MA 578 HW5 Solutions

We will be using the following packages and definitions:

```
library(rstan)
library(bayesplot)
library(beanplot)
library(tidyverse)

# sample from inverse (scaled) chi-square with parameters `nu` and `tau2`;
# nu_tau2 = nu * tau2 for convenience
rinvchisq <- function (ns, nu, nu_tau2) 1 / rgamma(ns, nu / 2, nu_tau2 / 2)

plot_hist <- function (x, ...)
  hist(x, col = "gray", border = "white", main = "", ...)

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))
}

iter <- 2000
warmup <- floor(iter / 2)
nsamples <- iter - warmup
nchains <- 4
```

1 (BDA 11.2)

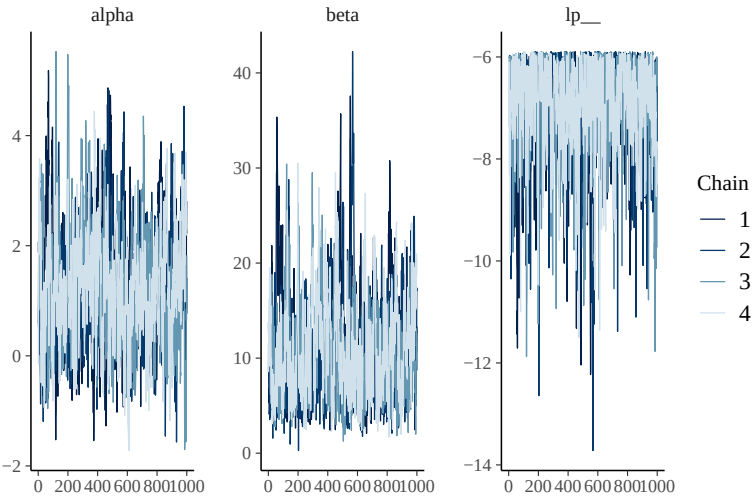
Using random walk Metropolis-Hastings with non-informative prior $\mathbb{P}(\alpha, \beta) \propto 1$.

```
bioassay <- read.csv("data/bioassay.csv", comment = "#")
n <- bioassay$n; y <- bioassay$deaths; x <- bioassay$logdose
log_p <- function (alpha, beta) { # log posterior
  p <- plogis(alpha + beta * x)
  sum(y * log(p) + (n - y) * log(1 - p))
}

sigma_alpha <- .25
sigma_beta <- 5 * sigma_alpha
nthin <- 10
sims <- mcmc_array(nsamples, nchains, c("alpha", "beta", "lp__"))

gb <- glm(cbind(y, n - y) ~ x, family = binomial)
alpha <- coef(gb)[1]; beta <- coef(gb)[2]
for (chain in 1:nchains) {
  for (it in 1:iter) {
    for (thin in 1:nthin) {
      alpha_c <- rnorm(1, alpha, sigma_alpha)
      log_r <- log_p(alpha_c, beta) - log_p(alpha, beta)
      if (log_r >= 0 || log(runif(1)) <= log_r) alpha <- alpha_c # accept?
```

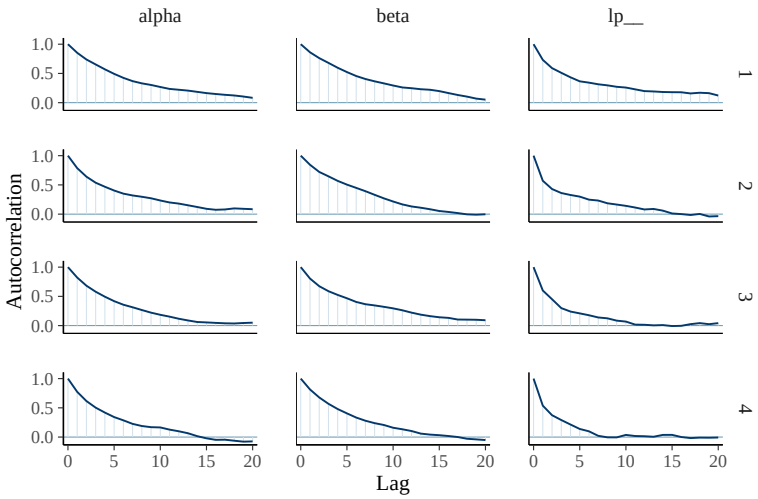
```
      beta_c <- rnorm(1, beta, sigma_beta)
      log_r <- log_p(alpha, beta_c) - log_p(alpha, beta)
      if (log_r >= 0 || log(runif(1)) <= log_r) beta <- beta_c # accept?
    }

    if (it > warmup)
      sims[it - warmup, chain, ] <- c(alpha, beta, log_p(alpha, beta))
  }
}

monitor(sims, warmup = 0)
```

```
Inference for the input samples (4 chains: each with iter = 1000; warmup = 0):

        Q5  Q50  Q95 Mean  SD  Rhat Bulk_ESS Tail_ESS
alpha -0.4  1.2  3.1  1.2 1.1  1.01      349      484
beta   4.2 10.1 21.4 11.1 5.4  1.01      350      473
lp__  -8.9 -6.6 -5.9 -6.9 1.0  1.00      604      582

For each parameter, Bulk_ESS and Tail_ESS are crude measures of
effective sample size for bulk and tail quantities respectively (an ESS > 100
per chain is considered good), and Rhat is the potential scale reduction
factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

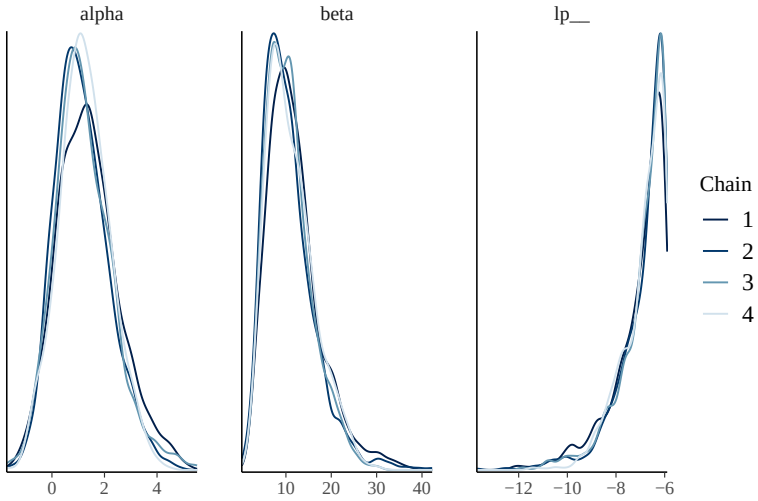
```
mcmc_trace(sims)
```



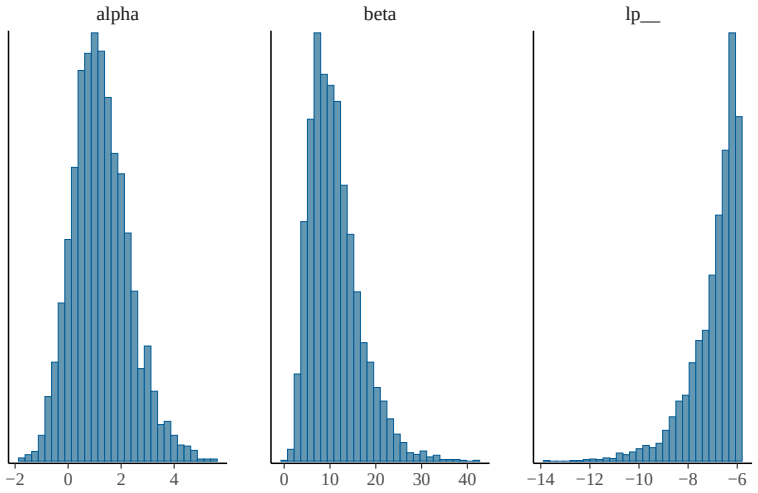

