PHP 2530 Assignment #4

Anthony Sisti

4/21/2020

Question 1 : Improtance Sampling Algorithm

a) In this question we are tasked with writing a function that implements the importance sampling algorithm, with the option for rejection control. We follow the guide offered on the homework and present the function below.

```r
### Question 1 ####
## A

ImpSampler <- function(nSamples, logTargetDensityFunc,
                       logProposalDensityFunc, proposalNewFunc = NULL,
                       rejectionControlConstant = FALSE) {

  samplefrom <- seq(-10, 10, by= 0.01)
  draws <- rnorm(nSamples, mean = 0, sd = 3)
  lp <- sapply(draws,logProposalDensityFunc)
  lt <- sapply(draws,logTargetDensityFunc)
  wts <- exp(lt - lp)
  if(rejectionControlConstant == FALSE){
    Awts <- wts/sum(wts)
    ESS <- 1/sum(Awts^2)
    return(list(draws, log(Awts), ESS))
  } else{
    c <- rejectionControlConstant
    unifsamp <- runif(nSamples, 0 ,1)
    prob <- pmin(rep(1,nSamples), wts/c)
    samps <- unifsamp <= prob
    accept_rate <- sum(samps)/nSamples
    accepted_draws <- draws[samps]
    Awts <- wts[samps]/prob[samps]
    ESS <- sum(Awts)^2/sum(Awts^2)
    return(list(accepted_draws,log(Awts/sum(Awts)), accept_rate, ESS))
    }
  }
```

For the effective sample size calculation we report the quantity,

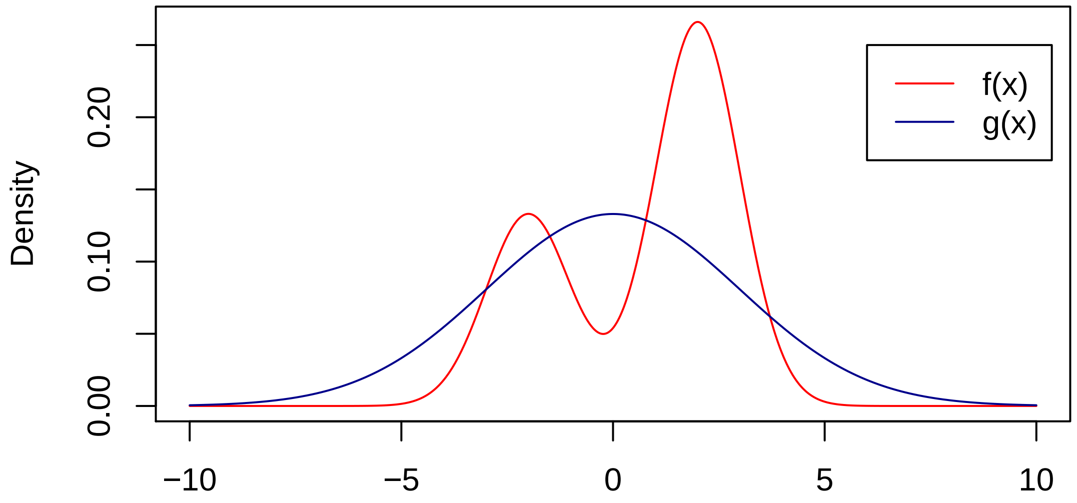
$$S_{eff} = \frac{\left(\sum_{s=1}^{S} w(\theta^s)\right)^2}{\sum_{s=1}^{S} w^2(\theta^s)}$$

We return normalized log weights , the samples corresponding to those weights, and when rejection control is implemented, the acceptance rate.

b/c) For parts (b) and (c) we consider

$$
\begin{aligned}
f(x) &= \frac{1}{3}N(x|-2,1^2) + \frac{2}{3}N(x|2,1^2) \\
g(x) &= N(x|0,3)
\end{aligned}
$$

Where $N(x|\mu,\sigma^2)$ is the normal density function with mean μ and variance σ^2, and $g(x)$ is the proposed importance density function. We plot them on the same graph below.



We notice $f(x)$ is concentrated within the high density areas of $g(x)$, indicating that $g(x)# is a reasonable importance density function.

d) We have

$$
\begin{aligned}
E[X] &= \int xf(x)dx \\
&= \frac{1}{3}\int xN(x|-2,1^2)dx + \frac{2}{3}\int xN(x|2,1^2)dx \\
&= \frac{-2}{3} + \frac{4}{3} \\
&= \frac{2}{3}
\end{aligned}
$$

Similarly, we can show $E[X^2] = 5$. We also have

$$
\begin{aligned}
E\left[e^X\right] &= \int e^x f(x)dx \\
&= \frac{1}{3}\int e^x N(x|-2,1^2)dx + \frac{2}{3}\int e^x N(x|2,1^2)dx \\
&= \frac{1}{3}M_{N(-2,1)}(t)|_{t=1} + \frac{2}{3}M_{N(2,1)}(t)|_{t=1} \\
&= \frac{1}{3}e^{-2+1/2} + \frac{2}{3}e^{2+1/2} \\
&\approx 8.2
\end{aligned}
$$

e) We now use the function we wrote in part (a) to obtain estimates for the quantities above.

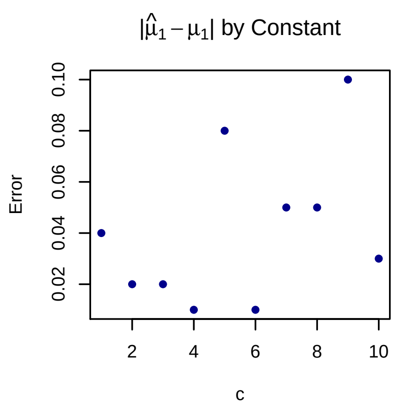
```
## [1] "Mu1 estimate: 0.6799"

## [1] "Mu2 estimate:  5.02"

## [1] "Theta estimate:  8.27"
```

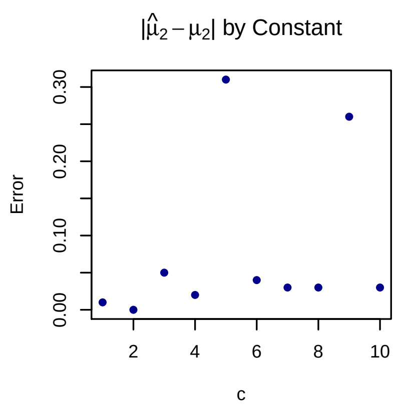
These compare closely to our theoretical calculations for each of the quantities.

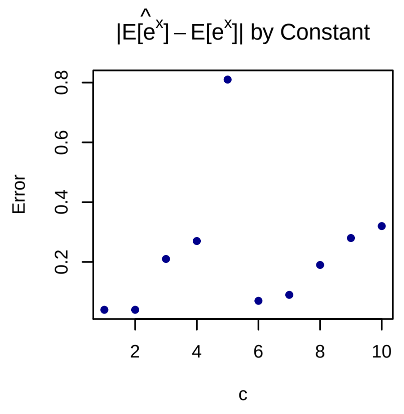
f) In this question, we include a rejection control constant, varying it from 1 to 10. At each level, we estimate μ_1, μ_2 and θ. We share our observations on the error in the estimators, acceptance rate and ESS.

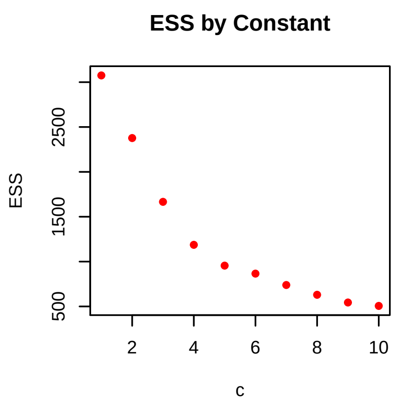
Table 1: Results of Importance Sampling

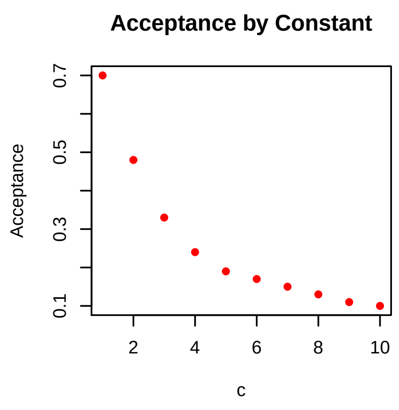
c	ESS	Acceptance	$\lvert\hat{\mu}_1 - \mu_1\rvert$	$\lvert\hat{\mu}_2 - \mu_2\rvert$	$\lvert\hat{\theta} - \theta\rvert$
1	3075	0.70	0.04	0.01	0.04
2	2377	0.48	0.02	0.00	0.04
3	1666	0.33	0.02	0.05	0.21
4	1187	0.24	0.01	0.02	0.27
5	955	0.19	0.08	0.31	0.81
6	866	0.17	0.01	0.04	0.07
7	739	0.15	0.05	0.03	0.09
8	630	0.13	0.05	0.03	0.19
9	544	0.11	0.10	0.26	0.28
10	506	0.10	0.03	0.03	0.32











Based on the plots above, we recommend that c be either chosen to be 6 or 2. These rejection control constants produce low errors for each estimated quantity, while maintaining effective sample size around 1000 or above.

BDA Question 10.5

a) In accordance with the guidelines in the question, we sample a data set at random from the model. For our covariates we make 10 draws from a $U(0,1)$, pull an α and β value from their prior distributions, obtain 10 n values corresponding to each y_j and sample $J = 10$ values from the corresponding binomial distributions to obtain our response.

Table 2: Random Data Set from Model

y	n	θ	x	α	β
3	9	0.31	0.27	-0.72	-0.35
1	3	0.30	0.37	-0.72	-0.35
2	6	0.28	0.57	-0.72	-0.35
0	3	0.26	0.91	-0.72	-0.35
2	4	0.31	0.20	-0.72	-0.35
1	4	0.26	0.90	-0.72	-0.35
1	1	0.26	0.94	-0.72	-0.35
0	4	0.28	0.66	-0.72	-0.35
3	8	0.28	0.63	-0.72	-0.35
1	4	0.32	0.06	-0.72	-0.35

b) We can write the posterior of (α, β) as follows,

$$
\begin{aligned}
p(\alpha, \beta|y) &\propto p(\alpha, \beta)p(y|\alpha, \beta) \\
&\propto \left(1 + \frac{1}{4}\left(\frac{\alpha}{2}\right)^2\right)^{\frac{-5}{2}} \left(1 + \frac{\beta^2}{4}\right)^{\frac{-5}{2}} \prod_{j=1}^{10} \text{logit}^{-1}(\alpha + \beta x_j)^{y_j}(1 - \text{logit}^{-1}(\alpha + \beta x_j))^{n_j - y_j}
\end{aligned}
$$

Using rejection sampling we draw 1000 samples from this distribution.

Table 3: Rejection Sampling Summary

	Mean	SD	0.025	0.50	0.975
α	-0.6670976	0.4849987	-1.601930	-0.6655289	0.2911486
β	-0.3522671	0.7913760	-1.941706	-0.3384857	1.0904952

c) The posterior density for $(\alpha,\ \beta)$ has a mode at (-0.658, 0.300). We include the curvature at the mode as the covariance matrix for the Normal approximation so that

$$
p(\alpha, \beta|y) \overset{approx}{\sim} N\left(\begin{pmatrix} -0.658 \\ 0.300 \end{pmatrix}, \begin{pmatrix} 0.217 & -0.247 \\ -0.247 & 0.518 \end{pmatrix}\right)
$$

d/e) As instructed in the question, we use importance sampling to estimate $E(\alpha|y)$ and $E(\beta|y)$, and report the effective sample size as given in the textbook.

```
## [1] "E(\alpha|y) =  -0.648"
```

```
## [1] "E(\beta|y) =  -0.375"
```

```
## [1] "ESS =  506"
```

These are similar to our estimates using rejection sampling.

BDA Question 10.8

a) For the bioassay example given in section 3.7, we are given that

$$
\begin{aligned}
y_i|\theta_i &\sim \text{Bin}(n_i, \theta_i) \\
\text{logit}(\theta_i) &= \alpha + \beta x_i \\
p(\alpha, \beta) &\propto 1
\end{aligned}
$$

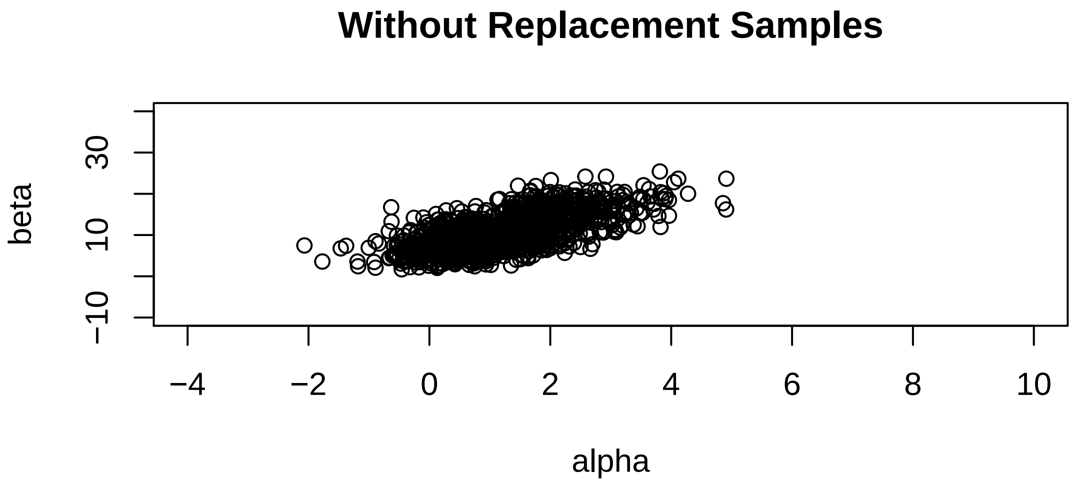
The posterior is given as

$$
\begin{aligned}
p(\alpha, \beta|y) &\propto p(\alpha, \beta)p(y|\alpha, \beta) \\
&\propto \prod_{i=1}^{4} \text{logit}^{-1}(\alpha + \beta x_i)^{y_i}(1 - \text{logit}^{-1}(\alpha + \beta x_i))^{n_i - y_i}
\end{aligned}
$$

since there are four dosing levels (x_i). Introducing the data in Table 3.1, we can use the `laplace` function from the `LearnBayes` package to determine the posterior mode, and estimate of the covariance matrix for the normal approximation. This method gives us $(\hat{\alpha}, \hat{\beta}) = (0.85, 7.75)$ and an estimated covariance matrix

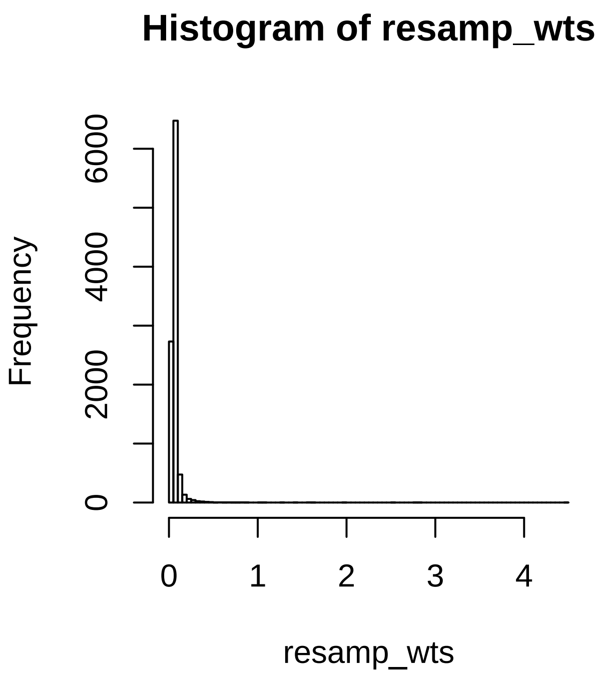
$$\mathrm{Var}(\alpha, \beta) = \begin{pmatrix} 1.04 & 3.55 \\ 3.55 & 23.76 \end{pmatrix}$$

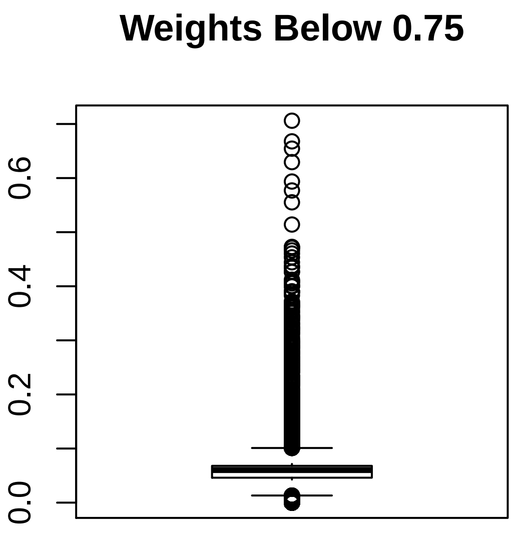
We can use these as the mean and covariance matrix for the normal approximation to the posterior. We implement importance resampling, using 10,000 samples from the approximate distribution and resample without replacement $k = 1000$ samples.



