Bayesian inference: practical exercises

Robert Goudie

Full worked solutions are provided at the back of this document. Feel free to consult these as you are going along, if you are stuck on any of the exercises.

The document and embedded code are also provided as an R Markdown document in `bayesian_inference_practical.Rmd`. This allows the R code blocks to be run conveniently from RStudio.

1 Conjugate inference in R

1. [Beta-Binomial prediction] Consider the disease you worked with in the previous session, where you determined that a Beta(4.168, 37.515) distribution was suitable to describe your prior belief. Suppose we observe that, in a particular population 9 out of 50 people survive this disease.

   a. Use the mathematical derivation in the Lecture, write down the posterior distribution for θ. Plot the density of the posterior distribution, using the function `dbeta`. The `ppoints` function may be helpful for creating a grid of points between zero and 1.

   Compare this with the prior results from the previous session. Can you explain the direction of the change between the prior and posterior?

   b. Again using the mathematical derivation in the Lecture, write down the posterior predictive distribution in a population of size 100. Use the function `pbb` from the `TailRank` package to calculate the posterior predicted probability that more than 20 people in this population of 100 people will survive the disease. (Note `pbb(x, )` gives $P(y \leq x)$)

   Compare this with the prior predictive results from the previous session. Can you explain the direction of the change between the prior and posterior?

   (c) Use the function `qbb` from `TailRank` to calculate a 95% equal-tailed posterior credible interval for y.

2. Use Monte Carlo simulation in R to verify the answers from part (1). Ensure that there are enough samples that the probability computed in (1b) is accurate to within around ± 0.001, as measured by a Monte Carlo standard error of 0.0005 or less.

3. Consider again the pollution measurement with a normal distribution with mean μ and standard deviation $\sigma = 8$. The mean has a normal prior distribution with mean $m_\mu = 120$ and standard deviation $s_\mu = 10$. Suppose we now observe that $y = 140$.

   (a) Using the mathematical derivation in the lectures, calculate the posterior distribution for μ and plot this in R. Also plot the prior for μ.

   Suppose the maximum legal level for the pollutant is 145 μ_g/m^3. Calculate the posterior probability that the measurement X will exceed this limit.

2 Basic JAGS model - demonstration

2.1 JAGS model code

The following JAGS code is the Drug model we considered in the lecture, with predictions for the number of successes out of m future trials.

```
drug_jagsmod <- "
model {
  theta ~ dbeta(a, b)
  r ~ dbin(theta, n)
  r.pred ~ dbin(theta, m)
}"
```

The fixed constants and data are defined in a R list.

```
dat <- list(a = 9.2, b = 13.8, n = 20, r = 15, m = 40)
```

Describe the role of each of the items in this list. Which of these quantities are data (observations) and which are fixed constants?

The model and data will be supplied to JAGS using the `jags.model` function.

2.2 Running JAGS

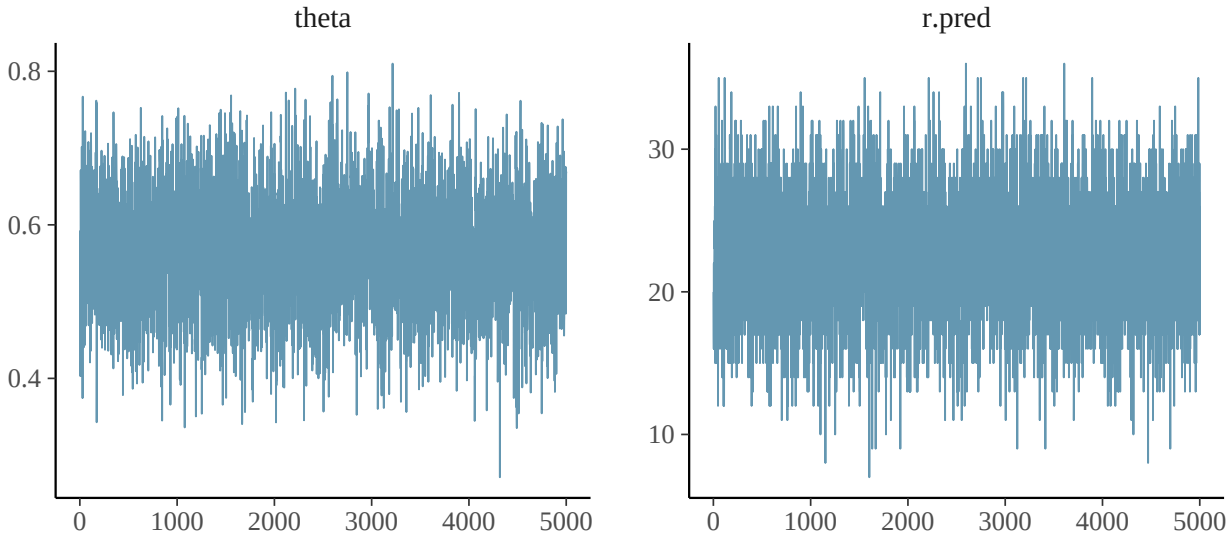
After calling `jags.model` to define the model and data, we then draw a sample of 5000 values from the MCMC chains for the parameter θ.

```
library(rjags)
drug_jag <- jags.model(textConnection(drug_jagsmod),
                       data=dat, n.chains=1, quiet=TRUE)
pars_basic <- c("theta", "r.pred")
sam <- coda.samples(drug_jag, c(pars_basic), n.iter=5000)
```

2.3 Convergence diagnostics

In more complex examples it will be necessary to look at "convergence diagnostics", such as a trace plot, which is a simple plot of the sampled values against the sample/iteration number.

```
library(bayesplot)
mcmc_trace(sam, pars_basic)
```



Another R package, called `posterior`[1], is then loaded to compute numerical convergence diagnostics for the parameters of interest α, β and σ.

```
library(posterior)
draws <- as_draws(sam)
subs <- subset_draws(draws, variable=pars_basic)
summary(subs)
```

```
## # A tibble: 2 x 10
##   variable   mean median     sd    mad     q5    q95  rhat ess_bulk ess_tail
##   <chr>     <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <dbl>    <dbl>    <dbl>
## 1 theta     0.563  0.563 0.0761 0.0777  0.439  0.687  1.00    2966.    2610.
## 2 r.pred   22.5    23     4.32   4.45   15     29      1.00    3821.    3950.
```

The probability of exceeding 25 successes in a future set of 40 trials can be calculated:

```
r.pred <- extract_variable(draws, "r.pred")
mean(r.pred >= 25)
```

```
## [1] 0.331
```

3 Normal distribution with known variance in JAGS

Implement a JAGS model for the Assay example in the lecture, and reproduce the results in the lecture.

In the notation of the lecture, there are 5 variables in the model: y_i, μ, σ^2, γ and ω^2. Which of these have a posterior distribution?

Suppose you are going to repeat the measurement again on this specimen, again using the assay with measurement error with standard deviation 5 IU/ml. What is the mean and standard deviation for a prediction for this observation?

[1] https://cran.r-project.org/web/packages/posterior/vignettes/posterior.html

4 Convergence assessment in JAGS

Consider the following model.

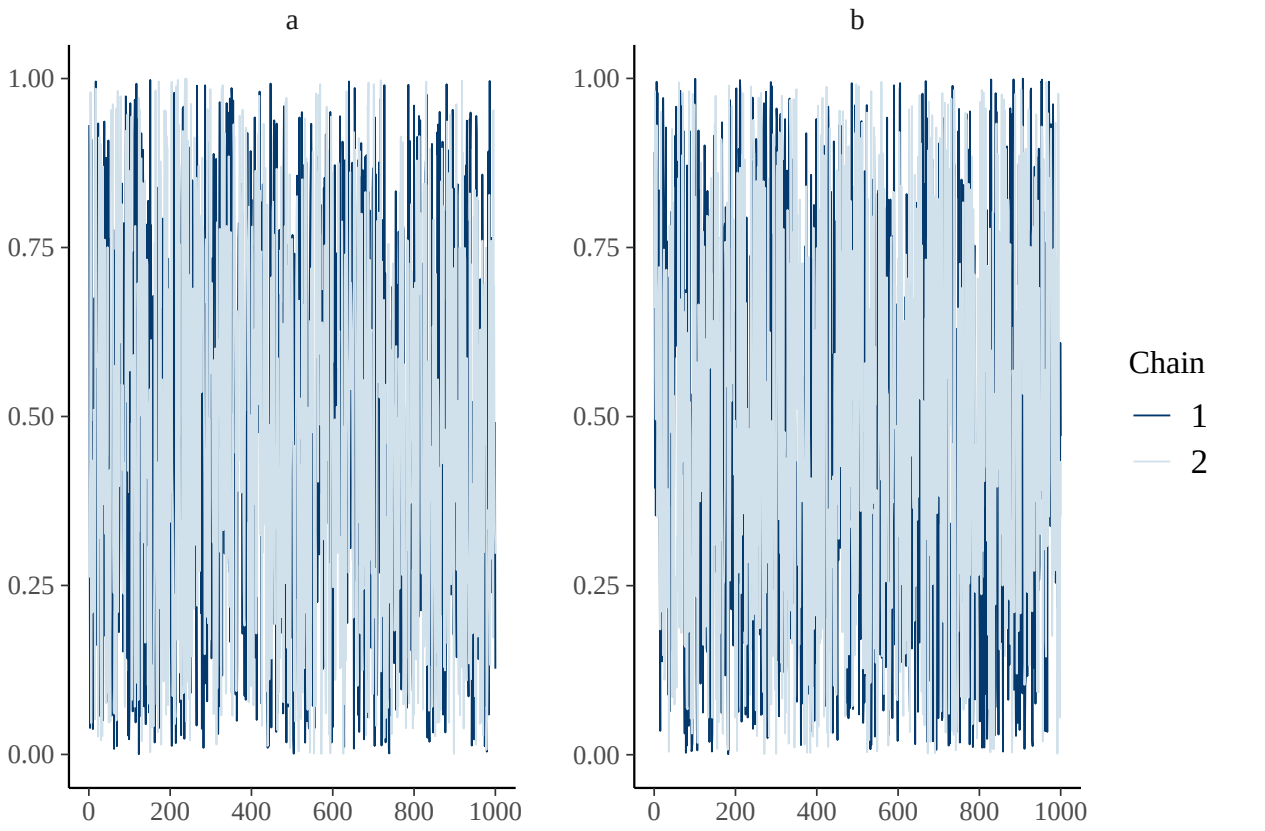
$$y \sim N(a+b, \sigma^2)$$
$$a \sim Uniform(0,1)$$
$$b \sim Uniform(0,1)$$

The model posits that the observed y is the sum of two unobserved quantites a and b, with the observations having fixed known variance σ^2. Suppose we observe that $y = 0.6$.

