# PHP2530 Problem Set 4

*Alyson Singleton*

*5/9/2020*

## 1. The Importance Sampling Algorithm

### Part A

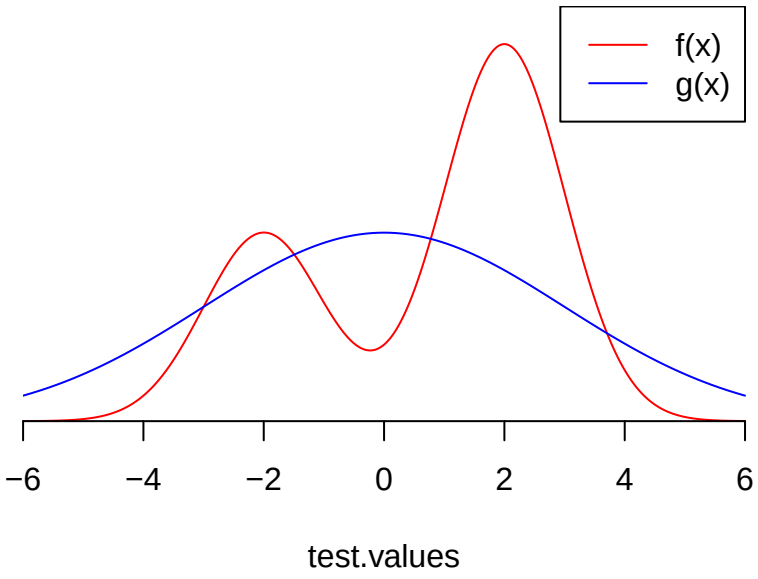
I used the following code as guidance but created my own version of the algorithm with explanations of each step to appropriately display my understanding and effort [http://sites.fas.harvard.edu/~stat221/ProblemSets/hw3_sols_code.R]. I also used your suggested reading [Jun S. Liu, 2012].

```
########################################
# 1. The Importance Sampling Algorithm
ImpSampler <- function(nSamples, logTargetDensityFunc, logProposalDensityFunc,
                       proposalNewFunc, rejectionControlConstant = NULL) {
  # first check what's up with the rejectionControlConstant, i.e. is it null? if yes then ...
  if (is.null(rejectionControlConstant)) {
    # initialize samples vector w proposalnewfunc values (i.e. N(0,3^2))
    samples.vec <- rep(NA,nSamples)
    for (i in 1:nSamples){
      samples.vec[i] <- proposalNewFunc()
    }
    # initialize and find log weights
    log.weights.vec <- rep(NA,nSamples)
    final.log.weights.vec <- sapply(samples.vec, logTargetDensityFunc) -
      sapply(samples.vec, logProposalDensityFunc)
    # store samples for output
    final.samples <- samples.vec
    # acceptance rate doesn't apply here
    acceptance.rate <- NA
    # calculated estimated ESS as directed
    estimated.ESS <- length(final.log.weights.vec) /
      (1 + var(exp(final.log.weights.vec)))
  # if rejectionControlConstant was not null then
  }else{
    #initialize storage
    list.log.ratios = c()
    final.log.weights.vec = c()
    final.samples = c()
    # keep going until we've found the number of samples we want
    while (length(final.samples) < nSamples) {
      # again initialize samples vector w proposalnewfunc values (i.e. N(0,3^2))
      samples.vec <- rep(NA,nSamples)
      for (i in 1:nSamples){
        samples.vec[i] <- proposalNewFunc()
      }
      # again initialize and find log weights
      log.weights.vec <- rep(NA,nSamples)
      log.weights.vec <- sapply(samples.vec, logTargetDensityFunc) -
        sapply(samples.vec, logProposalDensityFunc)
```

```
      # now perform rejection control across all weights
      # we're working on the log scale so subtract log("c" val)
      log.ratios <- log.weights.vec - log(rejectionControlConstant)
      # take log(min(1,w/"c")) i.e. min(0,log(weight/"c"))
      log.ratios <- ifelse(log.ratios>0,0,log.ratios)
      # accept or reject with calculated probability, again on log scale
      acceptance.bool <- (log(runif(nSamples)) < log.ratios)
      # add those that pass to the list of final samples
      final.samples <- c(final.samples, samples.vec[acceptance.bool])
      # store the log.ratios that we tried
      list.log.ratios <- c(list.log.ratios, log.ratios)
      # update weights (w/r) and add to list of final weights
      final.log.weights.vec <- c(final.log.weights.vec,
                                 log.weights.vec[acceptance.bool] -
                                   log.ratios[acceptance.bool])
    }
    #calculate acceptance rate
    acceptance.rate <- length(final.samples)/length(list.log.ratios)
    #make sure we have no extra samples (chop off the extras if we do),
        #and make final update to weights (p*w/r)
    final.log.weights.vec <- final.log.weights.vec[1:nSamples] +
      log(mean(exp(list.log.ratios)))
    #again remove extra samples
    final.samples <- final.samples[1:nSamples]
    #calculate the estimated ESS
    estimated.ESS <- nSamples/(1 + var(exp(final.log.weights.vec)))
  }
  return(list(final.samples, final.log.weights.vec, estimated.ESS, acceptance.rate))
}
```

### Part B/C

See the requested plot below. Indeed, the suggested g(x) does seem like an appropriate importance density function as it will help us identify the modes of our mixture model.



**Part D**

First explore the expectation of $\mu_1$.

$$\mu_1 = E\left(\frac{1}{3}\text{Norm}(-2,1) + \frac{2}{3}\text{Norm}(2,1)\right)$$

$$\mu_1 = E\left(\frac{1}{3}\text{Norm}(-2,1)\right) + E\left(\frac{2}{3}\text{Norm}(2,1)\right)$$

$$\mu_1 = \frac{1}{3}(-2) + \frac{2}{3}(2) = \frac{2}{3}$$

Next investigate the expectation of $\mu_2$.

$$\mu_2 = E\left(\left(\frac{1}{3}\text{Norm}(-2,1) + \frac{2}{3}\text{Norm}(2,1)\right)^2\right)$$

To start, we can write:

$$E[X^2] = \sum_{i=1}^{n} w_i(E[X_i^2])$$

Then, we know,

$$Var[X] = E[X^2] - E[X]^2$$

Which, in this situation, let's us write:

$$\sigma_i^2 = E[X_i^2] - \mu_i^2$$

And, finally,

$$E[X^2] = \sum_{i=1}^{n} w_i(\sigma_i^2 + \mu_i^2)$$

Therefore:

$$E[X^2] = \frac{1}{3}(1+4) + \frac{2}{3}(1+4) = 5$$

Finally let us compute the expectation of $\theta$.

$$\theta = E\left(\exp\left(\frac{1}{3}\text{Norm}(-2,1) + \frac{2}{3}\text{Norm}(2,1)\right)\right)$$

This paper explores the form of the multivariate lognormal's moment generating function [https://www.casact.org/pubs/forum/15spforum/Halliwell.pdf]. If I had more time I would have been interested to explore the derivation, but for now I will simply make use of Halliwell's result.

$$E[\exp(X_i)] = \exp\left\{\mu_i + \Sigma_{ii}\frac{1}{2}\right\}$$

Therefore, we can calculate $\theta$ as follows:

$$\theta = \frac{1}{3}\exp(-2 + (1/2)) + \frac{2}{3}\exp(2 + (1/2)) \approx 8.196$$

**Part E**

See below for the outputs from the importance sampling algorithm without rejection control. It seems like our model is able to do fairly well.

Table 1: Importance Sampling Without Rejection Control

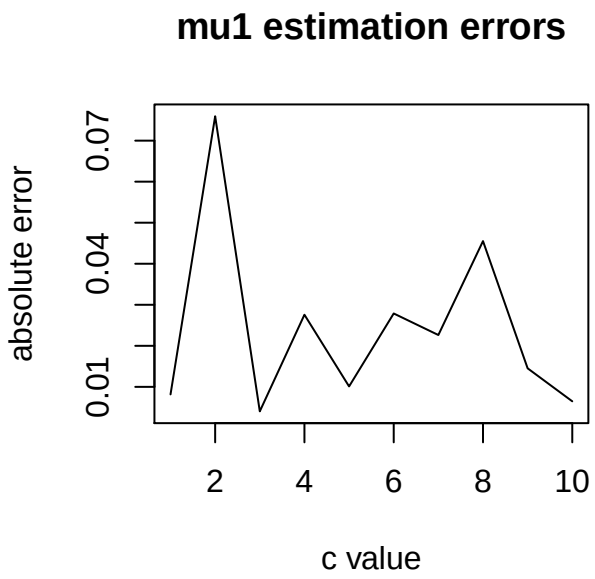
| | Theoretical Values | Importance Sampling Estimates | Estimated ESS |
|---|---|---|---|
| mu1 | 0.667 | 0.683 | 3208.26 |
| mu2 | 5.000 | 4.975 | NA |
| theta | 8.196 | 8.248 | NA |

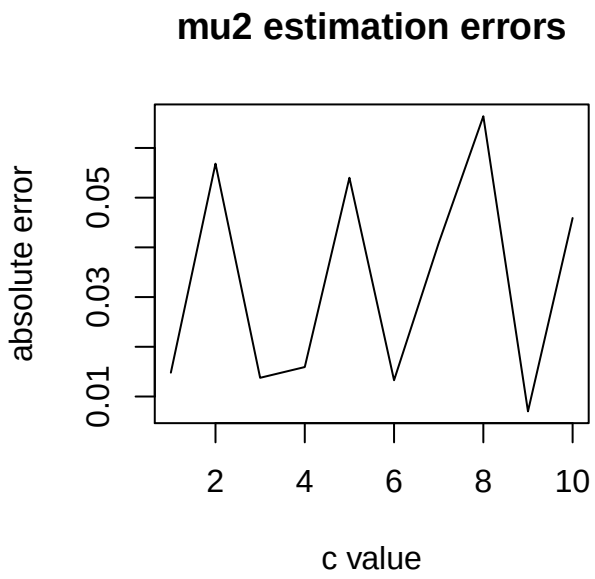
**Part F**

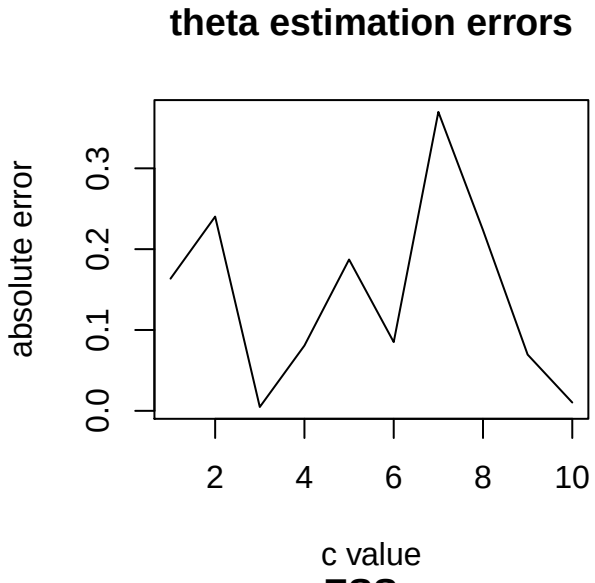
Below is a table summarizing the importance sampling with varying rejection control. The requested plots are on the following page. As the rejection control constant increases, the acceptance rate decreases (i.e. the higher constant, the more strict we are about what we accept). This prefaces what we would expect to see from the ESS. It increases to its maximum value (5000, the number of samples we are directed to pull) as the constant increases (the algorithm is in need of more samples as its restriction grow stronger). Lastly, the error plots show a sort of periodic behavior. I am unsure why this is happening, it is interesting to see that increasing the strength of our sample restrictions doesn't guarantee less error. This might simply be stochastic behavior given our model is already doing well. There is somewhat consistent decline in the error of $\mu_1$ as $c$ increases, so if one is particularly concerned with that estimate I might recommend using an increased $c$ value. Otherwise, it doesn't not appear to be too beneficial and could be left out for simplicity and/or interpretability.

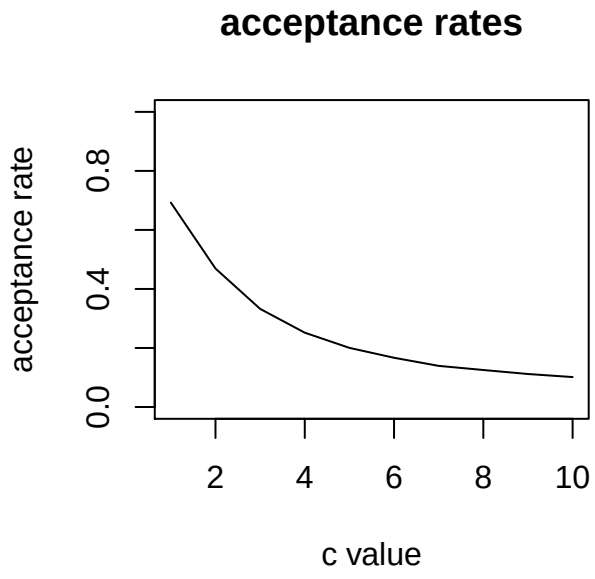
Table 2: Importance Sampling With Varying Rejection Control

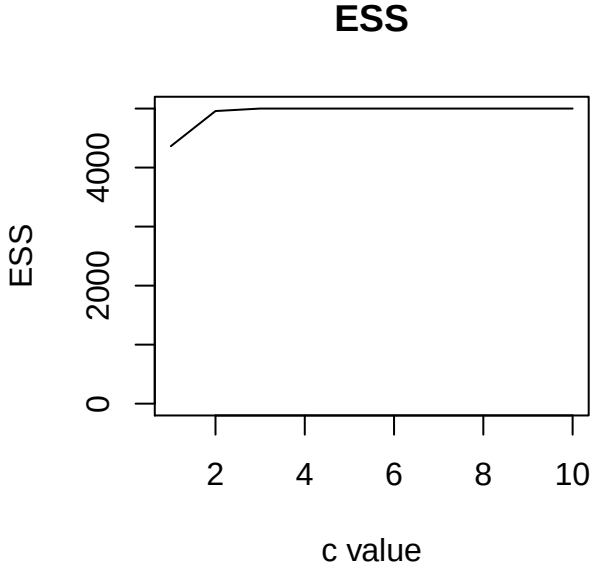
| | c=1 | c=2 | c=3 | c=4 | c=5 | c=6 | c=7 | c=8 | c=9 | c=10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Mu1 Errors | 0.01 | 0.08 | 0.00 | 0.03 | 0.01 | 0.03 | 0.02 | 0.05 | 0.01 | 1e-02 |
| Mu2 Errors | 0.01 | 0.06 | 0.01 | 0.02 | 0.05 | 0.01 | 0.04 | 0.07 | 0.01 | 5e-02 |
| Theta Errors | 0.16 | 0.24 | 0.00 | 0.08 | 0.19 | 0.09 | 0.37 | 0.22 | 0.07 | 1e-02 |
| Acceptance Rates | 0.69 | 0.47 | 0.33 | 0.25 | 0.20 | 0.17 | 0.14 | 0.13 | 0.11 | 1e-01 |
| Estimated ESS | 4362.47 | 4958.04 | 5000.00 | 5000.00 | 5000.00 | 5000.00 | 5000.00 | 5000.00 | 5000.00 | 5e+03 |


mu1 estimation errors
absolute error
0.01
0.04
0.07
2
4
6
8
10
c value


mu2 estimation errors
absolute error
0.01
0.03
0.05
2
4
6
8
10
c value


theta estimation errors
absolute error
0.0
0.1
0.2
0.3
2
4
6
8
10
c value


acceptance rates
acceptance rate
0.0
0.4
0.8
2
4
6
8
10
c value


ESS
ESS
0
2000
4000
2
4
6
8
10
c value

## 2. Chapter 10 Question 5

### Part A

Indeed, we are given the structure of the model from the problem statement. The information we have is as follows, where $J = 10$:

$$p(y_j|\theta_j) \sim \text{Bin}(n_j, \theta_j), \text{ i.e. } p(y|\theta, \alpha, \beta) = \prod_{j=1}^{10} \binom{n_j}{y_i} \theta_j^{y_i} (1-\theta_j)^{n_j - y_i}$$

$$p(\theta_j) \sim \text{logit}^{-1}(\alpha + \beta x_j)$$

$$\alpha \sim t_4(0, 2^2) \text{ and } \beta \sim t_4(0, 1^2)$$

$$x_j \sim U(0, 1) \text{ and } n_j \sim \text{Pois}(5)$$

I build the dataset of 10 samples from this model as directed (Table 1).