```
mcmc_acf(sims)
```




```
mcmc_dens_overlay(sims)
```




```
mcmc_hist(sims)
```



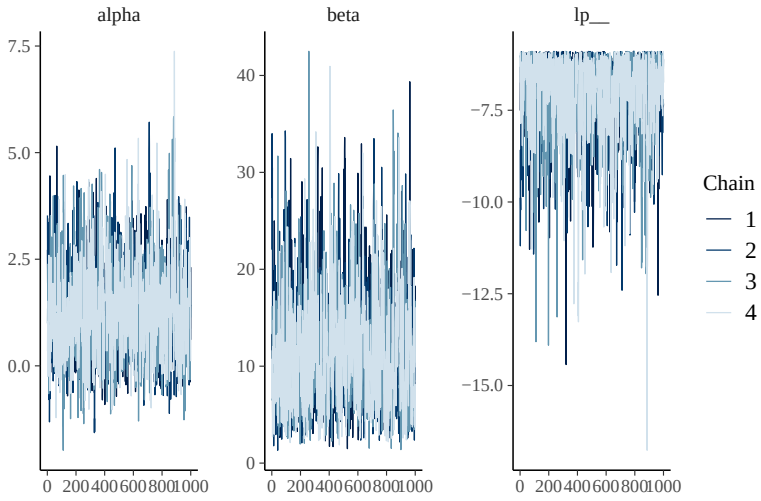
For comparison, here's the equivalent Stan output:

```
N <- length(y)
sm <- stan_model(model_code = "data { int<lower=0> N; int<lower=0> n[N];
  int<lower=0> y[N]; vector[N] x; }
parameters { real alpha; real beta; }
model { y ~ binomial_logit(n, alpha + beta * x); }")
sf <- sampling(sm, data = c("N", "n", "y", "x"))
```

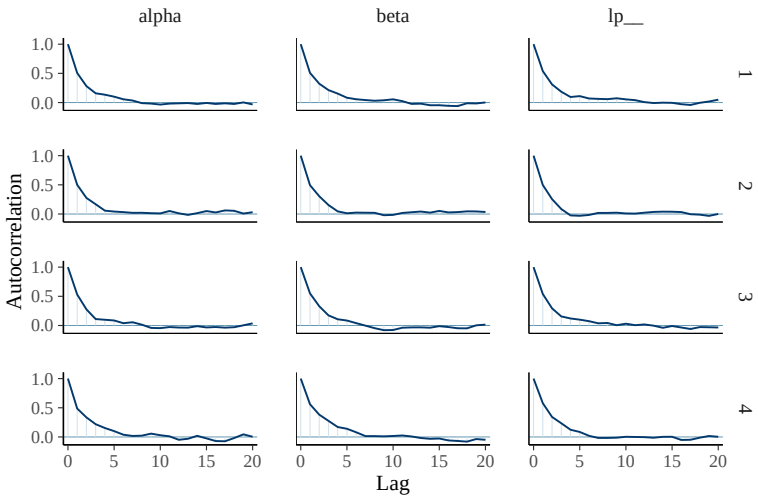
```
monitor(sf)
```

```
Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):

        Q5  Q50  Q95 Mean  SD  Rhat Bulk_ESS Tail_ESS
alpha -0.3  1.2  3.2  1.3 1.1     1     1191     1496
beta   4.2 10.8 22.3 11.6 5.7     1     1181     1281
lp__  -9.2 -6.7 -5.9 -7.0 1.1     1     1281     1501

For each parameter, Bulk_ESS and Tail_ESS are crude measures of
effective sample size for bulk and tail quantities respectively (an ESS > 100
per chain is considered good), and Rhat is the potential scale reduction
factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

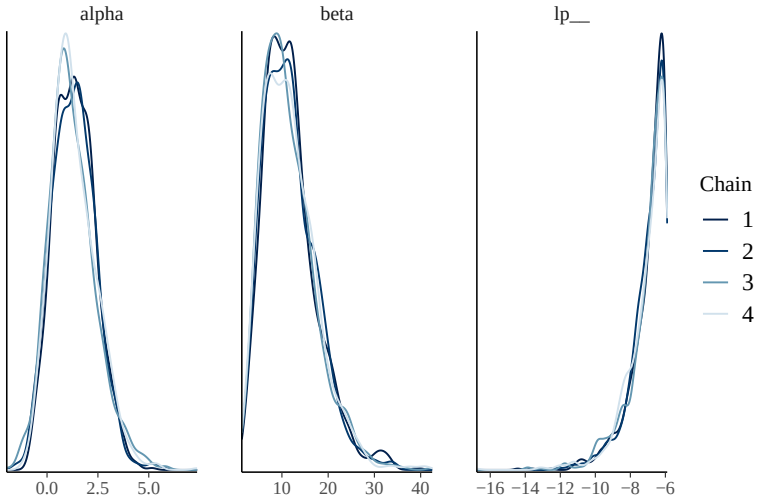
```
mcmc_trace(sf)
```



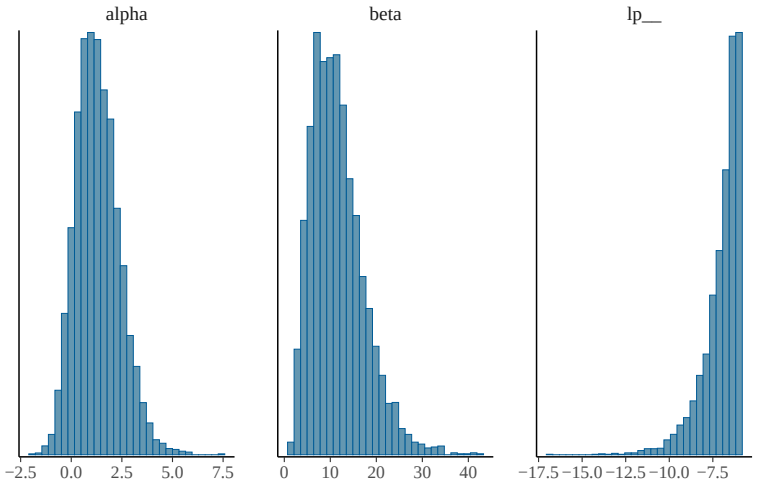

```
mcmc_acf(sf)
```




```
mcmc_dens_overlay(sf)
```




```
mcmc_hist(sf)
```



