MA578 HW5

Benjamin Draves

11/11/2019

Exercise 11.2

In this exercise we look to construct a Metropolis-Hastings algorithm for sampling from the posterior in the Bioassy example of Section 3.7. In this example, we modeled the outcomes of the five animals in each group i as

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

where θ_i is the probability of the death for animals in that group. Moreover, let $\text{logit}(\theta_i) = \alpha + \beta x_i$ where (α, β) are hyperparameters and x_i is the dose of the drug given to animals in group i. With this, the full likelihood can be written as

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

With this, and adopting the constant prior $p(\alpha, \beta) \propto 1$ we can write the full posterior as

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

From here, it is clear that this likelihood cannot be factorized with respect to α, β therefore we need a Metropolis sampler (in comparison to a Gibbs sampler) to draw from this posterior. We propose using a random walk proposal with tunable step size σ_α^2 and σ_β^2.

That is we make proposals $\alpha^* \sim N(\alpha^{(t)}, \sigma_\alpha^2)$ and $\beta^* \sim N(\beta^{(t)}, \sigma_\beta^2)$ and accept each proposal with probability

$$r_\alpha = \min\left\{1, \frac{p(\alpha^*, \beta^{(t)}|y, x_i, n)}{p(\alpha^{(t)}, \beta^{(t)}|y, x_i, n)}\right\} \qquad r_\beta = \min\left\{1, \frac{p(\alpha^{(t)}, \beta^*|y, x_i, n)}{p(\alpha^{(t)}, \beta^{(t)}|y, x_i, n)}\right\}$$

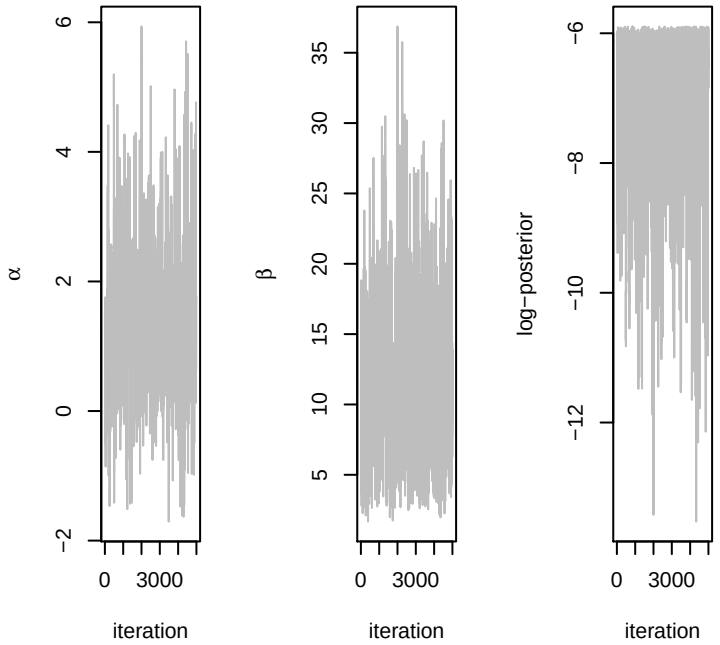
It will be helpful for us to write out this log posterior so we include that expression here

$$
\begin{aligned}
\log(\text{logit}^{-1}(x)) &= \log\left(\frac{\exp(x)}{1+\exp(x)}\right) = x - \log(1+\exp(x)) \\
\log(1-\text{logit}^{-1}(x)) &= \log\left(1-\frac{\exp(x)}{1+\exp(x)}\right) = -\log(1+\exp(x)) \\
\log p(\alpha,\beta|y,n,x) &= \sum_{i=1}^{n} y_i \log(\text{logit}^{-1}(\alpha+\beta x_i)) + (n_i - y_i)\log(1-\text{logit}^{-1}(\alpha+\beta x_i)) \\
&= \sum_{i=1}^{n} y_i(\alpha+\beta x_i) - y_i\log(1+\exp(\alpha+\beta x_i)) - (n_i-y_i)\log(1+\exp(\alpha+\beta x_i)) \\
&= \sum_{i=1}^{n} y_i(\alpha+\beta x_i) - n_i\log(1+\exp(\alpha+\beta x_i))
\end{aligned}
$$

We implement this sampler below.

