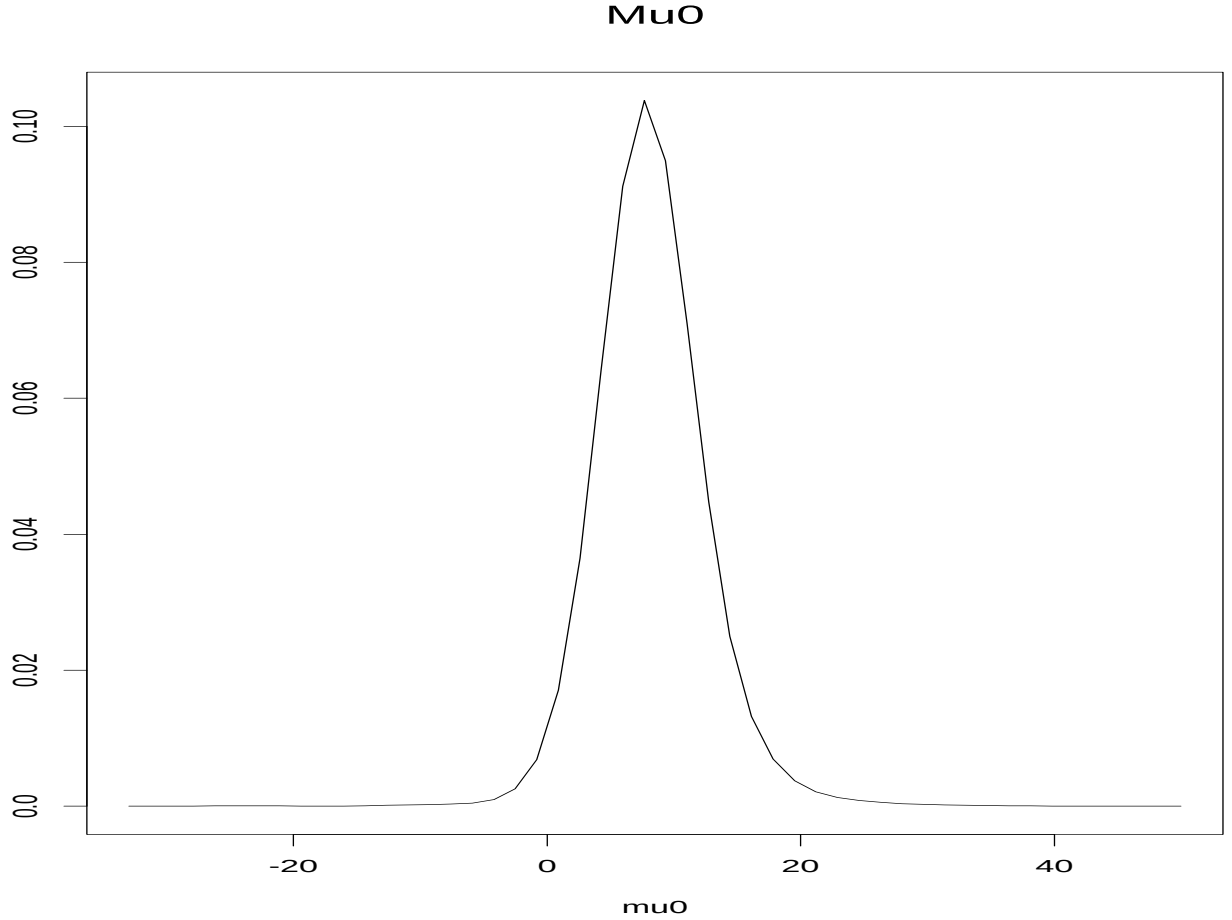

Figure 8: This is an estimate of the posterior density of μ_0.

Posterior for Precision and μ_0.

- Please note that the posterior distribution for τ_0 appears to have a long tail. Although the main mode is in the area between .005 and .818, the distribution still occasionally has very large values.
- In comparison, the distribution for τ appears much tighter.
- This is because the "data" for τ_0 for each iteration of the MCMC is only the three means, μ_1, μ_2, and μ_3. In addition, these data points are sampled at each pass. Therefore the degrees of freedom for the posterior of τ_0 is very small. In comparison, the data used to estimate τ is all the Y_{ij} values (and there are 22 of these values). So, there are more degrees of freedom used to estimate τ so the variation for this posterior is much smaller.

Posterior for Precision and μ_0.

(Continued)

- Similarly, there is not a lot of information gained from the data for the parameter μ_0. At each iteration, the only "data" is the last sampled value for μ_1, μ_2, and μ_3. So, again, there are only 3 "pseudo-data" points used.

Functions of the Precisions

- For the precision parameters, one statistic of interest might be the intraclass correlation. This has the following form:

$$\text{ICC}(\tau, \tau_0) = \frac{\text{VAR}(\mu_i|\mu_0)}{\text{VAR}(\mu_i|\mu_0) + \text{VAR}(Y_{ij}|\mu_i)} = \frac{1/\tau_0}{1/\tau_0 + 1/\tau}.$$

- Again, from the sample of τ and τ_0 values, one can simply calculate the value of $\text{ICC}^{(m)}$ from the sequence of values $\tau^{(m)}$ and $\tau_0^{(m)}$ obtained from the MCMC sampler. From here, one can learn about the posterior distribution of ICC.
- Next, is a plot of the posterior distribution of ICC.

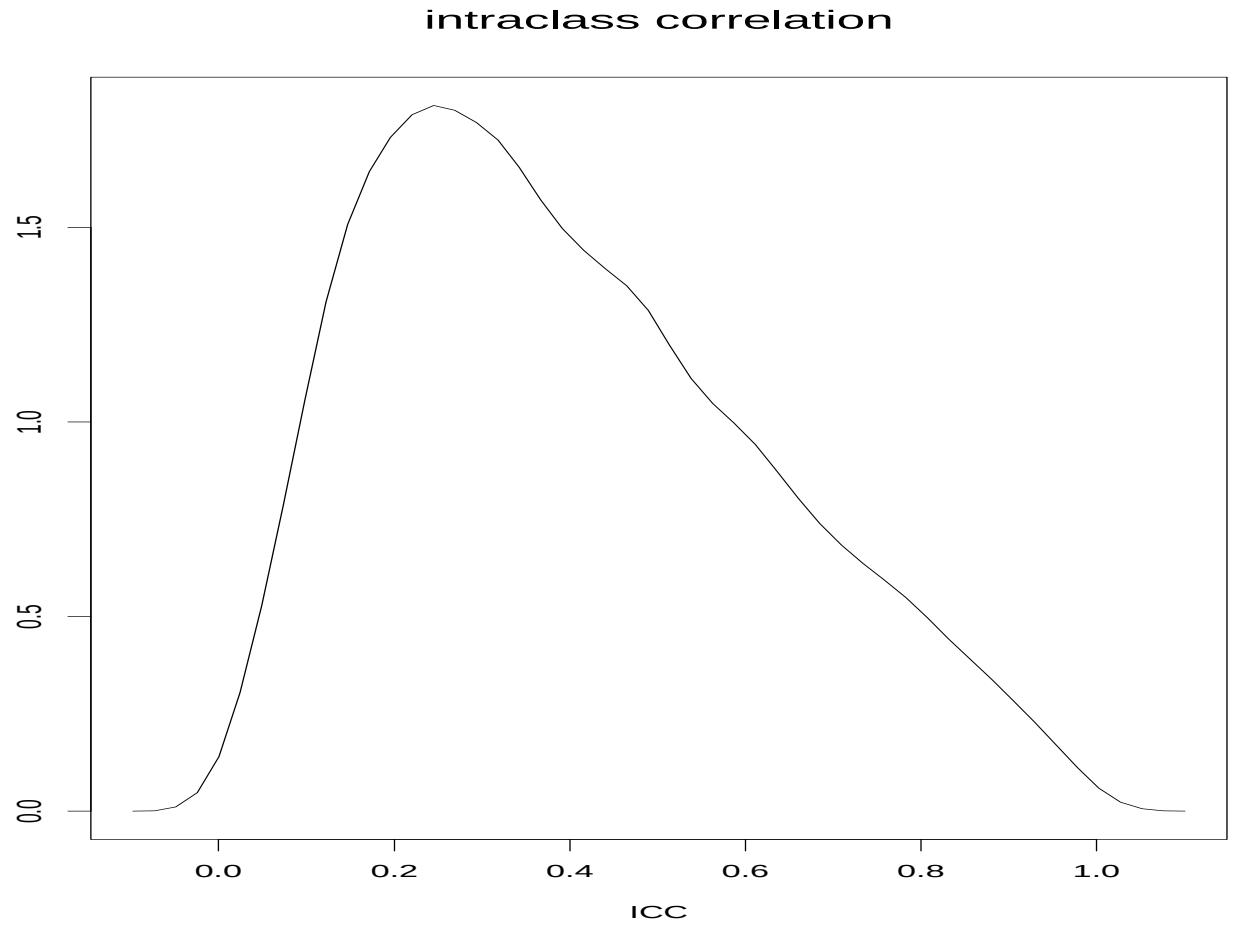

Figure 9: This is an estimate of the posterior density of differences between the means.

Anova Example: Comments

- One can break large complex models into smaller submodels based on the conditionally independent hierarchical structure. This allows one to develop a simple algorithm for potentially very complex models.
- The sampler can quite easily be used to calculate the posterior distribution of different functions of the parameters. Examples here include calculating the difference scores and the intraclass correlation.

Multilevel Models

- Note that in the 1-way ANOVA model, when sampling a new grand mean parameter (the μ_0), the conditional distribution used in the sampler did not use the observed values (the Y_{ij}'s), but instead simply used the latest sampled values of the group means (the μ_i's) instead.
- Consider a problem where one is looking at educational measures of school children. Suppose that there is a test given to several children. The children are grouped into classrooms. The classrooms in turn are grouped in school districts which may themselves be grouped into provinces, etc.

Multilevel Models

(continued)

- Consider the analysis of test scores in a province. Let Y_{ijk} be the test score for the i$^{\text{th}}$ child in the j$^{\text{th}}$ classroom in the k$^{\text{th}}$ school district. Also, the indices i, j, and k are such that $i = 1, \ldots, n_{jk}$, $j = 1, \ldots, n_k$, and $k = 1, \ldots, n$.
- Let the conditional distribution of Y_{ijk} be normal with mean μ_{jk} and precision τ_1.
- Let μjk have normal distribution with mean μ_k and precision τ_2. Let μ_k have a normal distribution with mean μ_0 and precision τ_3. Let the τ_m's have a gamma distribution with parameters α_m and β_m.

Multilevel Models

(continued)

- So we have the following model:

$$
\begin{aligned}
Y_{ijk}|\mu_{jk},\tau_1 &\sim N(Y_{ijk}|\mu_{ij},\tau_1)\\
\mu_{jk}|\mu_k,\tau_2 &\sim N(\mu_{jk}|\mu_k,\tau_2)\\
\mu_k|\mu_0,\tau_3 &\sim N(\mu_k|\mu_0,\tau_3)\\
\mu_0|\mu_{00},\tau_{00} &\sim N(\mu_0|\mu_{00},\tau_{00})\\
\tau_1|\alpha_1,\beta_1 &\sim \text{GAM}(\tau_1|\alpha_1,\beta_1)\\
\tau_2|\alpha_2,\beta_2 &\sim \text{GAM}(\tau_2|\alpha_2,\beta_2)\\
\tau_3|\alpha_3,\beta_3 &\sim \text{GAM}(\tau_3|\alpha_3,\beta_3),
\end{aligned}
$$

and to specify the priors, one assigns values to μ_{00}, τ_{00}, α_1, β_1, α_2, β_2, α_3, and β_3.

Multilevel Models: Sampling

- Notice that in the 1-way ANOVA model, when estimating the grand mean parameter (μ_0), the conditional distribution used in the sampler did not use the observed values Y_{ij} but just used the latest sampled values of the "near-by" parameters: μ_i, τ, μ_{00}, and τ_{00}.
- Sampling for parameters for the multilevel model can be accomplished in a similar way. We only need the "near-by" parameters. For example, let us consider the conditional distribution needed for to estimate μ_1, the mean of the first school district.

Multilevel Models: Sampling

(continued)

- The first school district has n_1 classrooms which each have a classroom mean μ_{j1}
  - These classroom means in the first school district have a prior distribution which is normal with mean μ_1 and precision τ_2. The parameters μ_1 and τ_2 are the district level parameters.
  - Note that if one knows the value of μ_{j1}, then there is no additional information about μ_1 from the individual scores of the children in the classrooms. So, when constructing the sampler for μ_1, one does not need to know the values of Y_{ij1}'s.
  - By a similar argument, one can see that to produce the conditional distribution for μ_1, it is sufficient to only know the values of the μ_{i1}'s, τ_2, μ_0, and τ_3. So, we are just interested in the following submodel:

$$\begin{aligned} \mu_{j1} &\sim N(\mu_{j1}|\mu_1, \tau_2) \\ \mu_1 &\sim N(\mu_1|\mu_0, \tau_3). \end{aligned}$$

  - The conditional posterior distribution for μ_1 is just the posterior distribution for μ_1 from the above submodel where we take as know the latest sampled values of μ_{i1}'s, τ_2, μ_0, and τ_3.

– From lecture 1, we know that the posterior distribution for this model is normal with precision $n_{j1}\tau_2 + \tau_3$ and the following mean:

$$\frac{n_{j1}\tau_2\bar{\mu}_{\cdot 1} + \tau_3\mu_0}{n_{j1}\tau_2 + \tau_3},$$

where $\bar{\mu}_{\cdot 1} = (1/n_{j1}) \sum_{j=1}^{n_{j1}} \mu_{j1}$.

Example 9: School Math Scores

Grade 3 Math scores.

- 15 Districts with 1 to 16 schools.
- 92 Schools with 17 to 181 students.
- 8383 Students.

Student Scores

District 1				· · ·	District 15
School 1	School 2	· · ·	School 5	· · ·	School 1
450	540	· · ·	410	· · ·	480
430	460	· · ·	410	· · ·	390
450	460	· · ·	450	· · ·	380
⋮	⋮	⋮	⋮	⋮	⋮

Example 9: Model

$$\begin{aligned} \text{Score} &\sim N(\text{SchoolMean}, \text{TauSc}) \\ \text{SchoolMean} &\sim N(\text{DistrictMean}, \text{TauSch}) \\ \text{DistrictMean} &\sim N(\text{GMean}, \text{TauDst}) \end{aligned}$$

With:

$$\begin{aligned} \text{TauSc, TauSch,TauDst} &\sim \text{Gamma}(.00001, .00001) \\ \text{GMean} &\sim N(500, .0001) \end{aligned}$$

Software: WinBugs, OpenBugs, and Jags

These are a public domain software packages which will automatically produce the necessary MCMC code for your model.

- WinBugs was the first and was a major innovation for Bayesian computing.
- Makes calculations for these models very easy.
- Some variations are now appearing.
- Can be called from R and other software packages.
- WinBugs is no longer being updated. So, use either OpenBugs or Jags. My code will be in OpenBugs. Jags code is similar.

Software: WinBugs

Example of code for the above model:

```
model{
  for(i in 1:Nstud){
    Score[i] ~dnorm(SchMean[School[i]], tauSc)}
  for(isch in 1:Nsch){
    SchMean[isch]~dnorm(DistMean[DistOfSch[isch]],tauSch)}
  for(idst in 1:Ndist){
    DistMean[idst]~dnorm(Gmean,tauDst)}
```

Software: WinBugs

(continue)

```
  Gmean~dnorm(500,.0001)
  tauSc~dgamma(.00001,.00001)
  tauSch~dgamma(.00001,.00001)
  tauDst~dgamma(.00001,.00001)
  }
```

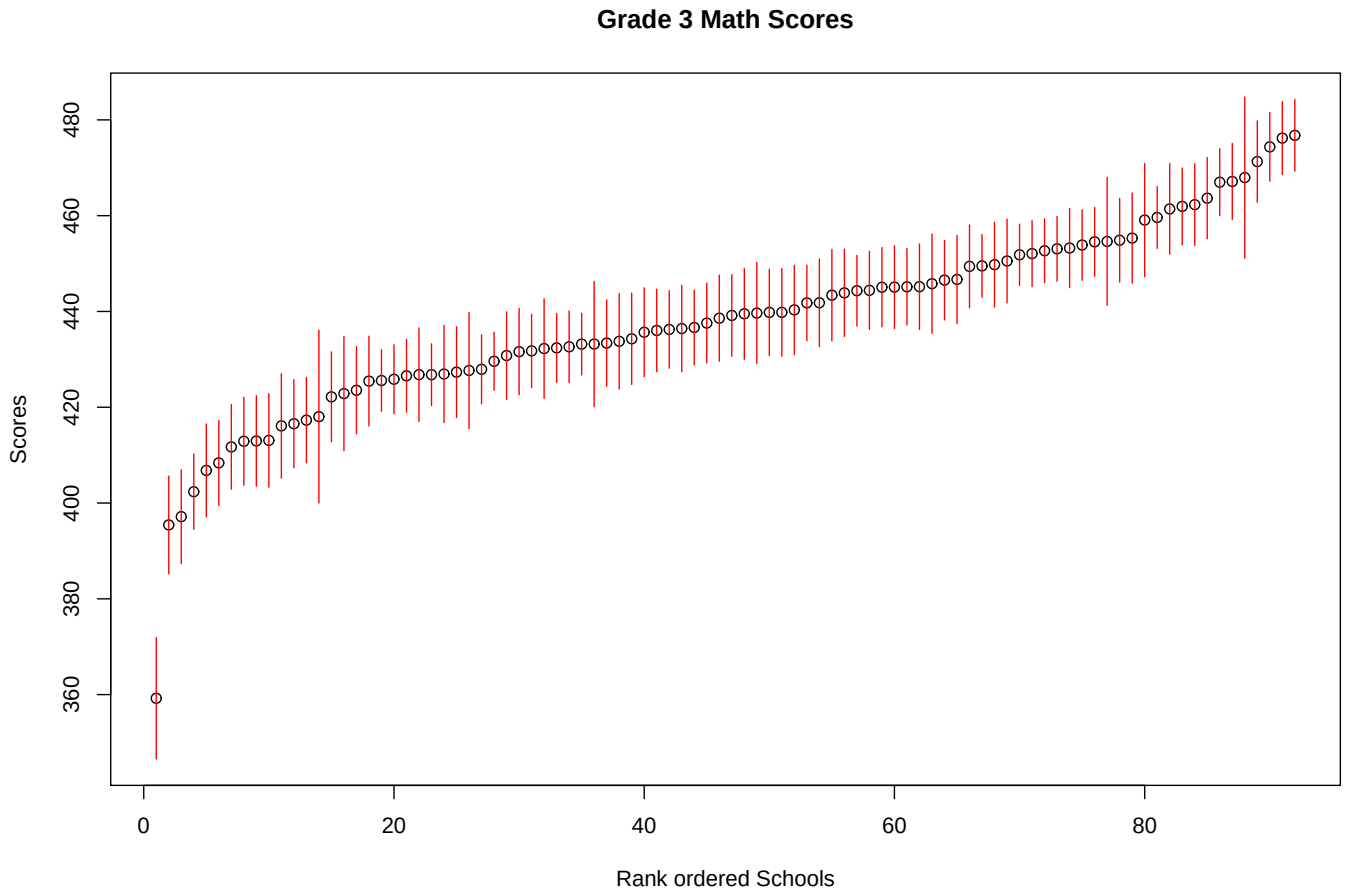

Figure 10: Posterior means with the 95% credible regions (in red) for the mean score of each school. Schools are ordered by their posterior means.

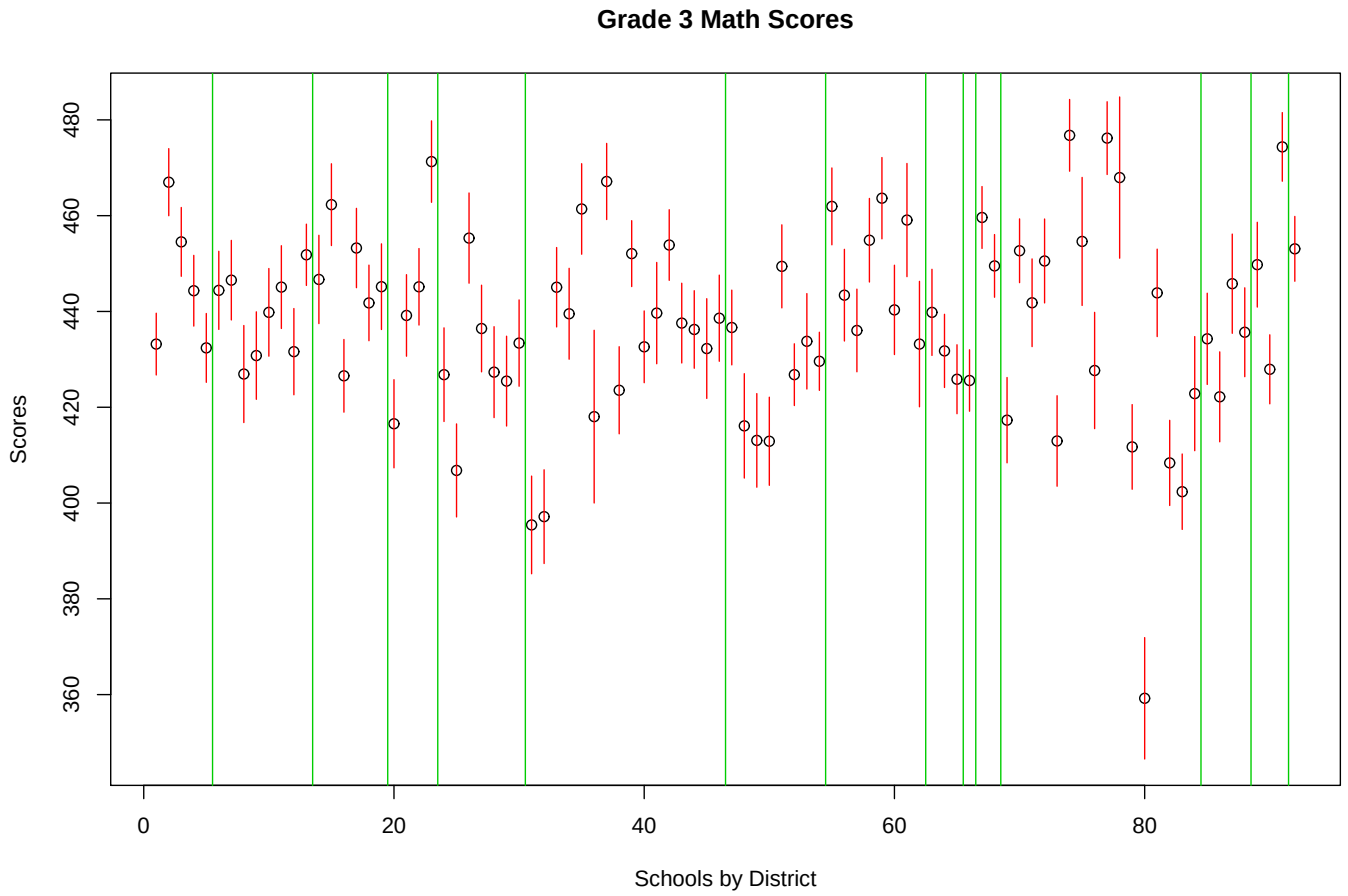

Figure 11: Posterior means with the 95% credible regions (in red) for the mean score of each school. Green lines separate the different districts.

Multilevel Models: Comments

- Similarly, the sampler for the other parameters can be constructed and they turn out to have a simple form like the 1-way ANOVA example.
- Also note that there is no increased complexity in making these conditional samplers if one adds more levels to the hierarchical model. The samplers for each parameter just use the local submodel.
- Again, conditionally independent hierarchical models are nicely handled by MCMC methods.

Linear Regression and Bayesian Software

Here we look at the basic linear regression model. Also, a Bayes software package is now introduced and used to perform inference on this model.

- Linear regression is another basic normal model. Suppose we have paired values X_i and Y_i and we assume that we can predict Y_i as a linear function of X_i. Let the linear function be $\alpha + \beta X_i$.
- The usual likelihood form for this model is that for each i, Y_i is independently normally distributed with mean $\alpha + \beta X_i$ and precision τ. It is assumed that the values X_i are known and we wish to make inference on (α, β, τ).

Linear Regression and Bayesian Software

(continued)

- For priors on this model, there are of course many choices. In keeping with our other work on normal models, here we assume that τ has a gamma distribution. For priors on α and β one could consider a joint multivariate distribution for a joint prior on these two parameters. For simplicity, here we only look at modelling α and β with separate independent normal priors.

Linear Regression and Bayesian Software

(continued)

Therefore, the following is a basic model for linear regression:

$$\begin{aligned} Y_i|\alpha,\beta,X_i,\tau &\sim N(Y_i|\alpha+\beta X_i,\tau) \\ \alpha|\mu_\alpha,\tau_\alpha &\sim N(\alpha|\mu_\alpha,\tau_\alpha) \\ \beta|\mu_\beta,\tau_\beta &\sim N(\mu_\beta,\tau_\beta) \\ \tau|a_\tau,b_\tau &\sim \mathsf{Gam}(a_\tau,b_\tau) \end{aligned}$$

To illustrate this method, we consider a simple example.

Example 9: Brain vs Body Weight

- Weisberg, using a data set originally from Allison and Cicchetti (1976)[a] looked at the relationship between the average brain weight and body weight of several species of mammals.
- The purpose of this analysis is to look at the relative size and to see which species have a higher brain weight than would be predicted by their body weight.
- In this analysis, the expected log(brain) weight is modeled as a linear function of the log(body) weight. So, the log(brain) is the "Y" variable and the log(body) is the "X" variable.
- Figure 12 shows the plot of the log transformed values.