2 (BDA 11.3)

We have three analyses—separate, pooled, and hierarchical (partially pooled)—of a set of quality control measurements. Here y_{ij} is the i-th measurement, out of n_j measurements, in the j-th machine from a total of $J = 6$ machines.

```
machine <- read.csv("data/machine.csv", comment = "#")
machine %>% group_by(machine) %>% summarize(qc = paste(qc, collapse = ", "))
```

```
# A tibble: 6 x 2
  machine qc
    <int> <chr>
1       1 83, 92, 92, 46, 67
2       2 117, 109, 114, 104, 87
3       3 101, 93, 92, 86, 67
4       4 105, 119, 116, 102, 116
5       5 79, 97, 103, 79, 92
6       6 57, 92, 104, 77, 100
```

```
# sufficient statistics:
nj <- with(machine, tapply(qc, machine, length))
yj <- with(machine, tapply(qc, machine, mean))
s2j <- with(machine, tapply(qc, machine, var))
S <- sum(s2j * (nj - 1))
J <- length(yj)
n <- sum(nj)
```

For each analysis, we want (i) $\theta_6 \,|\, y$, (ii) $\widetilde{y}_6 \,|\, y$, and (iii) $\theta_7 \,|\, y$. These posterior distributions should be estimated using Gibbs sampling, as requested.

Separate

The model we assume is $y_{ij} \,|\, \theta_j, \sigma^2 \overset{\text{ind}}{\sim} N(\theta_j, \sigma^2)$ for $i = 1, \ldots, n_j$, $j = 1, \ldots, J$, and non-informative priors $\mathbb{P}(\theta, \log \sigma) \propto 1$. For the Gibbs sampler, since $\bar{y}_j \,|\, \theta_j, \sigma^2 \overset{\text{ind}}{\sim} N(\theta_j, \sigma^2/n_j)$, we can first see that $\theta_j \,|\, \sigma^2, y \overset{\text{ind}}{\sim} N(\bar{y}_j, \sigma^2/n_j)$. The joint posterior is

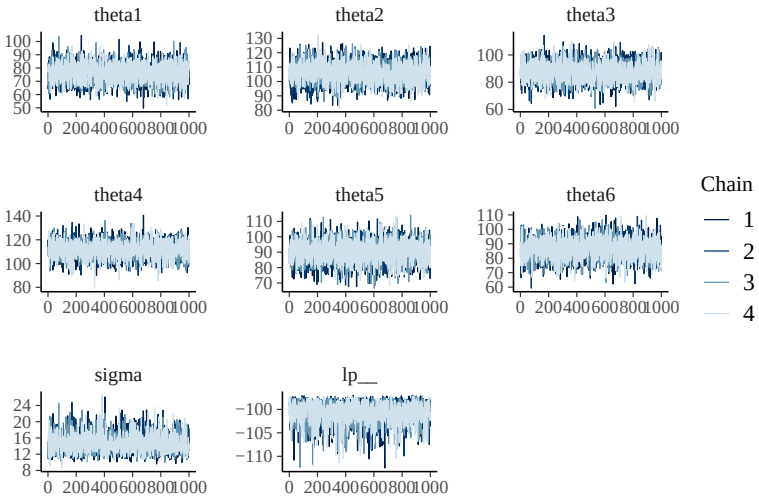
$$\mathbb{P}(\theta, \sigma^2 \,|\, y) \propto (\sigma^2)^{-\sum_j n_j/2} \exp\left\{ -\frac{\sum_{i,j}(y_{ij} - \theta_j)^2}{2\sigma^2} \right\} (\sigma^2)^{-1}$$

and so $\sigma^2 \,|\, \theta, y \sim \text{Inv-}\chi^2(\sum_j n_j, \sum_{i,j}(y_{ij} - \theta_j)^2 / \sum_j n_j)$.