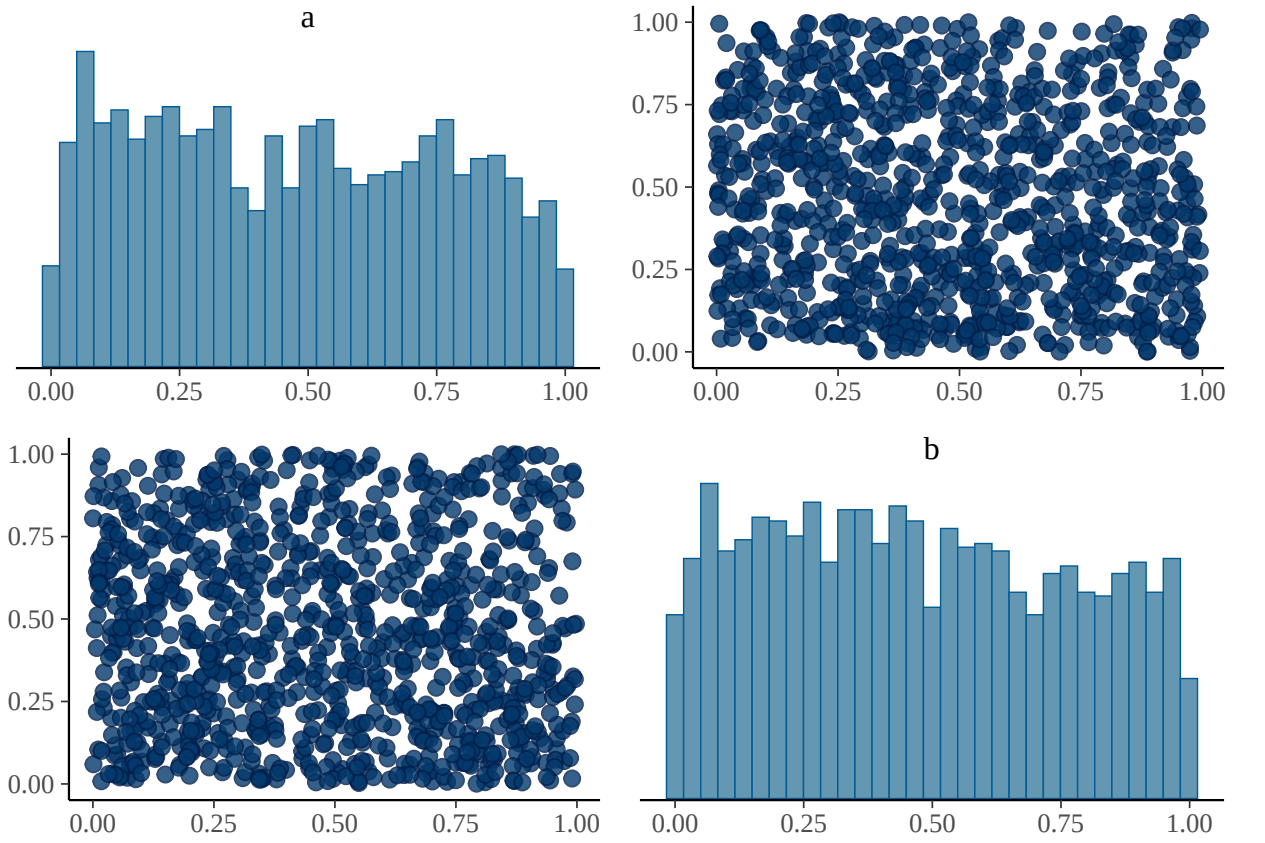
This model is fairly useless and ill-conceived, since we have no information to distinguish a or b (these parameters are not "identifiable"), but similar problems to those exhibited by this model often occur in more subtle ways in realistic (and useful!) models.

The following code fits this model, with $tau = 1/\sigma^2 = 1$, and running 1000 iterations for burn-in (which is required because this model is not conjugate), and then 1000 iterations post-burn-in for inference.

```
nasty_jagsmod <- "
model {
  y ~ dnorm(a + b, tau)
  a ~ dunif(0, 1)
  b ~ dunif(0, 1)
}"

nasty_dat <- list(y = 0.6, tau = 1)

# choose two initial values
nasty_in <- list(list(a = 0.1, b = 0.5),
                 list(a = 0.5, b = 0.1))

nasty_jag <- jags.model(textConnection(nasty_jagsmod),
                        data=nasty_dat, inits=nasty_in, n.chains=2, quiet=TRUE)
pars_basic <- c("a", "b")

# 1000 samples used as burn-in
# using update() means we don't even keep a record of what these samples
# are - assuming we are happy to discard these samples regardless, then there is no need to record them
update(nasty_jag, 1000)

# 1000 samples for inference
nasty_post <- coda.samples(nasty_jag, c(pars_basic), n.iter=1000)

library(bayesplot)
mcmc_trace(nasty_post, pars_basic)
```

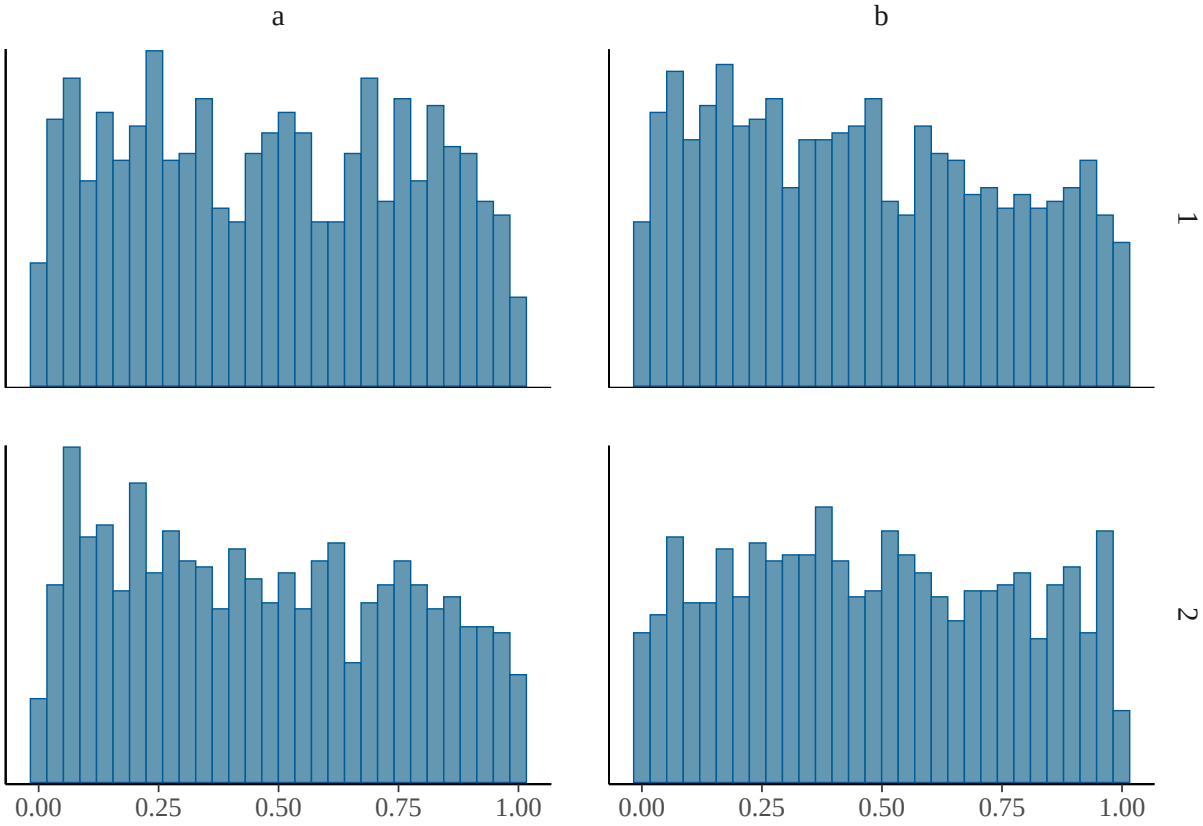



```
mcmc_pairs(nasty_post)
```



```
mcmc_hist_by_chain(nasty_post, pars_basic)
```

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




```
library(posterior)
assay_draws <- as_draws(nasty_post)
summary(assay_draws)
```

```
## # A tibble: 2 x 10
##   variable  mean median    sd   mad     q5   q95  rhat ess_bulk ess_tail
##   <chr>    <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 a        0.472  0.462 0.288 0.374 0.0502 0.936  1.00    1245.    1764.
## 2 b        0.472  0.456 0.288 0.359 0.0383 0.947  1.00    1217.    1583.
```

```
summary(assay_draws, default_convergence_measures())
```

```
## # A tibble: 2 x 4
##   variable  rhat ess_bulk ess_tail
##   <chr>    <dbl>    <dbl>    <dbl>
## 1 a         1.00    1245.    1764.
## 2 b         1.00    1217.    1583.
```

```
summary(assay_draws, default_mcse_measures())
```

```
## # A tibble: 2 x 6
##   variable mcse_mean mcse_median mcse_sd mcse_q5 mcse_q95
##   <chr>        <dbl>       <dbl>   <dbl>   <dbl>    <dbl>
## 1 a          0.00833      0.0167 0.00589 0.00352  0.00973
## 2 b          0.00821      0.0147 0.00581 0.00788  0.00619
```

Questions:

- (a) Are you happy with these results or do we need to run further MCMC iterations?
- (b) Suppose now that $tau = 1/\sigma^2 = 10000$. How many iterations do you need to obtain reliable inferences?

5 Two-stage Gibbs sampler in R

Consider the bivariate normal distribution with mean 0 and covariance matrix

$$\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$$

1. In R, write a two-stage Gibbs sampler for this distribution. With $\rho = 0$, draw 100 samples using your Gibbs sampler.
2. Alternatively, we could use the package `mvtnorm` to sample from this bivariate normal distribution directly:

   ```
   library(mvtnorm)
   sigma <- matrix(c(1, rho, rho, 1), 2, 2)
   theta_direct <- rmvnorm(niter, mean = c(0, 0), sigma = sigma)
   ```

   Since the target distribution is a bivariate normal, the marginal distributions of the target are (univariate) normal. Using QQ-plots, compare the normality of the marginal distributions of the samples drawn using the Gibbs sampler and using `rmvnorm`.
3. Repeat the process with $\rho = 0.9999$. Has the performance of the algorithms changed?

