PHP2530: Bayesian Statistical Methods

Homework 4

Antonella Basso

July 15, 2022

Problem 1:

Importance sampling algorithm:

a. Write a function which implements the importance sampling algorithm (`ImpSampler()`) with the following five arguments:

   1. `logTargetDensityFunc(x)`: The natural logarithm of the target density, as a function of one argument.
   2. `logProposalDensityFunc(x)`: The natural logarithm of the proposal density, as a function of one argument.
   3. `proposalNewFunc()`: A function of zero arguments that is equivalent to a draw from the proposal density.
   4. `nSamples`: The number of samples one wishes to draw.
   5. `rejectionControlConstant(default=NULL)`: An argument whose default is NULL. This model parameter can only take positive values and is used to to make the rejection sampling more restrictive.

   Note: If `rejectionControlConstant=NULL`, this function should return the natural logarithm of the importance sampling weights as well as the Effective Sample Size (ESS). Otherwise when `rejectionControlConstant`$\neq$`NULL`, this function should return the number of accepted samples, the natural logarithm of the importance weights, the acceptance rate of the samples along with the ESS.

b. Consider the following mixture of two normals set up in one dimension:

$$f(x) = \frac{1}{3}\mathrm{N}(-2; 1^2; x) + \frac{2}{3}\mathrm{N}(2; 1^2; x)$$
$$g(x) = \mathrm{N}(0; 3^2; x)$$

c. Plot the two density functions $f(x)$ and $g(x)$ on the same plot. Take a look at it and observe that $g(x)$ is a reasonable importance density function.

d. Let $X \sim f(.)$. Theoretically compute:

   i. $\mu_1 = \mathrm{E}(X)$
   ii. $\mu_2 = \mathrm{E}(X^2)$
   iii. $\theta = \mathrm{E}(e^X)$

e. Use the function you wrote `ImpSampler()`, get the samples and the weights, compute the weighted average of the relevant $h(.)$ functions, and thus estimate μ_1, μ_2, and θ. Use $m = 5,000$ samples without any rejection control and report the estimated ESS from your runs.

f. Now vary the rejection control constant $c \in \{1, ..., 10\}$. For each such c, as in the previous part, use the function you wrote `ImpSampler()` and estimate μ_1, μ_2, and θ from $m = 5,000$ samples with rejection control. For each c, report estimated ESS (e_c, say), acceptance rate of the samples (a_c, say) from your runs. Let the error in the estimates be $|\hat{\mu}_1 - \mu_1|$, $|\hat{\mu}_2 - \mu_2|$, and $|\hat{\theta} - \theta|$. Now plot $(c, |\hat{\mu}_1 - \mu_1|)$, $(c, |\hat{\mu}_2 - \mu_2|)$, $(c, |\hat{\theta} - \theta|)$, (c, e_c), and (c, a_c). Now share your observations on the error in the estimators, acceptance rate, ESS, recommendations on the choice of c.

Solution

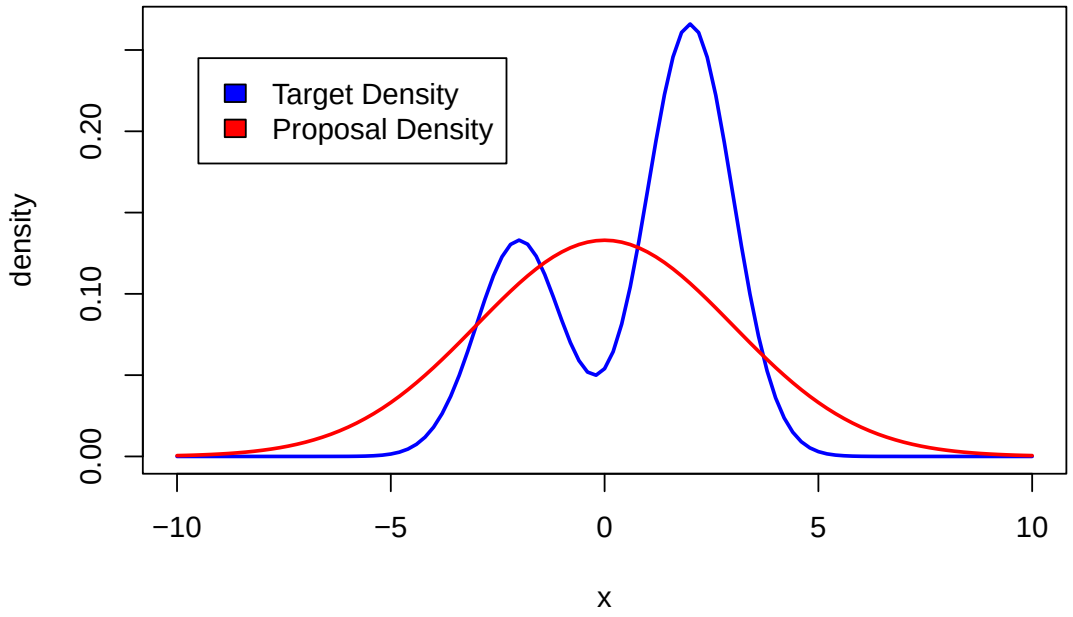
a. **Importance Sampler Algorithm**:

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

x_draws <- replicate(nSamples, proposalNewFunc())
log_f <- logTargetDensityFunc(x_draws)
log_g <- logProposalDensityFunc(x_draws)
log_w <- log_f-log_g
log_w <- log_w-max(log_w)
w <- exp(log_w)

ESS <- function(w){
  norm_w <- w/sum(w)
  ess <- 1/sum(norm_w^2)
  return(ess)
}

if (is.null(rejectionControlConstant)){
  return(list(ESS=ESS(w),
              Draws=x_draws,
              LogWeights=log_w))
} else{
  accepted <- 0
  mod_w <- c()
  for (i in 1:nSamples){
    r <- min(1, w[i]/rejectionControlConstant)
    u <- runif(1)
    if (r>u){
      accepted <- accepted+1
      mod_w[i] <- w[i]/r
    } else{
      mod_w[i] <- w[i]
    }
  }
  acceptance_rate <- accepted/nSamples
  return(list(ESS=ESS(mod_w),
              Draws=x_draws
              Accepted=accepted,
              AcceptanceRate=acceptance_rate,
              LogWeights=log(mod_w)))
} }
```

b. **Target and Proposal Densities**:

$$f(x) = \frac{1}{3}\mathrm{N}(-2; 1^2; x) + \frac{2}{3}\mathrm{N}(2; 1^2; x)$$
$$g(x) = \mathrm{N}(0; 3^2; x)$$

c. Plotting the above target and proposal density functions for $x \in [-10, 10]$, we obtain the following curves shown in Figure 1 below.



d. For $X \sim \frac{1}{3}\mathrm{N}(-2, 1^2) + \frac{2}{3}\mathrm{N}(2, 1^2)$, let X_1 and X_2 be random variables such that $X_1 \sim \mathrm{N}(-2, 1^2)$ and $X_2 \sim \mathrm{N}(2, 1^2)$.

i. $\mu_1 = \mathrm{E}(X)$

$$\begin{aligned}
\mu_1 &= \mathrm{E}(X) \\
&= \int_{-\infty}^{\infty} x f(x) dx \\
&= \int_{-\infty}^{\infty} x \left(\frac{1}{3} \frac{1}{\sqrt{2\pi}} e^{-\frac{(x+2)^2}{2}} + \frac{2}{3} \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-2)^2}{2}} \right) dx \\
&= \frac{1}{3} \int_{-\infty}^{\infty} x \left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(x+2)^2}{2}} \right) dx + \frac{2}{3} \int_{-\infty}^{\infty} x \left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(x-2)^2}{2}} \right) dx \\
&= \frac{1}{3}\mathrm{E}(X_1) + \frac{2}{3}\mathrm{E}(X_2)
\end{aligned}$$

ii. $\mu_2 = \mathrm{E}(X^2)$

$$\begin{aligned}
\mu_2 &= \mathrm{E}(X^2) \\
&= \int_{-\infty}^{\infty} x^2 f(x) dx \\
&= \int_{-\infty}^{\infty} x^2 \left(\frac{1}{3} \frac{1}{\sqrt{2\pi}} e^{-\frac{(x+2)^2}{2}} + \frac{2}{3} \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-2)^2}{2}} \right) dx \\
&= \frac{1}{3} \int_{-\infty}^{\infty} x^2 \left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(x+2)^2}{2}} \right) dx + \frac{2}{3} \int_{-\infty}^{\infty} x^2 \left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(x-2)^2}{2}} \right) dx \\
&= \frac{1}{3} \mathrm{E}(X_1^2) + \frac{2}{3} \mathrm{E}(X_2^2) \\
&= \frac{1}{3} \big(\mathrm{Var}(X_1) + \mathrm{E}(X_1)^2 \big) + \frac{2}{3} \big(\mathrm{Var}(X_2) + \mathrm{E}(X_2)^2 \big)
\end{aligned}$$

iii. $\theta = \mathrm{E}(e^X)$

$$\begin{aligned}
\theta &= \mathrm{E}(e^X) \\
&= \int_{-\infty}^{\infty} e^x f(x) dx \\
&= \int_{-\infty}^{\infty} e^x \left(\frac{1}{3} \frac{1}{\sqrt{2\pi}} e^{-\frac{(x+2)^2}{2}} + \frac{2}{3} \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-2)^2}{2}} \right) dx \\
&= \frac{1}{3} \int_{-\infty}^{\infty} e^x \left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(x+2)^2}{2}} \right) dx + \frac{2}{3} \int_{-\infty}^{\infty} e^x \left(\frac{1}{\sqrt{2\pi}} e^{-\frac{(x-2)^2}{2}} \right) dx \\
&= \frac{1}{3} \mathrm{E}(e^{X_1}) + \frac{2}{3} \mathrm{E}(e^{X_2}) \\
&= \frac{1}{3} \mathrm{MGF}_{X_1}(1) + \frac{2}{3} \mathrm{MGF}_{X_2}(1)
\end{aligned}$$

e. Given the theoretical derivations of the parameters in part (d) above, it follows that $\mu_1 = \frac{2}{3} \approx 0.67$, $\mu_2 = 5$, and $\theta = \frac{1}{3}(e^{-3/2} + 2e^{5/2}) \approx 8.2$. Using the importance sampler function from part (a), as well as the log target and log proposal density functions, we approximate these parameters with $5,000$ samples as follows:

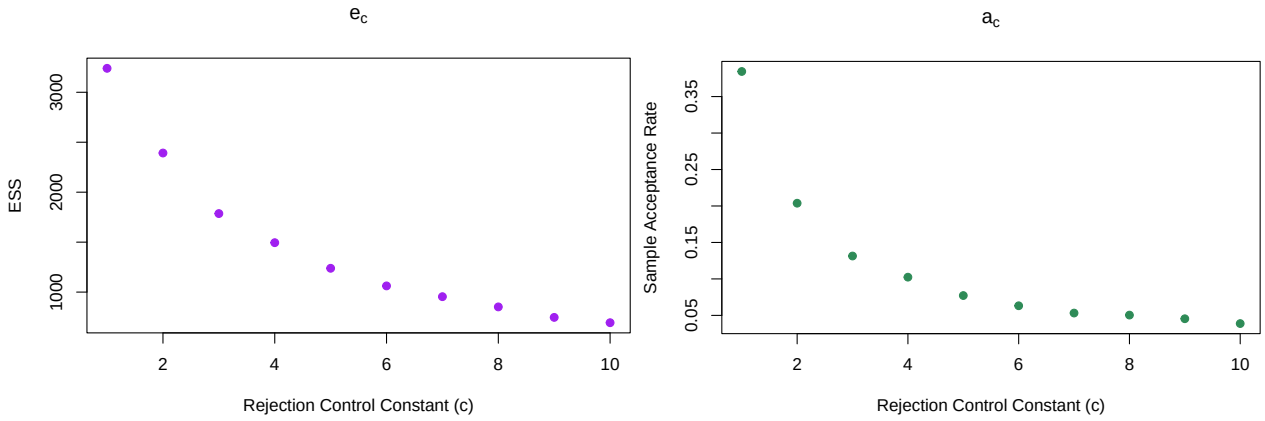
```
logTargetDensityFunc <- function(x){
  td_1 <- (1/3)*dnorm(x, -2, 1)
  td_2 <- (2/3)*dnorm(x, 2, 1)
  td <- td_1+td_2
  return(log(td))
}

logProposalDensityFunc <- function(x){
  pd <- dnorm(x, 0, 3)
  return(log(pd))
}