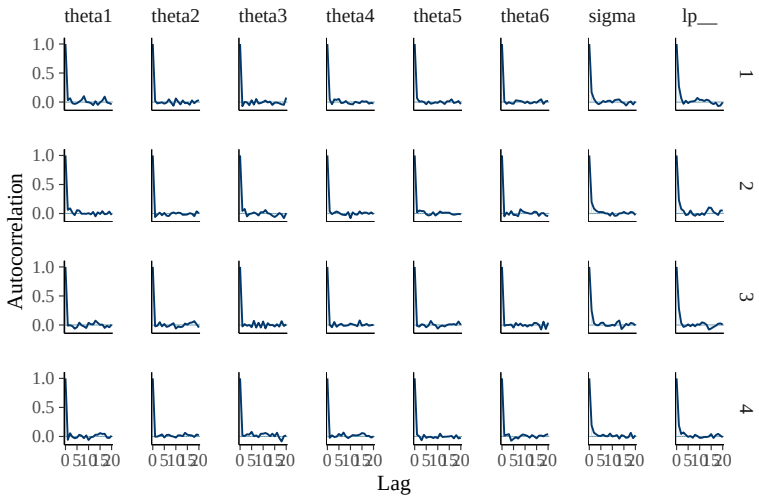
```
sims <- mcmc_array(nsamples, nchains, c(paste0("theta", 1:J), "sigma", "lp__"))

sigma2 <- var(machine$qc)
for (chain in 1:nchains) {
  for (it in 1:iter) {
    theta <- rnorm(J, yj, sqrt(sigma2 / nj))
    nt <- S + sum(nj * (yj - theta) ^ 2)
    sigma2 <- rinvchisq(1, n, nt)
    target <- -(n / 2 + 1) * log(sigma2) - nt / (2 * sigma2)
    if (it > warmup)
      sims[it - warmup, chain, ] <- c(theta, sqrt(sigma2), target)
  }
}

monitor(sims, warmup = 0)
```

```
Inference for the input samples (4 chains: each with iter = 1000; warmup = 0):

           Q5    Q50   Q95   Mean  SD  Rhat Bulk_ESS Tail_ESS
theta1   65.0   76.1  87.0   76.1 6.7     1     3628     3658
theta2   95.3  106.2 116.9  106.2 6.6     1     4146     3727
theta3   77.1   87.8  98.5   87.9 6.6     1     3912     3674
theta4  100.8  111.6 122.1  111.5 6.6     1     3712     3560
theta5   79.2   89.8 100.8   90.0 6.6     1     3677     3631
theta6   75.2   85.9  97.0   86.1 6.7     1     4023     3924
sigma    11.6   14.5  18.8   14.8 2.2     1     2500     2995
lp__   -104.9 -100.4 -98.0 -100.8 2.2     1     2527     3086

For each parameter, Bulk_ESS and Tail_ESS are crude measures of
effective sample size for bulk and tail quantities respectively (an ESS > 100
per chain is considered good), and Rhat is the potential scale reduction
factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

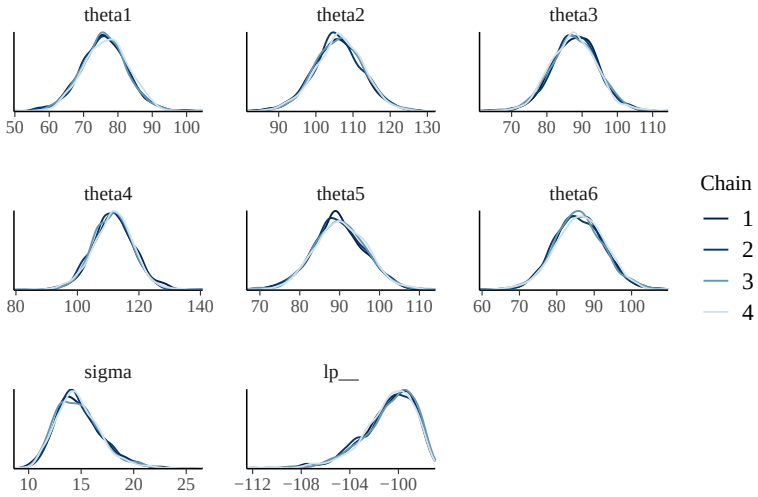
```
mcmc_trace(sims)
```




```
mcmc_acf(sims)
```

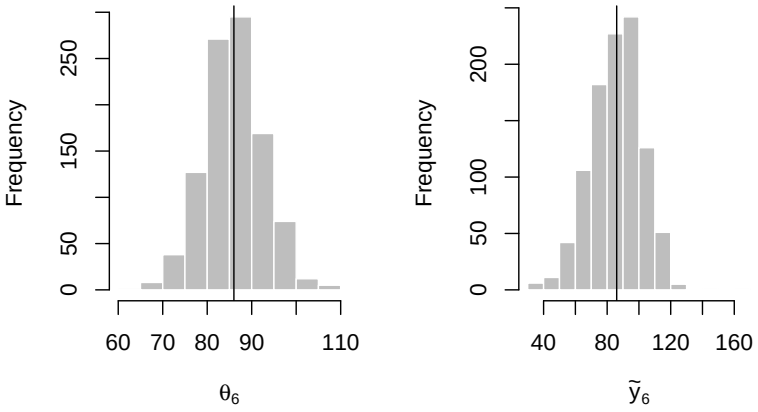


```
mcmc_dens_overlay(sims)
```



Now we can provide what has been asked. Since we are assuming a separate model, the posterior distribution for θ_7 is the same as its prior, flat, due to no borrowing of information from other machines. Not surprisingly, the posteriors are in good agreement with the data.

```
# we can just use the first chain
theta6 <- sims[, 1, 6]; sigma <- sims[, 1, 7]
ypred6 <- rnorm(nsamples, theta6, sigma)
op <- par(mfrow = c(1, 2))
plot_hist(theta6, xlab = expression(theta[6])); abline(v = yj[6])
plot_hist(ypred6, xlab = expression(tilde(y)[6])); abline(v = yj[6])
```




```
par(op)
```

For comparison, here's the Stan equivalent model; the results are not shown to save space, but they are equivalent to Gibbs sampling.

```
y <- machine$qc; idx <- c(0, cumsum(nj)[-J])
sm <- stan_model(model_code = "data {
  int<lower=0> J; int<lower=0> nj[J]; int<lower=0> idx[J]; int<lower=0> n; real y[n];
}
parameters { vector[J] theta; real<lower=0> sigma; }
model {
  for (j in 1:J) for (i in 1:nj[j]) y[idx[j] + i] ~ normal(theta[j], sigma);
  target += -log(sigma);
```

```
}")
sf <- sampling(sm, data = c("J", "nj", "idx", "n", "y"))
```

Pooled

Here we have $y_{ij} \,|\, \mu, \tau^2 \overset{\text{iid}}{\sim} N(\mu, \tau^2)$, with non-informative priors $\mathbb{P}(\mu, \log \tau) \propto 1$. The joint posterior is

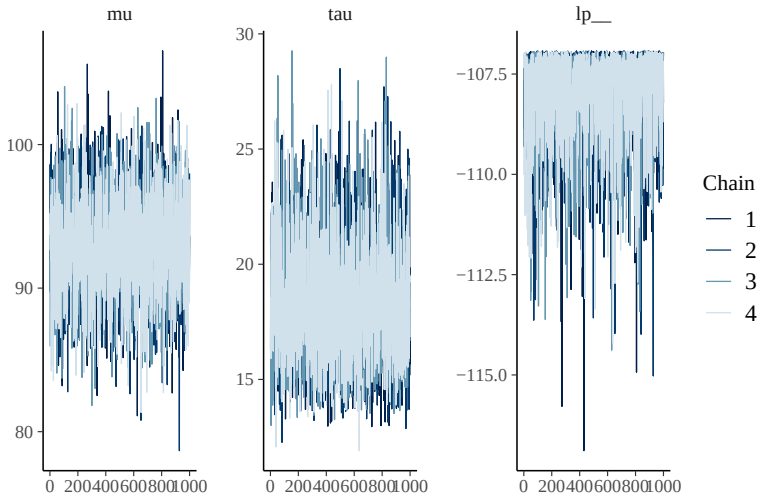
$$\mathbb{P}(\mu, \tau^2 \,|\, y) \propto (\tau^2)^{-\left(\sum_j n_j/2+1\right)} \exp\left\{ -\frac{\sum_{i,j}(y_{ij}-\mu)^2}{2\tau^2} \right\}.$$

Thus, we can see that $\mu \,|\, \tau^2, y \sim N(\bar{y}, \tau^2/\sum_j n_j)$, where $\bar{y} = \sum_{i,j} y_{ij}/\sum_j n_j$ is the overall mean, and $\tau^2 \,|\, \mu, y \sim \text{Inv-}\chi^2(\sum_j n_j, \sum_{i,j}(y_{ij}-\mu)^2/\sum_j n_j)$.

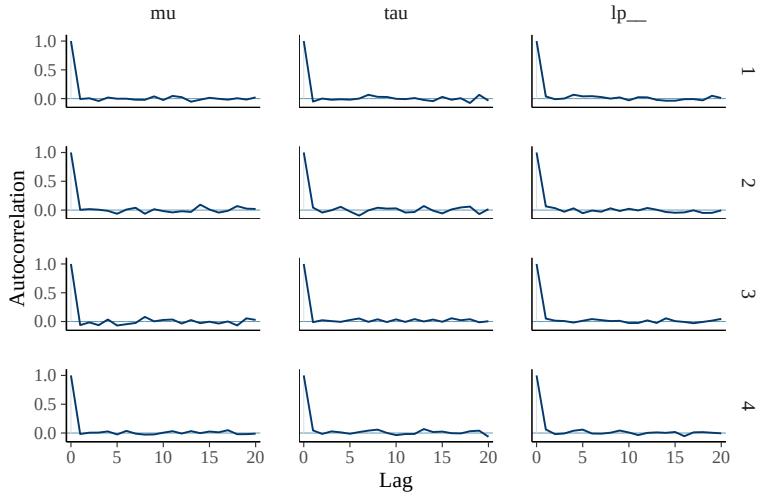
```
ybar <- weighted.mean(yj, nj); s2 <- (n - 1) * var(machine$qc)
sims <- mcmc_array(nsamples, nchains, c("mu", "tau", "lp__"))