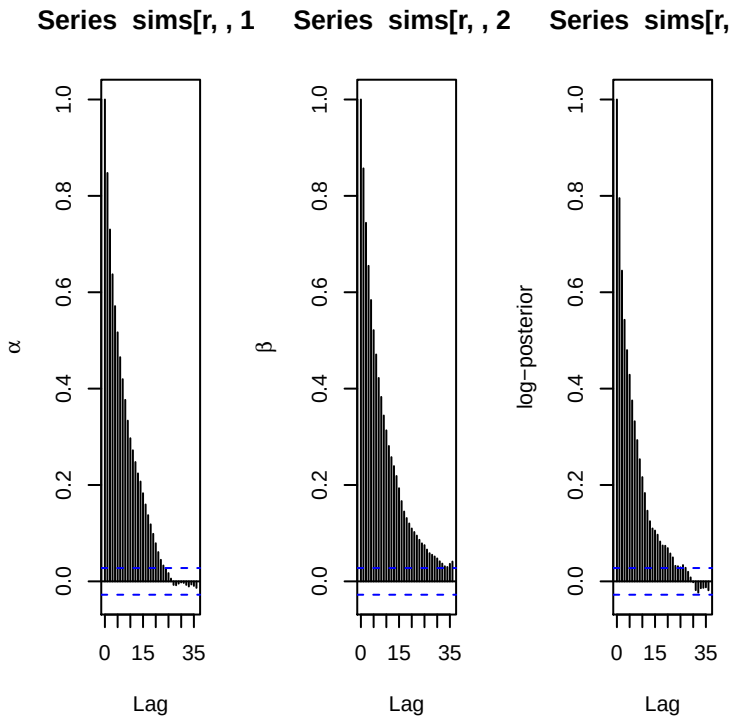
```
#load mcmc_array functionality
mcmc_array <- function (ns, nchains = 1, params) {
  nparams <- length(params)
  array(dim = c(ns, nchains, nparams),
        dimnames = list(iterations = NULL,
                        chains = paste0("chain", 1:nchains),
                        parameters = params))
}
plot_trace <- function (x, ...)
  plot(x, type = "l", col = "gray", xlab = "iteration", ...)
#read in data
bioassay <- read.csv("~/Documents/Work/github/Courses/MA 578/hw5/data/bioassay.csv")

#set up hyperparamters
sigma_alpha <- 5
sigma_beta <- 5
n <- nrow(bioassay)

#define loglikelihood
log_lik <- function(a,b){
  with(bioassay,sum(deaths*(a+b*logdose) - n*log(1 + exp(a + b*logdose))))
}

#set up sampling dataframe + parameters
no.iters <- 10000
params <- c("alpha", "beta", "lp__")
sims <- mcmc_array(no.iters, params = params)

#set initial parameters
alpha <- 1
beta <- 8

#set seed
set.seed(1985)

#start sampler
```

```r
for(t in 1:no.iters){

  #sample propsals
  alpha_star <- rnorm(1, alpha, sigma_alpha)
  beta_star <- rnorm(1, beta, sigma_beta)

  #accept / reject alpha
  log_r_alpha <- log_lik(alpha_star, beta) - log_lik(alpha, beta)
  if (log_r_alpha >= 0 || log(runif(1)) <= log_r_alpha) # accept?
      alpha <- alpha_star

  #accept / reject beta
  log_r_beta <- log_lik(alpha, beta_star) - log_lik(alpha, beta)
  if (log_r_beta >= 0 || log(runif(1)) <= log_r_beta) # accept?
      beta <- beta_star

  #store updates
  lp <- log_lik(alpha, beta)

  #store parameters
  sims[t, 1, ] <- c(alpha, beta, lp)
}

#process burnins
r <- floor(1/2*no.iters):no.iters
#view trace plots
par(mfrow = c(1, 3))
plot_trace(sims[r, , 1], ylab = expression(alpha))
plot_trace(sims[r, , 2], ylab = expression(beta))
plot_trace(sims[r, , 3], ylab = "log-posterior")
```

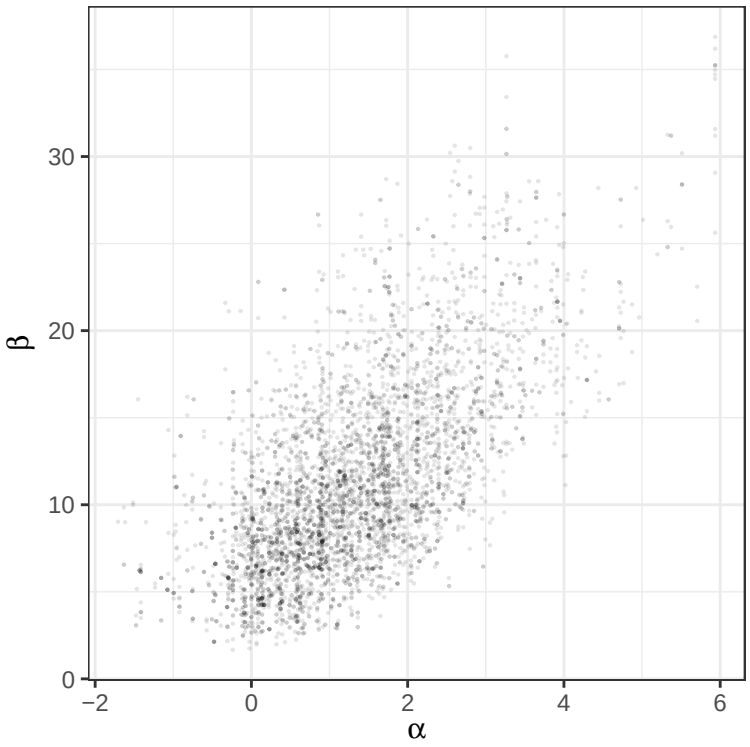



```r
#view acf plots
par(mfrow = c(1, 3))
```

```
acf(sims[r, , 1], ylab = expression(alpha))
acf(sims[r, , 2], ylab = expression(beta))
acf(sims[r, , 3], ylab = "log-posterior")


#plot heatmaps of alpha's and betas
library(ggplot2)
```




```
plotdf <- data.frame(alpha = sims[r, , 1],
                     beta = sims[r, , 2])

ggplot(plotdf, aes(alpha, beta)) +
  geom_point(alpha = .1,
             size = .1)+
  labs(x = expression(alpha),
       y = expression(beta))+
  theme_bw()
```



Exercise 11.3

In this exercise we look to compare and contrast three model approaches - pooled, separate, and hierarchical - models for each quality control measurement. Let y_{ij} be the ith measurement from the jth machine.