[a] Weisberg, Applied Linear regression. Originally source: Allison and Cicchetti, 1976, "Sleep in mammals: Ecological and constitutional correlates." Science.

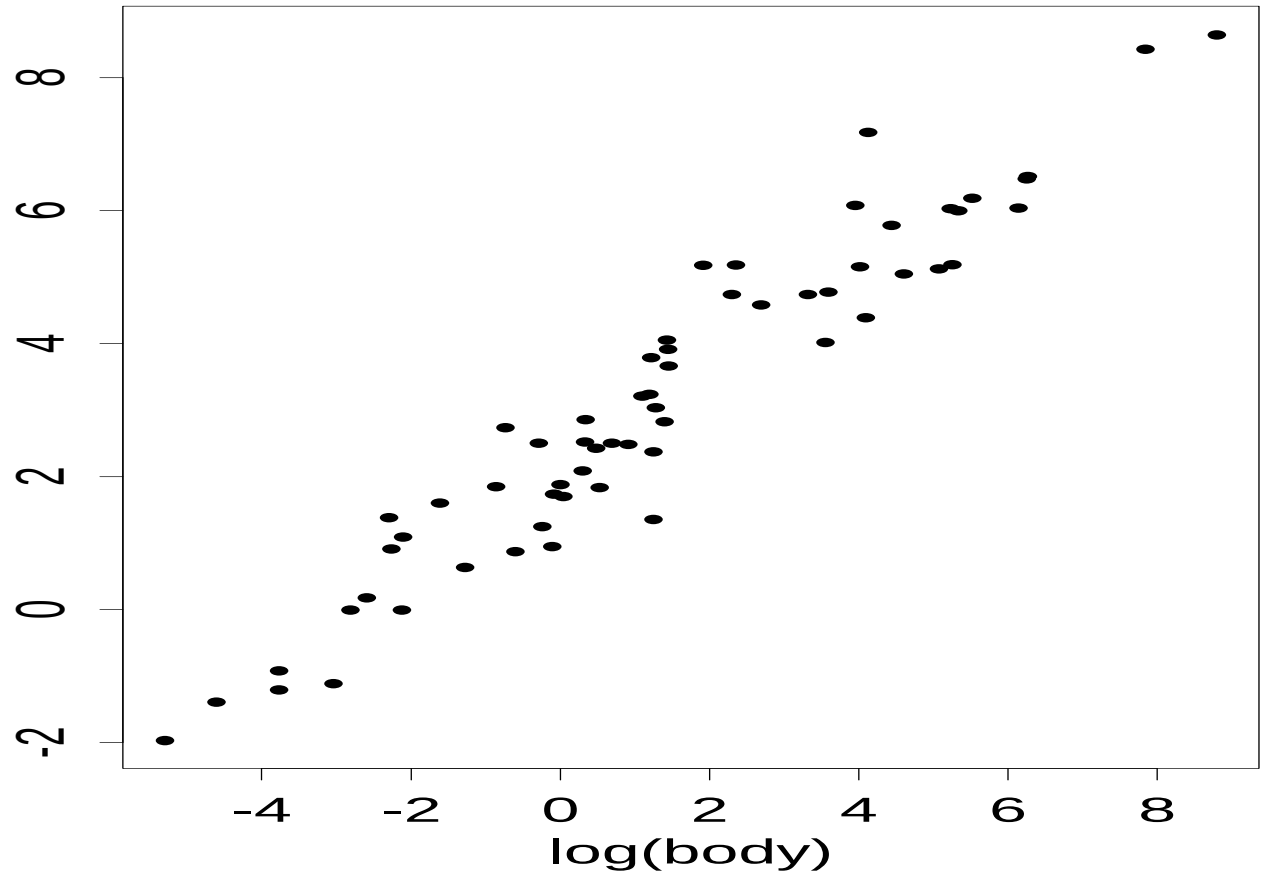

Figure 12: Plot of the log brain weight on the log body weight for several species of mammals.

Example 9: The model

Therefore, we have the following model:

$$\begin{aligned}
\text{lbrain}_i|\alpha, \beta, \text{lbody}_i, \tau &\sim N(\text{lbrain}_i|\alpha + \beta * \text{lbody}_i, \tau) \\
\alpha|\mu_\alpha, \tau_\alpha &\sim N(\alpha|\mu_\alpha, \tau_\alpha) \\
\beta|\mu_\beta, \tau_\beta &\sim N(\beta|\mu_\beta, \tau_\beta) \\
\tau|a_\tau, b_\tau &\sim \text{Gam}(a_\tau, b_\tau)
\end{aligned}$$

Example 9: The Priors

- If we want the prior distribution to be fairly weak then we would want the value τ_α, τ_β, a_τ, and b_τ to be relatively small.
- From our work with the normal/gamma model, we could see that if we let τ_α and τ_β be fairly small, then the data will dominate in the calculation of the posterior distribution of α and β. Also, the posterior distribution for τ will be dominated by the data if a_τ and b_τ are relatively small.
- So, for now, we set the value of τ_α and τ_β to be 0.0001 and also set a_τ and b_τ to also be 0.0001. Also, set the prior means of both α and β to be zero.

The BUGS Software Package

There are other software packages which seem to be coming on line. For example there are MCMC methods used in some procedure for SAS (for example for multiple imputation). However, BUGS and its variants such as WinBugs is the current state of the art.

- There are basically three flavours: Classic BUGS, WinBUGS, and OpenBugs. Classic BUGS is available for many platforms is is text based. Classic BUGS is no longer being developed. WinBUGS has a graphical user interface and is available on Windows machines. OpenBugs is an open source version of Bugs.

The BUGS Software Package

(Continued)

- Also, there is a GNU package which is similar called JAGS (i.e.: Just Another Gibbs Sampler) available at: ”http://www-fis.iarc.fr/ martyn/software/jags/”.
- Using a scripting language, one can often call WinBugs or OpenBugs from other packages. There are preprogrammed language add-ons to packages such as R, Matlab, and SAS which simplify the ability of these packages to call a version of Bugs. (See the WinBugs website for some of these add-ons.)

The BUGS Software Package

(Continued)

- A very nice feature with BUGS is that one simply needs to specify the model, the data, and the starting values and BUGS will create the sampler. There is no need to specify the conditional distributions. This makes BUGS extremely easy to use and very flexible.
- Because one simply needs to specify the model and not derive the conditional distributions, Bugs has set a standard for Bayesian computing. Any competing Bayesian software package will need to have this ability.

BUGS Software: Implementation

These are the steps needed in order to analyze a model with OpenBUGS.

- *Load the model into OpenBUGS*. One needs to use OpenBUGS syntax to specify one's model. Then, one needs to load in the data into OpenBUGS and compile the model and the data. OpenBUGS compiles the model and the data in order to generate the sampler algorithm.
- *Initializes the simulation*. This is the point where either the user or OpenBUGS selects the starting values to the chain (or to several parallel chains).

BUGS Software: Implementation

(Continued)

- *Running the simulation*. At this point, several parameters selected to be monitored. The chain is then run for several iterations. There are options to observe the points as they are being sampled as an aid in deciding when to stop the sampler. Also, there are options to estimate different values of the posterior distribution or to output the sampled values so that they can be processed by other programs.

Loading Basic Model in OpenBUGS

The following is the syntax for the linear regression model to compare brain weight to body weight.

```
model{
for(i in 1:N){
lgbrain[i]<-log(brain[i])
lgbody[i]<-log(body[i])
lgbrain[i]~dnorm(mu[i],tau)
mu[i]<-alpha +beta*lgbody[i]
}
alpha~dnorm(0,.0001)
beta~dnorm(0,.0001)
tau~dgamma(.0001,.0001)
}
```

Loading Basic Model in OpenBUGS

An explanation of the BUGS syntax for the model:

- The first part of the above code is the `model` statement. This tells OpenBUGS that the expressions enclosed in the open and closed 's contain the specification of the model.
- The code then loops over the different values of i. This loop is specified by the command: `for(i in 1:N){...}`. Everything in the brackets is then done for the different values of i.
- The next two lines are not strictly necessary. Since the data file that I use in this program contains the raw weights for the brain and body, I need to transform these values to logs before proceeding.

Loading Basic Model in OpenBUGS

(continued)

- The next line is where the probability model is specified. The code"`lgbrain[i]~dnorm(mu[i],tau)`" tells WinBUGS that the values `lgbrain[i]` are from a normal distribution with mean `mu[i]` and precision `tau`.
- The next line defines the value of `mu[i]` as the linear function. The next three lines give the priors for the parameters `alpha`, `beta`, and `tau`. The first two are given a normal prior and `tau` has a gamma prior.
- There are a number of other standard probability distributions which one can use in OpenBUGS/BUGS including the beta, poisson, bernoulli, binomial, multinomial, and Dirichlet to name some. Please see the user manual for more details.

Loading Basic Model in OpenBUGS

Inputting the model into OpenBUGS.

- To load the model into OpenBUGS, one first needs to click on the tab that says, “Model”.
- Then, on the drop down list of items, the first item is called “Specification...”, which one needs to click on. This will open a small window with the title “Specification Tool”.
- First one clicks on the word “model” in the user file. This will highlight the word “model”. With model highlighted, then one clicks on the button “check model” in the ‘Specification Tool” window.

Loading Basic Model in OpenBUGS

(continued)

- OpenBUGS then processes the model. If there are no problems, then in the bottom left hand corner of the OpenBUGS screen, OpenBUGS reports, “model is syntactically correct”. If there are some problems then WinBUGS gives a message which is usually somewhat helpful in pointing out what syntax error it found.

Loading Basic Model in OpenBUGS

Inputting the data:

- Besides inputting the model, one needs to also specify the data and also to initialize values for the Markov Chain Monte Carlo method.
- The data can be specified via an Splus list statement or as columns of values. When specifying columns of numbers, one needs to only include the columns which are used in the analysis.
- So, if one has a text file with four variable, each in there own column and one is only using the first two columns in the analysis, then one needs to remove the columns which are not being used. (Alternatively, model each of the unknown variables with unknown mean and variance and put a prior on these unknown means and variances.)

Loading Basic Model in OpenBUGS

Inputting the data (continuing)

The data for this problem was input in two steps.

- In the first step, the constant N is first input then the two columns corresponding to the brain and body weights are input.
- The constant N is inputted with the following command:

  ```
  list(N=62)
  ```

  On in the bottom left hand corner of the OpenBUGS window, OpenBUGS gives the message “data loaded”.

Loading Basic Model in OpenBUGS

Inputting the data (continuing)

- The data for the two variables are listed in the following format (with only the first few observations reproduced here):

body[]	brain[]
3.385	44.5
.480	15.5
1.35	8.1
465	423
36.33	119.5
27.66	115
14.83	98.2
1.04	5.5
⋮	⋮

Loading Basic Model in OpenBUGS

Inputting the data (continuing)

- The mouse is used to highlight the first line (which contains the column heading). Then, the “load data” button on the Specification Tool window is then clicked. Again, if everything is correct, then OpenBUGS gives the message: “data loaded”.
- Once this is done, then the “compile” button is clicked in the Specification Tool window. If everything is correct, then OpenBUGS gives the message “model compiled”. At this point, OpenBUGS has created the necessary code to produce the Markov chain Monte Carlo simulation. All that is now needed is to initialize the nodes which are going to be sampled and then carry out the simulation.

Initializing the Simulation

After the model is compiled, the initial values for the simulation are selected.

- Either the initial points can be specified by the user or the computer can generate initial values from the the prior distribution.
- For user specified values, the user would assign values for parameters in the same way as the data values were assigned.

Initializing the Simulation

(Continued)

- So, for example, the parameters `alpha`, `beta`, and `tau` can be started off to have the value 1. To do this, first there is the following line in the file:

  ```
  list(alpha=1,beta=1,tau=1)
  ```

- The word “list” is highlighted by clicking on it.
- Then, in the Specification Tool window, the button labeled “load inits” is clicked on. If everything is correct, then OpenBUGS provides the message, “initial values loaded: model initialized”.

Initializing the Simulation

(Continued)

- Alternatively, instead of having user specified initial values, the computer can generate a set of initial values.
- To do this, click on the button labeled “gen inits” in the Specification Tool window. The computer then uses the prior distribution to generate the initial values of the parameter.

Initializing the Simulation

(Continued)

- It is somewhat tempting to always have the computer specifying the initial values. However, there are some advantages of specifying the initial values manually.
- First, if you specify all the parameters for the model and OpenBUGS does not report that the model is completely initialized, then you don't fully understand your model.
- Also, to see how fast the chain takes to move to the area of high probability under the posterior distribution, it might be desirable to occasionally start the chain off far from the posterior distribution. Then, one can see the values move to the high probability area of the posterior distribution. A chain which mixes fast will move quickly to the area of high probability.

Initializing the Simulation

It is also possible to run several chains simultaneously.

- If there are some "pockets" of high probability or if the chain takes a while to mix over the posterior distribution, this would be easier to see if the chain was started from several different places.
- In a chain which mixes slowly, then one would possible expect to see the different chains take a while before they started to cross paths.
- To have several chains going at once, the above procedure is slightly modified. In the Specification Tool window, after clicking on the "check model" button and before clicking on the "compile" button, click on the "num of chains" option and change the 1 to the number of chains that is desired. Then, one needs to initialize the starting value of each chain.

Running the Simulation

With the model compiled and the initial values set, the simulation chain can be run and the sampled values can be stored in order to make inferences.

- This is done by opening up two more specialized OpenBUGS windows.
- First click on the word “Model” in the top toolbar. Under Model, click on the subcommand “Update...”. This opens up the Update Tool window. This window is used to tell OpenBUGS to generate samples from the Markov chain.
- Also, in the top toolbar, click on the item “Inference” and then under the subcommand “Samples...”. This opens up the Sample Monitor Tool window. This window directs OpenBUGS to store sampled values of different parameters as the Markov chain is run.

Running the Simulation

(continued)

- For example, consider making inferences on the parameters `alpha`, `beta`, and `tau`.
- To monitor these parameters, each of these parameter names are typed into the node space on the Sample Monitor Tool window. After the name is typed in, then the “set” button becomes active. When this button is clicked on, then OpenBUGS starts to monitor this parameter.
- After all the desired parameters are set up, then a “*” is typed into the node window. When this is done, then all the other buttons in the Sample Monitor Tool window are then activated.

Running the Simulation

(continued)

- Clicking on the “trace” button gives a running sample of the different parameters that are being monitored. As the chain is being updated, the values of the parameter should randomly span the same range over time.

Running the Simulation

Running the sampler:

- To run the chain, go to the Update Tool window and click on the update button. This starts the chain.
- To run more samples, keep clicking on the update window.
- To run more samples per click, change the sampling frequency in the Update Tool window.

Running the Simulation

How long to run the sampler? (Preliminary thoughts)

- The number of updates that need to be done depend on two things,
  1. How long will it take before the samples from the MCMC are approximately from the posterior distribution? and
  2. How many estimates are needed before one can accurately estimate the properties of the posterior distribution that are of interest?
- The trace plot gives some preliminary evidence that the chain has basically settled in the area of high posterior probability when the trace plots stop “drifting”.

Running the Simulation

Some other monitoring tools in OpenBUGS.

- The “history” button on the Sample Monitor Tool window gives the entire trace plot for the entire history of the different parameters which are monitored.
- The “autoC” button gives a plot of the autocorrelation function of the different parameters. It is desirable for the autocorrelation to quickly fall to zero. This would give some evidence that the sample chain is quickly mixing.

Running the Simulation

(More on monitoring tools)

- If multiple chains are run, then the “GR diag” button in the Sample Monitor Tool window will instruct WinBUGS to perform the Gelman-Rubin diagnostic test. More details on this diagnostics procedure and other diagnostic procedures are discussed below.
- The buttons “stats” and “density” perform basic inferences for the parameters being monitored. The stat button produces several basic quantities of the posterior distribution such as the posterior mean and 95% posterior credible regions. The density button produces an estimate of the posterior density for each of the monitored parameters.

Running the Simulation

(More on monitoring tools)

- The “coda” button is used to produce files which are used to output the sampled values to other programs so that more detailed analyses can be performed on the sample sequences. It is called coda because the first such program to do such an analysis was called CODA and is a collect of functions which are run in R. The package BOA is a second generation of CODA and is also a collect of function which can be run in either Splus or R. (There are separate packages for Splus and R.)
- Besides options available on the Sample Monitor Tool window, the Correlation Tool window is useful to get the cross-correlation between the parameters. This window is open from the “Inference” command on the top toolbar.

Looking at the Data

- Figures 13 and 14 contains the trace plot for the parameter `alpha`.
- From these plots it would appear that the chain is quickly sampling near the posterior distribution. Of course, to know this more, it would be helpful to see the joint distribution.
- Figures 15 and 16 shows the cross-correlation of `alpha` and `beta`. There does seem to be some correlation between these two parameters, but it does not look too bad.

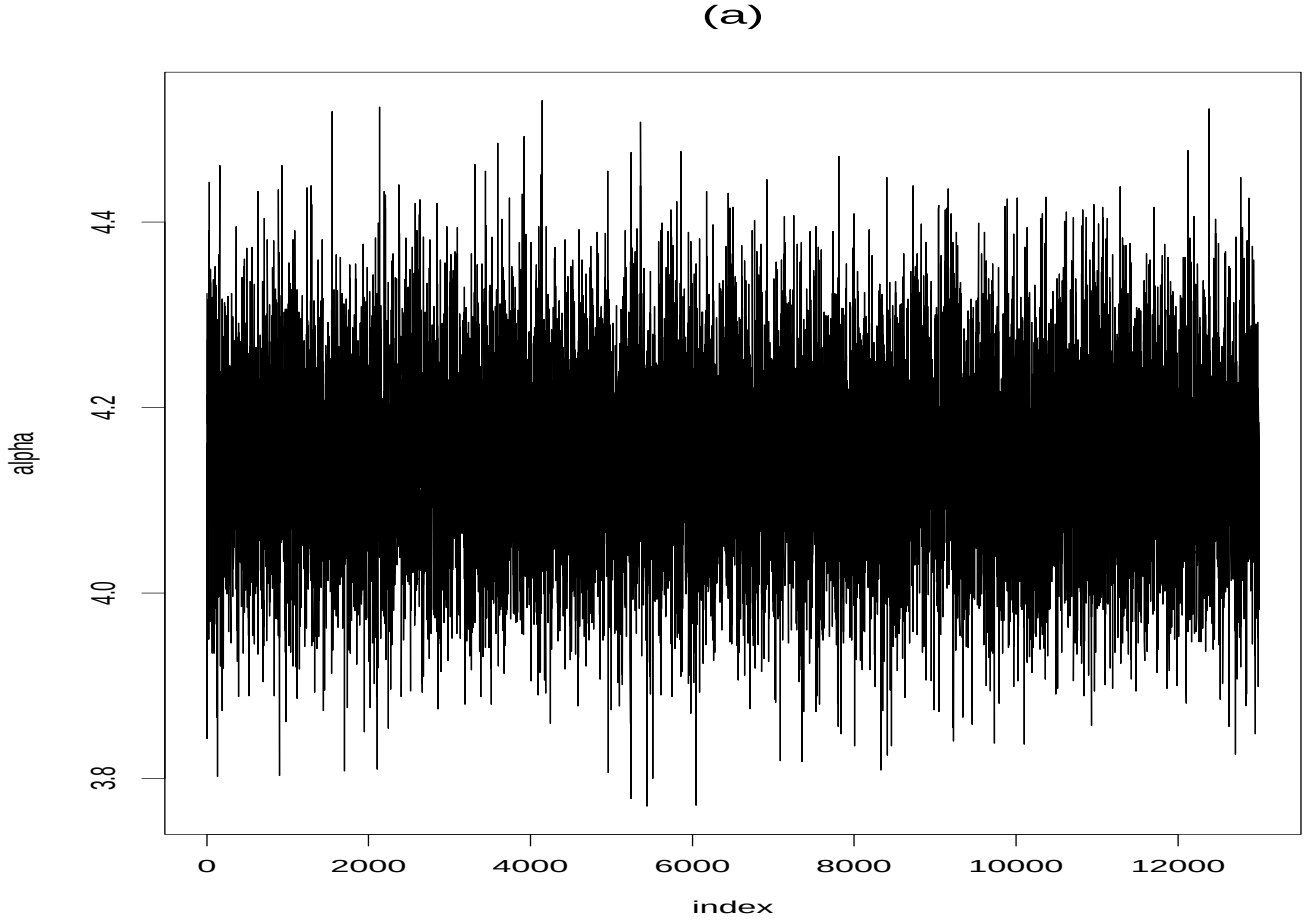

Figure 13: Trace plot for the parameter alpha for 14000 iterations.

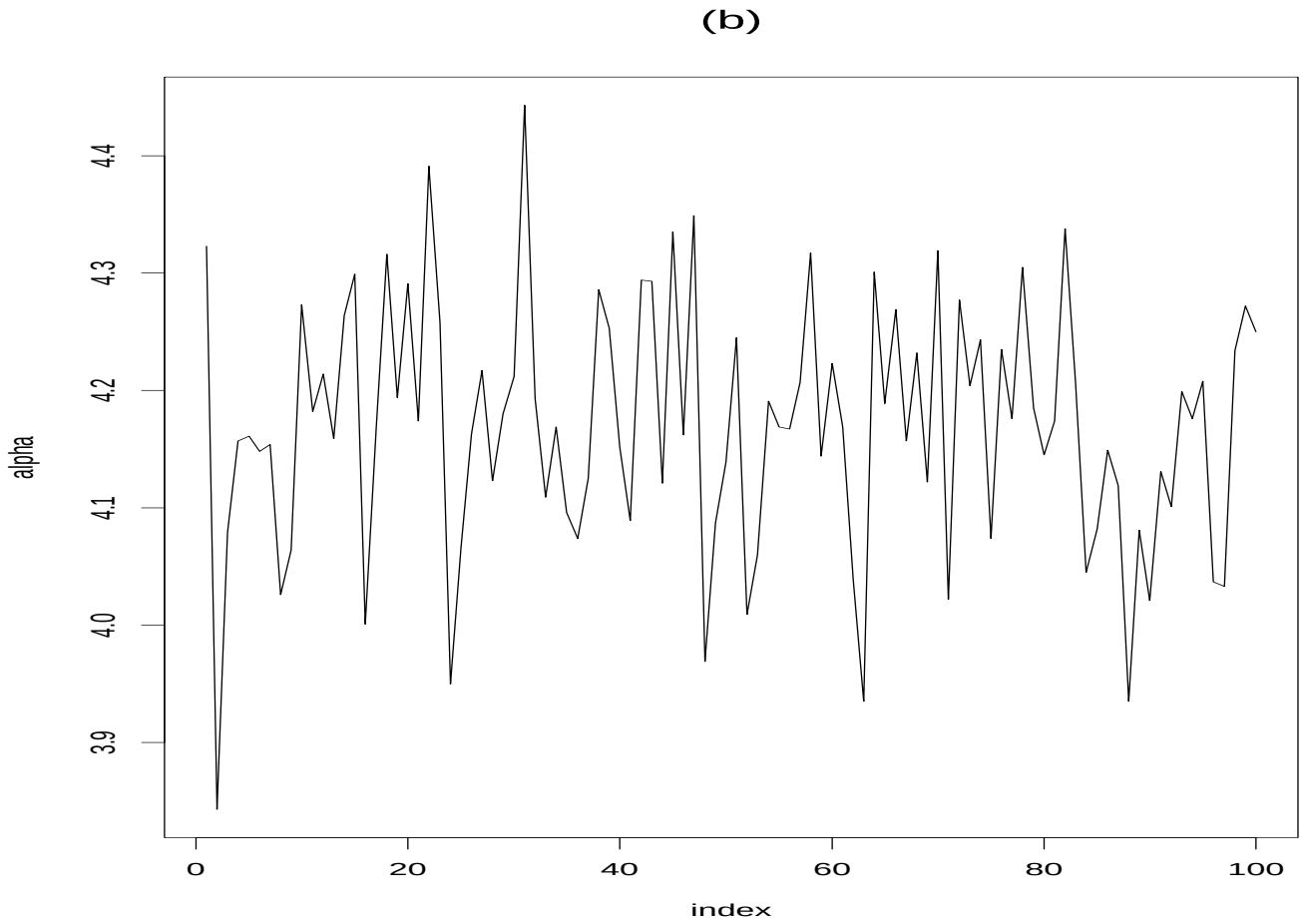

Figure 14: Trace plot for the parameter alpha for 100.