tau2 <- s2 / (n - 1)
for (chain in 1:nchains) {
  for (it in 1:iter) {
    mu <- rnorm(1, ybar, sqrt(tau2 / n))
    nt <- s2 + n * (ybar - mu) ^ 2
    tau2 <- rinvchisq(1, n, nt)
    target <- -(n / 2 + 1) * log(tau2) - nt / (2 * tau2)
    if (it > warmup)
      sims[it - warmup, chain, ] <- c(mu, sqrt(tau2), target)
  }
}

monitor(sims, warmup = 0)
```

```
Inference for the input samples (4 chains: each with iter = 1000; warmup = 0):

         Q5    Q50    Q95   Mean  SD  Rhat Bulk_ESS Tail_ESS
mu     87.4   92.9   98.3   92.9 3.3     1     4178     3695
tau    14.8   18.1   22.9   18.4 2.5     1     3908     3831
lp__ -110.3 -107.7 -107.0 -108.0 1.1     1     3752     3621

For each parameter, Bulk_ESS and Tail_ESS are crude measures of
effective sample size for bulk and tail quantities respectively (an ESS > 100
per chain is considered good), and Rhat is the potential scale reduction
factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

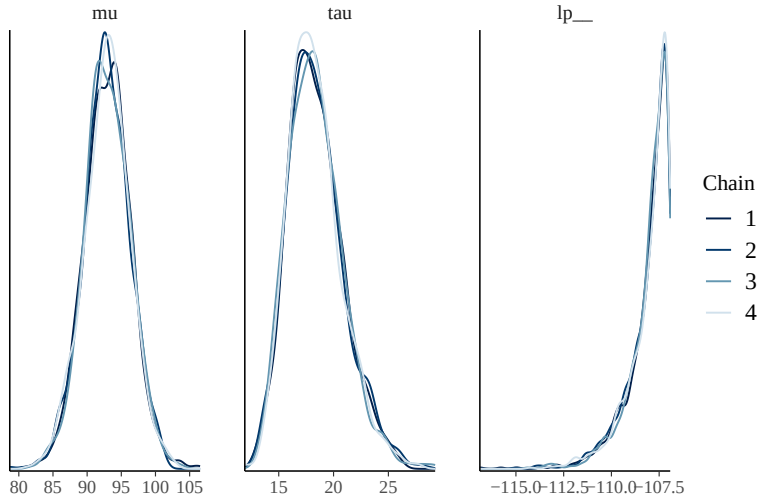
```
mcmc_trace(sims)
```




```
mcmc_acf(sims)
```

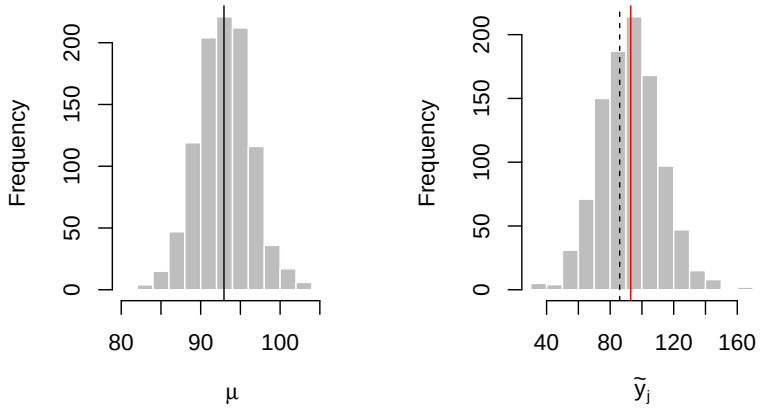



```
mcmc_dens_overlay(sims)
```



Due to pooling, $\mathbb{E}[y_6] = \mathbb{E}[y_7] = \mu$ and $\widetilde{y}_j$ has the same distribution independent of j.

```r
# as before, let's just use the first chain
mu <- sims[, 1, 1]; tau <- sims[, 1, 2]
ypred6 <- rnorm(nsamples, mu, tau)
op <- par(mfrow = c(1, 2))
plot_hist(mu, xlab = expression(mu)); abline(v = ybar)
plot_hist(ypred6, xlab = expression(tilde(y)[j]))
abline(v = ybar, col = "red"); abline(v = yj[6], lty = 2)
```




```r
par(op)
```

The Stan model is shown below. As in the separate case, the results are equivalent.

```r
sm <- stan_model(model_code = "data { int<lower=0> n; vector[n] y; }
parameters { real mu; real<lower=0> tau; }
model { y ~ normal(mu, tau); target += -2 * log(tau); }")
sf <- sampling(sm, data = c("n", "y"))
```