Under the separate model, we assume the following model on the data y_{ij}

$$y_{ij}|\theta_j, \sigma^2 \sim N(\theta_j, \sigma^2) \quad i = 1, 2, \ldots, 5$$

and $\mathbb{P}(\theta_j, \sigma^2) \propto 1$. If θ is the vector of each θ_j then this corresponds to the following log - posterior

$$\log p(\theta, \sigma^2|y) \propto \sum_{j=1}^{6} \log p(\theta_j, \sigma^2) + \sum_{j=1}^{6}\sum_{i=1}^{5} \log p(y_{ij}|\theta_j, \sigma^2) \propto \sum_{j=1}^{6}\sum_{i=1}^{5} \log \phi(y_{ij}; \theta_j, \sigma^2)$$

To derive each marginal, $\theta_j|\sigma^2, y$ we can drop any term without θ_j to write

$$p(\theta_j|\theta_{[-j]}, \sigma^2, y) \propto \prod_{i=1}^{5} \phi(y_{ij}; \theta_j, \sigma^2) \propto \exp\left(-\frac{\sum_{i=1}^{5}(y_{ij} - \theta_j)^2}{2\sigma^2}\right) \propto \exp\left(-\frac{2\bar{y}_{\cdot j}\theta_j - \theta_j^2}{2\sigma^2/n_j}\right)$$

Therefore we see that $\theta_j|\sigma^2, y \sim N(\bar{y}_{\cdot j}, \sigma^2/n_j)$. Similarly for the marginal over σ^2 we have

$$p(\sigma^2|\theta, y) \propto \frac{1}{\sigma^{5*6}} \exp\left(-\frac{\sum_{j=1}^{6}\sum_{i=1}^{5}(y_{ij} - \theta_j)^2}{2\sigma^2}\right) \propto (\sigma^2)^{-30/2} \exp\left(-\frac{S(y)}{2\sigma^2}\right)$$

Therefore, $\sigma^2|\theta, y \sim \text{Inv-}\chi^2(30-2, S(y))$ where $S(y) = \sum_{j=1}^{6}\sum_{i=1}^{5}(y_{ij}-\theta_j)^2$. Using both of these marginal distributions we can directly sample from each distribution in a Gibbs sampler setting.

Next, under the pooled model, we assume

$$y_{ij}|\theta, \sigma^2 \sim N(\theta, \sigma^2) \quad i = 1, 2, \ldots, 5 \quad j = 1, 2, \ldots, 6$$

where $\mathbb{P}(\theta, \sigma^2) \propto 1$. This model has corresponding log-likelihood

$$\log(\theta|y, \sigma^2) \propto \log p(\theta, \sigma^2) + \sum_{j=1}^{6}\sum_{i=1}^{5}\log p(y_{ij}|\theta, \sigma^2)$$

Finding the marginal distribution for $\theta|y, \sigma^2$ we have

$$\begin{aligned} p(\theta|\sigma^2, y) &\propto \prod_{j=1}^{6}\prod_{i=1}^{5}\phi(y_{ij}|\theta, \sigma^2) \propto \exp\left(-\frac{\sum_{j=1}^{6}\sum_{i=1}^{5}(y_{ij}-\theta)^2}{2\sigma^2}\right) \\ &\propto \exp\left(-\frac{2\bar{y}\theta - \theta^2}{2\sigma^2/n}\right) \end{aligned}$$

So $\theta|y, \sigma^2 \sim N(\bar{y}, \sigma^2/n)$. Moreover, we can find the marginal for $\sigma^2|y, \theta$ as follows

$$p(\sigma^2|\theta, y) \propto \frac{1}{\sigma^{30}}\exp\left(-\frac{S(y)}{2\sigma^2}\right) \propto (\sigma^2)^{-30/2}\exp\left(-\frac{S(y)}{2\sigma^2}\right)$$

which we conclude that $\sigma^2|y, \theta \sim \text{Inv-}\chi^2(30-2, S(y))$ where $S(y) = \sum_{j=1}^{6}\sum_{i=1}^{5}(y_{ij}-\theta_j)^2$.

Finally, under the hierarchical model, we assume the two tier model

$$\begin{aligned} y_{ij}|\theta_j &\sim N(\theta_j, \sigma^2) \\ \theta_j|\mu, \tau &\sim N(\mu, \tau^2) \\ \mathbb{P}(\mu, \log\sigma, \log\tau) &\propto \tau \end{aligned}$$

Notice we take the prior $\mathbb{P}(\mu, \log\sigma, \log\tau) \propto \tau$ as to achieve a proper posterior. Under this model the full log-posterior can be written as

$$\log p(\theta, \mu, \log\sigma, \log\tau|y) \propto \log(\tau) + \sum_{j=1}^{6}\log\phi(\theta_j; \mu, \tau^2) + \sum_{j=1}^{6}\sum_{i=1}^{5}\log\phi(y_{ij}; \theta_j, \sigma^2)$$

From this log posterior, we can readily find each marginal distribution for $\theta_j, \mu, \tau^2, \sigma^2$ The full marginals for this Gibbs sampler are derived in the page 289 of the text. Due to the simple conjugate priors chosen from this model each Gibbs update can be sampled directly from R without use of any custom Metropolis-Hastings steps.

With each of these marginal distributions derived, we can now implement the Gibbs sampler for each method below.

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
#import data
qc_dat <- data.frame(machine = rep(1:6, each = 5),
                     measurement = c(83,92,92,46,67,
                                     117,109,114,104,87,
                                     101,93,92,86,67,
                                     105,119,116,102,116,
                                     79,97,103,79,92,
                                     57,92,104,77,100))
#import r inv sampler
rinvsquare <- function (ns, nu, nu_tau2) 1 / rgamma(ns, nu / 2, nu_tau2 / 2)

#------------------------------
#
#   Separate Model
#
#------------------------------

no.iters <- 5000
params <- c(paste0("theta", 1:6), "sigma2", "lp__")
sims_sep <- mcmc_array(no.iters, params = params)

#set up sufficient statistics
nj <- with(qc_dat, tapply(measurement, machine, length))
yj <- with(qc_dat, tapply(measurement, machine, mean))
ybar <- with(qc_dat, mean(measurement))
s2j <- with(qc_dat, tapply(measurement, machine, var))
S <- sum(s2j * (nj - 1))
J <- length(nj)
n <- sum(nj)