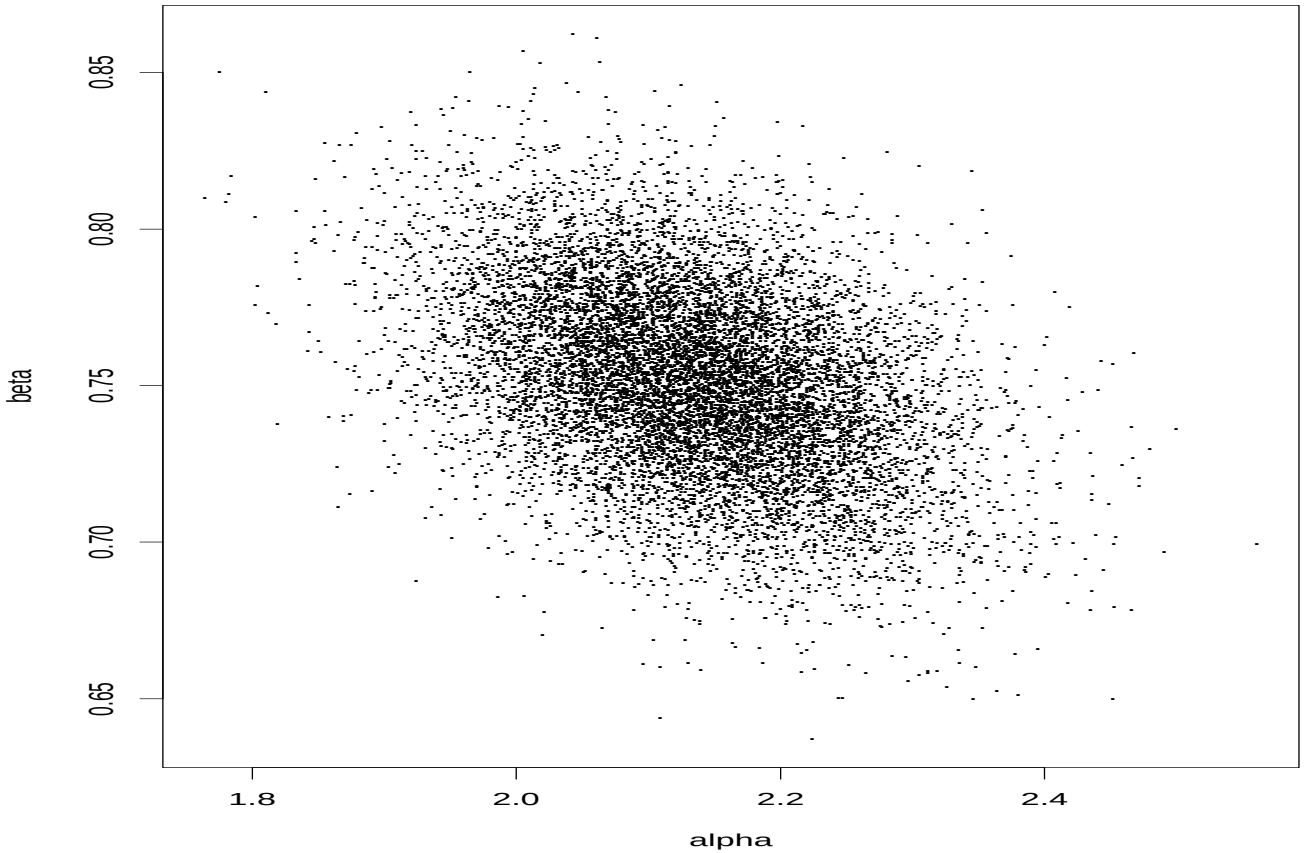

Figure 15: Point plot showing the correlation of alpha and beta.

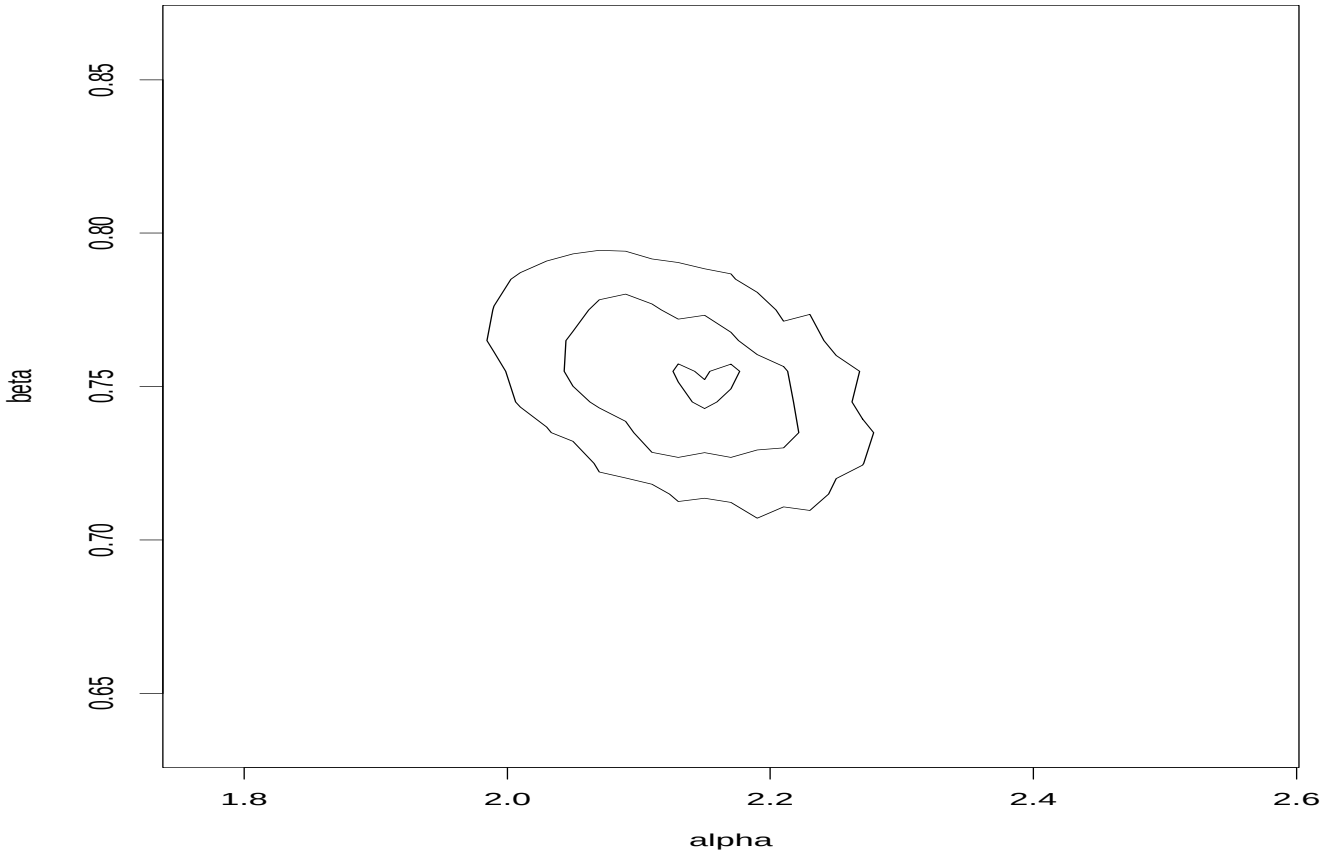

Figure 16: Contour plot showing the correlation of alpha and beta.

In earlier versions, the algorithm was not as effecient. So, there was more correlation between α and β.

- Please note that the contour plot looks a little rough. If more data points are sampled, then this contour plot would probably be smoother.
- Still there is way to greatly decrease the correlation between these two parameters. This is shown next. **OpenBug code has been modified so that these methods are not really needed anymore.**

Reducing the Correlation

From basic linear regression theory (like the kind that is taught in a frequentist course), the correlation of the parameters α and β can be reduced by reparameterizing the model. That is, for the probability model:

$$\begin{aligned} Y_i &= \alpha + \beta * X_i + \epsilon_i \\ &= \alpha^* + \beta * (X_i - \bar{X}) + \epsilon_i, \end{aligned}$$

where $\bar{X}$ is the average of the X_i's and $\alpha^* = \alpha + \beta * \bar{X}$.

The new model statement for OpenBUGS is now:

```
model{
lgbody.bar<-mean(lgbody[])
for(i in 1:N){
lgbrain[i]<-log(brain[i])
lgbody[i]<-log(body[i])

lgbrain[i]~dnorm(mu[i],tau)
mu[i]<-nalpha +beta*(lgbody[i]-lgbody.bar)
}
nalpha~dnorm(0,.0001)
beta~dnorm(0,.0001)
tau~dgamma(.0001,.0001)
alpha<-nalpha + beta*lgbody.bar
}
```

- The new initialization is: `list(nalpha=1,beta=1,tau=1)`.

- Figure 17 and 18 contains the new trace plots for `nalpha` and `alpha` and figures 19-22 contains the plots of the correlation between `alpha` and `beta` and the correlation plot between `nalpha` and `beta`. From the second set of plots, it appears that there does not appear to be any correlation between `nalpha` and `beta`.

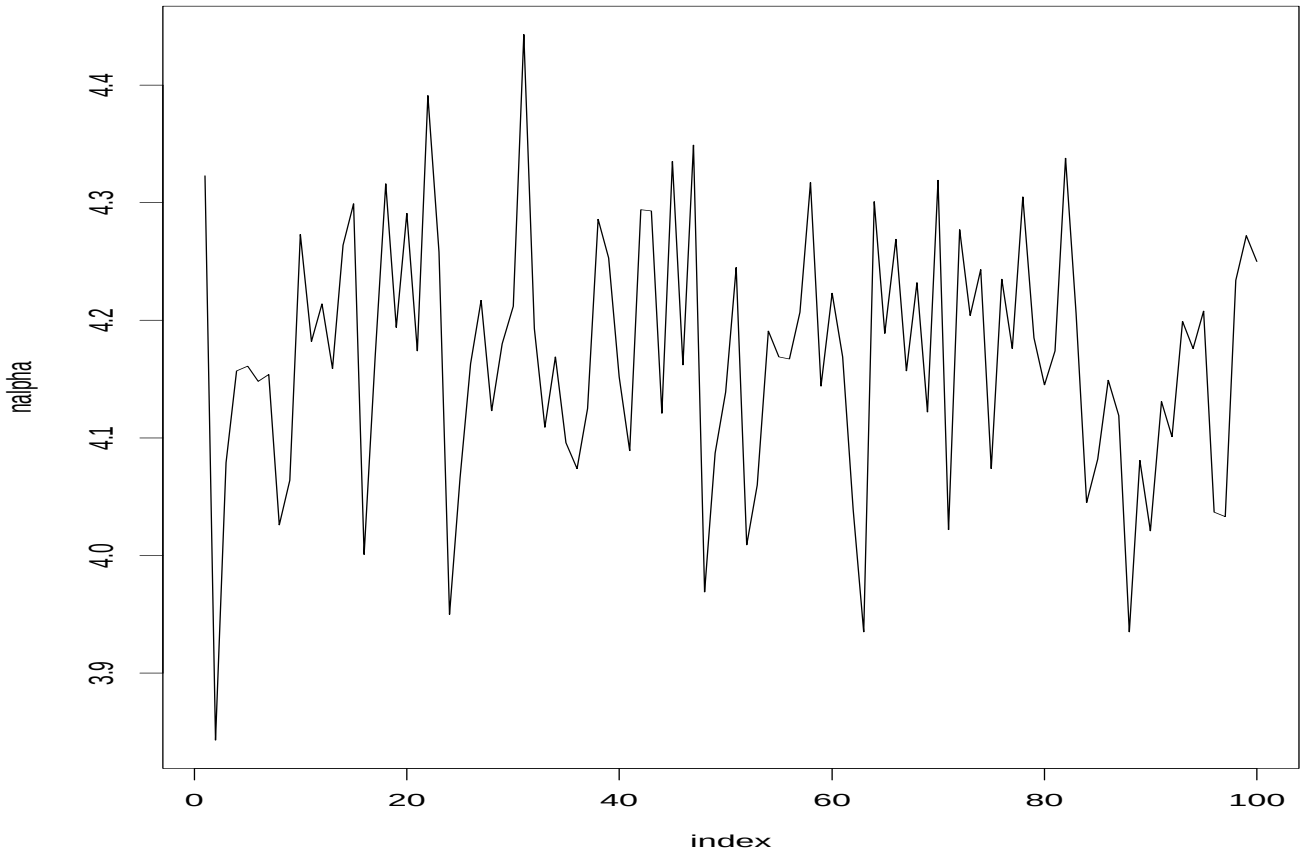

Figure 17: Trace plot for alpha. The noncentered parameter for α.

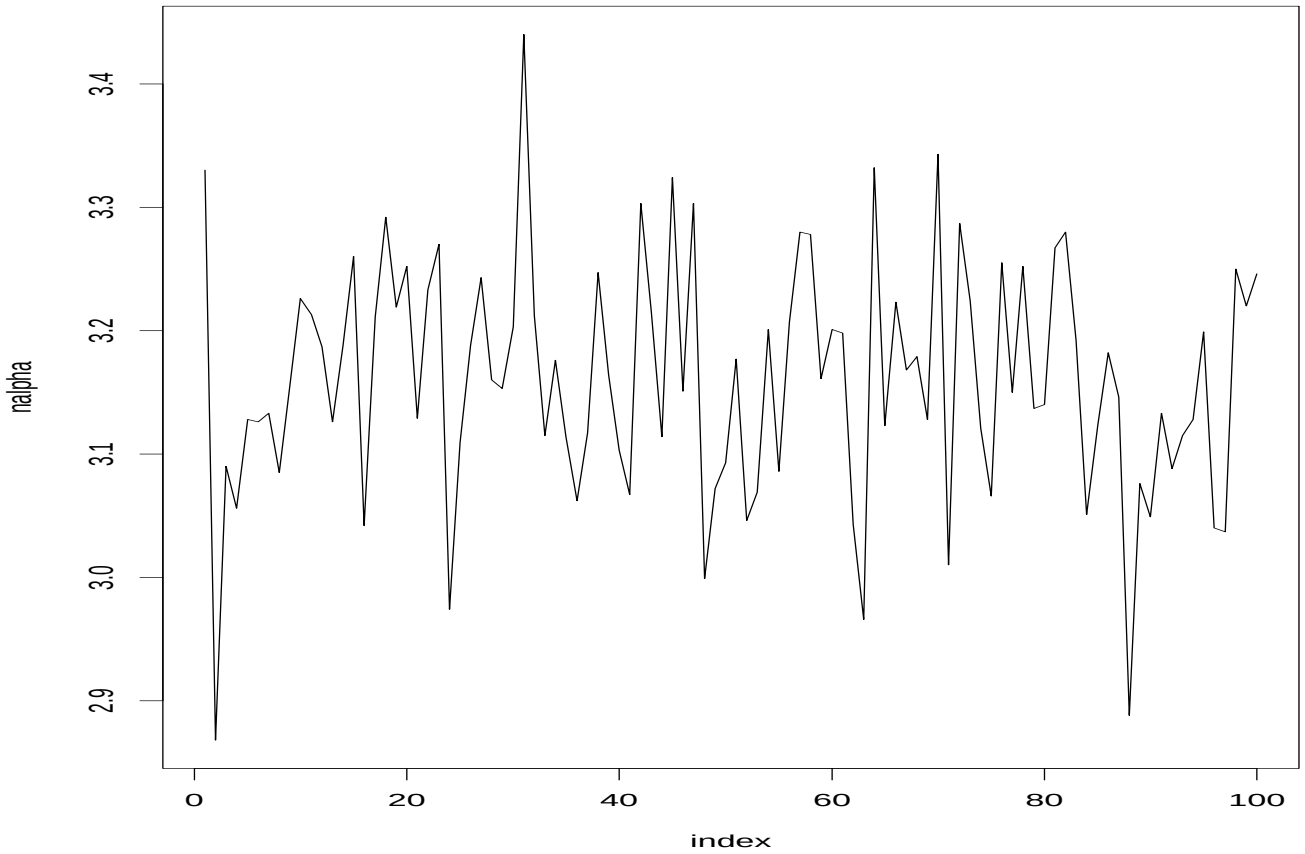

Figure 18: Trace plot for alpha. The centered parameter for α.

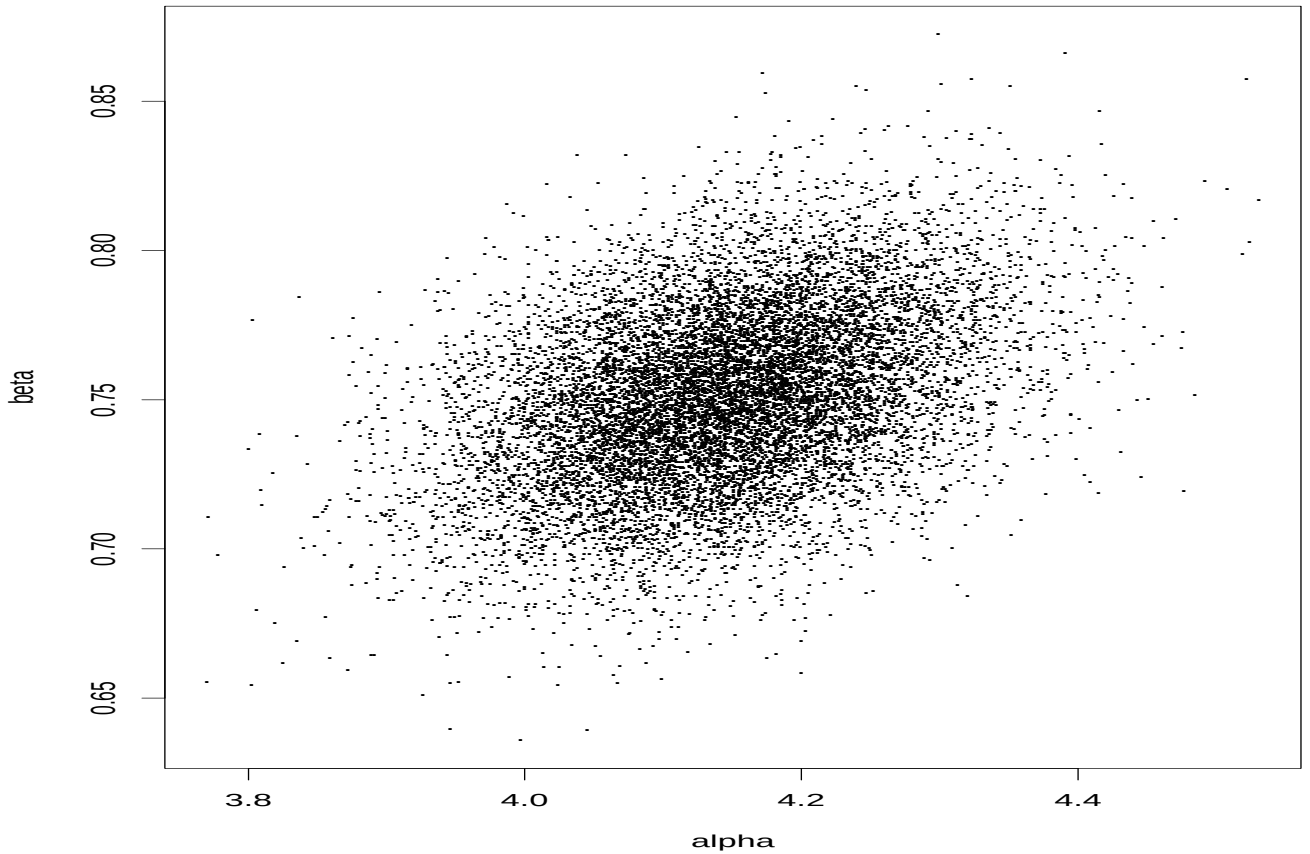

Figure 19: Correlation of alpha and beta. The point plot for the noncentered parameter for α

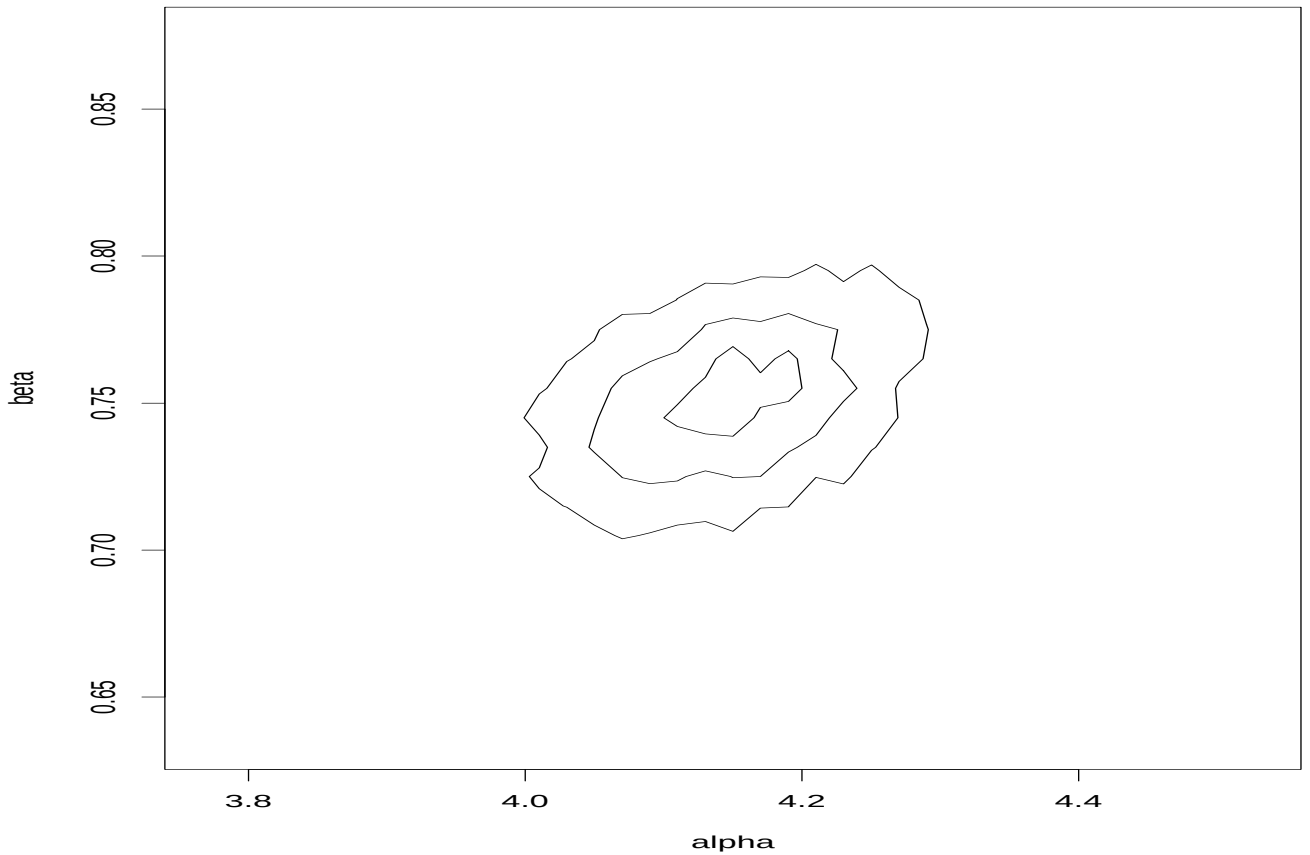

Figure 20: Correlation of alpha and beta. Contour plot for the noncentered parameter for α.

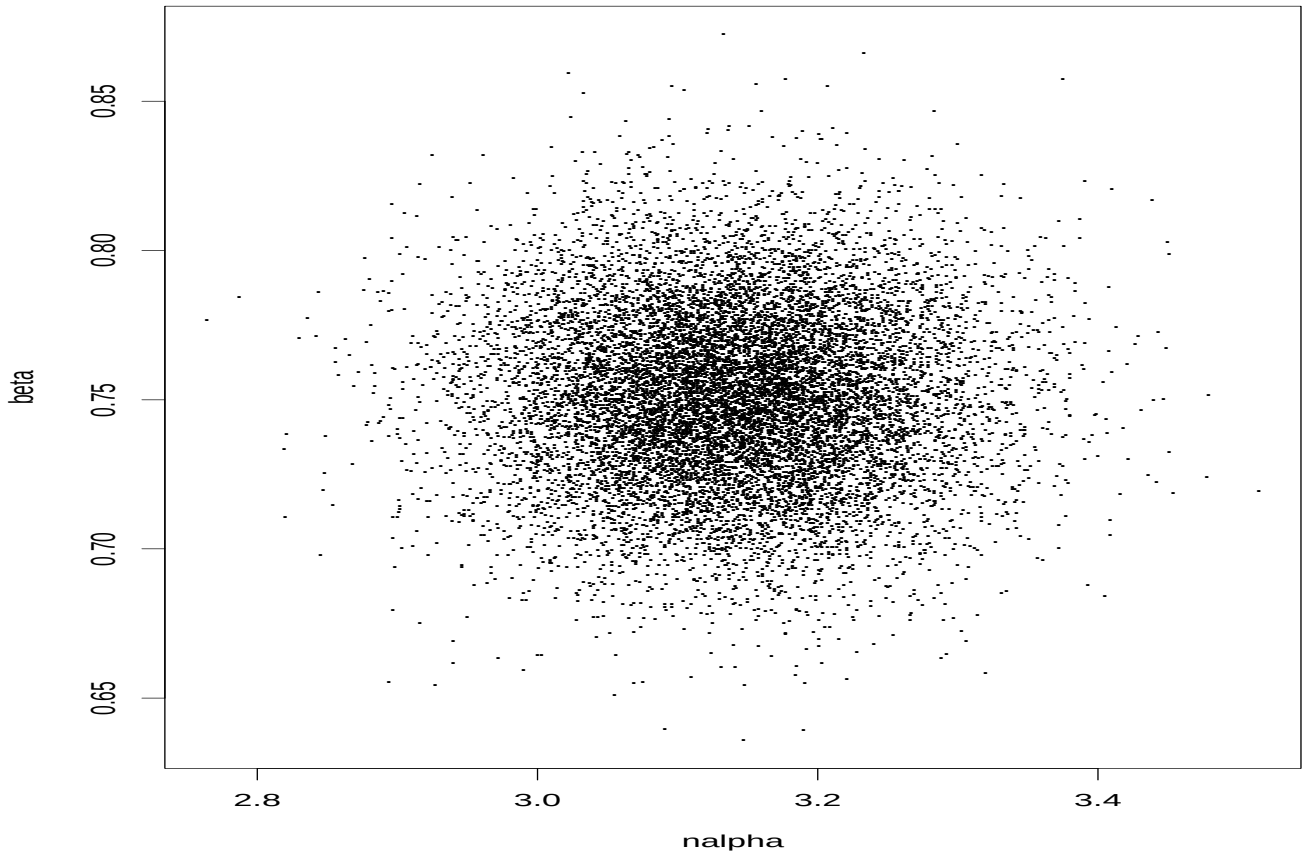

Figure 21: Correlation of alpha and beta. Point plot for the centered parameter for α.