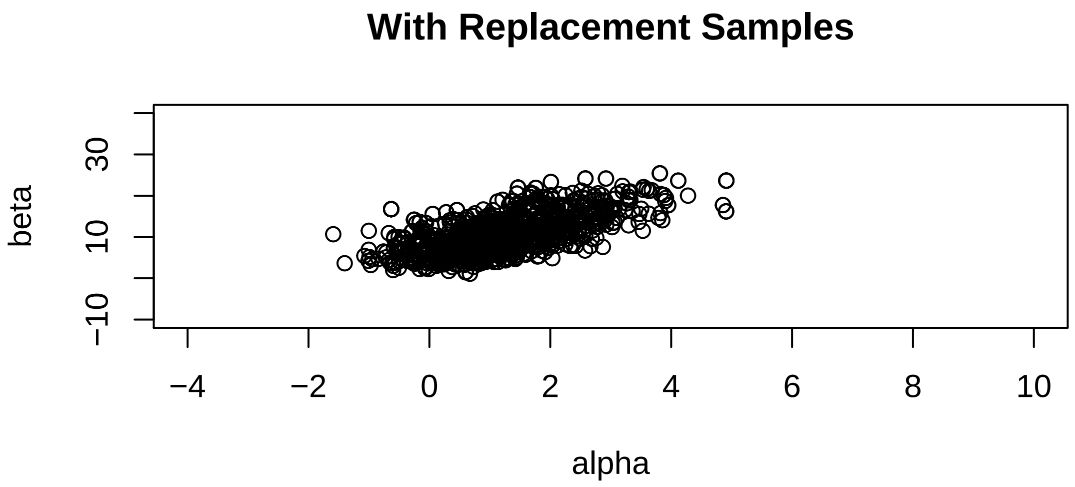
When we compare this to figure 3.3b, we find that our samples take values on a smaller range, but the shape of the distribution is approximately the same. This is to be expected, as our samples are limited to those taken from our approximating distribution.

b) Below, we plot a histogram of resampled weights, and a box plot of their distribution, cut off above 0.75, so it is easier to see. A majority of the weights are concentrated around 0, with some outliers ranging up to four.





c) We now conduct the resampling with replacement, and plot the samples.



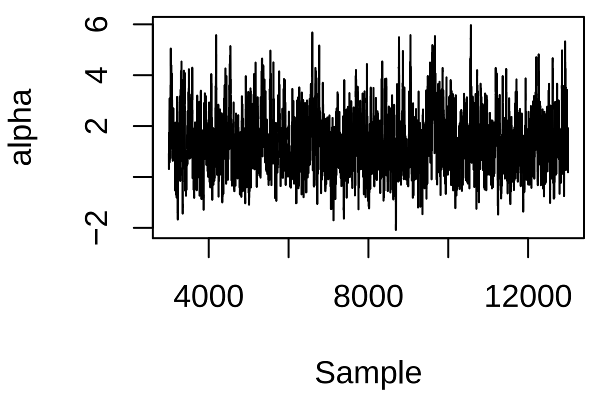
We find that there is not much of a difference when we sample with or without replacement in this case. This is likely because there are not enough large weights to significantly alter the sample, or the fact that we are only choosing 1,000 samples from the 10,000 we pulled from the approximating distribution allows for enough variety that we do not see large discrepancies between the two methods.

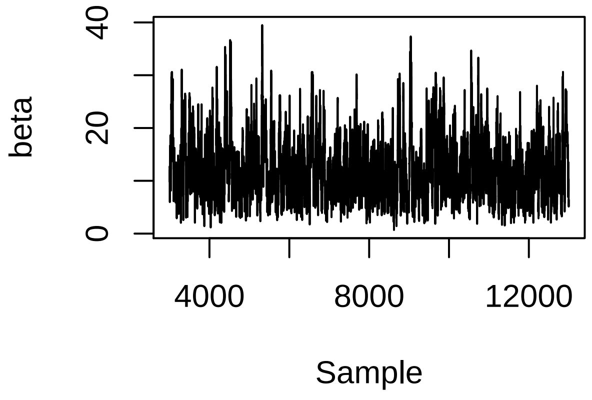
Question 11.2

In this question we are asked to implement the Metropolis algorithm for the bioassay example of section 3.7. Our starting points will be the MLE values presented in the book, namely $(\hat{\alpha}, \hat{\beta}) = (0.8, 7.7)$. Our jumping distribution will be

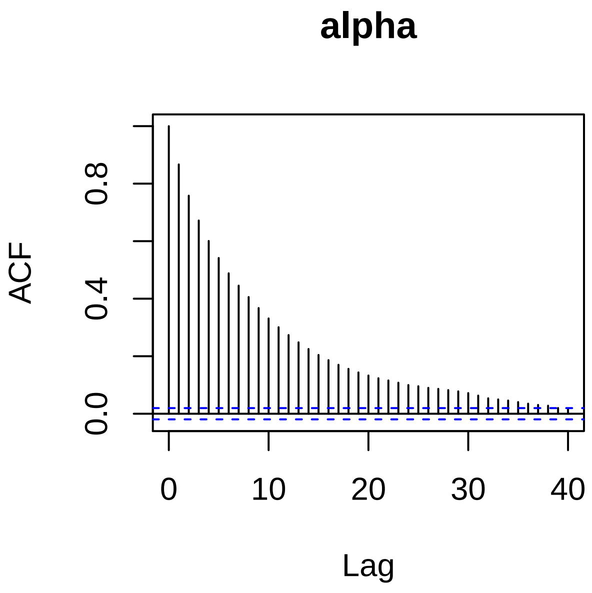
$$(\alpha^{(t+1)}, \beta^{(t+1)}) \sim U(\alpha^{(t)} - 0.5, \alpha^{(t)} + 0.5) \times U(\beta^{(t)} - 2, \alpha^{(t)} + 2)$$

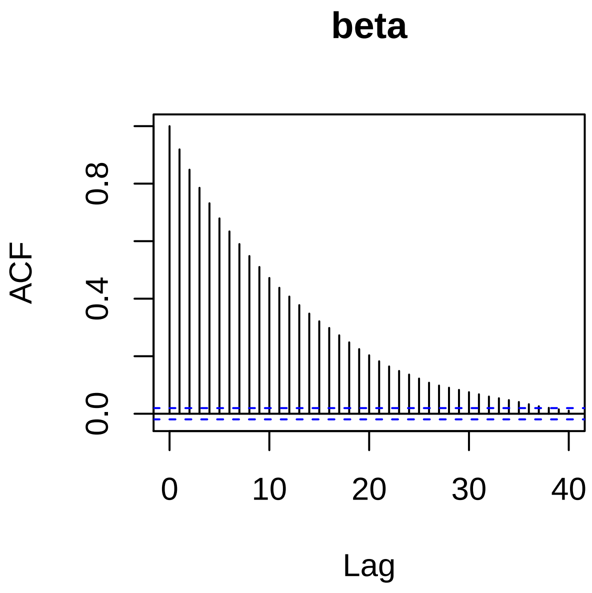
the uniform square centered at the previously sampled value.



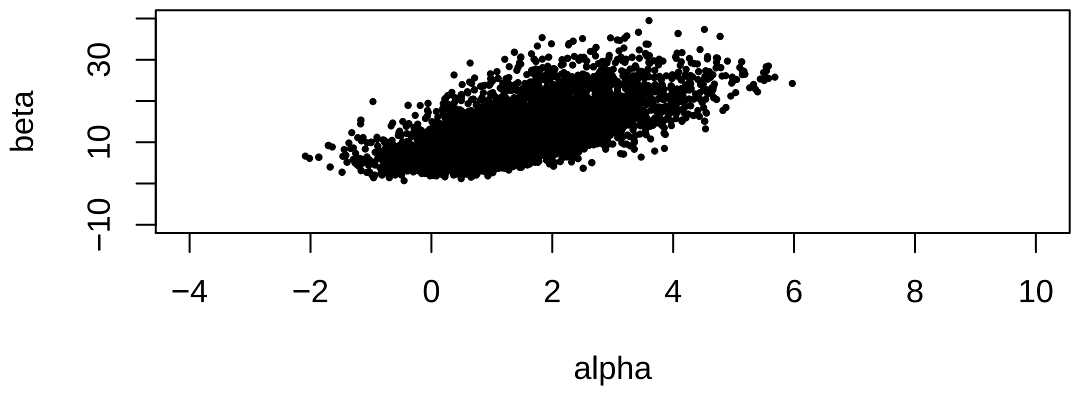


The trace plots above show good mixing for the 10,000 samples (post burn-in) for both α and β. To be thorough, we examine the acf plots of our chains to ensure that the autocorrelation goes to zero quickly. As the plots below demonstrate, this is the case.

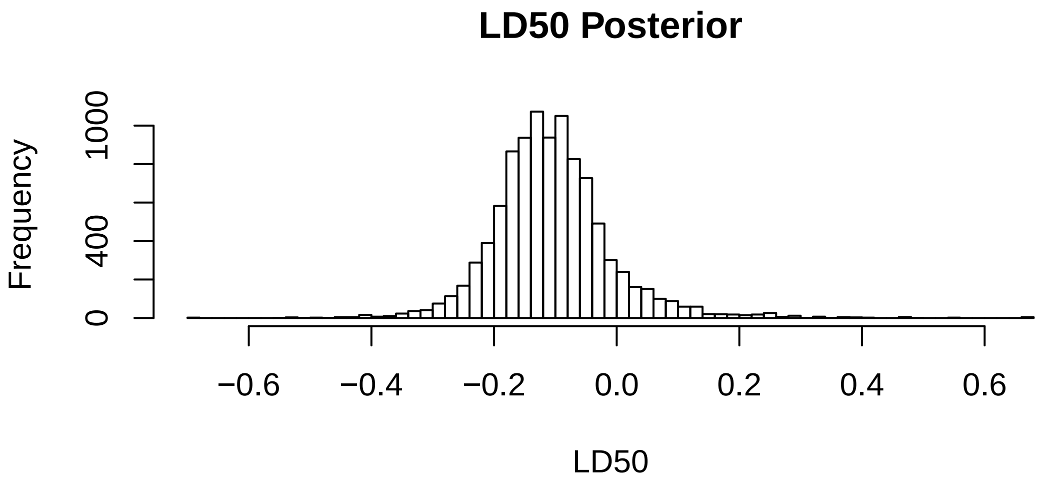




We now plot the samples drawn using the MH algorithm. Notice, this plot is almost exactly what we find in Figure 3.3(b)



We can also obtain the posterior distribution of $LD50$ as they did in Section 3.7.



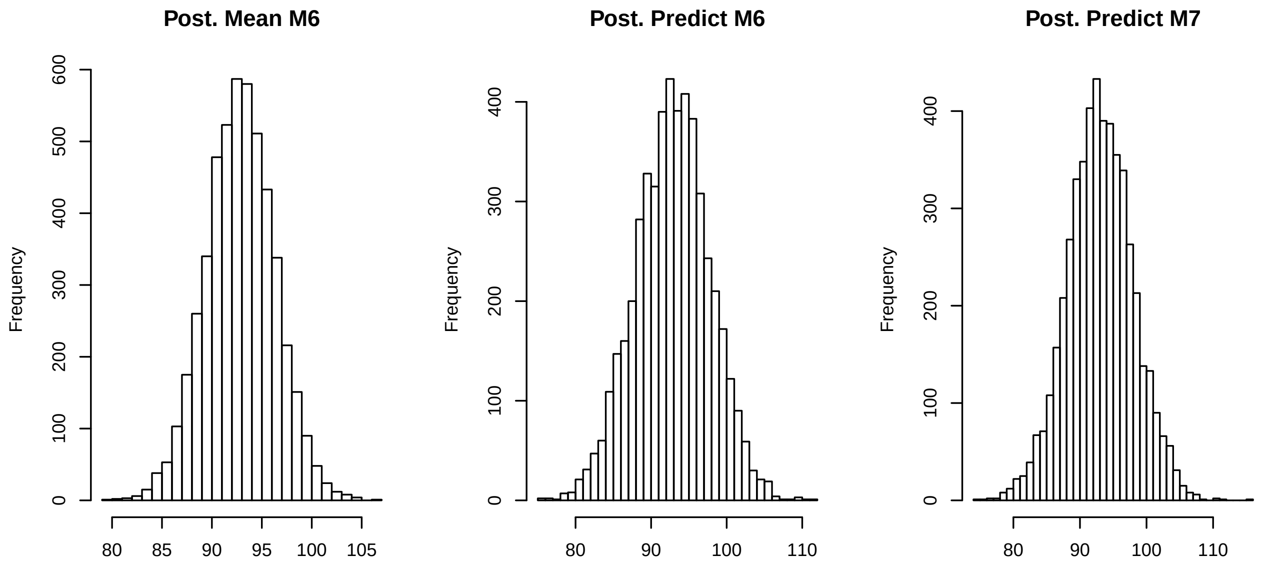
Again, this is exactly what we find in figure 3.4 from the text.

BDA Question 11.3

In this question, we are asked to implement Gibbs sampling for the quality control measurements found in Table 11.4. We do this using a separate, a pooled and a hierarchical Gaussian model with common variance. We begin with the pooled model, where there is one mean θ applied for each machine. Let $n = 5$ be the number of measurements being taken for each machine. In this case, we have

$$\begin{aligned} y_{ij} &\sim N(\theta, \sigma^2) \\ \theta|\sigma^2, y_1, ..., y_6 &\sim N\left(\bar{y}, \frac{\sigma}{6n}^2\right) \\ \sigma^2|y_1, ..., y_6 &\sim \text{Scaled-Inv}\chi^2\left(6n-1, \frac{\sum_{j=1}^6 \sum_{i=1}^n (y_{ij}-\bar{y})^2}{6n-1}\right) \end{aligned}$$

In order to obtain a distribution of the posterior mean of machine 6, we can sample from the posterior for σ^2, and for each sample, draw a value of θ from its posterior. Since each machine has the same θ value, we are not concerned with distinguishing between them. Similarly, if we want to obtain predictive values for machine 6, we follow the same process and draw a new y value for each θ and σ^2 we obtain from their posteriors. The same process can be followed to obtain a posterior predictive distribution for machine 7, since we would assume it follows the same distribution as all the other machines. Each of these plots is presented below.

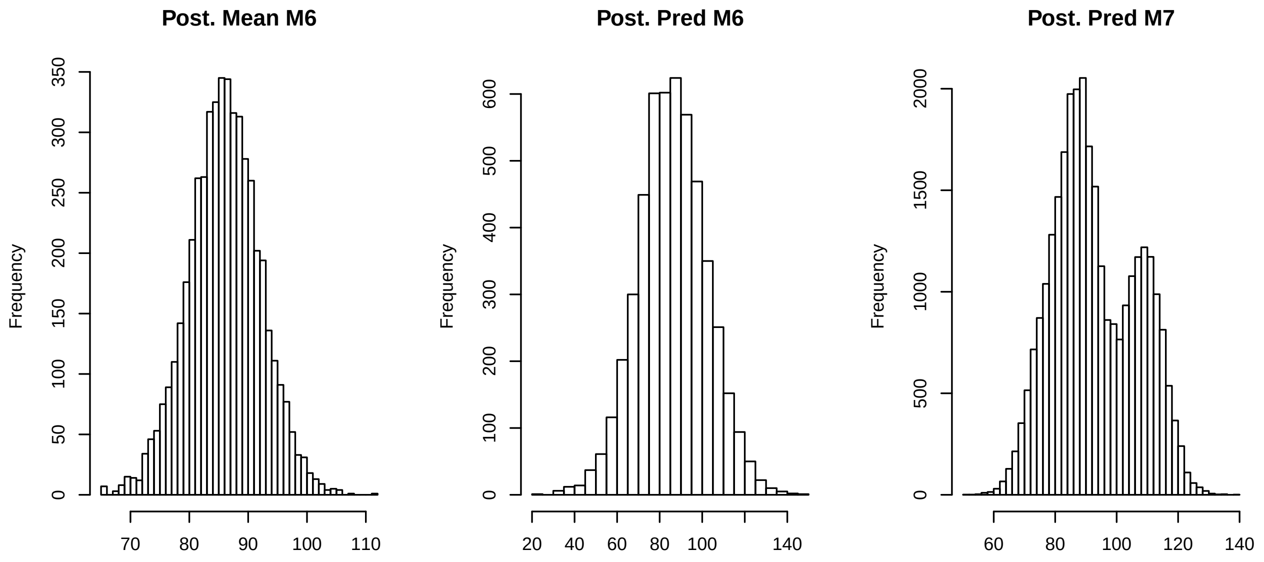


The posterior mean for machine 6 is centered around 92.5, as are the posterior predictive distributions for both machine 6 and 7, as we would expect. Since we are drawing samples for y in the posterior predictive distributions for each machine, there is more variation in the samples.