#set initial parameters
theta <- as.numeric(select(qc_dat %>% group_by(machine) %>% summarize(theta = mean(measurement)), theta)
mu <- mean(theta)
sigma2 <- tau2 <- 1

# iterate
for (t in 1:no.iters) {

  # sample sigma2
  sigma2 <- rinvsquare(1, 30-2, S + sum(nj * (yj - theta) ^ 2))

  for(j in 1:J){
```

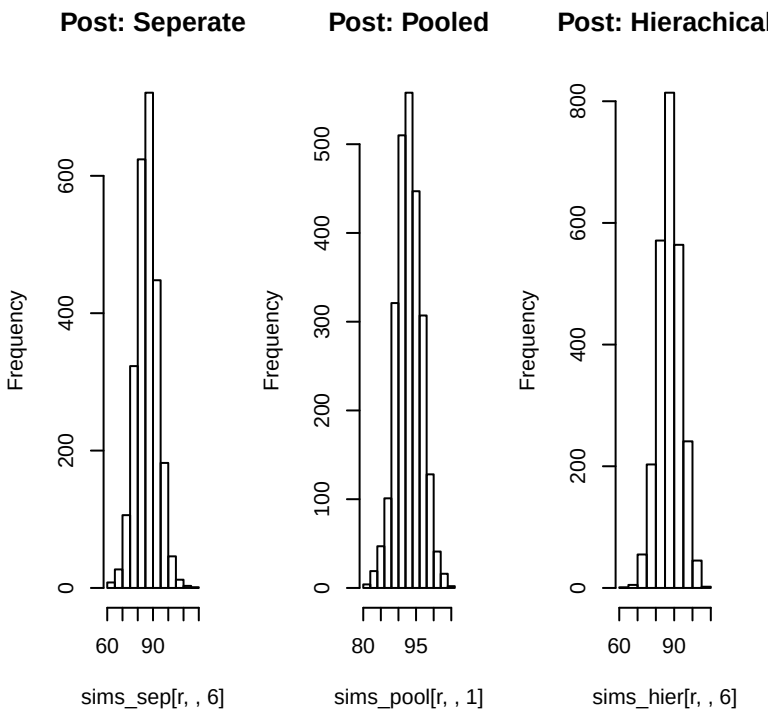
```
    #sample thetas
    theta[j] <- rnorm(1, yj[j], sqrt(sigma2 / nj[j]) )
  }

  # update log posterior
  lp <- with(qc_dat,
             sum(dnorm(measurement, rep(theta, each = J), sd = sqrt(sigma2), log = TRUE)))

  #store values
  sims_sep[t, 1, ] <- c(theta, sigma2, lp)
}


#------------------------------
#
#   Pooled Model
#
#------------------------------

no.iters <- 5000
params <- c("theta", "sigma2", "lp__")
sims_pool <- mcmc_array(no.iters, params = params)

#set up sufficient statistics
nj <- with(qc_dat, tapply(measurement, machine, length))
yj <- with(qc_dat, tapply(measurement, machine, mean))
ybar <- with(qc_dat, mean(measurement))
s2j <- with(qc_dat, tapply(measurement, machine, var))
S <- sum(s2j * (nj - 1))
J <- length(nj)
n <- sum(nj)

#set initial parameters
theta <- ybar
sigma2 <- 1

# iterate
for (t in 1:no.iters) {

  # sample sigma2
  sigma2 <- rinvsquare(1, 30-2, with(qc_dat, sum((measurement - theta)^2)))

  #sample theta
  theta <- rnorm(1, ybar, sqrt(sigma2 / 30))

  # update log posterior
  lp <- with(qc_dat,
             sum(dnorm(measurement, rep(theta, 30), sd = sqrt(sigma2), log = TRUE)))

  #store values
  sims_pool[t, 1, ] <- c(theta, sigma2, lp)
}
```

```
#------------------------------
#
#   Hierachical Model
#
#------------------------------
library(dplyr)

#set up sampling dataframe + parameters
no.iters <- 5000
params <- c(paste0("theta", 1:6), "mu", "sigma2", "tau2", "lp__")
sims_hier <- mcmc_array(no.iters, params = params)

#set up sufficient statistics
nj <- with(qc_dat, tapply(measurement, machine, length))
yj <- with(qc_dat, tapply(measurement, machine, mean))
s2j <- with(qc_dat, tapply(measurement, machine, var))
S <- sum(s2j * (nj - 1))
J <- length(nj)
n <- sum(nj)

#set initial parameters
theta <- as.numeric(select(qc_dat %>% group_by(machine) %>% summarize(theta = mean(measurement)), theta)
mu <- mean(theta)
sigma2 <- tau2 <- 1

# iterate
for (t in 1:no.iters) {
  # sample sigma2
  sigma2 <- rinvsquare(1, n, S + sum(nj * (yj - theta) ^ 2))
  # sample tau2
  tau2 <- rinvsquare(1, J - 1, sum((theta - mu) ^ 2))
  # sample thetas
  theta <- rnorm(J, (nj * yj / sigma2 + mu / tau2) / (nj / sigma2 + 1 / tau2),
                 1 / sqrt(nj / sigma2 + 1 / tau2))
  # sample mu
  mu <- rnorm(1, mean(theta), sqrt(tau2 / J))
  # update log posterior
  lp <- with(qc_dat, sum(dnorm(measurement, theta[machine], sqrt(sigma2), log = TRUE))) +
    sum(dnorm(theta, mu, sqrt(tau2), log = TRUE)) - log(sigma2) - .5 * log(tau2)