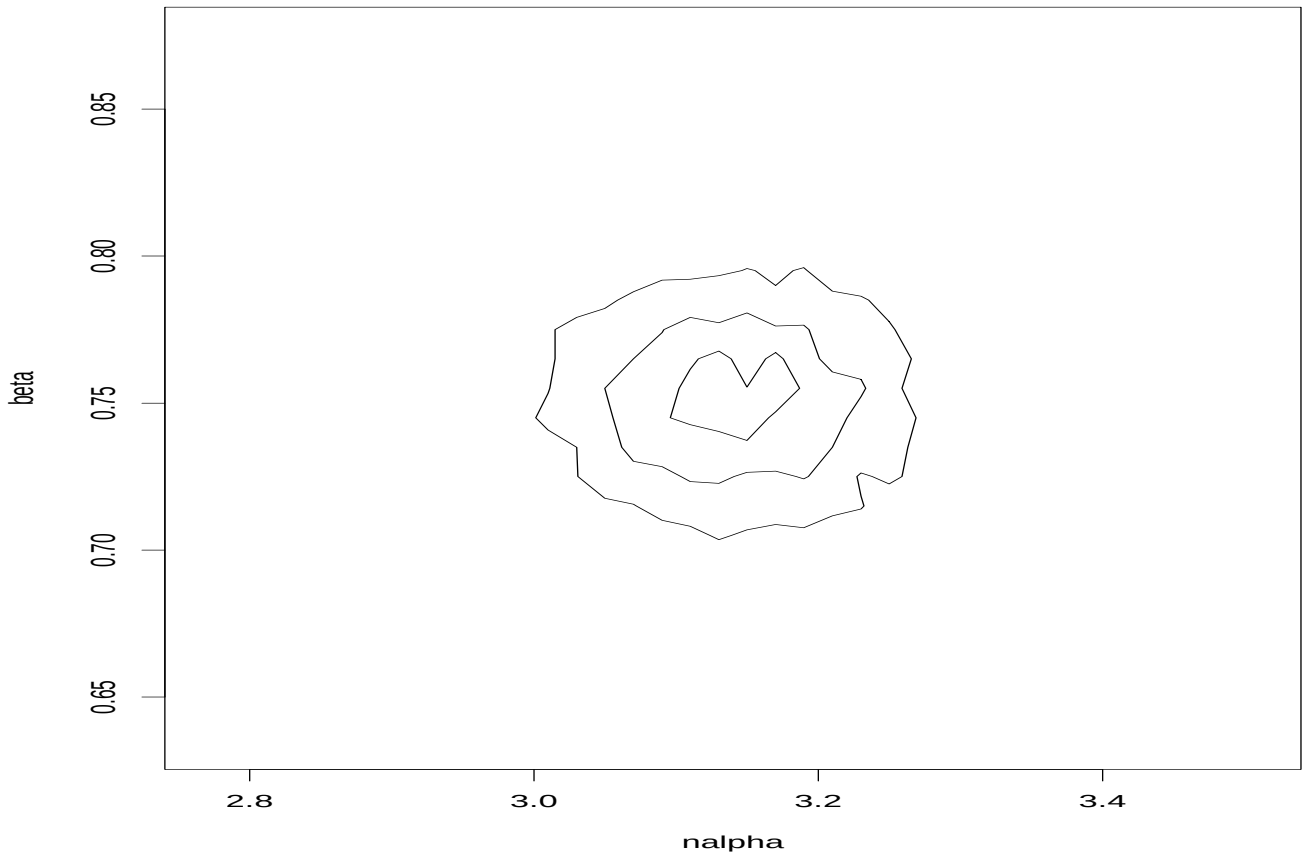

Figure 22: Correlation of alpha and beta. Contour plot for the centered parameter for α.

Comparing the Improvement

To see the improvement gained by using the centered data, see figures 17 and 22.

- These plots show the autocorrelation of `alpha` for the two models.
- Note that for the centered model, there is no autocorrelation for lags 1 and bigger. In this model, for the noncentered data, there is some autocorrelation for lag of 1 and possibly of lag 2.
- Still, this means that for both models, the chain must quickly move over the space of main probability mass.

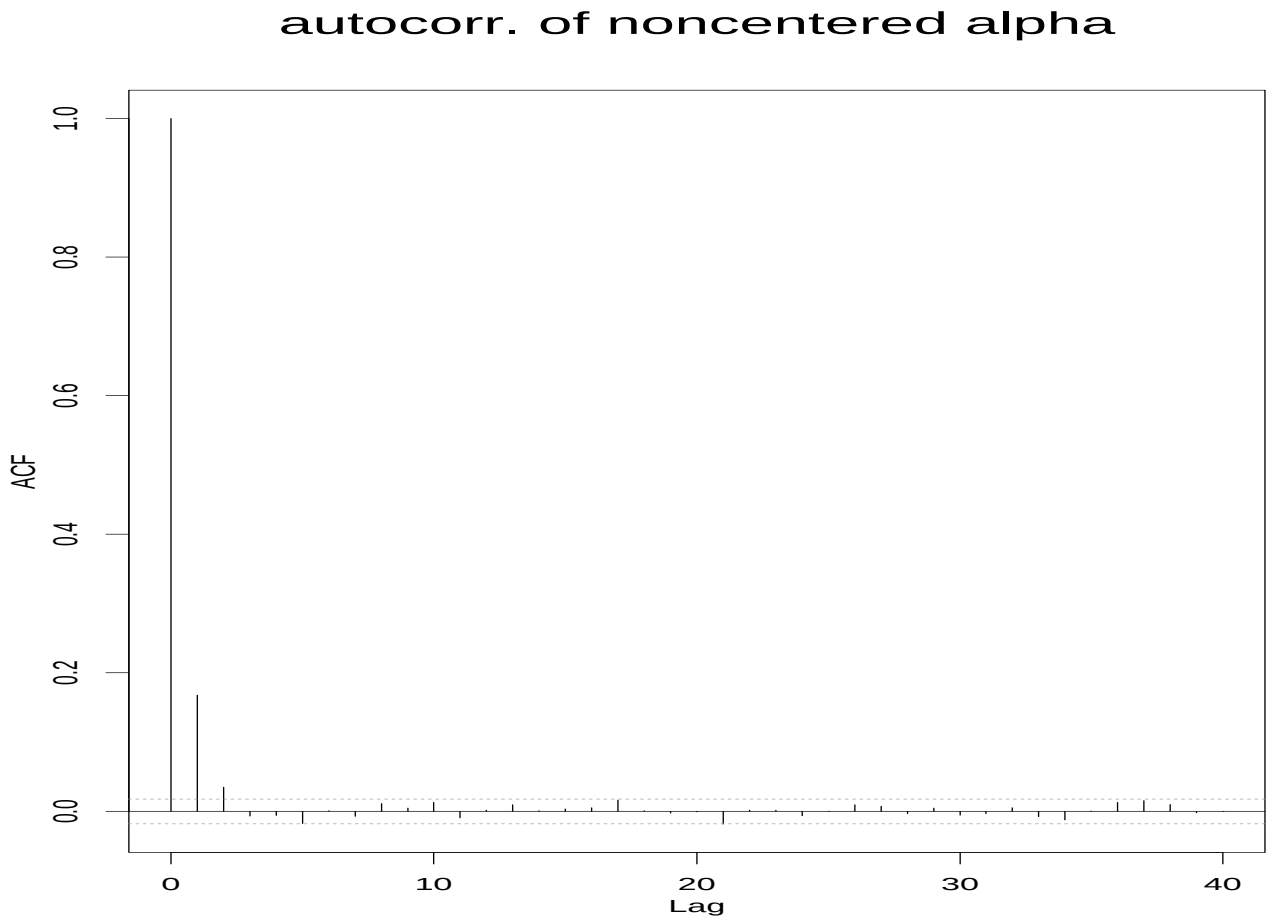

Figure 23: The autocorrelation for α at different lags. This plot is for the uncentered model.

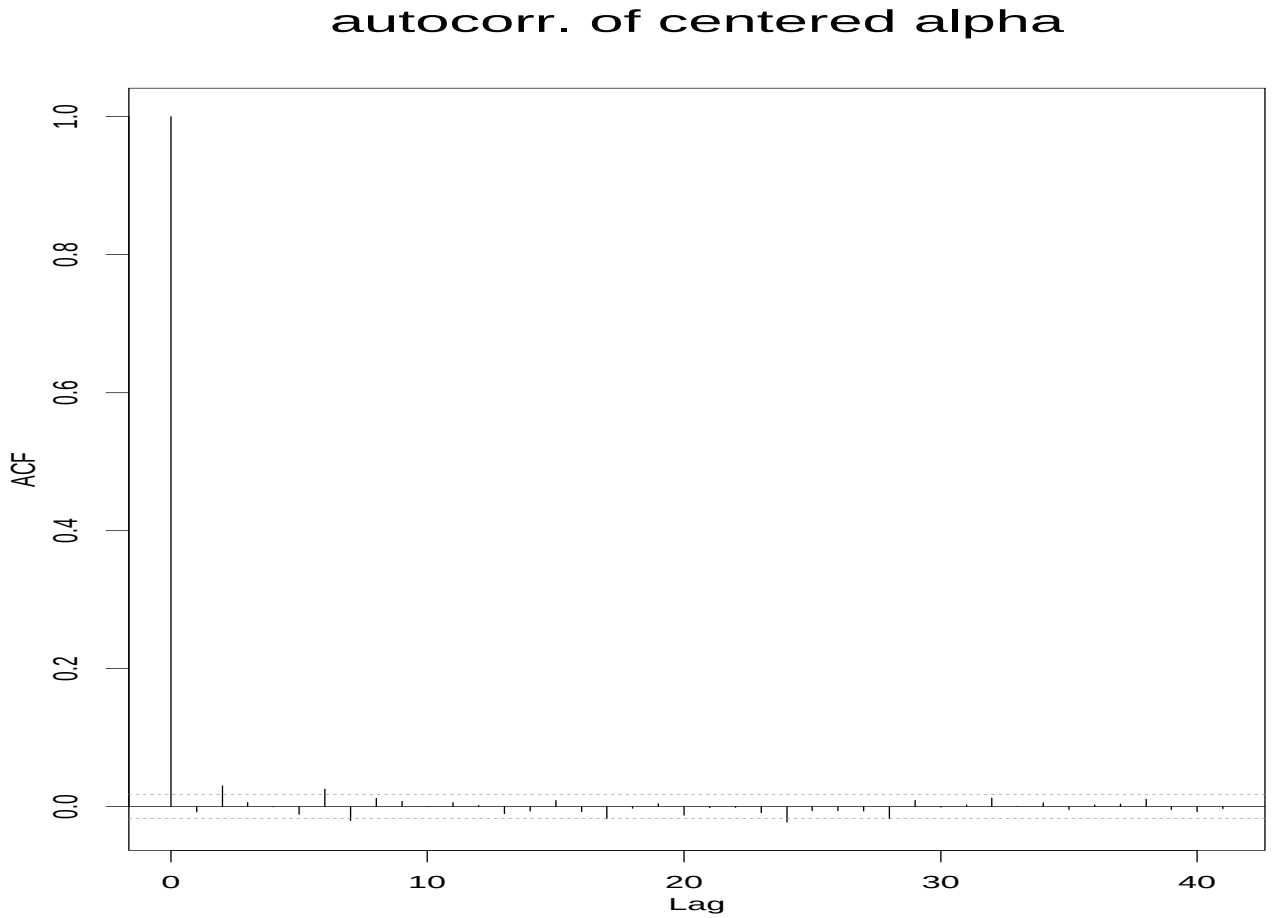

Figure 24: The autocorrelation for α at different lags. This plot is for the centered model.

- From this model, the marginal posterior densities can be estimated.
- Figures 25-28 contains these estimates.

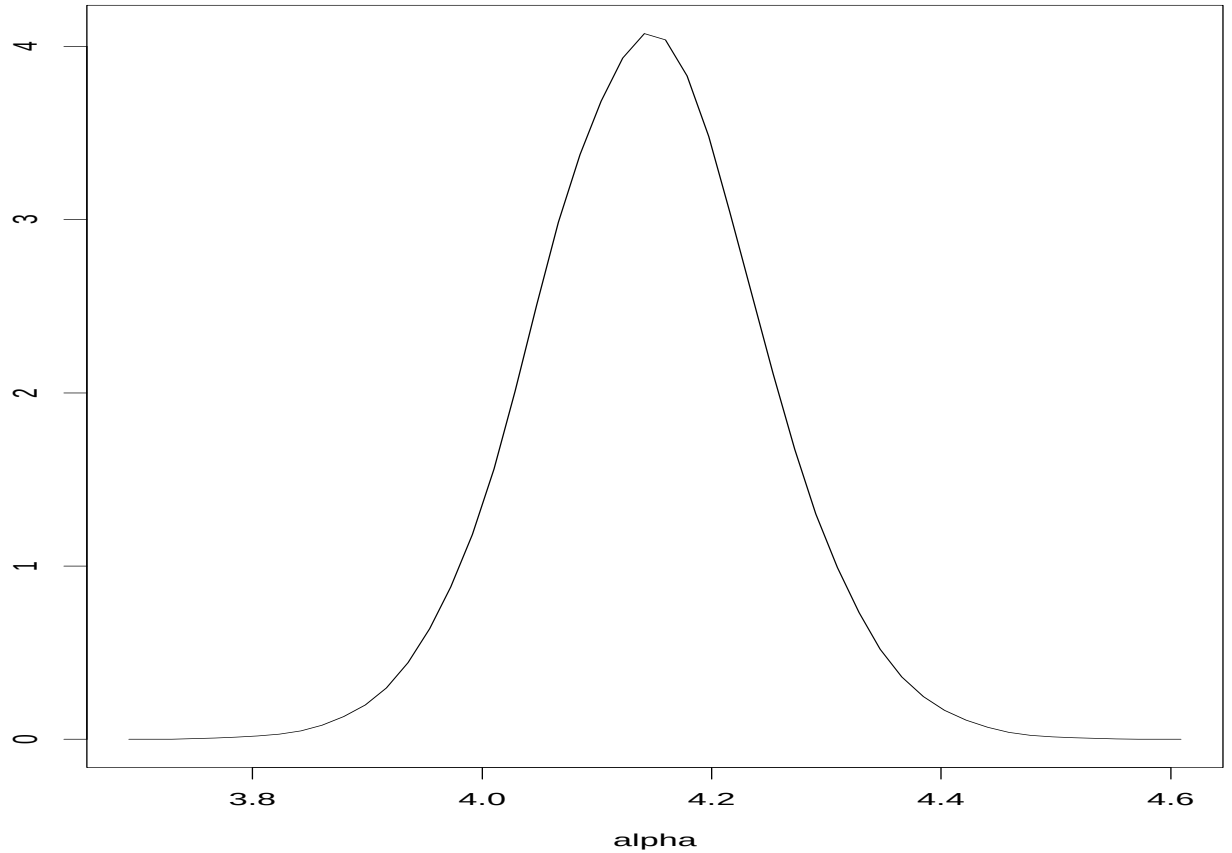

Figure 25: Posterior density estimates of alpha, nalpha, beta, and tau. These graphs are draw with the Splus kernel density estimator.

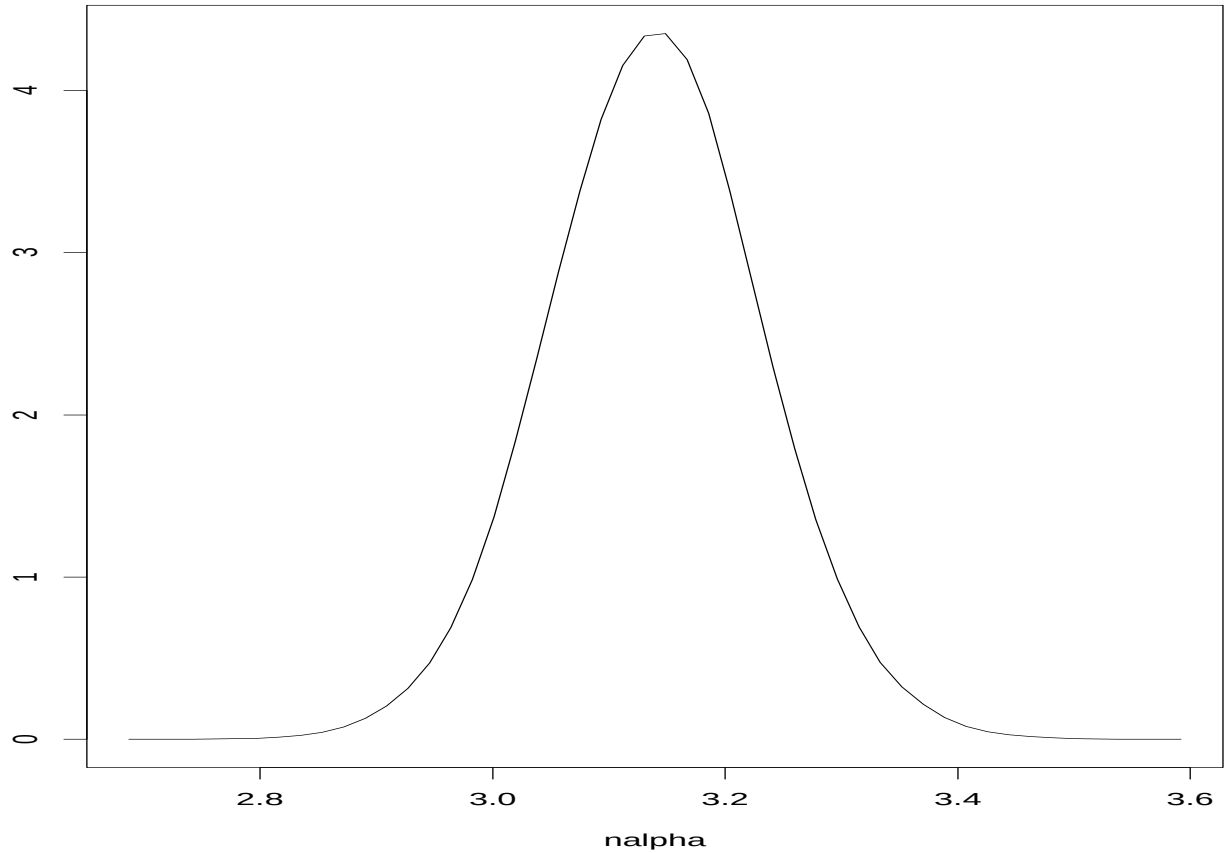

Figure 26: Posterior density estimates of alpha, nalpha, beta, and tau. These graphs are draw with the Splus kernel density estimator.

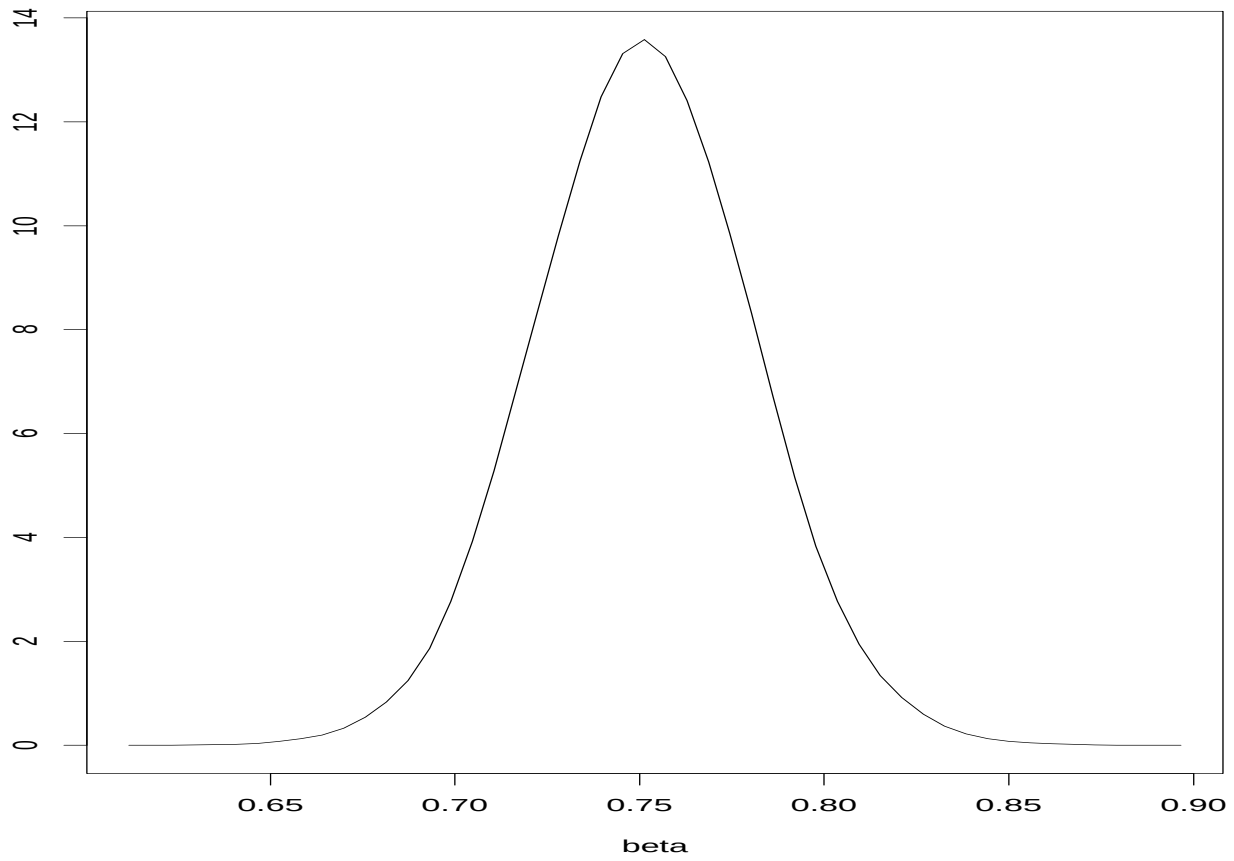

Figure 27: Posterior density estimates of alpha, nalpha, beta, and tau. These graphs are draw with the Splus kernel density estimator.

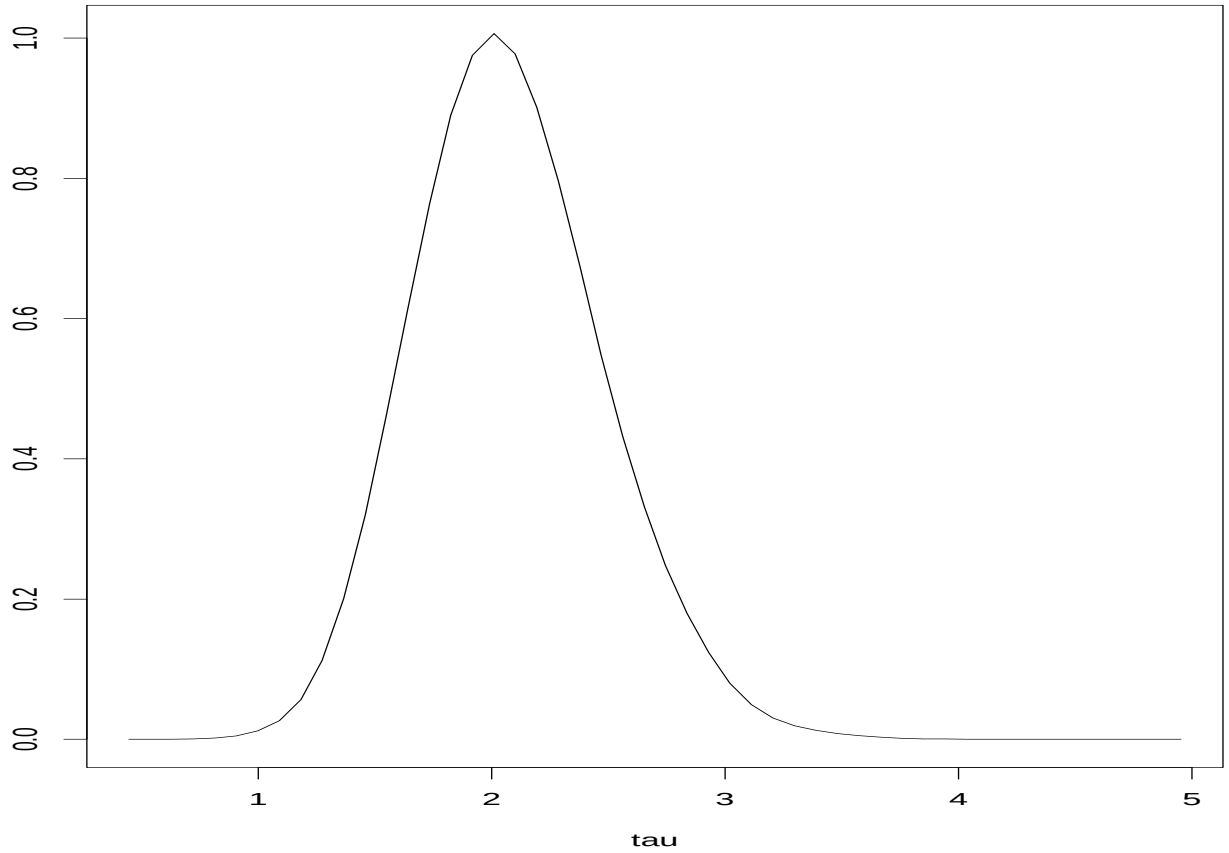

Figure 28: Posterior density estimates of alpha, nalpha, beta, and tau. These graphs are draw with the Splus kernel density estimator.