We now consider the separate model, where each θ_j comes from its own normal distribution.

$$\begin{aligned} y_{ij} &\sim N(\theta_j, \sigma^2) \\ \theta_j|\sigma^2, y_j &\sim N\left(\bar{y}_{.j}, \frac{\sigma^2}{n}\right) \\ \sigma^2|y_1, ..., y_6 &\sim \text{Scaled-Inv}\chi^2\left(\sum_{j=1}^{6}(n_j-1), \frac{\sum_{j=1}^{6}\sum_{i=1}^{n}(y_{ij}-\bar{y}_{.j})^2}{\sum_{j=1}^{6}(n_j-1)}\right) \end{aligned}$$

Note, since we assume the distributions of the θ's have unique means, we use the pooled sample variance for $\hat{\sigma}$ in $\sigma^2|y_1, ..., y_6$. In order to get posterior samples of the mean for machine 6, we can sample from the posterior for σ^2, and for each sample draw from $\theta_6|\sigma^2, y_6$. To obtain samples from the posterior predictive distribution for machine 6, we can follow a similar procedure , except for each draw from $\theta_6|\sigma^2, y_6$, we sample from $N(\theta_6, \sigma^2)$. To obtain a posterior predictive distribution for machine 7, we can just pool the posterior predictive samples we obtain from machine 1 through 6. These plots are displayed below.



The results above show the posterior mean of machine 6 to be centered around 87. The posterior predictive distribution for machine 7 is a mixture between the 6 posterior predictive distributions from the machines in the data set.

Now we move to the hierarchical model. The group means are assumed to follow a normal distribution with unknown mean μ and variance τ^2, and a uniform prior distribution is assumed for $(\mu, \tau, \log\sigma)$ so that $p(\mu, \log\sigma, \log\tau) \propto \tau$. Chapter 5 of BDA demonstrated that this is necessary, or the posterior would be improper. We can write the joint posterior as follows

$$p(\theta, \mu \log\sigma, \log\tau) \propto \tau \prod_{j=1}^{6} N(\theta_j|\mu, \tau) \prod_{j=1}^{6}\prod_{i=1}^{5} N(y_{ij}|\theta_j, \sigma^2)$$

In the Gibbs Sampler, we will use the following,

$$\begin{aligned}
\hat{\sigma}^2 &= \frac{\sum_{j=1}^{6}\sum_{i=1}^{n}(y_{ij} - \theta_j)}{6n} \\
\hat{\mu} &= \frac{1}{6}\sum_{i=1}^{6}\theta_j \\
\hat{\tau}^2 &= \frac{1}{5}\sum_{j=1}^{6}(\theta_j - \mu)^2
\end{aligned}$$

As is demonstrated in Chapter 11,

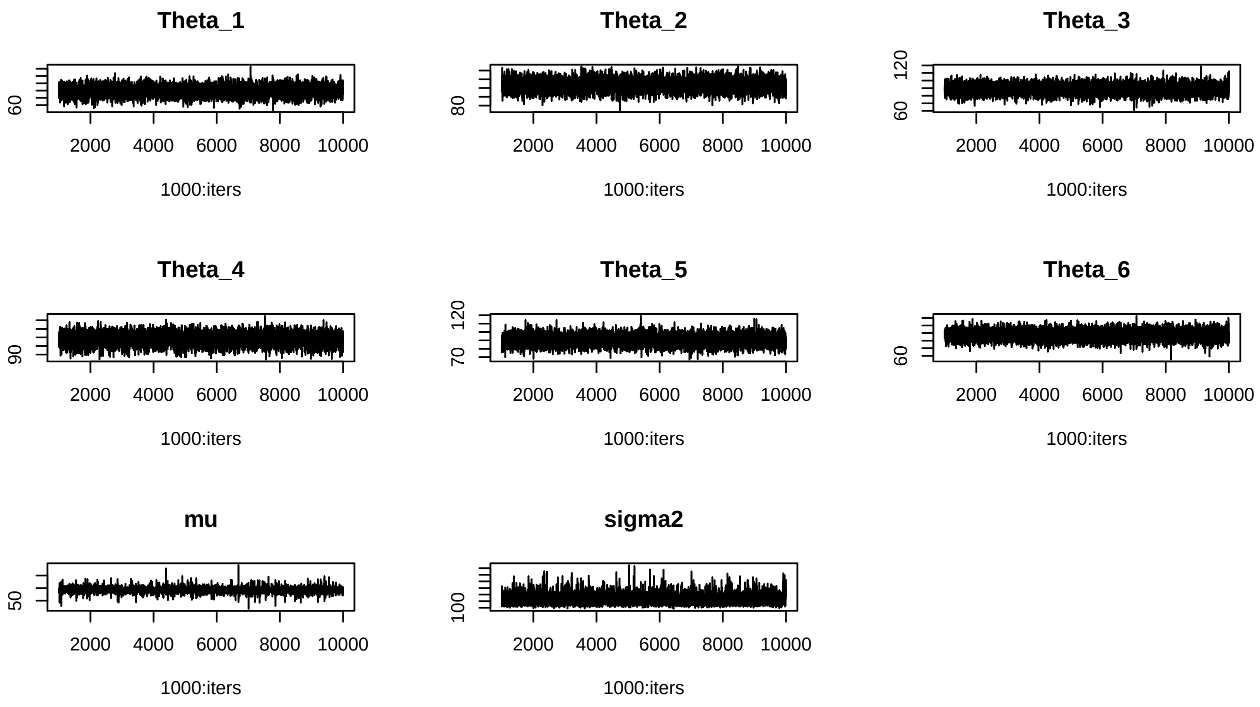
$$\begin{aligned}
\tau^2|\theta, \mu, \sigma, y &\sim \text{Inv}\chi^2(5, \hat{\tau}^2) \\
\mu|\theta, \sigma, \tau, y &\sim N(\hat{\mu}, \tau^2/5) \\
\sigma^2|\theta, \mu, \tau, y &\sim \text{Inv}\chi^2(6n, \hat{\sigma}^2) \\
\theta_j|\mu, \sigma, \tau, y &\sim N(\hat{\theta}_j, V_{\theta_j})
\end{aligned}$$

where

$$\hat{\theta}_j = \frac{\frac{1}{\tau^2}\mu + \frac{n_j}{\sigma^2}\bar{y}_{\cdot j}}{\frac{1}{\tau^2} + \frac{n_j}{\sigma^2}}$$

$$V_{\theta_j} = \frac{1}{\frac{1}{\tau^2} + \frac{n_j}{\sigma^2}}$$

To obtain starting values for the algorithm, we randomly sample values y_{ij} from group j. These will act as the initial $\hat{\theta}_j$ values. The rest of the estimates can be filled in using these values and the data. The plots for this model are displayed below.



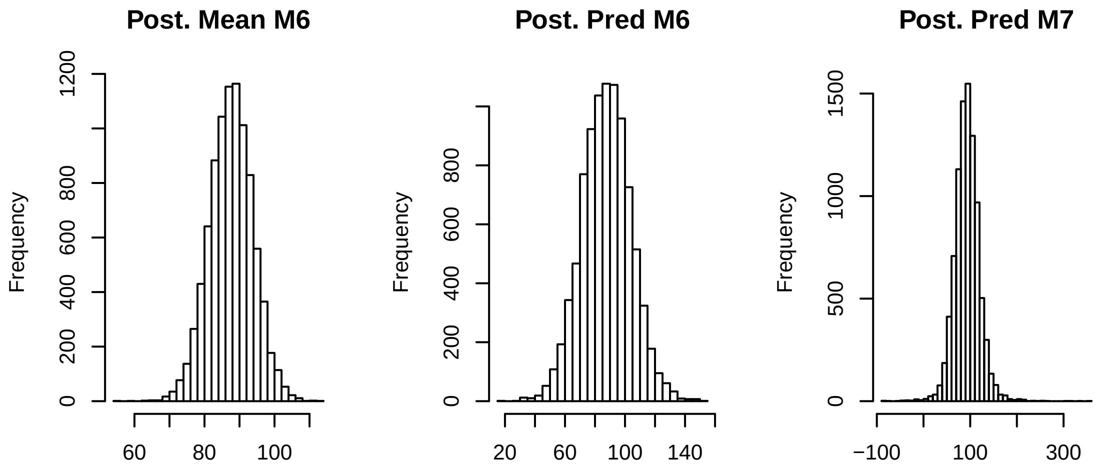


Table 4: Rejection Sampling Summary

	Post_Mean_M6	Post_Pred_M6	Post_Pred_M7
Mean	87.60	87.54	93.14
SD	6.16	16.28	26.83
0.025	75.43	55.53	43.77
0.5	87.64	87.66	92.67
0.975	99.63	119.45	146.26

The posterior mean of machine 6 is centered around 87.5 , as is the posterior predictive distribution, with a larger standard deviation. The posterior predictive distribution of machine 7 is centered around 93.1 with with an even larger standard deviation as we might expect. The mixing of the draws was good as well, as can be seen in the trace plots.

BDA Question 11.4

For this question we are asked to extend the hierarchical model in 11.3 so that the variances between groups may vary. We use and Inv-χ^2 proper distribution for the variances with unknown scale σ_0^2 and fixed degrees of freedom ν. The posterior takes the following form

$$p(\theta, \sigma^2, \mu, \log\tau, \sigma_0^2|y) \propto p(\sigma_0^2)\tau \prod_{j=1}^{6}\prod_{i=1}^{n} N(y_{ij}|\theta_j, \sigma_j^2) \prod_{j=1}^{6} N(\theta_j|\mu, \tau) \prod_{j=1}^{6} \text{Inv-}\chi^2(\sigma_j^2|\nu, \sigma_0^2)$$

This means that

$$\begin{aligned}
p(\sigma_j^2|\theta, \mu, \log\tau, \sigma_0^2, y) &\propto \text{Inv-}\chi^2(\sigma_j^2|\nu, \sigma_0^2)\prod_{i=1}^{n} N(y_{ij}|\theta_j, \sigma_j^2) \\
&\propto \left(\sigma_j^2\right)^{-(\nu/2+1)} e^{-\frac{\nu\sigma_0^2}{2\sigma_j^2}} \left(\sigma_j^2\right)^{-n/2} e^{\frac{-\sum_{i=1}^{n}(y_{ij}-\theta_j)^2}{2\sigma_j^2}} \\
&\propto \left(\sigma_j^2\right)^{-((n+\nu)/2+1)} \exp\left\{-\frac{\nu\sigma_0^2}{2\sigma_j^2} - \frac{\sum_{i=1}^{n}(y_{ij}-\theta_j)^2}{2\sigma_j^2}\right\} \\
&\propto \left(\sigma_j^2\right)^{-((n+\nu)/2+1)} \exp\left\{-\frac{(n+v)\left(\frac{\sum_{i=1}^{n}(y_{ij}-\theta_j)^2/n}{\nu} + \frac{\sigma_0^2}{n}\right)}{2\sigma_j^2}\right\} \\
&\propto \text{Inv-}\chi^2\left(n+v, \frac{\sum_{i=1}^{n}(y_{ij}-\theta_j)^2/n}{\nu} + \frac{\sigma_0^2}{n}\right)
\end{aligned}$$

We also have,

$$\begin{aligned}
p(\sigma_0^2|\theta,\mu,\log\tau,\sigma_j^2,y) &\propto p(\sigma_0^2)\prod_{j=1}^{6}\text{Inv-}\chi^2(\sigma_j^2|\nu,\sigma_0^2)\\
&\propto p(\sigma_0^2)\prod_{j=1}^{6}(\sigma_0)^{\nu}\,e^{-\frac{\nu\sigma_0^2}{2\sigma_j^2}}\\
&\propto p(\sigma_0^2)\,(\sigma_0)^{6\nu}\exp\left\{-\frac{\nu\sigma_0^2}{2}\sum_{j=1}^{6}\frac{1}{\sigma_j^2}\right\}\\
&\propto p(\sigma_0^2)\left(\sigma_0^2\right)^{3\nu}\exp\left\{-\sigma_0^2B\right\}
\end{aligned}$$

where $B=\frac{\nu\sigma_0^2}{2}\sum_{j=1}^{6}\frac{1}{\sigma_j^2}$. If $p(\sigma_0^2)\propto 1/\sigma_0^2$, the uniform prior on the log scale, this last integral is finite, so we use this. For simplicity, we let $\nu=2$. Also, we note that

$$\theta_j|\mu,\sigma_j^2,\tau,\sigma_0^2,y\sim N(\hat{\theta}_j,V_{\theta_j})$$

where

$$\begin{aligned}
\hat{\theta}_j &= \frac{\frac{1}{\tau^2}\mu+\frac{n}{\sigma_j^2}\bar{y}_{.j}}{\frac{1}{\tau^2}+\frac{n_j}{\sigma_j^2}}\\
V_{\theta_j} &= \frac{1}{\frac{1}{\tau^2}+\frac{n_j}{\sigma_j^2}}
\end{aligned}$$

We add these distributions to our Gibbs sampler from the previous problem, and present the results below.

Table 5: Gibbs Sample Summary

	Mean	SD	0.025	0.5	0.975
theta1	79.59	7.28	65.98	79.26	95.50
theta2	104.90	4.92	94.54	105.09	114.19
theta3	88.53	5.02	78.54	88.42	98.84
theta4	110.43	3.82	101.92	110.63	117.32
theta5	90.37	4.33	81.73	90.33	99.38
theta6	87.51	6.54	74.81	87.34	101.04
sigma1	16.09	5.64	9.00	14.91	30.32
sigma2	10.46	3.69	5.84	9.68	19.75
sigma3	11.06	3.74	6.21	10.22	20.52
sigma4	7.80	2.86	4.20	7.18	15.26
sigma5	9.68	3.39	5.40	8.96	18.16
sigma6	15.44	5.31	8.86	14.28	28.75
mu	93.67	9.55	74.76	93.72	111.71
tau	18.06	10.28	7.34	15.44	45.86
sig0	9.40	2.44	5.35	9.15	14.88

The distribution for θ_6 is similar to what we saw in the previous example. The standard deviations of the distributions range from a mean of 7.8 up to 16. This variation indicates that it might be worth using this model rather than maintaining a fixed variance between distributions. The overall μ is centered around 93, similar to what we saw in the previous question.

BDA Question 13.5

a) We are asked to derive the distribution $p(x_i|\alpha,\beta)$. We have

$$\begin{aligned} p(x_i|\alpha,\beta) &= \int p(x_i|\lambda_i)p(\lambda_i|\alpha,\beta)d\lambda_i \\ &\propto \frac{1}{x!}\int \lambda_i^{x_i}e^{-\lambda_i}\lambda_i^{\alpha-1}e^{-\beta\lambda_i}d\lambda_i \\ &\propto \frac{1}{x!}\int \lambda_i^{\alpha+x_i-1}e^{-(\beta+1)\lambda_i}d\lambda_i \\ &\quad \text{Kernel of Gam}(\alpha+x_i,\beta+1) \\ &\propto \frac{\Gamma(x_i+\alpha)}{x!\,(\beta+1)^{x_i+\alpha}} \\ &\propto \frac{\Gamma(x_i+\alpha)}{\Gamma(x_i+1)}\left(\frac{1}{\beta+1}\right)^{x_i} \\ &\quad \text{Kernel of NegBin}\left(\alpha,\frac{\beta}{\beta+1}\right) \end{aligned}$$

Here, we use the parameterization that the random variable represents the number of failures before success α occurs when the success probability is $\beta/(\beta+1)$.

b) Assume that N total species are considered. Let \$ y = (y__1, y__2,. . .) \$ be a multinomial distribution where y_k , $k = 1,2,3...$ represents the number of these N species that were observed exactly k times. Recall that x_i represents the number of animals of the i^{th} species that were observed. In part (a) we showed that

$$x_i|\alpha,\beta \sim \text{NegBin}\left(\alpha,\frac{\beta}{\beta+1}\right)$$

Notice, for each of the N species, the parametrization is is the same. From this, we can construct a multinomial distribution so that

$$\begin{aligned} P(y=(y_1,y_2,...)) &= \frac{N!}{y_1!y_2!....}\,[P(x_i=1)]^{y_1}\,[P(x_i=2)]^{y_2}\,...\ \text{ where}\sum_{i=1}^{\infty}y_i=N \\ &= \frac{N!}{y_1!y_2!....}\left[\begin{pmatrix} 1+\alpha-1 \\ 1 \end{pmatrix}\left(\frac{\beta}{\beta+1}\right)^{\alpha}\left(\frac{1}{\beta+1}\right)\right]^{y_1}\left[\begin{pmatrix} 2+\alpha-1 \\ 2 \end{pmatrix}\left(\frac{\beta}{\beta+1}\right)^{\alpha}\left(\frac{1}{\beta+1}\right)^{2}\right]^{y_2}... \\ &\text{where}\sum_{i=1}^{\infty}y_i=N \end{aligned}$$

The above multinomial distribution is technically infinite dimensional, but the number of trials is restricted to N. Notice, the probability of observing the k^{th} category of this the multinomial distribution is just $P(X=k)$ where X is a negative binomial random variable. Let π_k denote this probability. We have

$$\sum_{i=1}^{\infty} \pi_k = \sum_{i=1}^{\infty} P(X = k)$$
$$= 1$$

as needed for the requirements of the multinomial distribution.