Table 3: 10.5 Sampled Data

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| yis | 3 | 2 | 0 | 0 | 1 | 3 | 2 | 1 | 1 | 5 |
| nis | 3 | 3 | 3 | 7 | 6 | 8 | 4 | 8 | 6 | 6 |

### Part B

Next, I used rejection sampling to acquire to get 1000 independent posterior draws from $(\alpha, \beta)$. The likelihood follows:

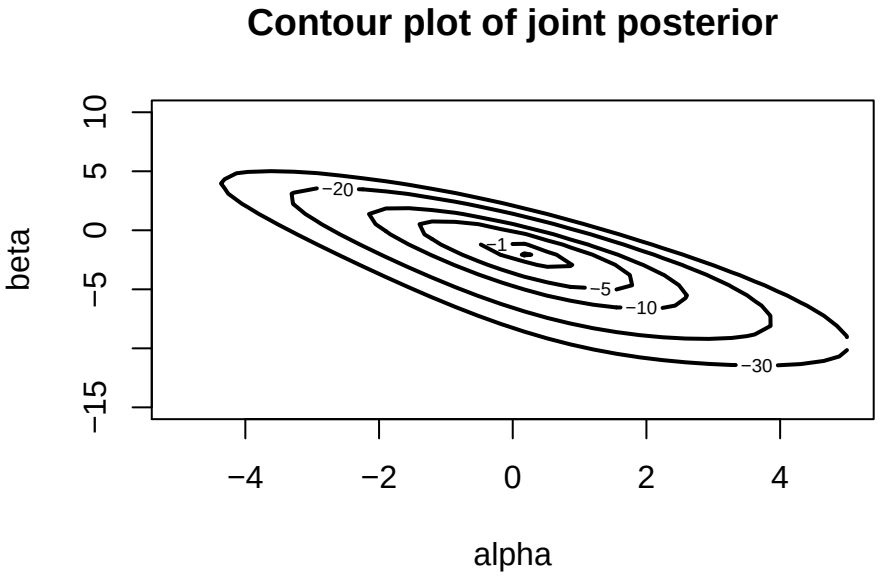
$$p(\alpha, \beta|y, n, x) \propto p(\alpha, \beta) \prod_{i=1}^{k} p(y_i|\alpha, \beta, n_i, x_i)$$

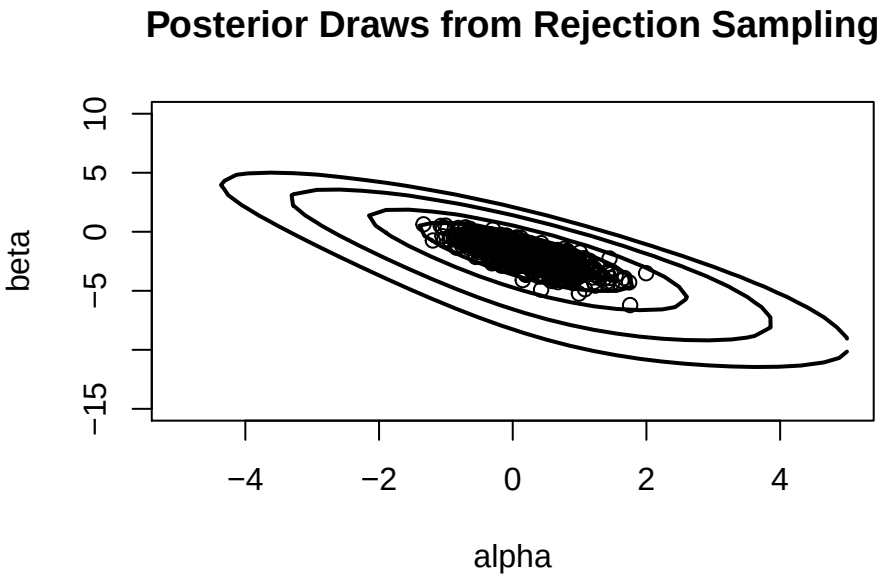
$$p(\alpha, \beta|y, n, x) \propto p(\alpha, \beta) \prod_{i=1}^{k} [\text{logit}^{-1}(\alpha + \beta x_i)]^{y_i} [1 - \text{logit}^{-1}(\alpha + \beta x_i)]^{n_i - y_i}$$

For simplitcity's sake, we will let the prior distribution of $\alpha$ and $\beta$ equal the inverse of the normalization constants (is this what you said, Jerson? I can't quite remember if this was what you recommended for forgoeing the specification of a prior, but I can't think of what else it might have been). Given that we also have no knowledge that would given us reason to choose a particular prior, this is as appropriate a choice as any as it is uninformative. In any case, I derived and calculated the posterior as follows:

$$p(\alpha, \beta|y, n, x) \propto \prod_{i=1}^{k} [\text{logit}^{-1}(\alpha + \beta x_i)]^{y_i} [1 - \text{logit}^{-1}(\alpha + \beta x_i)]^{n_i - y_i}$$

First, I display the contours of the joint posterior distribution of $(\alpha, \beta)$ for your reference. The following plot shows the draws from the rejection sampling. I used Roee's lecture code for this section.




**Part C**

Here we use the Laplace function to estimate the approximate the posterior density for $(\alpha, \beta)$ with a normal centered at the posterior mode with covariance matrix fit to the curvature at the mode. See below for the estimates:

Table 4: Estimated Modes

| alpha | beta |
|---|---|
| 0.1787823 | -1.879348 |

**Part D**

Using the calculated posterior mode and covariance matrix to build two-dimensional $t_4$ distributions, I employed importance sampling to estimate $E(\alpha|y)$ and $E(\beta|y)$. First, see the samples displayed on the

Table 5: Estimated Covariance Matrix

| | alpha | beta |
|---|---|---|
| alpha | 0.9735595 | -1.418346 |
| beta | -1.4183461 | 3.298762 |

contour plot. The table below that displayes the distribution of the estimates for $(\alpha, \beta)$. The means of the 1,000 samples were calculated as 0.21 for $\alpha$ and -2.03 for $\beta$.

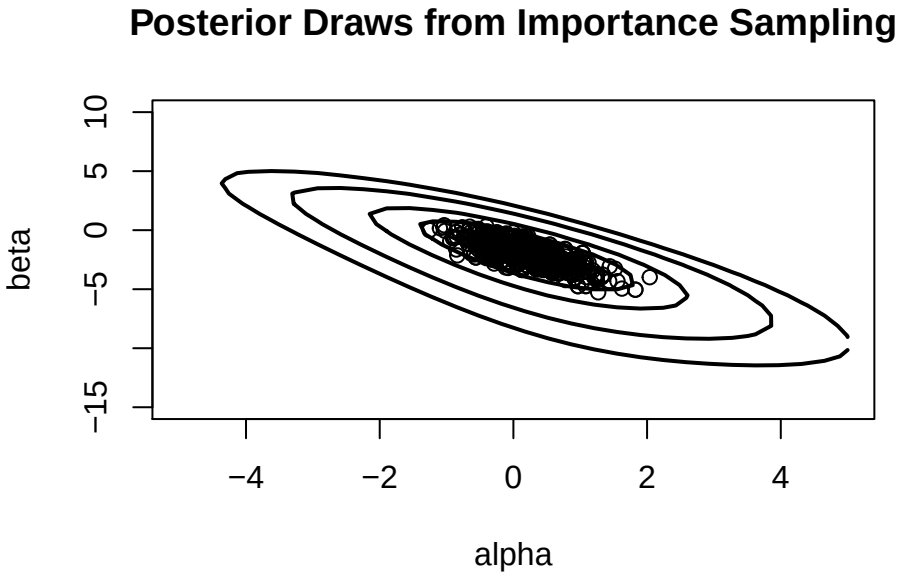



Table 6: 10.5 Estimates for Expectations

| | Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. |
|---|---|---|---|---|---|---|
| alpha | -1.253618 | -0.1307945 | 0.2221591 | 0.2111028 | 0.5273095 | 2.0396251 |
| beta | -5.264367 | -2.6463008 | -2.0114441 | -2.0271028 | -1.4244264 | 0.4191837 |

**Part E**

I calculated the effective sample size to be 918.74 using the following equation from BDA3 (pg266, 10.4):

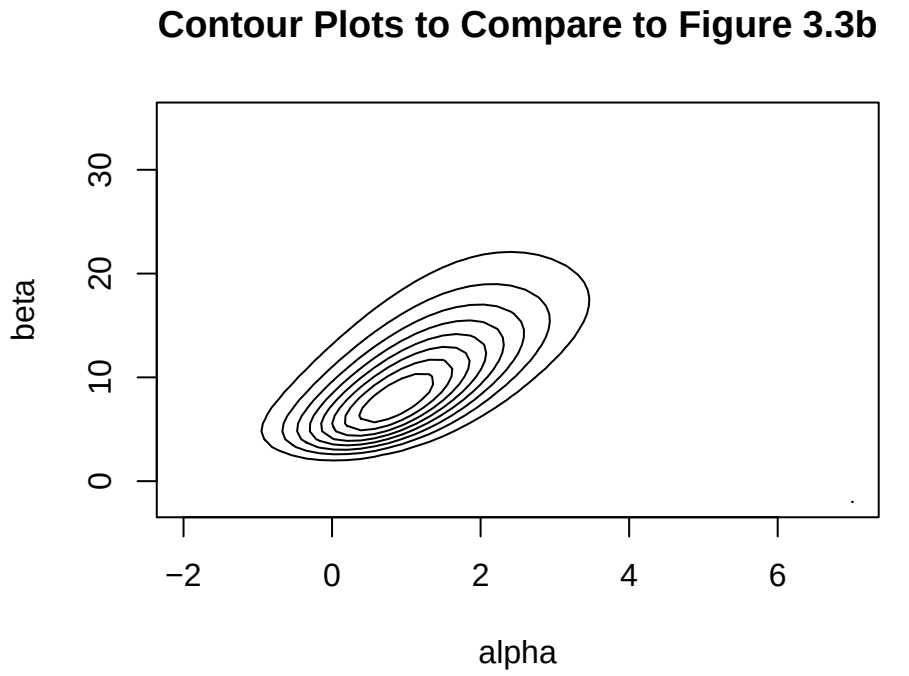
$$S_{eff} = \frac{1}{\sum_{s=1}^{S} (\tilde{w}(\theta^s))^2}$$

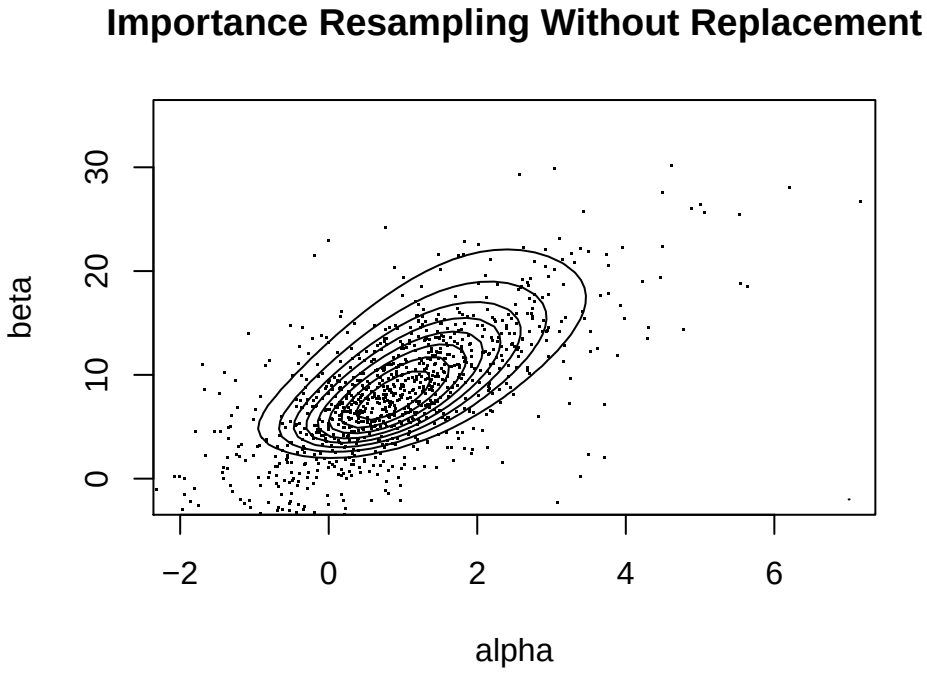
where $\tilde{w}(\theta^s)$ is simply the standardized weights. This is slightly below the 1000 samples we were instructed to pull from our distribution. This makes sense to me after seeing how the samples are near to the modes of the posteriors on the contour plots, but also that the min and max values are a fair distance away from the calculated expectations?

## 3. Chapter 10 Question 8

### Part A

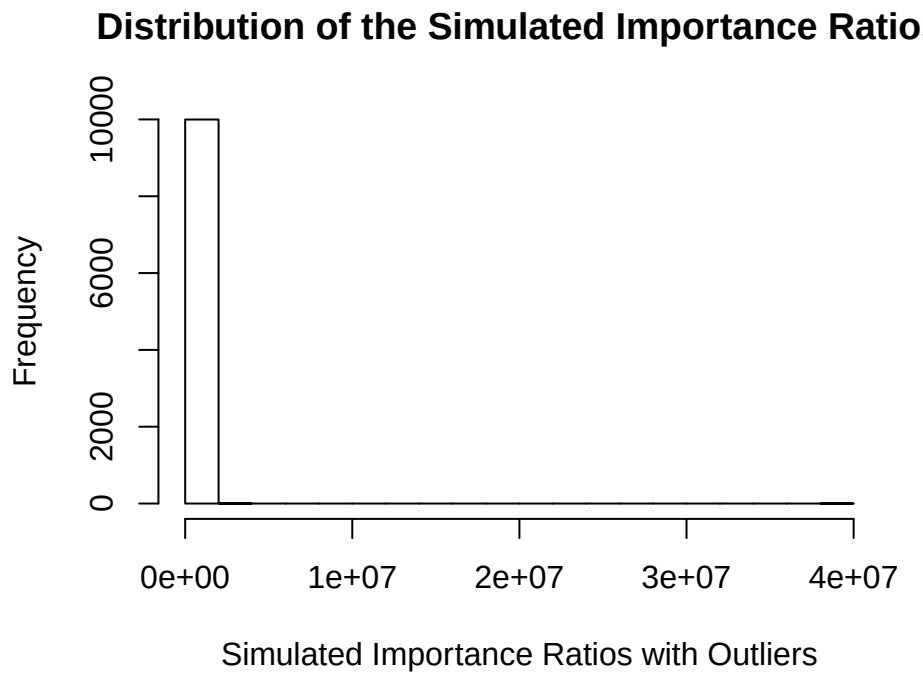
First, I set out to recreate the approximate posterior distribution of the bioassay example in Section 3.7. The model has the same structure as was written out for Problem 10.7. I will forgoe re-writing it here. My first plot below is my recreation of Figure 3.3b to confirm that I am recreating the distribution correctly. After sampling 10,000 from the approximate distribution, I resampled without replacement $k = 1000$ samples. I again based these sampling procedures off of Roee's lecture code. The plot of those draws are shown on the next plot.



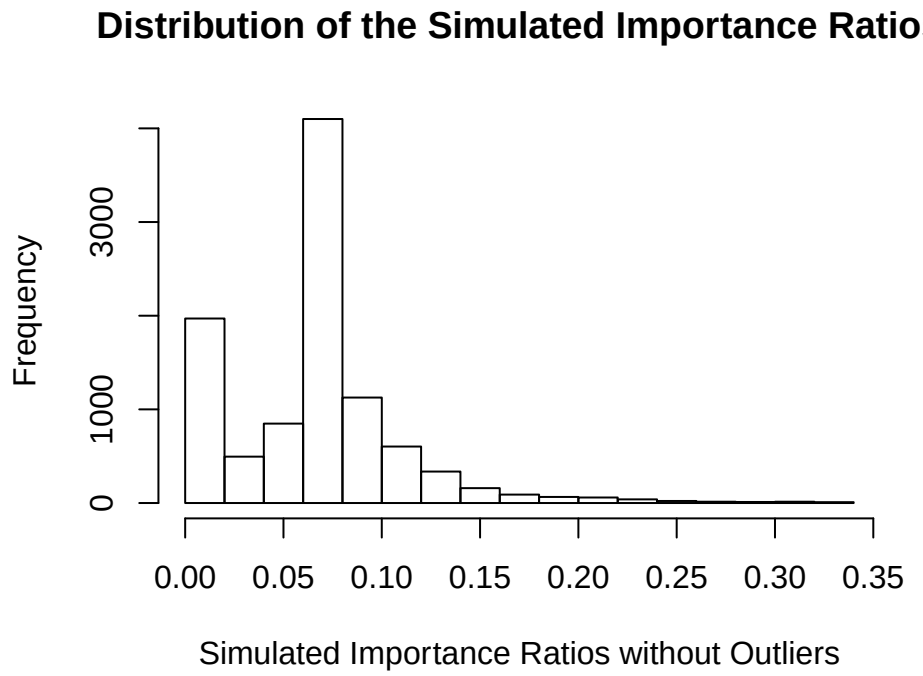



**Part B**

See below for my first attempt at displaying the distribution of the simulated importance ratios.

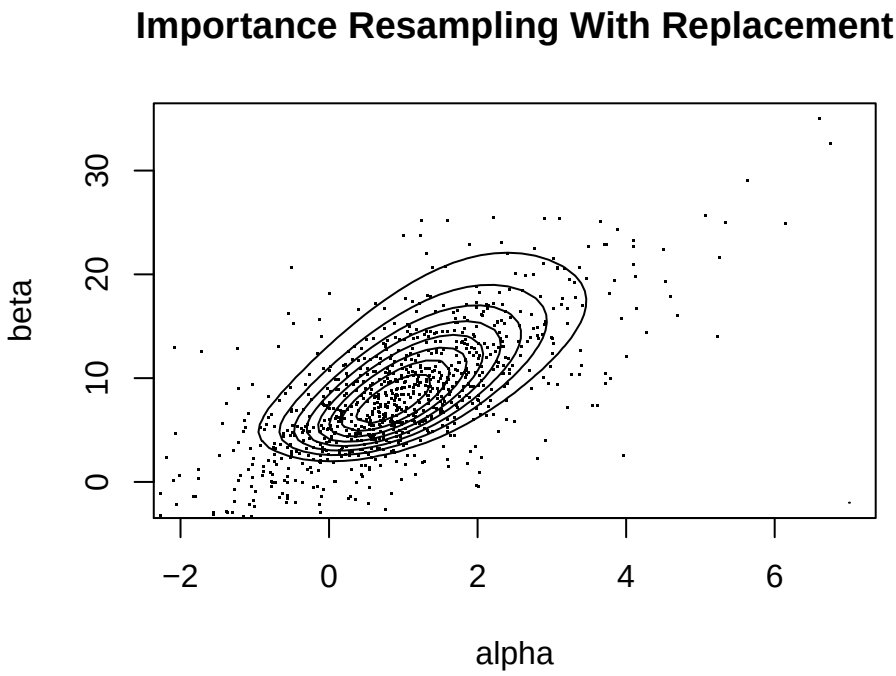



For a more representative display of the simulated importance ratios, I remove multiple outliers from the upper end of the distribution. The resulting distribution is displayed below. The importance ratios appear to vary as we might look for.



**Part C**

Below I display the results from doing importance resampling with replacement. There does not appear to be much difference between the samples drawn when replacing than drawn in Part A where there was no replacing. This is an indicator that the importance weights are moderate, which is in line with what we saw in Part B. In situations like these, it would make sense that sampling with and without replacement gives similar results (as compared to a situation where there are a few extremely large weights and many small weights and sampling with replacement would pick the same few values repeatedly, leading to a different set that a more conservative without replacement approach). Although there were some outliers with larger values, they do not appear to be strong enough to substantially differentiate the sampling procedures' results.



## 4. Chapter 11 Question 2

Following up on questions 10.7 and 10.8, this problem encourages us to try sampling using the the Metropolis algorithm. First, I defined my starting points and my jumping rule. I used a normal approximation to construct estimates for these parameters (Tables 7 and 8).