proposalNewFunc <- function(){
  pd_draw <- rnorm(1, 0, 3)
}
```

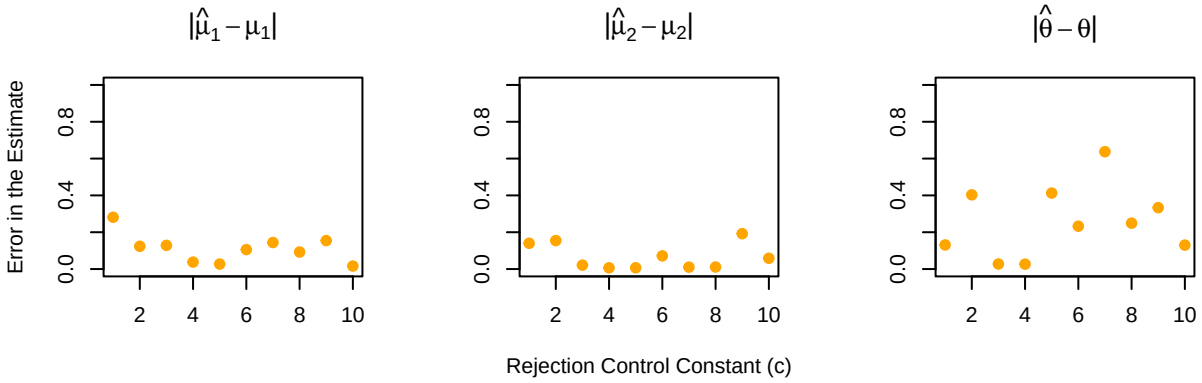
```
imps_1 <- ImpSampler(nSamples=5000,
                     logTargetDensityFunc=logTargetDensityFunc,
                     logProposalDensityFunc=logProposalDensityFunc,
                     proposalNewFunc=proposalNewFunc,
                     rejectionControlConstant=NULL)

hat_mu_1 <- sum(exp(imps_1$LogWeights)*imps_1$Draws)/sum(exp(imps_1$LogWeights))
hat_mu_2 <- sum(exp(imps_1$LogWeights)*(imps_1$Draws)^2)/sum(exp(imps_1$LogWeights))
hat_theta <- sum(exp(imps_1$LogWeights)*exp(imps_1$Draws))/sum(exp(imps_1$LogWeights))