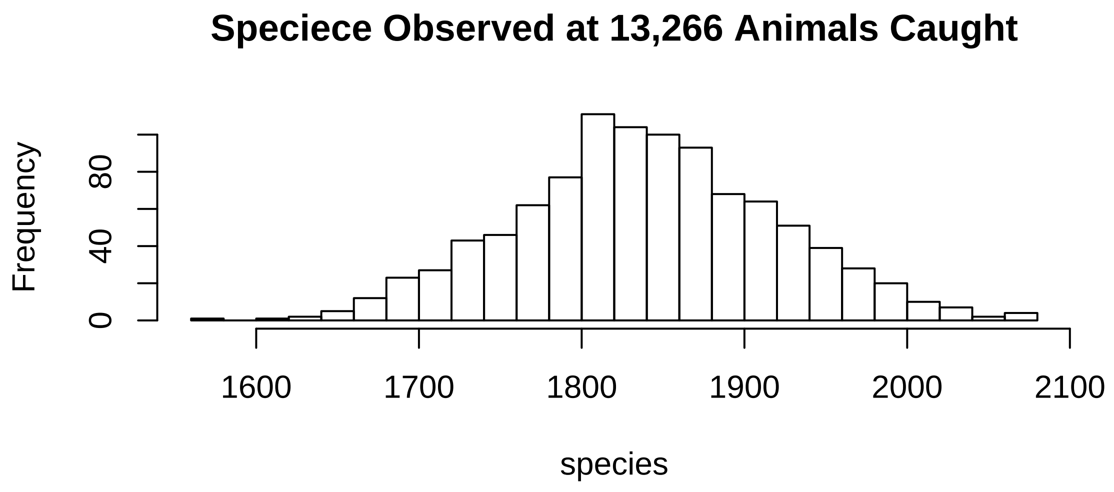
c) The likelihood for (α,β) given the data can be written,

$$L(\alpha, \beta|y) \propto \left[\begin{pmatrix} 1+\alpha-1 \\ 1 \end{pmatrix} \left(\frac{\beta}{\beta+1}\right)^{\alpha} \left(\frac{1}{\beta+1}\right) \right]^{118} \left[\begin{pmatrix} 2+\alpha-1 \\ 2 \end{pmatrix} \left(\frac{\beta}{\beta+1}\right)^{\alpha} \left(\frac{1}{\beta+1}\right)^{2} \right]^{74} \cdots$$
$$\cdots \left[\begin{pmatrix} 24+\alpha-1 \\ 1 \end{pmatrix} \left(\frac{\beta}{\beta+1}\right)^{\alpha} \left(\frac{1}{\beta+1}\right)^{24} \right]^{3}$$

We can stop at the 24th multinomial cell because the rest are observed as 0's. We find the mode of this likelihood using a Laplace approximation with the `laplace` function in the `LearnBayes` package. We have mode $(\alpha, \beta) = (1.37, 0.21)$ and

$$\text{Var}(\alpha, \beta) \approxeq \begin{pmatrix} 0.01 & 0.0016 \\ 0.0016 & 0.000315 \end{pmatrix}$$

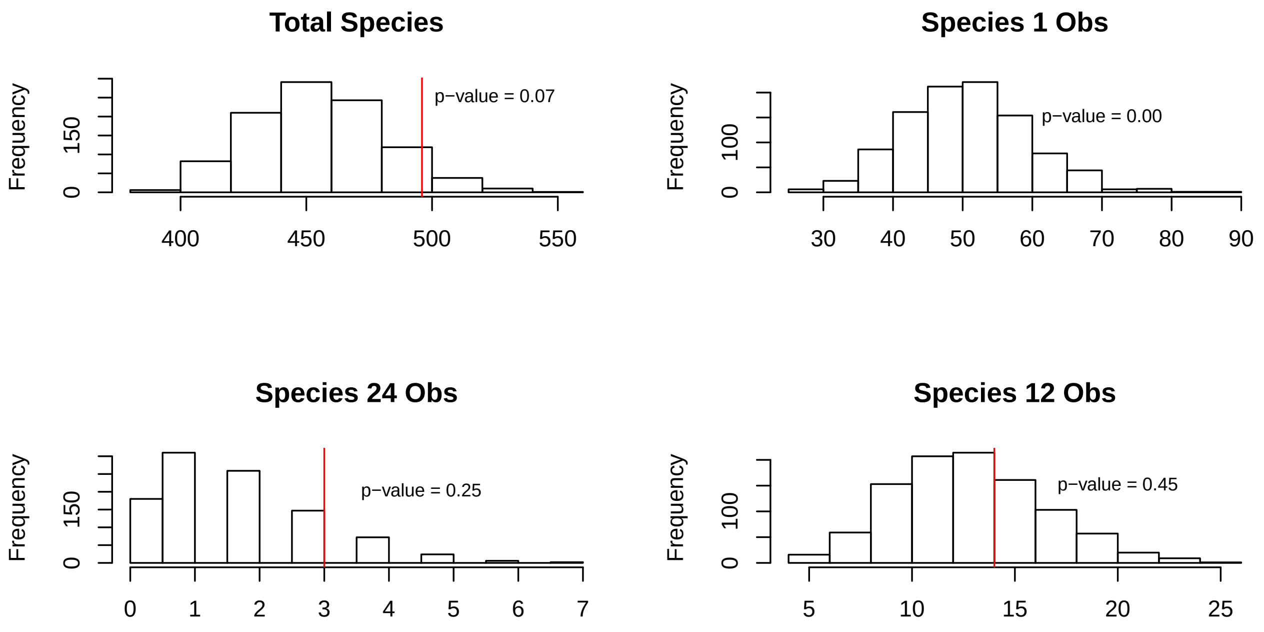
d) Our goal is to obtain a 95% posterior interval for the number of additional species we were to catch if we observed 10,000 more animals. This would require us to observe a total of 13,266 animal. By properties of the multinomial distribution, if $y_i \sim \text{Mult}(1, \pi)$, then $\sum_{i=1}^{N} y_i \sim \text{Mult}(N, \pi)$. In our case, N represents the number of species. So, in order to obtain the number of species we would observe if we catch a total of 13,266 animals, we can add $y_i \sim \text{Mult}(1, \pi)$ where π is our Negative Binomial probability vector at a posterior sample of (α, β), until the total number of animals observed is 13,266. We can check how many animals have been observed at a given point by multiplying the cell value (number of species), by cell number (number of times the species were observed), for each entry in the multinomial vector, and summing them up. This sum will tell us the total number of observed species. We can stop adding additional single observations of the multinomial vector (corresponding to additional species) when we meet the 13,266 animal threshold. In order to obtain the posterior interval, we can sample from the posterior for (α, β) using importance resampling, and collect the number of species it requires for us to reach 13,266 animals for 1000 iterations, giving us a distribution of total species we would see were we to have caught 13,266 total animals. We conduct the simulation and display the results below.



```
##  2.5%   50% 97.5%
##  1689  1839  1998
```

The distribution has a median at 1838 with 95% credible interval [1689, 1998].

e) In order to conduct posterior predictive checks, we run the same simulation, but restrict our observations to 3266 (what was actually observed). We evaluate the performance of the model below.



The above plots show the posterior predictive distribution for (1) total number of species observed when 3266 animals are caught, and of those animals, (2) how many were caught once, the smallest observed number, (3) how many were caught 24 times, the max observed number, and (4) how many were caught 12 times, the median observed number. The posterior predictive distribution for total species shows that the data is not being captured very well by the model. While a p-value of 0.07 doesn't mean that the data would be impossible to observe, it is unlikely. This is demonstrated further by the number of species that were observed once. The model was very different from the data in this aspect, not getting close to the actual observed value of 189. Despite this, the model operates well at the other extreme, and at the median, with the observed data in these cases being likely candidates to be generated from the model. Overall, this model is capturing some aspects of the data, but not others. It is difficult to say how trustworthy it is.

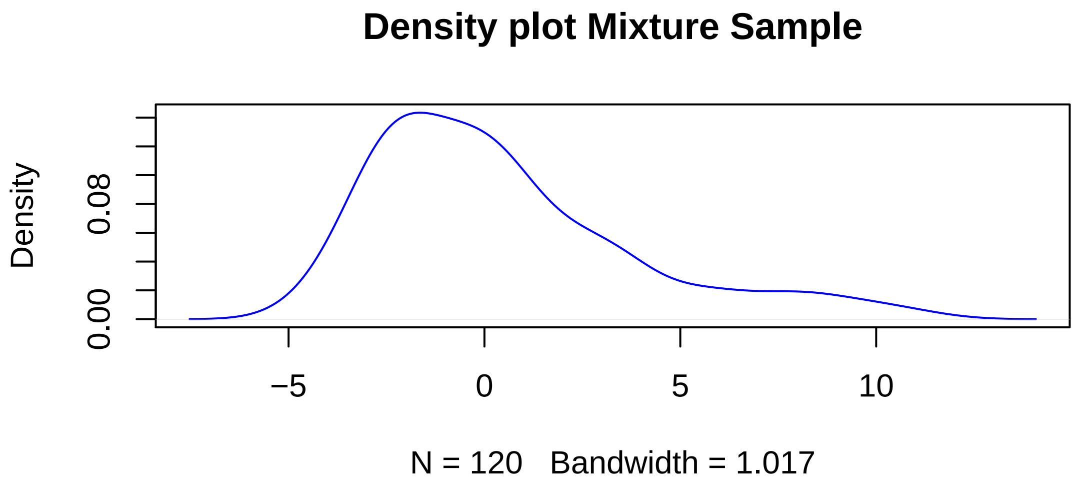
f) In order to construct the interval in (d), we made some assumptions. First, we assumed that each species represented an independent draw from a multinomial distribution, with the same probability vector. The independence assumption might be feasible, but we used the distribution of $x_i|\alpha, \beta$ to construct our multinomial distribution, so that each x_i became an iid negative binomial observation. It is unlikely that the probability of seeing each species 5 times, for example, is the same. The model is extremely sensitive to this assumption because in order to obtain the multinomial form, we needed the probabilities for each animal to be identical. Furthermore, we are assuming that we are equally likely to see any animal at any given time, since the observations are not considered for their order. It might be less likely that you see one animal early on, rather than later on and vice versa. This again impacts our ability to draw from the posterior of (α, β) and obtain credible intervals for the number of species seen.

Question 8 EM and Gibbs

a) In part (a), we set out to implement the EM algorithm to determine the MLE for (p, μ, σ^2) coming from a three component Gaussian mixture model. The actual parameters for this mixture are

$$\begin{aligned}(p_1, p_2, p_3) &= (0.1, 0.3, 0.6)\\ (\mu_1, \sigma_1^2) &= (0, 1)\\ (\mu_2, \sigma_2^2) &= (-2, 2)\\ (\mu_3, \sigma_3^2) &= (3, 16)\end{aligned}$$

We generate 120 samples from this model as data to use in solving for the maximum likelihood estimates.



Let $\Theta = (p, \mu, \sigma^2)$. The likelihood of our has the form

$$p(x|p, \sigma^2, \mu) = \prod_{i=1}^{120}\left[\sum_{j=1}^{3} p_i \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left\{-\frac{(x_i - \mu_j)^2}{2\sigma^2}\right\}\right]$$

and so the log-likelihood would be

$$\log L(p, \sigma^2, \mu|x) = \sum_{i=1}^{120} \log \left(\sum_{j=1}^{3} p_i \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left\{ -\frac{(x_i - \mu_j)^2}{2\sigma^2} \right\} \right)$$

Relying on methodology from Blimes (1998) we implement the EM algorithm to determine the MLE for Θ. In his paper, he suggests we look at X as incomplete, and posit the existence of unobserved $Y = \{y_i\}_{i=1}^{N}$ whose values inform which density was the data generator for each observation. We let $y_i = k$ for $k = 1, 2, 3$ to indicate which mixture component the i^{th} sample was generated from. If we were to know y_i we would have

$$\log L(\Theta|X, Y) = \sum_{i=1}^{N} \log\left(f(x_i|y_i)f(y)\right) = \sum_{i=1}^{N} \log(p_{y_i} f_{y_i}(x_i|\theta_{y_i}))$$

Since we do not know Y, we have to assume it is a random vector and derive a distribution. Using Bayes rule, where Θ^g represents a guess at the mixture density parameters, we have

$$f(y_i|x_i, \Theta^g) = \frac{p_{y_i}^g f_{y_i}(x_i|\theta_{y_i}^g)}{\sum_{k=1}^{3} p_k^g f_k(x_i|\theta_k^g)}$$

and so

$$f(\mathbf{y}|X, \Theta^g) = \prod_{i=1}^{N} f(y_i|x_i\Theta^g)$$

Blimes(1998) shows the first step of the EM algorithm finds the expected value of the complete-data log-likelihood with respect to the unknown Y given the observed data X and the current parameter estimates. He defines

$$Q(\Theta, \Theta^{(i-1)}) = E\left[\log f(X, Y|\Theta)|X, \Theta^{(i-1)}\right]$$

which he shows can be simplified to

$$Q(\Theta, \Theta^{(i-1)}) = \sum_{\ell=1}^{3}\sum_{i=1}^{120} \log(p_\ell) f(\ell|x_i\Theta^g) + \sum_{\ell=1}^{3}\sum_{i=1}^{120} \log(f_\ell(x_i|\theta_\ell)) f(\ell|x_i\Theta^g)$$

where we can maximize the first and second term separately, and notes that in this setting, the “E” and “M” steps of the EM algorithm are performed simultaneously. The expressions for the new parameter estimates from the old ones are shown to be

$$\begin{aligned} p_\ell^{new} &= \frac{1}{N}\sum_{i=1}^{N} f(\ell|x_i, \Theta^g) \\ \mu_\ell^{new} &= \frac{\sum_{i=1}^{N} x_i f(\ell|x_i\Theta^g)}{\sum_{i=1}^{N} f(\ell|x_i\Theta^g)} \\ \sigma_\ell^2 &= \frac{\sum_{i=1}^{N} f(\ell|x_i, \Theta^g)(x_i - \mu_\ell^{new})^2}{\sum_{i=1}^{N} f(\ell|x_i, \Theta^g)} \end{aligned}$$

Using the steps above outlined in Blimes(1998), we implement the EM algorithm to estimate the MLE for Θ. First, as a “sanity check”, we set the parameters to

$$
\begin{aligned}
(p_1, p_2, p_3) &= (0.1, 0.3, 0.6) \\
(\mu_1, \sigma_1^2) &= (0, 1) \\
(\mu_2, \sigma_2^2) &= (-20, 2) \\
(\mu_3, \sigma_3^2) &= (20, 16)
\end{aligned}
$$

and draw 1000 samples to ensure that the algorithm we have prepared is effective. The result for the EM algorithm on these samples is displayed below

```
##       p_current   mu_current sigma2_current
## [1,] 0.09643895   0.08157066       1.005823
## [2,] 0.30400000 -20.05105173       2.021463
## [3,] 0.59956105  20.00609588      16.373924
```

As can be seen by these results, the EM algorithm we have coded properly finds the maximum likelihood estimates, as they are close to the actual parameters we set.

We now use the algorithm on the 120 samples from the mixture distribution proposed in the question. We set initial points to be

$$
\begin{aligned}
(p_1, p_2, p_3)^{(0)} &= (0.33, 0.33, 0.34) \\
(\mu_1, \sigma_1^2)^{(0)} &= (0, 1) \\
(\mu_2, \sigma_2^2)^{(0)} &= (-1, 2) \\
(\mu_3, \sigma_3^2)^{(0)} &= (1, 3)
\end{aligned}
$$

to offer some variety while still being uninformative in our initial guesses. The results of running the algorithm on the mixture samples with these starting points are shown below.

```
##      p_current mu_current sigma2_current
## [1,] 0.1731059   0.167777      0.1824333
## [2,] 0.3963274  -2.205768      0.8524449
## [3,] 0.4305671   2.983822     11.9957398
```

Based on the results above, we find that the algorithm has done a modest job in detecting the true parameters. The mean values are approximately correct, The values of p have the correct ordering in terms of magnitude but the third distribution in the mixture does not have enough weight. The variance estimates also have the correct ordering in terms of magnitude, but are too small relative to the true parameter values.

b) We now implement Gibbs sampling to find the posterior of (p, μ, σ^2). Define X and Y similarly to how we did in part (a). Following a framework similar to that seen in Casella et al. (2001), we propose conjugate priors

$$
\begin{aligned}
\sigma_j^2 &\sim IG(\alpha_j, \beta_j) \\
\mu_j | \sigma_j^2 &\sim N(\lambda_j, \sigma_j^2 / \tau_j) \\
(p_1, p_2, p_3) &\sim Dir(\gamma_1, \gamma_2, \gamma_3)
\end{aligned}
$$

where $\lambda_j, \tau_j, \alpha_j, \beta_j$ and γ_j are known hyperparametes. Given this specification, it has been shown that

$$\mu_j^{(t)}|\sigma_j^{(t-1)}, x, y^{(t)} \sim N\left(\frac{\lambda_j\delta_j + \left(\sum_{i=1}^n 1_{\{y^{(t)}=j\}}x_i\right)^{(t)}}{\tau_j + n_j}, \frac{\sigma_j^{2(t-1)}}{\tau_j + n_j}\right)$$

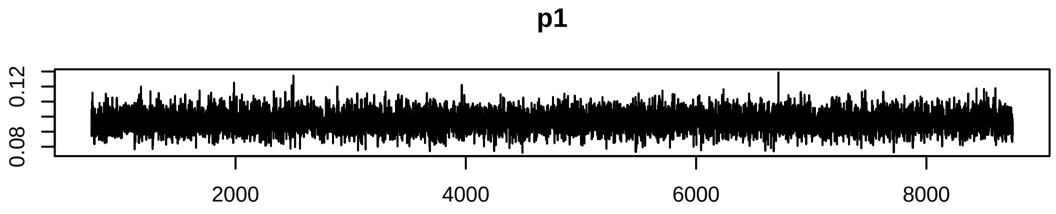
$$\sigma_j^{2(t)}|\mu_j^{(t)}x, y^{(t)} \sim IG(\alpha_j + 0.5(n_j+1), \beta_j + 0.5\tau_j(\mu_j^{(t)} - \lambda_j)^2 + 0.5\left(\sum_{i=1}^n 1_{\{y^{(t)}=j\}}(x_i - \mu_j)^2\right)$$

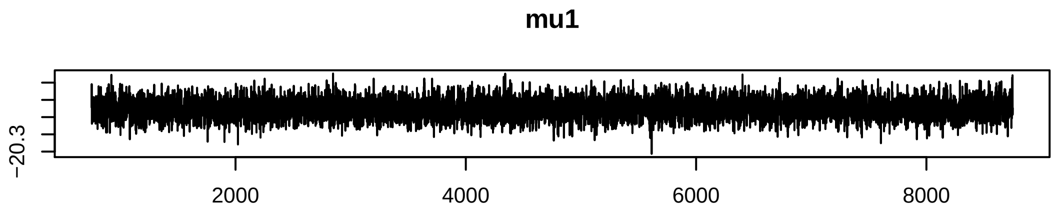
$$p^{(t)} \sim Dir(\gamma_1 + n_1, \gamma_2 + n_2, \gamma_3 + n_3)$$

This process is a data augmentation algorithm, so that we generate the "missing" vector $y^{(t)}$ in each iteration based on the draws for the other parameters. Martin et al. (2005) show that for $j = 1, 2, 3$

$$P(y_i^{(t)} = j) \propto \frac{p_j^{(t-1)}}{\sigma_j^{(t-1)}} \exp\left\{-(x_i - \mu_j^{(t-1)})^2/2(\sigma_j^2)^{(t-1)}\right\}$$

and so $n_j^{(t)} = \sum_{i=1}^n 1_{\{y_i^{(t)}=j\}}$, the number of observations that are labeled as being from density j in iteration t. For the hyperparameters, Raftery (1996b) used $\lambda = \bar{x}$, $\tau = 2.6/(x_{max} - x_{min})^2$, $\alpha = 1.28$, $\beta = 0.36s_x^2$. We also decide, for the sake of this exercise, to use $\gamma = 1$. First, we conduct a "sanity check"" similar to that for the EM algorithm in (a). We use the same data as seen in that test run, and present the results below.