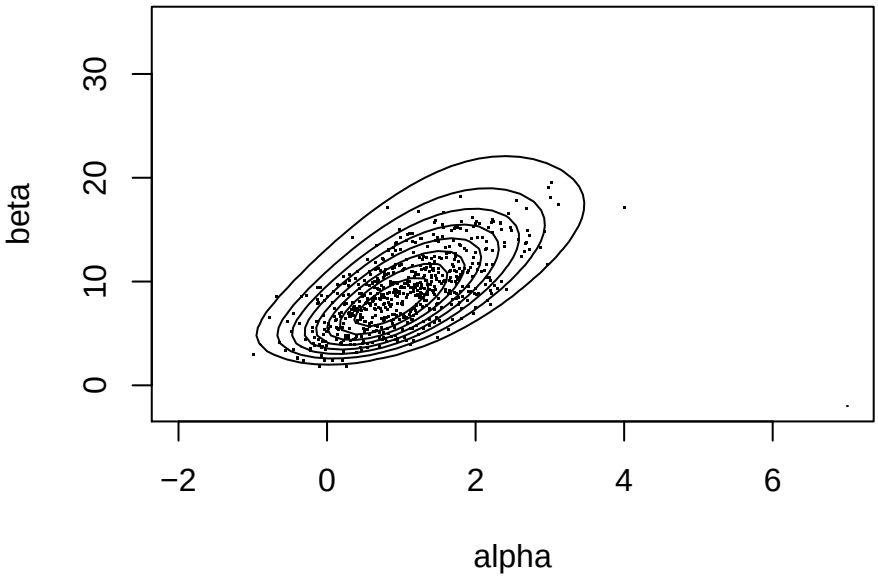
Table 7: Estimated Modes

| alpha | beta |
|---|---|
| 0.847 | 7.749 |

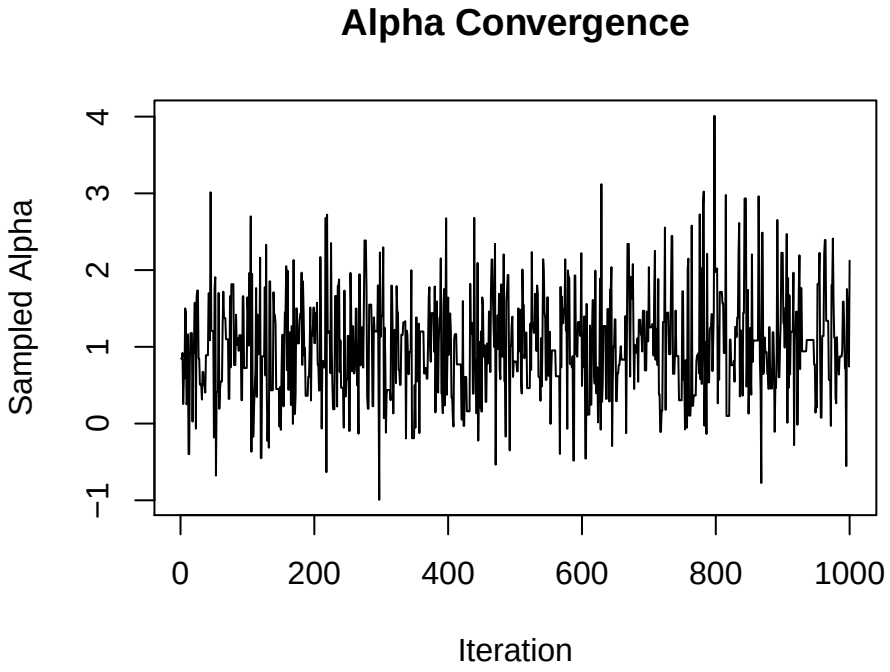
Table 8: Estimated Covariance Matrix

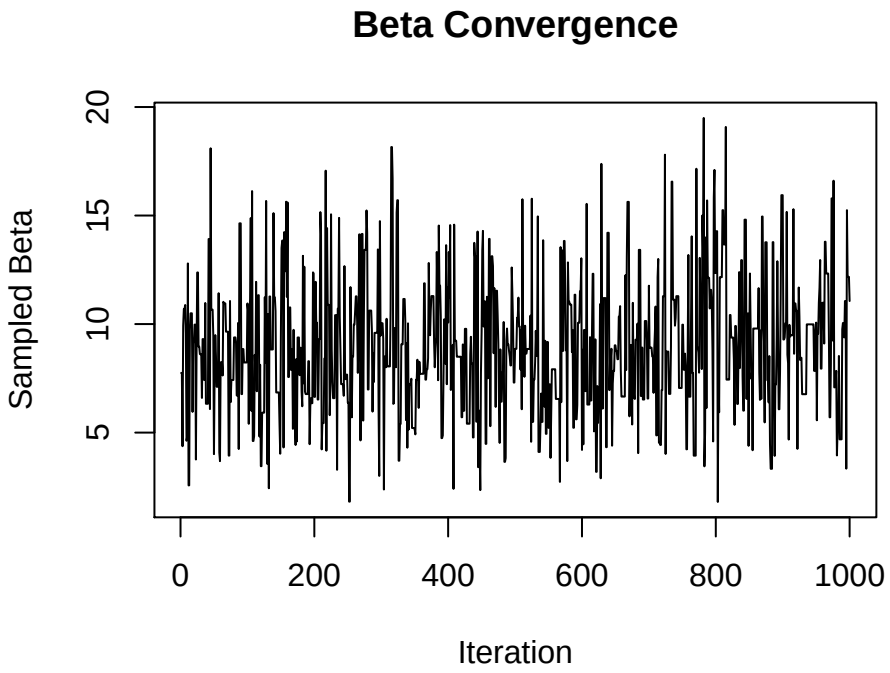
| | alpha | beta |
|---|---|---|
| alpha | 1.042533 | 3.567936 |
| beta | 3.567936 | 23.853039 |

Now, we use the metropolis algorithm to pull a set of samples. I based my code off of Roee's lecture code. These samples are displayed below. They seem to be an improvement over the importance sampling attempts.




Lastly, we are instructed to check if we have approximated convergence. Below I display the values of alpha and beta as we progress through the iterations. They look as we would expect and hope! There are few moments where they seem to get stuck briefly, but they seem to be performing well overall.


Alpha Convergence
Sampled Alpha
−1
0
1
2
3
4
0
200
400
600
800
1000
Iteration


Beta Convergence
Sampled Beta
5
10
15
20
0
200
400
600
800
1000
Iteration

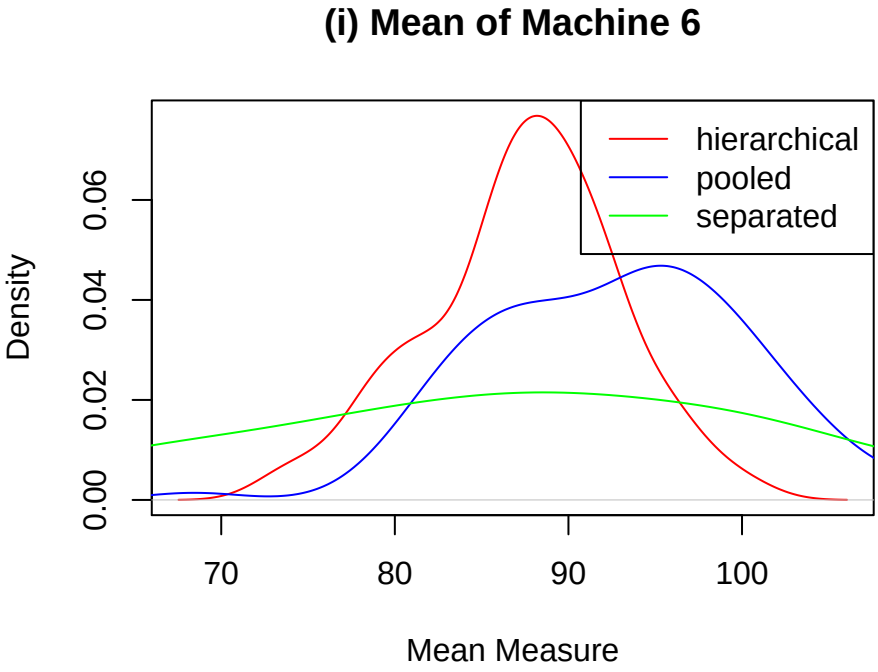
## 5. Chapter 11 Question 3

The question asks us to compare the results from using three types of models on the given data: the hierarchical normal model, the seperate model, and the pooled model. To construct the hierarchical model I used the given conditional densities in the BDA3 text (equations 11.9-11.7). I will not re-write them here to save time, but they are included the code appendix. I based my sampler and functions off of the work posted here: [https://rstudio-pubs-static.s3.amazonaws.com/160665_a78abc64a9b247ffabf2d4547dc75ebb.html]. The distributions of the output from the hierarchical model are displayed below. I see that the estimates are slightly off from what we might hope. I have combed through my code and can't seem to find what might be causing the discrepancy. There were quite a few mistakes made by the original poster that I have already corrected for, and I have updated the code as I worked through my personal understanding. However, there must be something small that I am still, unfortunately, missing.

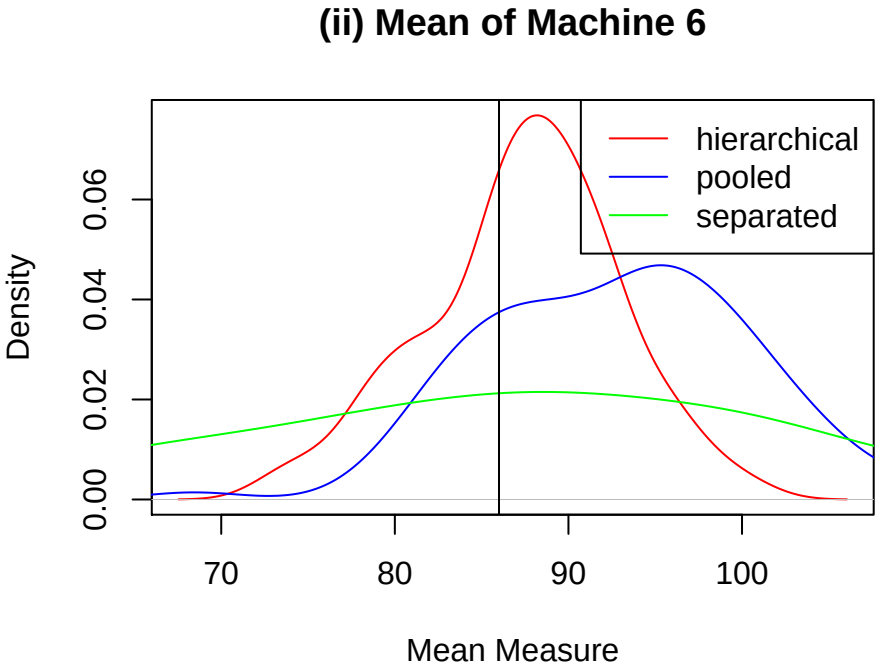
Table 9: Hierarchical Model Estimates

| | 2.5% | 25% | 50% | 75% | 97.5% |
|---|---|---|---|---|---|
| theta1 | 68.226090 | 75.421106 | 80.36012 | 85.48695 | 94.65629 |
| theta2 | 90.198936 | 96.618055 | 101.97768 | 107.69612 | 113.35909 |
| theta3 | 78.833358 | 85.847478 | 89.01924 | 93.62395 | 99.81877 |
| theta4 | 94.264990 | 102.451558 | 107.14791 | 111.43417 | 118.46376 |
| theta5 | 79.180597 | 86.839922 | 90.26511 | 94.61633 | 98.87628 |
| theta6 | 75.661817 | 84.534089 | 87.73719 | 91.37322 | 97.42413 |
| mu | 83.438952 | 88.912486 | 92.94220 | 96.81082 | 105.40103 |
| sigma2 | 10.950300 | 12.857649 | 14.62866 | 16.10260 | 19.45944 |
| tau2 | 4.626044 | 8.358545 | 12.10177 | 16.14303 | 30.72942 |

First, they ask us to report (i) the posterior distribution of the mean of the quality measurements of the sixth machine for each of the three models. I referred to the construction of the hierarchical distribution above. I drew from this model by drawing samples from: $N(\theta_6, |\sigma^2)$. To build the "seperate" model, I simply calculated the mean and variance of the sixth machine measurements without taking into account any of the other machine values and then used them as the parameters of the normal distribution, i.e. $N(\bar{y_{6.}}, V_6^{sep})$. Finally, to build the "pooled" model, I simply calculated the mean across all of the samples and the variance of the sixth machine measurements compared to the overall mean, i.e. $N(\bar{y_{..}}, V_6^{pooled})$. These distributions are displayed below. Notably, they appear as we might expect: the hierarchical has the highest density around the sampled mean of the sixth machine, the pooled distribution is pulled higher by the other machine values, and the seperated has the widest spread.




Next, they ask us to report (ii) the predictive distribution for another quality measurement of the sixth machine. I was a bit unsure what they were asking for here, but I decided to simply compare how each of the three models are performing by calculating p-values for a text statistic of the mean of the sixth machine. The observed values is displayed with a black line on the plot below. The p-values are displayed below the graph for your reference. They suggest that the hierarchical model and separate model will predict the mean of the sixth machine best.

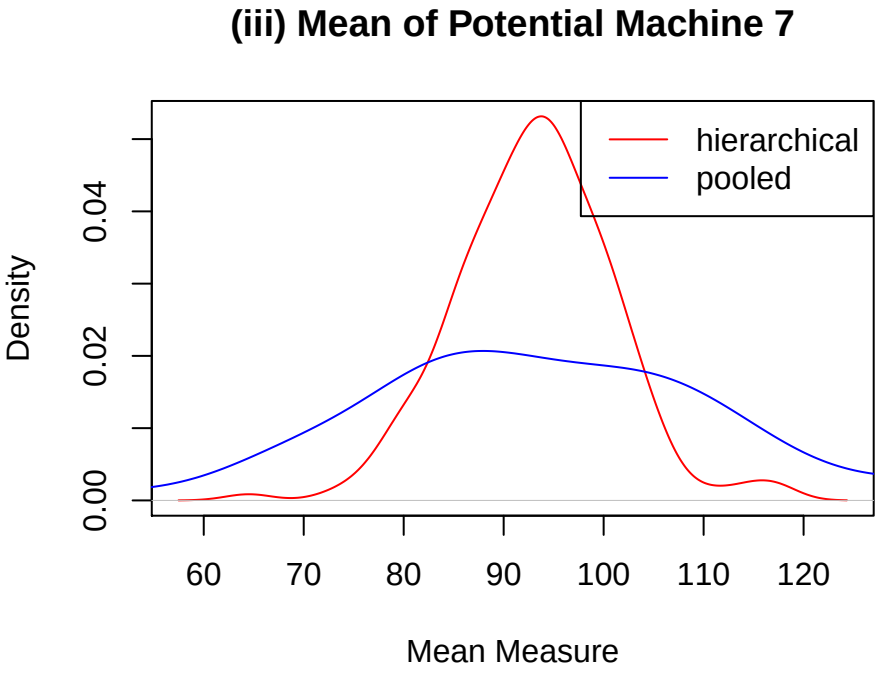



Lastly, they ask us to report (iii) the posterior distribution of the mean of the quality measurements of the seventh machine. As we are given no information about the seventh machine, such as if it shares any characteristics with one of the sixth machines in our dataset, we take a cautious approach and create

Table 10: Posterior Predictive Check

| | Hierarchical | Pooled | Seperate |
|---|---|---|---|
| P-Value | 0.66 | 0.8 | 0.57 |

distributions using the grand means of the distributions. In practice, this means we pull from the following distributions: Hierarchical: $N(\mu, \tau)$ and Pooled: $N(\bar{y}_{..}, V_{pooled})$. We would not be able use the Seperate model for this prediction as the seperate model only claims to provide information about the six machines in our data set. The distributions of the Hierarchical and Pooled models are displayed below. We can see that the hierachical model put the majority of its predictions in a much narrower interval than the pooled, as we would expect.



## 6. Chapter 11 Question 4

The problem statement directs us to allow different $\sigma_j$ values for each machine in our hierarchical model. This means that we need to update out distribution for $\sigma_j$. They direct us to use a $Inv-\chi^2$ distribution with fixed degrees freedom and unknown scale $\sigma_0$. Therefore,

$$p(\sigma_j) \sim Inv-\chi^2(v_0, \sigma_0^2)$$

Now we need to derive its conditional density. The only terms in the posterior that include $\sigma_j$ are now the following:

$$p(\sigma_j^2|\theta, \mu, \log\tau, \sigma_0^2, y) \propto \chi^{-2}(\sigma_j^2|v_0, \sigma_0^2) \prod_{i=1}^{n} N(y_{ij}|\theta_j, \sigma_j)$$

$$p(\sigma_j^2|\theta, ...) \propto (\sigma_j^2)^{-\frac{v_0}{2}-1} \exp\left\{-\frac{v_0\sigma_0^2}{2\sigma_j^2}\right\} (\sigma_j^2)^{\frac{-n}{2}} \exp\left\{\frac{-\frac{1}{n}\sum_{i=1}^{n}(y_{ij}-\theta_j)^2}{2\sigma_j^2}\right\}$$

$$p(\sigma_j^2|\theta, ...) \propto (\sigma_j^2)^{-\frac{v_0-n}{2}-1} \exp\left\{-\frac{v_0\sigma_0^2}{2\sigma_j^2} - \frac{\frac{1}{n}\sum_{i=1}^{n}(y_{ij}-\theta_j)^2}{2\sigma_j^2}\right\}$$

And we notice that this is sneakily in the form of an inverse chi-squared distribution that we can write as follows:

$$p(\sigma_j^2|\theta, ...) \propto \chi^{-2}(v_k, \sigma_k^2)$$

where

$$v_k = v_0 + n$$

and

$$\sigma_k^2 = \frac{v_0\sigma_0^2 + \sum_{i=1}^{n}(y_{ij}-\theta_j)^2}{v_0+n}$$

From here, I let $v_0 = 1$ for simplicity's sake (they tell us to set a value). Our last task is to deal with $\sigma_0^2$. They tell us that the conditional distribution of $\sigma_0^2$ is "not of simple form." With this is mind, I opted to pull potential $\sigma_0^2$ values, creating a grid from 0.25 to 30 in steps of 0.25. From here, I calculated all of the corresponding $\sigma_k^2$ values and distribution of probabilities created by evaluating $p(\sigma_j^2|\theta, ...) \propto \chi^{-2}(v_k, \sigma_k^2)$ for each $\sigma_k^2$ option. I then finally used these probabilities to sample a conditional $\sigma_j^2$ value to return as part of the Gibbs sequence.

As you can see below, the values are slightly off from what we might hope and what I saw when I discussed this question with you, Jerson. I am encouraged by the fact that they are close and that there is variability across the six machines. However, I wonder if some of this more eradict behavior is because I don't derive a specific form for $\sigma_0^2$. I was interpreting the question to mean that we should not, but potentially the better option is to pull probabilities to use to sample across the conditional distribution of $\sigma_0^2$. With more time I would have liked to explore this more fully and try out variable priors on $\sigma_0^2$. I hope that what I have created below can demonstrate that I have indeed allowed the $\sigma_j^2$'s to vary, although with potentially more stochasticity than would be ideal.