6 Gibbs sampler for a normal with unknown mean and variance

Consider the model

$$\begin{aligned} y_i &\sim N(\mu, \sigma^2) && i = 1, \ldots, n \\ \mu &\sim N(\mu_0, \psi^2) \\ \sigma^2 &\sim \text{Inverse-Gamma}(a, b) \end{aligned}$$

The full conditional distributions under the posterior are

$$\mu \,|\, (y_1, \ldots y_n, \mu) \sim \text{N}\left(\frac{\sigma^2}{\sigma^2 + n\psi^2}\mu_0 + \frac{n\psi^2}{\sigma^2 + n\psi^2}\frac{1}{n}\sum_{i=1}^{n} y_i \,,\; \frac{\sigma^2\psi^2}{\sigma^2 + n\psi^2} \right)$$

$$\sigma^2 \,|\, (y_1, \ldots y_n, \mu) \sim \text{Inverse-Gamma}\left(\frac{n}{2} + a \,,\; \frac{1}{2}\sum_{i=1} n(y_i - \mu)^2 + b \right)$$

Note that we can draw an inverse gamma by calculating the inverse of a draw from a gamma in R.

```
g <- rgamma(1, shape = a, rate = b)
h <- 1/g
# h is a draw from an inverse-gamma(a, b)
```

Suppose we observe $y = (3, 5, 2, 3, 7)$ and choose $a = 2$ and $b = 3$ and $\psi = 2$. Write a two-stage Gibbs sampler in R to sample the posterior distribution. Draw 1000 samples using your Gibbs sampler, and summarise your results.

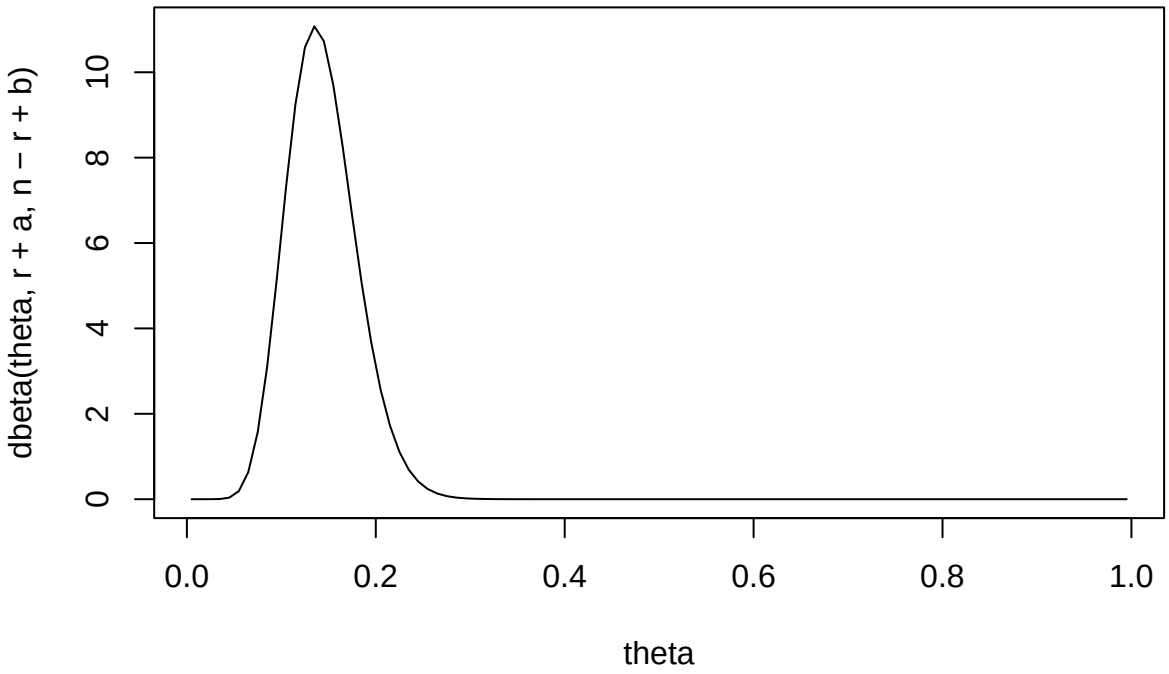
7 Solutions

7.1 Conjugate inference in R

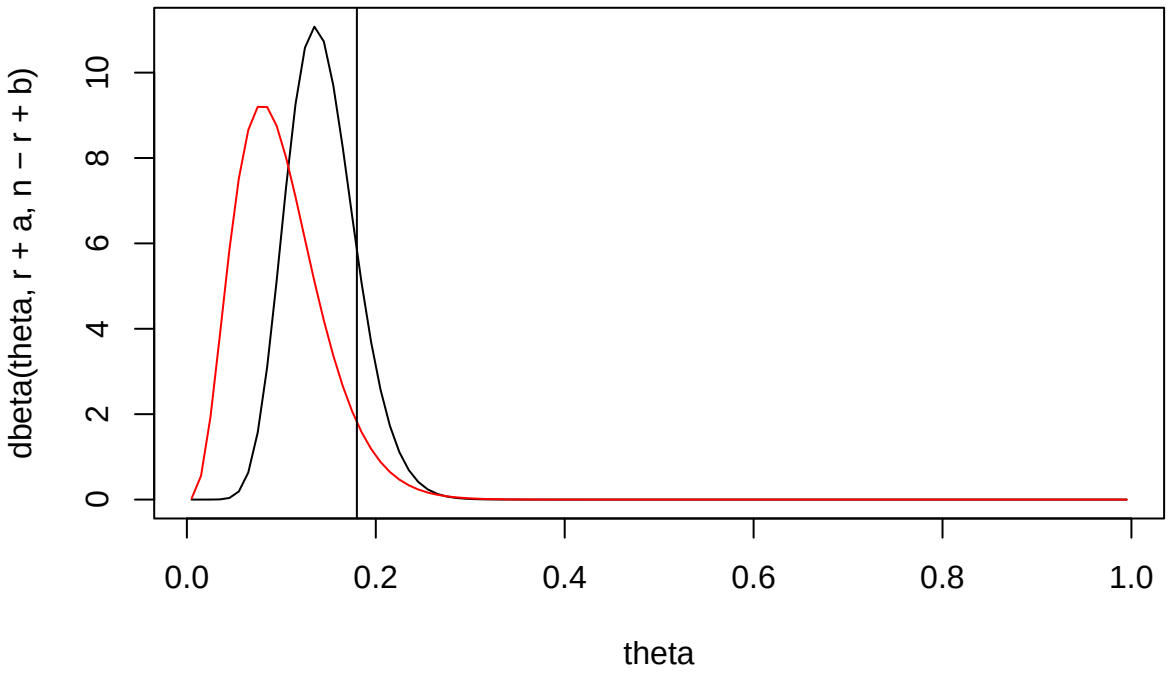
1. a.

The posterior distribution is $\theta \sim \text{Beta}(9 + 4.168, 50 - 9 + 37.515)$, that is $\theta \sim \text{Beta}(13.168, 78.515)$

```
a <- 4.168
b <- 37.515
r <- 9
n <- 50
theta <- ppoints(100)
plot(theta, dbeta(theta, r + a, n - r + b), type="l")
```




```
plot(theta, dbeta(theta, r + a, n - r + b), type="l")
# add prior
lines(theta, dbeta(theta, a, b), col = "red")
# add line showing data
abline(v = 9/50)
```



The observed data suggests θ is higher than most of the prior, so the posterior is shifted upwards toward this.

b. *The posterior predictive distribution is* $y_{pred} \sim$ *Beta-Binomial*$(9 + 4.168, 50 - 9 + 37.515, 100)$*, that is* $y_{pred} \sim$ *Beta-Binomial*$(13.168, 78.515, 100)$

```
library(TailRank)
```

```
## Loading required package: oompaBase
```

```
a <- 4.168
b <- 37.515
r <- 9
n <- 50
1 - pbb(20, 100, r + a, n - r + b)
```

```
## [1] 0.1170895
```

The observed data suggests θ is higher than most of the prior, so the posterior predictive distribution is shifted upwards compared to the prior predictive distribution.

(c)

```
qbb(c(0.025, 0.975), 100, r + a, n - r + b)
```

```
## [1]  6 25
```

2.
```
a <- 4.168
b <- 37.515
r <- 9
n <- 50

R <- 450000
theta <- rbeta(R, r + a, n - r + b)
y_pred <- rbinom(R, 100, theta)
mean(y_pred > 20)
```

```
## [1] 0.1169289
```

```
(mcse <- sd(y_pred > 20) / sqrt(R))
```

```
## [1] 0.0004790193
```

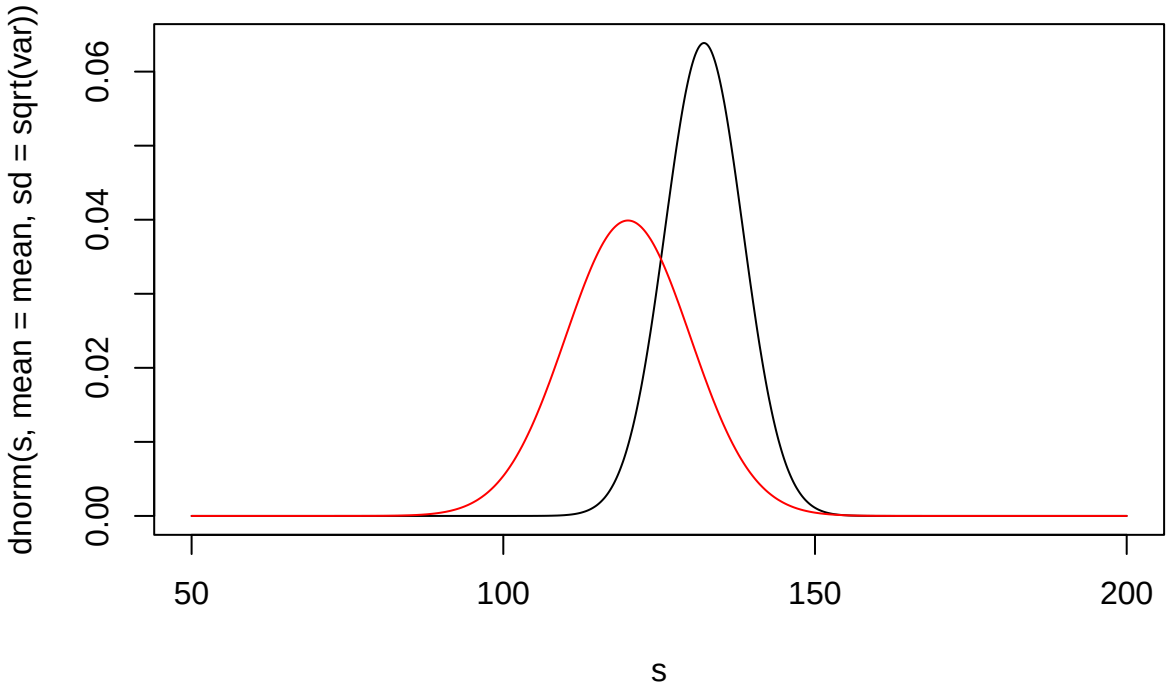
Predicting $R = 450000$ samples gives a Monte Carlo standard error (MCSE) of just less than 0.0005 for the probability (in theory, the estimate of MCSE itself is affected by Monte Carlo error, but the finding in this case is stable).

```
quantile(y_pred, c(0.025, 0.975))
```

```
##  2.5% 97.5%
##     6    25
```