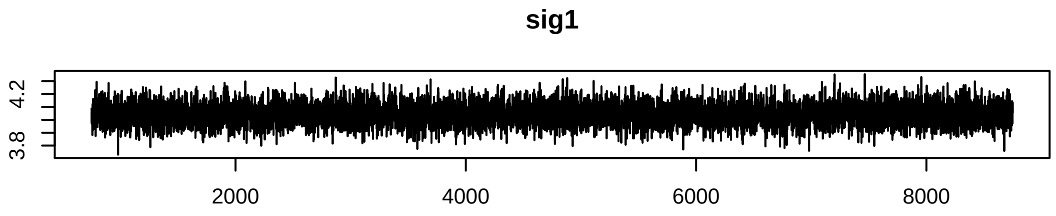
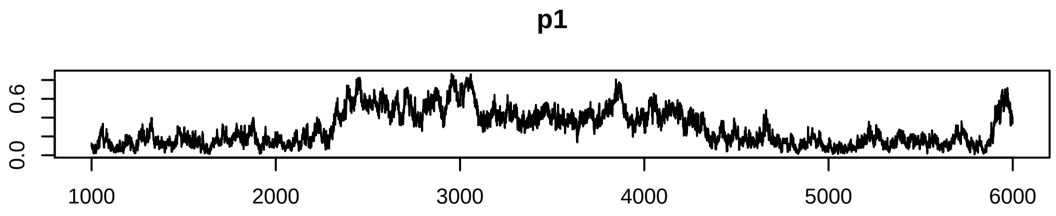
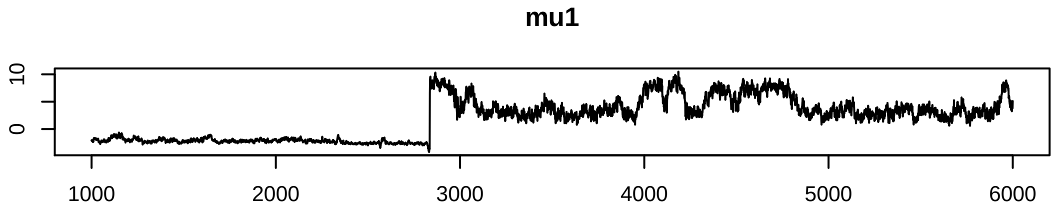


Table 6: Gibbs Sanity Check

	Mean	SD	0.025	0.5	0.975
p1	0.10	0.01	0.08	0.10	0.11
p2	0.30	0.01	0.28	0.30	0.32
p3	0.60	0.01	0.58	0.60	0.62
mu1	0.08	0.07	-0.06	0.08	0.23
mu2	-20.05	0.06	-20.16	-20.05	-19.93
mu3	20.01	0.12	19.79	20.00	20.23
sig1	1.00	0.05	0.91	1.00	1.11
sig2	1.42	0.04	1.34	1.42	1.50
sig3	4.04	0.08	3.88	4.04	4.21

We find that this Gibbs sampling procedure had good mixing and was able to locate the actual parameters, only being slightly off on the mean of mu_1, where we were supposed to obtain 0 rather than 0.08. Given this success, we move to use it on the data proposed in the question. Unfortunately, we do not find the same successful result. The convergence for these parameters is poor and we do not get near the actual results.





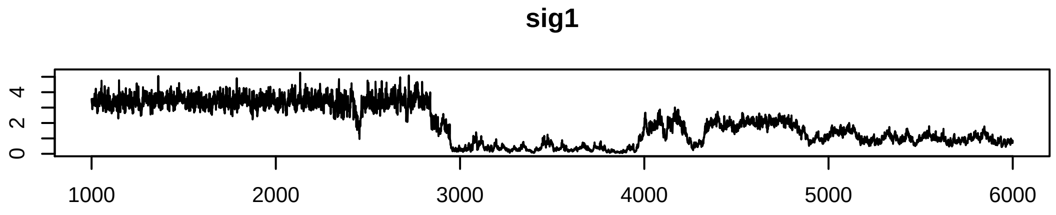


Table 7: Gibbs Sampler Mixture

	Mean	SD	0.025	0.5	0.975
p1	0.29	0.19	0.05	0.23	0.71
p2	0.33	0.15	0.08	0.36	0.61
p3	0.38	0.17	0.06	0.38	0.73
mu1	-0.79	1.14	-2.67	-0.22	0.55
mu2	1.95	3.59	-2.71	2.58	8.43
mu3	0.92	2.22	-2.42	0.39	5.01
sig1	0.74	0.54	0.09	0.60	1.92
sig2	2.19	1.24	0.24	2.32	4.08
sig3	1.91	1.29	0.15	1.71	4.07

We find that the mixing is very bad and the parameter estimates are nonsensical. I believe this has to do with the data dependent prior used by Raftery (1996b) that we have adopted. The data in this setting may not be offering a diffuse enough set of hyperparameters, or may be forcing the algorithm to get stuck in certain areas. After many modifications, better results were difficult to obtain, so due to the time constraint, we leave the result as is.

c) The results have been summarized in parts (a) and (b) , so in this section we compare the results and further not the struggles we have found in this problem. The sampling was unsuccessful, even though we know we were using a correct algorithm given our "sanity check". This is likely due to the reliance of the inference on the prior distributions, which were difficult to specify in a manor that allowed for the posterior space to be explored. A less severe complication was noticed with the EM algorithm. The results for the sample we investigated were altered based on starting points. If you were to set "bad" starting points, far away from the true parameters, it was easy to obtain poor estimates of the MLE. The data was not rich enough to obtain good samples or exact estimates in this case, but the EM algorithm seemed to work better.

BDA Question 15.3

a) We first fit an ordinary linear regression model including an intercept and the nine explanatory variables. For basic regression we let $y|\beta, \sigma, X \sim N(X\beta, \sigma^2 I)$ and use the uniform prior on $(\beta, \log\sigma)$ so that $p(\beta, \sigma^2) \propto \sigma^{-2}$. In this model, the posterior distributions are of the form

$$\begin{aligned} \beta|\sigma, y &\sim N(\hat{\beta}, V_\beta \sigma^2) \\ \sigma^2|y &\sim \text{Inv-}\chi^2(n-k, s^2) \end{aligned}$$

where n are the number of observations and k is the dimension of β and

$$\begin{aligned} \hat{\beta} &= (X^T X)^{-1} X^T y \\ V_\beta &= (X^T X)^{-1} \\ s^2 &= \frac{1}{n-k}(y - X\beta)^T (y - X\beta) \end{aligned}$$

We iterate between estimating σ^2 and β, and report the results based on scaled and centered data below.

Table 8: Simple Regression Summary

	Mean	SD	0.025	0.5	0.975
beta0	35.48	0.39	34.70	35.48	36.26
beta1	960.13	674.92	-398.29	964.25	2302.08
beta2	179.80	41.94	95.51	180.37	263.16
beta3	842.62	567.07	-301.38	846.46	1970.04
beta4	-165.53	39.70	-244.10	-165.99	-85.75
beta5	-684.15	477.29	-1628.70	-687.91	279.77
beta6	-22.63	6.23	-35.19	-22.69	-10.28
beta7	-862.20	631.67	-2117.50	-865.65	406.85
beta8	-7.09	3.23	-13.68	-7.04	-0.76
beta9	-71.82	44.86	-160.88	-72.19	18.15
sigma	1.45	0.54	0.81	1.33	2.82

The coefficients are very large, with large standard deviation. This is as we might expect given the high collinearity between the variables.

b) We now fit a mixed-effects linear regression model with a uniform prior distribution on the constant term and a shared normal prior distribution on the coefficients of the nine variables in the model. We use JAGS with model string

```
model{

mu <- X%*%beta

for(i in 1:n){ y[i] ~ dnorm(a + mu[i], tau1) }

for(j in 1:m){beta[j]~dnorm(0, tau2)}

a ~ dunif(-10000,10000)

tau1 ~ dgamma(0.0001,0.0001)

tau2 ~ dgamma(0.1,0.1)

var1 <- 1/tau1

var2 <- 1/tau2}
```

We print a summary of the results below.

```
##         Lower95 Median Upper95   Mean     SD MC%ofSD SSeff AC.150  psrf
## a        34.089 35.474  36.870 35.476  0.700     0.7 22500 -0.005 1.000
## beta[1]  -2.982  4.606  12.645  4.809  3.882     1.5  4177  0.028 1.000
## beta[2]  -6.410  1.415   9.578  1.644  3.934     1.6  4159  0.059 1.001
## beta[3] -11.306 -2.268   6.200 -2.377  4.351     1.6  4087  0.045 1.001
## beta[4]  -7.139  0.632   8.489  0.521  3.827     1.4  5134  0.026 1.001
## beta[5] -10.728 -2.299   5.896 -2.248  4.043     1.3  5594  0.005 1.000
## beta[6]  -1.241  2.455   6.000  2.419  1.826     0.8 16018  0.010 1.000
## beta[7]  -2.987  4.576  12.049  4.582  3.714     1.5  4571  0.015 1.000
## beta[8]  -7.947 -2.268   3.184 -2.365  2.815     1.0  9531  0.011 1.000
## beta[9]  -5.715  0.018   5.762  0.067  2.881     1.3  5768 -0.001 1.000
## var1      2.288  6.745  16.016  7.816  4.383     0.7 20880  0.006 1.000
## var2      1.868 17.606  70.061 25.357 27.770     1.6  4012  0.088 1.000
```

These results are interpreted in (c).

c) We see that running the mixed effects model with the normal prior vastly reduces the the magnitude of the coefficients and standard deviations. The auto-correlation and Gelman-Rubin statistics for each parameter indicates good mixing and convergence of the chains. The hierarchical variance parameter, labeled `var2`, shows that β could reasonably come from a multivariate normal distribution with variance around 17.5. Allowing for this hierarchical variance stabilizes the coefficient estimates, and allows us to separate the between coefficient variations the response level variation. I do agree that the inference in (a) is unacceptable, because the coefficient magnitudes are so large, with such large standard deviations, that they are nearly impossible to decipher. Also the fact that the variance coefficient in this model is well estimated indicates that there should be some hierarchical structure included to properly determine the coefficient magnitudes.

d) We now replicate part (a) using a t_4 prior on the variable coefficients. We again use JAGS with model string

```
model{

mu <- X%*%beta

for(i in 1:n){ y[i] ~ dnorm(a + mu[i], tau1) }

for(j in 1:m){beta[j]~dt(0, scale2 ,4)}

a ~ dunif(-10000,10000)

tau1 ~ dgamma(0.0001,0.0001)

scale2 ~ dchisq(0.5)

var1 <- 1/tau1 }
```

and present a summary below.

```
##          Lower95 Median Upper95   Mean    SD MC%ofSD SSeff AC.200  psrf
## a         34.048 35.475  36.862 35.476 0.708     0.7 22267  0.002 1.000
## beta[1]   -3.010  4.679  14.592  5.094 4.648     1.8  3177  0.068 1.003
## beta[2]   -5.141  1.010   8.506  1.362 3.366     1.5  4637  0.017 1.003
## beta[3] -10.740 -1.615   5.971 -2.034 4.173     1.7  3454  0.067 1.002
## beta[4]   -6.475  0.396   6.856  0.388 3.200     1.4  5433  0.017 1.001
## beta[5] -10.714 -1.577   5.113 -1.911 3.842     1.5  4378  0.051 1.001
## beta[6]   -1.182  2.138   5.736  2.168 1.772     0.8 14200 -0.007 1.000
## beta[7]   -3.212  4.336  13.433  4.614 4.387     1.7  3439  0.061 1.002
## beta[8]   -7.358 -1.573   2.813 -1.829 2.566     1.0  9096  0.010 1.000
## beta[9]   -5.155 -0.203   5.343 -0.113 2.572     1.3  6204  0.002 1.000
## var1       2.354  6.878  16.298  7.993 4.526     0.7 20874 -0.007 1.000
## scale2     0.003  0.111   0.516  0.173 0.208     1.0  9898 -0.007 1.000
```

When using the t_4 prior, we find the coefficient medians have been shrunk slightly toward 0, with wider tails, as we might expect. Again, the AC and Gelman-Rubin statistic indicate good convergence for each of the chains.

e) Other potential models for the regression coefficients might involve a correlation structure that ties, for example, the coefficients for x_1^2 and x_1, and any other pair of predictors that involve the same component.

Also, since the predictors are highly collinear, we could put shrinkage priors on the coefficients to drive some of their estimates to zero.