Table 11: Hierarchical Model Estimates

| | 2.5% | 25% | 50% | 75% | 97.5% |
|---|---|---|---|---|---|
| theta1 | 69.4854983 | 75.511231 | 78.019289 | 81.451991 | 98.26266 |
| theta2 | 88.8713265 | 99.417857 | 105.081280 | 107.507207 | 114.51133 |
| theta3 | 81.5725321 | 86.463253 | 87.678462 | 89.515365 | 95.37400 |
| theta4 | 86.9815204 | 103.115189 | 107.853633 | 110.860487 | 117.38666 |
| theta5 | 81.4971090 | 89.123822 | 90.163118 | 91.353563 | 96.31472 |
| theta6 | 79.2377252 | 84.542918 | 85.880929 | 88.601123 | 92.85275 |
| sigma1 | 1.3894805 | 7.217019 | 12.391173 | 18.608880 | 35.84393 |
| sigma2 | 1.2823696 | 6.540877 | 11.083932 | 16.156223 | 36.89614 |
| sigma3 | 0.3673566 | 3.112010 | 5.451076 | 9.066421 | 22.50215 |
| sigma4 | 0.7050181 | 8.414874 | 12.867193 | 20.401493 | 36.91912 |
| sigma5 | 0.4482310 | 2.436907 | 4.302000 | 7.230269 | 15.87121 |
| sigma6 | 0.4533744 | 3.259517 | 6.548145 | 9.845186 | 23.15424 |
| tau2 | 6.3056039 | 9.907665 | 12.484024 | 15.813745 | 22.53850 |
| mu | 77.4652987 | 87.688743 | 91.466161 | 95.407433 | 104.81319 |

## 7. Chapter 13 Question 5

### Part A

Recall that we are given the following:

$$x_i \sim Poisson(\lambda_i)$$
$$\lambda \sim Gamma(\alpha, \beta)$$
$$p(\alpha, \beta) \propto 1$$

Therefore, we can start by writing the joint density as follows:

$$p(x_i, \lambda_i) = \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda_i^{\alpha-1} e^{-\beta\lambda_i} \frac{e^{-\lambda_i} \lambda_i^{x_i}}{x_i!}$$

We can integrate over $\lambda_i$ to find the unconditional distribution of $x_i$.

$$p(x_i) = \int_0^\infty \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda_i^{\alpha-1} e^{-\beta\lambda_i} \frac{e^{-\lambda_i} \lambda_i^{x_i}}{x_i!} d\lambda_i$$

$$p(x_i) = \frac{\beta^\alpha}{x_i!\Gamma(\alpha)} \int_0^\infty \lambda_i^{x_i+\alpha-1} e^{-(\beta+1)\lambda_i} d\lambda_i$$

$$p(x_i) = \frac{\beta^\alpha}{x_i!\Gamma(\alpha)} \frac{\Gamma(x_i+\alpha)}{(\beta+1)^{x_i+\alpha}} \int_0^\infty \frac{(\beta+1)^{x_i+\alpha}}{\Gamma(x_i+\alpha)} \lambda_i^{x_i+\alpha-1} e^{-(\beta+1)\lambda_i} d\lambda_i$$

We have contructed the above expression such that the inegrand is of the form of the PDF of the Gamma distribution, so the integral, in fact, simply equals 1. Now,

$$p(x_i) = \frac{\beta^\alpha}{x_i!\Gamma(\alpha)} \frac{\Gamma(x_i+\alpha)}{(\beta+1)^{x_i+\alpha}}$$

$$p(x_i) = \frac{\Gamma(x_i+\alpha)}{x_i!\Gamma(\alpha)} \left(\frac{\beta}{\beta+1}\right)^\alpha \left(\frac{1}{\beta+1}\right)^{x_i}$$

We note that this is a form of the negative binomial distribution!

If we transform $\alpha$ and $\beta$ such that $p = 1/(\alpha+1)$ and $r = \beta$, we reach the more recognizable form of :

$$p(x_i) = \frac{\Gamma(x_i + r)}{x_i!\Gamma(r)}(1-p)^r(p)^{x_i}$$

**Part B**

We are given that $\sum_{i=1}^{k} y_i = N$. Therefore, we use the general form of the multinomial to write:

$$p(y_i) = \frac{N!}{x_1!x_2!...x_k!}\prod_{i=1}^{k} p(x_i)^{y_i}$$

Which can also be written as follows using the gamma function:

$$p(y_i) = \frac{\Gamma(\sum_i x_i + 1)}{\prod_i \Gamma(x_i+1)}\prod_{i=1}^{k} p(x_i)^{y_i}$$

**Part C**

We can write the likelihood of $y$ as:

$$L(\alpha, \beta, y_k) \propto \prod_{k=1}^{24}\left[\frac{\Gamma(\alpha + x_k)}{x_k!\Gamma(\alpha)}\frac{\beta^\alpha}{(\beta+1)^{\alpha+x_k}}\right]^{y_k}$$

$$L(\alpha, \beta, y_k) \propto \left[\frac{\Gamma(\alpha+1)}{1!\Gamma(\alpha)}\frac{\beta^\alpha}{(\beta+1)^{\alpha+1}}\right]^{118}\left[\frac{\Gamma(\alpha+2)}{2!\Gamma(\alpha)}\frac{\beta^\alpha}{(\beta+1)^{\alpha+2}}\right]^{74}...\left[\frac{\Gamma(\alpha+24)}{24!\Gamma(\alpha)}\frac{\beta^\alpha}{(\beta+1)^{\alpha+24}}\right]^{3}$$

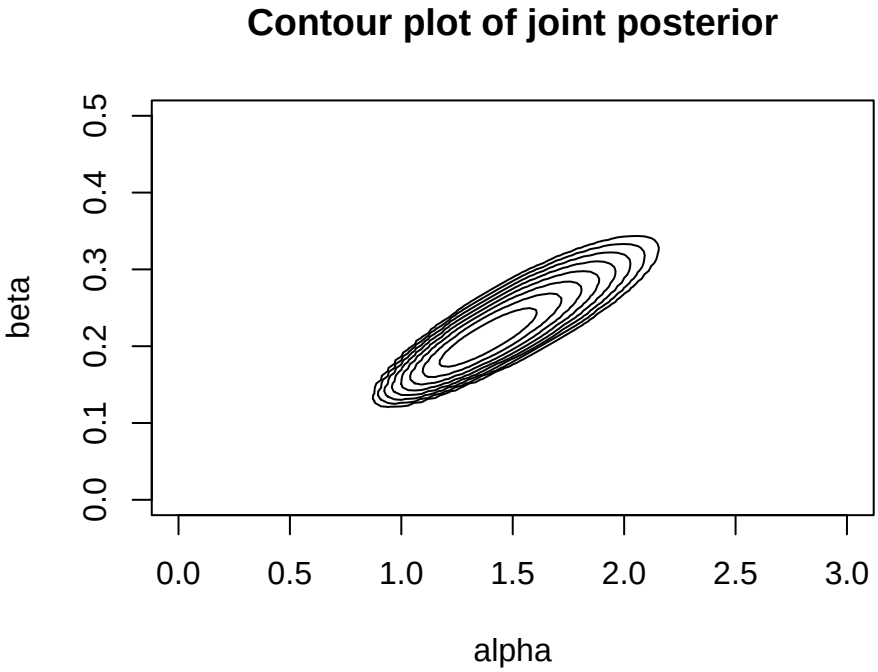
Now, to find the mode of $(\alpha, \beta)$, I will use a normal approximation with the "laplace" function. See the output below and the contour plot of the joint density that aligns with the estimates.

Table 12: Estimated Modes

| alpha | beta |
|---|---|
| 1.372831 | 0.2084905 |

Table 13: Estimated Covariance Matrix

| | alpha | beta |
|---|---|---|
| alpha | 0.0102788 | 0.0015610 |
| beta | 0.0015610 | 0.0003142 |

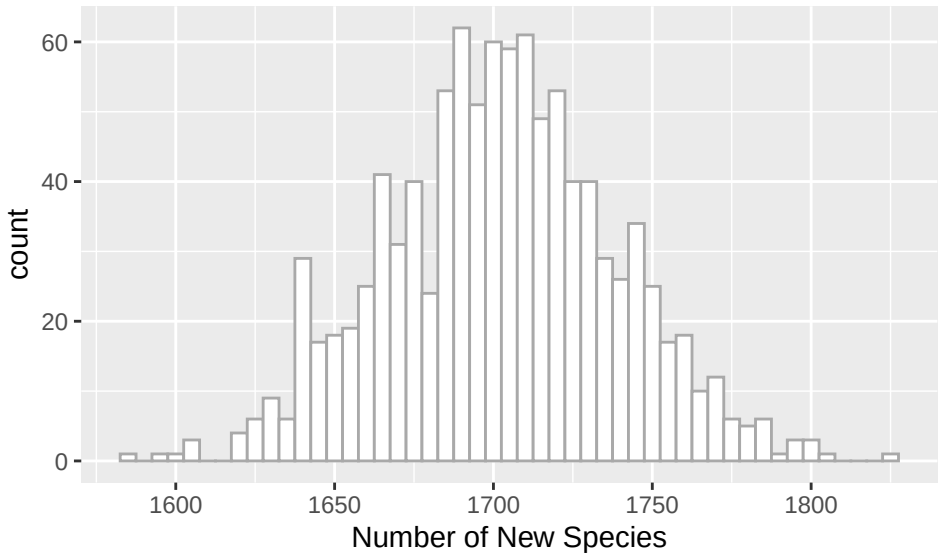



**Part D**

For this section I wrote a loop that first pulled alpha and beta values based on the joint probability distribution displayed above. Next, I used these values to pull a value from the Zero-Truncated Negative Binomial Distribution (function: rztnbinom), given that we are instructed to disregard "unseen species." From here, I update the number of animals seen and store the pulled value (i.e. the number of times a species was seen). From here, I repeat until the number of animals seens reaches 10,000. Lastly, I calculate and store the number of species that was seen over the course of seeing these 10,000 animals. I repeat this 1,000 times to produce a posterior distribution. You can see the derived estimate and 95% posterior interval in the table below, along with a histogram.

Table 14: Estimate and Posterior Interval

| 2.5% | 50% | 97.5% |
|---|---|---|
| 1633.95 | 1702.16 | 1773.02 |



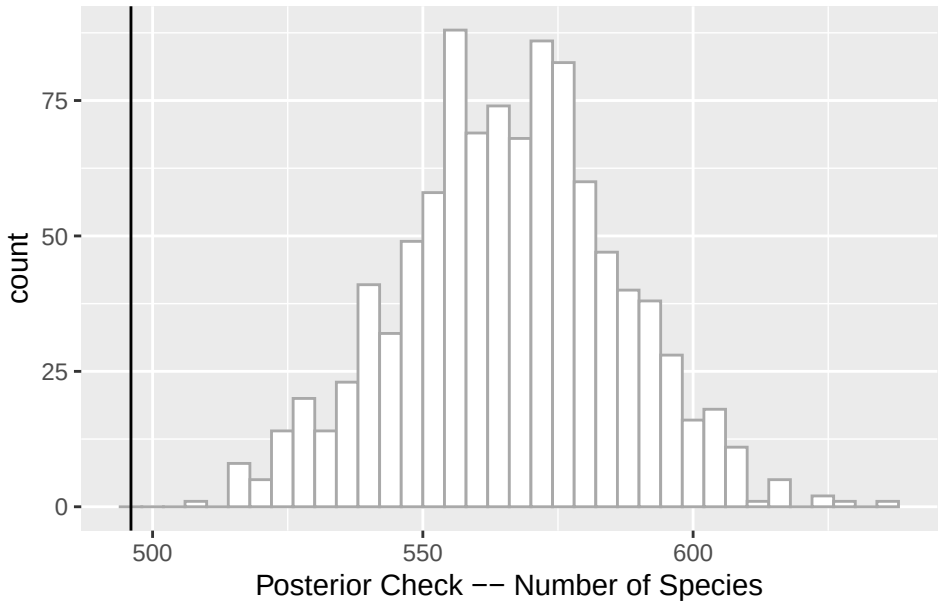

**Part E**

For this section I will evaluate the fit of the model using three test statistics: (1) the total number of species observed, (2) the mean number times an animal is seen, and (3) the maximum (the largest number of times a species was seen). This is a bit tricky however, given the observed data we are given only has 3,326 animals caught. I am a bit unsure what the most appropriate course of action here might be. I have opted to build another data set in the same fashion as Part D, but only running the while loop until it hits 3,326, rather than 10,000. This will allow me to directed compare the number of species seen to the 496 species observed. Additionally, I will store all of the "means" and "maxima" and will compare this to the observed mean of 20.67 and observed maximum of 24. See below for the information about the spread, the p-values, and the histograms for each of the test statistics.

First, I see that my model appears to be overestimating the number of total species observed. Although not completely out of the ballpark, the model seems to be consistently predicting that we will see 50 more species than then data would let us believe.

Table 15: Posterior Check – Number of Species

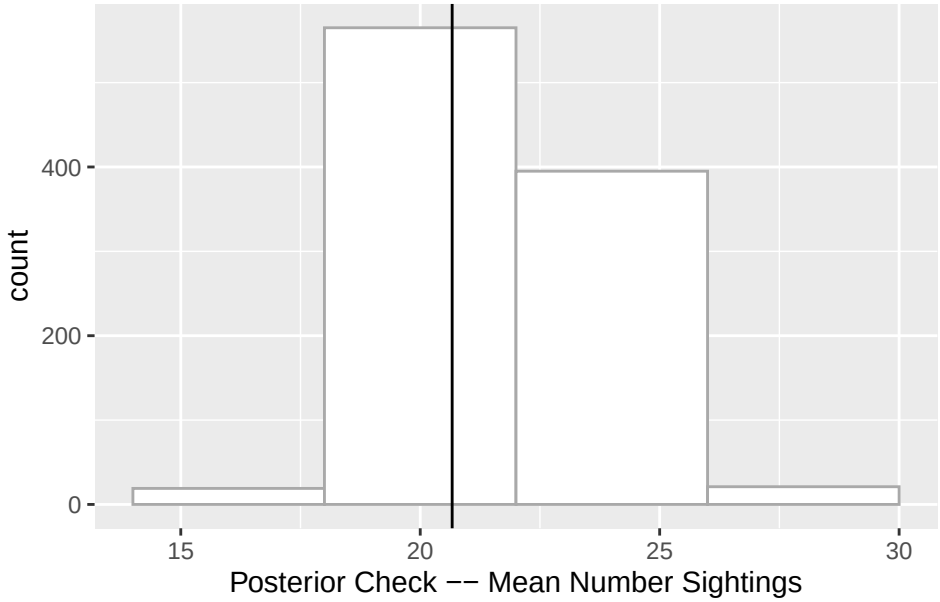
| Observed | 2.5% | 50% | 97.5% | Pval |
|---|---|---|---|---|
| 496 | 525.97 | 566.54 | 605 | 1 |




Next, the model does appear to do a good job representing the mean number of total sightings

Table 16: Posterior Check – Mean Number Sightings

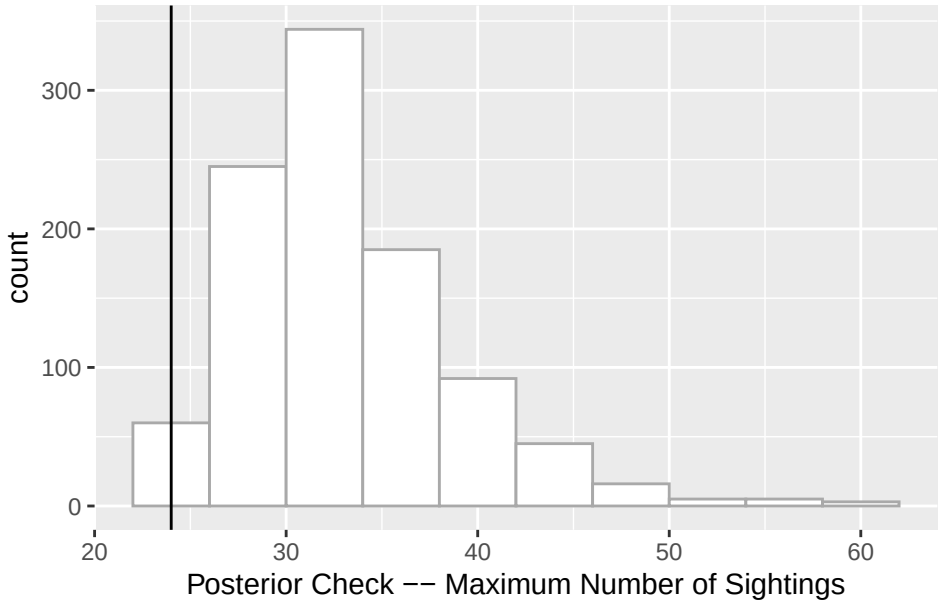
| Observed | 2.5% | 50% | 97.5% | Pval |
|---|---|---|---|---|
| 20.67 | 18.21 | 21.73 | 25.63 | 0.69 |




Lastly, the model again overestimates the maximum number of sightings. This is in line with the overestimation of the first predictive check, the "tail" of the y values appears to be longer than we think it should be.

Table 17: Posterior Check – Maximum Number of Sightings

| Observed | 2.5% | 50% | 97.5% | Pval |
|---|---|---|---|---|
| 24 | 25 | 33.59 | 47 | 0.99 |

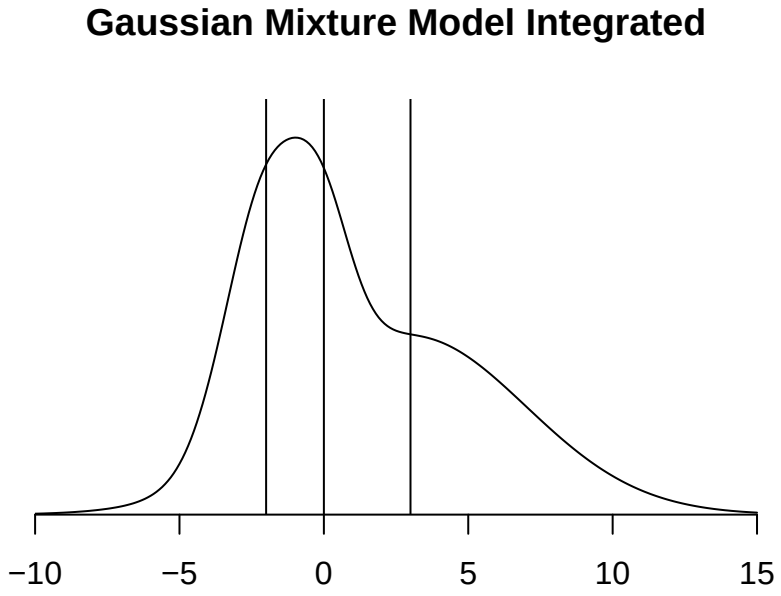


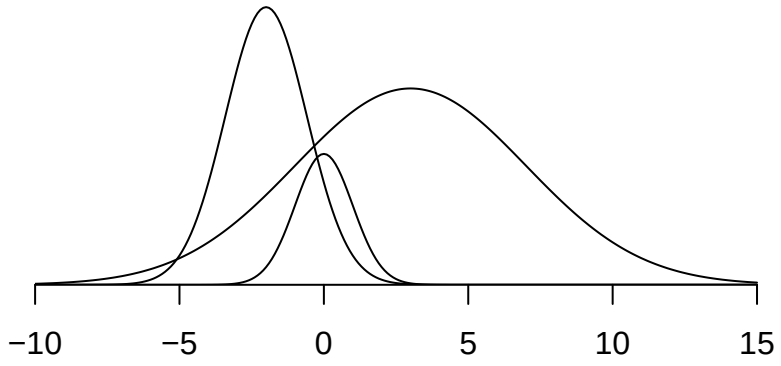

**Part F**

I would predict that the model would be sensitive to the assumption that each of these individual observations are independent. In reality, different locations have different densities of species and species are dependent on one another. I believe that reslaxing the assumption that all of these observations are independent would have a strong effect.