3. (a)

```
m_mu <- 120
sigma <- 8
s_mu <- 10
n0 <- (sigma/10)^2
y <- 140
n <- length(y)
ybar <- mean(y)
mean <- (n0 * m_mu + n * ybar)/(n0 + n)
var <- sigma^2/(n0 + n)

s <- seq(50, 200, length.out = 1000)
plot(s, dnorm(s, mean = mean, sd = sqrt(var)), type="l")
lines(s, dnorm(s, mean = m_mu, sd = s_mu), col = "red")
```




```
1 - pnorm(145, mean=mean, sd = sqrt(var))
```

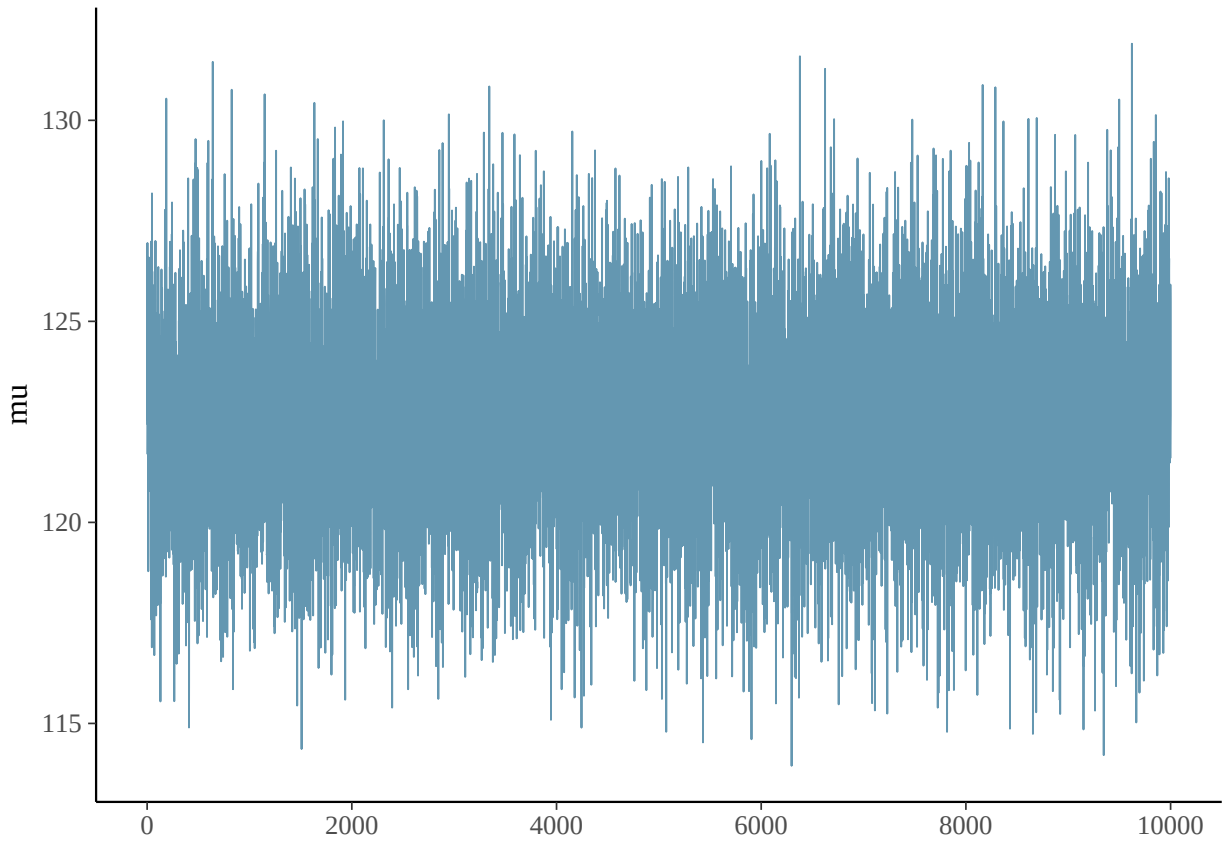
```
## [1] 0.02019292
```

7.2 Basic JAGS model - demonstration

The quantity r is an observation, whereas n is the fixed total number of trials and m is the fixed total number of trials in the future. a and b are fixed constants that specify the prior distribution for theta.

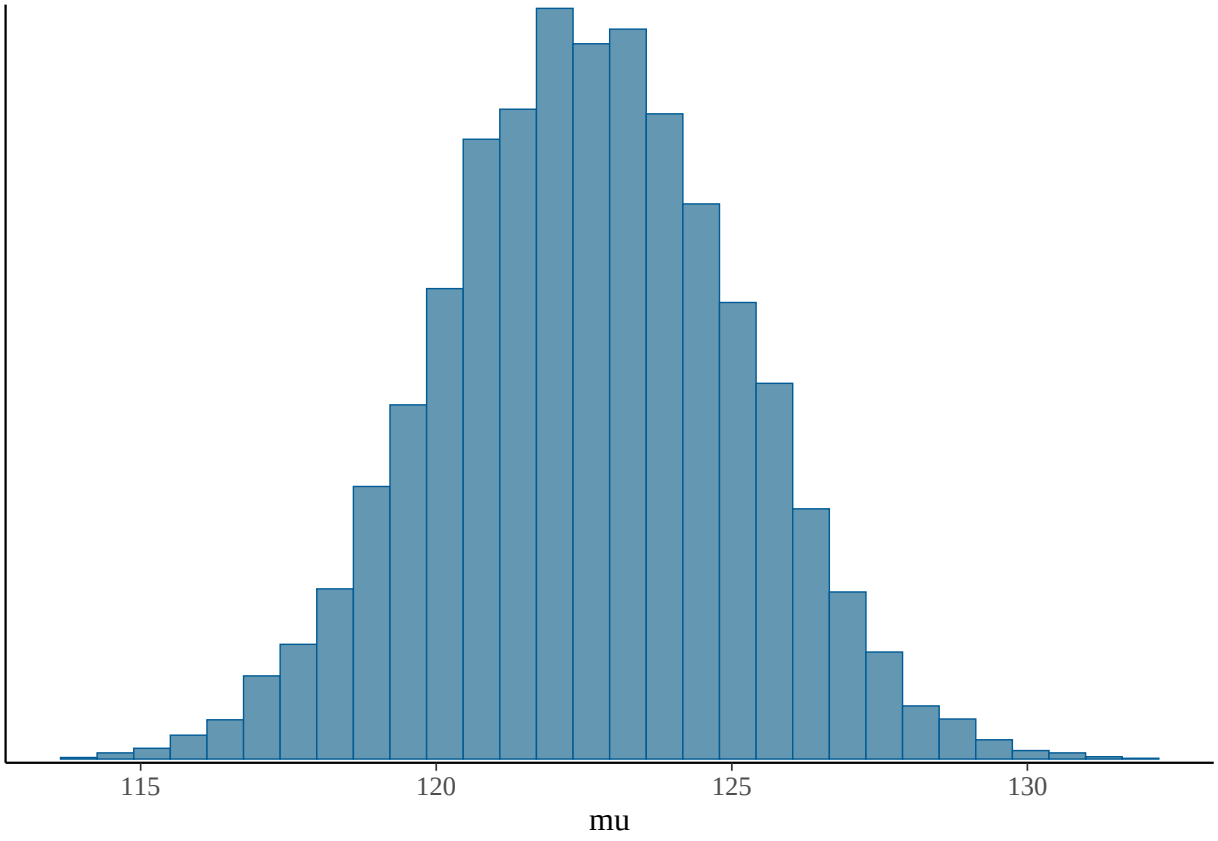
7.3 Normal distribution with known variance in JAGS

```
assay_jagsmod <- "
model {
  for (i in 1:n){
    y[i] ~ dnorm(mu, tau)
  }
  mu ~ dnorm(gamma, phi)
  omega.squared <- 1/phi
}"
y <- c(119.04, 144.08, 116.57)
assay_dat <- list(y = y,
            n = length(y),
            tau = 1/(5^2),
            gamma = 108,
            phi = 1/(5.5^2))
assay_in <- list(list(mu = 120),
                 list(mu = 150))
library(rjags)
assay_jag <- jags.model(textConnection(assay_jagsmod),
                        data=assay_dat, inits=assay_in[[1]], n.chains=1, quiet=TRUE)
pars_basic <- c("mu")
update(assay_jag, 5000)
assay_post <- coda.samples(assay_jag, c(pars_basic), n.iter=10000)

library(bayesplot)
mcmc_trace(assay_post, pars_basic)
```




```
mcmc_hist(assay_post, pars_basic)
```

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




```
library(posterior)
assay_draws <- as_draws(assay_post)
summary(assay_draws)
```

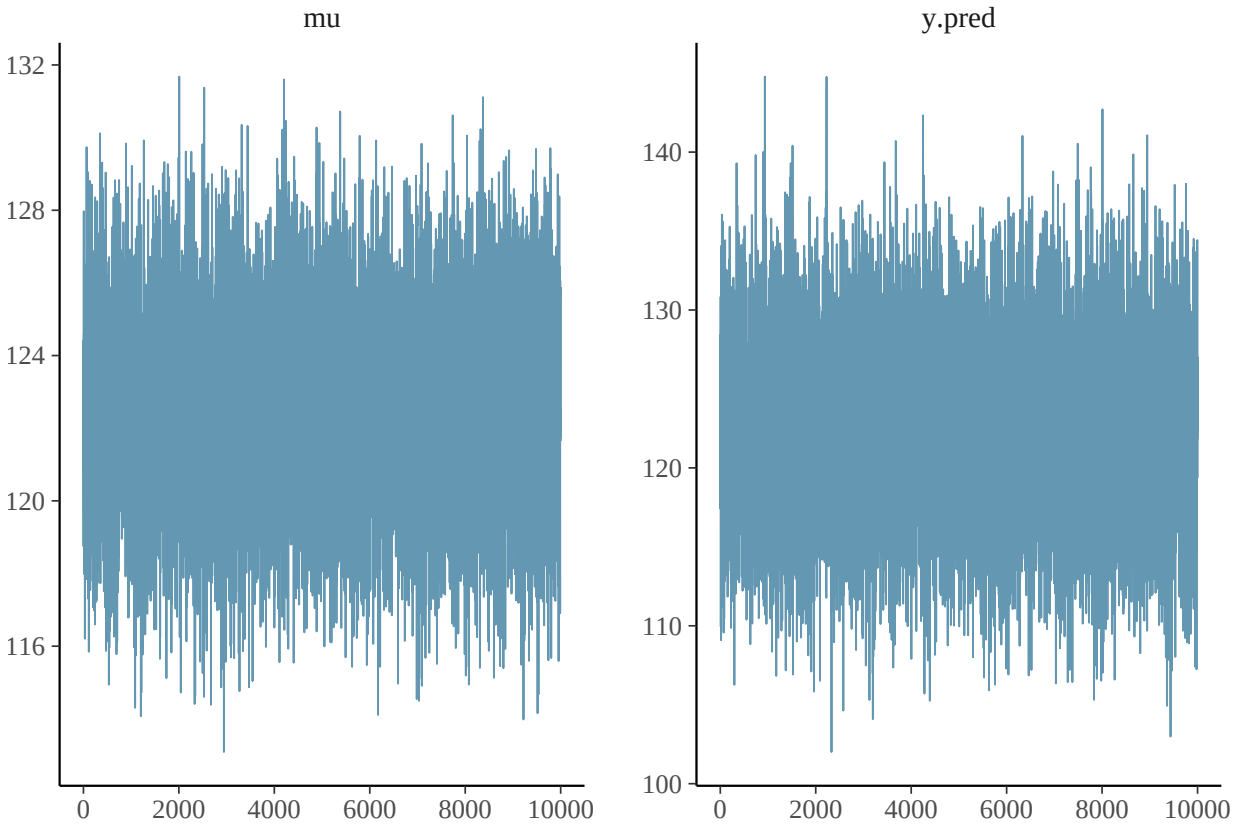
```
## # A tibble: 1 x 10
##   variable  mean median    sd   mad    q5   q95  rhat ess_bulk ess_tail
##   <chr>    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 mu        123.   123.  2.54  2.55  118.  127.  1.00    9515.    9488.
```