c(hat_mu_1, hat_mu_2, hat_theta)
0.6361071, 5.0383150, 8.1052634
```

Additionally, we obtain an estimate ESS of 3199.363 (by calling `imps_1$ESS`).

f. Implementing the same function as in part (e) above, now with a rejection control constant, $c \in \{1, ..., 10\}$, we obtain the following results given by the plots and table below.

c	e_c	a_c	$\lvert\hat{\mu}_1 - \mu_1\rvert$	$\lvert\hat{\mu}_2 - \mu_2\rvert$	$\lvert\hat{\theta} - \theta\rvert$
1	3239.7037	0.3844	0.2812597	0.1400300	0.1309267
2	2391.9850	0.2038	0.1235915	0.1546712	0.4030618
3	1785.9345	0.1314	0.1286104	0.0213281	0.0273921
4	1494.6167	0.1024	0.0377832	0.0068003	0.0266598
5	1238.3347	0.0772	0.0272242	0.0068948	0.4126486
6	1062.0466	0.0632	0.1057008	0.0715042	0.2326837
7	954.2002	0.0532	0.1441266	0.0103540	0.6373841
8	851.8181	0.0504	0.0924322	0.0110174	0.2491762
9	746.8802	0.0454	0.1544281	0.1919703	0.3334488
10	693.6993	0.0388	0.0161659	0.0585689	0.1299729





From the graphs, it is clear that both the ESS (e_c) and the acceptance rate (a_c) decrease rather exponentially as the value of `rejectionControlConstant`$= c$ increases. This makes sense because the larger c becomes, the smaller the value of r gets, and hence, samples from the proposal density ($g(.)$) are less likely to be chosen. In turn, this not only decreases the acceptance rate, but also the amount of information we obtain about the target density ($f(.)$), resulting in a similar decrease in the ESS. Specifically, since we are rejecting samples on the basis of:

$$r = \frac{w(x)}{c}, \quad \text{where} \quad w(x) = \frac{f(x)}{g(x)},$$

it follows that we are rejecting samples on the basis of:

$$r = \frac{f(x)}{c \cdot g(x)}.$$

That is, how close $g(.)$ scaled by a factor of c is to $f(.)$. Thus, the closer r is to 1, the greater the chance of acceptance and ESS. When $r \geq 1$, this indicates that (for a particular value) $c \cdot g(.)$ is certainly within (or equal to) $f(.)$, and hence, the sample is accepted. Conversely, when c is large and $r < 1$, the rejection region increases, as the target density becomes engulfed by the proposal, and we become less certain in our approximation of $f(.)$ (resulting in a smaller ESS). In our case, it seems that choices for c are not incredibly significant when it comes to estimating the desired parameters, as shown by the graphs, particularly for μ_1 and μ_2. But, when considering all three parameters, as well as the e_c and a_c, it is clear that smaller values of c are preferable for preventing loss of precision and a waste in computation power.

Problem 2: (Ch.10.5)

Rejection sampling and importance sampling: Consider the model, $y_j = \text{Binomial}(n_j, \theta_j)$, where $\theta_j = \text{logit}^{-1}(\alpha + \beta x_j)$, for $j = 1, ..., J$, and with independent prior distributions, $\alpha \sim t_4(0, 2^2)$ and $\beta \sim t_4(0, 1)$. Suppose $J = 10$, the x_j values are randomly drawn from a $\text{U}(0, 1)$ distribution, and $n_j \sim \text{Poisson}^+(5)$, where Poisson^+ is the Poisson distribution restricted to positive values.

a. Sample a dataset at random from the model.

b. Use rejection sampling to get 1,000 independent posterior draws from (α, β).

c. Approximate the posterior density for (α, β) by a normal centered at the posterior mode with covariance matrix fit to the curvature at the mode.

d. Take 1,000 draws from the two-dimensional t_4 distribution with that center and scale matrix and use importance sampling to estimate $\text{E}(\alpha|y)$ and $\text{E}(\beta|y)$.

e. Compute an estimate of effective sample size for importance sampling using (10.4) on page 266.

Solution

a. **Model**:

$$\alpha \sim t_4(0, 2^2), \quad \beta \sim t_4(0, 1^2)$$

For $j = 1, 2, ..., J$,

$$x_j \sim \text{U}(0, 1)$$
$$\theta_j = \text{logit}^{-1}(\alpha + \beta x_j)$$
$$n_j \sim \text{Poisson}^{+}(5)$$
$$y_j \sim \text{Binomial}(n_j, \theta_j)$$

Sampled Dataset from the Model:

J	α	β	x_j	θ_j	n_j	y_j
1	3.570079	0.2419151	0.9248920	0.9779862	6	6
2	3.570079	0.2419151	0.1387976	0.9734975	4	4
3	3.570079	0.2419151	0.7019872	0.9767948	6	6
4	3.570079	0.2419151	0.1621936	0.9736431	6	6
5	3.570079	0.2419151	0.5993070	0.9762250	4	4
6	3.570079	0.2419151	0.5060361	0.9756957	7	6
7	3.570079	0.2419151	0.9019735	0.9778665	5	5
8	3.570079	0.2419151	0.4005028	0.9750829	2	2
9	3.570079	0.2419151	0.0309450	0.9728160	2	1
10	3.570079	0.2419151	0.0713582	0.9730733	6	6

b. Let $q(.)$ be the target distribution, and let $g(.)$ be the enveloping density such that $q \leq g$. Given that our prior density is proper, we use it as our proposal distribution from which we sample (α, β). This gives us:

Unnormalized Posterior:

$$\begin{aligned} q(\alpha, \beta | y, n, x) &= p(y | \alpha, \beta, n, x) p(\alpha, \beta) \\ &= p(\alpha, \beta) \prod_{j=1}^{J} p(y_j | \alpha, \beta, n_j, x_j), \end{aligned}$$

Enveloping Distribution:

$$\begin{aligned} g(\alpha, \beta) &= p(y | \hat{\alpha}, \hat{\beta}, n, x) p(\alpha, \beta) \\ &= p(\alpha, \beta) \prod_{j=1}^{J} p(y_j | \hat{\alpha}, \hat{\beta}, n_j, x_j), \end{aligned}$$

where $(\hat{\alpha}, \hat{\beta})$ is the MLE of the likelihood function[1].

Probability of Acceptance:

$$\frac{q}{g} = \frac{p(y | \alpha, \beta, n, x)}{p(y | \hat{\alpha}, \hat{\beta}, n, x)}.$$

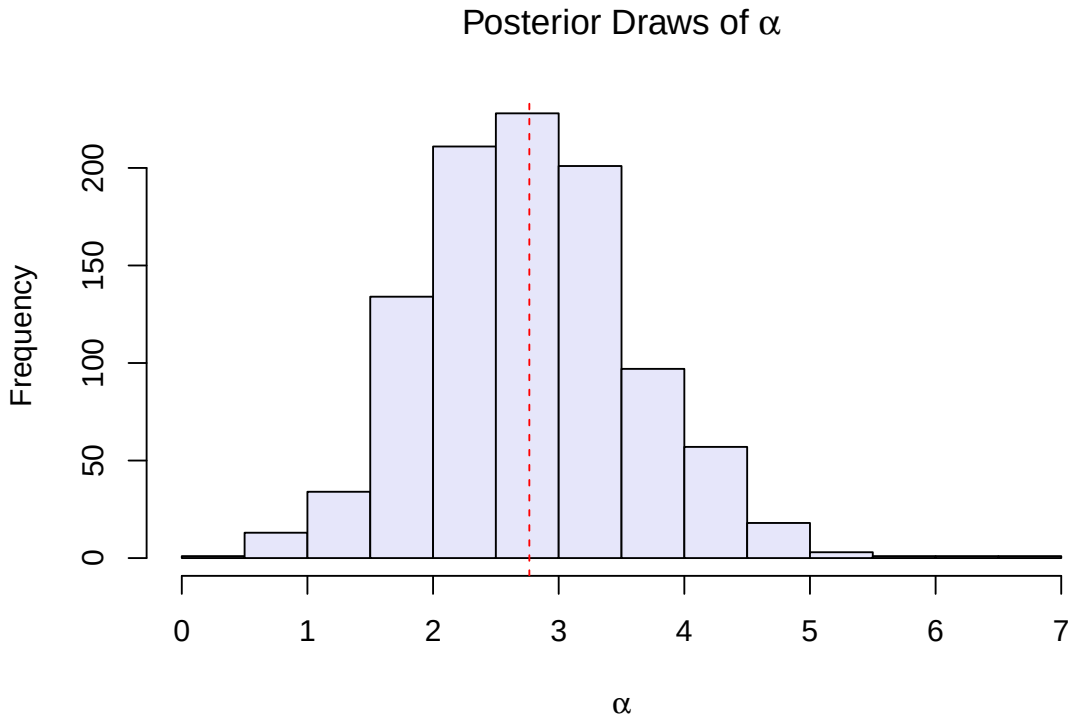
[1]Given that $p(y | \alpha, \beta, n, x) \leq p(y | \hat{\alpha}, \hat{\beta}, n, x)$ when $(\hat{\alpha}, \hat{\beta})$ is the MLE of the likelihood, it follows that $q \leq g$, making $g(.)$ an appropriate enveloping distribution for rejection sampling.

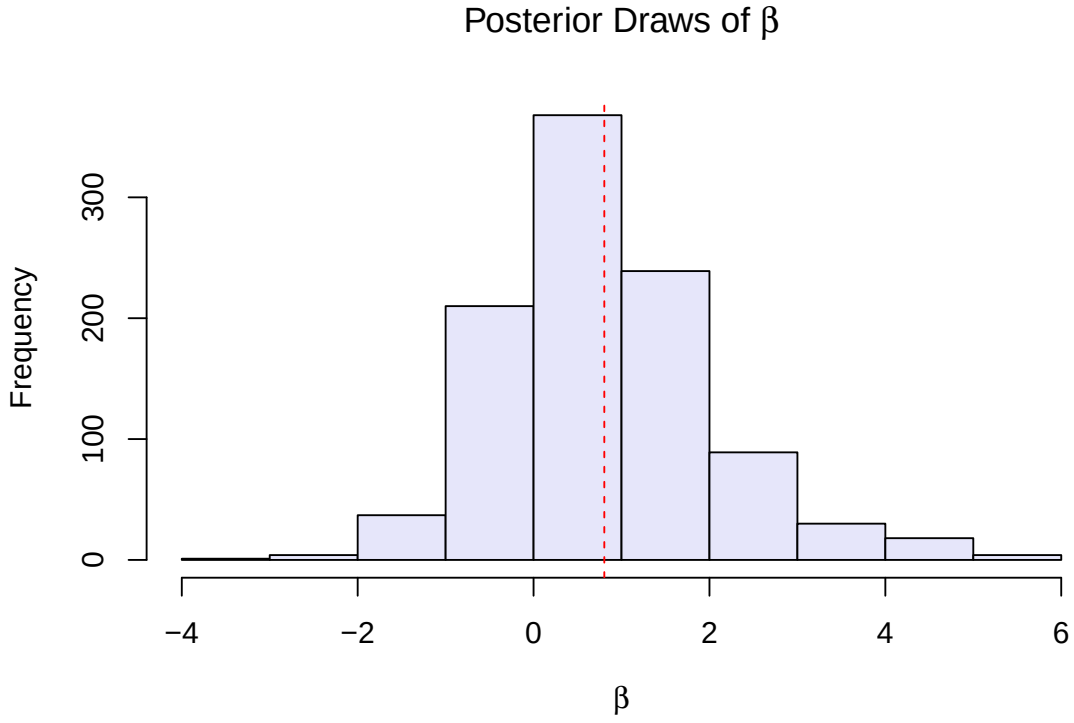
Now using rejection sampling to obtain 1,000 (independent) posterior draws of (α, β), we obtain the following results.

Observed Acceptance Rate: 0.06938181

Posterior Mode of α: 2.766464

Posterior Mode of β: 0.8042834

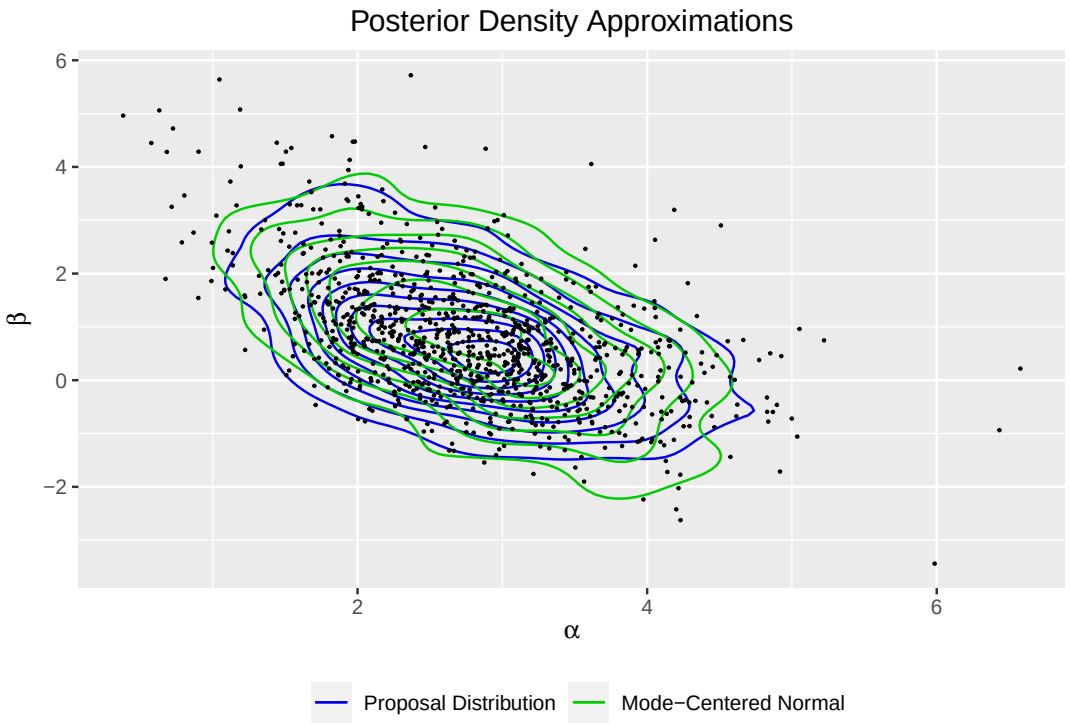




c. Let $\psi_i = (\alpha_i, \beta_i)$, such that $\hat{\psi} = (\hat{\alpha}, \hat{\beta}) \approx (2.767, 0.804)$ is the posterior mode. Then, the covariance matrix fit to the curvature at the mode is given by:

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(\psi_i - \hat{\psi})(\psi_i - \hat{\psi})^T = \begin{pmatrix} cov_{\alpha,\alpha} & cov_{\alpha,\beta} \\ cov_{\beta,\alpha} & cov_{\beta,\beta} \end{pmatrix} \approx \begin{pmatrix} 0.699 & -0.55 \\ -0.55 & 1.47 \end{pmatrix}.$$

Approximating the posterior density for (α, β) with $\mathrm{N}(\hat{\psi}, \Sigma)$, we obtain the following contour plot given below.



d. Using importance sampling with $S = 1,000$ draws from (two-dimensional) $t_4(\hat{\psi}, \Sigma)$ we obtain the following estimates.

$$\mathrm{E}(\alpha|y) \approx 2.7721844$$
$$\mathrm{E}(\beta|y) \approx 0.6602076$$

e. Let $\tilde{w}(\theta^s)$ be the normalized weights, $w(\theta^s)$, such that $\tilde{w}(\theta^s) = w(\theta^s)/\sum_{s'=1}^{S} w(\theta^{s'})$. Using Equation 10.4, we obtain the following effective sample size (ESS) estimate, S_{eff}, for the importance sampling done in part (d) with $S = 1,000$ draws.

$$S_{\text{eff}} = \frac{1}{\sum_{s=1}^{S}(\tilde{w}(\theta^s))^2} = 897.122$$

Problem 3: (Ch.10.8)

Importance resampling with and without replacement:

a. Consider the bioassay example introduced in Section 3.7. Use importance resampling to approximate draws from the posterior distribution of the parameters (α, β), using the normal approximation of Section 4.1 as the starting distribution. Sample $S = 10,000$ from the approximate distribution, and resample without replacement $k = 1,000$ samples. Compare your simulations of (α, β) to Figure 3.3b and discuss any discrepancies.

b. Comment on the distribution of the simulated importance ratios.

c. Repeat part (a) using importance sampling with replacement. Discuss how the results differ.

Table 3.1:

Dose, x_i (log g/ml)	Animals, n_i	Deaths, y_i
-0.86	5	0
-0.30	5	1
-0.05	5	3
0.73	5	5

Solution

a. Assuming a uniform prior, $p(\alpha, \beta) \propto 1$, on our parameters, we have the following model.

Model:

$$\alpha \sim t_4(0, 2^2), \quad \beta \sim t_4(0, 1^2)$$

For $i = 1, 2, ..., k$,

$$y_i|\theta_i \sim \text{Binomial}(n_i, \theta_i)$$

$$\theta_i = \text{logit}^{-1}(\alpha + \beta x_i)$$

$$p(y_i|\alpha, \beta, n_i, x_i) \propto \left[\text{logit}^{-1}(\alpha + \beta x_j)\right]^{y_i} \left[1 - \text{logit}^{-1}(\alpha + \beta x_j)\right]^{n_i - y_i}$$

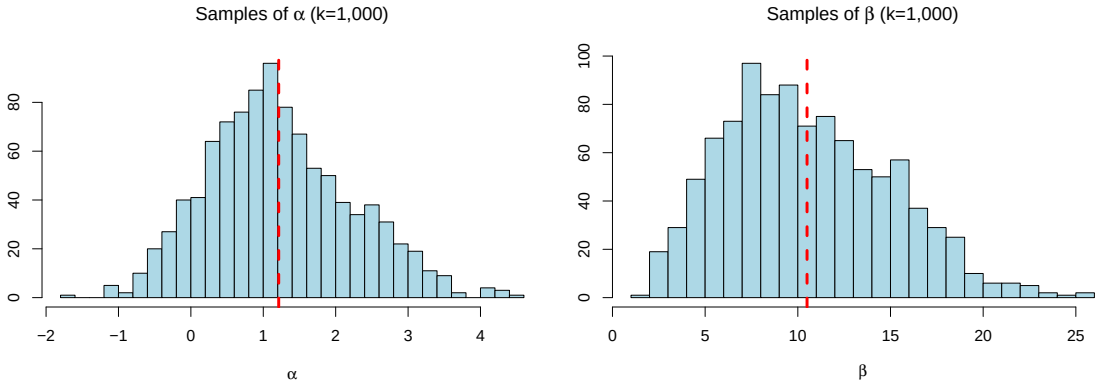
$$p(\alpha, \beta|y, n, x) \propto p(\alpha, \beta) \prod_{i=1}^{k} p(y_i|\alpha, \beta, n_i, x_i)$$

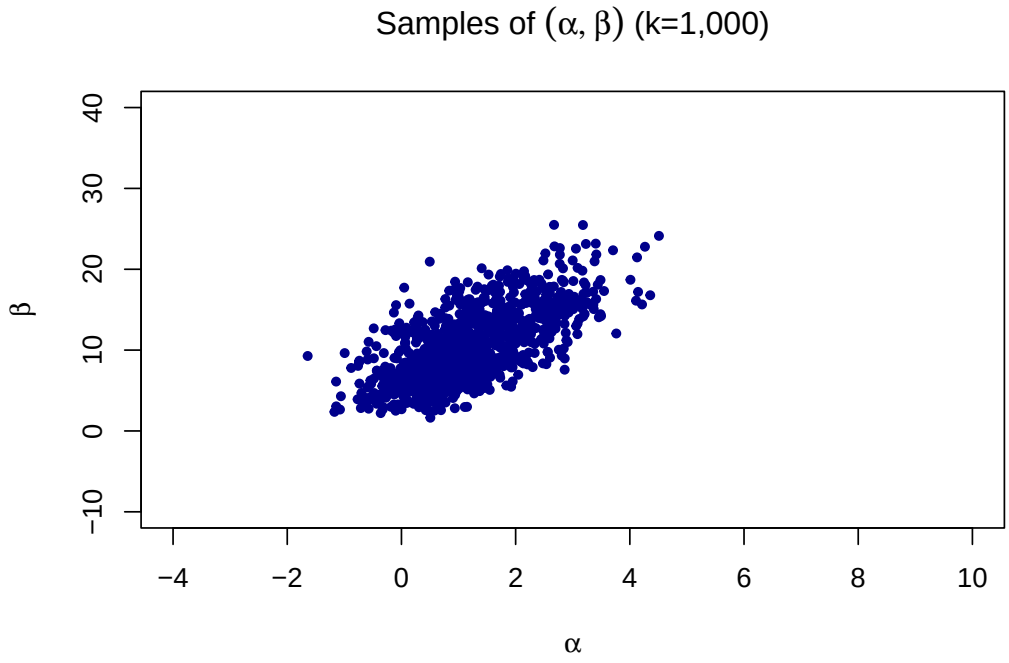
Normal Approximation:

$$p(\phi|y) \approx \text{N}(\hat{\phi}, [I(\hat{\phi})]^{-1}),$$

$$\text{where } \hat{\phi} \approx (0.847, 7.748), \text{ and } [I(\hat{\phi})]^{-1} \approx \begin{pmatrix} 1.038 & 3.545 \\ 3.545 & 23.74 \end{pmatrix}.$$

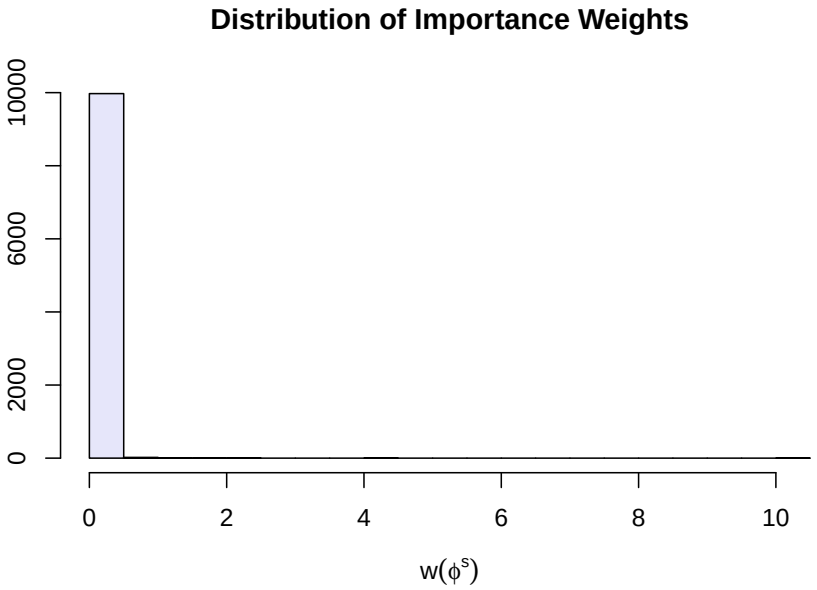
From the model, we compute the mode of the posterior distribution, $\hat{\phi} = (\hat{\alpha}, \hat{\beta})$, (via maximum likelihood estimation) and the inverse of the Fisher information matrix at the mode, $[I(\hat{\phi})]^{-1}$, to obtain the normal approximation (Equation (4.2), above). Using this as the starting distribution, we proceed with importance resampling on $S = 10,000$ draws, subsequently resampling without replacement on $k = 1,000$ samples. The resulting scatterplot (for comparison with Figure 3.3b) and histograms of α and β samples are given below.





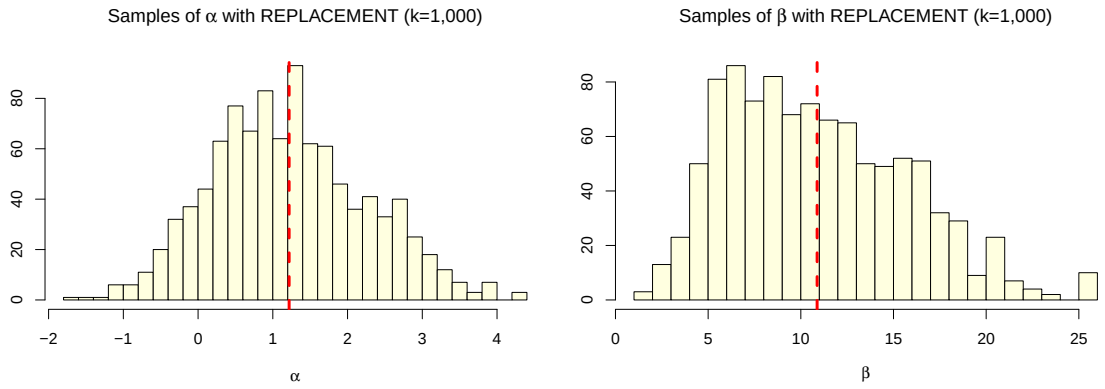
The scatterplot above shows that our samples of (α, β), obtained via importance resampling from the normal approximation distribution, are similar enough to the posterior draws shown in Figure 3.3b that we may argue this method was successful in adequately approximating the true posterior density. A notable difference however, is the variance of β, which appears larger when sampled directly from the posterior distribution (as seen in Figure 3.3b). Particularly, while we see no values of β greater than 30 in our samples, Figure 3.3b displays a few that are as large as 40.

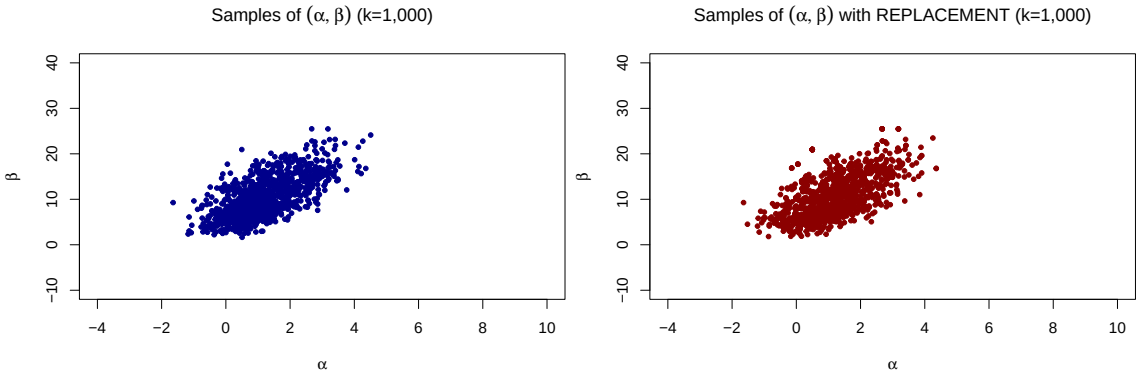
b. The distribution of the simulated importance weights/ratios is given by the histogram below. Here, it is clear that the majority of weights were close to 0. Specifically, given that there are only 26 (out of $S = 10,000$) that greater than 0.5 (6 of which are greater than 1), with the largest being as great as 10, it can be expected that most, if not all, of the samples associated with these comparatively large weights, were drawn during resampling.



c. Comparing the samples of (α, β), obtained via importance resampling with replacement as opposed to without replacement (from the normal approximation distribution), we see that there are very minimal differences. With almost identical distributions and means having gone from (1.2131, 10.4980) in part (a) to (1.2184, 10.8793), it is evident that both methods produced equally adequate approximations to the true posterior distribution.

This however contradicts what we would expect to see given the large discrepancy in our weight values (discussed in part (b)). That is, provided that only a handful of weights are drastically large compared to the majority, we should have observed a much denser distribution (smaller variance) when sampling with replacement, as the few samples with larger weights should have been drawn repeatedly. This is especially true given that such weights corresponded to parameter values in which β was significantly large, something that is not reflected in the figures below.





Problem 4: (Ch.11.2)

Metropolis algorithm: Replicate the computations for the bioassay example of Section 3.7 using the Metropolis algorithm. Be sure to define your starting points and your jumping rule. Compute with log-densities (see page 261). Run the simulations long enough for approximate convergence.

Solution

a. **Distributions**:

```
target_dist <- function(x_t){bioassay_log_posterior(x_t, x_i, n_i, y_i)}
jumping_dist_draw <- function(x_t){rmnorm(n=1, mean=x_t, varcov=bioassay_Sigma)}
starting_dist_draw <- function(){rmnorm(n=1, mean=bioassay_mode, varcov=bioassay_Sigma)}
```

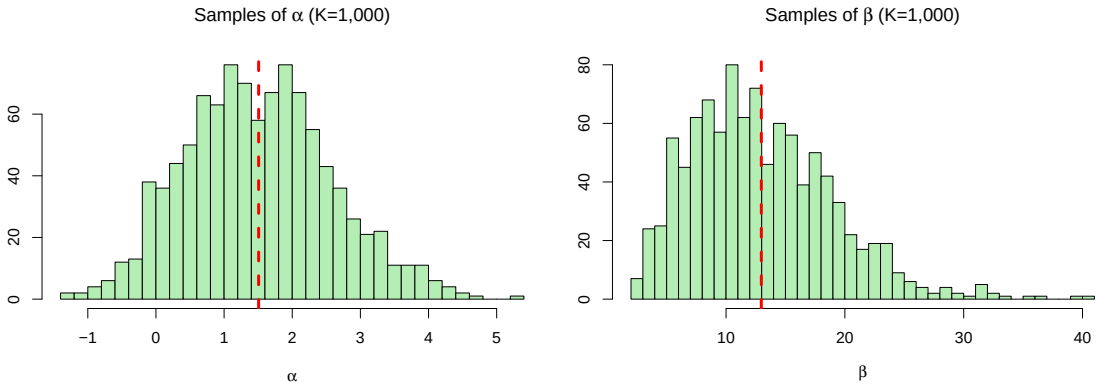
b. **Metropolis Algorithm**:

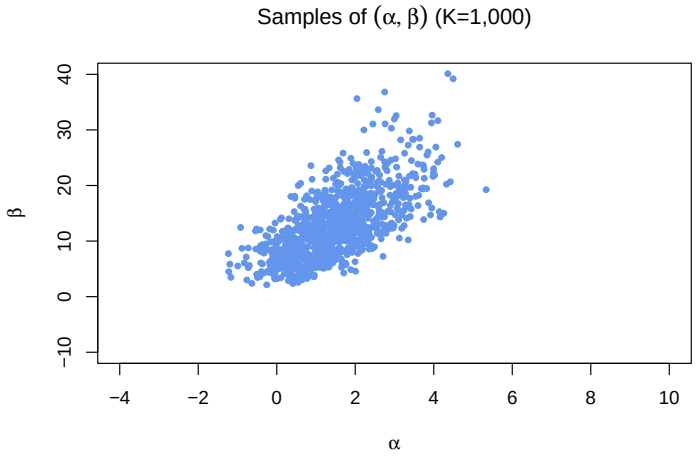
```
K <- 1000
x_t <- starting_dist_draw()

total <- matrix(0, nrow=K*2, ncol=2)
accepted <- c()
i <- 0
accept <- 0

while (accept!=K){
  i <- i+1
  x_prime <- jumping_dist_draw(x_t)
  imp <- exp(target_dist(x_prime)-target_dist(x_t))
  r <- min(1, imp)
  u <- runif(1)
  if(u<r){
    accept <- accept+1
    accepted <- rbind(accepted, x_prime)
    total[i,] <- x_prime
  } else{
    total[i,] <- x_t
  }
  x_t <- total[i,]
}
```

c. **Acceptance Rate**: $\frac{1{,}000}{1{,}716} \approx 0.583$

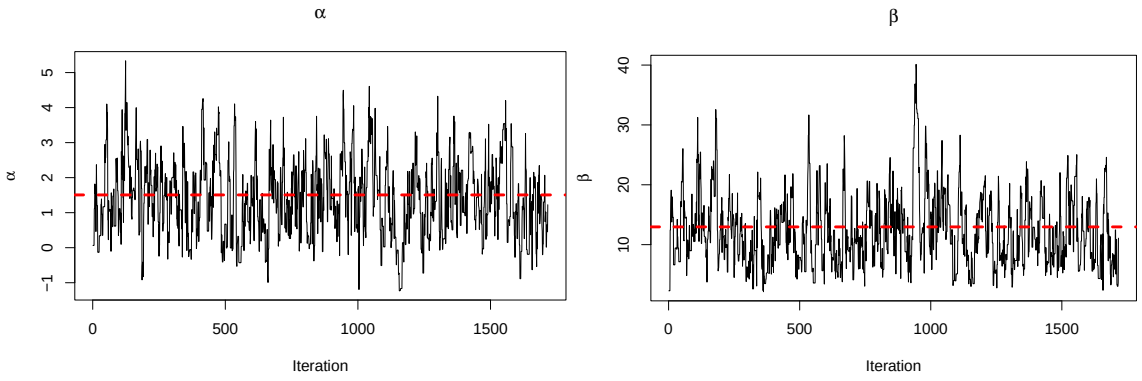
d. **Parameter Statistics**:

	Mean	Standard Deviation
α	1.507	1.072
β	12.967	5.984





e. **Trace Plots**:



Problem 5: (Ch.11.3)

Gibbs sampling: Table 11.4 contains quality control measurements from 6 machines in a factory. Quality control measurements are expensive and time-consuming, so only 5 measurements were done for each machine. In addition to the existing machines, we are interested in the quality of another machine (the seventh machine). Implement a separate, a pooled, and hierarchical Gaussian model with common variance described in Section 11.6. Run the simulations long enough for approximate convergence. Using each of three models (separate, pooled, and hierarchical) report:

i. the posterior distribution of the mean of the quality measurements of the sixth machine,

ii. the predictive distribution for another quality measurement of the sixth machine, and

iii. the posterior distribution of the mean of the quality measurements of the seventh machine.

Table 11.4:

Machine	Measurements
1	83, 92 ,92, 46, 67
2	117, 109, 114, 104, 87
3	101, 93, 92, 86, 67
4	105, 119, 116, 102, 116
5	79, 97, 103, 79, 92
6	57, 92, 104, 77, 100

Solution

Update Functions:

```
theta_post <- function(J, n, y_bar, mu, sigma, tau){
  V_theta <- 1/((1/(tau^2))+(n/(sigma^2)))
  theta_hat <- V_theta*((mu/(tau^2))+(n*y_bar/(sigma^2)))
  rnorm(J, theta_hat, sqrt(V_theta))
}

mu_post <- function(J, theta, tau){
  mu_hat <- mean(theta)
  var <- (tau^2)/J
  rnorm(1, mu_hat, sqrt(var))
}

sigma_post <- function(n, y_bar, s, theta){
  sigma2_hat <- sum(((n-1)*s^2)+(n*(y_bar-theta)^2))/sum(n)
  sqrt(rinvchisq(1, sum(n), sigma2_hat))
}

tau_post <- function(J, mu, theta){
  tau2_hat <- sum((theta-mu)^2)/(J-1)
  sqrt(rinvchisq(1, J-1, tau2_hat))
}
```

Gibbs Sampler Algorithm:

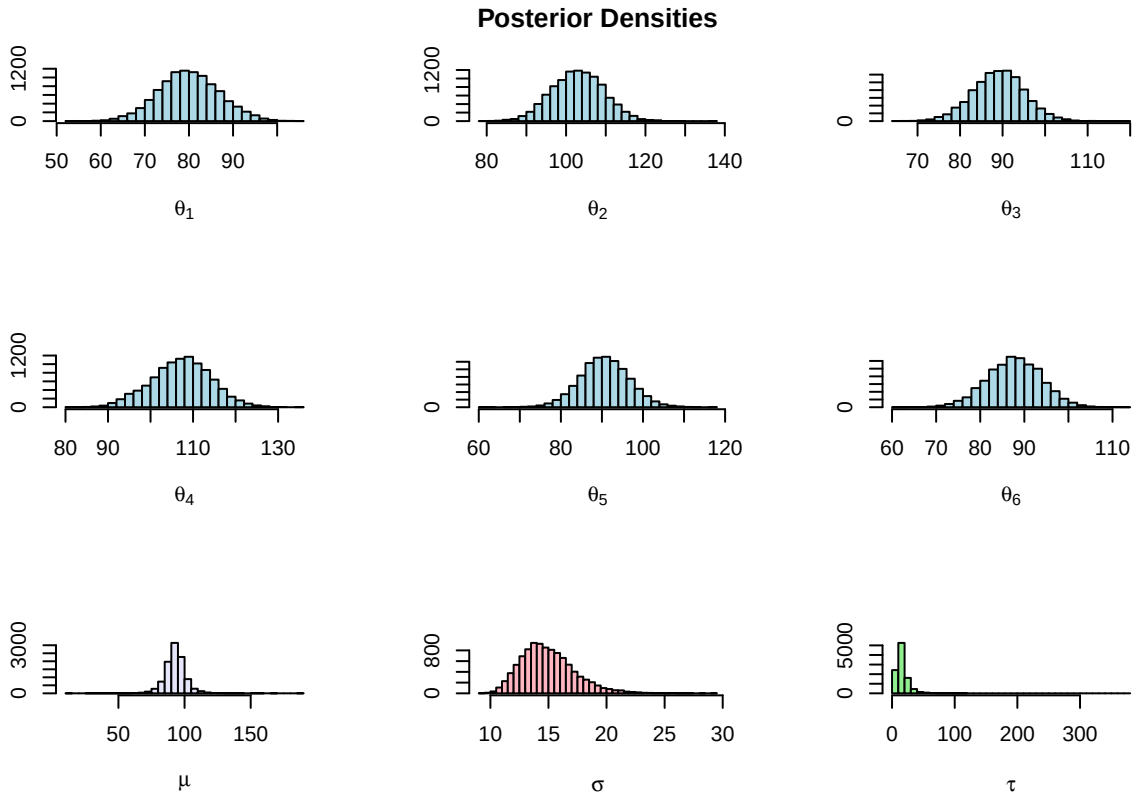
```
machine_chain <- function(iters, J, n, s, theta0, mu0, sigma0, tau0){

  theta_chain <- matrix(NA, iters, J)
  mu_chain <- rep(NA, iters)
  sigma_chain <- rep(NA, iters)
  tau_chain <- rep(NA, iters)

  theta <- theta0
  mu <- mu0
  sigma <- sigma0
  tau <- tau0

  for(i in 1:iters){
    theta <- theta_post(J, n, y_bar, mu, sigma, tau)
    mu <- mu_post(J, theta, tau)
    sigma <- sigma_post(n, y_bar, s, theta)
    tau <- tau_post(J, mu, theta)