## 8. Generate / simulate n = 120 observations from a three component mixture

We are instructed to simulate n = 120 observations from a three component mixture Gaussian model where $(p_1, p_2, p_3) = (0.1, 0.3, 0.6)$ and $(\mu_1, \sigma_1^2) = (0, 1)$, $(\mu_2, \sigma_2^2) = (-2, 2)$, and $(\mu_3, \sigma_3^2) = (3, 16)$. See below for the mixture model's distribution where the three means are displayed with black lines. I have also built a figure that displays the three normal distributions seperately as this will come in handy when we are comparing the output from our EM and MCMC algorithms.



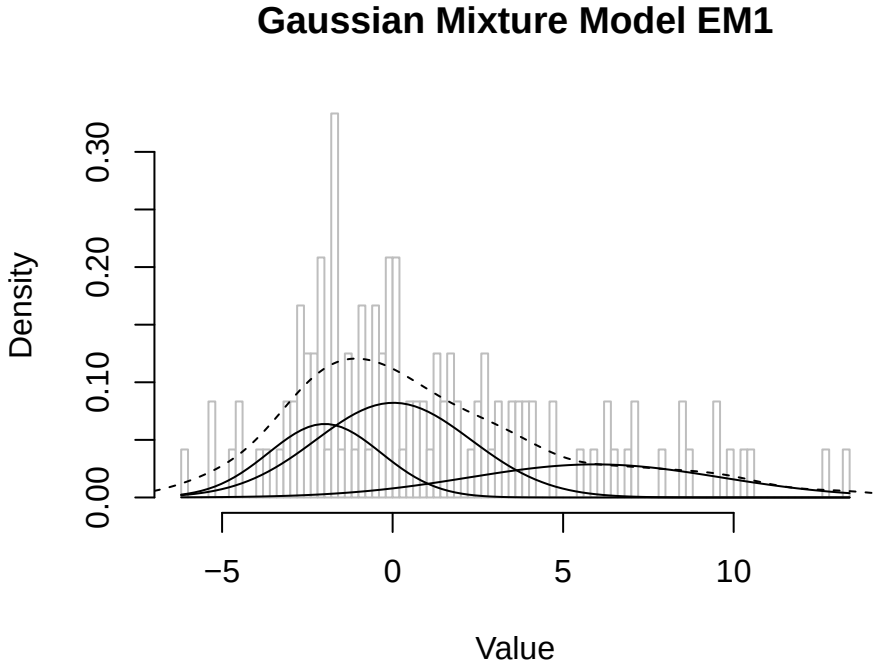



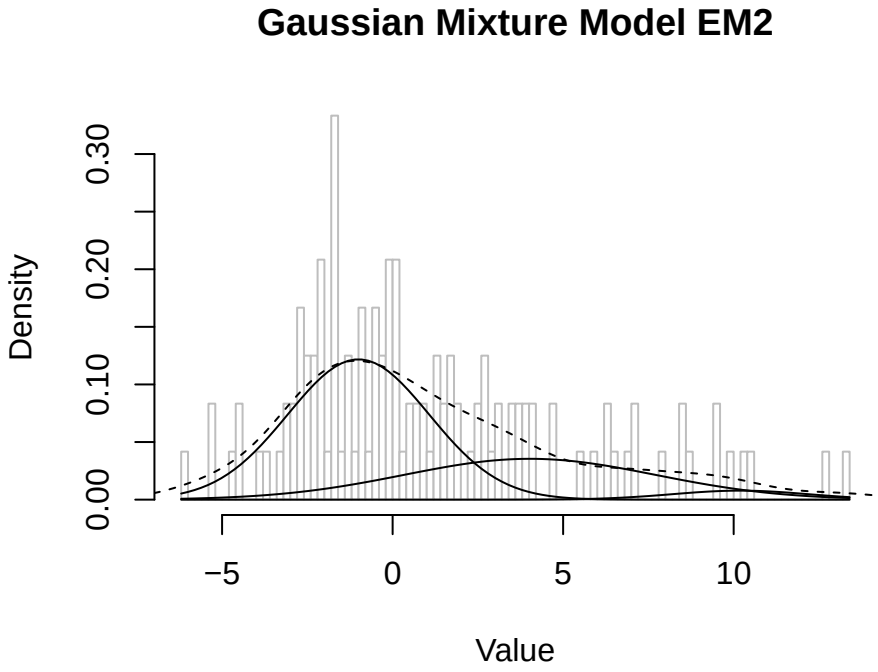

**Part A**

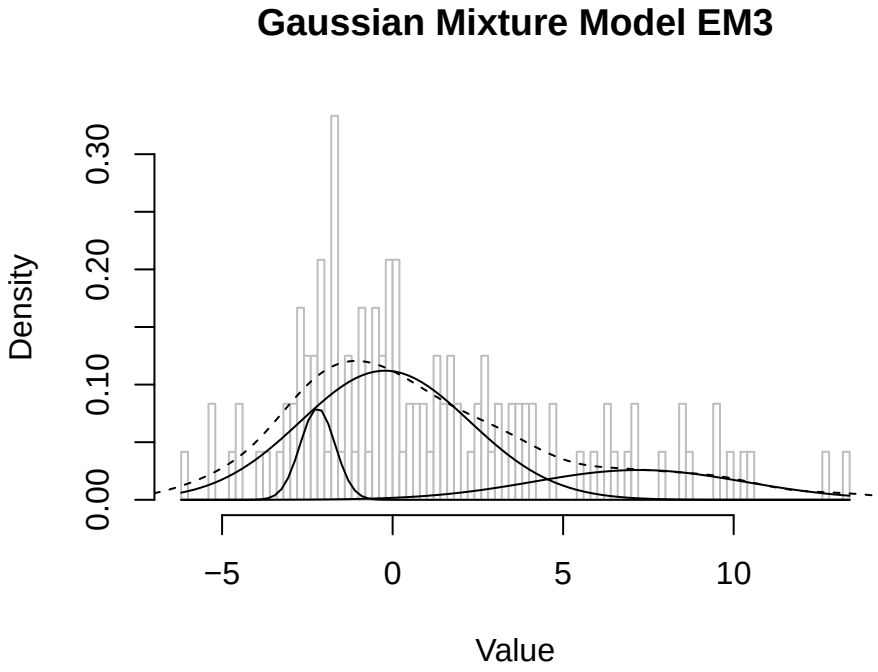
First, we implement an EM algorithm to find the MLEs for $p_i, \mu_i$, and $\sigma_i^2$ for $i \in \{1, 2, 3\}$. I used the package "mixtools" to implement the EM algorithm. Three different outputs of this algorithm are displayed below, first in a table to compare the three sets of values, followed by figures for a visual display. They obviously fit the data well but do not necessarily end up at the values that we used to build the mixture model. There is also some variability at where each of the models end up–the reason I decided to display and compare three different rounds. See Part C for further discussion.

Table 18: EM Algorithm MLEs

| True Values | | | Model 1 | | | Model 2 | | | Model 3 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| mu | sigma^2 | p | mu | sigma^2 | p | mu | sigma^2 | p | mu | sigma^2 | p |
| 0 | 1 | 0.1 | -1.99 | 1.65 | 0.26 | -0.88 | 2.15 | 0.68 | -2.20 | 0.51 | 0.1 |
| -2 | 2 | 0.3 | 0.03 | 2.28 | 0.47 | 4.93 | 3.95 | 0.26 | -0.22 | 2.48 | 0.7 |
| 3 | 16 | 0.6 | 5.94 | 3.69 | 0.27 | 5.95 | 3.78 | 0.06 | 7.22 | 3.10 | 0.2 |



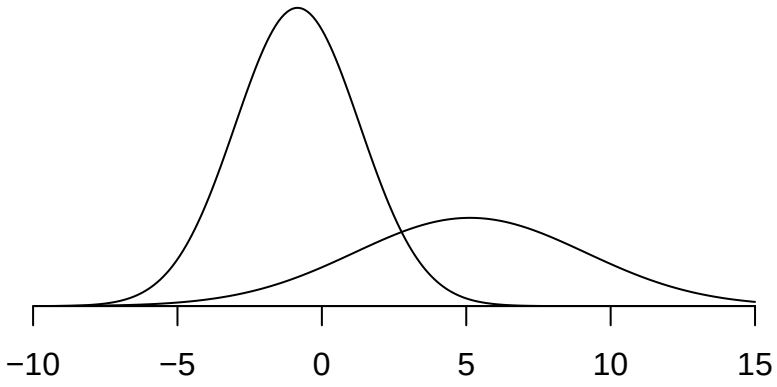



**Part B**

Next, we implement an the MCMC Gibbs algorithm to find the posterior distributions for $p_i, \mu_i$, and $\sigma_i^2$ for $i \in \{1, 2, 3\}$. I used a package called "bayesmix" that makes use of the JAGS to run MCMC Gibbs on various types of Gaussian mixture models. I used "proper diffused priors" as instructed. The results are displayed below (I only show one chain because my personal runs never varied). See Part C for discussion.

Table 19: Gibbs Algorithm Estimates

| mu | sigma^2 | p |
|---|---|---|
| -8.435756e-01 | 4.666895 | 0.6406271 |
| 5.131333e+00 | 16.022501 | 0.3512224 |
| 2.300376e+06 | Inf | 0.0081505 |



**Part C**

Summarize the results and compare between the two approaches, any problems observed during the sampling or maximum likelihood estimator. This was a very interesting problem. I will been my discussion by addressing the possibility that I have used the aforementioned packages improperly. I will proceed as though I have not, but if there is something drastically wrong it is likely due to my limited time and fast review of the default package settings.

Onto the good stuff. As hinted at in the problem statement, there were problems observed during both the maximum likelihood estimation and the sampling. The EM algorithm did a good job fitting models to the data, but was (1) unable to truly approach the "true" values and (2) reached a variety of modes. The closest attempt was the third (I changed nothing except the seed). It seeems as though the first model (the one centered at 0, $\mu_1 = 0$, with low density, $p_1 = 0.1$), is very hard for these models to identify. Instead, most seem to put too little density in the model with large mean and large variance. I assume that the problem was constructed purposely to demonstrate an issue such as this one.

Lastly, the sampling also struggled–even more so. In fact, my sampler effectively only built two normals, even though I explicitly instructed it to build three. I wonder if I parameterized the model incorrectly, but I also think it is possible that this is simply where the Gibbs sampler converged to. The output it gives is representative of the distribution we are looking for. I thought this was extremely interesting. Given more time, I would have been interested to see what I could have done to force three normals, and then subsequently what they would have looked like. I have forgone including trace plots as it gets quite complicated with the "Infy" value for the third normal's sigma^2 value. Aside from this however, the plots that do print appear to be converging as we would expect.

## 9. Chapter 15 Question 3

### Part A

First, I made sure to standardize my predictors along with my outcome (through both mean centering and scaling) as Marquardt and Snee (1975) recommend that we be sure to do. This helps us deal with the collinearity in the predictors that is quite intuitive given that the last six are derived directly from the first three. Now, we run a basic ordinary linear regression model (nonhierarchical with a uniform prior distribution on the coefficients). I used both a frequentist method and a bayesian method with the defined priors to compare the results. I fit the frequentist model using a simple 'lm' function and used the packages 'stan' and 'rstanarm' to build the bayesian. The two were consistent, as I would expect them to be. See the beta coefficient estimates for these models below. I will be reporting the standardized coefficients for ease.

Table 20: Frequentist Model Estimates

| | 2.5% | 50% | 97.5% | Std Dev |
|---|---|---|---|---|
| (Intercept) | -0.17 | 0.13 | 0.43 | 0.12 |
| x1 | -0.99 | 0.25 | 1.48 | 0.51 |
| x2 | 0.08 | 0.16 | 0.24 | 0.03 |
| x3 | -2.59 | -0.92 | 0.74 | 0.68 |
| I(x1^2) | -5.66 | -2.28 | 1.10 | 1.38 |
| I(x2^2) | -0.21 | -0.11 | -0.01 | 0.04 |
| I(x3^2) | -3.78 | -1.67 | 0.45 | 0.86 |
| x1:x2 | -1.24 | -0.84 | -0.44 | 0.16 |
| x1:x3 | -9.86 | -4.08 | 1.69 | 2.36 |
| x2:x3 | -1.28 | -0.83 | -0.37 | 0.19 |

Table 21: Bayesian Model Estimates

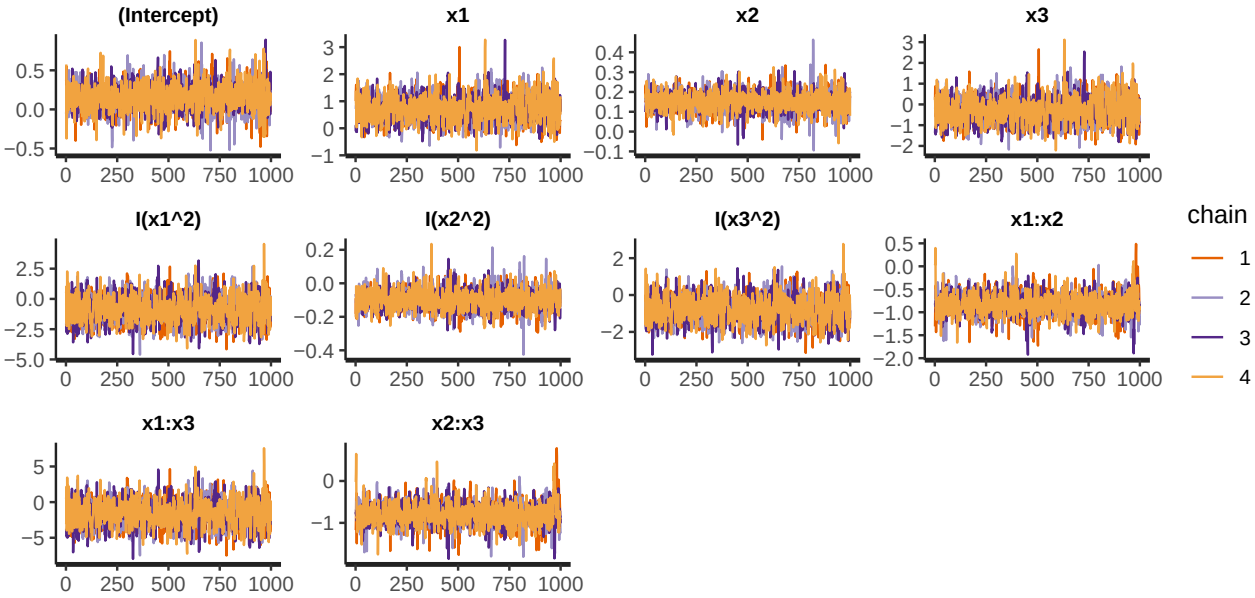
| | 10% | 50% | 90% | Std Dev |
|---|---|---|---|---|
| (Intercept) | -0.06 | 0.13 | 0.31 | 0.16 |
| x1 | -0.56 | 0.25 | 1.06 | 0.70 |
| x2 | 0.10 | 0.16 | 0.21 | 0.05 |
| x3 | -2.00 | -0.92 | 0.18 | 0.94 |
| I(x1^2) | -4.49 | -2.26 | 0.05 | 1.94 |
| I(x2^2) | -0.17 | -0.11 | -0.04 | 0.06 |
| I(x3^2) | -3.05 | -1.66 | -0.20 | 1.23 |
| x1:x2 | -1.07 | -0.83 | -0.58 | 0.22 |
| x1:x3 | -7.83 | -4.06 | -0.13 | 3.33 |
| x2:x3 | -1.10 | -0.82 | -0.53 | 0.24 |

**Part B**

Next, I fit a mixed-effects linear regression model again using the 'stan' and 'rstanarm' packages. As requested, I set a uniform prior distribution on the constant term and a shared normal prior distribution on the coefficients of the nine predictors. If you use iterative simulation in your computations, be sure to use multiple sequences and monitor their joint convergence. See the table below for the predicted coefficient estimates. Below the table is the set of traces plots I used to monitor their joint convergence. Indeed, we see appropriate convergence across multiple sequences.

Table 22: Hierarchical Normal Bayesian Model Estimates

| | 10% | 50% | 90% | Std Dev |
|---|---|---|---|---|
| (Intercept) | -0.04 | 0.15 | 0.35 | 0.16 |
| x1 | 0.16 | 0.67 | 1.28 | 0.45 |
| x2 | 0.10 | 0.15 | 0.20 | 0.05 |
| x3 | -1.02 | -0.33 | 0.48 | 0.60 |
| I(x1^2) | -2.24 | -0.94 | 0.50 | 1.07 |
| I(x2^2) | -0.16 | -0.10 | -0.03 | 0.06 |
| I(x3^2) | -1.67 | -0.83 | 0.11 | 0.69 |
| x1:x2 | -1.08 | -0.82 | -0.56 | 0.22 |
| x1:x3 | -4.01 | -1.79 | 0.68 | 1.82 |
| x2:x3 | -1.07 | -0.79 | -0.49 | 0.25 |




**Part C**

The model built in Part B appears to be more stable than that of Part A. This assertion is supported by the narrower credible intervals and the smaller standard deviations for $\beta_1$, $\beta_3$, $\beta_4$, and $\beta_8$ in Part B than in Part A. This is a reflection of the shared normal priors drawing the posterior means closer together, rather than allowing for unncessarily extreme values with the non-informative priors. I agree with Marquardt and Snee that model (b) is much preferred in this context. Lastly, by not unscaling the priors I recognize that the magnitude of the changes/differences between the coefficients in Part A and Part B is slightly obscured. However, I opted to leave them standardized to save time, and the expected effect is still displayed. If

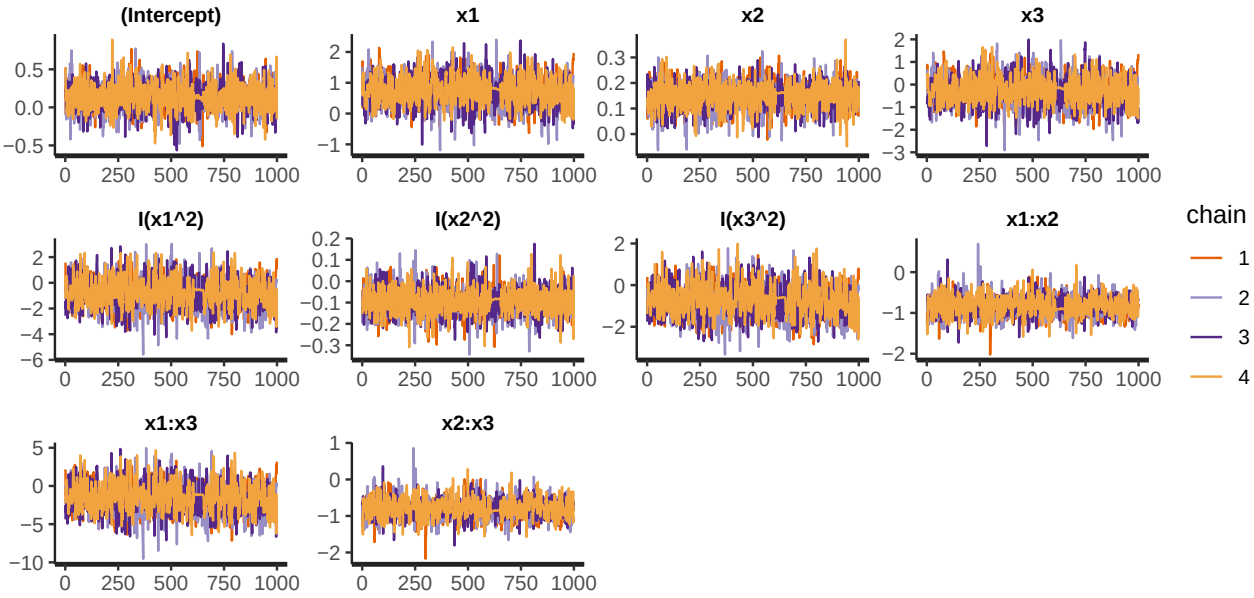
anything, I would only expect for the unstandardizing process to show the difference are even greater than displayed here.