- Only μ has a posterior distribution.
- Each y_i represents a single data point. Before observing the data point, it has a distribution (the prior predictive distribution); but after observing it, it is a fixed quantity.
- The variance σ^2 is assumed known in this example so is a fixed quantity, with no prior or posterior distribution.
- γ and ω^2 are fixed quantities that specify the mean and variance of the prior distribution for μ. Since they are fixed they themselves have no prior distribution or posterior distribution. Note, however, that analogous quantities for γ and ω^2 exist for the posterior distribution for μ: specifically its mean $\frac{n_0\gamma+n\overline{y}}{n_0+n}$ and variance $\frac{\sigma^2}{n_0+n}$

Adding a prediction for the future measurement:

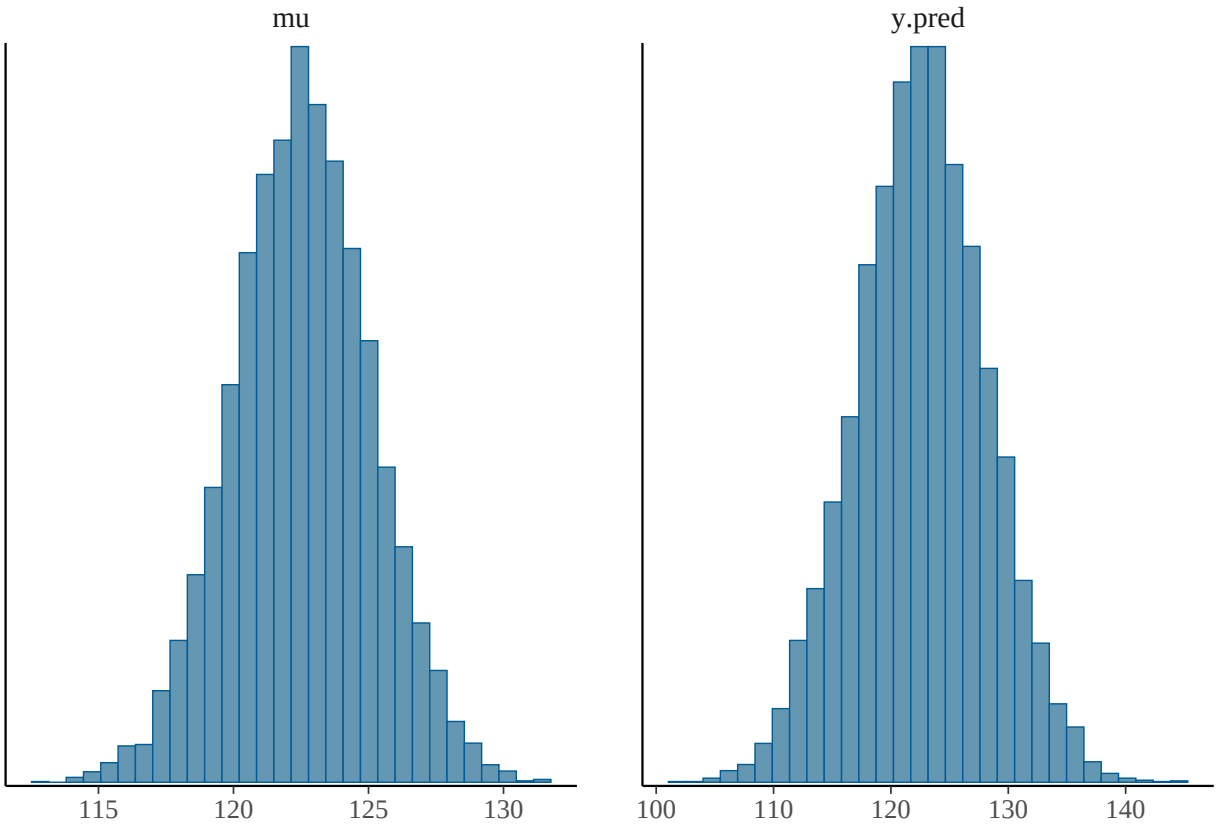
```
assay_jagsmod2 <- "
model {
  for (i in 1:n){
    y[i] ~ dnorm(mu, tau)
  }
  mu ~ dnorm(gamma, phi)
```

```
  y.pred ~ dnorm(mu, tau)
}"
assay_jag <- jags.model(textConnection(assay_jagsmod2),
                        data=assay_dat, inits=assay_in[[1]], n.chains=1, quiet=TRUE)
pars_basic <- c("mu", "y.pred")
update(assay_jag, 5000)
assay_post <- coda.samples(assay_jag, c(pars_basic), n.iter=10000)

library(bayesplot)
mcmc_trace(assay_post, pars_basic)
```




```
mcmc_hist(assay_post, pars_basic)
```

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




```
library(posterior)
assay_draws <- as_draws(assay_post)
summary(assay_draws)
```

```
## # A tibble: 2 x 10
##   variable  mean median    sd   mad    q5   q95  rhat ess_bulk ess_tail
##   <chr>    <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 mu        123.   123.  2.57  2.55  118.  127.  1.00   10016.    9878.
## 2 y.pred    123.   123.  5.55  5.52  113.  132.  1.00   10009.    9125.
```

The mean is 123 with standard deviation 5.59 – note this is much larger than the standard deviation for θ.

7.4 Convergence assessment in JAGS

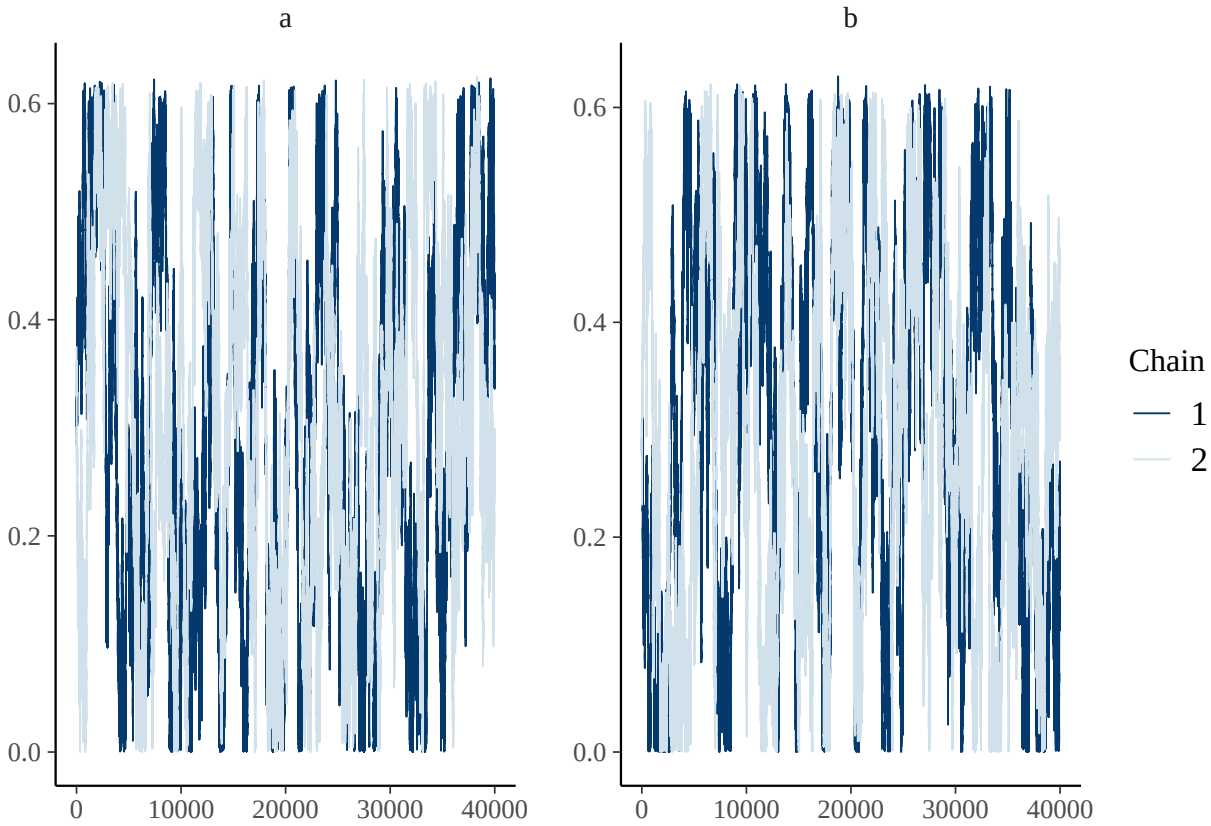
- (a) The convergence of the model with $tau = 1$ appears fine: the traceplots appear to show a "fat hairy caterpillar": a stable mean with random dispersion around the mean. Both chains completely overlap. The rhat statistic is estimated to be 1.00, giving no suggestion of any lack of convergence.

  Supposing that the quantity of interest was, say, the mean of a, then this would be enough samples post-convergence if $mean(a) \pm 2 * mcse\_mean(a)$ was constant to number of decimal places of accuracy required.