Summary of Normal Models

- This linear regression was a simply model in order to get use to these types of tools.
- MCMC methods allow one to break complex models into smaller submodels. This allows for very large models to be analysed.
- BUGS is a very power and flexible software package. It allows one to fit many different types of models with ease.
- Using the sampled data one can learn many features of the joint posterior distribution.

A look at a growth curve model

- Growth curve models or longitudinal models are models where several subjects are followed over time.
- Very common in pre-clinical studies with animal models. Also, used when following a group of people over time.
- These models can be complicated from a frequentist approach, but they are very easy to work with using Bayes and MCMC.
- At this point in the course, we quickly look at an example to illustrate how one can use Bayes and MCMC.

Growth curve models

For this example, let us consider the example that was used in Gelfand et al (1990, JASA) in their big paper which showed the ease of MCMC methods. (Aside, one of the groups is in the "examples" of Winbugs/OpenBugs.)

- In this example, there are two groups of rats: a control group and a treatment group.
- In each group there are 30 rats and they are measure every week for 5 weeks. Therefore, they are measured on days 8, 15, 22, 29, and 36.

Figures 29-31 show the growth curves.

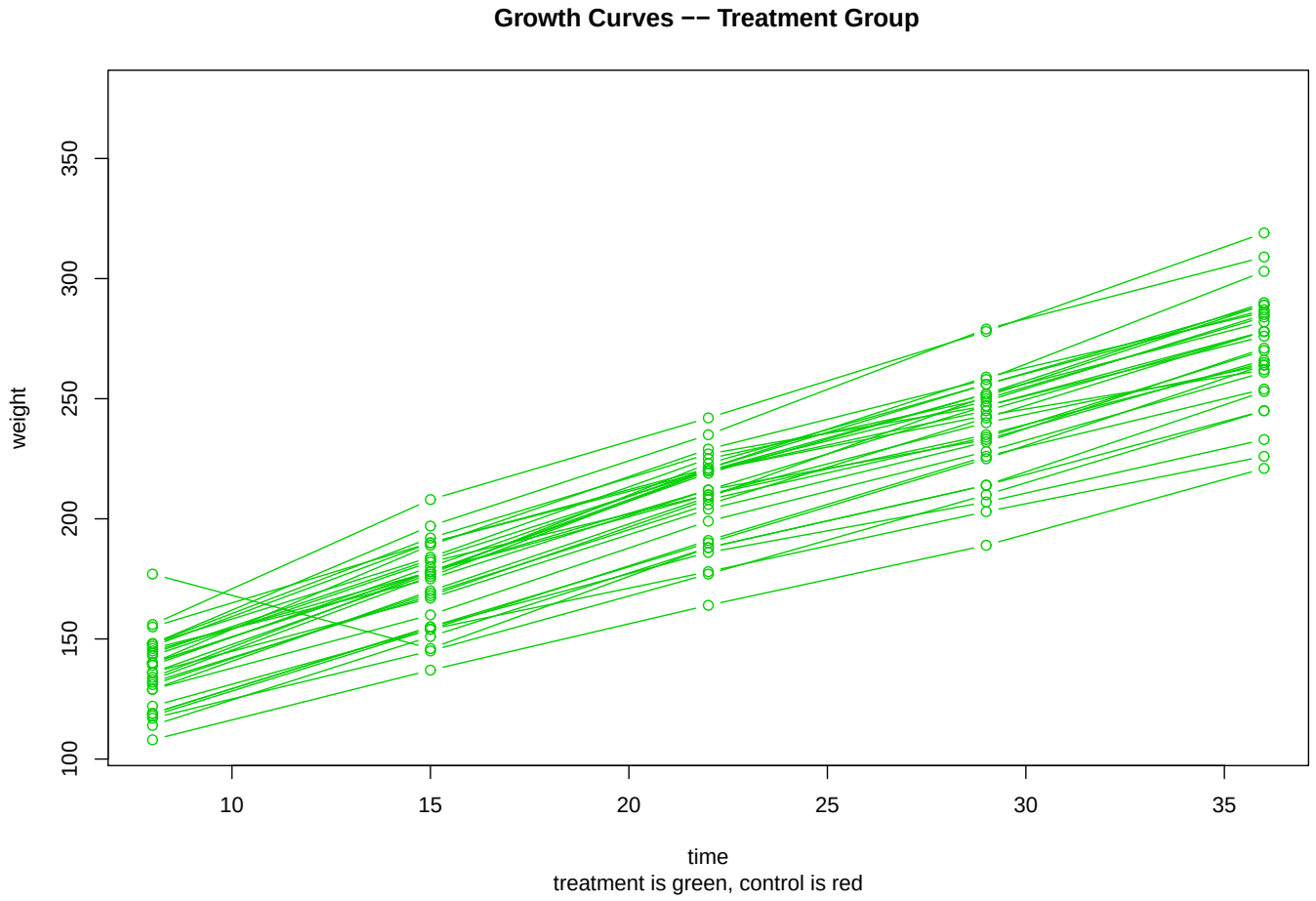

Figure 29: Growth curve for the treated group of rats.

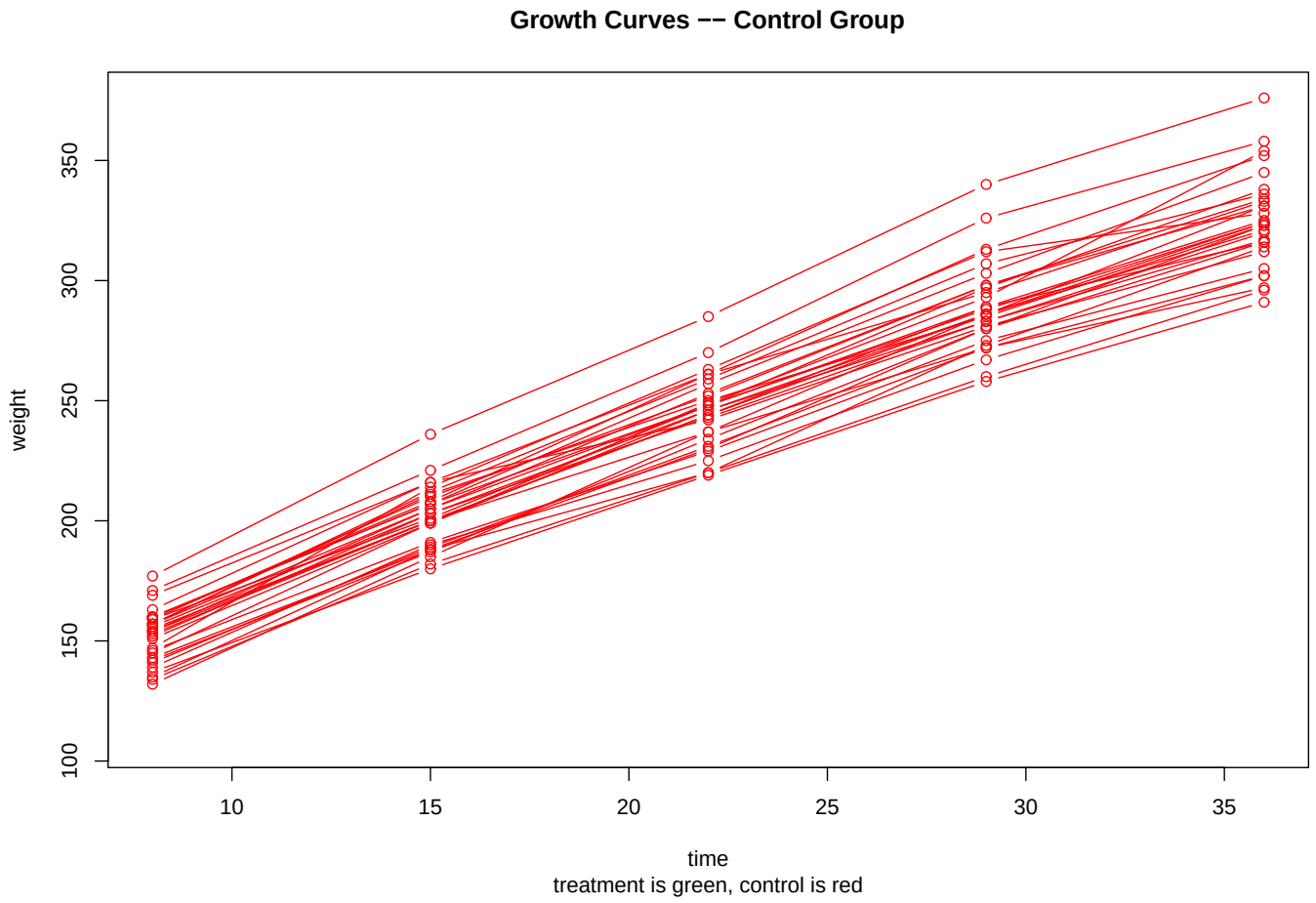

Figure 30: Growth curve for the control group of rats.

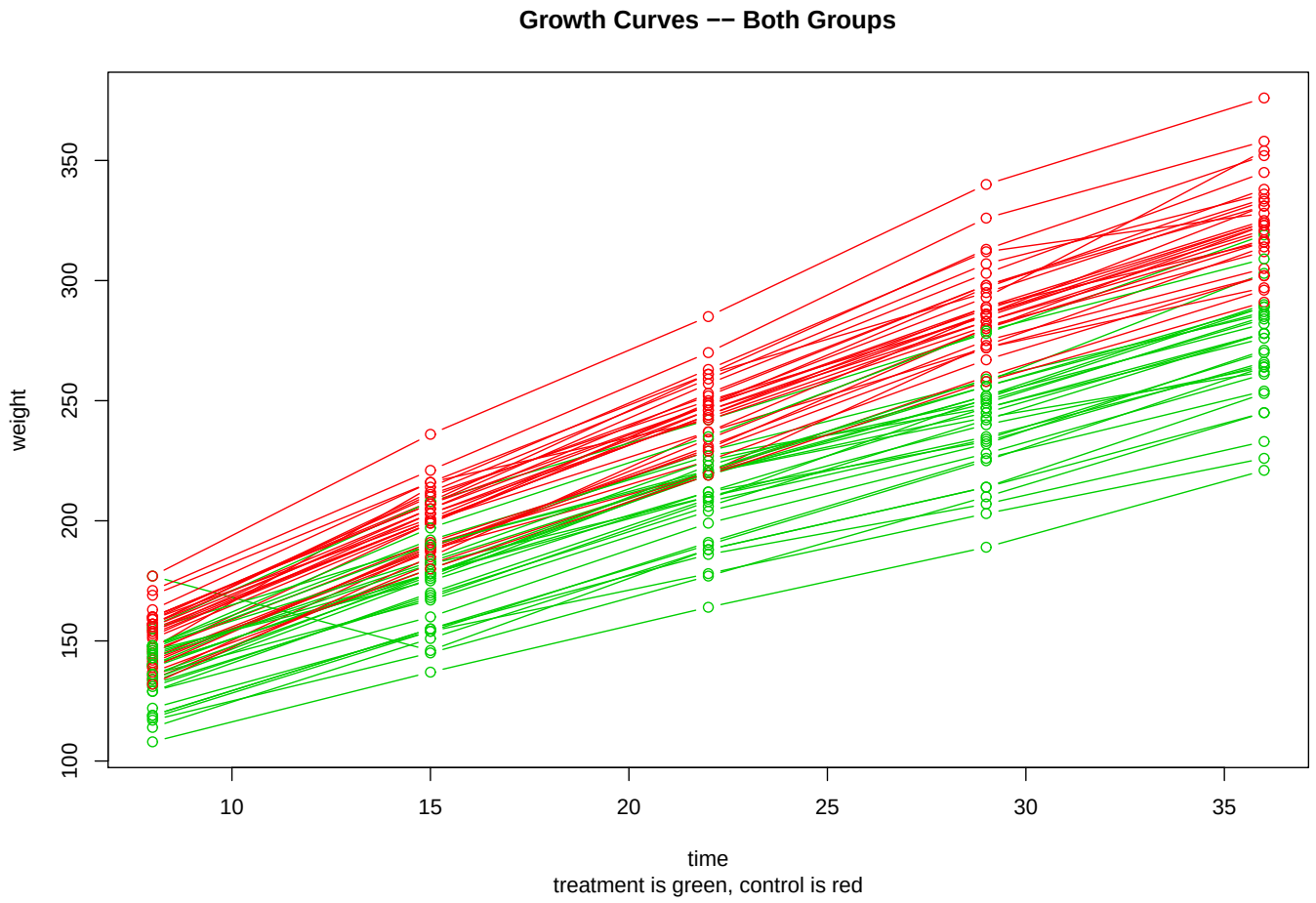

Figure 31: Growth curve for both groups of rats.

Rat growth curves

- It looks like the control group is growing much faster than the treatment group. So, perhaps these graphs have too much “inter-ocular impact”. However, let us continue with the example.
- To see the data, let us look at the differences in the groups over time. See figures 32-36.