Hierarchical

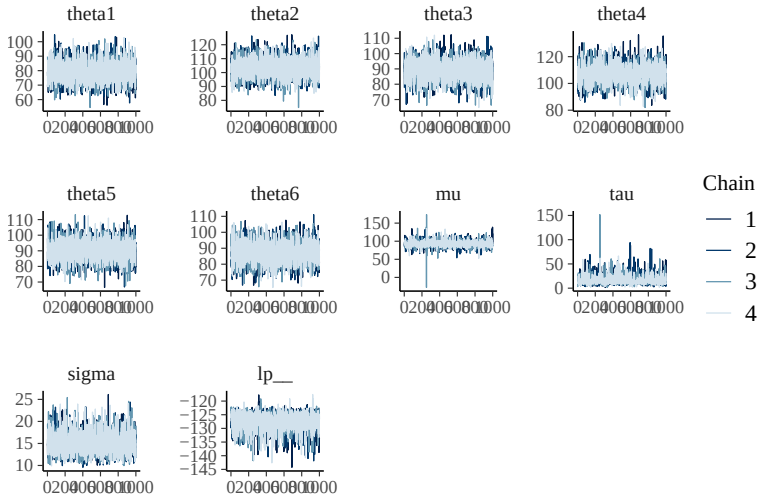
This is the model from Section 11.6 we discussed in class: $y_{ij} \,|\, \theta_j, \sigma^2 \overset{\text{ind}}{\sim} N(\theta_j, \sigma^2)$, $\theta_j \,|\, \mu, \tau^2 \overset{\text{iid}}{\sim} N(\mu, \tau^2)$, and non-informative priors $\mathbb{P}(\mu, \tau, \log\sigma) \propto 1$.

```r
sims <- mcmc_array(nsamples, nchains,
                   c(paste0("theta", 1:J), "mu", "tau", "sigma", "lp__"))

theta <- yj; mu <- mean(theta)
nt <- S + sum(nj * (yj - theta) ^ 2)
for (chain in 1:nchains) {
  for (it in 1:iter) {
    sigma2 <- rinvchisq(1, n, nt)
    tau2 <- rinvchisq(1, J - 1, sum((theta - mu) ^ 2))
    theta <- rnorm(J, (nj * yj / sigma2 + mu / tau2) / (nj / sigma2 + 1 / tau2),
                   1 / sqrt(nj / sigma2 + 1 / tau2))
    mu <- rnorm(1, mean(theta), sqrt(tau2 / J))
    nt <- S + sum(nj * (yj - theta) ^ 2)
    target <- -n / 2 * log(sigma2) - nt / (2 * sigma2) +
      sum(dnorm(theta, mu, sqrt(tau2), log = TRUE)) - log(sigma2) - .5 * log(tau2)

    if (it > warmup)
      sims[it - warmup, chain, ] <- c(theta, mu, sqrt(tau2), sqrt(sigma2), target)
  }
}
```

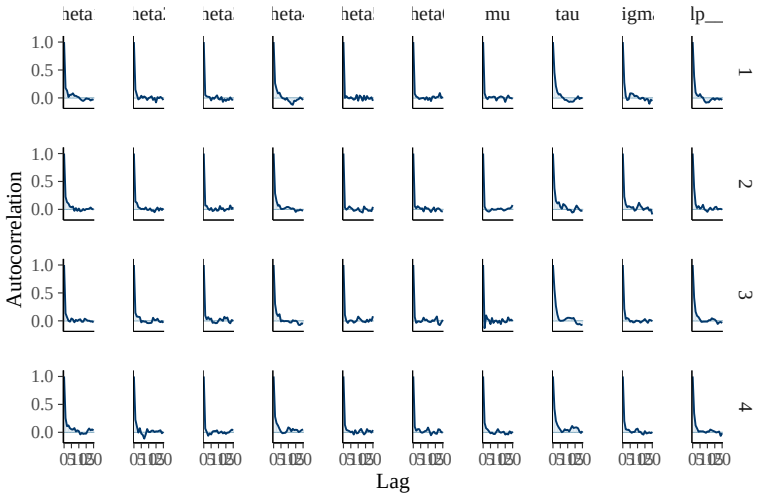
```
monitor(sims, warmup = 0)
```

```
Inference for the input samples (4 chains: each with iter = 1000; warmup = 0):

          Q5    Q50    Q95   Mean  SD  Rhat Bulk_ESS Tail_ESS
theta1  69.1   79.7   91.4   79.8 6.8     1     1834     2168
theta2  92.9  103.3  113.8  103.3 6.3     1     2480     3082
theta3  78.9   89.1   98.7   89.0 6.1     1     3278     3644
theta4  95.6  107.6  118.7  107.5 7.0     1     1820     1674
theta5  80.6   90.7  100.0   90.6 6.0     1     3371     3776
theta6  77.4   87.5   97.6   87.5 6.1     1     3178     3519
mu      81.3   93.1  105.4   93.1 8.1     1     3072     3136
tau      6.1   14.1   33.0   16.2 9.5     1     1106      911
sigma   11.7   14.7   19.2   14.9 2.3     1     2169     3239
lp__  -133.5 -127.2 -123.5 -127.7 3.1     1     1721     2241

For each parameter, Bulk_ESS and Tail_ESS are crude measures of
effective sample size for bulk and tail quantities respectively (an ESS > 100
per chain is considered good), and Rhat is the potential scale reduction
factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

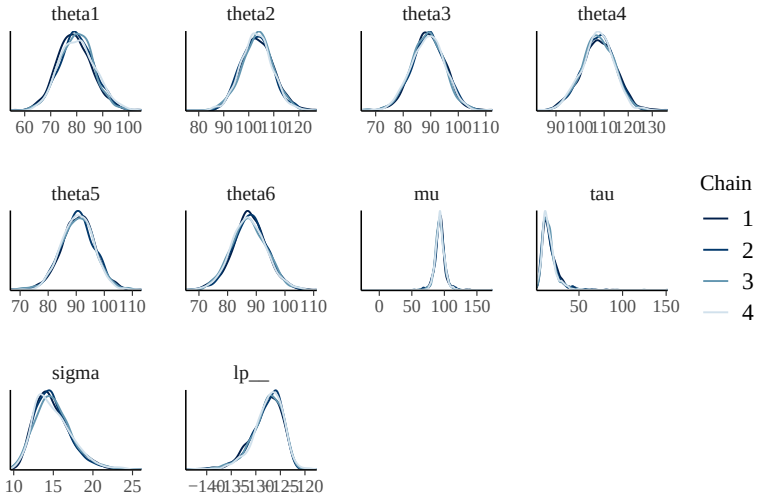
```
mcmc_trace(sims)
```




```
mcmc_acf(sims)
```

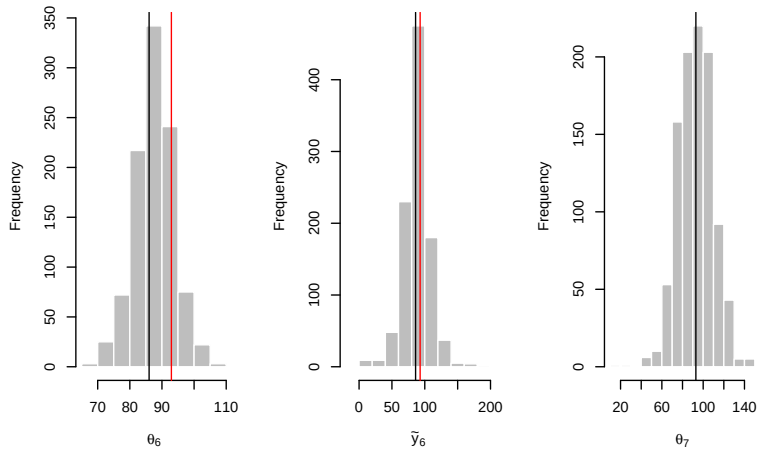



```
mcmc_dens_overlay(sims)
```



The posterior on θ_6 can be estimated directly from the MCMC samples. For $\widetilde{y}_6$ and θ_7 we have to use posterior predictive estimates.