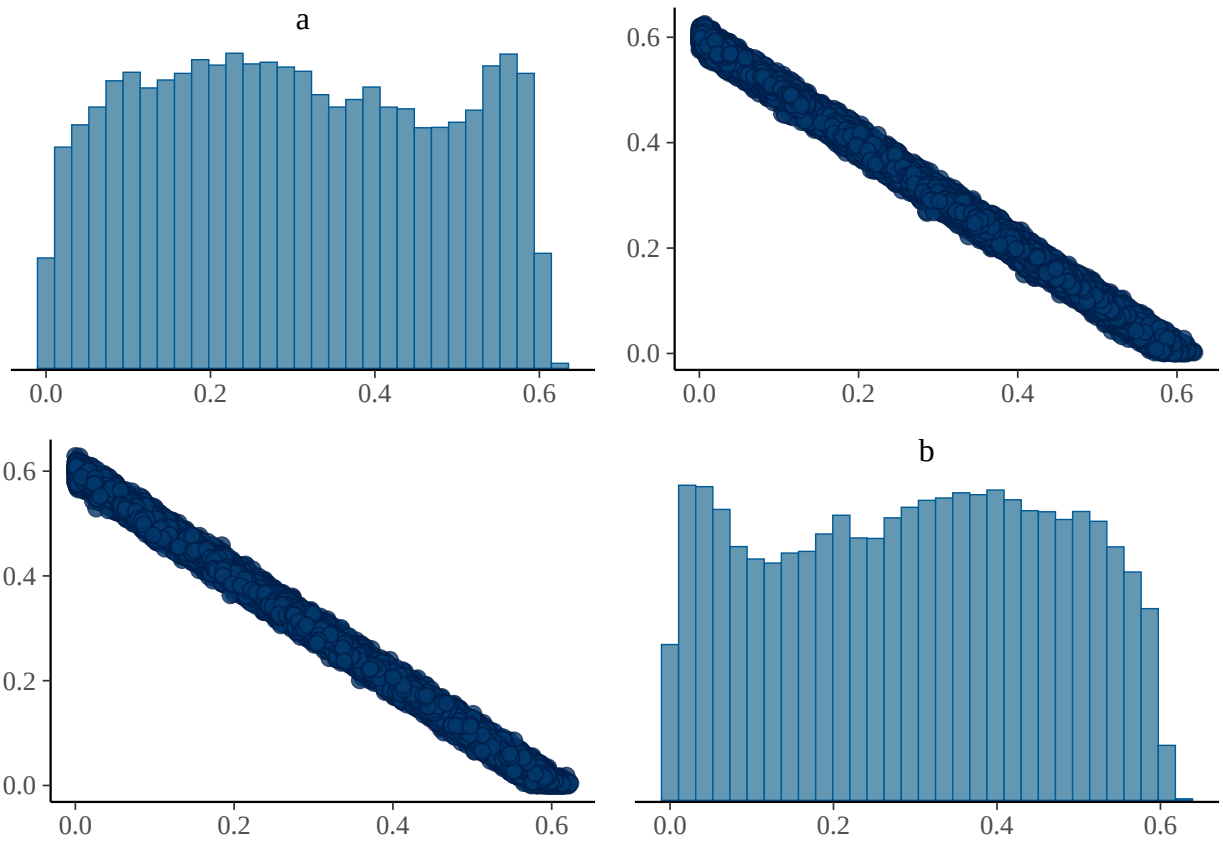
- (b) In all versions of this model a and b are correlated under the posterior distribution – since if one is big the other must be small, and vice-versa. However, when $\tau = 10000$, we are saying that $a + b$ must be almost exactly equal to y – this makes a and b extremely correlated, which vastly increases the number of iterations required to get reliable results – at least 40,000 ierations are needed, but even with this the effective sample size (ess) is small.

```
nasty_jagsmod <- "
model {
```

```
  y ~ dnorm(a + b, tau)
  a ~ dunif(0, 1)
  b ~ dunif(0, 1)
}"
nasty_dat <- list(y = 0.6, tau = 10000)
nasty_in <- list(list(a = 0.1, b = 0.5),
                 list(a = 0.5, b = 0.1))
nasty_jag <- jags.model(textConnection(nasty_jagsmod),
                        data=nasty_dat, inits=nasty_in, n.chains=2, quiet=TRUE)
pars_basic <- c("a", "b")
update(nasty_jag, 1000)
nasty_post <- coda.samples(nasty_jag, c(pars_basic), n.iter=40000)

library(bayesplot)
mcmc_trace(nasty_post, pars_basic)
```

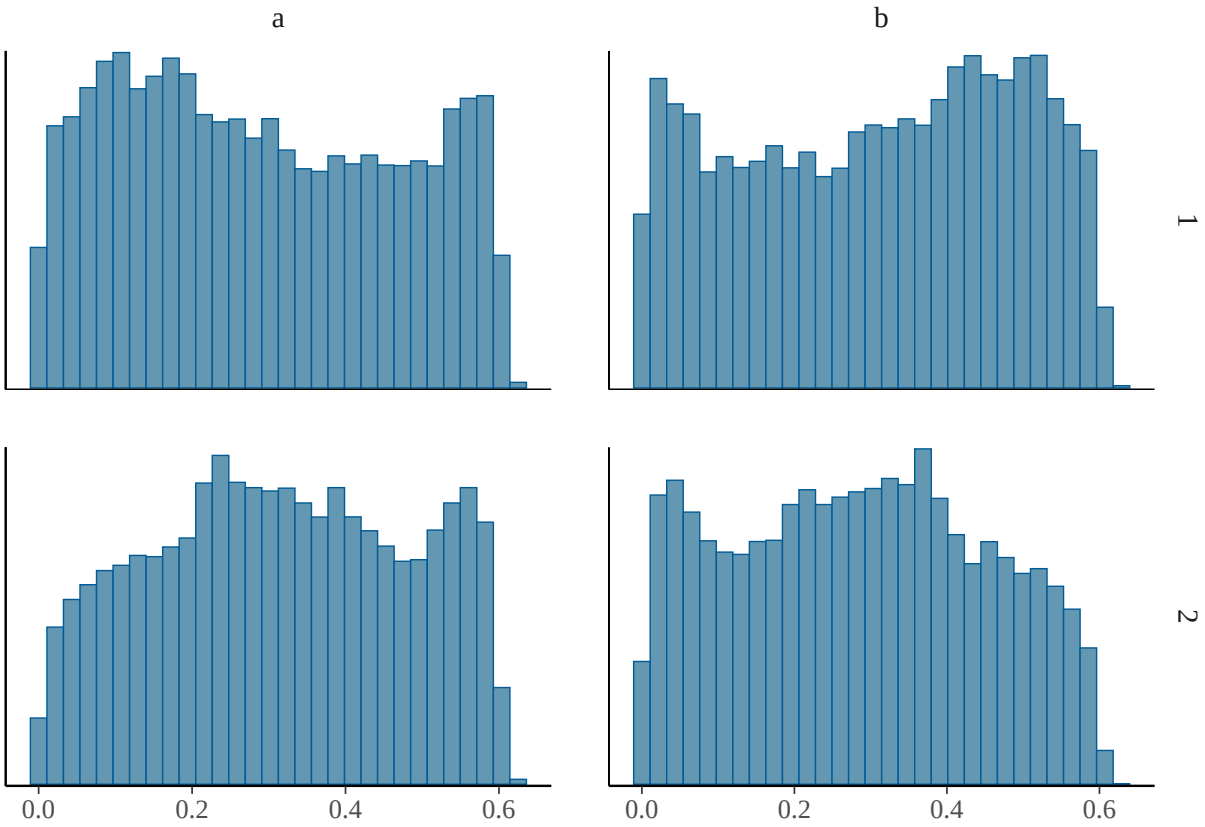



```
mcmc_pairs(nasty_post)
```




```
mcmc_hist_by_chain(nasty_post, pars_basic)
```

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




```
library(posterior)
assay_draws <- as_draws(nasty_post)
summary(assay_draws)
```

```
## # A tibble: 2 x 10
##   variable  mean median    sd   mad     q5   q95  rhat ess_bulk ess_tail
##   <chr>    <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 a        0.302  0.295 0.171 0.215 0.0374 0.574  1.01     94.1     465.
## 2 b        0.298  0.305 0.171 0.215 0.0266 0.562  1.01     94.1     461.
```

```
summary(assay_draws, default_convergence_measures())
```

```
## # A tibble: 2 x 4
##   variable  rhat ess_bulk ess_tail
##   <chr>    <dbl>    <dbl>    <dbl>
## 1 a         1.01     94.1     465.
## 2 b         1.01     94.1     461.
```

```
summary(assay_draws, default_mcse_measures())
```

```
## # A tibble: 2 x 6
##   variable mcse_mean mcse_median mcse_sd mcse_q5 mcse_q95
##   <chr>        <dbl>       <dbl>   <dbl>   <dbl>    <dbl>
## 1 a           0.0182      0.0263  0.0130 0.00595  0.00542
## 2 b           0.0182      0.0266  0.0129 0.00548  0.00591
```

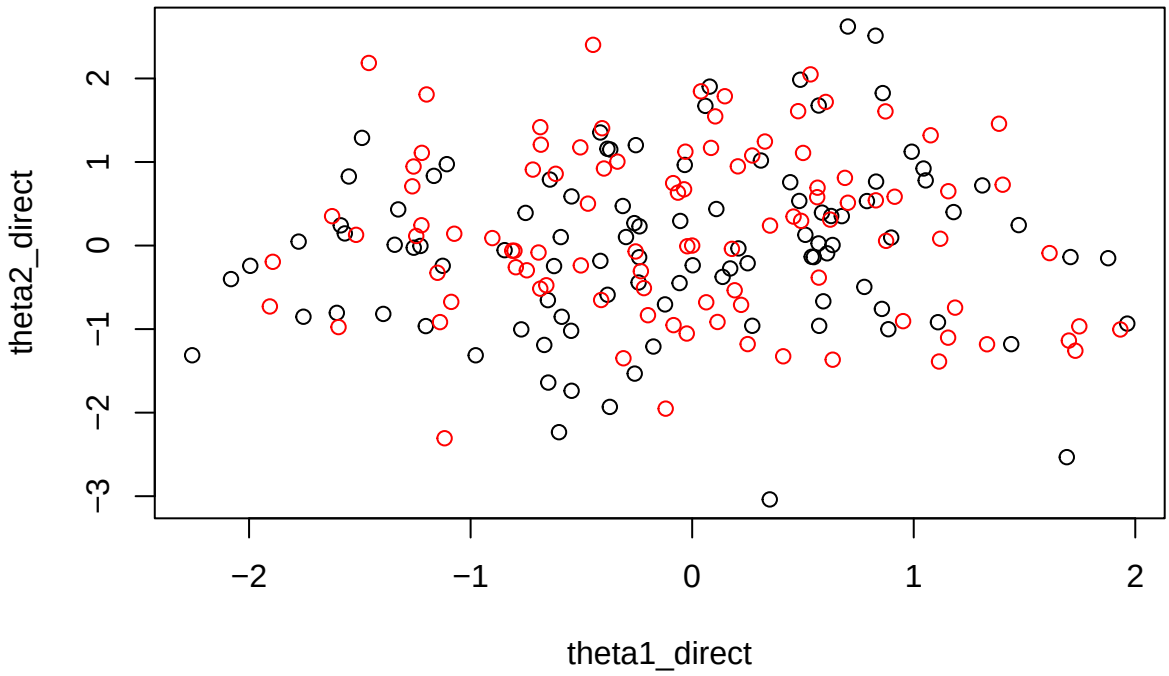
7.5 Two-stage Gibbs sampler in R

1.