Code Appendix

```
knitr::opts_chunk$set(message = F)
knitr::opts_chunk$set(warning = F)
knitr::opts_chunk$set(fig.height = 3)
knitr::opts_chunk$set(fig.width = 6)
knitr::opts_chunk$set(fig.align="center")
knitr::opts_chunk$set(echo = F)
`%notin%` <- Negate(`%in%`)
library(kableExtra)
library(latex2exp)
library(LearnBayes)
### Question 1 ####
## A

ImpSampler <- function(nSamples, logTargetDensityFunc,
                       logProposalDensityFunc, proposalNewFunc = NULL,
                       rejectionControlConstant = FALSE) {

  samplefrom <- seq(-10, 10, by= 0.01)
  draws <- rnorm(nSamples, mean = 0, sd = 3)
  lp <- sapply(draws,logProposalDensityFunc)
  lt <- sapply(draws,logTargetDensityFunc)
  wts <- exp(lt - lp)
  if(rejectionControlConstant == FALSE){
    Awts <- wts/sum(wts)
    ESS <- 1/sum(Awts^2)
    return(list(draws, log(Awts), ESS))
  } else{
    c <- rejectionControlConstant
    unifsamp <- runif(nSamples, 0 ,1)
    prob <- pmin(rep(1,nSamples), wts/c)
    samps <- unifsamp <= prob
    accept_rate <- sum(samps)/nSamples
    accepted_draws <- draws[samps]
    Awts <- wts[samps]/prob[samps]
    ESS <- sum(Awts)^2/sum(Awts^2)
    return(list(accepted_draws,log(Awts/sum(Awts)), accept_rate, ESS))
    }
  }
##B /C


logProposalDensityFunc <- function(x){log(dnorm(x,mean = 0, sd = 3))}
logTargetDensityFunc <- function(x){log(1/3*dnorm(x ,mean = -2, sd = 1) +
                                  2/3*dnorm(x , mean = 2, sd = 1))}

nSamples <- 5000
samplefrom <- seq(-10, 10, by= 0.01)
fx <- exp(sapply(samplefrom, logTargetDensityFunc))
gx <- exp(sapply(samplefrom, logProposalDensityFunc))

plot(samplefrom, fx, type = "l", col = "red", xlab = "", ylab = "Density")
```

```r
lines(samplefrom,gx, col = "darkblue")
legend(6,0.25, legend = c("f(x)", "g(x)"), col = c("red", "darkblue"), lty = 1)
### E
set.seed(6)

full <- ImpSampler(nSamples, logTargetDensityFunc, logProposalDensityFunc)
draws <- full[[1]]
wts <- exp(full[[2]])
ESS <- full[[3]]

mu1_hat <- sum(wts*draws)/sum(wts)
mu2_hat <- sum(wts*draws^2)/sum(wts)
theta_hat  <- sum(wts*exp(draws))/sum(wts)

mu1_hat
mu2_hat
theta_hat
print(paste("Mu1 estimate:", round(mu1_hat,4)))
print(paste("Mu2 estimate: ", round(mu2_hat,2)))
print(paste("Theta estimate: ", round(theta_hat,2)))
set.seed(20)

RejCont <- data.frame(Constant = 1:10,
                      ESS = rep(NA, 10),
                      Acceptance = rep(NA, 10),
                      mu1_error = rep(NA,10),
                      mu_2_error = rep(NA,10),
                      theta_error = rep(NA, 10)
                      )

for (c in 1:10) {

full <- ImpSampler(nSamples, logTargetDensityFunc, logProposalDensityFunc, rejectionControlConstant = c
draws <- full[[1]]
wts <- exp(full[[2]])
Acceptance <- full[[3]]
ESS <- full[[4]]

mu1_hat <- sum(wts*draws)/sum(wts)
mu2_hat <- sum(wts*draws^2)/sum(wts)
theta_hat  <- sum(wts*exp(draws))/sum(wts)

RejCont[c,] <- round(c(c, round(ESS,0), Acceptance, abs(mu1_hat-2/3), abs(mu2_hat-5), abs(theta_hat-8.2
}

colnames(RejCont) <- c("$c$", "ESS", "Acceptance", "$|\\hat{\\mu}_1 - \\mu_1|$", "$|\\hat{\\mu}_2 - \\mu
                       "$|\\hat{\\theta} - \\theta|$")

kable_styling(kable(RejCont, format = "latex", booktabs= TRUE, align = 'c',
                    caption = "Results of Importance Sampling", escape = F), latex_options = "HOLD_posi

par(mfrow = c(2,3))
plot(RejCont$`$c`, RejCont[,4], main = TeX("$|\\hat{\\mu}_1 - \\mu_1|$ by Constant"), xlab = TeX("$c$")
plot(RejCont$`$c`, RejCont[,5], main = TeX("$|\\hat{\\mu}_2 - \\mu_2|$ by Constant"), xlab = TeX("$c$")
plot(RejCont$`$c`, RejCont[,6], main = TeX("$|\\hat{E\\[e^x\\]} - E\\[e^x\\]|$ by Constant"), xlab = Te
plot(RejCont$`$c`, RejCont$ESS, main = "ESS by Constant", xlab = TeX("$c$"), ylab = "ESS", pch = 16, co
plot(RejCont$`$c`, RejCont$Acceptance, main = "Acceptance by Constant", xlab = TeX("$c$"), ylab = "Acce



## Question 10.5
## A

library(boot)
library(LearnBayes)
set.seed(1)
X<- runif(10)
real_a <- as.numeric(rmt(1, mean = 0, S = 2, df = 4))
real_b <- as.numeric(rmt(1, mean = 0, S = 1, df = 4))
real_ns <- rpois(10,5)
y <- rbinom(10, real_ns, inv.logit(real_a+real_b*X))


random_Data <- data.frame(Y= y,
                          n = real_ns,
                          theta = inv.logit(real_a+real_b*X),
                          X = X,
                          alpha = real_a,
                          beta = real_b
                          )
random_Data <- round(random_Data, 2)
colnames(random_Data) <- c("$y$", "$n$", "$\\theta$", "$x$", "$\\alpha$", "$\\beta$")


kable_styling(kable(random_Data, format = "latex", booktabs= TRUE, align = 'c',
                    caption = "Random Data Set from Model", escape = F), latex_options = "HOLD_position

## B

logpost <- function(theta, data){
  a <- theta[1]
  b <- theta[2]
  y <- data[,1]
  n <- data[,2]
  x <- data[,3]

  pri <- (-5/2)*log(1+(1/4)*(a/2)^2)+(-5/2)*log(1+(b^2)/4)
  like <- sum(y*log(inv.logit(a+b*x))) +sum((n-y)*log(1-inv.logit(a+b*x)))
  return(pri+like)
}

data <- cbind(random_Data$`$y$`, random_Data$`$n$`, random_Data$`$x$`)

Post_summary <- laplace(logpost, mode= c(-0.5,0.5),data = data)
# Post_summary
```

```
logG <- function(x){dmt(x, mean = Post_summary$mode, S= 2*Post_summary$var, df = 4, log = T )}


check_diff <- function(theta, data){logpost(theta, data) - logG(theta)}


max_diff <- laplace(check_diff, mode = c(-1,1), data = data)

dmax <- check_diff(max_diff$mode, data)


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,])
}
set.seed(123)

Rej_Samples <- rejectsamp(logpost, Post_summary, dmax , 10000, data)
Rej_Samples <- Rej_Samples[1:1000,]

# mean(Rej_Samples[,1])
# mean(Rej_Samples[,2])


alBetrej <- matrix(nrow = 2, ncol = 5)

alBetrej[1,] <- c(mean(Rej_Samples[,1]), sd(Rej_Samples[,1]), quantile(Rej_Samples[,1], c(0.025, 0.5,0.
alBetrej[2,] <- c(mean(Rej_Samples[,2]), sd(Rej_Samples[,2]), quantile(Rej_Samples[,2], c(0.025, 0.5,0.

alBetrej <- as.data.frame(alBetrej, rownames <- c("$\\alpha$", "$\\beta$"))

colnames(alBetrej) <- c("Mean", "SD", "0.025", "0.50", "0.975")

kable_styling(kable(alBetrej, format = "latex", booktabs= TRUE, align = 'c',
                    caption = "Rejection Sampling Summary", escape = F), latex_options = "HOLD_position



##C
Post_summary


##D

logpost <- function(theta, data){
  a <- theta[1]
  b <- theta[2]
  y <- data[,1]
  n <- data[,2]
  x <- data[,3]

  pri <- (-5/2)*log(1+(1/4)*(a/2)^2)+(-5/2)*log(1+(b^2)/4)
  like <- sum(y*log(inv.logit(a+b*x))) +sum((n-y)*log(1-inv.logit(a+b*x)))
  return(pri+like)
}

ImpSampler_t <- function(nSamples, logTargetDensityFunc, tpar, data) {
  draws <- rmt(n=nSamples, mean = tpar$mode, S = tpar$var, df = 4)
  lp <- apply(draws,1 ,dmt, mean = tpar$mode, S = tpar$var, df = 4, log = T)
  lt <- apply(draws, 1,logpost, data= data)
  maxdiff <- max(lt-lp)
  wts <- exp(lt - lp - maxdiff)
  return(list(draws, wts, ESS))
}


alpha_beta_sampling <- ImpSampler_t(nSamples= 1000, logpost, Post_summary, data = data)

abdraws <-alpha_beta_sampling[[1]]
abwts <- alpha_beta_sampling[[2]]
abESS <- alpha_beta_sampling[[3]]

a_hat <- sum(abwts*abdraws[,1])/sum(abwts)
b_hat <- sum(abwts*abdraws[,2])/sum(abwts)

print(paste("E(\alpha|y) = ", round(a_hat,3)))
print(paste("E(\beta|y) = ", round(b_hat,3)))
print(paste("ESS = ", abESS))


#Question 10.8
##A

logpostBio <- function(theta, data){
  a <- theta[1]
  b <- theta[2]
  y <- data[,1]
  n <- data[,2]
  x <- data[,3]

  # pri <- log(1)
  like <- sum(y*log(inv.logit(a+b*x))) +sum((n-y)*log(1-inv.logit(a+b*x)))
  return(like)
}


dataBio <- cbind(c(0,1,3,5), c(5,5,5,5), c(-0.86,-0.30,-0.05,0.73))

BioPost_Summary <- laplace(logpostBio, mode= c(1,8),data = dataBio)

# BioPost_Summary
```

```r
set.seed(14)
Samps_approx <- rmnorm(n=10000, mean = BioPost_Summary$mode, varcov = BioPost_Summary$var)

calc_resamp_wts <- function(theta, data, npar){logpostBio(theta, data) -
    dmnorm(theta, mean = npar$mode, varcov = npar$var, log=TRUE)}

log_resamp_wts <- apply(Samps_approx, 1, calc_resamp_wts, data = dataBio, npar = BioPost_Summary)

resamp_wts <- exp(log_resamp_wts)

sample_index <- sample(1:10000, 1000, replace = FALSE, prob = resamp_wts)

WOreplace_samples <- Samps_approx[sample_index,]

plot(WOreplace_samples[,1], WOreplace_samples[,2], xlim = c(-4,10), ylim = c(-10,40),
     xlab = "alpha", ylab = "beta", main = "Without Replacement Samples")


# mean(WOreplace_samples[,1])
# mean(WOreplace_samples[,2])

# boxplot(resamp_wts)

##B

par(mfrow=c(1,2))
hist(resamp_wts, breaks = 100)
boxplot(resamp_wts[which(resamp_wts<=0.75)], main = "Weights Below 0.75")

## C


sample_index_replace <- sample(1:10000, 1000, replace = TRUE, prob = resamp_wts)


Wreplace_samples <- Samps_approx[sample_index_replace,]

plot(Wreplace_samples[,1], Wreplace_samples[,2], xlim = c(-4,10), ylim = c(-10,40),
     xlab = "alpha", ylab = "beta", main = "With Replacement Samples")

# mean(Wreplace_samples[,1])
# mean(Wreplace_samples[,2])


## Question 11.2

set.seed(4)
pulls <- 13000

MetropPulls <- cbind(rep(NA, pulls),rep(NA, pulls))

MetropPulls[1,] <- c(0.8, 7.7)
```

```r
logpostBio <- function(theta, data){
  a <- theta[1]
  b <- theta[2]
  y <- data[,1]
  n <- data[,2]
  x <- data[,3]

  pri <- log(1)
  like <- sum(y*log(inv.logit(a+b*x))) +sum((n-y)*log(1-inv.logit(a+b*x)))
  return(pri+like)
}


r_jump <- function(prevVals, newVals, data){logpostBio(newVals, data) - logpostBio(prevVals, data )}

for(i in 2:pulls){
  prevVals <- MetropPulls[i-1,]
  newVals <- c(runif(1, prevVals[1]-1.5, prevVals[1]+1.5), runif(1, prevVals[2]-6, prevVals[2]+6))

  r <- r_jump(prevVals, newVals, dataBio)

  if(log(runif(1)) < r){
    MetropPulls[i,] <- newVals
  }else{
    MetropPulls[i,] <- prevVals
  }

}

par(mfrow = c(1,2))
plot(3000:pulls, MetropPulls[3000:pulls,1], type = "l",
     ylab = "alpha", xlab="Sample")
plot(3000:pulls, MetropPulls[3000:pulls,2], type = "l",
     ylab = "beta", xlab="Sample")
par(mfrow =c(1,2))
acf(MetropPulls[3000:pulls, 1], main = "alpha")
acf(MetropPulls[3000:pulls, 2], main= "beta")
plot(MetropPulls[3000:pulls,1], MetropPulls[3000:pulls, 2], pch = 16, cex = 0.5,
     xlim = c(-4,10), ylim = c(-10,40), xlab = "alpha", ylab = "beta")
hist(-MetropPulls[3000:pulls,1]/MetropPulls[3000:pulls, 2],
     xlab = "LD50", breaks = 50, main = "LD50 Posterior")

set.seed(200)

QQ_data <- data.frame(M1 = c(83,92,92,46,67),
                      M2 = c(117,109,114,104,87),
                      M3 = c(101,93,92,86,67),
                      M4 = c(105,119,116, 102,116),
                      M5 = c(79,97,103,79,92),
                      M6 = c(57,92,104,77,100)
                      )

library(geoR)
```

```r
s2_pool <- var(c(QQ_data$M1,QQ_data$M2,QQ_data$M3,QQ_data$M4,QQ_data$M5,QQ_data$M6))
mean_pool <- mean(c(QQ_data$M1,QQ_data$M2,QQ_data$M3,QQ_data$M4,QQ_data$M5,QQ_data$M6))

posterior_sig2s1 <- rinvchisq(5000, 29, s2_pool/29)
posterior_mus1 <- rnorm(5000, mean_pool, sqrt(posterior_sig2s1))

posterior_sig2s2 <- rinvchisq(5000, 29, s2_pool/29)
posterior_mus2 <- rnorm(5000, mean_pool, sqrt(posterior_sig2s2))

Machine6pooled <- rnorm(5000, posterior_mus1, sqrt(posterior_sig2s1))
Machine7pooled <- rnorm(5000, posterior_mus2, sqrt(posterior_sig2s2))
par(mfrow=c(1,3))
hist(posterior_mus1, main = "Post. Mean M6", breaks = 35, xlab = "")
hist(Machine6pooled, main = "Post. Predict M6", breaks = 35, xlab = "")
hist(Machine7pooled, main = "Post. Predict M7", breaks = 35, xlab = "")


set.seed(12)

s2_separate <- 4*(var(QQ_data$M1) +var(QQ_data$M2) +
                    var(QQ_data$M3)  +var(QQ_data$M4)  +
                    var(QQ_data$M5)  +var(QQ_data$M6) )

mean_sep6 <- mean(QQ_data$M6)

seppost_sig2s6 <- rinvchisq(5000, 24, s2_separate/24)
seppost_mus6 <- rnorm(5000, mean_sep6, sqrt(seppost_sig2s6/6))

seppost_ys6 <- rnorm(5000, seppost_mus6, sqrt(seppost_sig2s6))

mean_sep5 <- mean(QQ_data$M5)

seppost_sig2s5 <- rinvchisq(5000, 24, s2_separate/24)
seppost_mus5 <- rnorm(5000, mean_sep5, sqrt(seppost_sig2s5/6))


mean_sep4 <- mean(QQ_data$M4)

seppost_sig2s4 <- rinvchisq(5000, 24, s2_separate/24)
seppost_mus4 <- rnorm(5000, mean_sep4, sqrt(seppost_sig2s4/6))


mean_sep3 <- mean(QQ_data$M3)

seppost_sig2s3 <- rinvchisq(5000, 24, s2_separate/24)
seppost_mus3 <- rnorm(5000, mean_sep3, sqrt(seppost_sig2s3/6))


mean_sep2 <- mean(QQ_data$M2)
```

```
seppost_sig2s2 <- rinvchisq(5000, 24, s2_separate/24)
seppost_mus2 <- rnorm(5000, mean_sep2, sqrt(seppost_sig2s2/6))


mean_sep1 <- mean(QQ_data$M1)

seppost_sig2s1 <- rinvchisq(5000, 24, s2_separate/24)
seppost_mus1 <- rnorm(5000, mean_sep1, sqrt(seppost_sig2s1/6))