    theta_chain[i,] <- theta
    mu_chain[i] <- mu
    sigma_chain[i] <- sigma
    tau_chain[i] <- tau
    }

  list(theta_chain=theta_chain, mu_chain=mu_chain,
       sigma_chain=sigma_chain, tau_chain=tau_chain)
}
```

Starting Values:

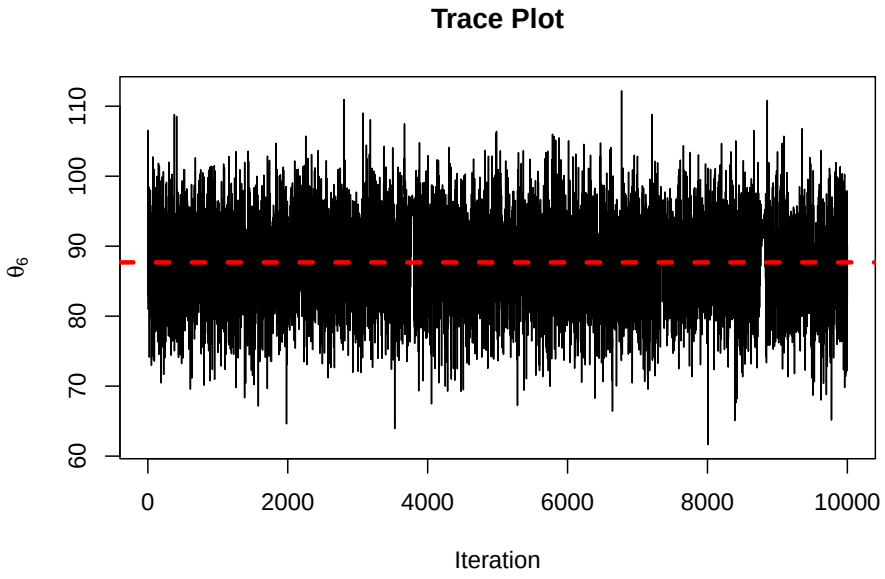
```
theta0 <- y_bar
mu0 <- mean(theta0)
sigma0 <- sqrt(mean(s^2))
tau0 <- sd(y_bar)
```

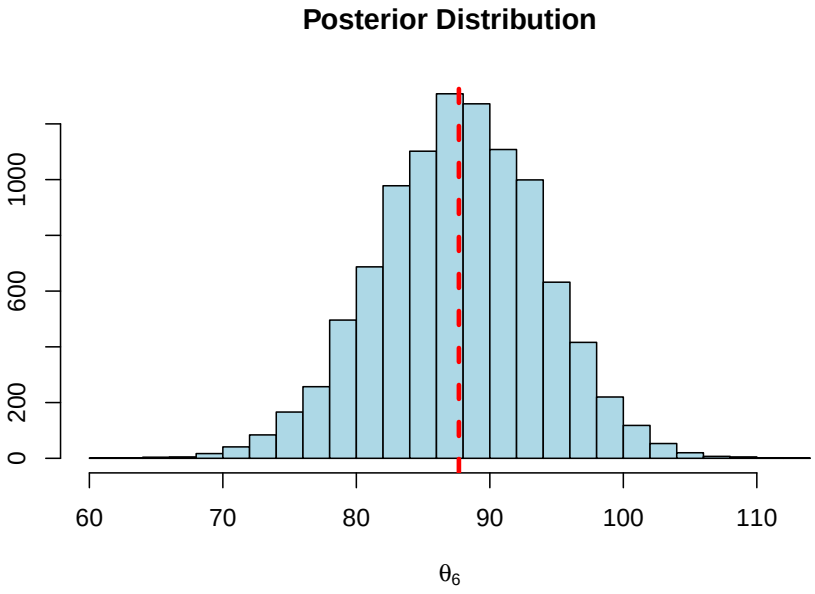
Posterior Results from 10,000 Iterations:

	2.5%	25%	50%	75%	97.5%
θ_1	66.6	75.4	79.9	84.5	93.9
θ_2	90.5	98.5	103.0	107.4	115.5
θ_3	76.9	85.1	89.2	93.1	101.1
θ_4	93.0	102.6	107.4	112.0	120.6
θ_5	78.8	86.7	90.6	94.6	102.6
θ_6	75.3	83.5	87.8	91.9	99.4
μ	76.9	88.7	93.0	97.1	109.7
σ	11.3	13.3	14.7	16.3	20.3
τ	3.7	10.1	14.0	19.4	40.5



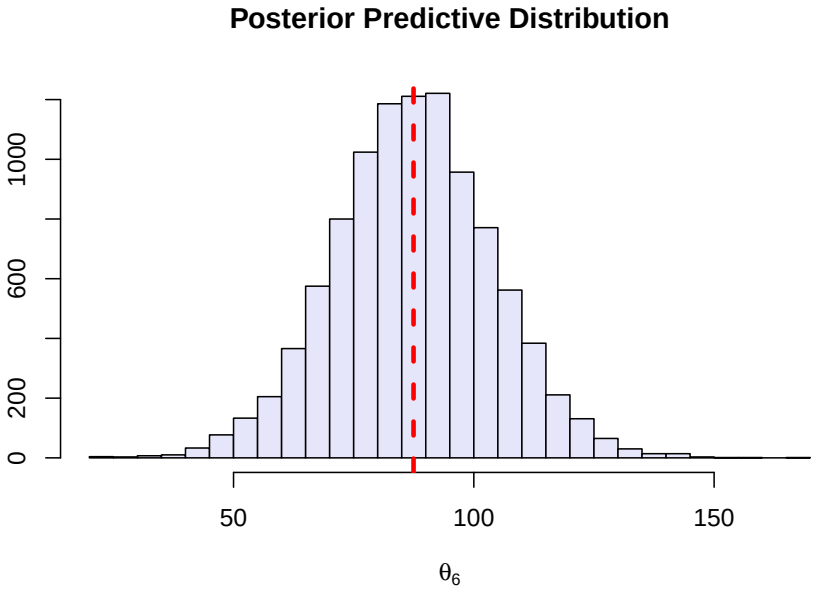
i. **Posterior Distribution of the Mean of Quality Measurements in Machine 6**:





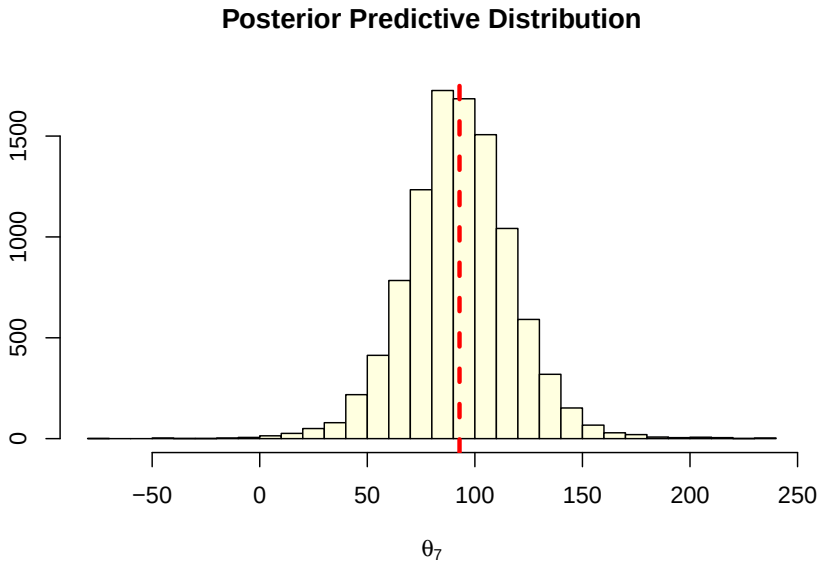
ii. **Predictive Distribution for Another Quality Measurement in Machine 6**:

2.5%	25%	50%	75%	97.5%
54.59365	76.53365	87.37981	98.35327	120.314



iii. **Posterior Distribution of the Mean of Quality Measurements in Machine 7**:

2.5%	25%	50%	75%	97.5%
43.72623	77.53501	92.49672	108.2411	142.164



Problem 8:

Generate/simulate $n = 120$ observations from a three-component mixture Gaussian model with:

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

Use these n data points to estimate p_i, μ_i, and σ_i^2 using:

a. **EM**: Implement EM to find their MLE.

b. **MCMC** (Gibbs): Use proper diffused prior on p_i, μ_i, and σ_i^2, and implement the Gibbs algorithm to find the posterior distribution of p_i, μ_i, and σ_i^2.

c. Summarize the results and compare between the two approaches. Are there any problems observed during the sampling or maximum likelihood estimation?

Solution

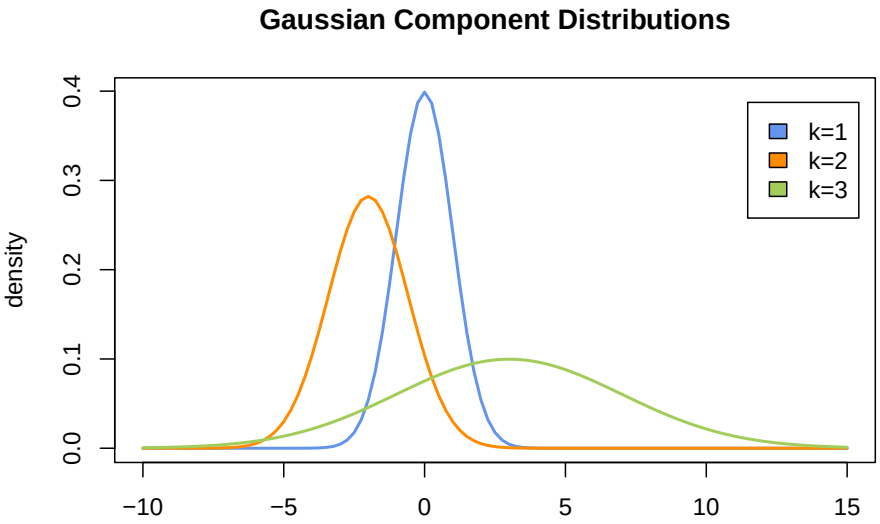
Model: For $j = 1, 2, ..., J$ observations and $k = 1, 2, ..., K$ mixture components with K=3,

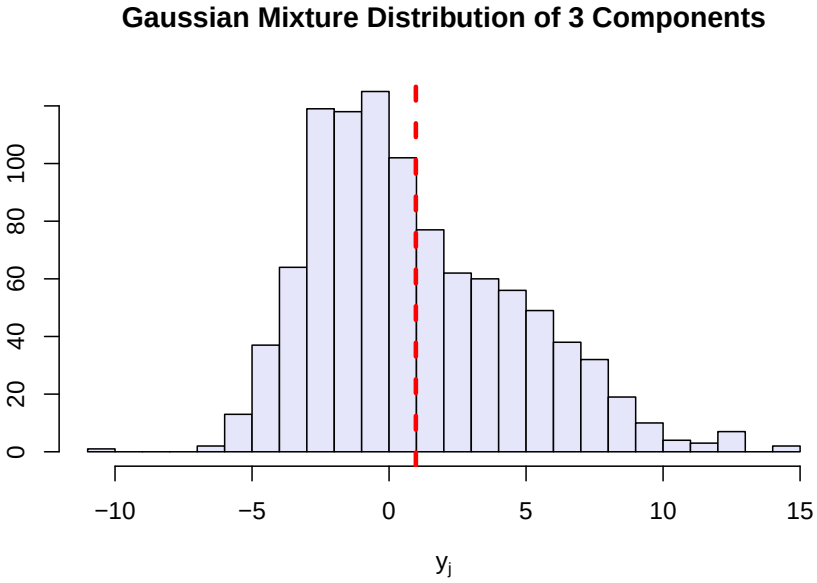
$$y|z_k, \theta \sim \mathrm{N}(\mu_k, \sigma_k^2)$$

$$z|p \sim \mathrm{Multinomial}(1; p_1, p_2, ..., p_k), \qquad \text{where} \quad z_k = \begin{cases} 1 & \text{if drawn from } k^{\text{th}} \text{ mixture component,} \\ 0 & \text{otherwise.} \end{cases}$$

$$p(y, z|\theta, p) = p(z|p)p(y|z, \theta)$$
$$= \prod_{j=1}^{J} \prod_{k=1}^{K} \Big(p_k \cdot p(y_j|\theta_k) \Big)^{z_{jk}}$$

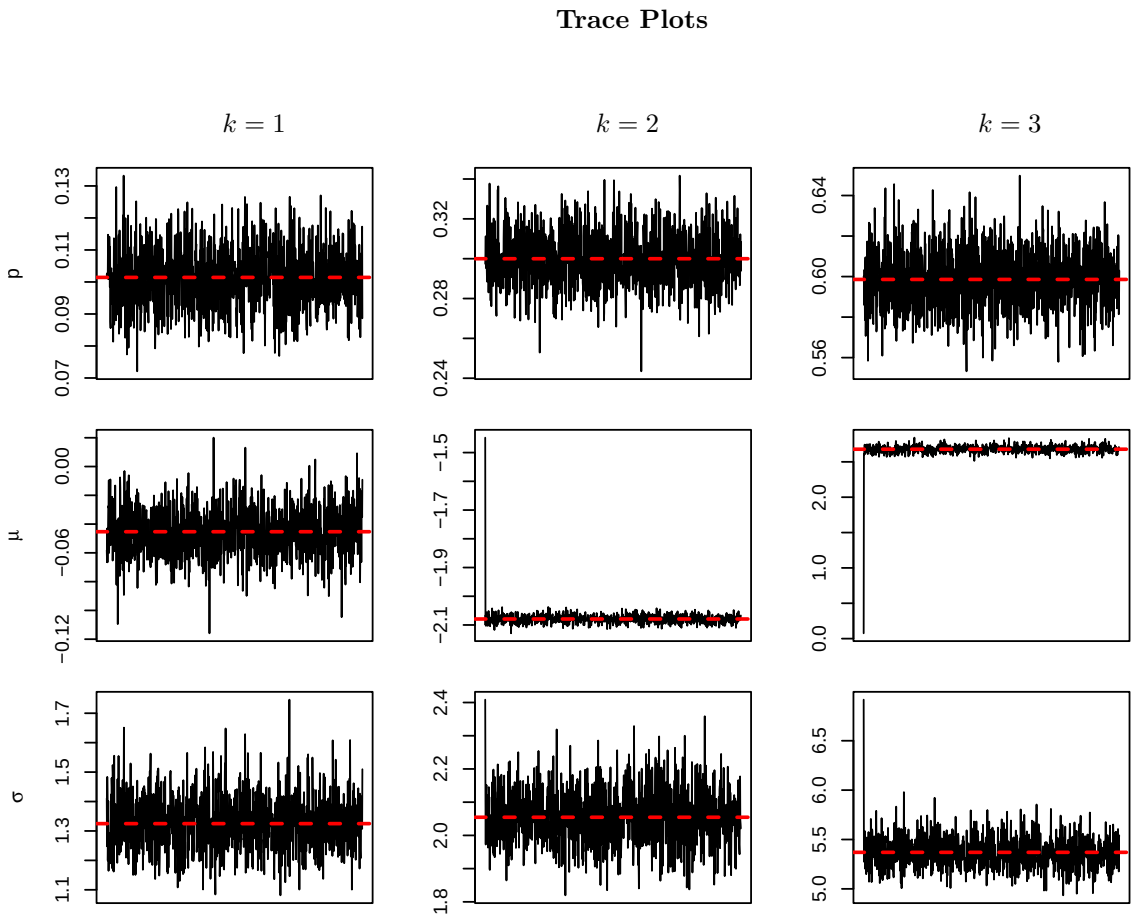
a. **EM Algorithm Results**:





k	p_k	μ_k	σ_k
1	0.1014897	-0.0453205	0.9357345
2	0.3295230	-2.0805594	1.4539220
3	0.5689873	2.6877519	3.8001156

b. **Gibbs MCMC Results**[2]:



k	p_k	μ_k	σ_k
1	0.1014228	-0.0453122	1.324895
2	0.2999950	-2.0791563	2.054639
3	0.5985822	2.6786823	5.369900

c. When comparing estimates of all parameters from both methods, we see that while the Gibbs sampler performed slightly better than the EM algorithm with respect to p_k, it overestimated values for σ_k more than did the MLE approach. Regarding μ_k however, it appears that both methods performed equally well.

[2]Note that estimates are equal to the means of the corresponding chains excluding the first iteration due to extreme values.

Code