### Part D

As directed, I fit a mixed-effects linear regression model with a uniform prior distribution on the constant term and a shared t4 prior distribution on the nine predictors. The output is displayed below. The t4 distribution seems to pull the coefficients slightly closer to 0 (the prior mean) than the normal distributions. This is not exactly what I would expected, given the wider tails associated with the student-t distribution. This could potentially have to do with the choice of degrees of freedom, or the data simply might not support the coefficients taking extreme values, even when given a somewhat of an opportunity. They did take extreme values in Part A, however, so maybe this just wasn't *enough* of an opporunity.

Table 23: Hierarchical T4 Bayesian Model Estimates

| | 10% | 50% | 90% | Std Dev |
|---|---|---|---|---|
| (Intercept) | -0.05 | 0.14 | 0.33 | 0.16 |
| x1 | 0.17 | 0.70 | 1.28 | 0.44 |
| x2 | 0.09 | 0.15 | 0.20 | 0.05 |
| x3 | -1.00 | -0.29 | 0.46 | 0.59 |
| I(x1^2) | -2.22 | -0.84 | 0.51 | 1.08 |
| I(x2^2) | -0.16 | -0.10 | -0.03 | 0.06 |
| I(x3^2) | -1.64 | -0.77 | 0.14 | 0.70 |
| x1:x2 | -1.09 | -0.82 | -0.57 | 0.22 |
| x1:x3 | -3.95 | -1.64 | 0.70 | 1.83 |
| x2:x3 | -1.08 | -0.79 | -0.50 | 0.24 |




### Part E

Other potential models for the regression coefficients could include Laplace distributions, creating a Lasso regression. This would be and even more severe restriction on the beta values, given that, as a prior, the double exponential function puts even more weight closer to zero. If we wanted to futher parse out which of the coefficients are most supported by the data, that could be a plausible option. On the other hand, if we wanted to allow more flexibility in the coefficients, we could use a t-distribution with a lower degree of freedom (to widen the tails).