```
theta6 <- sims[, 1, 6]; mu <- sims[, 1, 7]
sigma <- sims[, 1, 8]; tau <- sims[, 1, 9]
ypred6 <- rnorm(nsamples, theta6, sigma)
theta7 <- rnorm(nsamples, mu, tau)
op <- par(mfrow = c(1, 3))
plot_hist(theta6, xlab = expression(theta[6]))
abline(v = yj[6]); abline(v = ybar, col = "red")
plot_hist(ypred6, xlab = expression(tilde(y)[6]))
abline(v = yj[6]); abline(v = ybar, col = "red")
plot_hist(theta7, xlab = expression(theta[7]))
abline(v = ybar)
```




```
par(op)
```

3

```
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)
n <- length(y)

eps <- 1e-6
boxcox <- function (y, lambda)
  if (abs(lambda) < eps) log(y) else (y ^ lambda - 1) / lambda
```

(a)

By variable transformation, we know that

$$\mathbb{P}(y_i \,|\, \mu, \sigma^2, \lambda) = \left|\frac{dw_i}{dy_i}\right| \mathbb{P}(w_i \,|\, \mu, \sigma^2) = y_i^{\lambda-1} N(w_i \,|\, \mu, \sigma^2),$$

and so we have the joint posterior from likelihood independence and by incorporating the prior,

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

(b)

Conditional on λ we have the Box-Cox transformed responses w_i, and so the conditional posterior for μ and σ^2 depend only on w. It is straightforward to see that from $\bar{w} \,|\, \mu, \sigma^2 \sim N(\mu, \sigma^2/n)$ and a flat prior on μ we have $\mu \,|\, \lambda, \sigma^2, y \sim N(\bar{w}, \sigma^2/n)$ and, from the joint posterior above,

$$\mathbb{P}(\sigma^2 \,|\, \mu, \lambda, y) \propto (\sigma^2)^{-((n-1)/2+1)} \exp\left\{-\frac{\sum_i (w_i(\lambda) - \mu)^2}{2\sigma^2}\right\},$$

that is, $\sigma^2 \,|\, \mu, \lambda, y \sim \mathsf{Inv}\text{-}\chi^2(n-1, \sum_i (w_i(\lambda) - \mu)^2/n)$. For the conditional posterior $\lambda \,|\, \mu, \sigma^2, y$ we have to resort to a random walk MH step. By hand tuning we set the random walk standard deviation at 0.1 and thin out every 10th MCMC sample to reduce autocorrelation.

```
sigma_lambda <- .1
nthin <- 10
sims <- mcmc_array(nsamples, nchains, c("mu", "sigma", "lambda", "lp__"))

for (chain in 1:nchains) {
  lambda <- 0; sigma <- 0
  w <- boxcox(y, lambda)
  for (it in 1:iter) {
    for (thin in 1:nthin) {
      mu <- rnorm(1, mean(w), sigma / sqrt(n))
      sigma <- sqrt(rinvchisq(1, n, sum((w - mu) ^ 2)))
      target <- sum(dnorm(w, mu, sigma, log = TRUE)) +
        (lambda - 1) * sum(log(y))
      # lambda | mu, sigma2:
      lambda_c <- rnorm(1, lambda, sigma_lambda)
      w_c <- boxcox(y, lambda_c)
      target_c <- sum(dnorm(w_c, mu, sigma, log = TRUE)) +
        (lambda_c - 1) * sum(log(y))
      log_r <- target_c - target
      if (log_r >= 0 || log(runif(1)) <= log_r) { # accept?
        lambda <- lambda_c; w <- w_c; target <- target_c
      }
    }

    if (it > warmup)
      sims[it - warmup, chain, ] <- c(mu, sigma, lambda, target - log(sigma))
  }
}
```

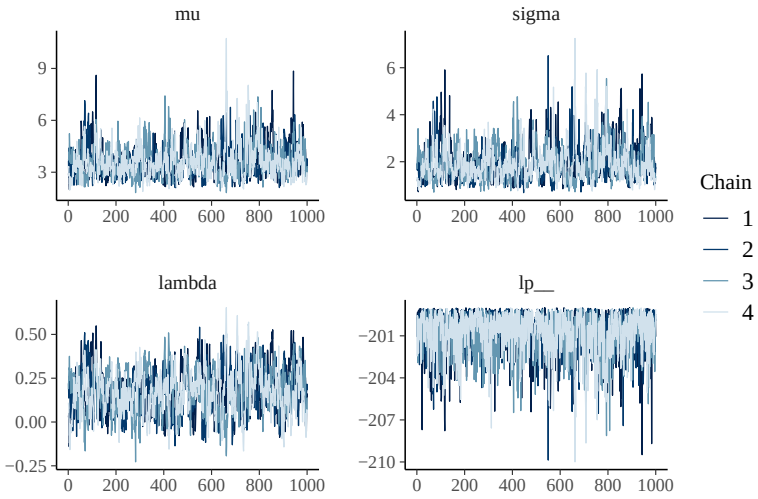
(c)

The chains seem to have converged, but with high autocorrelation (even after thinning). Since the 95% credible interval for λ includes 0 but not 0.5, a log transformation is more likely than a square root transformation.

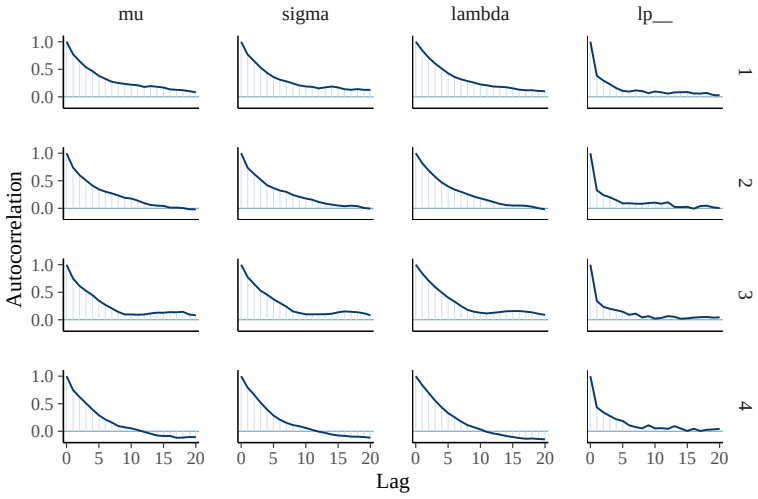
```
monitor(sims, warmup = 0)
```