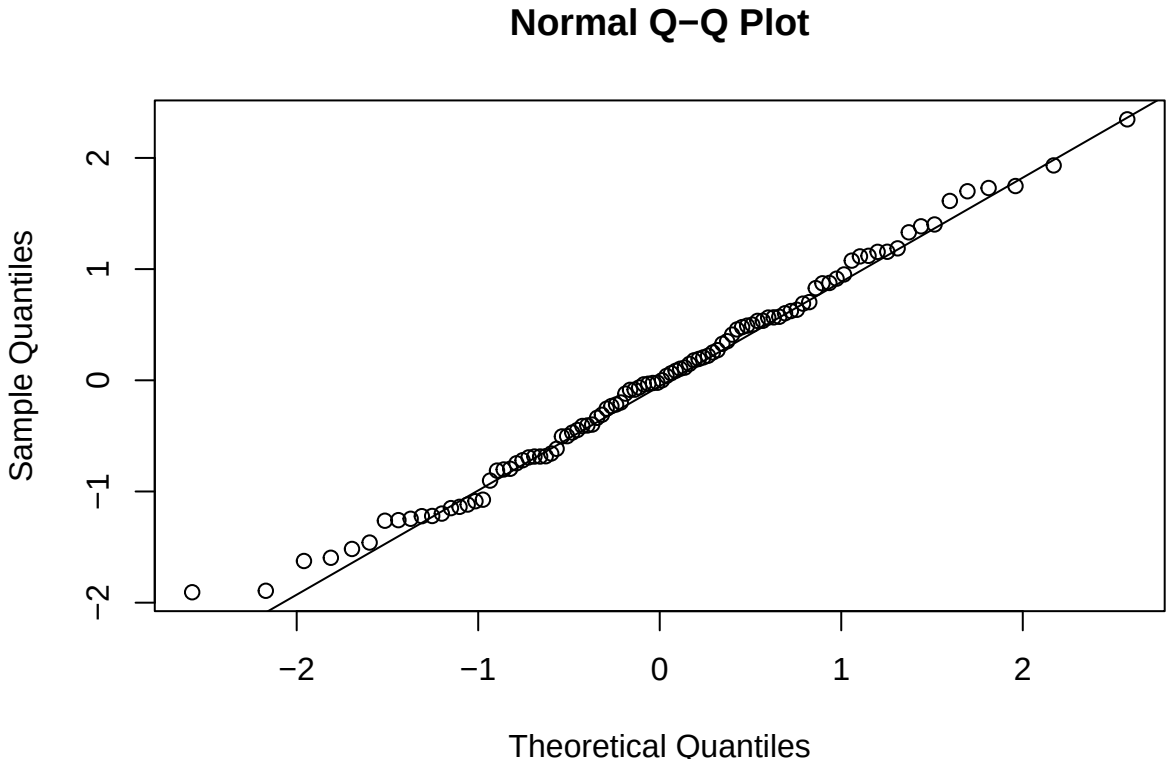
```
rho <- 0
niter <- 100
theta1 <- numeric(length = niter)
theta2 <- numeric(length = niter)
theta1[1] <- 0
theta2[1] <- 0
for (i in 2:niter){
  mean1 <- rho * theta2[i-1]
  variance <- 1- rho^2
  theta1[i] <- rnorm(1, mean = mean1, sd = sqrt(variance))
  mean2 <- rho * theta1[i]
  theta2[i] <- rnorm(1, mean = mean2, sd = sqrt(variance))
}
```

2.

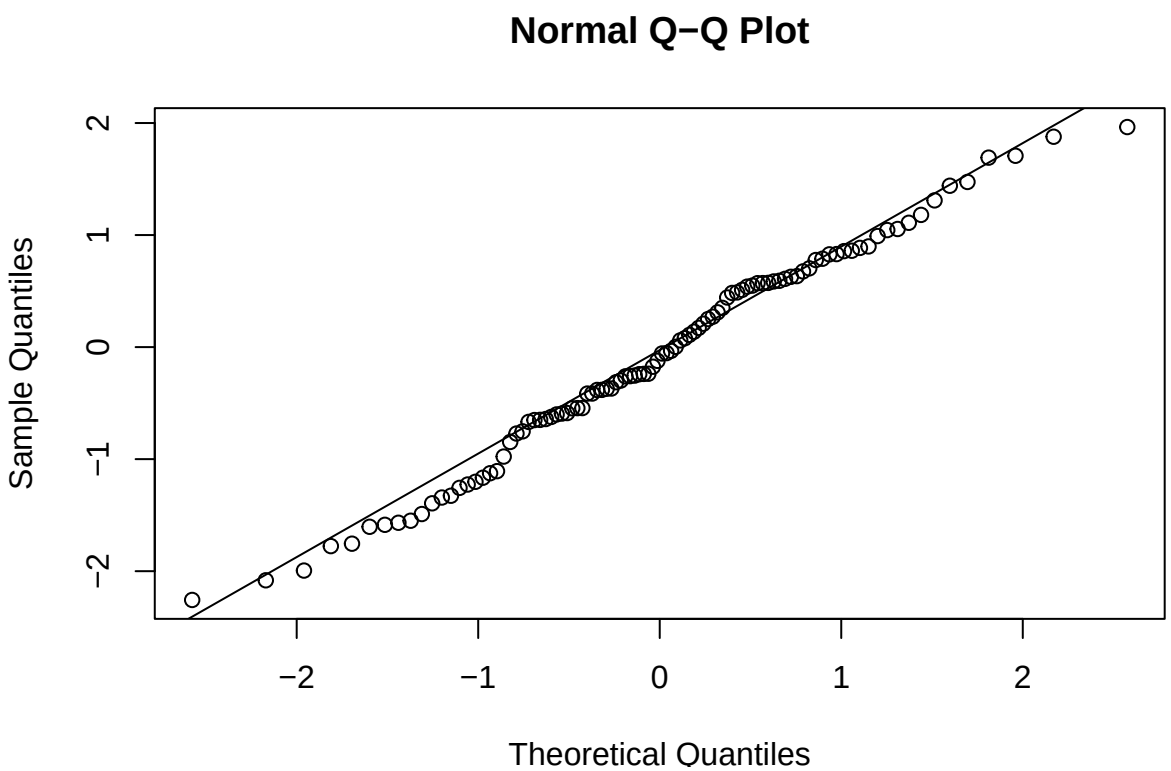
```
library(mvtnorm)

sigma <- matrix(c(1, rho, rho, 1), 2, 2)
theta_direct <- rmvnorm(niter, mean = c(0, 0), sigma = sigma)
theta1_direct <- theta_direct[, 1]
theta2_direct <- theta_direct[, 2]

plot(theta1_direct, theta2_direct)
#lines(theta1, theta2)
points(theta1, theta2, col = "red")
```




```
qqnorm(theta1)
qqline(theta1)
```



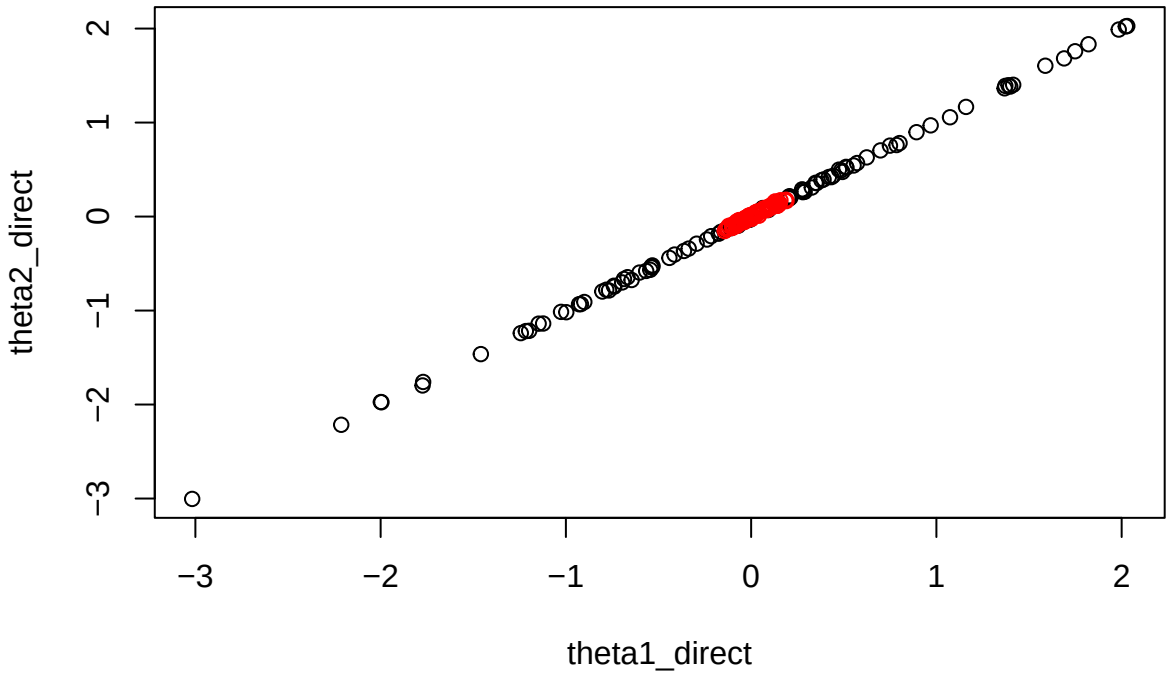

```
qqnorm(theta1_direct)
qqline(theta1_direct)
```