## Code Appendix

```
knitr::opts_chunk$set(echo = F, results='asis', warning=F, message=F, cache=T,
                      fig.height=4, fig.width=5, fig.align="center")
pacman::p_load(actuar, dplyr, kableExtra, knitr, LearnBayes, boot, lattice, gtools, ggplot2,
               LaplacesDemon, rstan, rstanarm, bayesplot, mixtools, bayesmix, coda)
#######################################
# 1. The Importance Sampling Algorithm
ImpSampler <- function(nSamples, logTargetDensityFunc, logProposalDensityFunc,
                       proposalNewFunc, rejectionControlConstant = NULL) {
  # first check what's up with the rejectionControlConstant, i.e. is it null? if yes then ...
  if (is.null(rejectionControlConstant)) {
    # initialize samples vector w proposalnewfunc values (i.e. N(0,3^2))
    samples.vec <- rep(NA,nSamples)
    for (i in 1:nSamples){
      samples.vec[i] <- proposalNewFunc()
    }
    # initialize and find log weights
    log.weights.vec <- rep(NA,nSamples)
    final.log.weights.vec <- sapply(samples.vec, logTargetDensityFunc) -
      sapply(samples.vec, logProposalDensityFunc)
    # store samples for output
    final.samples <- samples.vec
    # acceptance rate doesn't apply here
    acceptance.rate <- NA
    # calculated estimated ESS as directed
    estimated.ESS <- length(final.log.weights.vec) /
      (1 + var(exp(final.log.weights.vec)))
  # if rejectionControlConstant was not null then
  }else{
    #initialize storage
    list.log.ratios = c()
    final.log.weights.vec = c()
    final.samples = c()
    # keep going until we've found the number of samples we want
    while (length(final.samples) < nSamples) {
      # again initialize samples vector w proposalnewfunc values (i.e. N(0,3^2))
      samples.vec <- rep(NA,nSamples)
      for (i in 1:nSamples){
        samples.vec[i] <- proposalNewFunc()
      }
      # again initialize and find log weights
      log.weights.vec <- rep(NA,nSamples)
      log.weights.vec <- sapply(samples.vec, logTargetDensityFunc) -
        sapply(samples.vec, logProposalDensityFunc)

      # now perform rejection control across all weights
      # we're working on the log scale so subtract log("c" val)
      log.ratios <- log.weights.vec - log(rejectionControlConstant)
      # take log(min(1,w/"c")) i.e. min(0,log(weight/"c"))
      log.ratios <- ifelse(log.ratios>0,0,log.ratios)
      # accept or reject with calculated probability, again on log scale
      acceptance.bool <- (log(runif(nSamples)) < log.ratios)
```

```
      # add those that pass to the list of final samples
      final.samples <- c(final.samples, samples.vec[acceptance.bool])
      # store the log.ratios that we tried
      list.log.ratios <- c(list.log.ratios, log.ratios)
      # update weights (w/r) and add to list of final weights
      final.log.weights.vec <- c(final.log.weights.vec,
                                 log.weights.vec[acceptance.bool] -
                                   log.ratios[acceptance.bool])
    }
    #calculate acceptance rate
    acceptance.rate <- length(final.samples)/length(list.log.ratios)
    #make sure we have no extra samples (chop off the extras if we do),
        #and make final update to weights (p*w/r)
    final.log.weights.vec <- final.log.weights.vec[1:nSamples] +
      log(mean(exp(list.log.ratios)))
    #again remove extra samples
    final.samples <- final.samples[1:nSamples]
    #calculate the estimated ESS
    estimated.ESS <- nSamples/(1 + var(exp(final.log.weights.vec)))
  }
  return(list(final.samples, final.log.weights.vec, estimated.ESS, acceptance.rate))
}
test.values <- seq(-6,6,.01)
fx.normal.samples <- (1/3)*dnorm(test.values,mean=-2,sd=1) + (2/3)*dnorm(test.values,mean=2,sd=1)
gx.normal.samples <- dnorm(test.values,mean=0,sd=3)

plot(test.values, fx.normal.samples, ylim=c(0,1.1*max(fx.normal.samples)),
      type="l", xlab="test.values", ylab="", xaxs="i",
      yaxs="i", yaxt="n", bty="n", cex=2, col = "red")
lines(test.values, gx.normal.samples, col = "blue")
legend("topright", c("f(x)", "g(x)"), col = c("red", "blue"), cex = 1, lty = 1)
#store the theoretically calculated values
mu1.theo <- 2/3
mu2.theo <- 5
theta.theo <- (1/3)*exp(-3/2) + (2/3)*exp(5/2)
#write functions to use as inputs to ImpSampler
logfx <- function(x){
  log((1/3)*dnorm(x,mean=-2,sd=1) + (2/3)*dnorm(x,mean=2,sd=1))
}

loggx <- function(x){
  log(dnorm(x,mean=0,sd=3))
}

gSampleFunc <- function(){
  rnorm(1,mean=0,sd = 3)
}

####importance sampling without rejection control
output.partE <- ImpSampler(5000, logfx, loggx, gSampleFunc)
# calculations estimate mu1, mu2, and theta
mu1 <- function(x){x}
mu2 <- function(x){x^2}
```

```r
theta <- function(x){exp(x)}

# function to calculate the weighted average of each of the h(x) functions:
w.avg.func <- function(hx, imp.samp.list){
  return(sum(hx(imp.samp.list[[1]]) * exp(imp.samp.list[[2]]))/sum(exp(imp.samp.list[[2]])))
}

estimates <-  c(w.avg.func(mu1, output.partE), w.avg.func(mu2, output.partE), w.avg.func(theta, output.p
ess <- output.partE[[3]]

partE.df <- data.frame(c(mu1.theo,mu2.theo,theta.theo), estimates, c(ess,NA,NA))
partE.df <- round(partE.df, 3)
rownames(partE.df) <- c("mu1", "mu2", "theta")
colnames(partE.df) <- c("Theoretical Values", "Importance Sampling Estimates", "Estimated ESS")

kable(partE.df, "latex", caption = "Importance Sampling Without Rejection Control", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))
####importance sampling with rejection control (1:10)
#initialize
acc.rates <- NULL
estimated.ESS <- NULL
mu1.errors <- NULL
mu2.errors <- NULL
theta.errors <- NULL

#run through and store output with varying values for the rejection control as directed
for (i in 1:10) {
  output.partF <- ImpSampler(5000, logfx, loggx, gSampleFunc, rejectionControlConstant = i)
  acc.rates <- c(acc.rates, output.partF[[4]])
  estimated.ESS <- c(estimated.ESS, output.partF[[3]])
  mu1.errors <- c(mu1.errors, abs(w.avg.func(mu1, output.partF) - mu1.theo))
  mu2.errors <- c(mu2.errors, abs(w.avg.func(mu2, output.partF) - mu2.theo))
  theta.errors <- c(theta.errors, abs(w.avg.func(theta, output.partF) - theta.theo))
}

partF.df <- data.frame(round(rbind(mu1.errors, mu2.errors, theta.errors, acc.rates, estimated.ESS), 2))
rownames(partF.df) <- c("Mu1 Errors", "Mu2 Errors", "Theta Errors", "Acceptance Rates", "Estimated ESS"
colnames(partF.df) <- c("c=1","c=2","c=3","c=4","c=5","c=6","c=7","c=8","c=9","c=10")
kable(partF.df, "latex", caption = "Importance Sampling With Varying Rejection Control", booktabs = T) %
kable_styling(latex_options = c("striped", "hold_position"), font_size = 8)
plot(mu1.errors, type = "l", xlab = "c value", main = "mu1 estimation errors", ylab = "absolute error")
plot(mu2.errors, type = "l", xlab = "c value", main = "mu2 estimation errors", ylab = "absolute error")
plot(theta.errors, type = "l", xlab = "c value", main = "theta estimation errors", ylab = "absolute erro

plot(acc.rates, ylim = c(0, 1), type = "l", xlab = "c value", main = "acceptance rates", ylab = "accepta
plot(estimated.ESS, ylim = c(0, 5000), type = "l", xlab = "c value", main = "ESS", ylab = "ESS")
########################################
# 2. Chapter 10 Question 5
###10.5A
#build random dataset from the model
yis <- c(rep(NA, 10))
nis <- c(rep(NA, 10))
xis <- c(rep(NA, 10))
```

```
alphais <- c(rep(NA, 10))
betais <- c(rep(NA, 10))

set.seed(1)
for (i in 1:10) {
  alpha <- rmt(1,0,2,4)
  alphais[i] <- alpha

  beta <- rmt(1,0,1,4)
  betais[i] <- beta

  xi <- runif(1, min=0, max=1)
  xis[i] <- xi

  ni <- rpois(1,5)
  nis[i] <- ni

  p <- inv.logit(alpha + beta*xi)

  yis[i] <- rbinom(1, ni, p)
}

dataset <- cbind(yis,nis)
dataset.df <- t(data.frame(dataset))
kable(dataset.df, "latex", caption = "10.5 Sampled Data", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))
###10.5B
#posterior function
logitBin = function (theta, data) {
  alpha <- theta[1]
  beta <- theta[2]
  y = data[,1]
  n = data[,2]
  N = length(xis)
  prelikelihood <- function (y, n, alpha, beta) y*log(inv.logit(alpha + beta*xis)) +
    (n-y)*log((1-inv.logit(alpha + beta*xis)))
  loglikelihood <- sum(prelikelihood(y, n, alpha, beta))
  post = loglikelihood #+ logPrior
  return(post)
}

#look at contours of posterior function across grid of alphas and betas
ng = 30
x0 = seq(-5, 5, len = ng)
y0 = seq(-15, 10, len = ng)
X = outer(x0, rep(1, ng))
Y = outer(rep(1, ng), y0)
n2 = ng^2
Z = apply(cbind(X[1:n2], Y[1:n2]), 1, logitBin, data = dataset)
Z = Z - max(Z)
Z = matrix(Z, c(ng, ng))
contour(x0, y0, Z, levels = c(-30, -20, -10, -5, -1, -0.1, -0.01), lwd = 2,
        xlab="alpha",
```

```
        ylab="beta",
        main="Contour plot of joint posterior")

####Rejection sampling
#find the mode
tpar=LearnBayes::laplace(logitBin,array(c(3,1),c(1,2)),data=dataset)
logitBinT=function(theta,data,tpar) {
    data=data
    d=logitBin(theta,data)-dmt(theta,mean=c(tpar$mode), S=2*tpar$var,df=4,
    log=TRUE)
    return(d)
}

start=array(c(3,1),c(1,2))
fit1=LearnBayes::laplace(logitBinT, start, data=dataset, tpar=tpar)
#logitBinT(fit1$mode,dataset,tpar)

rejectsamp=function(logf,tpar,dmax,n,data){
    theta=rmt(n,mean=c(tpar$mode),S=2*tpar$var,df=4)
    lf=apply(theta,1,logf,data=data)
    lg=dmt(theta,mean=c(tpar$mode),S=2*tpar$var,df=4,log=TRUE)
    prob=exp(lf-lg-dmax)
    return(theta[runif(n)<prob,])
}

theta=rejectsamp(logitBin,tpar,logitBinT(fit1$mode,dataset,tpar),2000,dataset)
#dim(theta)
contour(x0, y0, Z, levels = c(-30, -20, -10, -5, -1, -0.1, -0.01), lwd = 2,
        xlab="alpha",
        ylab="beta",
        main="Posterior Draws from Rejection Sampling",
        drawlabels = F)
points(theta[,1],theta[,2])
###10.5C
# build normal approximation
tpar=LearnBayes::laplace(logitBin,array(c(-1,1),c(1,2)),data=dataset)
logitBinT=function(theta,data,tpar) {
  data=data
  d=logitBin(theta,data)-dmt(theta,mean=c(tpar$mode), S=2*tpar$var,df=4,log=TRUE)
  return(d)
}
start=array(c(-1,1),c(1,2))
fit1=LearnBayes::laplace(logitBinT,start,data=dataset,tpar=tpar)
#logitBinT(fit1$mode,dataset,tpar)

dataset.df10.5C1 <- data.frame(fit1$mode)
colnames(dataset.df10.5C1) <- c("alpha", "beta")
rownames(dataset.df10.5C1) <- c("")
kable(dataset.df10.5C1, "latex", caption = "Estimated Modes", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))

dataset.df10.5C2 <- data.frame(fit1$var)
colnames(dataset.df10.5C2) <- c("alpha", "beta")
```

```r
rownames(dataset.df10.5C2) <- c("alpha", "beta")
kable(dataset.df10.5C2, "latex", caption = "Estimated Covariance Matrix", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))
###10.5D
##importance sampling using normal centered at the posterior mode with covariance
    #matrix fit to the curvature at the mode
theta = rmt(1000, mean = c(tpar$mode), S = tpar$var, df = 4)
lf = apply(theta,1,logitBin,data=dataset)
lp = dmt(theta, mean = c(tpar$mode), S = tpar$var, df = 4,log = TRUE)
wt = exp(lf - lp)
probs=wt/sum(wt, na.rm = T)
probs[is.na(probs)] <- 0
theta.S=theta[sample(1:1000,size=1000,prob=probs,replace=TRUE),]

contour(x0, y0, Z, levels = c(-30, -20, -10, -5, -1, -0.1, -0.01), lwd = 2,
        xlab="alpha",
        ylab="beta",
        main="Posterior Draws from Importance Sampling",
        drawlabels = F)
points(theta.S[,1],theta.S[,2])

#estimates
dataset.df10.5D <- t(data.frame(apply(theta.S,2,summary)))
rownames(dataset.df10.5D) <- c("alpha", "beta")
kable(dataset.df10.5D, "latex", caption = "10.5 Estimates for Expectations", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))
###10.5E
normalize = function(x){return(x/sum(x))}
seff <- 1/sum(normalize(wt)^2)
########################################
# 3. Chapter 10 Question 8
###10.8A
# load data
x108 <- c(-0.86,-0.30,-0.05,0.74)
n108 <- rep(5,length(x108))
y108 <- c(0,1,3,5)
dataset108<-cbind(y108,n108)

#posterior function
logitBin108 = function (theta, data)
{
  alpha <- theta[1]
  beta <- theta[2]
  y = data[,1]
  n = data[,2]
  prelikelihood <- function (y, n, alpha, beta) y*log(inv.logit(alpha + beta*x108)) +
    (n108-y)*log((1-inv.logit(alpha + beta*x108)))
  loglikelihood <- sum(prelikelihood(y, n, alpha, beta))
  logPrior <- 0 + 0 #non informative
  post = loglikelihood + logPrior
  return(post)
}
```

```r
#look at contours of posterior function across grid of alphas and betas
ng = 50
#grid suggestion from book
x0 = seq(-2, 7, len = ng)
y0 = seq(-2, 35, len = ng)
X = outer(x0, rep(1, ng))
Y = outer(rep(1, ng), y0)
n2 = ng^2
Z = apply(cbind(X[1:n2], Y[1:n2]), 1, logitBin108, data = dataset108)
Z = exp(Z)
Z = matrix(Z, c(ng, ng))
contour(x0, y0, Z, levels = c(seq(4.822959e-39, 2.737619e-03, length.out=10)),
        lwd = 1, ylab = "beta", xlab= "alpha",
        main="Contour Plots to Compare to Figure 3.3b",
        drawlabels = F)

# build normal approximation
tpar=LearnBayes::laplace(logitBin108,array(c(1,10),c(1,2)),data=dataset108)
betabinT108=function(theta,data,tpar) {
    data=data
    d=logitBin108(theta,data)-dmt(theta,mean=c(tpar$mode), S=2*tpar$var,df=4,log=TRUE)
    return(d)
}
start=array(c(1,10),c(1,2))
fit1=LearnBayes::laplace(betabinT108,start,data=dataset108, tpar=tpar)
#betabinT108(fit1$mode,dataset108,tpar)

#importance resampling SIR without replacement
theta = rmt(10000, mean = c(tpar$mode), S = tpar$var, df = 4)
lf = apply(theta,1,logitBin108,data=dataset108)
lf[is.na(lf)] <- 0
lp = dmt(theta, mean = c(tpar$mode), S = tpar$var, df = 4,log = TRUE)
md = max(lf - lp)
wt = exp(lf - lp)
probs=wt/sum(wt)
theta.S=theta[sample(1:length(probs),size=1000,prob=exp(probs),replace=F),]
#apply(theta.S,2,summary)

contour(x0, y0, Z, levels = c(seq(4.822959e-39, 2.737619e-03, length.out=10)),
        lwd = 1, ylab = "beta", xlab= "alpha",
        main="Importance Resampling Without Replacement",
        drawlabels = F)
points(theta.S[,1],theta.S[,2], pch='.')
###10.8B
hist(wt, main="Distribution of the Simulated Importance Ratios",
     xlab="Simulated Importance Ratios with Outliers")
#remove outliers
for(i in 1:40){
  wt<-wt[wt < max(wt)]
}

hist(wt, main="Distribution of the Simulated Importance Ratios",
     xlab="Simulated Importance Ratios without Outliers")
```

```r
###10.8C
#importance resampling SIR WITH replacement
theta = rmt(10000, mean = c(tpar$mode), S = tpar$var, df = 4)
lf = apply(theta,1,logitBin108,data=dataset108)
lf[is.na(lf)] <- 0
lp = dmt(theta, mean = c(tpar$mode), S = tpar$var, df = 4,log = TRUE)
md = max(lf - lp)
wt = exp(lf - lp)
probs=wt/sum(wt)
theta.S=theta[sample(1:length(probs),size=1000,prob=exp(probs),replace=T),]
#apply(theta.S,2,summary)

contour(x0, y0, Z, levels = c(seq(4.822959e-39, 2.737619e-03, length.out=10)),
        lwd = 1, ylab = "beta", xlab= "alpha",
        main="Importance Resampling With Replacement",
        drawlabels = F)
points(theta.S[,1],theta.S[,2], pch='.')
########################################
# 4. Chapter 11 Question 2
###11.2
#initializations
sims <- 1000
theta <-  matrix(0, nrow = sims, ncol = 2)
r.all <- rep(0,sims)

#starting positions found from normal approximation
theta[1,] <- c(0.847,7.749)

dataset.df11.2.1 <- t(data.frame(theta[1,]))
colnames(dataset.df11.2.1) <- c("alpha", "beta")
rownames(dataset.df11.2.1) <- c("")
kable(dataset.df11.2.1, "latex", caption = "Estimated Modes", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))

#jumping function
jumping <- function(theta, scale=.1) rmnorm(1, mean = c(tpar$mode), varcov = matrix(tpar$var,2,2))

dataset.df11.2.2 <- data.frame(matrix(tpar$var,2,2))
colnames(dataset.df11.2.2) <- c("alpha", "beta")
rownames(dataset.df11.2.2) <- c("alpha", "beta")
kable(dataset.df11.2.2, "latex", caption = "Estimated Covariance Matrix", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))
#metropolis algorithm
for(i in 2:sims) {
  theta.new = jumping(theta[i-1,], .1)
  uu = runif(1)
  r <- min(exp(logitBin108(theta.new, dataset108) - logitBin108(theta[i-1,], dataset108)), 1)
  if (uu < r) {
    theta[i,] <- theta.new
  }else{
    theta[i,] = theta[i-1,]
  }
}
```

```r
#samples/walks
contour(x0, y0, Z, levels = c(seq(4.822959e-39, 2.737619e-03, length.out=10)),
        lwd = 1, ylab = "beta", xlab= "alpha",
        drawlabels = F)
points(theta[,1],theta[,2], pch='.')
# plots for convergence? / investigation / display
firstChain <- c(rep(NA, sims))
for (i in 1:sims) {
  firstChain[i] <- exp(logitBin108(theta[i,],dataset108))
}
# probability
#plot(seq(1:sims),firstChain,type="l",xlab="Iteration",
    #ylab="posterior prob",main="post convergence?")
#alpha
plot(seq(1:sims),theta[,1],type="l",xlab="Iteration",
     ylab="Sampled Alpha",main="Alpha Convergence")
#beta
plot(seq(1:sims),theta[,2],type="l",xlab="Iteration",
     ylab="Sampled Beta",main="Beta Convergence")
########################################
# 5. Chapter 11 Question 3
###11.3
#load in data
dataset113 <-  data.frame(machineno = rep(1:6, each=5),
                          ments = c(83, 92, 92, 46, 67,
                                    117, 109, 114, 104, 87,
                                    101, 93, 92, 86, 67,
                                    105, 119, 116, 102, 116,
                                    79, 97, 103, 79, 92,
                                    57, 92, 104, 77, 100))
J <- length(unique(dataset113$machineno))
n <- length(dataset113$ments)

#starting points
# from text "obtain 10 starting points for the simulations by drawing thetaj
    #independently in this way for each group"

set.seed(142857)
theta.start <- sapply(1:6,function(x) sample(dataset113$ments[dataset113$machineno==x], 10, replace=TRU
mu.start <- apply(theta.start, 2, mean)
sigma.start <- sqrt(apply(theta.start, 2, var))

# ------------------------------------------------------------------------------------------------------

#heirarchical model functions
#tau (step 4)
tau.post.sample <- function(theta) {
  #first find tau hat
  tau.hat.func <- function(theta) {
    mu <- mean(theta)
    tau.hat <- (1/(J-1)) * sum((theta-mu)^2)
    return(tau.hat)
  }
```

```
  #next find conditional tau
  tau.hat <- tau.hat.func(theta)
  tau.cond <- rinvchisq(1,J-1,tau.hat)
  return(tau.cond)
}
#sigma (step 3)
sigma.post.sample <-  function(theta) {
  #first find sigma hat
  sigma.hat.func <- function(theta) {
    sigma.hat <- sapply(1:6, function(x) (dataset113$ments[dataset113$machineno==x] - theta[x])^2)
    sigma.hat <- (1/n) * sum(unlist(sigma.hat))
    return(sigma.hat)
  }
  #next find conditional sigma
  sigma.hat <- sigma.hat.func(theta)
  sigma.cond <- rinvchisq(1,n,sigma.hat)
  return(sigma.cond)
}

#mu (step 2)
mu.post.sample <- function(theta,tau) {
  #first find mu hat
  mu.hat <- mean(theta)
  #next find conditional mu
  mu.cond <- rnorm(1,mu.hat,sqrt(tau/J))
  return(mu.cond)
}

#theta (step 1)
theta.post.sample <- function(mu,sigma,tau){
  theta <- NULL
  for(j in 1:J) {
    n.j  <- length(dataset113$ments[dataset113$machineno==j])
    y.bar.j <- mean(dataset113$ments[dataset113$machineno==j])
    #first find V hat
    V.hat <- 1 / ((n.j/sigma)+(1/tau))
    #next find theta hat
    theta.hat <- (((y.bar.j*n.j)/sigma) + (mu/tau)) * V.hat
    #last find conditional theta
    theta[j] <- rnorm(1,theta.hat,sqrt(V.hat))
  }
  return(theta)
}

# ------------------------------------------------------------------------------------------------------------------------

#gibbs sampler function
sims <- 200

gibbs.sampler <- function(sample.i) {
  #initializations
  no.params <- 9
  param.storage <- matrix(NA, sims, no.params)
```

```
  colnames(param.storage) <- c("theta1", "theta2", "theta3", "theta4", "theta5", "theta6",
                               "mu", "sigma2", "tau2")

  #starting values for theta, tau, sigma and mu
  param.storage[1,1:6] <- sample.i
  param.storage[1,9] <- tau.post.sample(sample.i)
  param.storage[1,8] <- sigma.post.sample(sample.i)
  param.storage[1,7] <- mu.post.sample(sample.i,param.storage[1,9])#param.storage[1,9])

  #interations
  for (i in 2:sims) {
    param.storage[i,1:6] <- theta.post.sample(param.storage[i-1,7], param.storage[i-1,8], param.storage
    param.storage[i,9] <- tau.post.sample(param.storage[i,1:6])
    param.storage[i,8] <- sigma.post.sample(param.storage[i,1:6])
    param.storage[i,7] <- mu.post.sample(param.storage[i,1:6], param.storage[i,9])
  }
return(param.storage)
}

#run sampler!
param.storage <- gibbs.sampler(mu.start)
#snag output (second halves as directed in text)
gibbs.output <- param.storage[seq(sims/2+1, sims, 1),]
gibbs.output[,8:9] <- sqrt(gibbs.output[,8:9])

#summary table
df.11.3.output <- data.frame(t(apply(gibbs.output, 2, function(x) quantile(x, c(.025,.25,.5,.75,.975)))))
colnames(df.11.3.output) <- c("2.5%", "25%", "50%", "75%", "97.5%")
kable(df.11.3.output, "latex", caption = "Hierarchical Model Estimates", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))
#measures for theta 6 for seperate model
y.bar.6.dot <- mean(dataset113$ments[dataset113$machineno==6])
var.sep.6 <- var(dataset113$ments[dataset113$machineno==6])
theta.sep.6 <- rnorm(sims, y.bar.6.dot, sqrt(var.sep.6))

#measures for theta 6 for pooled model
y.bar.dot.dot <- mean(dataset113$ments)
var.pooled <- sum((dataset113$ments[dataset113$machineno==6] - mean(dataset113$ments))^2) /
  (length(dataset113$ments) - 1)
theta.pooled.6 <- rnorm(sims, y.bar.dot.dot, sqrt(var.pooled))

# posterior distribution of the mean of the quality measurements of the sixth machine
plot(density(gibbs.output[,"theta6"]), col="red",
     xlab="Mean Measure",
     ylab="Density",
     main="(i) Mean of Machine 6")
lines(density(theta.pooled.6[seq(sims/2+1, sims, 1)]), col="blue")
lines(density(theta.sep.6[seq(sims/2+1, sims, 1)]), col="green")
legend("topright",col = c("red","blue", "green"),
       legend=c("hierarchical","pooled", "separated"),
       lty = c(1,1,1))
# predictive distribution for another quality measurement of the sixth machine (maximum?)
plot(density(gibbs.output[,"theta6"]), col="red",
```

```
     xlab="Mean Measure",
     ylab="Density",
     main="(ii) Mean of Machine 6")
lines(density(theta.pooled.6[seq(sims/2+1, sims, 1)]), col="blue")
lines(density(theta.sep.6[seq(sims/2+1, sims, 1)]), col="green")