```
#### Q1 PART A ####

### Importance Sampler Algorithm
ImpSampler <- function(nSamples,
                       logTargetDensityFunc,
                       logProposalDensityFunc,
                       proposalNewFunc,
                       rejectionControlConstant=NULL){

  x_draws <- replicate(nSamples, proposalNewFunc()) # n draws of x from g(.)
  log_f <- logTargetDensityFunc(x_draws) # log target density function log-f(.)
  log_g <- logProposalDensityFunc(x_draws) # log proposal density function log-g(.)
  log_w <- log_f-log_g # log importance weights
  log_w <- log_w-max(log_w) # stabilizing
  w <- exp(log_w) # weights in original scale

  ESS <- function(w){ # function to calculate ESS
    norm_w <- w/sum(w) # normalized weights
    ess <- 1/sum(norm_w^2) # ESS (ESS=(sum(w)^2)/sum(w^2) <-non-normalized weights)
    return(ess)
  }

  # if c=NULL, returns ESS, draws, and log of importance weights
  if (is.null(rejectionControlConstant)){
    return(list(ESS=ESS(w),
                Draws=x_draws,
                LogWeights=log_w))
  # if c!=NULL, returns ESS, draws, acceptance rate, # of accepted samples, and log of weights
  } else{
    c <- rejectionControlConstant

    accepted <- 0 # number of accepted draws
    mod_w <- c() # modified weights

    for (i in 1:nSamples){ # (can be done without a for loop)
      r <- min(1, w[i]/c) # calculates r for each weight
      u <- runif(1) # random uniform value
      # if r>u, accepts draw and modifies weight
      if (r>u){
        accepted <- accepted+1
        mod_w[i] <- w[i]/r
      # if r!>u, rejects draw and leaves weight as is
      } else{
        mod_w[i] <- w[i]
      }
    }
    acceptance_rate <- accepted/nSamples # sample acceptance rate
    log_mod_w <- log(mod_w) # log modified (importance) weights
    return(list(ESS=ESS(mod_w),
                Draws=x_draws,
                Accepted=accepted,
                AcceptanceRate=acceptance_rate,
```

```
                LogWeights=log_mod_w))
  }
}
```

```
#### Q1 PART C ####
set.seed(47)

## Target Density Function
target_density <- function(x){
  td_1 <- (1/3)*dnorm(x, -2, 1)
  td_2 <- (2/3)*dnorm(x, 2, 1)
  td <- td_1+td_2
  return(td)
}

## Proposal Density Function
proposal_density <- function(x){
  pd <- dnorm(x, 0, 3)
  return(pd)
}

## Plotting Target and Proposal Density Curves
curve(target_density(x), -10, 10,
      add=FALSE, col="blue", lwd=2, ylab="density")
curve(proposal_density(x), -10, 10,
      add=TRUE, col="red", lwd=2, ylab="density")
legend(-9.5, 0.245,
       legend=c("Target Density", "Proposal Density"),
       fill=c("blue","red"))
```

```
set.seed(47)
#### Q1 PART E ####

## Log Target Density Function
logTargetDensityFunc <- function(x){
  td_1 <- (1/3)*dnorm(x, -2, 1)
  td_2 <- (2/3)*dnorm(x, 2, 1)
  td <- td_1+td_2
  return(log(td))
}

## Log Proposal Density Function
logProposalDensityFunc <- function(x){
  pd <- dnorm(x, 0, 3)
  return(log(pd))
}

## Random Draw from Proposal Density
proposalNewFunc <- function(){
  pd_draw <- rnorm(1, 0, 3)
}

## Running Importance Sampler Algorithm
imps_1 <- ImpSampler(nSamples=5000, # n=5000 samples
```

```
                    logTargetDensityFunc=logTargetDensityFunc,
                    logProposalDensityFunc=logProposalDensityFunc,
                    proposalNewFunc=proposalNewFunc,
                    rejectionControlConstant=NULL) # no rejection constant

## ESS Estimate
imps_1$ESS  # ESS ~ 3199.363

## Parameter Estimates
m1 <- sum(exp(imps_1$LogWeights)*imps_1$Draws)/sum(exp(imps_1$LogWeights)) # E(X)
m2 <- sum(exp(imps_1$LogWeights)*(imps_1$Draws)^2)/sum(exp(imps_1$LogWeights)) # E(X^2)
t <- sum(exp(imps_1$LogWeights)*exp(imps_1$Draws))/sum(exp(imps_1$LogWeights)) # E(e^X)

c(m1, m2, t)
```

```
set.seed(47)
#### Q1 PART F ####

## Parameters
mu_1 <- 2/3
mu_2 <- 5
theta <- (1/3)*(exp(-3/2)+(2*exp(5/2)))

## Running Importance Sampler Algorithm with C

c <- 1:10 # rejection control constants

e_c <- c() # ESS estimates
a_c <- c() # acceptance rates
mu_1_error <- c() # mu_1 estimate errors
mu_2_error <- c() # mu_2 estimate errors
theta_error <- c() # theta estimate errors

for (i in c){
  imps_2 <- ImpSampler(nSamples=5000, # n=5000 samples
                       logTargetDensityFunc=logTargetDensityFunc,
                       logProposalDensityFunc=logProposalDensityFunc,
                       proposalNewFunc=proposalNewFunc,
                       rejectionControlConstant=i) # rejection constants

  e_c <- c(e_c, imps_2$ESS)
  a_c <- c(a_c, imps_2$AcceptanceRate)

  # parameter estimates
  hat_mu_1 <- sum(exp(imps_2$LogWeights)*imps_2$Draws)/sum(exp(imps_2$LogWeights))
  hat_mu_2 <- sum(exp(imps_2$LogWeights)*(imps_2$Draws)^2)/sum(exp(imps_2$LogWeights))
  hat_theta <- sum(exp(imps_2$LogWeights)*exp(imps_2$Draws))/sum(exp(imps_2$LogWeights))

  # parameter estimate error
  m1_err <- abs(hat_mu_1-mu_1)
  m2_err <- abs(hat_mu_2-mu_2)
  t_err <- abs(hat_theta-theta)

  mu_1_error <- c(mu_1_error, m1_err)
```

```
  mu_2_error <- c(mu_2_error, m2_err)
  theta_error <- c(theta_error, t_err)
}

imps_2_results <- as.data.frame(cbind(c, e_c, a_c,
                                      mu_1_error,
                                      mu_2_error,
                                      theta_error))
```

```
## Plots

#par(mfrow=c(2, 3))

# ESS (e_c)
plot(c, e_c, col="purple", pch=19,
     main=TeX(r"($e_c$)"),
     xlab="Rejection Control Constant (c)",
     ylab="ESS")

# Sample Acceptance Rate (a_c)
plot(c, a_c, col="seagreen", pch=19,
     main=TeX(r"($a_c$)"),
     xlab="Rejection Control Constant (c)",
     ylab="Sample Acceptance Rate")
```

```
par(mfrow=c(2, 3))

# Error in the Estimate of mu_1
plot(c, mu_1_error, col="orange", pch=19, ylim=c(0, 1),
     main=TeX(r"($|\hat{\mu}_1-\mu_1|$)"),
     xlab=" ",
     ylab="Error in the Estimate")

# Error in the Estimate of mu_2
plot(c, mu_2_error, col="orange", pch=19, ylim=c(0, 1),
     main=TeX(r"($|\hat{\mu}_2-\mu_2|$)"),
     xlab="Rejection Control Constant (c)",
     ylab=" ")

# Error in the Estimate of theta
plot(c, theta_error, col="orange", pch=19, ylim=c(0, 1),
     main=TeX(r"($|\hat{\theta}-\theta|$)"),
     xlab=" ",
     ylab=" ")
```

```
set.seed(47)
#### Q2 PART A ####

## Generating Data from Model

# groups
J <- 10

# theta_j
```

```r
alpha <- rlst(n=1, df=4, mu=0, sigma=2) # location-scale t-dist. with 4 df
beta <- rlst(n=1, df=4, mu=0, sigma=1) # location-scale t-dist. with 4 df
x_j <- runif(J) # uniform x_j
theta_j = 1/(1+exp(-(alpha+(beta*x_j)))) # inverse logit

# sample size
n_j <- rpois(J, 5)+1 # zero-truncated Poisson (Poisson shifted to the right by 1)

# y_j
y_j <- rbinom(J, n_j, theta_j) # binomial

# dataset
dataset <- data.frame(J=1:J,
                      alpha=rep(alpha, J),
                      beta=rep(beta, J),
                      x_j=x_j,
                      theta_j=theta_j,
                      n_j=n_j,
                      y_j=y_j)
```

```r
set.seed(47)
#### Q2 PART B ####

# log-likelihood
log_lik <- function(v, x_j, n_j, y_j){
  alpha <- v[1]
  beta <- v[2]
  expression <- alpha+(beta*x_j)
  ll <- sum((y_j*expression)-(n_j*log(1+exp(expression))))
  return(ll)
}

# MLE
optim_output <- optim(par=c(0.01, 0.01),
                      fn=log_lik,
                      y=y_j, n=n_j, x=x_j,
                      control=list(fnscale=-1),
                      hessian=TRUE)

mode <- optim_output$par # maximizer (mode)

## Rejection Sampling

m <- 1000 # m=1,000 posterior draws

# accepted posterior draws
alpha_post <- numeric(m)
beta_post <- numeric(m)

total <- 0 # total iterations
accepted <- 0 # total accepted samples

# enveloping function/distribution
Mg <- log_lik(mode, x_j, n_j, y_j)
```

```r
# generating m posterior draws of (alpha, beta)
# *NOTE: using log-scale
while (accepted!=m){
  total <- total+1 # counts the number of samples (iterations needed)
  alpha_draw <- rlst(n=1, df=4, mu=0, sigma=2)
  beta_draw <- rlst(n=1, df=4, mu=0, sigma=1)
  param <- c(alpha_draw, beta_draw)
  p <- log_lik(param, x_j, n_j, y_j)-Mg # probability of acceptance (same as r in Q1)
  u <- log(runif(1)) # can also exponentiate p
  if(u<p){
    accepted <- accepted+1 # counts the number of accepted samples
    # accepted (alpha, beta) sample
    alpha_post[accepted] <- alpha_draw
    beta_post[accepted] <- beta_draw
  }
}

acceptance_rate <- accepted/total # acceptance rate ~ 0.06938181

## Posterior Estimates

mean(alpha_post) # posterior mode of alpha ~ 2.766464
mean(beta_post) # posterior mode of beta ~ 0.8042834