par(mfrow=c(1,3))
hist(seppost_mus6, main = "Post. Mean M6", breaks = 35, xlab = "")
hist(seppost_ys6, main = "Post. Pred M6", breaks = 35, xlab = "")
hist(c(seppost_mus1,seppost_mus2,seppost_mus3,seppost_mus4,seppost_mus5,seppost_mus6) , main = "Post. P

set.seed(12345)

theta1_prev <- 83
theta2_prev <- 104
theta3_prev <- 86
theta4_prev <- 116
theta5_prev <- 97
theta6_prev <- 77

thetas_prev <- c(theta6_prev, theta5_prev, theta4_prev, theta3_prev, theta2_prev, theta1_prev)

mu_hat <- mean(thetas_prev)

tau2_hat <- var(thetas_prev)

sig2_hat <- sum( sum( (QQ_data$M1 - theta1_prev )^2 ) + sum( (QQ_data$M2 - theta2_prev)^2 ) +
                   sum( (QQ_data$M3 - theta3_prev)^2 ) + sum( (QQ_data$M4 - theta4_prev)^2 ) +
                   sum( (QQ_data$M5 - theta5_prev)^2 ) + sum( (QQ_data$M6 - theta6_prev)^2 ) ) / 30

iters <- 10000

tau2s <- rep(NA,iters)
sig2s <- rep(NA, iters)
mus <- rep(NA, iters)
thetas <- matrix( nrow = iters, ncol = 6)



for (i in 1:iters) {

  tau2s[i] <- rinvchisq(1,5, tau2_hat)
  mus[i] <- rnorm(1,mu_hat, sqrt(tau2s[i]/5))
  sig2s[i] <- rinvchisq(1, 30, sig2_hat)

  thetas_hat <- (mus[i]/tau2s[i] + 5*apply(QQ_data,2,mean) / sig2s[i]) / (1/tau2s[i] + 5/sig2s[i])
  Vthetas <-  1/ (1/tau2s[i] + 5/sig2s[i])

  thetas[i,] <- rnorm(6, thetas_hat, sqrt(1/ (1/tau2s[i] + 5/sig2s[i])))

  mu_hat <- mean(thetas[i,])
```

```r
  tau2_hat <- sum( (thetas[i,] - mus[i])^2 ) / 5

  sig2_hat <- sum( sum( (QQ_data$M1 - thetas[i,1] )^2 ) + sum( (QQ_data$M2 - thetas[i,2])^2 ) +
                   sum( (QQ_data$M3 - thetas[i,3])^2 ) + sum( (QQ_data$M4 - thetas[i,4])^2 ) +
                   sum( (QQ_data$M5 - thetas[i,5])^2 ) + sum( (QQ_data$M6 - thetas[i,6])^2 ) ) / 30

}




par(mfrow=c(3,3))
plot(1000:iters, thetas[1000:iters,1] , type = "l", main= "Theta_1", ylab = "")
plot(1000:iters, thetas[1000:iters,2] , type = "l", main= "Theta_2", ylab = "")
plot(1000:iters, thetas[1000:iters,3] , type = "l", main= "Theta_3", ylab = "")
plot(1000:iters, thetas[1000:iters,4] , type = "l", main= "Theta_4", ylab = "")
plot(1000:iters, thetas[1000:iters,5] , type = "l", main= "Theta_5", ylab = "")
plot(1000:iters, thetas[1000:iters,6] , type = "l", main= "Theta_6", ylab = "")
plot(1000:iters, mus[1000:iters] , type = "l", main= "mu", ylab = "")
plot(1000:iters, sig2s[1000:iters] , type = "l", main= "sigma2", ylab = "")



# plot(1:iters, mus, type = "l")

# hist(mus, breaks = 100)

set.seed(123)

postpred6 <- rnorm(iters , thetas[,6], sqrt(sig2s))
theta_preds <-rnorm(iters, mus, sqrt(tau2s))
postpred7<- rnorm(iters, theta_preds, sqrt(sig2s))

# quantile(theta_preds, c(0.025,0.975))

par(mfrow = c(1,3))
hist(thetas[1000:iters,6], main = "Post. Mean M6", breaks = 40, xlab = "")
hist(postpred6[1000:iters],  main = "Post. Pred M6", breaks = 40, xlab = "")
hist(postpred7[1000:iters],  main = "Post. Pred M7", breaks = 40, xlab = "")
Post_Dists <- data.frame(Post_Mean_M6 = c(mean(thetas[1000:iters,6]), sd(thetas[1000:iters,6]),
                                          quantile(thetas[1000:iters,6], c(0.025, 0.5, 0.975))),
           Post_Pred_M6 = c(mean(postpred6[1000:iters]), sd(postpred6[1000:iters]),
                            quantile(postpred6[1000:iters], c(0.025, 0.5, 0.975))),
          Post_Pred_M7 = c(mean(postpred7[1000:iters]), sd(postpred7[1000:iters]),
                           quantile(postpred7[1000:iters], c(0.025, 0.5, 0.975))),
          row.names = c("Mean", "SD", 0.025, 0.50, 0.975)
         )

kable_styling(kable(round(Post_Dists,2), format = "latex", booktabs= TRUE, align = 'c',
                    caption = "Rejection Sampling Summary"), latex_options = "HOLD_position")

library(geoR)
set.seed(123)
```

```r
theta1_prev <- mean(QQ_data$M1)
theta2_prev <- mean(QQ_data$M2)
theta3_prev <- mean(QQ_data$M3)
theta4_prev <- mean(QQ_data$M4)
theta5_prev <- mean(QQ_data$M5)
theta6_prev <- mean(QQ_data$M6)

thetas_prev <- c(theta6_prev, theta5_prev, theta4_prev, theta3_prev, theta2_prev, theta1_prev)

mu_hat <- mean(thetas_prev)

tau2_hat <- var(thetas_prev)

sig2_hat <- c( sum( (QQ_data$M1 - theta1_prev )^2 ) , sum( (QQ_data$M2 - theta2_prev)^2 ) ,
                   sum( (QQ_data$M3 - theta3_prev)^2 ) , sum( (QQ_data$M4 - theta4_prev)^2 ) ,
                   sum( (QQ_data$M5 - theta5_prev)^2 ) , sum( (QQ_data$M6 - theta6_prev)^2 ) ) / 5

iters <- 10000

df <- 2

tau2s <- rep(NA,iters)
sig20s <- rep(NA, iters)
mus <- rep(NA, iters)
thetas <- matrix( nrow = iters, ncol = 6)
sig2js <- matrix( nrow = iters, ncol = 6)

sig20_grid <- seq(0.001, 1000, length.out = 10000)

sig20_logconditional <- function(x, sigs){(3*df-1)*log(x) + -x*(df/2)*sum(1/sigs)}

s2j_calc <- function(yjs , thetaj){sum((yjs - thetaj)^2)/(5*df)}

thetas[1,] <- thetas_prev


for (i in 1:iters) {

  tau2s[i] <- rinvchisq(1,5, tau2_hat)
  mus[i] <- rnorm(1,mu_hat, sqrt(tau2s[i]/5))

  sig20_density <- sapply(sig20_grid, sig20_logconditional, sigs = sig2_hat)
  sig20_density <- exp(sig20_density - max(sig20_density))
  sig20s[i] <- base::sample(sig20_grid,1,prob = sig20_density)


  for (j in 1:6) {

    scale_sigj <- s2j_calc(QQ_data[,j], thetaj = thetas_prev[j]) + sig20s[i]/5
    sig2js[i,j] <- rinvchisq(1, 5+df, scale_sigj)
    thetas_hat <- (mus[i]/tau2s[i] + 5*mean(QQ_data[,j]) / sig2js[i,j]) / (1/tau2s[i] + 5/sig2js[i,j])

    Vthetas <-  1/ (1/tau2s[i] + 5/sig2js[i,j])

    thetas[i,j] <- rnorm(1, thetas_hat, sqrt(Vthetas))
  }


  mu_hat <- mean(thetas[i,])
  tau2_hat <- sum( (thetas[i,] - mus[i])^2 ) / 5

  sig2_hat <- sig2js[i,]

  thetas_prev <- thetas[i,]

  if(i%%50 == 0){print(i)}
}
# plot(1:iters, mus, type = "l")
# plot(1:iters, thetas[,1], type = "l")
# plot(1:iters, thetas[,2], type = "l")
# plot(1:iters, thetas[,3], type = "l")
# plot(1:iters, thetas[,4], type = "l")
# plot(1:iters, thetas[,5], type = "l")
# plot(1:iters, thetas[,6], type = "l")
# plot(1:iters, sig2js[,1], type = "l")
# plot(1:iters, sig2js[,2], type = "l")
# plot(1:iters, sig2js[,3], type = "l")
# plot(1:iters, sig2js[,4], type = "l")
# plot(1:iters, sig2js[,5], type = "l")
# plot(1:iters, sig2js[,6], type = "l")

# hist(mus, breaks = 100)
# hist(thetas[,3], breaks = 50)
#
# hist(sqrt(sig2js[,1]), breaks = 100)
# hist(sqrt(sig2js[,2]), breaks = 100)
# hist(sqrt(sig2js[,3]), breaks = 100)
# hist(sqrt(sig2js[,4]), breaks = 100)
# hist(sqrt(sig2js[,5]), breaks = 100)
# hist(sqrt(sig2js[,6]), breaks = 100)

# quantile(sqrt(sig2js[,1]), c(0.025,0.975))
# quantile(sqrt(sig2js[,2]), c(0.025,0.975))
# quantile(sqrt(sig2js[,3]), c(0.025,0.975))
# quantile(sqrt(sig2js[,4]), c(0.025,0.975))
# quantile(sqrt(sig2js[,5]), c(0.025,0.975))
# quantile(sqrt(sig2js[,6]), c(0.025,0.975))

# hist(sig20s)

sigma1 <- round(c(mean(sqrt(sig2js[,1])),sd(sqrt(sig2js[,1])), quantile(sqrt(sig2js[,1]), c(0.025, 0.5,
sigma2 <- round(c(mean(sqrt(sig2js[,2])),sd(sqrt(sig2js[,2])),quantile(sqrt(sig2js[,2]), c(0.025, 0.5,0
sigma3 <- round(c(mean(sqrt(sig2js[,3])),sd(sqrt(sig2js[,3])),quantile(sqrt(sig2js[,3]), c(0.025, 0.5,0
```

```r
sigma4 <- round(c(mean(sqrt(sig2js[,4])),sd(sqrt(sig2js[,4])),quantile(sqrt(sig2js[,4]), c(0.025, 0.5,0
sigma5 <- round(c(mean(sqrt(sig2js[,5])),sd(sqrt(sig2js[,5])),quantile(sqrt(sig2js[,5]), c(0.025, 0.5,0
sigma6 <- round(c(mean(sqrt(sig2js[,6])),sd(sqrt(sig2js[,6])),quantile(sqrt(sig2js[,6]), c(0.025, 0.5,0

theta1 <- round(c(mean(thetas[,1]),sd(thetas[,1]), quantile(thetas[,1], c(0.025, 0.5, 0.975))),2)
theta2 <- round(c(mean(thetas[,2]),sd(thetas[,2]), quantile(thetas[,2], c(0.025,0.5,0.975))),2)
theta3 <- round(c(mean(thetas[,3]),sd(thetas[,3]), quantile(thetas[,3], c(0.025,0.5,0.975))),2)
theta4 <- round(c(mean(thetas[,4]),sd(thetas[,4]), quantile(thetas[,4], c(0.025,0.5,0.975))),2)
theta5 <- round(c(mean(thetas[,5]),sd(thetas[,5]), quantile(thetas[,5], c(0.025,0.5,0.975))),2)
theta6 <- round(c(mean(thetas[,6]),sd(thetas[,6]), quantile(thetas[,6], c(0.025,0.5,0.975))),2)

mu <- round(c(mean(mus),sd(mus), quantile(mus, c(0.025,0.5,0.975))),2)
tau <- round(c(mean(sqrt(tau2s)),sd(sqrt(tau2s)), quantile(sqrt(tau2s), c(0.025,0.5,0.975))),2)
sig0 <- round(c(mean(sqrt(sig20s)),sd(sqrt(sig20s)), quantile(sqrt(sig20s), c(0.025,0.5,0.975))),2)


bigGibbs <- rbind(theta1, theta2,theta3, theta4,theta5, theta6,
                  sigma1, sigma2, sigma3,sigma4,sigma5,sigma6,
                  mu, tau, sig0)

bigGibbs <- as.data.frame(bigGibbs)

colnames(bigGibbs) <- c("Mean", "SD", 0.025, 0.50, 0.975)

# mean(sig2js[,6])
# var(QQ_data$M6)

kable_styling(kable(bigGibbs, format = "latex", booktabs= TRUE, align = 'c',
                    caption = "Gibbs Sample Summary", escape = F), latex_options = "HOLD_position")
library(LearnBayes)

specloglike <- function(theta, y){
  a <- theta[1]
  b <- theta[2]
  n <- length(y)

  loglike <- 0
  for (i in 1:n) {
    loglike <- loglike + y[i]*(log(choose(i+a-1,i))+ a*log(b/(b+1)) + i*log(1/(b+1)))
  }
  return(loglike)
}

y_spec = 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)



npar_spec <- laplace(specloglike, start(0,0), y= y_spec)

npar_spec$mode
npar_spec$var
```

```
multinomial_probs <- function(theta, cell){
  a <- theta[1]
  b <- theta[2]
  i <-cell

  prob <- log(choose(i+a-1,i))+ a*log(b/(b+1)) + i*log(1/(b+1))
  prob <-exp(prob)

  return(prob)
}
pulls <- 1000

# abs_spec <- rmnorm(pulls, npar_spec$mode, npar_spec$var)
#
# plot(abs_spec[,1], abs_spec[,2])


#
#
#
set.seed(1)
Samps_approx_spec <- rmnorm(n=10000,  npar_spec$mode, npar_spec$var)

calc_resamp_wts <- function(theta, data, npar){specloglike(theta, data) -
    dmnorm(theta, mean = npar$mode, varcov = npar$var, log=TRUE)}

log_resamp_wts <- apply(Samps_approx_spec, 1, calc_resamp_wts, data = y_spec, npar = npar_spec)

resamp_wts <- exp(log_resamp_wts - max(log_resamp_wts))

sample_index <- sample(1:10000, 1000, replace = T, prob = resamp_wts)

WOreplace_samples <- Samps_approx_spec[sample_index,]

# plot(WOreplace_samples[,1], WOreplace_samples[,2])

# plot(abs_spec[,1], abs_spec[,2])

abs_spec <- WOreplace_samples




set.seed(541)


ans_caught <- 13266

species <- rep(NA , pulls)

cells <-250
y_matrix <- matrix(nrow = pulls, ncol = cells)
```

```r
for(i in 1:pulls){
  probs <- c()
  for (j in 1:cells) {
    probs <- c(probs, multinomial_probs(abs_spec[i,], j))
  }
  probs <- probs/(sum(probs))

  y<- rep(0, cells)
  while (sum(y *1:cells) < ans_caught) {
    y <- y + rmultinom(1,1,probs)
  }

  y_matrix[i,] <- y
  species[i] <- sum(y)
  if(i %% 200 == 0){print(i)}
}



# sum(species>=496)/length(species)
#
# quantile(species - 496, c(0.025,0.975))
#
# sum(y_matrix[,24] >=3)/pulls
hist(species, main = "Speciece Observed at 13,266 Animals Caught", breaks = 20)

quantile(species, c(0.025, 0.50, 0.975))
set.seed(541)


ans_caught <- 3266

species <- rep(NA , pulls)

cells <-250
y_matrix <- matrix(nrow = pulls, ncol = cells)

for(i in 1:pulls){
  probs <- c()
  for (j in 1:cells) {
    probs <- c(probs, multinomial_probs(abs_spec[i,], j))
  }
  probs <- probs/(sum(probs))

  y<- rep(0, cells)
  while (sum(y *1:cells) < ans_caught) {
    y <- y + rmultinom(1,1,probs)
  }

  y_matrix[i,] <- y
  species[i] <- sum(y)
  if(i %% 200 == 0){print(i)}
}
```

```r
par(mfrow = c(2,2))

hist(species, main = "Total Species", xlab = "")
abline(v=496, col = "red")
text(525, 250, "p-value = 0.07", cex = 0.8)

hist(y_matrix[,1], main = "Species 1 Obs", xlab = "")
text(70, 150, "p-value = 0.00", cex = 0.8)

hist(y_matrix[,24], main = "Species 24 Obs", xlab = "")
abline(v=3, col = "red")
text(4.5, 200, "p-value = 0.25", cex = 0.8)

hist(y_matrix[,12], main = "Species 12 Obs", xlab = "")
abline(v=14, col = "red")
text(20, 150, "p-value = 0.45", cex = 0.8)

tot_samp <- 120
set.seed(25)
mixunif <- runif(tot_samp)
sampmixed <- rep(NA,tot_samp)

for(i in 1:tot_samp){
  if(mixunif[i]<= 0.1){
    sampmixed[i] <- rnorm(1)
  }else if(mixunif[i]<= 0.4){
    sampmixed[i] <- rnorm(1,-2, sqrt(2))
  }else{
    sampmixed[i] <- rnorm(1,3, sqrt(16))
  }
}


plot(density(sampmixed), main = "Density plot Mixture Sample", col = "blue")

tot_samp <- 2000
set.seed(25)
mixunif <- runif(tot_samp)
sampmixed_test <- rep(NA,tot_samp)

for(i in 1:tot_samp){
  if(mixunif[i]<= 0.1){
    sampmixed_test[i] <- rnorm(1)
  }else if(mixunif[i]<= 0.4){
    sampmixed_test[i] <- rnorm(1,-20, sqrt(2))
  }else{
    sampmixed_test[i] <- rnorm(1,20, sqrt(16))
  }
}


# plot(density(sampmixed_test), main = "Density plot Mixture Sample", col = "blue")
# p_current <- c(1/3,1/3,1/3)
# mu_current <- c(0,0,0)
# sigma2_current <- c(1,1,1)

p_current <- c(0.2,0.4,0.4)
mu_current <- c(0,-15,15)
sigma2_current <- c(2,1,12)

Theta <- cbind(p_current, mu_current, sigma2_current)

l_dens <- function(l ,Theta ,x){