```
Inference for the input samples (4 chains: each with iter = 1000; warmup = 0):

          Q5    Q50    Q95   Mean  SD  Rhat Bulk_ESS Tail_ESS
mu       2.5    3.5    5.3    3.6 0.9  1.01      431      795
sigma    1.0    1.8    3.2    1.9 0.7  1.01      417      843
lambda   0.0    0.2    0.4    0.2 0.1  1.01      390      659
lp__  -203.6 -200.4 -199.2 -200.8 1.5  1.00      865     1337

For each parameter, Bulk_ESS and Tail_ESS are crude measures of
effective sample size for bulk and tail quantities respectively (an ESS > 100
per chain is considered good), and Rhat is the potential scale reduction
factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

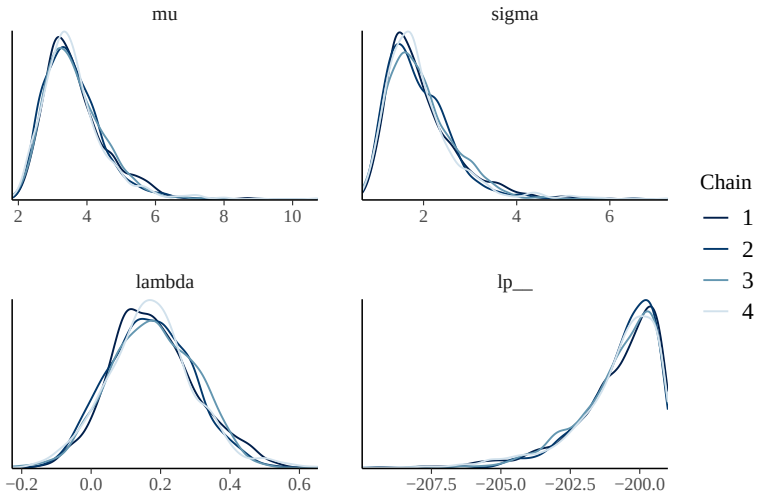
```
mcmc_trace(sims)
```




```
mcmc_acf(sims)
```




```
mcmc_dens_overlay(sims)
```




```
Q <- function (alpha) function (x) quantile(x, alpha)
alpha <- .05
as_tibble(sims[, 1, 1:3]) %>%
```

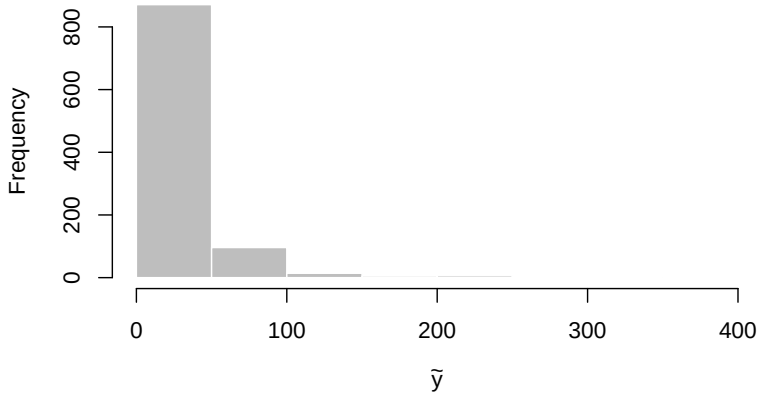
```
  summarize_all(list(lower = Q(alpha / 2), upper = Q(1 - alpha / 2)))
```

```
# A tibble: 1 x 6
  mu_lower sigma_lower lambda_lower mu_upper sigma_upper lambda_upper
     <dbl>       <dbl>        <dbl>    <dbl>       <dbl>        <dbl>
1     2.39       0.996      -0.0463     5.80        3.85        0.447
```

(d)

The posterior predictive distribution is smoother and unimodal given the simple model we have adopted, but it seems to be in good agreement, overall, with the data.

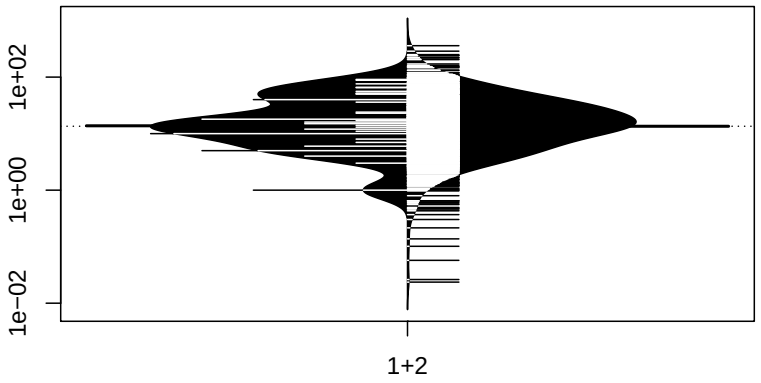
```
mu <- sims[, 1, 1]; sigma <- sims[, 1, 2]; lambda <- sims[, 1, 3]
wpred <- rnorm(nsamples, mu, sigma)
ypred <- (lambda * wpred + 1) ^ (1 / lambda)
plot_hist(ypred, xlab = expression(tilde(y)))
```




```
quantile(ypred, c(.025, .975), na.rm = TRUE)
```

```
      2.5%      97.5%
  1.126085 105.503216
```

```
beanplot(y, ypred, side = "both")
```



(extra: Stan)

Here's Stan code for the same model. The results are very similar, but Stan has slightly less autocorrelation.

```
functions {
  real boxcox (real y, real lambda) {
    real eps = 1e-6;
```

```
    if (fabs(lambda) < eps) return log(y);
    return (y ^ lambda - 1) / lambda;
  }
}
data {
  int<lower=0> n;
  vector[n] y;
}
parameters {
  real mu;
  real<lower=0> sigma;
  real lambda;
}
transformed parameters {
  vector[n] w;
  for (i in 1:n) w[i] = boxcox(y[i], lambda);
}
model {
  w ~ normal(mu, sigma);
  target += (lambda - 1) * sum(log(y)); // log Jacobian
}
```