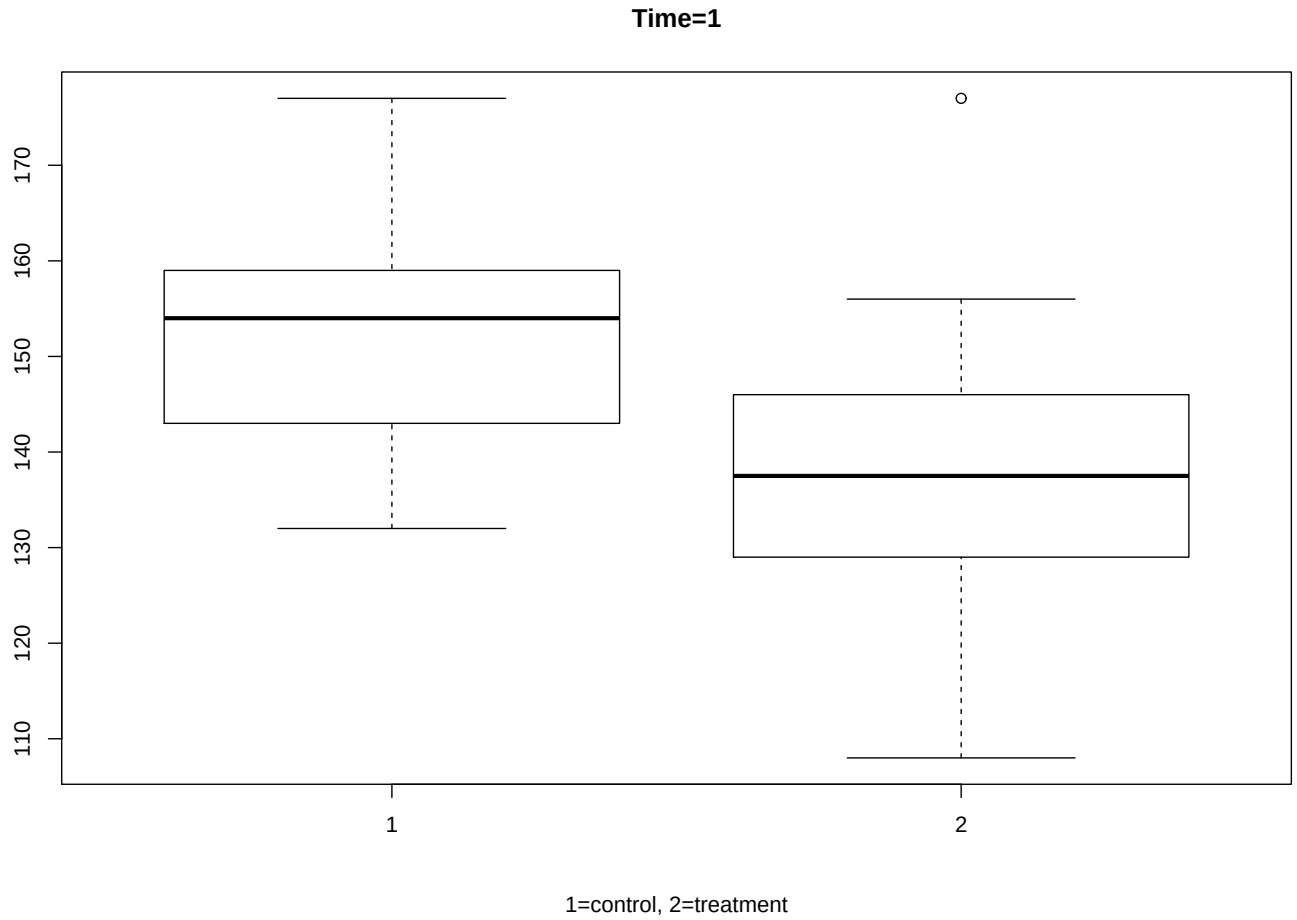

Figure 32: rat notes

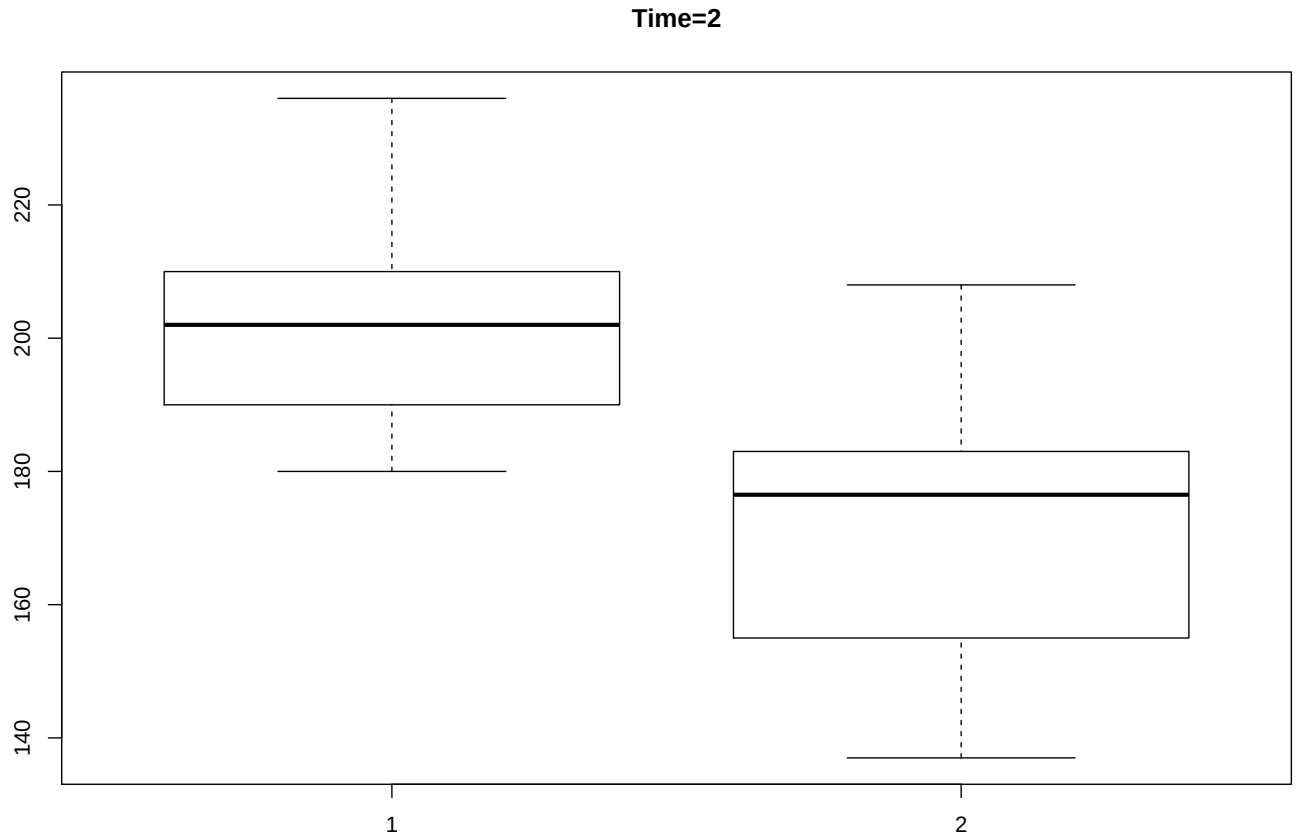

Figure 33: rat notes

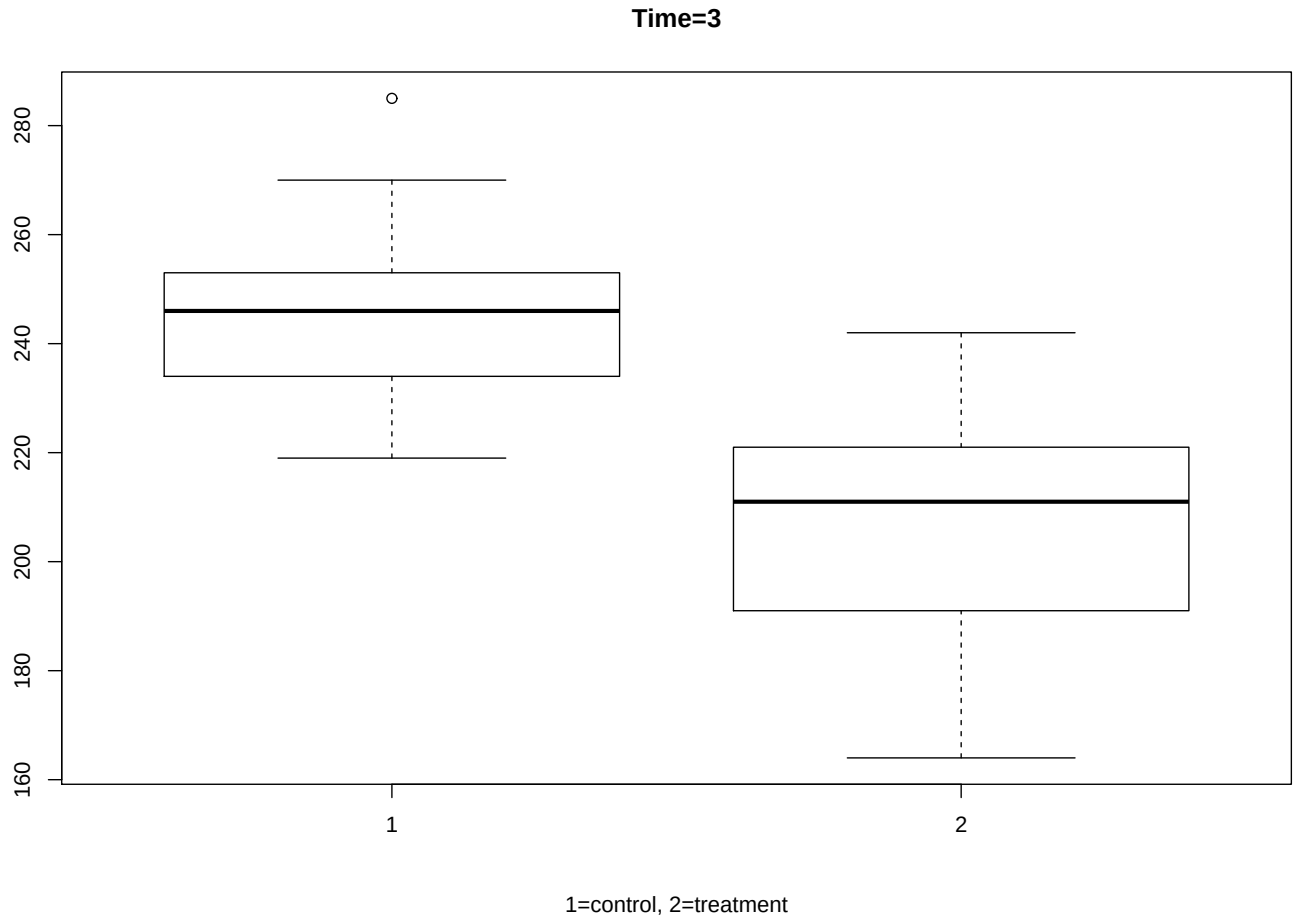

Figure 34: rat notes

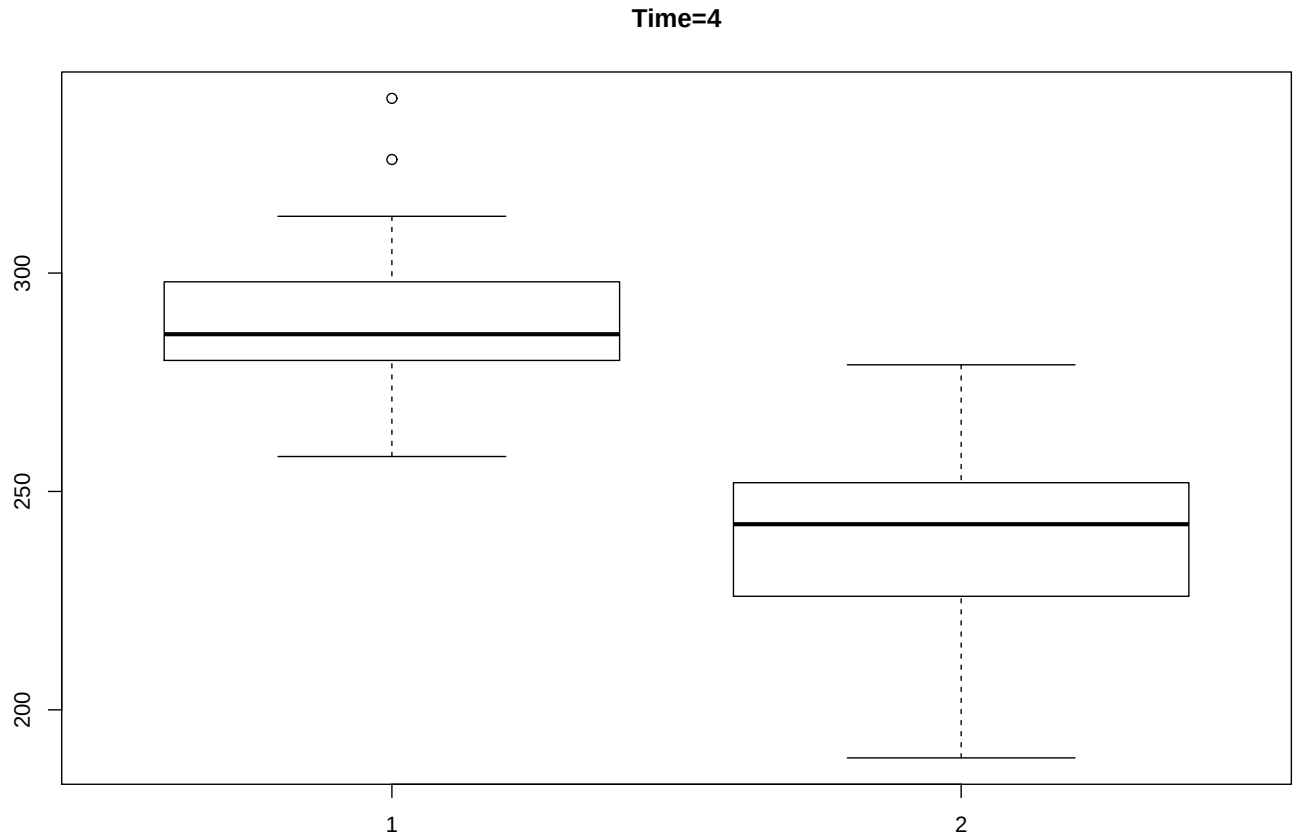

Figure 35: rat notes

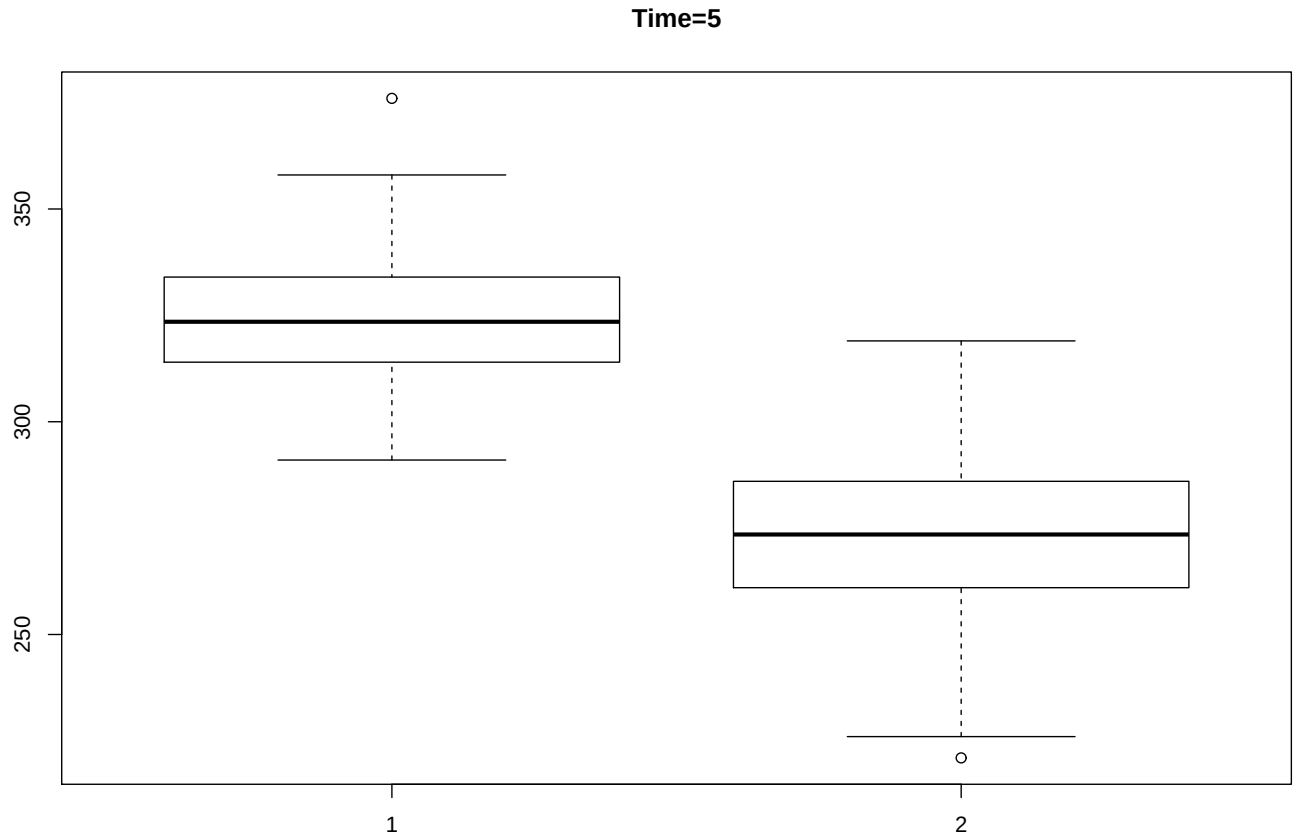

Figure 36: rat notes

Rat example

- So, it appears that one group is growing faster. So, this might imply that the slopes of the growth curve for one group is larger, on average, then for the other group.
- So, consider two stages.
  - In the first stage one models the line for each rat i as, say, $Y_{ij} = \alpha_i + \beta_i \text{Time}_j$.
  - In the second stage, one looks at the (α_i, β_i) for each group and see if these populations are different.
- As a first pass, simple approach, let us just fit the MLE for the line for each rat. These are in figures 37-39.

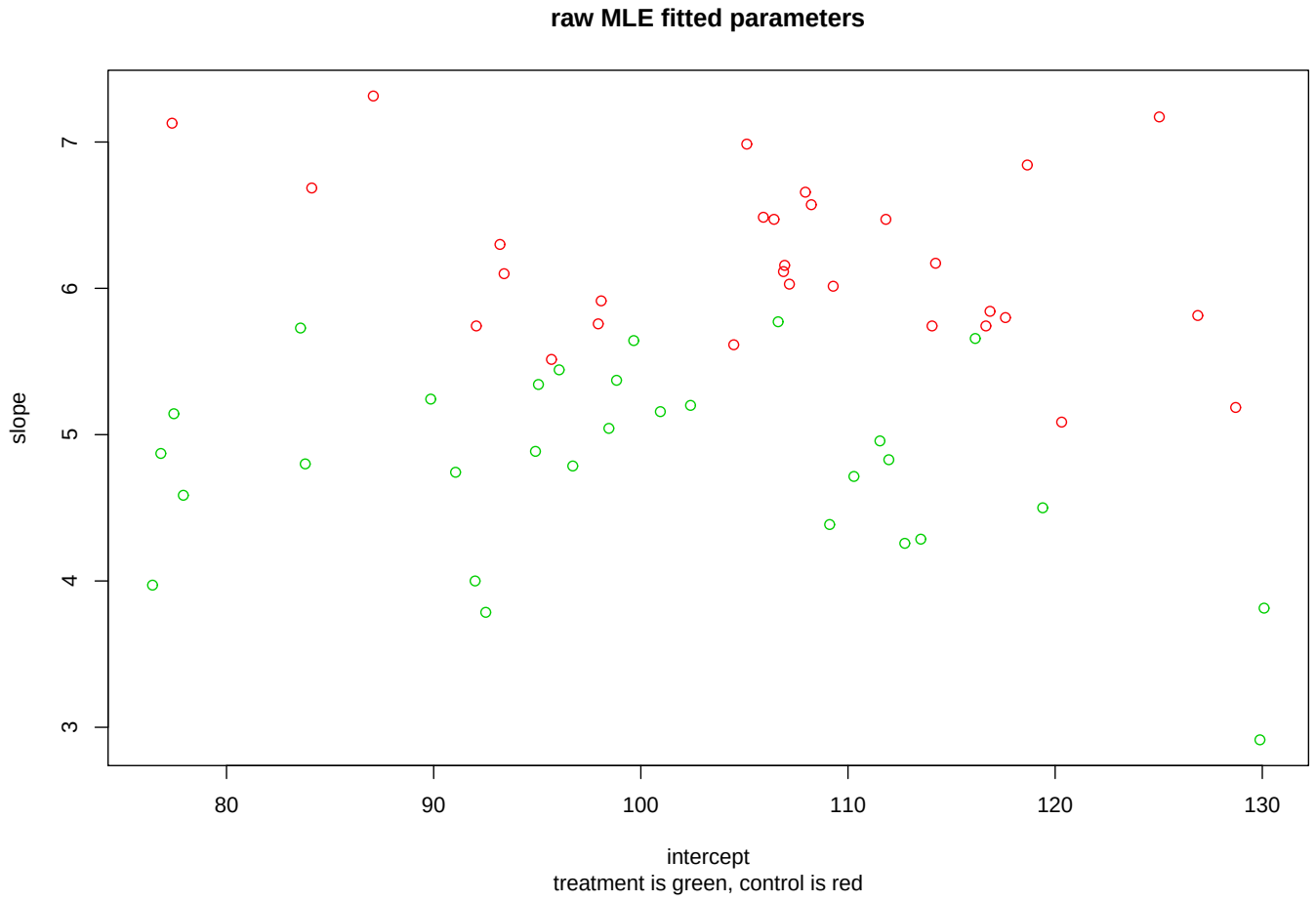

Figure 37: rat notes

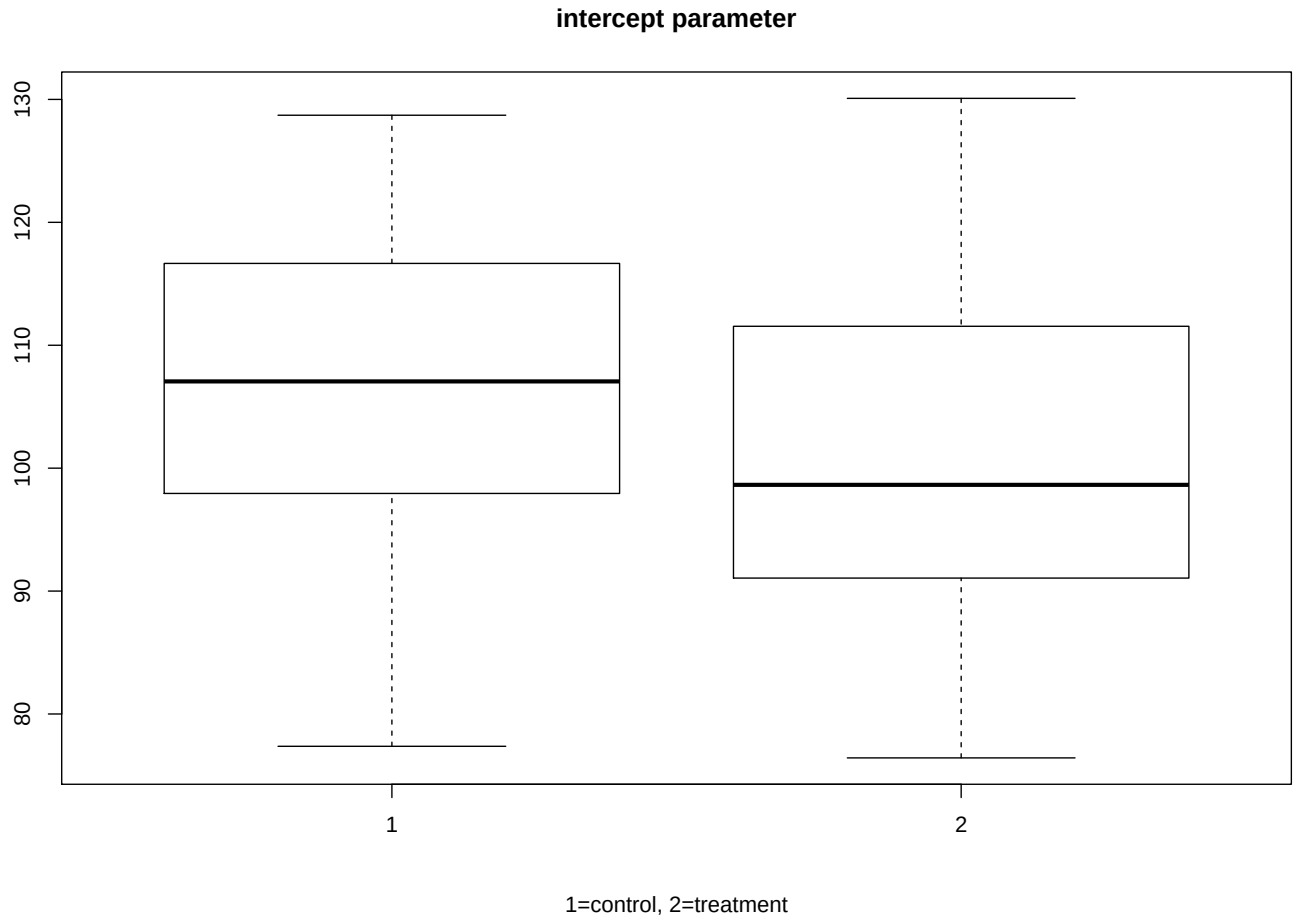

Figure 38: rat notes

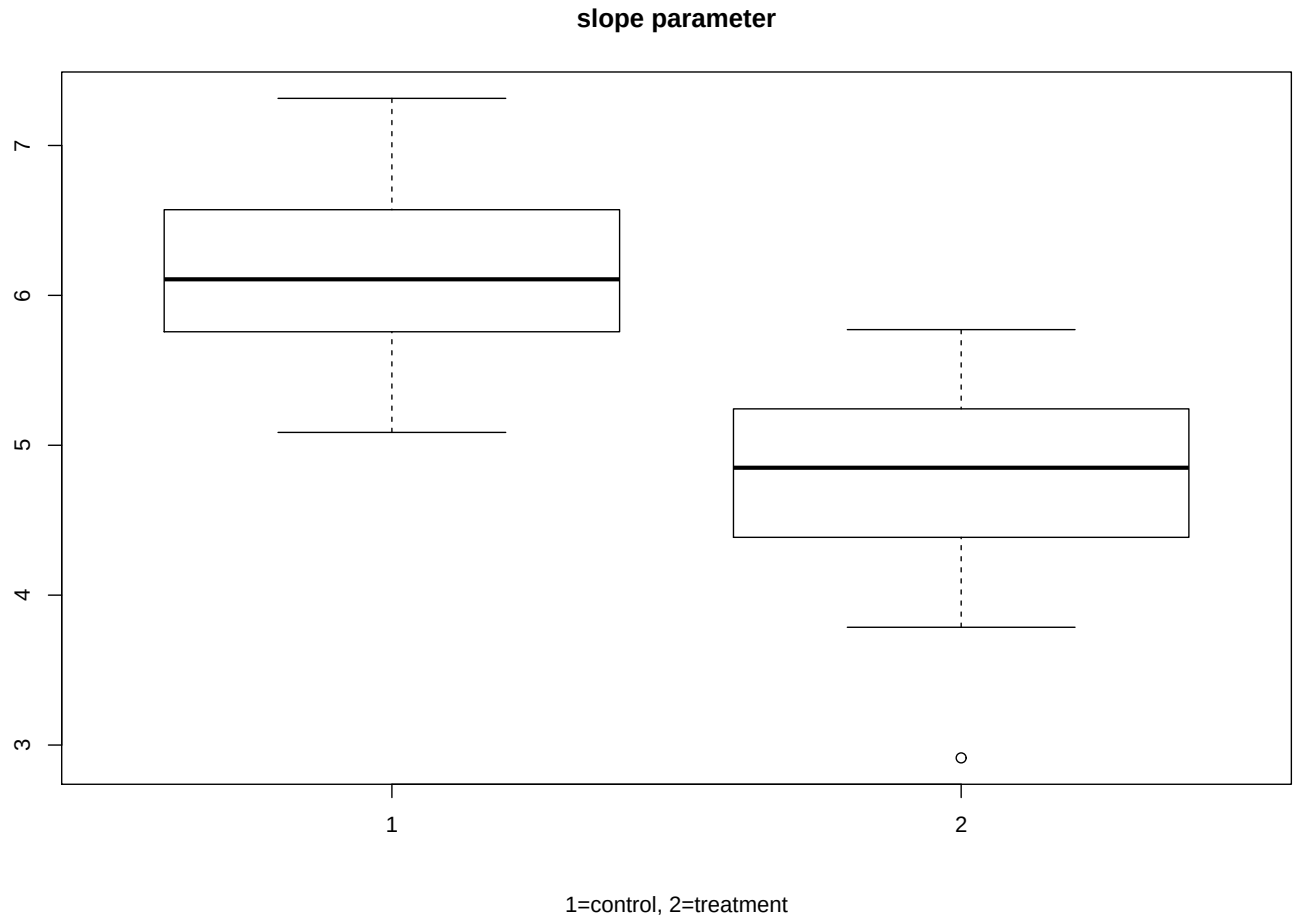

Figure 39: rat notes

Bayes approach to Rat growth curves

- The problem with just finding the individual MLE's is that this approach assumes that the MLE values are the exact data. This does not account for the error propagation due to estimating the MLE for each individual rat.
- There are frequentist methods which do calculate the overall MLE correctly. (These does get complicated.)
- Instead, let us consider the Bayesian model.

Bayes approach to Rat growth curves

- Note, that this is like the multi-level school problem.
- When running the MCMC, we can sample the (α_i, β_i) for each rat conditionalizing on all the other parameters in the model. So, at this stage, we are looking at the curve for each rat separately.
- For the difference between the two groups, we can just consider the (α_i, β_i) values know and look at the difference in the groups. Since these are sampled from the conditional distribution, this accounts for the uncertainty in the fitted values.
- Also, there will be some "regression" to the mean for the estimated parameters. This is sometimes important when there might be a very small number of observations for a particular subject.

Bayes approach to Rat growth curves

The following WinBugs/OpenBugs code fits this model:

```
for(i in 1:N){
 for(j in 1:5){
   y[i,j]~dnorm(mu[i,j],tau)
   mu[i,j]<- a[i]+b[i]*x[j]
   }
  a[i]~dnorm(ma[i],ta)
  b[i]~dnorm(mb[i],tb)
  ma[i]<-ma0+ (2*drug[i]-3)*madiff
  mb[i]<-mb0+ (2*drug[i]-3)*mbdiff
  }
```

Bayes approach to Rat growth curves

with the following hyperpriors:

```
ma0~dnorm(100,.00001)
 mb0~dnorm(0,.0001)
 madiff~dnorm(0,.0001)
 mbdiff~dnorm(0,.0001)
 sau~dunif(0,250)
 sa~dunif(0,250)
 sb~dunif(0,250)
 tau<-pow(sau,-2)
 ta<-pow(sa,-2)
 tb<-pow(sb,-2)
```

Figures 40-41 shows the fits from the Bayesian model.

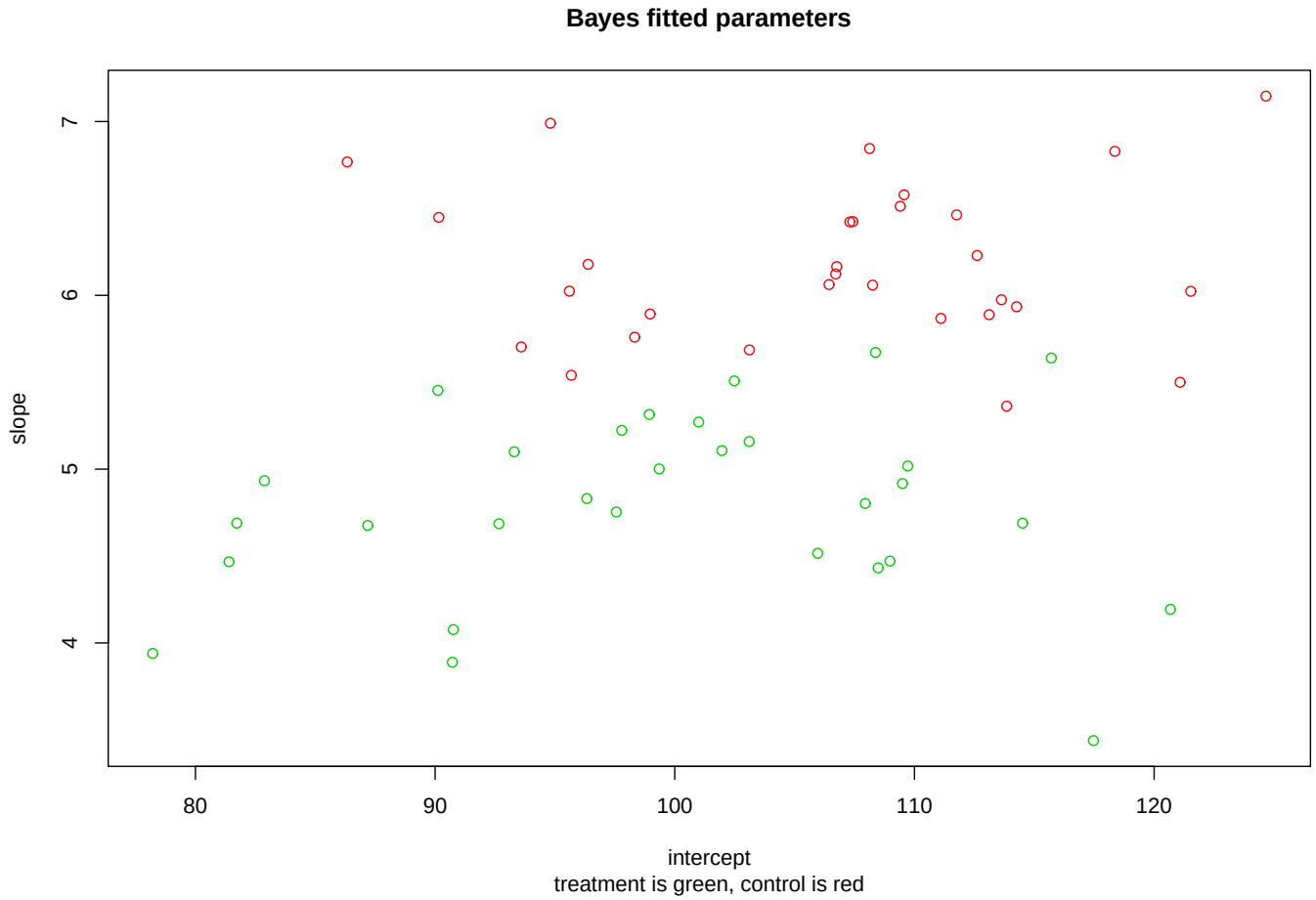

Figure 40: rat notes

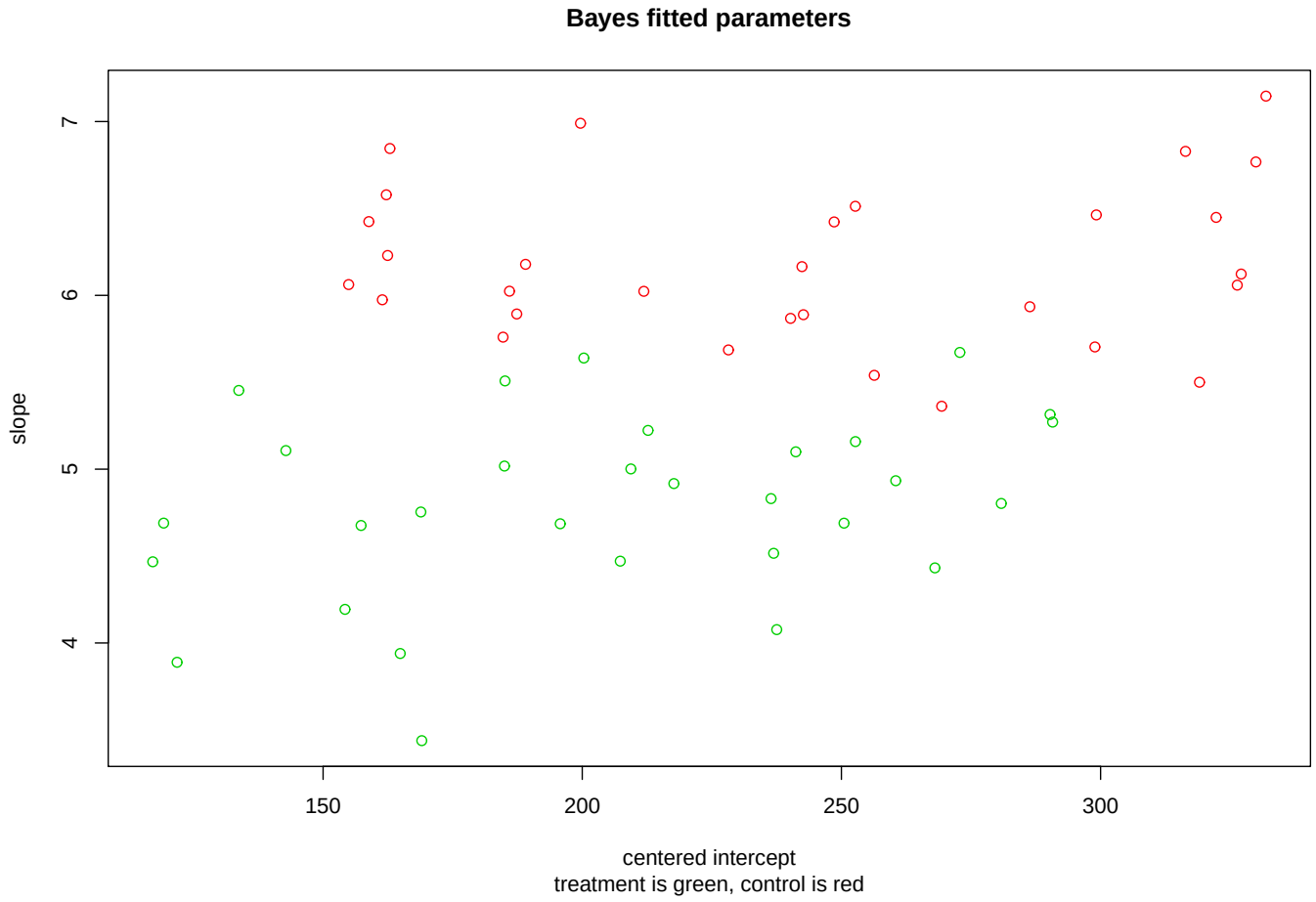

Figure 41: rat notes

The next several plots compare the difference between the Bayes fit and the naive MLE fit. Note that the Bayesian estimates are less extreme.