  sims_hier[t, 1, ] <- c(theta, mu, sigma2, tau2, lp)
}
```

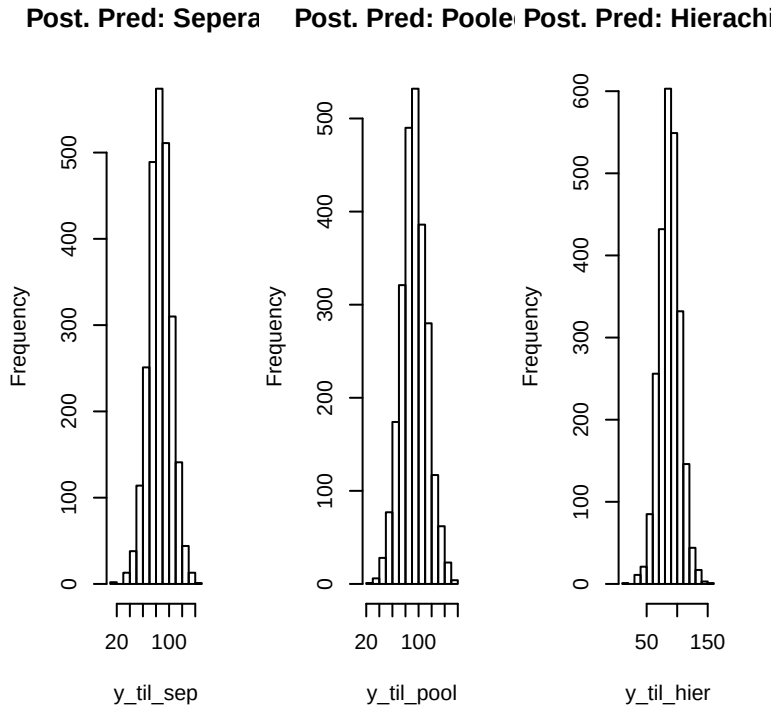
Now that we have constructed samples from each model we can proceed with posterior inference. We plot the posterior distribution for θ_6 under each model. Moreover we generate posterior predictive for each model using the mcmc samples from above. Finally, for mean of a 7-th machine, the separate model will not be able to account for this new machine. The pooled and the hierarchical models readily lend themselves to inference on the final machine. From the pooled, we simply take the posterior mcmc samples from the simulation above. For the hierarchical model, we sample from $N(\mu, \tau^2)$ where μ, τ^2 are samples drawn from the mcmc process above.

```
#process burnins
r <- floor(1/2*no.iters):no.iters
```

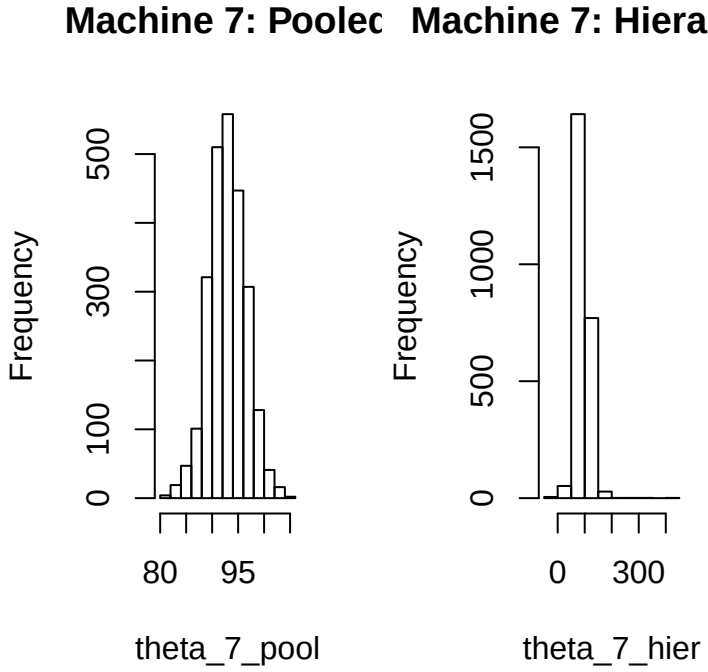
```r
#plot 6th machine's histogram
par(mfrow = c(1,3))
hist(sims_sep[r, , 6], main = "Post: Seperate")
hist(sims_pool[r, , 1], main = "Post: Pooled")
hist(sims_hier[r, , 6], main = "Post: Hierachical")
```




```r
#get predictive probabilities
y_til_sep <- rnorm(length(r), sims_sep[r, , 6], sqrt(sims_sep[r, , 7]))
y_til_pool <- rnorm(length(r), sims_pool[r, , 1], sqrt(sims_pool[r, , 2]))
y_til_hier <- rnorm(length(r), sims_hier[r, , 6], sqrt(sims_hier[r, , 8]))

#plot predict
par(mfrow = c(1,3))
hist(y_til_sep, main = "Post. Pred: Seperate")
hist(y_til_pool, main = "Post. Pred: Pooled")
hist(y_til_hier, main = "Post. Pred: Hierachical")
```




```
#get theta_7 posterior draws from pooled
theta_7_pool <- sims_pool[r,,1]
theta_7_hier <- rnorm(length(r), sims_hier[r, , 7], sqrt(sims_hier[r, , 9]))