lines(x=c(y.bar.6.dot,y.bar.6.dot),y=c(-1,1), col="black")
legend("topright",col = c("red","blue", "green"),
       legend=c("hierarchical","pooled", "separated"),
       lty = c(1,1,1))

pvalmean1 <- length(which(gibbs.output[,"theta6"] > mean(y.bar.6.dot))) /
  length(gibbs.output[,"theta6"])

pvalmean2 <- length(which(theta.pooled.6[seq(sims/2+1, sims, 1)] > mean(y.bar.6.dot))) /
  length(theta.pooled.6[seq(sims/2+1, sims, 1)])

pvalmean3 <- length(which(theta.sep.6[seq(sims/2+1, sims, 1)] > mean(y.bar.6.dot))) /
  length(theta.sep.6[seq(sims/2+1, sims, 1)])

df.11.3.pvals <- data.frame(pvalmean1, pvalmean2, pvalmean3)
colnames(df.11.3.pvals) <- c("Hierarchical", "Pooled", "Seperate")
rownames(df.11.3.pvals) <- c("P-Value")
kable(df.11.3.pvals, "latex", caption = "Posterior Predictive Check", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))

#measures for machine 7 for pooled model
y.bar.dot.dot <- mean(dataset113$ments)
var.pooled <- var(dataset113$ments)
#sum((dataset113$ments[dataset113$machineno==6] - mean(dataset113$ments))^2) /
 # (length(dataset113$ments) - 1)
theta.pooled.7 <- rnorm(sims, y.bar.dot.dot, sqrt(var.pooled))

#hierarchical
theta.hier.7 <- rnorm(sims, gibbs.output[,"mu"], sqrt(gibbs.output[,"sigma2"]))

# posterior distribution of the mean of the quality measurements of the potential seventh machine
plot(density(theta.hier.7), col="red", xlab="Mean Measure",
    ylab="Density", main="(iii) Mean of Potential Machine 7")
lines(density(theta.pooled.7[seq(sims/2+1, sims, 1)]), col="blue")
legend("topright",col = c("red","blue"),
       legend=c("hierarchical","pooled"),
       lty = c(1,1,1))
########################################
# 6. Chapter 11 Question 4
###11.4
#heirarchical model functions
#tau (step 4)
tau.post.sample <- function(theta) {
  #first find tau hat
  tau.hat.func <- function(theta) {
    mu <- mean(theta)
    tau.hat <- (1/(J-1)) * sum((theta-mu)^2)
    return(tau.hat)
```

```
  }
  #next find conditional tau
  tau.hat <- tau.hat.func(theta)
  tau.cond <- rinvchisq(1,J-1,tau.hat)
  return(tau.cond)
}

# change sigma function
sigma.post.sample.p2 <- function(theta, mu) {
  sigma2.star <- NULL
  for(j in 1:J) {
    k <- 5
    nu.k <- 1 + k
    sigma.0.grid <- seq(0.25, 30, 0.25)
    sigma2.star.vals<-NULL
    probs <- NULL
    for (i in 1:length(sigma.0.grid)) {
      v <- mean((theta[j] - mu)^2)
      sigma2.k.option <- (1 * sigma.0.grid[i] + k*v) / (1 + k)
      sigma2.star.vals[i] <- rinvchisq(1, nu.k, sigma2.k.option)
      probs[i] <- dinvchisq(sigma2.star.vals, nu.k, sigma2.k.option)
    }
    #print(mean(probs))
    sigma2.star[j] <- sample(sigma2.star.vals, 1, prob=probs)
  }
  return(sigma2.star)
}

#mu (step 2)
mu.post.sample <- function(theta,tau) {
  #first find mu hat
  mu.hat <- mean(theta)
  #next find conditional mu
  mu.cond <- rnorm(1,mu.hat,sqrt(tau/J))
  return(mu.cond)
}

#theta (step 1)
theta.post.sample <- function(mu,sigma,tau){
  theta <- NULL
  for(j in 1:J) {
    n.j  <- length(dataset113$ments[dataset113$machineno==j])
    y.bar.j <- mean(dataset113$ments[dataset113$machineno==j])
    #first find V hat
    V.hat <- 1 / ((n.j/sigma[j])+(1/tau))
    #next find theta hat
    theta.hat <- (((y.bar.j*n.j)/sigma[j]) + (mu/tau)) * V.hat
    #last find conditional theta
    theta[j] <- rnorm(1,theta.hat,sqrt(V.hat))
  }
  return(theta)
}
```

```r
# ------------------------------------------------------------------------------------------------------------

#gibbs sampler function
sims <- 200

gibbs.sampler <- function(sample.i) {
  #initializations
  no.params <- 14
  param.storage <- matrix(NA, sims, no.params)
  colnames(param.storage) <- c("theta1", "theta2", "theta3", "theta4", "theta5", "theta6",
                               "sigma1", "sigma2", "sigma3", "sigma4", "sigma5", "sigma6",
                               "tau2","mu")
  #starting values for theta, tau, sigma and mu
  param.storage[1,1:6] <- sample.i
  param.storage[1,13] <- tau.post.sample(sample.i)
  param.storage[1,14] <- mu.post.sample(sample.i,param.storage[1,13])
  param.storage[1,7:12] <- sigma.post.sample.p2(sample.i,param.storage[1,14])

  #interations
  for (i in 2:sims) {
    param.storage[i,1:6] <- theta.post.sample(param.storage[i-1,14], param.storage[i-1,7:12], param.sto
    param.storage[i,13] <- tau.post.sample(param.storage[i,1:6])
    param.storage[i,14] <- mu.post.sample(param.storage[i,1:6], param.storage[i,13])
    param.storage[i,7:12] <- sigma.post.sample.p2(param.storage[i,1:6], param.storage[i,14])

  }
return(param.storage)
}

#run sampler!
param.storage <- gibbs.sampler(mu.start)
#snag output (second halves as directed in text)
gibbs.output <- param.storage[seq(sims/2+1, sims, 1),]
gibbs.output[,7:13] <- sqrt(gibbs.output[,7:13])

#summary table
df.11.4.output <- data.frame(t(apply(gibbs.output, 2, function(x) quantile(x, c(.025,.25,.5,.75,.975)))
colnames(df.11.4.output) <- c("2.5%", "25%", "50%", "75%", "97.5%")
kable(df.11.4.output, "latex", caption = "Hierarchical Model Estimates", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))
yks = c(118, 74, 44, 24, 29, 22, 20, 14, 20, 15, 12, 14, 6, 12, 6, 9, 9, 6, 10, 10, 11, 5, 3, 3)
xks = c(1:24)
dataset13.5 <- cbind(yks,xks)

species.observed <- sum(yks)
animals.caught <- sum(yks * c(1:24))
probs <- (1/species.observed) * c(1:24)

# joint posterior function
negmultbin = function (theta, data) {
  alpha <- theta[1]
  beta <- theta[2]
  y = data[,1]
```

```
  x = data[,2]
  prelikelihood <- function(y,alpha,beta) y*log( (gamma(alpha+x)/(factorial(x)*gamma(alpha))) * ((beta^a
  loglikelihood <- sum(prelikelihood(y, alpha, beta))
  post = loglikelihood
  return(post)
}

#find mode
negmultbinlaplace <- LearnBayes::laplace(negmultbin,array(c(1,0.1),c(1,2)),data=dataset13.5)

#output desired information
dataset.df13.5C1 <- data.frame(negmultbinlaplace$mode)
colnames(dataset.df13.5C1) <- c("alpha", "beta")
rownames(dataset.df13.5C1) <- c("")
kable(dataset.df13.5C1, "latex", caption = "Estimated Modes", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))

dataset.df13.5C2 <- data.frame(negmultbinlaplace$var)
colnames(dataset.df13.5C2) <- c("alpha", "beta")
rownames(dataset.df13.5C2) <- c("alpha", "beta")
kable(dataset.df13.5C2, "latex", caption = "Estimated Covariance Matrix", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))

#build contours
ng = 100
x0 = seq(0, 3, len = ng)
y0 = seq(0, 0.5, len = ng)
X = outer(x0, rep(1, ng))
Y = outer(rep(1, ng), y0)
n2 = ng^2
Z = apply(cbind(X[1:n2], Y[1:n2]), 1, negmultbin, data = dataset13.5)
Z = exp(Z-max(Z,na.rm=T))
Z = matrix(Z, c(ng, ng))
contour(x0, y0, Z, levels = c(0.00000001, 0.0000001, 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1), lwd =
        xlab="alpha", ylab="beta", drawlabels = F,
        main="Contour plot of joint posterior")
Z <- ifelse(is.na(Z),0,Z)
no.species.stor <- c()
nsim <- 1
no.animals <- 0

for(i in 1:1000){
  no.animals <- 0
  samples <- c()
  samples.storage <- c()
  while (no.animals < 10000) {
    dens.alpha <- apply(Z,1,sum)
    alpha.indices <- sample(1:length(x0), nsim, replace=T, prob=dens.alpha)
    alpha <- x0[alpha.indices]
    beta <- rep(NA,nsim)
    for (i in (1:nsim)) {
      beta[i] <- sample(y0, 1, prob=Z[alpha.indices[i],])
    }
```

```
    sample <- rztnbinom(nsim, size=alpha, prob=beta)
    samples.storage <- c(samples.storage, sample)
    no.animals <- no.animals + sample
  }
  no.species.stor <- c(no.species.stor,sum(table(samples.storage)))
}

no.species.stor.df <- data.frame(no.species.stor)
colnames(no.species.stor.df) <- c("val")

dataset.df13.5D <- data.frame((quantile(no.species.stor.df$val, c(0.025, 0.975)))[1],
                              mean(no.species.stor.df$val),
                              (quantile(no.species.stor.df$val, c(0.025, 0.975)))[2])
dataset.df13.5D <- round(dataset.df13.5D, 2)
colnames(dataset.df13.5D) <- c("2.5%", "50%", "97.5%")
rownames(dataset.df13.5D) <- c("")

kable(dataset.df13.5D, "latex", caption = "Estimate and Posterior Interval", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))

ggplot(data=no.species.stor.df, aes(val)) +
  geom_histogram(stat = "bin", binwidth = 5, color="darkgrey", fill="white") +
  xlab("Number of New Species")

no.species.stor <- c()
nsim <- 1
no.animals <- 0
maxima <- c()
mean <- c()

for(i in 1:1000){
  no.animals <- 0
  samples <- c()
  samples.storage <- c()
  while (no.animals < 3326) {
    dens.alpha <- apply(Z,1,sum)
    alpha.indices <- sample(1:length(x0), nsim, replace=T, prob=dens.alpha)
    alpha <- x0[alpha.indices]
    beta <- rep(NA,nsim)
    for (i in (1:nsim)) {
      beta[i] <- sample(y0, 1, prob=Z[alpha.indices[i],])
    }
    sample <- rztnbinom(nsim, size=alpha, prob=beta)
    samples.storage <- c(samples.storage, sample)
    no.animals <- no.animals + sample
  }
  no.species.stor <- c(no.species.stor,sum(table(samples.storage)))
  mean <- c(mean, mean(table(samples.storage)))
  maxima <- c(maxima, as.numeric(rownames(table(samples.storage))[length(table(samples.storage))]))
}


no.species.stor.df <- data.frame(no.species.stor)
```

```
colnames(no.species.stor.df) <- c("val")
pval <- length(which(496 < no.species.stor.df$val)) / length(no.species.stor.df$val)
dataset.df13.5D1 <- data.frame(496,
                               (quantile(no.species.stor.df$val, c(0.025, 0.975)))[1],
                               mean(no.species.stor.df$val),
                               (quantile(no.species.stor.df$val, c(0.025, 0.975)))[2],
                               pval)
dataset.df13.5D1 <- round(dataset.df13.5D1, 2)
colnames(dataset.df13.5D1) <- c("Observed", "2.5%", "50%", "97.5%", "Pval")
rownames(dataset.df13.5D1) <- c("")

kable(dataset.df13.5D1, "latex", caption = "Posterior Check -- Number of Species", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))

ggplot(data=no.species.stor.df, aes(val)) +
  geom_histogram(stat = "bin", binwidth = 4, color="darkgrey", fill="white") +
  geom_vline(xintercept = 496) +
  xlab("Posterior Check -- Number of Species")
mean.df <- data.frame(mean)
colnames(mean.df) <- c("val")
pval <- length(which(mean(yks) < mean.df$val)) / length(mean.df$val)
dataset.df13.5D2 <- data.frame(mean(yks),
                               (quantile(mean.df$val, c(0.025, 0.975)))[1],
                               mean(mean.df$val),
                               (quantile(mean.df$val, c(0.025, 0.975)))[2],
                               pval)
dataset.df13.5D2 <- round(dataset.df13.5D2, 2)
colnames(dataset.df13.5D2) <- c("Observed", "2.5%", "50%", "97.5%", "Pval")
rownames(dataset.df13.5D2) <- c("")

kable(dataset.df13.5D2, "latex", caption = "Posterior Check -- Mean Number Sightings", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))

ggplot(data=mean.df, aes(val)) +
  geom_histogram(stat = "bin", binwidth = 4, color="darkgrey", fill="white") +
  geom_vline(xintercept = mean(yks)) +
  xlab("Posterior Check -- Mean Number Sightings")
maxima.df <- data.frame(maxima)
colnames(maxima.df) <- c("val")
pval <- length(which(24 < maxima.df$val)) / length(maxima.df$val)
dataset.df13.5D3 <- data.frame(24,
                               (quantile(maxima.df$val, c(0.025, 0.975)))[1],
                               mean(maxima.df$val),
                               (quantile(maxima.df$val, c(0.025, 0.975)))[2],
                               pval)
dataset.df13.5D3 <- round(dataset.df13.5D3, 2)
colnames(dataset.df13.5D3) <- c("Observed", "2.5%", "50%", "97.5%", "Pval")
rownames(dataset.df13.5D3) <- c("")

kable(dataset.df13.5D3, "latex", caption = "Posterior Check -- Maximum Number of Sightings", booktabs =
kable_styling(latex_options = c("striped", "hold_position"))

ggplot(data=maxima.df, aes(val)) +
```

```
  geom_histogram(stat = "bin", binwidth = 4, color="darkgrey", fill="white") +
  geom_vline(xintercept = 24) +
  xlab("Posterior Check -- Maximum Number of Sightings")
#######################################
# 8. Generate / simulate n = 120 observations from a three component mixture
set.seed(100)
#investigate model and pull 120 observations as directed
y <- seq(-10,15,0.1)
#dens <- function (x, theta, stnd){dnorm (x, theta, sqrt(1000*theta*(1-theta)))}
dens.mix <- 0.1*dnorm(y,0,1) + 0.3*dnorm(y,-2,sqrt(2)) + 0.6*dnorm(y,3,sqrt(16))
plot(y, dens.mix, ylim=c(0,1.1*max(dens.mix)),
     type="l", xlab="", ylab="", xaxs="i",
     yaxs="i", yaxt="n", bty="n", cex=2,
     main="Gaussian Mixture Model Integrated")
lines(x=c(0,0), y=c(0,0.5))
lines(x=c(-2,-2), y=c(0,0.5))
lines(x=c(3,3), y=c(0,0.5))

vals.mix <- c(rnorm(120*0.1,0,1), rnorm(120*0.3,-2,sqrt(2)), rnorm(0.6*120,3,sqrt(16)))
labels <- c(rep(1, 12), rep(2, 36), rep(3, 72))
vals.mix <- data.frame(cbind(vals.mix, labels))

#use k means to initialize
vals.kmeans <- kmeans(vals.mix$vals.mix, 3)
vals.kmeans.cluster <- vals.kmeans$cluster
vals.df <- data.frame(x = vals.mix$vals.mix,
                      cluster = vals.kmeans.cluster)

vals.summary.df <- vals.df %>%
  dplyr::group_by(cluster) %>%
  dplyr::summarize(mu = mean(x), variance = var(x), std = sd(x), size = n()) %>%
  dplyr::mutate(p = size / sum(size))

dens.mix1 <- 0.1*dnorm(y,0,1)
dens.mix2 <- 0.3*dnorm(y,-2,sqrt(2))
dens.mix3 <- 0.6*dnorm(y,3,sqrt(16))
plot(y, dens.mix1, ylim=c(0,max(dens.mix)),
     type="l", xlab="", ylab="", xaxs="i",
     yaxs="i", yaxt="n", bty="n", cex=2,
     main="Gaussian Mixture Model Separated")
lines(y, dens.mix2)
lines(y, dens.mix3)
plot.normal.components <- function(mixture,component.number,...) {
  curve(mixture$lambda[component.number] *
          dnorm(x,mean=mixture$mu[component.number],
                sd=mixture$sigma[component.number]), add=TRUE, ...)
}

set.seed(3)
EM.output.1 <- normalmixEM(vals.mix$vals.mix,k=3,maxit=1000,epsilon=0.01)
EM.output.1.df <- data.frame(EM.output.1$mu, EM.output.1$sigma, EM.output.1$lambda)
colnames(EM.output.1.df) <- c("mu", "sigma^2", "p")
```

```
set.seed(20)
EM.output.2 <- normalmixEM(vals.mix$vals.mix,k=3,maxit=1000,epsilon=0.01)
EM.output.2.df <- data.frame(EM.output.2$mu, EM.output.2$sigma, EM.output.2$lambda)
colnames(EM.output.2.df) <- c("mu", "sigma^2", "p")

set.seed(100)
EM.output.3 <- normalmixEM(vals.mix$vals.mix,k=3,maxit=1000,epsilon=0.01)
EM.output.3.df <- data.frame(EM.output.3$mu, EM.output.3$sigma, EM.output.3$lambda)
colnames(EM.output.3.df) <- c("mu", "sigma^2", "p")
Gaus.mix.df <- data.frame(c(0, -2, 3), c(1, 2, 16), c(0.1, 0.3, 0.6))
colnames(Gaus.mix.df) <- c("mu", "sigma^2", "p")

df.8A <- cbind(Gaus.mix.df, EM.output.1.df,EM.output.2.df,EM.output.3.df)
df.8A <- round(df.8A, 2)
table.8A <- kable(df.8A, "latex", caption = "EM Algorithm MLEs", booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
add_header_above(table.8A, c("True Values" = 3, "Model 1" = 3, "Model 2" = 3, "Model 3" = 3))
hist(vals.mix$vals.mix,breaks=101,col="white",border="grey",freq=FALSE,
     xlab="Value",main="Gaussian Mixture Model EM1")
lines(density(vals.mix$vals.mix),lty=2)
invisible(sapply(1:4,plot.normal.components,mixture=EM.output.1))

hist(vals.mix$vals.mix,breaks=101,col="white",border="grey",freq=FALSE,
     xlab="Value",main="Gaussian Mixture Model EM2")
lines(density(vals.mix$vals.mix),lty=2)
invisible(sapply(1:4,plot.normal.components,mixture=EM.output.2))

hist(vals.mix$vals.mix,breaks=101,col="white",border="grey",freq=FALSE,
     xlab="Value",main="Gaussian Mixture Model EM3")
lines(density(vals.mix$vals.mix),lty=2)
invisible(sapply(1:4,plot.normal.components,mixture=EM.output.3))
model <- BMMmodel(vals.mix$vals.mix, k = 3, initialValues = list(S0 = 2),
                  priors = BMMpriors(y=vals.mix$vals.mix))
control <- JAGScontrol(variables = c("mu", "tau", "eta", "S"),
                       burn.in = 1000, n.iter = 5000, seed = 10)
z <- JAGSrun(vals.mix$vals.mix, model = model, control = control)
zSort <- Sort(z, by = "mu")

etas <- c(mean(z$results[,121]),mean(z$results[,122]),mean(z$results[,123]))
mus <- c(mean(z$results[,124]),mean(z$results[,125]),mean(z$results[,126]))
sigmas <- c(mean(z$results[,127]),mean(z$results[,128]),mean(z$results[,129]))

EM.output.1.df <- data.frame(mus, sigmas, etas)
colnames(EM.output.1.df) <- c("mu", "sigma^2", "p")
kable(EM.output.1.df, "latex", caption = "Gibbs Algorithm Estimates", booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hold_position"))

dens.mix1 <- etas[1]*dnorm(y,mus[1],sqrt(sigmas[1]))
dens.mix2 <- etas[2]*dnorm(y,mus[2],sqrt(sigmas[2]))
dens.mix3 <- etas[3]*dnorm(y,mus[3],ifelse(is.finite(sqrt(sigmas[3])), sqrt(sigmas[3]), 1000000))
plot(y, dens.mix1, ylim=c(0,max(dens.mix)),
     type="l", xlab="", ylab="", xaxs="i",
     yaxs="i", yaxt="n", bty="n", cex=2,
```

```
     main="Gaussian Mixture Model Gibbs Sampler")
lines(y, dens.mix2)
lines(y, dens.mix3)
#plot(z)
#######################################
# 9. Chapter 15 question 3
#load data
dataset153 <- data.frame(c(1300, 1300, 1300, 1300, 1300, 1300, 1200, 1200, 1200, 1200, 1200, 1200, 1100
                           c(7.5, 9.0, 11.0, 13.5, 17.0, 23.0, 5.3, 7.5, 11.0, 13.5, 17.0, 23.0, 5.3, 7.5
                           c(0.0120, 0.0120, 0.0115, 0.0130, 0.0135, 0.0120, 0.0400, 0.0380, 0.0320, 0.020
                             0.0840, 0.0980, 0.0920, 0.0860),
                           c(49.0, 50.2, 50.5, 48.5, 47.5, 44.5, 28.0, 31.5, 34.5, 35.0, 38.0, 38.5, 15.0
colnames(dataset153) <- c("x1", "x2", "x3", "y")

# standardizations
#get means for later in case
dataset153<-data.frame(scale(dataset153))
# MODEL 1 : frequentist model
freq.mod <- lm(y ~ x1 + x2 + x3 + x1*x2 + x1*x3 + x2*x3 + I(x1^2) + I(x2^2) + I(x3^2), data = dataset153
#summary(freq.mod)
output <- summary(freq.mod)$coef[, 1:2]
out1 <- cbind(output, confint(freq.mod)[c(1:10),])
colnames(out1) <- c("50%", "sd", "2.5%", "97.5%")
out1 <- out1[, c(3, 1, 4, 2)]
out1 <- data.frame(round(out1, 2))
colnames(out1) <- c("2.5%", "50%", "97.5%", "Std Dev")

kable(out1, "latex", caption = "Frequentist Model Estimates", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))
#vif(freq.mod)

# MODEL 2 : use simple MCMC regress (uniform priors?)
library(MCMCpack)
part.a.mod2 <- MCMCregress(y ~ x1 + x2 + x3 + x1*x2 + x1*x3 + x2*x3 + I(x1^2) + I(x2^2) + I(x3^2),
                           data = dataset153, burnin = 1000, mcmc = 10000)
summary(part.a.mod2)
plot(part.a.mod2)
# MODEL 3 : use rjags
library(rjags)
library(R2jags)

dataset153.dat<-list(y=dataset153$y,
                     x1=dataset153$x1,
                     x2=dataset153$x2,
                     x3=dataset153$x3)

cat(
    "model{
    for (i in 1:16) {
      y[i] ~dnorm(mu[i] , tau)
      mu[i] <- beta0 + beta1*x1[i] + beta2*x2[i] + beta3*x3[i]
    }
    beta0 ~dunif(-1000, 1000)
```

```
    beta1  ~dnorm(0, 0.00001)
    beta2 ~dnorm(0, 0.00001)
    beta3 ~dnorm(0, 0.00001)
    tau ~ dinvchi (0.001, 0.001)
    sigma2 <- 1/tau}",
    file="m1.jag"
    )

m1.inits<-list(list("beta0"=1,"beta1"=0,"beta2"=0,"beta3"= 0,
          "tau"=1))

parameters <- c("beta0", "beta1", "beta2", "beta3", "mu",
           "sigma2")

m1 <- jags(data = dataset153.dat,
        inits = m1.inits,
        parameters.to.save = parameters,
        model.file = "m1.jag",
        n.chains = 1,
        n.iter = 5000,
        n.burnin = 2000,
        n.thin = 1)

m1
plot(m1)
traceplot(m1)
plot(as.mcmc(m1))
# MODEL 4 : use stan
library(rstan)
library(rstanarm)
library(bayesplot)

#intercept:constant term,fixed effect
#mean center and scale:standardize

glm_post1 <- stan_glm(y ~ x1 + x2 + x3 + x1*x2 + x1*x3 + x2*x3 + I(x1^2) + I(x2^2) + I(x3^2),
                      data = dataset153,
                      family=gaussian,
                      prior=NULL)
#summary(glm_post1)
out2 <- summary(glm_post1)[1:10,3:6]
out2 <- out2[, c(2, 3, 4, 1)]
out2 <- data.frame(round(out2, 2))
colnames(out2) <- c("10%", "50%", "90%", "Std Dev")

kable(out2, "latex", caption = "Bayesian Model Estimates", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))

# Look at trace plots
# Ideally we want the chains in each trace plot to be stable (centered around one value) and well-mixed
    #(all chains are overlapping around the same value).
#stan_trace(glm_post1)
```

```r
#posterior predictive chekcs
#pp_check(glm_post1)
#ppc_intervals(y = dataset153$y, yrep = posterior_predict(glm_post1), x = dataset153$x1)

#stan_hist(glm_post1, pars=c("x1"), bins=40)
post_samps_speed <- as.data.frame(glm_post1, pars=c("x1"))[,"x1"]
mn_speed <- mean(post_samps_speed) # posterior mean
ci_speed <- quantile(post_samps_speed, probs=c(0.05, 0.95)) # posterior 90% interval

#stan_hist(glm_post1, pars=c("x2"), bins=40)
#stan_hist(glm_post1, pars=c("x3"), bins=40)
glm_post_mixed <- stan_glm(y ~ x1 + x2 + x3 + x1*x2 + x1*x3 + x2*x3 + I(x1^2) + I(x2^2) + I(x3^2),
                      data = dataset153,
                      family = gaussian(),
                      prior = normal(),
                      prior_intercept = NULL)

#summary(glm_post_mixed)
out3 <- summary(glm_post_mixed)[1:10,3:6]
out3 <- out3[, c(2, 3, 4, 1)]
out3 <- data.frame(round(out3, 2))
colnames(out3) <- c("10%", "50%", "90%", "Std Dev")

kable(out3, "latex", caption = "Hierarchical Normal Bayesian Model Estimates", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))

#tau:chisquarred15.4

# Look at trace plots
# Ideally we want the chains in each trace plot to be stable (centered around one value) and well-mixed
    #(all chains are overlapping around the same value).
stan_trace(glm_post_mixed)
glm_t4 <- stan_glm(y ~ x1 + x2 + x3 + x1*x2 + x1*x3 + x2*x3 + I(x1^2) + I(x2^2) + I(x3^2),
                      data = dataset153,
                      family = gaussian(),
                      prior=student_t(df = 4),
                      prior_intercept = NULL)
#stan_trace(glm_t4)
#summary(glm_t4)
out4 <- summary(glm_t4)[1:10,3:6]
out4 <- out4[, c(2, 3, 4, 1)]
out4 <- data.frame(round(out4, 2))
colnames(out4) <- c("10%", "50%", "90%", "Std Dev")

kable(out4, "latex", caption = "Hierarchical T4 Bayesian Model Estimates", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))

#summary(glm_t4)
stan_trace(glm_t4)
```