See figures 42-44

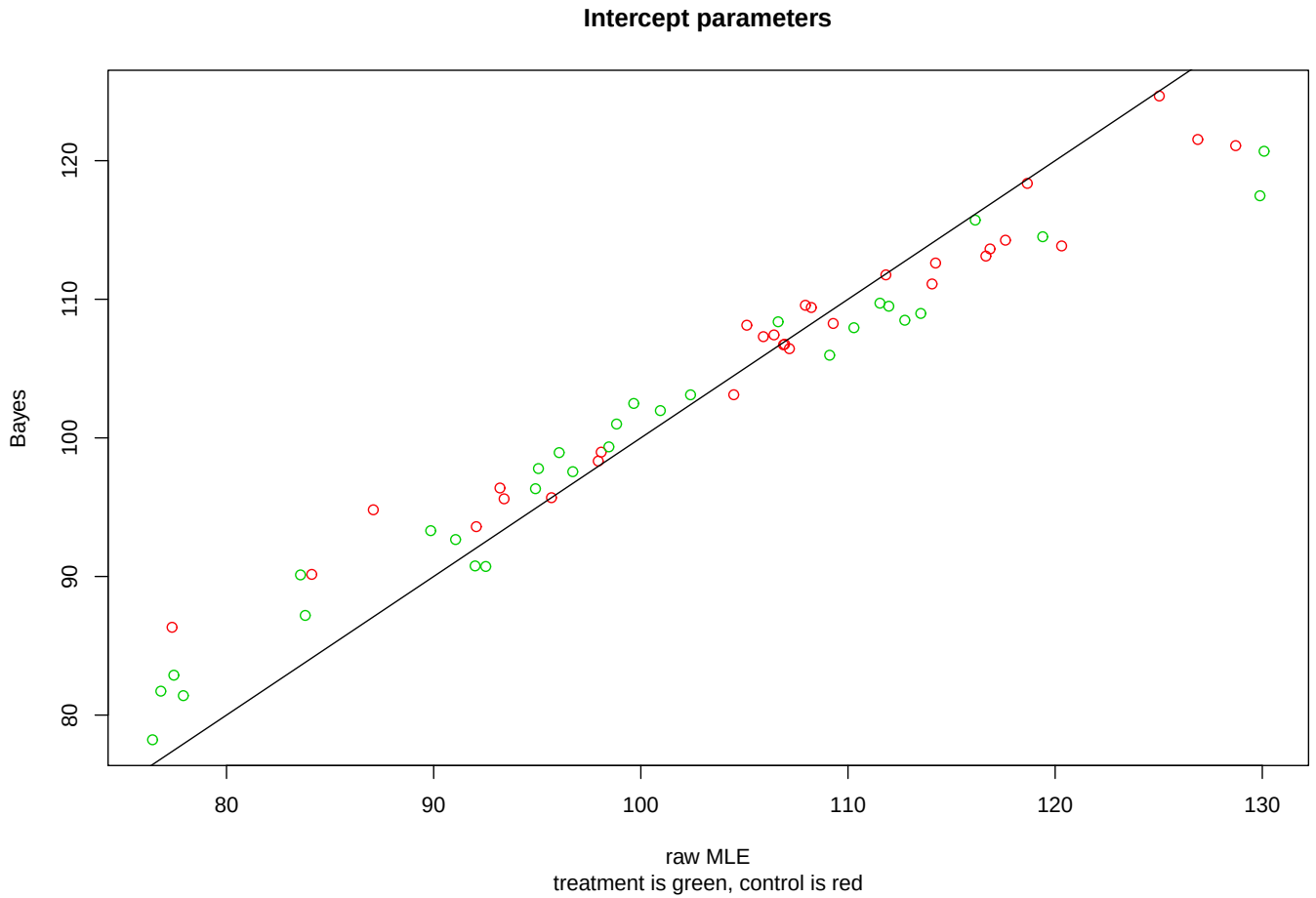

Figure 42: rat notes

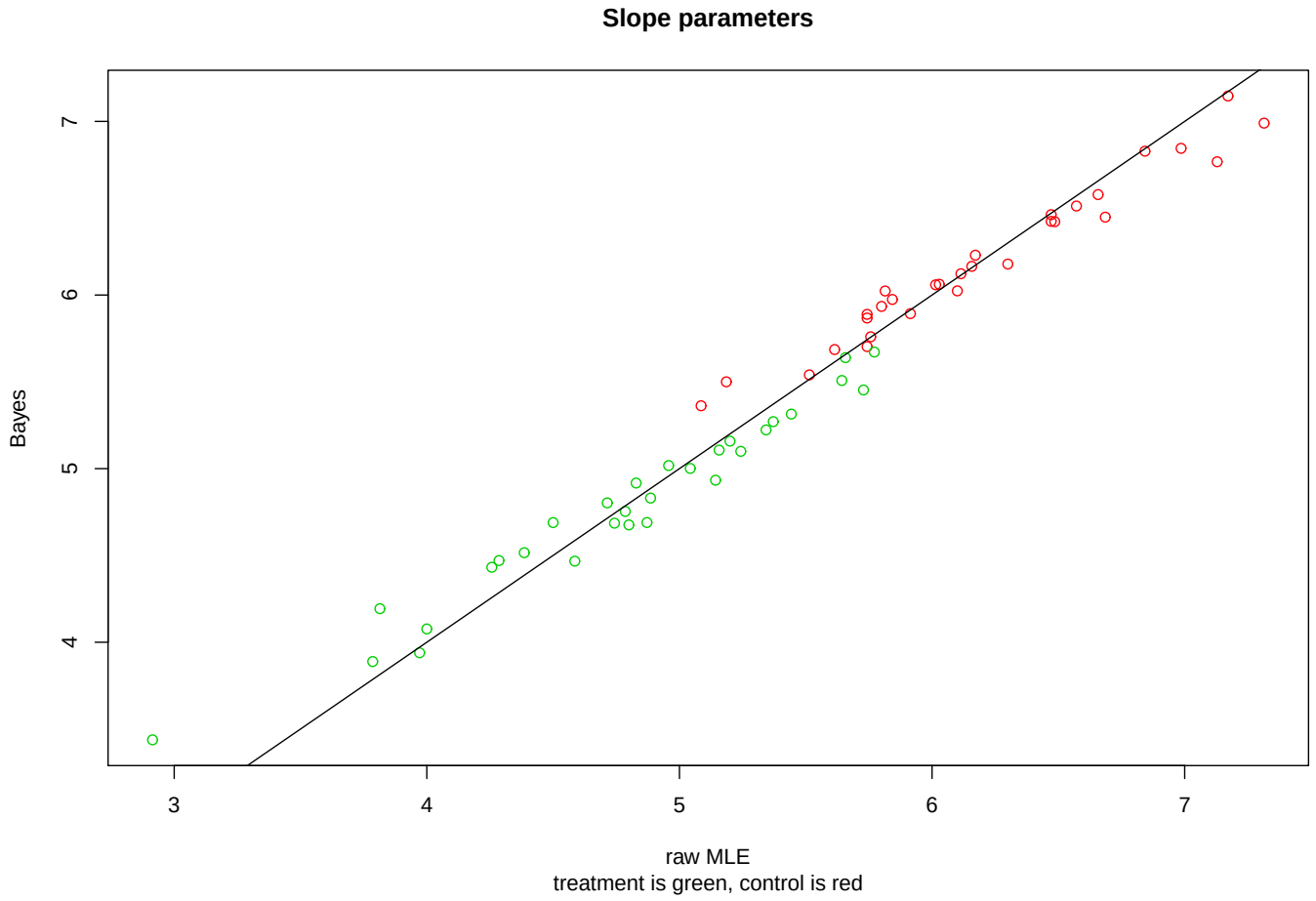

Figure 43: rat notes

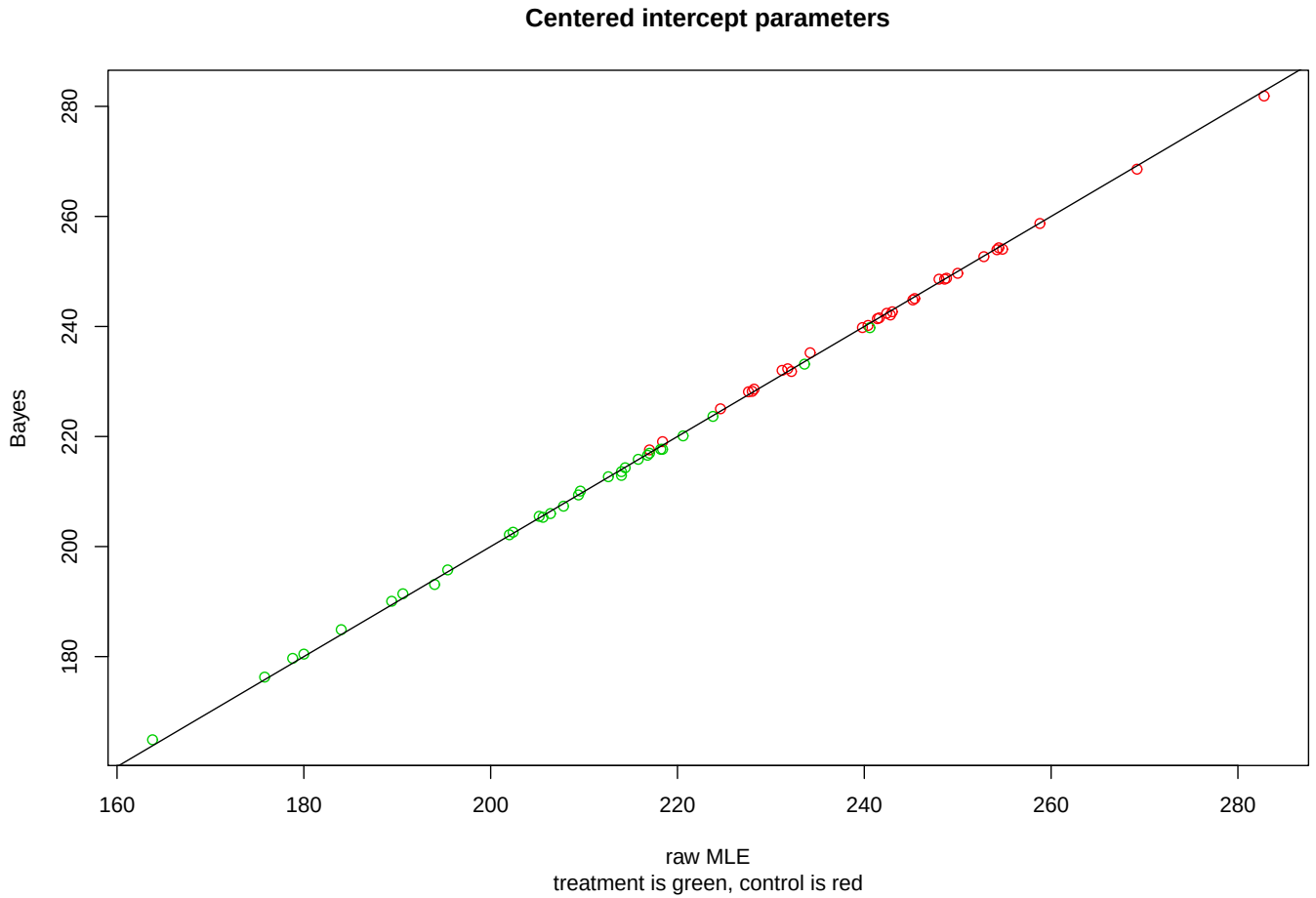

Figure 44: rat notes

The next several plots show the Bayesan contour plot for the difference between the two groups on both the slope and intercept dimension.

Then the next several plots look at the parameters of the model at the group level.