#plot theta_7
par(mfrow = c(1,2))
hist(theta_7_pool, main = "Machine 7: Pooled")
hist(theta_7_hier, main = "Machine 7: Hierachical")
```



Exercise 3

Part (a)

Given that $w_i|\mu,\sigma^2 \overset{iid}{\sim} N(\mu,\sigma^2)$ and $\mathbb{P}(\mu,\sigma^2,\lambda) \propto \frac{1}{\sigma}$ we can readily derive the posterior distribution.

$$\mathbb{P}(\lambda,\mu,\sigma^2|y) \propto p(\lambda,\sigma^2,\mu)p(y|\lambda,\sigma^2,\mu) \propto \frac{1}{\sigma}\prod_{i=1}^{n} p(y_i|\lambda,\sigma^2,\mu)$$

Now notice that as w_i and y_i are related through a one to one invertible function, we can use a transformation of variables to determine the distribution of $p(y_i|\lambda,\sigma^2,\mu)$. Let $w_i(\lambda)$ the Box-Cox transformation. Then taking a derivative with respect to y_i, we achieve $y_i^{\lambda-1}$ when $\lambda > 0$ and $\frac{1}{y_i}$ when $\lambda = 0$. As λ is considered a continuous parameter, this later event occurs with probability 0 as we can write our transformed posterior as follows.

$$\mathbb{P}(\lambda,\mu,\sigma^2|y) \propto \frac{1}{\sigma}\prod_{i=1}^{n} N(w_i(\lambda)|\lambda,\sigma^2,\mu)|y_i^{\lambda-1}|$$

Part (b)

First we look to construct the marginals for μ and σ through the form of the posterior derived in part (a) first consider μ.

$$\begin{aligned} p(\mu|\sigma,\lambda,y) &\propto \prod_{i=1}^{n} N(w_i(\lambda)|\lambda,\sigma^2,\mu) \propto \exp\left(-\frac{\sum_{i=1}^{n}(w_i-\mu)^2}{2\sigma^2}\right) \\ &\propto \exp\left(-\frac{2\bar{w}\mu-\mu^2}{2\sigma^2/n}\right) \end{aligned}$$

Therefore we see that $\mu|\sigma,\lambda,y \sim N(\bar{w},\sigma^2/n)$. We can use this result in our update in the Gibbs algorithm implemented below. Following suite with σ^2 we have

$$\begin{aligned} p(\sigma^2|\mu,\lambda,y) &\propto \frac{1}{\sigma}\prod_{i=1}^{5} N(w_i(\lambda)|\lambda,\sigma^2,\mu) \propto \frac{1}{\sigma}\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(w_i-\mu)^2}{2\sigma^2}\right) \\ &\propto \frac{1}{\sigma^{n+1}}\exp\left(-\frac{S(w)}{2\sigma^2}\right) \propto (\sigma^2)^{-\frac{n+1}{2}}\exp\left(-\frac{S(w)}{2\sigma^2}\right) \end{aligned}$$

From here we recognize this distribution as a Inv-$\chi^2(n-1,\frac{1}{n-1}S(w))$ where $S(w)=\sum_{i=1}^{n}(w_i-mu)^2$, Finally, to sample from λ, we use a Metropolis-within-Gibbs algorithm to sample from λ With these observations, we are now ready to implement our sampler.

```
#load up data
y <- c(13, 52, 6, 40, 10, 7, 66, 10, 10, 14, 16, 4,
65, 5, 11, 10, 15, 5, 76, 56, 88, 24, 51, 4,
40, 8, 18, 5, 16, 50, 40, 1, 36, 5, 10, 91,
18, 1, 18, 6, 1, 23, 15, 18, 12, 12, 17, 3)
```

```r
#set up hyperparamters
sigma_lam <- .01

#define boxcox function
w <- function(y, lam){
  if(lam == 0){
    return(log(y))
  }
  else{
    return((y^(lam) -1) / lam)
  }
}

#define sufficient statistics
Sy <- sum(y)
n <- length(y)


#define loglikelihood
log_lik <- function(mu,lam,sigma2){
  -log(sqrt(sigma2)) + sum(dnorm(sapply(y, function(i) w(i, lam)), mu, sqrt(sigma2), log = TRUE)) + (lam
}


box_cox_sampler <- function(no.iters =  10000){

  #set up sampling dataframe + parameters
  params <- c("mu", "sigma2","lam", "lp__")
  sims <- mcmc_array(no.iters, params = params)

  #set initial parameters
  mu <- lam <- 1
  sigma2 <- 1

  #start sampler
  for(t in 1:no.iters){

    #cache w's
    w_vec <- sapply(y, function(i) w(i, lam))

    #sample mu
    mu <- rnorm(1, mean(w_vec), sigma2/n)

    #sample sigma2
    sigma2 <- rinvsquare(1, n - 1, sum((w_vec - mu) ^ 2))

    #propose lambda
    lam_star <- rnorm(1, lam, sigma_lam)

    #accept / reject
    if(lam_star>0){
      log_r_lam <- log_lik(mu, lam_star, sqrt(sigma2)) - log_lik(mu, lam, sqrt(sigma2))
      if (log_r_lam >= 0 || log(runif(1)) <= log_r_lam) # accept?
```

```
          lam <- lam_star
    }



    #store updates
    lp <- log_lik(mu, lam, sqrt(sigma2))

    #store parameters
    sims[t, 1, ] <- c(mu, sigma2, lam, lp)
  }

  #return samples
  return(sims)
}
```

Part (c)

In this section we create diagnostic checks for MCMC sampler. We run two chains in parallel and use the Bayesplot and rstan package for visualize diagnostic checks. We use a 5000 step burn in for chains of length 10000.

```
#load up visualization libraries
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
## rstan (Version 2.19.2, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
library(bayesplot)
```