# histogram of posterior alpha draws
hist(alpha_post, col="lavender",
     main=TeX(r"(Posterior Draws of $\alpha$)"), xlab=TeX(r"($\alpha$)"))
abline(v=mean(alpha_post), lty=2, col="red")

# histogram of posterior beta draws
hist(beta_post, col="lavender",
     main=TeX(r"(Posterior Draws of $\beta$)"), xlab=TeX(r"($\beta$)"))
abline(v=mean(beta_post), lty=2, col="red")
```

```r
set.seed(47)
#### Q2 PART C ####

# posterior mode (vector)
post_mode <- c(mean(alpha_post), mean(beta_post))

# covariance matrix
sigma <- matrix(c(var(alpha_post),
                  cov(alpha_post, beta_post),
                  cov(alpha_post, beta_post),
                  var(beta_post)),
                ncol=2)

# normal approximation centered at the posterior mode (1000 samples)
n_approx <- mvrnorm(n=1000, mu=post_mode, Sigma=sigma)

# contour plot
colors <- c("Proposal Distribution"="blue2",
            "Mode-Centered Normal"="green3")
```

```
ggplot(data.frame(alpha_post, beta_post, n_approx[,1], n_approx[,2])) +
  stat_density2d(aes(alpha_post, beta_post,
                     color="Proposal Distribution")) +
  stat_density2d(aes(n_approx[,1], n_approx[,2],
                     color="Mode-Centered Normal")) +
  geom_point(aes(alpha_post, beta_post),
             size=0.25) +
  labs(title="Posterior Density Approximations",
       x=TeX(r"($\alpha$)"),
       y=TeX(r"($\beta$)")) +
  scale_color_manual(values=colors) +
  theme(legend.position="bottom",
        legend.title=element_blank(),
        plot.title=element_text(hjust=0.5))
```

```
set.seed(47)
#### Q2 PART D ####

## Importance Sampling
S <- 1000
t_draws <- rmt(S, post_mode, sigma) # draws
t_weights <- numeric(S) # importance weights

for (i in 1:S){
  target <- log_lik(t_draws[i,], x_j, n_j, y_j) +
    dt(t_draws[i, 1]/2, df=4, log=TRUE) +
    dt(t_draws[i, 2], df=4, log=TRUE)
  proposal <- dmt(t_draws[i,], post_mode, sigma, df=4, log=TRUE)
  t_weights[i] <- exp(target-proposal)
}

alpha_post_t <- sum(t_weights*t_draws[,1]/sum(t_weights)) # expected alpha | y
beta_post_t <- sum(t_weights*t_draws[,2]/sum(t_weights))  # expected beta | y

c(alpha_post_t, beta_post_t)
```

```
set.seed(47)
#### Q2 PART E ####

## ESS
norm_t_weights <- t_weights/sum(t_weights) # normalized weights
S_eff <- 1/sum(norm_t_weights^2) # ESS ~ 897.122
```

```
set.seed(47)
#### Q3 PART A ####

## Normal Approximation

# bioassay data (table 3.1, pg. 74)
x_i <- c(-0.86, -0.30, -0.05, 0.73) # dose in log(g/ml)
n_i <- c(5, 5, 5, 5) # number of animals
y_i <- c(0, 1, 3, 5) # number of deaths
bioassay_data <- as.data.frame(x_i, n_i, y_i) # df
```

```r
# log posterior
bioassay_func <- function(v, x=x_i, n=n_i, y=y_i){
  alpha <- v[1]
  beta <- v[2]
  t <- alpha+(beta*x)
  inverse_logit <- exp(t)/(1+exp(t))
  log_lik <- (y*log(inverse_logit))+((n-y)*log(1-inverse_logit)) # =y*t-n*log1p(exp(t))
  return(sum(log_lik))
}

# optimizing log posterior
b_optim_output <- optim(par=c(0.01, 0.01),
                        fn=bioassay_func,
                        x=x_i, n=n_i, y=y_i,
                        control=list(fnscale=-1),
                        hessian=TRUE)

# posterior mode and covariance matrix
bioassay_mode <- b_optim_output$par # (alpha, beta) ~ (0.8466206, 7.7476333)
bioassay_Sigma <- solve(-b_optim_output$hessian) # inverse information
```

```r
set.seed(47)

## Importance Resampling (without replacement)

# obtaining S draws from normal approx. with positive beta values
S <- 10000
n_draws <- rmnorm(S, bioassay_mode, bioassay_Sigma)

# calculating importance weights
n_weights <- numeric(S)
for (i in 1:S){
  target <- bioassay_func(n_draws[i,], x_i, n_i, y_i)
  proposal <- dmnorm(n_draws[i,], bioassay_mode, bioassay_Sigma, log=TRUE)
  n_weights[i] <- exp(target-proposal)
}

# obtaining k<S samples via importance resampling
k <- 1000
k_indx <- sample(1:S, k, replace=FALSE, prob=n_weights)
k_draws <- n_draws[k_indx, ]

## HISTOGRAMS
# k alpha samples
hist(k_draws[,1], breaks=30, col="lightblue",
     main=TeX(r"(Samples of $\alpha$ (k=1,000))"),
     xlab=TeX(r"($\alpha$)"),
     ylab=" ")
abline(v=mean(k_draws[,1]), lty=2, lwd=3, col="red") # mean ~ 1.213126
# k beta samples
hist(k_draws[,2], breaks=30, col="lightblue",
     main=TeX(r"(Samples of $\beta$ (k=1,000))"),
     xlab=TeX(r"($\beta$)"),
     ylab=" ")
```

```
abline(v=mean(k_draws[,2]), lty=2, lwd=3, col="red") # mean ~ 10.49802
```

```
## SCATTERPLOT (comparison to 3.3b)
plot(k_draws[,1], k_draws[,2],
     xlim=c(-4, 10), ylim=c(-10, 40),
     pch=20, col="darkblue",
     main=TeX(r"(Samples of $(\alpha, \beta)$ (k=1,000))"),
     xlab=TeX(r"($\alpha$)"),
     ylab=TeX(r"($\beta$)"))
```

```
set.seed(47)
#### Q3 PART B ####

## IMPORTANCE RATIOS
hist(n_weights, breaks=30, col="lavender",
     main="Distribution of Importance Weights",
     xlab=TeX(r"($w(\phi^s)$)"),
     ylab=" ")
```

```
set.seed(47)
#### Q3 PART C ####

## Importance Resampling (with replacement)

# obtaining k<S samples via importance resampling
k <- 1000
k_indx2 <- sample(1:S, k, replace=TRUE, prob=n_weights)
k_draws2 <- n_draws[k_indx2, ]

## HISTOGRAMS
# k alpha samples
hist(k_draws2[,1], breaks=30, col="lightyellow",
     main=TeX(r"(Samples of $\alpha$ with REPLACEMENT (k=1,000))"),
     xlab=TeX(r"($\alpha$)"),
     ylab=" ")
abline(v=mean(k_draws2[,1]), lty=2, lwd=3, col="red") # mean ~ 1.218361
# k beta samples
hist(k_draws2[,2], breaks=30, col="lightyellow",
     main=TeX(r"(Samples of $\beta$ with REPLACEMENT (k=1,000))"),
     xlab=TeX(r"($\beta$)"),
     ylab=" ")
abline(v=mean(k_draws2[,2]), lty=2, lwd=3, col="red") # mean ~ 10.87927
```

```
## SCATTERPLOTS
# resampling without replacement
plot(k_draws[,1], k_draws[,2],
     xlim=c(-4, 10), ylim=c(-10, 40),
     pch=20, col="darkblue",
     main=TeX(r"(Samples of $(\alpha, \beta)$ (k=1,000))"),
     xlab=TeX(r"($\alpha$)"),
     ylab=TeX(r"($\beta$)"))
# resampling WITH replacement
plot(k_draws2[,1], k_draws2[,2],
     xlim=c(-4, 10), ylim=c(-10, 40),
```

```
     pch=20, col="darkred",
     main=TeX(r"(Samples of $(\alpha, \beta)$ with REPLACEMENT (k=1,000))"),
     xlab=TeX(r"($\alpha$)"),
     ylab=TeX(r"($\beta$)"))
```

```
set.seed(9)
#### Q4 ####

### Metropolis Algorithm (on bioassay data)

## target distribution <- log posterior from Q3
# bioassay_func(v[a,b], x_i, n_i, y_i)

## starting distribution <- normal approximation from Q3
# dmnorm(v[a,b], bioassay_mode, bioassay_Sigma, log=TRUE)

# jumping distribution <- normal approximation centered at v at current time t
# dmnorm(x, v_t, bioassay_Sigma, log=TRUE)

# sampling function for jumping distribution
jumping_dist_draw <- function(v_t){
  rmnorm(1, as.vector(v_t), bioassay_Sigma)
}


## ALGORITHIM

K <- 1000 # number of desired samples
v <- rmnorm(1, bioassay_mode, bioassay_Sigma) # starting point

# storing matrices
total <- matrix(0, nrow=K*2, ncol=2) # all iterations (transition distribution?)
accepted <- c() # accepted samples (posterior approx.)

# counters
accept <- 0
i <- 0

while (accept!=K){
  i <- i+1
  v_prime <- jumping_dist_draw(v)
  imp <- exp(bioassay_func(v_prime)-bioassay_func(v))
  r <- min(1, imp)
  u <- runif(1)
  if(u<r){
    accept <- accept+1
    accepted <- rbind(accepted, v_prime)
    total[i,] <- v_prime
  } else{
    total[i,] <- v
  }
  v <- total[i,]
}
```

```r
total <- subset(total, total[,1]!=0 & total[,2]!=0) # removing leftover empty rows

# ACCEPTANCE RATE
nrow(accepted)/nrow(total) # ~ 0.58

# STATISTICS
mean_alpha <- mean(accepted[,1])
mean_beta <- mean(accepted[,2])
sd_alpha <- sd(accepted[,1])
sd_beta <- sd(accepted[,2])
c(mean_alpha, mean_beta, sd_alpha, sd_beta)
```

```r
# HISTOGRAMS
hist(accepted[,1], breaks=30, # alpha
     col="darkseagreen2",
     main=TeX(r"(Samples of $\alpha$ (K=1,000))"),
     xlab=TeX(r"($\alpha$)"),
     ylab=" ")
abline(v=mean(accepted[,1]), lty=2, lwd=3, col="red")

hist(accepted[,2], breaks=30, # beta
     col="darkseagreen2",
     main=TeX(r"(Samples of $\beta$ (K=1,000))"),
     xlab=TeX(r"($\beta$)"),
     ylab=" ")
abline(v=mean(accepted[,2]), lty=2, lwd=3, col="red")
```

```r
# SCATTERPLOT (posterior approximation)
plot(accepted[,1], accepted[,2],
     xlim=c(-4, 10), ylim=c(-10, 40),
     pch=20, col="cornflowerblue",
     main=TeX(r"(Samples of $(\alpha, \beta)$ (K=1,000))"),
     xlab=TeX(r"($\alpha$)"),
     ylab=TeX(r"($\beta$)"))
```

```r
# TRACE PLOTS

# alpha
plot(total[,1], type="l",
     main=TeX(r"($\alpha$)"),
     xlab="Iteration",
     ylab=TeX(r"($\alpha$)"))
abline(h=mean(accepted[,1]), lty=2, lwd=3, col="red")

# beta
plot(total[,2], type="l",
     main=TeX(r"($\beta$)"),
     xlab="Iteration",
     ylab=TeX(r"($\beta$)"))
abline(h=mean(accepted[,2]), lty=2, lwd=3, col="red")
```

```r
set.seed(47)
#### Q5 ####
```

```r
J <- 6 # number of machines
n <- rep(5, J) # number of observations per machine

# machines
y1 <- c(83, 92, 92, 46, 67)
y2 <- c(117, 109, 114, 104, 87)
y3 <- c(101, 93, 92, 86, 67)
y4 <- c(105, 119, 116, 102, 116)
y5 <- c(79, 97, 103, 79, 92)
y6 <- c(57, 92, 104, 77, 100)

# sample means and sds
y_bar <- c(mean(y1), mean(y2), mean(y3), mean(y4), mean(y5), mean(y6))
s <- c(sd(y1), sd(y2), sd(y3), sd(y4), sd(y5), sd(y6))

# update functions
theta_post <- function(J, n, y_bar, mu, sigma, tau){
  V_theta <- 1/((1/(tau^2))+(n/(sigma^2)))
  theta_hat <- V_theta*((mu/(tau^2))+(n*y_bar/(sigma^2)))
  rnorm(J, theta_hat, sqrt(V_theta))
}

mu_post <- function(J, theta, tau){
  mu_hat <- mean(theta)
  var <- (tau^2)/J
  rnorm(1, mu_hat, sqrt(var))
}

sigma_post <- function(n, y_bar, s, theta){
  sigma2_hat <- sum(((n-1)*s^2)+(n*(y_bar-theta)^2))/sum(n)
  sqrt(rinvchisq(1, sum(n), sigma2_hat))
  # = sqrt((sum(n)*sigma2_hat)/rchisq(1, sum(n)))
}

tau_post <- function(J, mu, theta){
  tau2_hat <- sum((theta-mu)^2)/(J-1)
  sqrt(rinvchisq(1, J-1, tau2_hat))
  # = sqrt(((J-1)*tau2_hat)/rchisq(1, J-1))
}