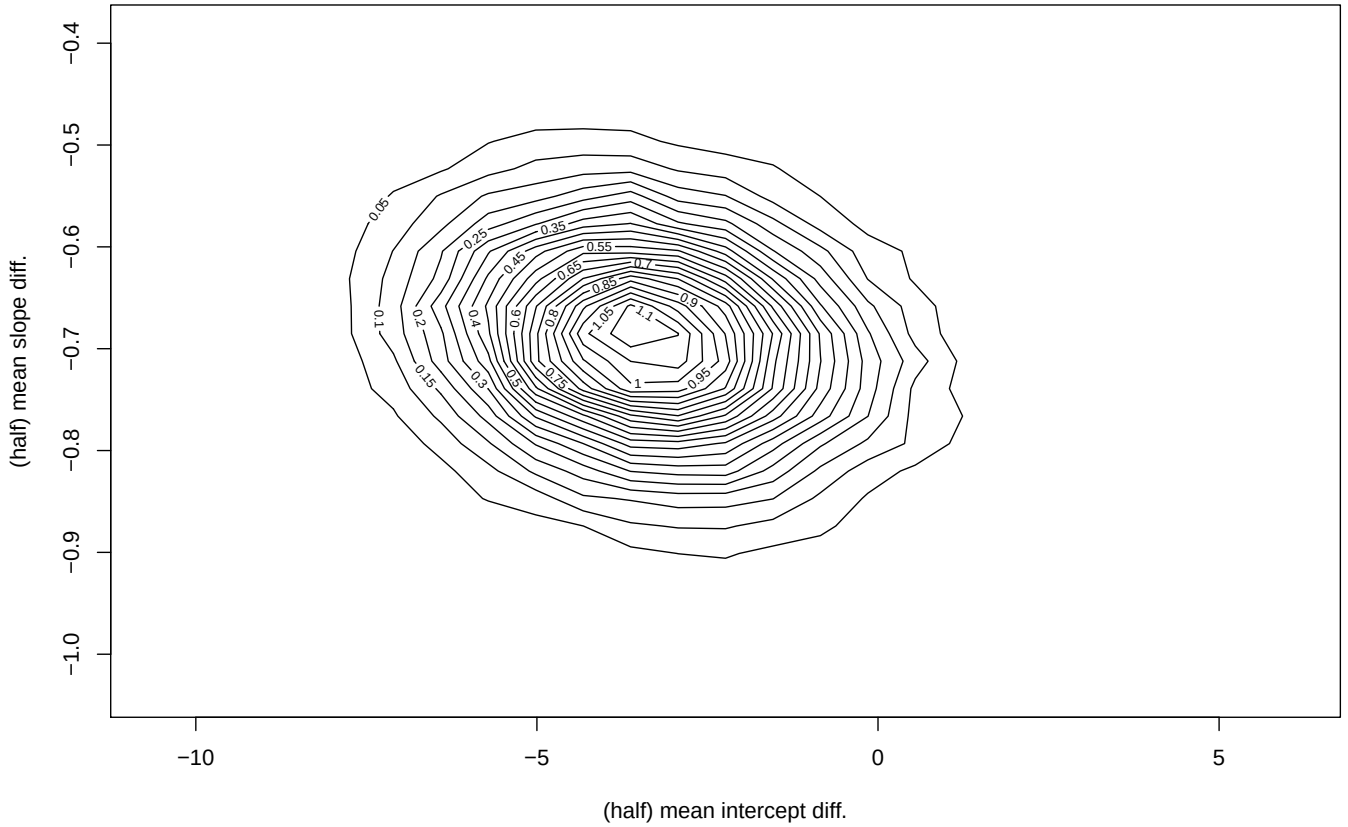

Figure 45: rat notes

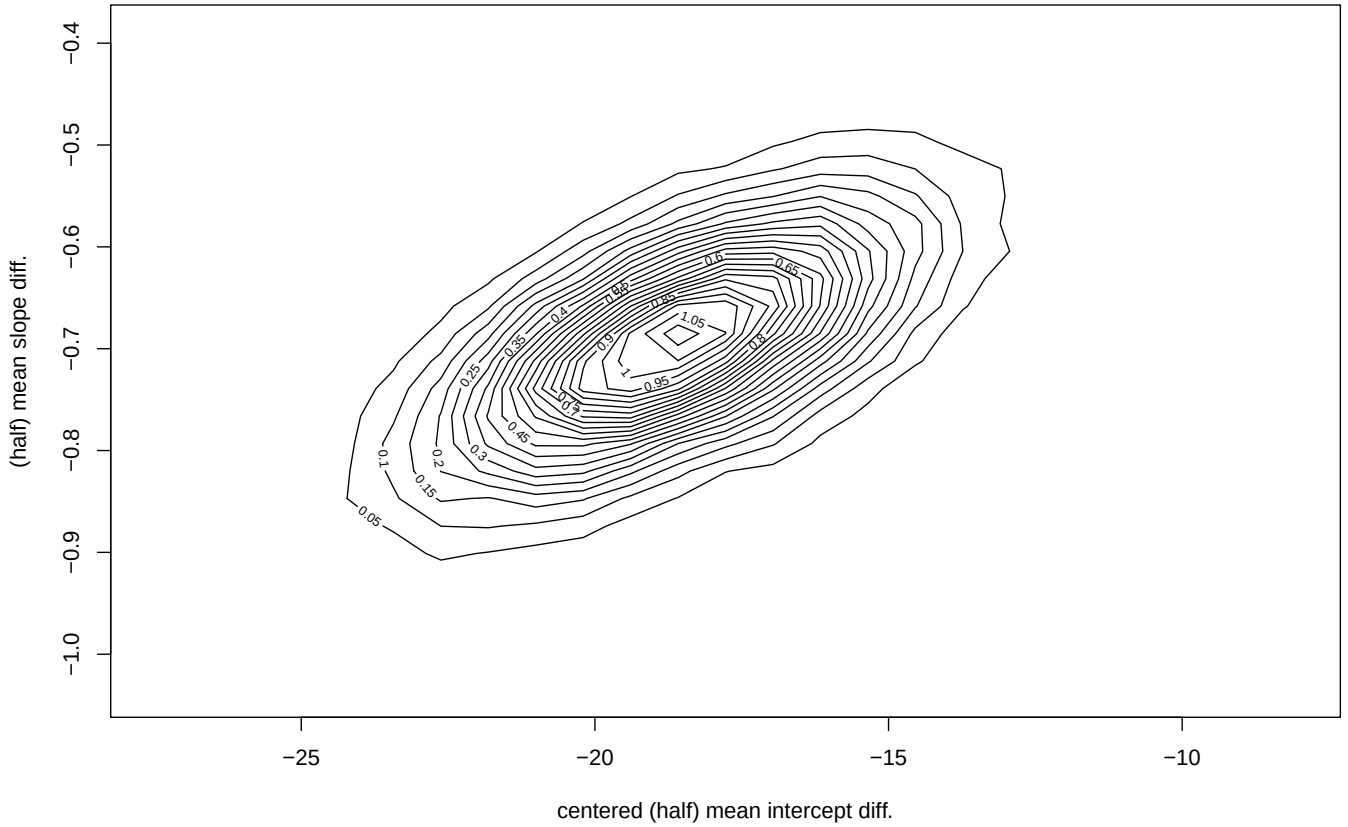

Figure 46: rat notes

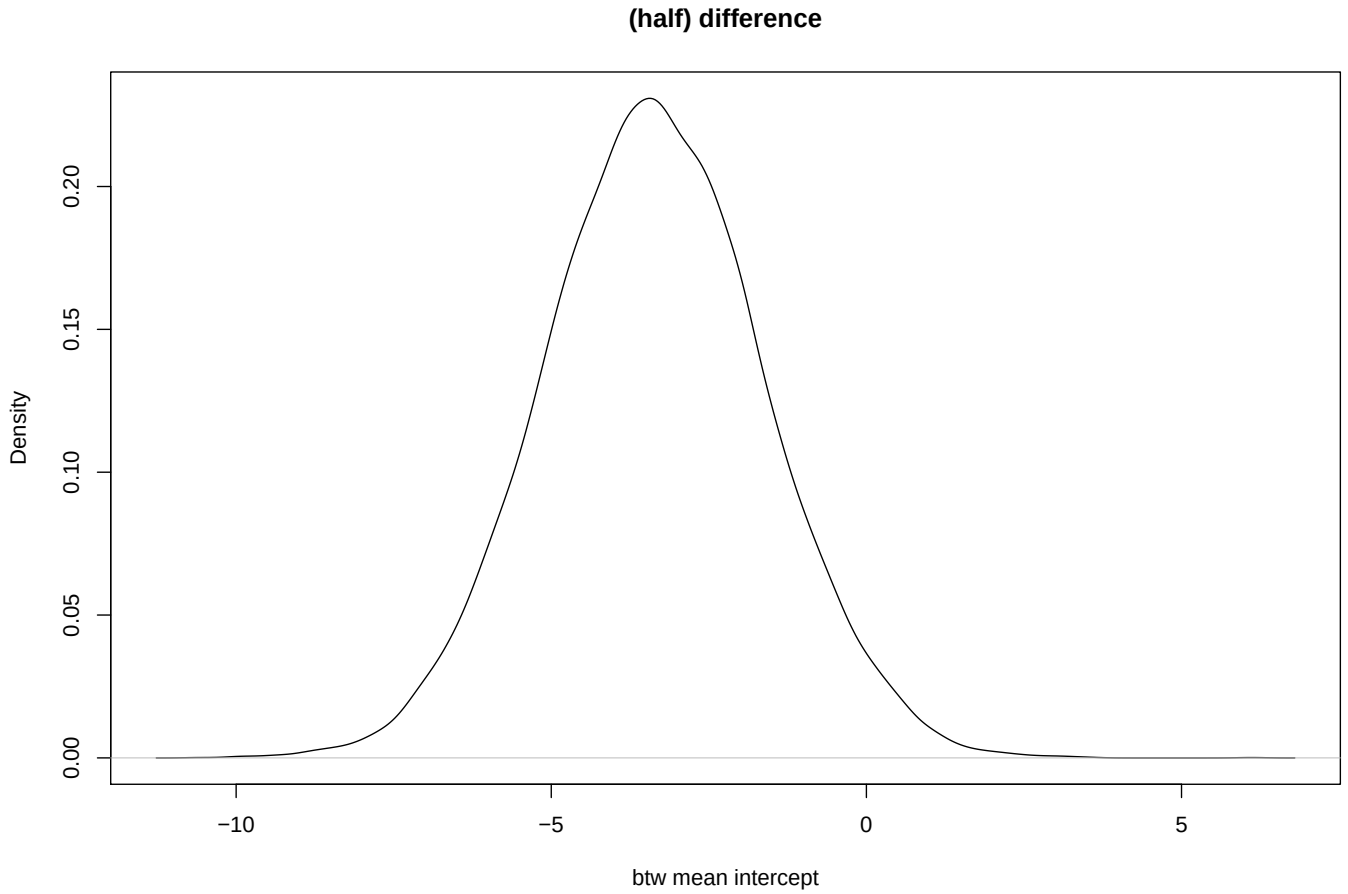

Figure 47: rat notes

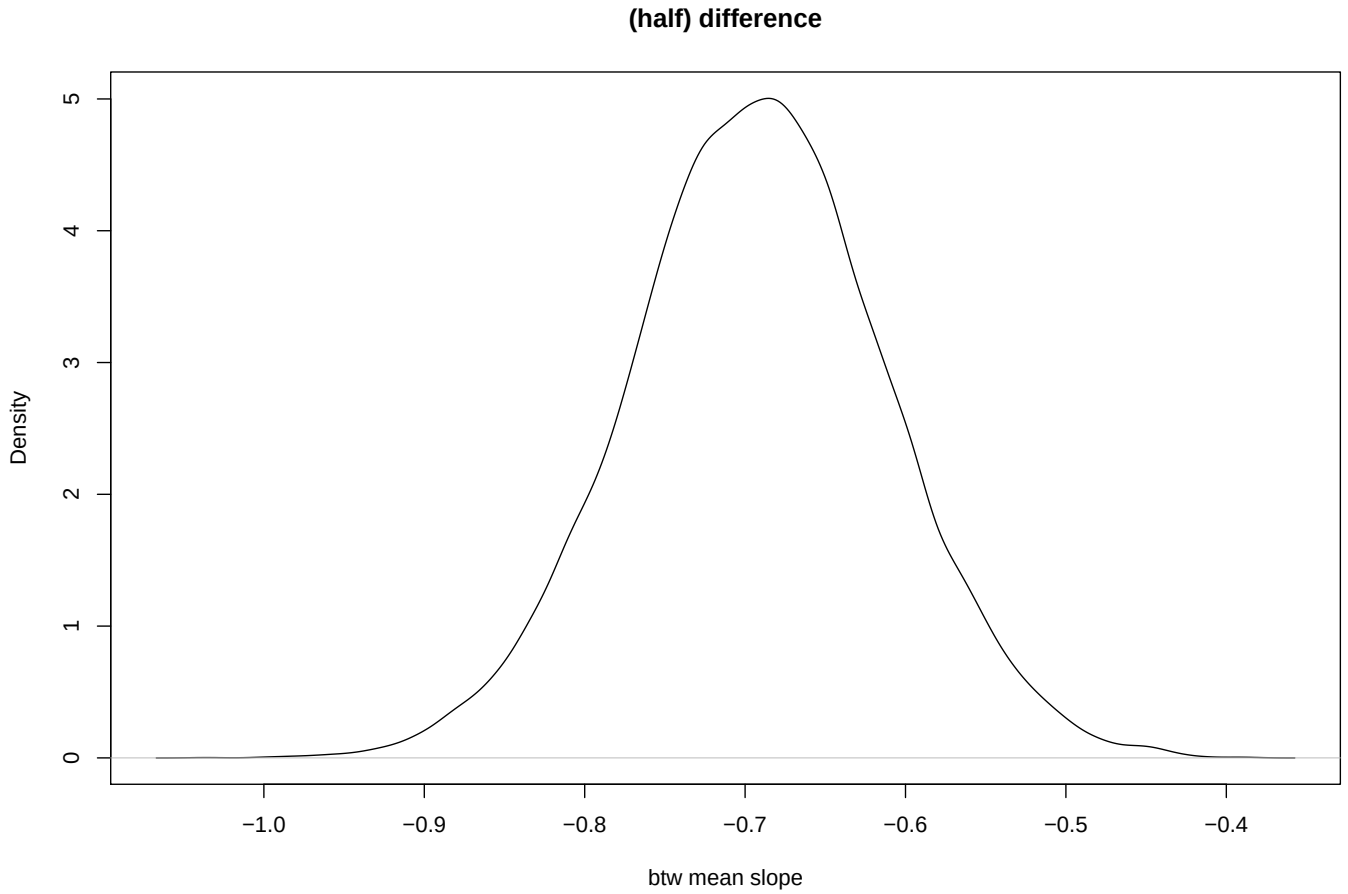

Figure 48: rat notes

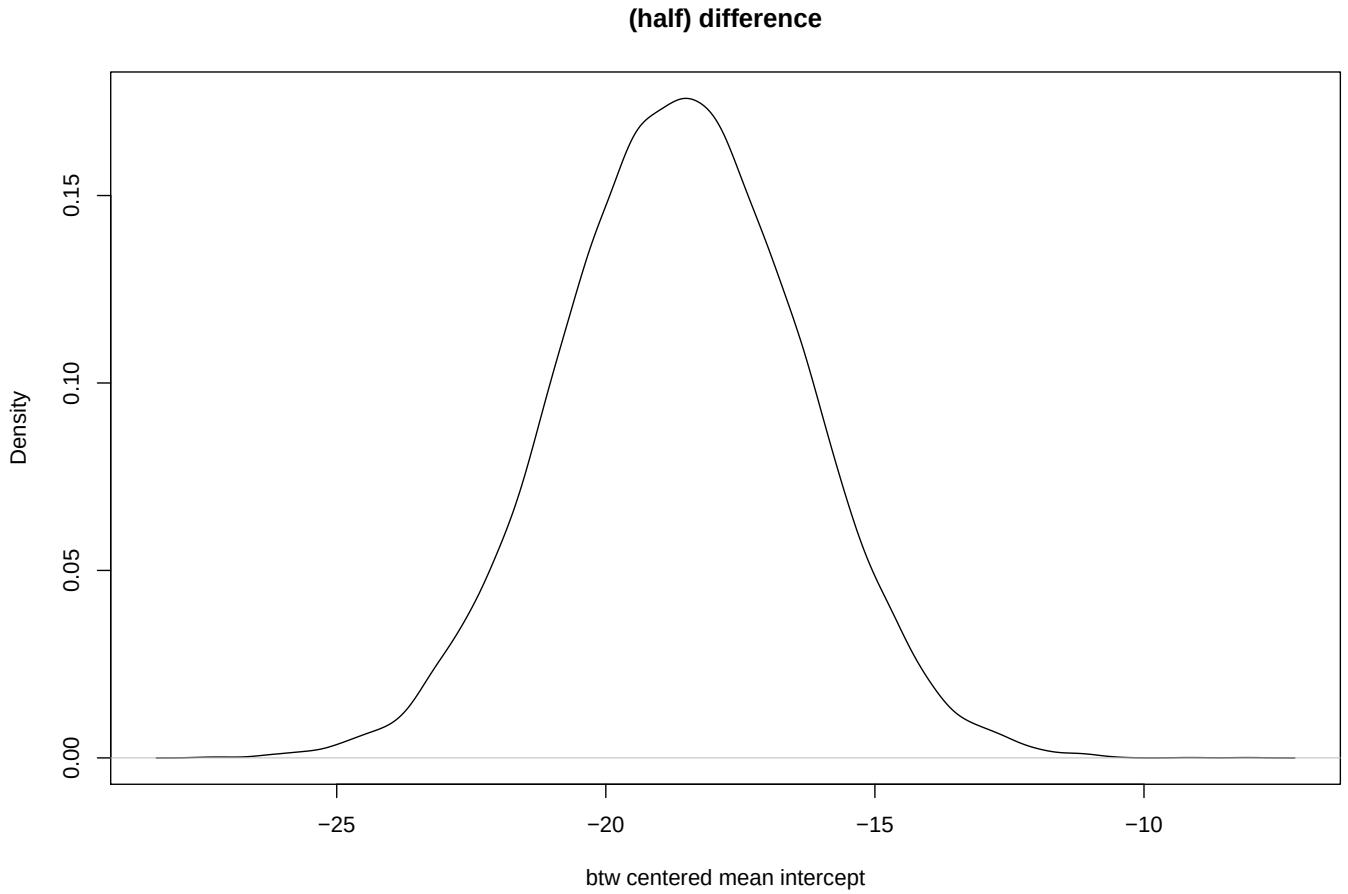

Figure 49: rat notes

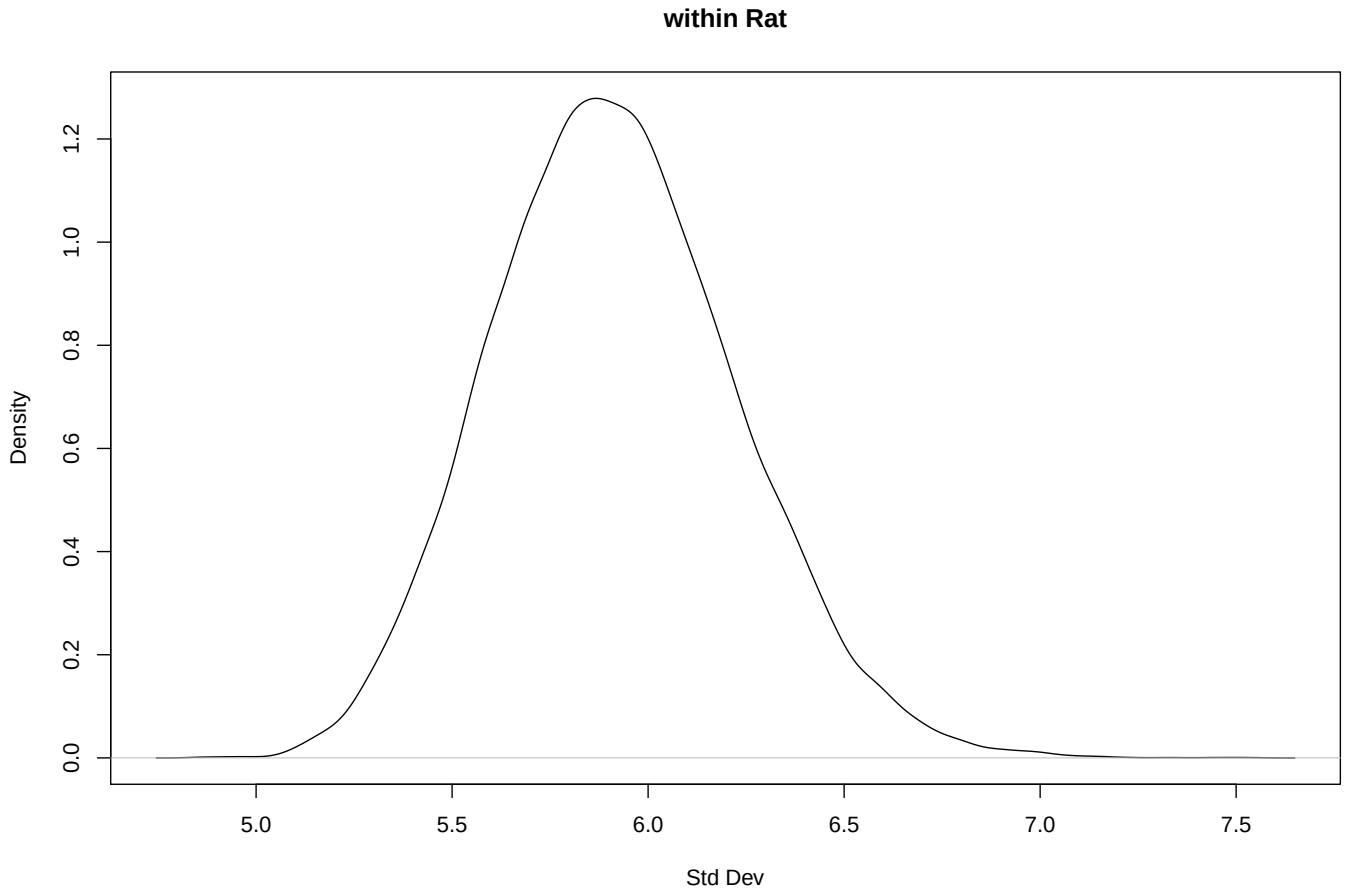

Figure 50: rat notes

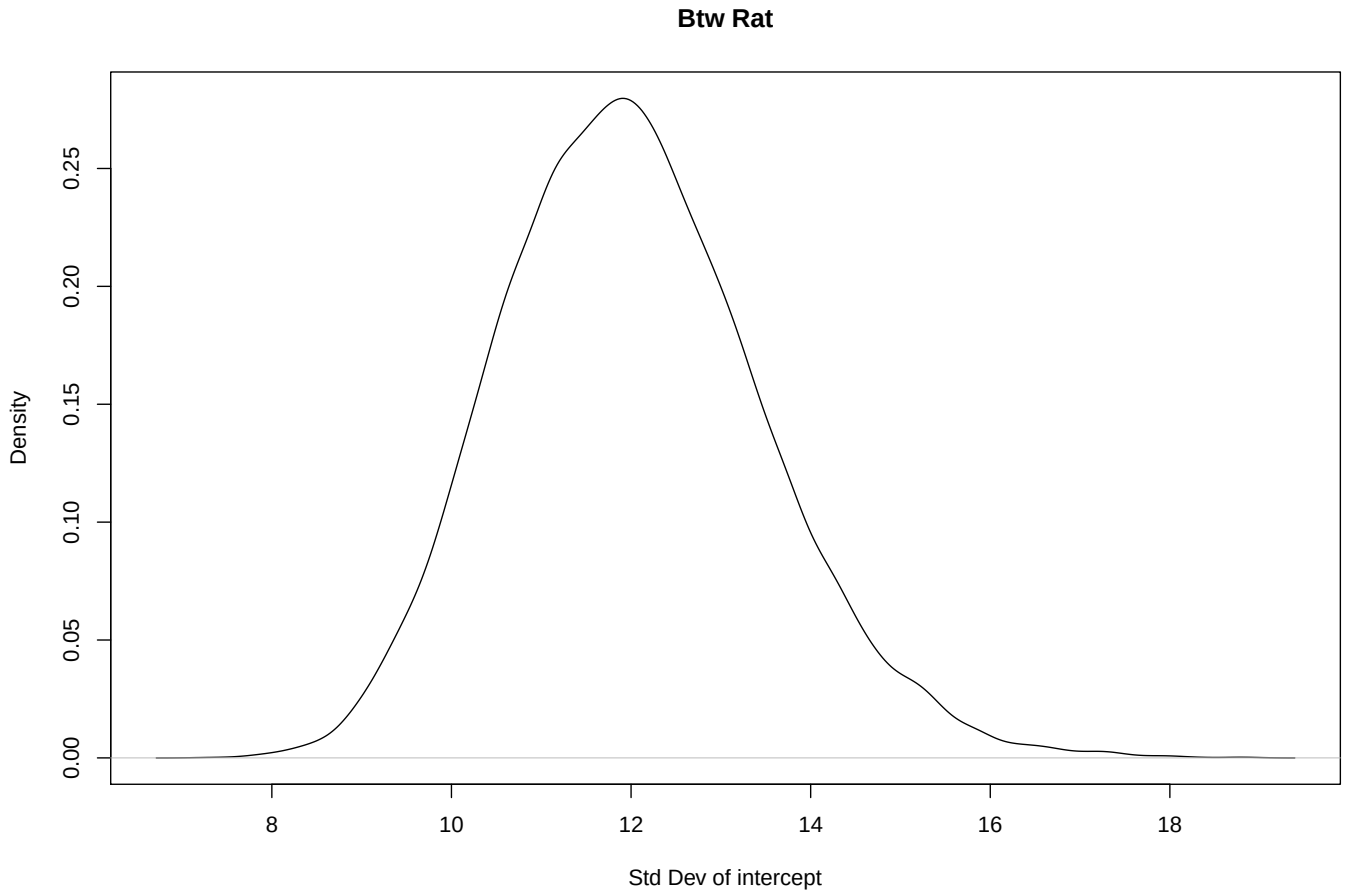

Figure 51: rat notes

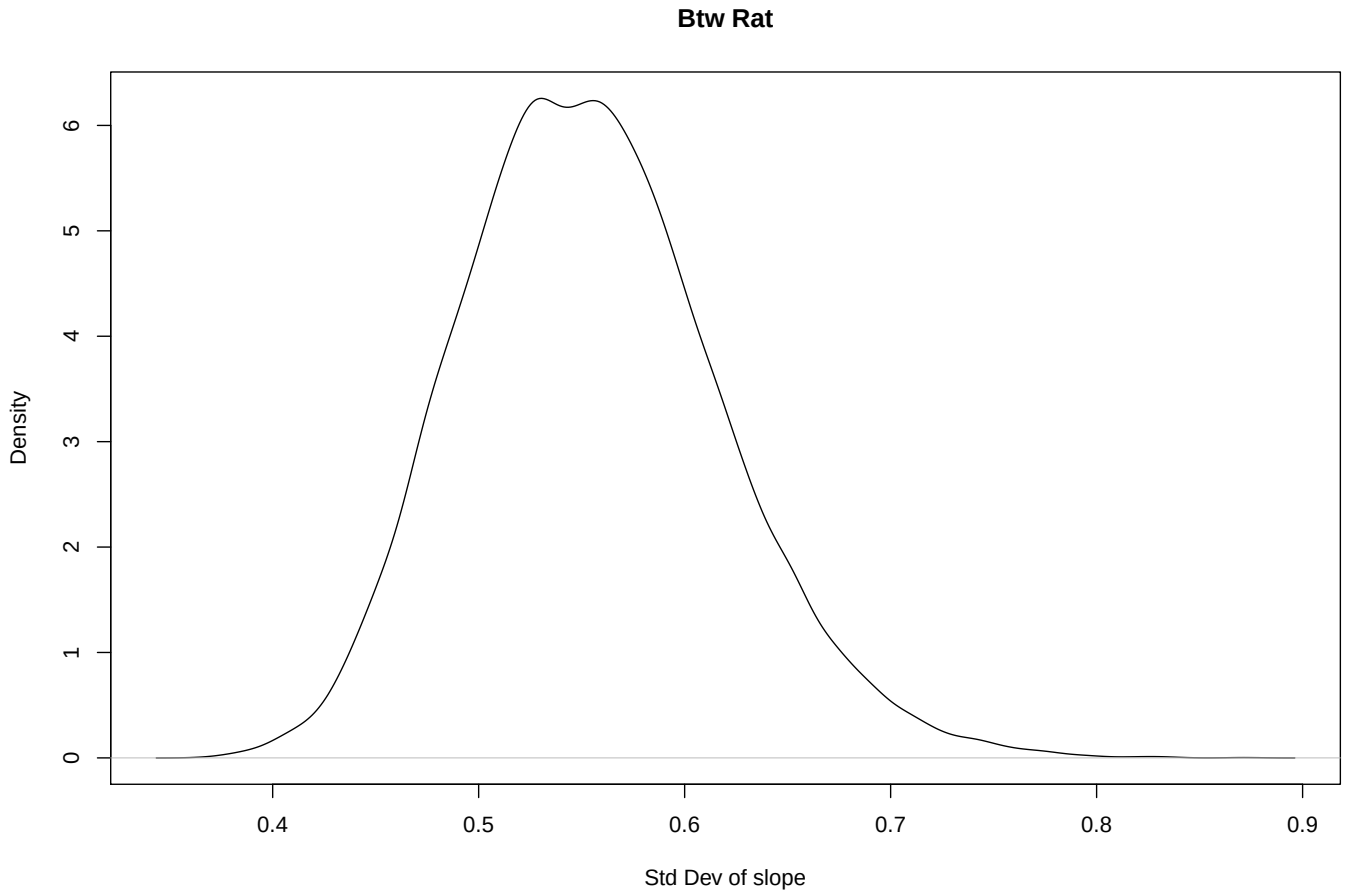

Figure 52: rat notes

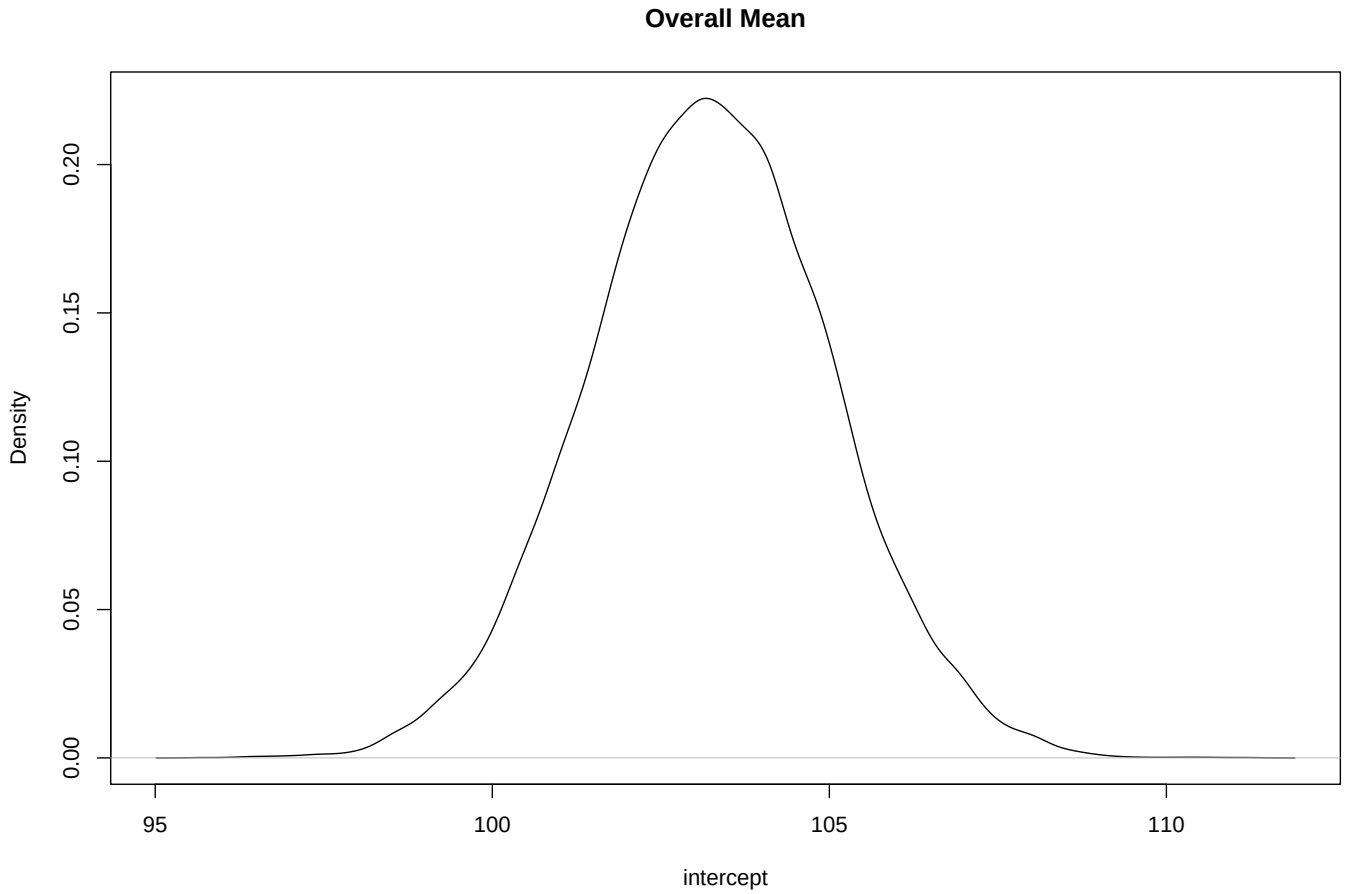

Figure 53: rat notes

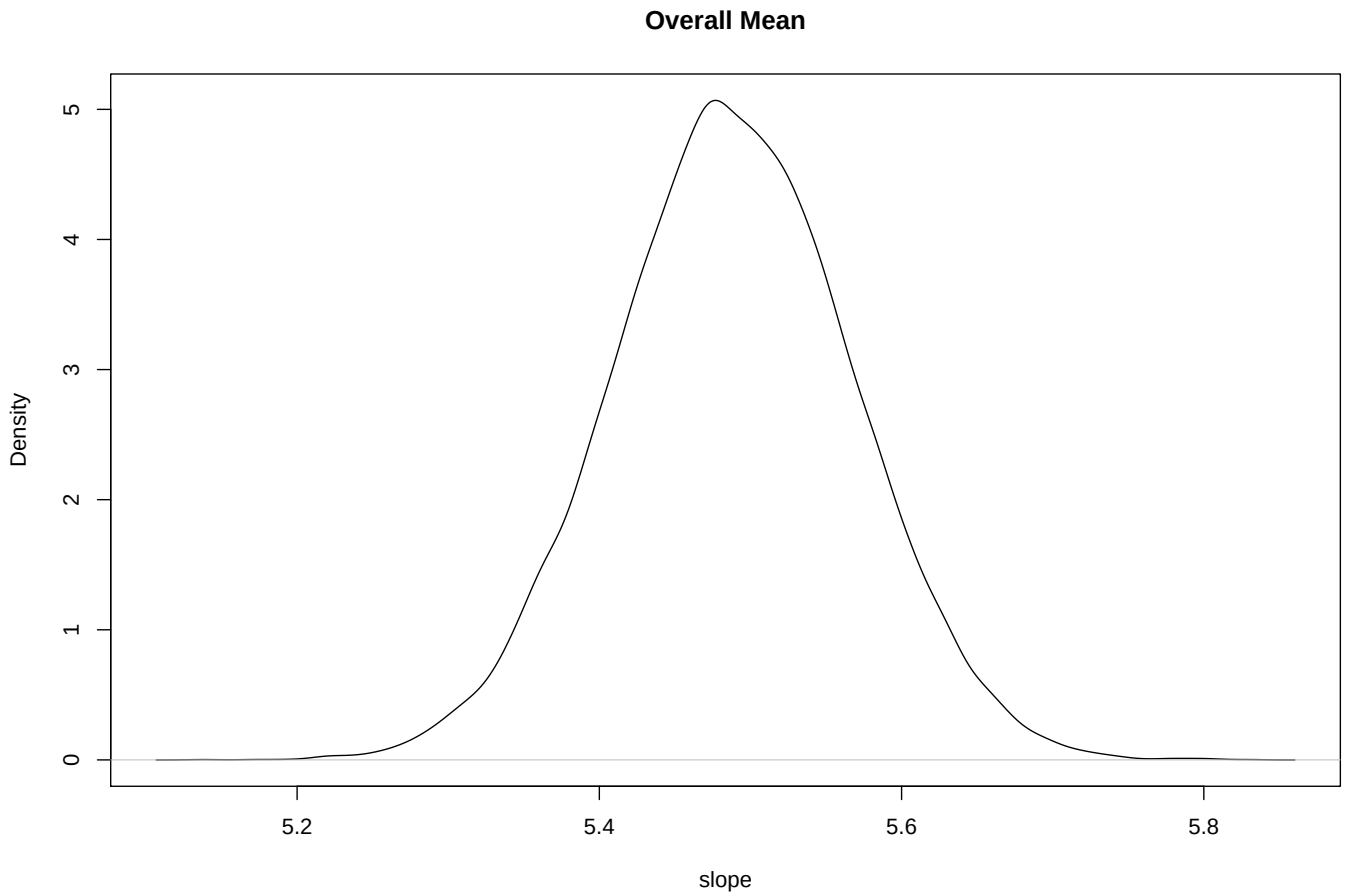

Figure 54: rat notes