```
## This is bayesplot version 1.7.0
```

```
## - Online documentation and vignettes at mc-stan.org/bayesplot
```

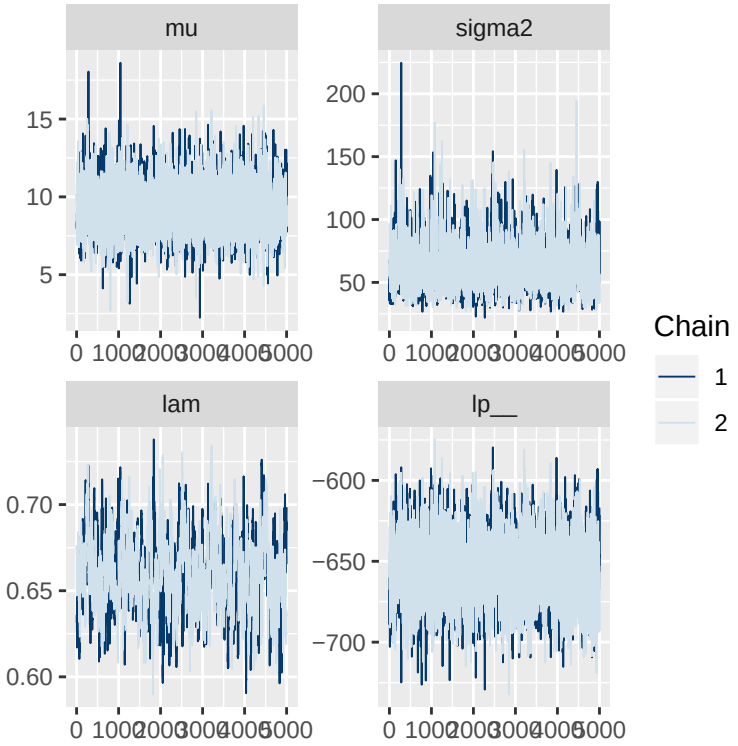
```
## - bayesplot theme set to bayesplot::theme_default()
```

```
##    * Does _not_ affect other ggplot2 plots
```

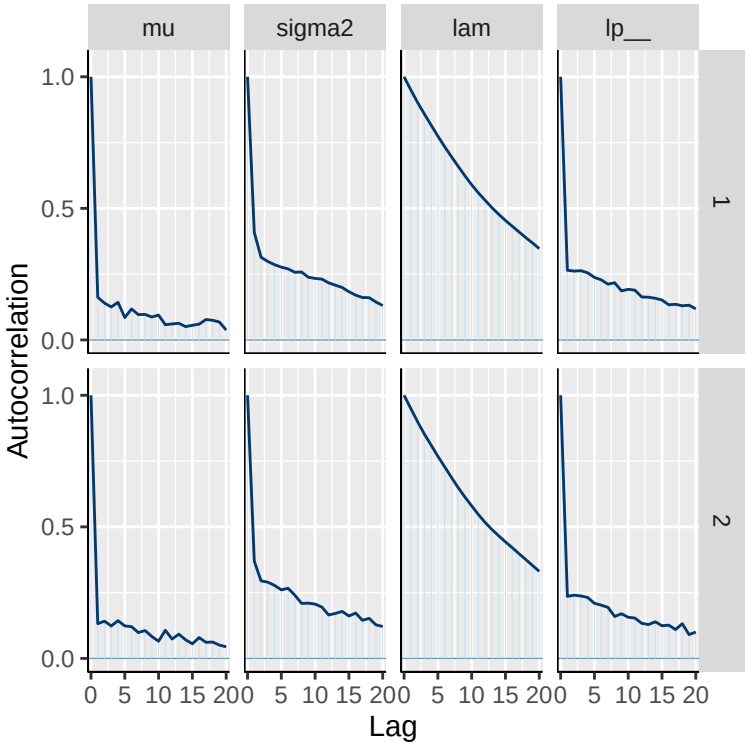
```
##    * See ?bayesplot_theme_set for details on theme setting
```

```
#define parameters
 params <- c("mu", "sigma2","lam", "lp__")

#process burnins
no.iters <- 10000
r <- floor(.5*no.iters):no.iters

#set up bayes plot storage
set.seed(1985)
sims <- mcmc_array(no.iters*.5 +1, 2, params)
sims[,1,] <- box_cox_sampler(no.iters)[r,,]
sims[,2,] <- box_cox_sampler(no.iters)[r,,]
```

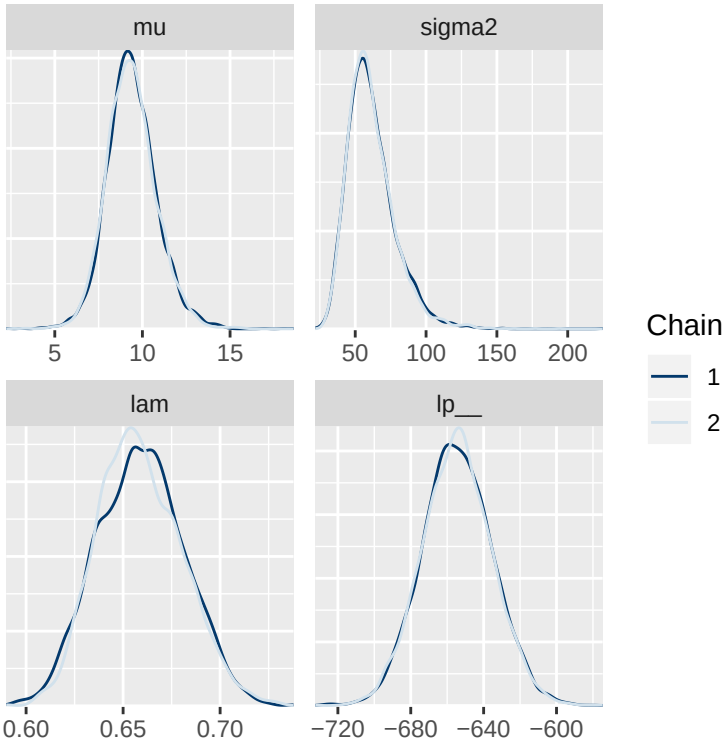
```
#make diagnostic plots
mcmc_trace(sims, params)
```