  num <- Theta[l, 1]*dnorm(x, Theta[l,2], sqrt(Theta[l,3]))
  den=0
  for(i in 1:3){
    den <- den + Theta[i, 1]*dnorm(x,Theta[i,2], sqrt(Theta[i,3]))
  }
  return(num/den)
}


a_l_new <- function(l, l_dens, Theta, samps){

  n <- length(samps)
  a <- 0
  for (i in 1:n) {
    a <- a + l_dens(l, Theta, samps[i])
  }
  return(a/n)
}


mu_l_new <- function(l,l_dens,Theta,samps){
  n <- length(samps)
  num <- 0
  den <- 0
  for (i in 1:n) {
    num <- num + samps[i]*l_dens(l,Theta,samps[i])
    den <- den + l_dens(l,Theta,samps[i])
  }
  return(num/den)
}


sig2_l_new <- function(l, l_dens, Theta, samps){
  n <- length(samps)
  num <- 0
  den <- 0
  for (i in 1:n) {
    num <- num + ((samps[i]-Theta[l,2])^2)*l_dens(l,Theta,samps[i])
    den <- den + l_dens(l,Theta,samps[i])
  }
  return(num/den)
```

```r
}


mu_prev <- c(10,0,0)
p_prev <- c(10,0,0)
sig2_prev <- c(10,0,0)

while (sum((mu_prev - Theta[,2])^2)>=1e-8 |
       sum((p_prev - Theta[,1])^2)>=1e-8 |
       sum((sig2_prev - Theta[,3])^2)>=1e-8
       ) {

  p_prev <- Theta[,1]
  mu_prev <- Theta[,2]
  sig2_prev <- Theta[,3]

  for (l in 1:3) {
    Theta[l,1] <- a_l_new(l,l_dens,Theta,sampmixed_test)
  }

  for (l in 1:3) {
    Theta[l,2] <- mu_l_new(l,l_dens,Theta,sampmixed_test)
  }

  for (l in 1:3) {
    Theta[l,3] <- sig2_l_new(l,l_dens,Theta,sampmixed_test)
  }
}

Theta



p_current <- c(0.33,0.33,0.34)
mu_current <- c(0,-1,1)
sigma2_current <- c(1,2,3)

Theta <- cbind(p_current, mu_current, sigma2_current)

l_dens <- function(l ,Theta ,x){
  num <- Theta[l, 1]*dnorm(x, Theta[l,2], sqrt(Theta[l,3]))
  den=0
  for(i in 1:3){
    den <- den + Theta[i, 1]*dnorm(x,Theta[i,2], sqrt(Theta[i,3]))
  }
  return(num/den)
}


a_l_new <- function(l, l_dens, Theta, samps){

  n <- length(samps)
  a <- 0
  for (i in 1:n) {
    a <- a + l_dens(l, Theta, samps[i])
  }
  return(a/n)
}


mu_l_new <- function(l,l_dens,Theta,samps){
  n <- length(samps)
  num <- 0
  den <- 0
  for (i in 1:n) {
    num <- num + samps[i]*l_dens(l,Theta,samps[i])
    den <- den + l_dens(l,Theta,samps[i])
  }
  return(num/den)
}


sig2_l_new <- function(l, l_dens, Theta, samps){
  n <- length(samps)
  num <- 0
  den <- 0
  for (i in 1:n) {
    num <- num + ((samps[i]-Theta[l,2])^2)*l_dens(l,Theta,samps[i])
    den <- den + l_dens(l,Theta,samps[i])
  }
  return(num/den)
}


mu_prev <- c(10,0,0)
p_prev <- c(10,0,0)
sig2_prev <- c(10,0,0)

while (sum((mu_prev - Theta[,2])^2)>=1e-10 |
       sum((p_prev - Theta[,1])^2)>=1e-10 |
       sum((sig2_prev - Theta[,3])^2)>=1e-10
       ) {

  p_prev <- Theta[,1]
  mu_prev <- Theta[,2]
  sig2_prev <- Theta[,3]

  for (l in 1:3) {
    Theta[l,1] <- a_l_new(l,l_dens,Theta,sampmixed)
  }

  for (l in 1:3) {
    Theta[l,2] <- mu_l_new(l,l_dens,Theta,sampmixed)
  }

  for (l in 1:3) {
    Theta[l,3] <- sig2_l_new(l,l_dens,Theta,sampmixed)
  }
}

Theta


set.seed(100)

x <- sampmixed_test
lambda <- mean(x)
tau <- 2.6/(max(x)-min(x))^2
alpha <- 1.28
beta <- 0.001*var(x)

n<-length(x)


p_current <- c(0.30,0.30,0.4)
mu_current <- c(0,-1,1)
sigma2_current <- c(1,3,3)

Theta <- cbind(p_current, mu_current, sigma2_current)

burn <- 750
samples <- 8000 +burn

probs_calc <- function(j,Theta, samp){Theta[j,1]*dnorm(samp, Theta[j,2], sqrt(Theta[j,3]))}


dist1 <- matrix(ncol = 3, nrow = samples)
dist2 <- matrix(ncol = 3, nrow = samples)
dist3 <- matrix(ncol = 3, nrow = samples)


for(s in 1:samples){

y <- rep(NA, n)
for(i in 1:n){
 py_i <- rep(NA,3)
 py_i <- sapply(1:3, probs_calc, Theta = Theta, samp = x[i])
 y[i] <-sample(1:3, 1, prob = py_i)
}

ns <- c(sum(y==1), sum(y==2), sum(y==3))
sx <- c(sum(as.numeric(y==1)*x), sum(as.numeric(y==2)*x), sum(as.numeric(y==3)*x))
Theta[,1] <- rdirichlet(1, c(1 + ns[1], 1+ ns[2], 1+ ns[3]))
Theta[,2] <- rnorm(3, mean = (lambda*tau+sx)/(tau+ns), sd= sqrt(Theta[,3]/(tau+ns)))

s2x <- c(sum(as.numeric(y==1)*(x-Theta[1,2])^2), sum(as.numeric(y==2)*(x-Theta[2,2])^2), sum(as.numeric

Theta[,3] <- 1/rgamma(3, alpha + 0.5*(ns+1), beta+0.5*tau*(Theta[,2]-lambda)^2+0.5*s2x)
```

```
dist1[s,] <- Theta[1,]
dist2[s,] <- Theta[2,]
dist3[s,] <- Theta[3,]
}


par(mfrow = c(3,1))
plot(burn:samples, dist1[burn:samples,1], type="l", main = "p1", xlab = "", ylab = "")
plot(burn:samples, dist2[burn:samples,2], type="l", main = "mu1", xlab = "", ylab = "")
plot(burn:samples, sqrt(dist3[burn:samples,3]), type="l", main = "sig1", xlab = "", ylab = "")


p1 <- c(mean(dist1[burn:samples,1]), sd(dist1[burn:samples,1]), quantile(dist1[burn:samples,1], c(0.025
p2 <- c(mean(dist2[burn:samples,1]), sd(dist2[burn:samples,1]), quantile(dist2[burn:samples,1], c(0.025
p3 <- c(mean(dist3[burn:samples,1]), sd(dist3[burn:samples,1]), quantile(dist3[burn:samples,1], c(0.025
mu1 <- c(mean(dist1[burn:samples,2]), sd(dist1[burn:samples,2]), quantile(dist1[burn:samples,2], c(0.02
mu2 <- c(mean(dist2[burn:samples,2]), sd(dist2[burn:samples,2]), quantile(dist2[burn:samples,2], c(0.02
mu3 <- c(mean(dist3[burn:samples,2]), sd(dist3[burn:samples,2]), quantile(dist3[burn:samples,2], c(0.02
sig1 <- c(mean(sqrt(dist1[burn:samples,3])), sd(sqrt(dist1[burn:samples,3])), quantile(sqrt(dist1[burn:
sig2 <- c(mean(sqrt(dist2[burn:samples,3])), sd(sqrt(dist2[burn:samples,3])), quantile(sqrt(dist2[burn:
sig3 <- c(mean(sqrt(dist3[burn:samples,3])), sd(sqrt(dist3[burn:samples,3])), quantile(sqrt(dist3[burn:

sanityGibbs <- round(rbind(p1,p2,p3,mu1,mu2,mu3,sig1,sig2,sig3),2)

sanityGibbs <- as.data.frame(sanityGibbs)

colnames(sanityGibbs) <- c("Mean", "SD", 0.025, 0.50, 0.975)

kable_styling(kable(sanityGibbs, format = "latex", booktabs= TRUE, align = 'c',
                    caption = "Gibbs Sanity Check", escape = F), latex_options = "HOLD_position")

set.seed(100)

x <- sampmixed
lambda <- mean(x)
tau <- 2.6/(max(x)-min(x))^2
alpha <- 1.28
beta <- 0.001*var(x)
n<-length(x)


p_current <- c(0.2,0.30,0.5)
mu_current <- c(0,-1,2)
sigma2_current <- c(0,1,2)

Theta <- cbind(p_current, mu_current, sigma2_current)

burn <- 1000
samples <- 5000 +burn

probs_calc <- function(j,Theta, samp){Theta[j,1]*dnorm(samp, Theta[j,2], sqrt(Theta[j,3]))}
```

```r
dist1 <- matrix(ncol = 3, nrow = samples)
dist2 <- matrix(ncol = 3, nrow = samples)
dist3 <- matrix(ncol = 3, nrow = samples)


for(s in 1:samples){

y <- rep(NA, n)
for(i in 1:n){
 py_i <- rep(NA,3)
 py_i <- sapply(1:3, probs_calc, Theta = Theta, samp = x[i])
 y[i] <-sample(1:3, 1, prob = py_i)
}

ns <- c(sum(y==1), sum(y==2), sum(y==3))
sx <- c(sum(as.numeric(y==1)*x), sum(as.numeric(y==2)*x), sum(as.numeric(y==3)*x))
Theta[,1] <- rdirichlet(1, c(3 + ns[1], 3+ ns[2], 3+ ns[3]))
Theta[,2] <- rnorm(3, mean = (lambda*tau+sx)/(tau+ns), sd= sqrt(Theta[,3]/(tau+ns)))

s2x <- c(sum(as.numeric(y==1)*(x-Theta[1,2])^2), sum(as.numeric(y==2)*(x-Theta[2,2])^2), sum(as.numeric

Theta[,3] <- 1/rgamma(3, alpha + 0.5*(ns+1), beta+0.5*tau*(Theta[,2]-lambda)^2+0.5*s2x)

dist1[s,] <- Theta[1,]
dist2[s,] <- Theta[2,]
dist3[s,] <- Theta[3,]
}




par(mfrow = c(3,1))
plot(burn:samples, dist1[burn:samples,1], type="l", main = "p1", xlab = "", ylab = "")
plot(burn:samples, dist2[burn:samples,2], type="l", main = "mu1", xlab = "", ylab = "")
plot(burn:samples, sqrt(dist3[burn:samples,3]), type="l", main = "sig1", xlab = "", ylab = "")


p1 <- c(mean(dist1[burn:samples,1]), sd(dist1[burn:samples,1]), quantile(dist1[burn:samples,1], c(0.025
p2 <- c(mean(dist2[burn:samples,1]), sd(dist2[burn:samples,1]), quantile(dist2[burn:samples,1], c(0.025
p3 <- c(mean(dist3[burn:samples,1]), sd(dist3[burn:samples,1]), quantile(dist3[burn:samples,1], c(0.025
mu1 <- c(mean(dist1[burn:samples,2]), sd(dist1[burn:samples,2]), quantile(dist1[burn:samples,2], c(0.02
mu2 <- c(mean(dist2[burn:samples,2]), sd(dist2[burn:samples,2]), quantile(dist2[burn:samples,2], c(0.02
mu3 <- c(mean(dist3[burn:samples,2]), sd(dist3[burn:samples,2]), quantile(dist3[burn:samples,2], c(0.02
sig1 <- c(mean(sqrt(dist1[burn:samples,3])), sd(sqrt(dist1[burn:samples,3])), quantile(sqrt(dist1[burn:
sig2 <- c(mean(sqrt(dist2[burn:samples,3])), sd(sqrt(dist2[burn:samples,3])), quantile(sqrt(dist2[burn:
sig3 <- c(mean(sqrt(dist3[burn:samples,3])), sd(sqrt(dist3[burn:samples,3])), quantile(sqrt(dist3[burn:

sanityGibbs <- round(rbind(p1,p2,p3,mu1,mu2,mu3,sig1,sig2,sig3),2)

sanityGibbs <- as.data.frame(sanityGibbs)

colnames(sanityGibbs) <- c("Mean", "SD", 0.025, 0.50, 0.975)

kable_styling(kable(sanityGibbs, format = "latex", booktabs= TRUE, align = 'c',
                    caption = "Gibbs Sampler Mixture", escape = F), latex_options = "HOLD_position")
ChemEx <- readxl::read_excel("/Users/Anthony/Documents/Brown 2/Bayesian/Bayesian HW4/ChemEx.xlsx")

ChemEx$ratTemp <- ChemEx$ratio*ChemEx$temperature
ChemEx$tempCont <- ChemEx$temperature*ChemEx$contact
ChemEx$ratCont <- ChemEx$ratio*ChemEx$contact
ChemEx$temperature2 <- ChemEx$temperature^2
ChemEx$ratio2 <- ChemEx$ratio^2
ChemEx$contact2 <- ChemEx$contact^2

startfit <- lm(conversion~., data = as.data.frame(scale(ChemEx)))
summary(startfit)
#
# bOLS <- startfit$coefficients
# sigOLS <- sum(startfit$residuals^2)/(nrow(ChemEx) - ncol(ChemEx))
# startfit <- summary(startfit)
# solve(startfit$cov.unscaled)

y <- as.vector(ChemEx$conversion)
X <- as.matrix(cbind(rep(1, nrow(ChemEx)), as.matrix(scale(ChemEx[,-4]))))

QrX <- qr(X)

QrR <- qr.R(QrX)

draws <- 10000

beta_mat <- matrix(nrow = draws, ncol = ncol(X))
regSig2 <- rep(NA,draws)



betaHat <- solve(QrR) %*% t(solve(QrR)) %*% t(X) %*% y
Vbeta <-solve(QrR) %*% t(solve(QrR))
s2reg <- t(y-X%*%betaHat)%*%(y-X%*%betaHat)/(nrow(ChemEx)  - 10)

n <- nrow(ChemEx)
k <- ncol(X)


for (i in 1:draws) {
  regSig2[i] <- rinvchisq(1,n-k,s2reg)
  beta_mat[i,] <- rmnorm(1,mean = betaHat, Vbeta*regSig2[i])
}

beta0 <- c(mean(beta_mat[,1]), sd(beta_mat[,1]), quantile(beta_mat[,1], c(0.025, 0.5, 0.975)))
beta1 <- c(mean(beta_mat[,2]), sd(beta_mat[,2]), quantile(beta_mat[,2], c(0.025, 0.5, 0.975)))
beta2 <- c(mean(beta_mat[,3]), sd(beta_mat[,3]), quantile(beta_mat[,3], c(0.025, 0.5, 0.975)))
beta3 <- c(mean(beta_mat[,4]), sd(beta_mat[,4]), quantile(beta_mat[,4], c(0.025, 0.5, 0.975)))
beta4 <- c(mean(beta_mat[,5]), sd(beta_mat[,5]), quantile(beta_mat[,5], c(0.025, 0.5, 0.975)))
beta5 <- c(mean(beta_mat[,6]), sd(beta_mat[,6]), quantile(beta_mat[,6], c(0.025, 0.5, 0.975)))
beta6 <- c(mean(beta_mat[,7]), sd(beta_mat[,7]), quantile(beta_mat[,7], c(0.025, 0.5, 0.975)))
beta7 <- c(mean(beta_mat[,8]), sd(beta_mat[,8]), quantile(beta_mat[,8], c(0.025, 0.5, 0.975)))
```

```r
beta8 <- c(mean(beta_mat[,9]), sd(beta_mat[,9]), quantile(beta_mat[,9], c(0.025, 0.5, 0.975)))
beta9 <- c(mean(beta_mat[,10]), sd(beta_mat[,10]), quantile(beta_mat[,10], c(0.025, 0.5, 0.975)))
sigma <- c(mean(sqrt(regSig2)), sd(sqrt(regSig2)), quantile(sqrt(regSig2), c(0.025, 0.5, 0.975)))




simpleReg <- round(rbind(beta0,beta1,beta2,beta3,beta4,beta5,beta6,beta7,beta8,beta9, sigma),2)

simpleReg <- as.data.frame(simpleReg)
colnames(simpleReg) <- c("Mean", "SD", 0.025, 0.5, 0.975)


kable_styling(kable(simpleReg, format = "latex", booktabs= TRUE, align = 'c',
                    caption = "Simple Regression Summary", escape = F), latex_options = "HOLD_position"

X <- X[,-1]
library(runjags)
library(rjags)



datalist.mixed <- list(X=X, y = y, n=dim(X)[1], m=dim(X)[2])
n <- dim(X)[1]
m <- dim(X)[2]



model.string <- "
model{

mu <- X%*%beta

for(i in 1:n){
y[i] ~ dnorm(a + mu[i], tau1)
}

for(j in 1:m){beta[j]~dnorm(0, tau2)}
a ~ dunif(-10000,10000)
tau1 ~ dgamma(0.0001,0.0001)
tau2 ~ dgamma(0.1,0.1)
var1 <- 1/tau1
var2 <- 1/tau2


}
"

out.mixed <- run.jags(model.string, data = datalist.mixed, monitor = c("a","beta","var1", "var2"), burn

res.mixed <- as.matrix(out.mixed$mcmc)

# plot(1:nrow(res.mixed), res.mixed[,12], type = "l")
```

```r
#
#
# hist(res.mixed[,2],breaks = 100)
#
# median(res.mixed[,1])
# median(res.mixed[,2])
# median(res.mixed[,3])
# median(res.mixed[,4])
round(summary(out.mixed)[,-c(6,7)],3)
library(runjags)
library(rjags)



datalist.mixed <- list(X=X, y = y, n=dim(X)[1], m=dim(X)[2])
n <- dim(X)[1]
m <- dim(X)[2]



model.string <- "
model{

mu <- X%*%beta

for(i in 1:n){
y[i] ~ dnorm(a + mu[i], tau1)
}

for(j in 1:m){beta[j]~dt(0, scale2 ,4)}
a ~ dunif(-10000,10000)
tau1 ~ dgamma(0.0001,0.0001)
scale2 ~ dchisq(0.5)
var1 <- 1/tau1


}
"

out.mixed <- run.jags(model.string, data = datalist.mixed, monitor = c("a","beta","var1", "scale2"), bu

# res.mixed <- as.matrix(out.mixed$mcmc)
#
# plot(1:nrow(res.mixed), res.mixed[,2], type = "l")
#
#
# hist(res.mixed[,10],breaks = 100)
#
# median(res.mixed[,1])
# median(res.mixed[,2])
# median(res.mixed[,3])
# median(res.mixed[,4])
round(summary(out.mixed)[,-c(6,7)],3)
```