# Gibbs sampler algorithm
machine_chain <- function(iters, J, n, s, theta0, mu0, sigma0, tau0){

  # initializing chains
  theta_chain <- matrix(NA, iters, J)
  mu_chain <- rep(NA, iters)
  sigma_chain <- rep(NA, iters)
  tau_chain <- rep(NA, iters)

  # starting values
  theta <- theta0
  mu <- mu0
  sigma <- sigma0
```

```
  tau <- tau0

  # updating params
  for(i in 1:iters){
    theta <- theta_post(J, n, y_bar, mu, sigma, tau)
    mu <- mu_post(J, theta, tau)
    sigma <- sigma_post(n, y_bar, s, theta)
    tau <- tau_post(J, mu, theta)

    theta_chain[i,] <- theta
    mu_chain[i] <- mu
    sigma_chain[i] <- sigma
    tau_chain[i] <- tau
    }

  list(theta_chain=theta_chain, mu_chain=mu_chain,
       sigma_chain=sigma_chain, tau_chain=tau_chain)
}

# starting values
theta0 <- y_bar
mu0 <- mean(theta0)
sigma0 <- sqrt(mean(s^2))
tau0 <- sd(y_bar)

# chain
C <- 10000
chain <- machine_chain(C, J, n, s, theta0, mu0, sigma0, tau0)
theta_chain <- chain$theta_chain
mu_chain <- chain$mu_chain
sigma_chain <- chain$sigma_chain
tau_chain <- chain$tau_chain
```

```
# results
post_samps <- matrix(NA, J+3, 5)
probs <- c(0.025, 0.25, 0.50, 0.75, 0.975)

for(j in 1:J){
  post_samps[j,] <- quantile(theta_chain[,j], probs)
}
post_samps[J+1,] <- quantile(mu_chain, probs) # mu
post_samps[J+2,] <- quantile(sigma_chain, probs) # sigma
post_samps[J+3,] <- quantile(tau_chain, probs) # tau

rownames(post_samps) <- c(paste("$\\theta_", 1:J, "$", sep=""),
                          "$\\mu$", "$\\sigma$", "$\\tau$")
colnames(post_samps) <- paste(probs*100, "%", sep="")
post_samps_df <- as.data.frame(round(post_samps, 1))
```

```
par(mfrow = c(3, 3))
hist(theta_chain[,1], breaks=30,
     main=" ", ylab=" ", xlab=TeX(r"($\theta_1$)"), col="lightblue")
hist(theta_chain[,2], breaks=30,
     main="Posterior Densities", ylab=" ", xlab=TeX(r"($\theta_2$)"), col="lightblue")
```

```r
hist(theta_chain[,3], breaks=30,
     main=" ", ylab=" ", xlab=TeX(r"($\theta_3$)"), col="lightblue")
hist(theta_chain[,4], breaks=30,
     main=" ", ylab=" ", xlab=TeX(r"($\theta_4$)"), col="lightblue")
hist(theta_chain[,5], breaks=30,
     main=" ", ylab=" ", xlab=TeX(r"($\theta_5$)"), col="lightblue")
hist(theta_chain[,6], breaks=30,
     main=" ", ylab=" ", xlab=TeX(r"($\theta_6$)"), col="lightblue")
hist(mu_chain, breaks=30,
     main=" ", ylab=" ", xlab=TeX(r"($\mu$)"), col="lavender")
hist(sigma_chain, breaks=30,
     main=" ", ylab=" ", xlab=TeX(r"($\sigma$)"), col="lightpink")
hist(tau_chain, breaks=30,
     main=" ", ylab=" ", xlab=TeX(r"($\tau$)"), col="lightgreen")
```

```r
## i
plot(theta_chain[,6], type="l",
     main="Trace Plot",
     ylab=TeX(r"($\theta_6$)"), xlab="Iteration")
abline(h=mean(theta_chain[,6]), lty=2, lwd=3, col="red") # mean ~ 87.6826
```

```r
hist(theta_chain[,6], breaks=30,
     main="Posterior Distribution", ylab=" ", xlab=TeX(r"($\theta_6$)"), col="lightblue")
abline(v=mean(theta_chain[,6]), lty=2, lwd=3, col="red") # mean ~ 87.6826
```

```r
set.seed(47)

## ii

# simplest method
pred_dist_y6 <- rnorm(10000, theta_chain[,6], sigma_chain)

# alternative method
theta_sigma_chain <- cbind(theta_chain[,6], sigma_chain)
theta_sigma_chain_samp <- theta_sigma_chain[sample(1:10000, 5000), ]

pred_dist_y6_draws <- c()

for (i in 1:5000){
  draw <-  rnorm(1, theta_sigma_chain_samp[i,1], theta_sigma_chain_samp[i,2])
  pred_dist_y6_draws <- c(pred_dist_y6_draws, draw)
}

# results
quantile(pred_dist_y6, probs=probs)
#quantile(pred_dist_y6_draws, probs=probs)
```

```r
hist(pred_dist_y6, breaks=30,
     main="Posterior Predictive Distribution",
     ylab=" ", xlab=TeX(r"($\theta_6$)"), col="lavender")
abline(v=mean(pred_dist_y6), lty=2, lwd=3, col="red")
#hist(pred_dist_y6_draws)
```

```r
set.seed(47)
```

```r
## iii

theta_7 <- rnorm(10000, mu_chain, tau_chain)
pred_dist_y7 = rnorm(10000, theta_7, sigma_chain)

quantile(pred_dist_y7, probs=probs)
```

```r
hist(pred_dist_y7, breaks=30,
     main="Posterior Predictive Distribution",
     ylab=" ", xlab=TeX(r"($\theta_7$)"), col="lightyellow")
abline(v=mean(pred_dist_y7), lty=2, lwd=3, col="red")
```

```r
set.seed(47)
#### Q8 PART A ####

## EM Algorithm

mu_k <- c(0, -2, 3)
sigma_k <-  sqrt(c(1, 2, 16))
p_k <- c(0.1, 0.3, 0.6)

z_jk <- rmultinom(n=1000, size=1, prob=p_k)
y_j <- rnorm(1000, mean=colSums(mu_k*z_jk), sd=colSums(sigma_k*z_jk))

curve(dnorm(x, mu_k[1], sigma_k[1]), -10, 15,
      add=FALSE, col="cornflowerblue", lwd=2, ylab="density",
      main="Gaussian Component Distributions", xlab=" ")
#abline(v=mu_k[1], lty=2, lwd=2, col="red")
curve(dnorm(x, mu_k[2], sigma_k[2]), -10, 15,
      add=TRUE, col="darkorange", lwd=2, ylab="density")
#abline(v=mu_k[2], lty=2, lwd=2, col="red")
curve(dnorm(x, mu_k[3], sigma_k[3]), -10, 15,
      add=TRUE, col="darkolivegreen3", lwd=2, ylab="density")
#abline(v=mu_k[3], lty=2, lwd=2, col="red")
legend(11.47, 0.3874,
       legend=c("k=1", "k=2", "k=3"),
       fill=c("cornflowerblue", "darkorange", "darkolivegreen3"))
```

```r
hist(y_j, breaks=30,
     main="Gaussian Mixture Distribution of 3 Components",
     ylab=" ", xlab=TeX(r"($y_j$)"),
     col="lavender")
abline(v=mean(y_j), lty=2, lwd=3, col="red")
```

```r
set.seed(47)

mixture_model <- matrix(c(dnorm(y_j, mean=mu_k[1], sd=sigma_k[1]),
                          dnorm(y_j, mean=mu_k[2], sd=sigma_k[2]),
                          dnorm(y_j, mean=mu_k[3], sd=sigma_k[3])),
                        nrow=length(y_j), ncol=3)

EM_steps <- function(mixture_model, w){
  indicator <- mixture_model
  for (j in 1:ncol(mixture_model)){  # E
```

```
    indicator[,j] <- w[j]*indicator[,j]
  }
  indicator <- indicator/rowSums(indicator)
  return(colSums(indicator)/sum(indicator)) # M
  }

EM <- function(iters, mixture_model, w){
  for (i in 1:iters){
    w <- EM_steps(mixture_model, w)
    }
  return(w)
  }

p_estimate <- EM(1000, mixture_model, c(1/3, 1/3, 1/3))

Z <- apply(z_jk, 2, which.max)

mu_estimate <- c()
sigma_estimate <- c()

for (i in 1:3){
  mu_estimate[i] <- mean(y_j[Z==i])
  sigma_estimate[i] <- sd(y_j[Z==i])
}

EM_estimates <- data.frame(k=c(1, 2, 3), p_estimate, mu_estimate, sigma_estimate)
```

```
set.seed(47)
#### Q8 PART B ####

# initial values from prior
p <- c(1/3, 1/3, 1/3)
mu <- rnorm(3, 0, 5)
sigma <- 1/rgamma(3, 0.1, 0.1)

# update functions
p_post <- function(Z){
  rdirichlet(1, c(sum(Z==1), sum(Z==2), sum(Z==3)))
}

mu_post <- function(y_j, Z, mu_prior=c(0, 5), sigma){
  mu <- c()
  for (j in 1:3){
    denominator <- (1/(mu_prior[2]^2))+(sum(Z==j)/(sigma[j]^2))
    mu_hat <- ((mu_prior[1]/(mu_prior[2]^2))+(sum(y_j[Z==j])/(sigma[j]^2)))/denominator
    var <- 1/denominator
    mu[j] <- rnorm(1, mu_hat, var)
  }
  return(mu)
}

sigma_post <- function(y_j, Z, sigma_prior=c(0.1, 0.1), mu){
  sigma <- c()
  for (j in 1:3){
```

```
    alpha <- sigma_prior[1]+(sum(Z==j)/2)
    beta <- sigma_prior[2]+(sum((y_j[Z==j]-mu[j])^2))
    sigma[j] <- sqrt(1/rgamma(1, alpha, beta))
  }
  return(sigma)
}

# Gibbs MCMC
GMCMC <- function(iters, y_j, Z, p0, mu0, sigma0){

  # initializing chains
  p_chain <- matrix(NA, iters, 3)
  mu_chain <- matrix(NA, iters, 3)
  sigma_chain <- matrix(NA, iters, 3)

  # starting values
  p <- p0
  mu <- mu0
  sigma <- sigma0

  # updating params
  for(i in 1:iters){
    p <- p_post(Z)
    mu <- mu_post(y_j=y_j, Z=Z, sigma=sigma)
    sigma <- sigma_post(y_j=y_j, Z=Z, mu=mu)

    p_chain[i,] <- p
    mu_chain[i,] <- mu
    sigma_chain[i,] <- sigma
    }

  list(p_chain=p_chain, mu_chain=mu_chain, sigma_chain=sigma_chain)
}

# chain
GMCMC_chain <- GMCMC(1000, y_j, Z, p, mu, sigma)
p_chain <- GMCMC_chain$p_chain
mu_chain <- GMCMC_chain$mu_chain
sigma_chain <- GMCMC_chain$sigma_chain
```

```
par(mfrow=c(3, 3), mar=c(1.5, 3.74, 0.5, 0.5)+0.1)
#par(mfrow=c(3, 3))

plot(p_chain[,1], type="l",
     main=NULL, ylab=TeX(r"($p$)"), xlab=NULL,
     xaxt="n")
abline(h=mean(p_chain[,1]), lty=2, lwd=2, col="red")

for (i in 2:3){
  plot(p_chain[,i], type="l",
       main=NULL,
       ylab=NULL, xlab=TeX(r"($p$)"),
       xaxt="n", ann=FALSE)
  abline(h=mean(p_chain[,i]), lty=2, lwd=2, col="red")
```

```
}

plot(mu_chain[-c(1),1], type="l",
     main=NULL, ylab=TeX(r"($\mu$)"), xlab=NULL,
     xaxt="n")
abline(h=mean(mu_chain[-c(1), 1]), lty=2, lwd=2, col="red")

for (i in 2:3){
  plot(mu_chain[-c(1),i], type="l",
       main=NULL,
       ylab=NULL, xlab=TeX(r"($\mu$)"),
       xaxt="n", ann=FALSE)
  abline(h=mean(mu_chain[-c(1), i]), lty=2, lwd=2, col="red")
}

plot(sigma_chain[-c(1),1], type="l",
     main=NULL, ylab=TeX(r"($\sigma$)"), xlab=NULL,
     xaxt="n")
abline(h=mean(sigma_chain[-c(1), 1]), lty=2, lwd=2, col="red")

for (i in 2:3){
  plot(sigma_chain[-c(1),i], type="l",
       main=NULL,
       ylab=NULL, xlab=TeX(r"($\sigma$)"),
       xaxt="n", ann=FALSE)
  abline(h=mean(sigma_chain[-c(1), i]), lty=2, lwd=2, col="red")
}
```

```
p_estimate2 <- c()
mu_estimate2 <- c()
sigma_estimate2 <- c()

# Gibbs MCMC estimates excluding the first initial (expreme) value
for (i in 1:3){
  p_estimate2[i] <- mean(p_chain[-c(1), i])
  mu_estimate2[i] <- mean(mu_chain[-c(1), i])
  sigma_estimate2[i] <- mean(sigma_chain[-c(1), i])
}

GMCMC_estimates <- data.frame(k=c(1, 2, 3), p_estimate2, mu_estimate2, sigma_estimate2)
```