```
mcmc_acf(sims, params)
```

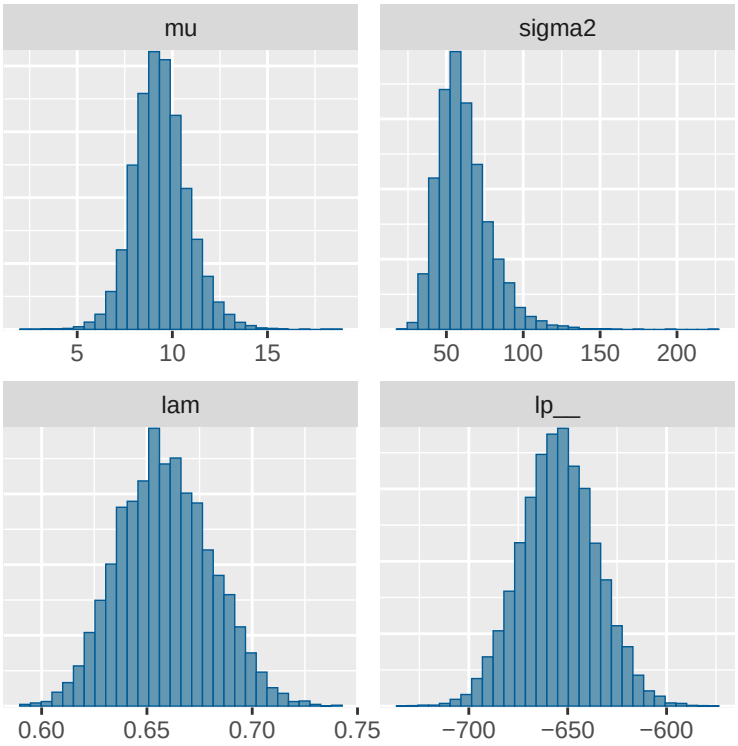


```
mcmc_dens_overlay(sims, params)
```




```
mcmc_hist(sims, params)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



From these plots, we see that the chains are mixing relatively well with the exception of λ. While μ and σ^2

appear to be exploring the space well and do not exhibit large autocorrelative behavior, the λ chain appears to be correlated with itself. Further adjusting the proposal used here could exhibit better results.

Next, we construct posterior intervals based on these samples.

```
#first merge chains
mu <- c(sims[,1,1], sims[,2,1])
sigma2 <- c(sims[,1,2], sims[,2,2])
lam <- c(sims[,1,3], sims[,2,3])

#make posterior intervals
quantile(mu, probs =c(.025, .975))
```

```
##      2.5%     97.5%
##  6.760226 12.379778
```

```
quantile(sigma2, probs =c(.025, .975))
```

```
##     2.5%    97.5%
##  36.2252 100.6608
```

```
quantile(lam, probs =c(.025, .975))
```

```
##      2.5%     97.5%
## 0.6185275 0.7029330
```

From these quantiles, we see that the posterior interval for lambda is (.62,.70) suggesting that a square root transform ($\lambda = 1/2$) would be more appropriate than a log ($\lambda = 0$) transformation.

Part (d)

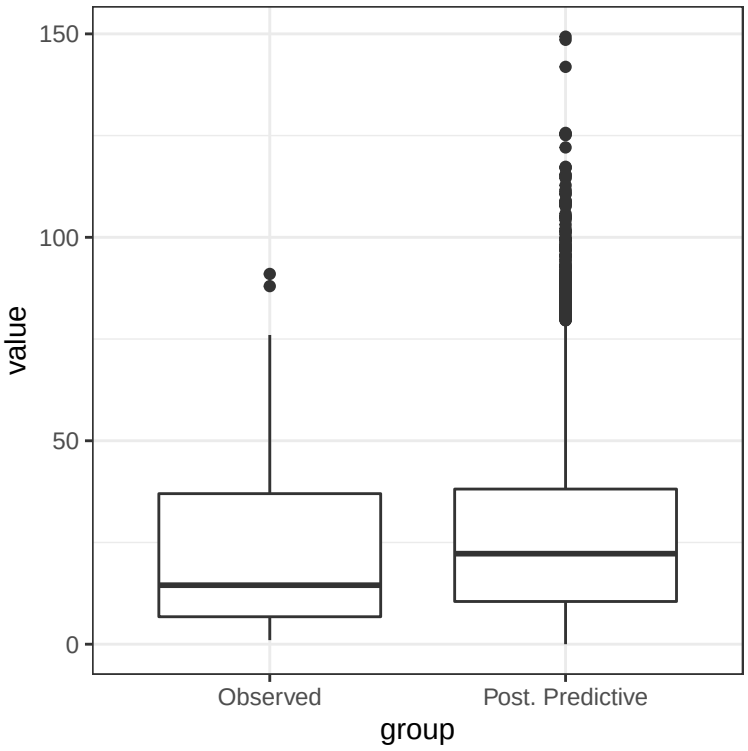
Suppose that a new patient enters the study with unknown survival $\tilde{y}$. Then using our posterior samples from part(c) we can (i) sample $w_i|\mu^{(t)}, (\sigma^2)^{(t)}$ and then calculate $\tilde{y}^{(t)} = w_i(\lambda^{(t)})$, the inverse Box Cox transformation given a certain λ value. We implement sampling below.

```
#define inverse box cox
boxcox.inv <- function(w, lam) (lam*w +1)^(1/lam)

#sample w tilde
n <- length(mu)
w_tilde <- rnorm(n, mu, sqrt(sigma2))

#sample ytilde
y_tilde <- sapply(1:n, function(i) boxcox.inv(w_tilde[i], lam[i]))
```

From here we can plot this predictive probability against the observed as a posterior check.

```
#make boxplot
plotdf <- data.frame(group = c(rep("Observed", length(y)), rep("Post. Predictive", n)),
                     value = c(y, y_tilde))

ggplot(plotdf, aes(x = group, y = value))+
  geom_boxplot()+
  theme_bw()
```

```
## Warning: Removed 811 rows containing non-finite values (stat_boxplot).
```



From the above figure it appears that the posterior predictive might be more slightly more skewed than the observed data. However, in general, there is not a large discrepancy between these two plots suggesting that the model fit the data well.