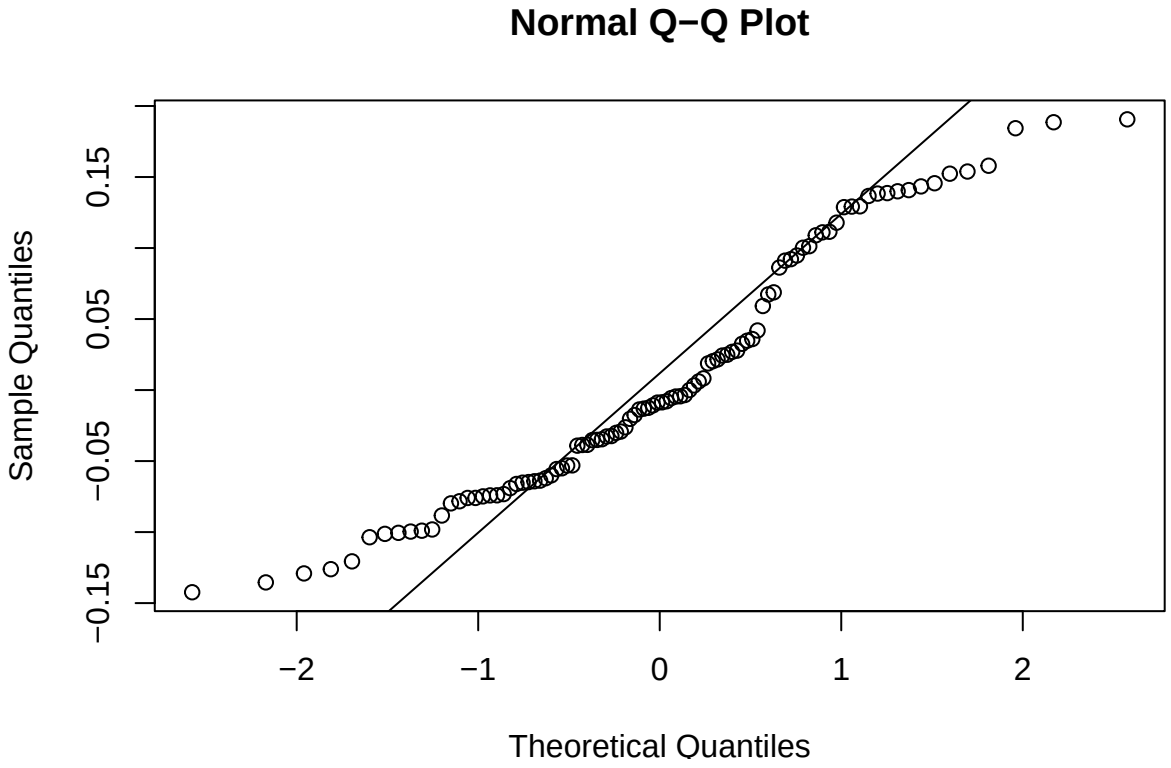
Comparing the results from the Gibbs sampler and `rmvnorm` results there is little apparent difference. Both QQ plots look good.

3.

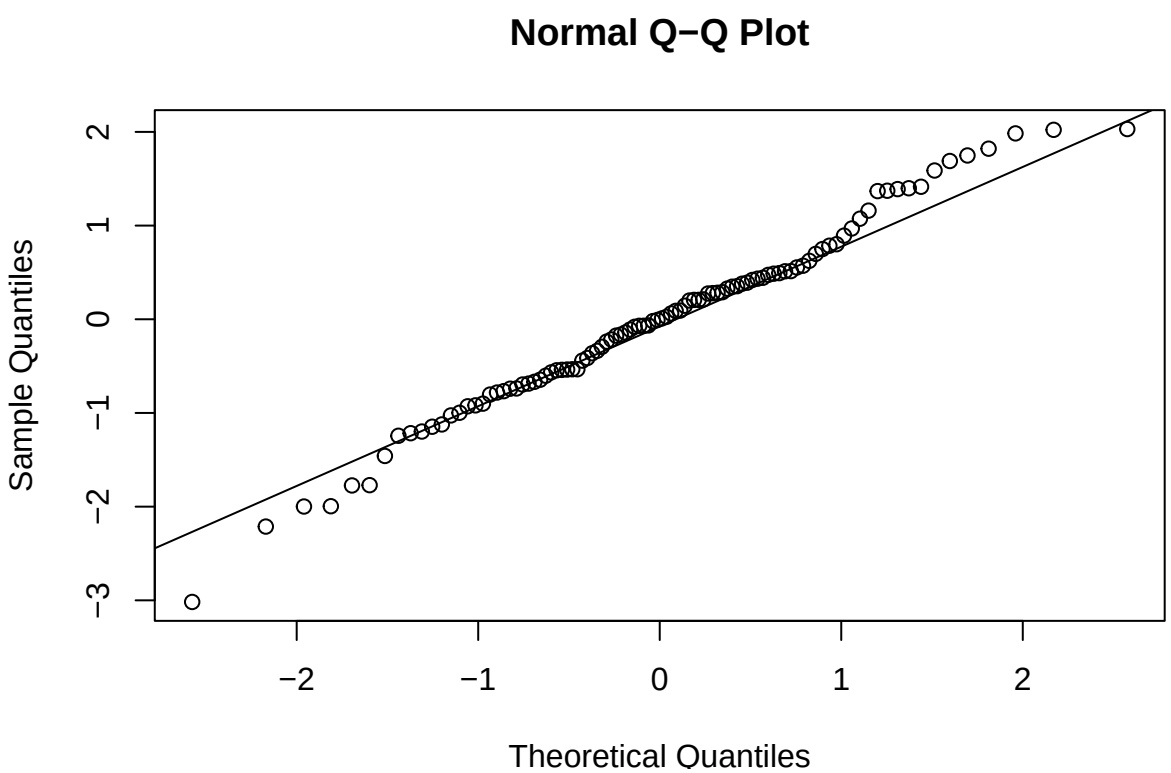
```r
rho <- 0.9999
niter <- 100
theta1 <- numeric(length = niter)
theta2 <- numeric(length = niter)
theta1[1] <- 0
theta2[1] <- 0
for (i in 2:niter){
  mean1 <- rho * theta2[i-1]
  variance <- 1- rho^2
  theta1[i] <- rnorm(1, mean = mean1, sd = sqrt(variance))
  mean2 <- rho * theta1[i]
  theta2[i] <- rnorm(1, mean = mean2, sd = sqrt(variance))
}

library(mvtnorm)
sigma <- matrix(c(1, rho, rho, 1), 2, 2)
theta_direct <- rmvnorm(niter, mean = c(0, 0), sigma = sigma)
theta1_direct <- theta_direct[, 1]
theta2_direct <- theta_direct[, 2]

plot(theta1_direct, theta2_direct)
#lines(theta1, theta2)
points(theta1, theta2, col = "red")
```




```r
qqnorm(theta1)
qqline(theta1)
```




```
qqnorm(theta1_direct)
qqline(theta1_direct)
```

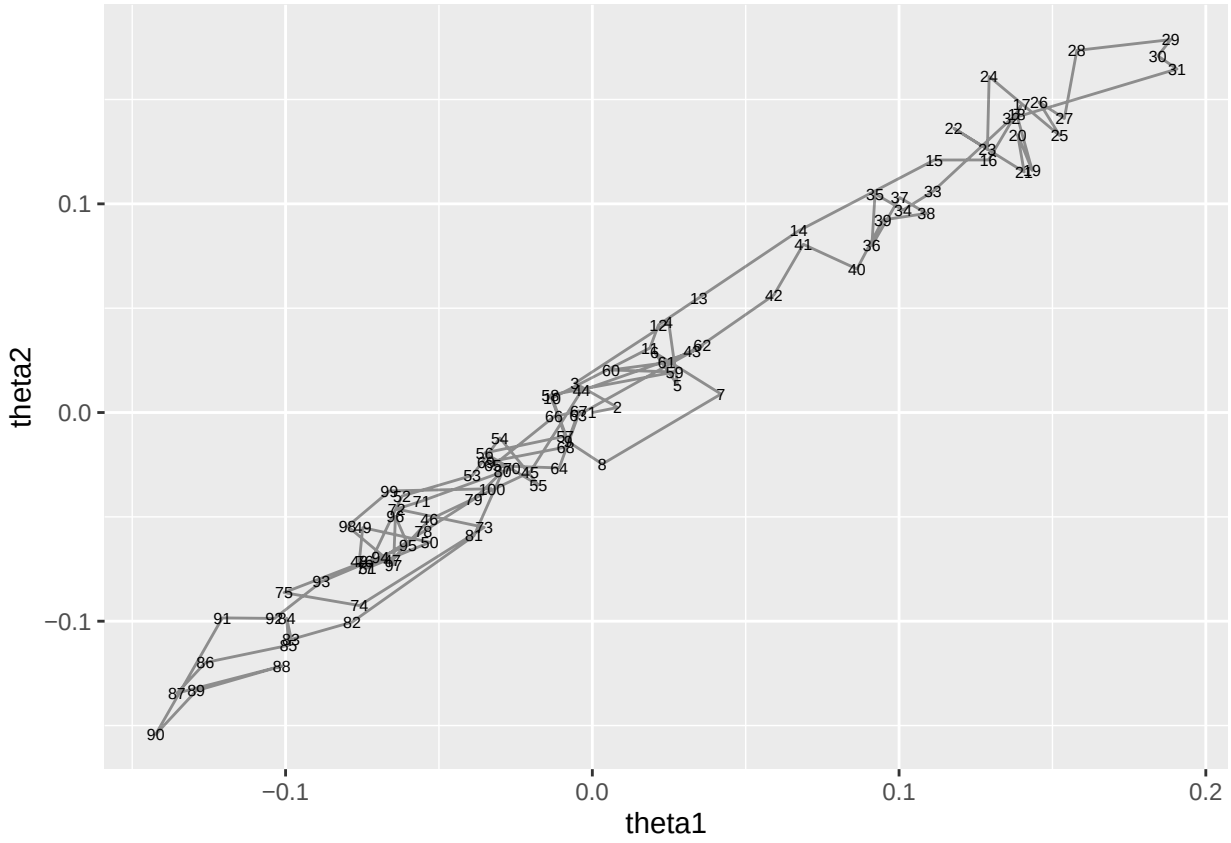


As we can see this in the QQ plots for the Gibbs sampler, the performance of the Gibbs sampler is less good now.

The reason for this is that when ρ close to 1, θ_1 and θ_2 are highly correlated. This means that the conditional distribution of θ_1, given θ_2 is tight (relative to the marginal distribution of θ_1), meaning that the θ_1 step

of the two-stage Gibbs sampler can not move freely: it has to stay near to its current value, meaning the effective sample size is low. The same is true for θ_2 given θ_1.

```
samples <- data.frame(theta1 = theta1,
                      theta2 = theta2,
                      iter = seq_along(theta1))
library(ggplot2)
ggplot(samples, aes(x = theta1, y = theta2, label = iter)) +
  geom_text(size = 2) +
  geom_path(alpha = 0.4)
```



Labelling each iteration with the iteration number, we can see that the sampler gets stuck in particular areas.

The main take-away message to remember is that Gibbs sampling can perform very poorly when two (or more) components of the target distribution are strongly correlated. The problem is particularly acute for Gibbs samplers (but it affects other MCMC samplers to varying degrees).

(Note the Gibbs sampler's performance could be fixed in this case by targetting a rotated the bivariate normal with its main directions of variation aligned with the x and y axes.)

7.6 Gibbs sampler for a normal with unknown mean and variance

```
niter <- 1000
y <- c(3, 5, 2, 3, 7)

a <- 2
b <- 3
psi2 <- 2^2
mu0 <- 3
```

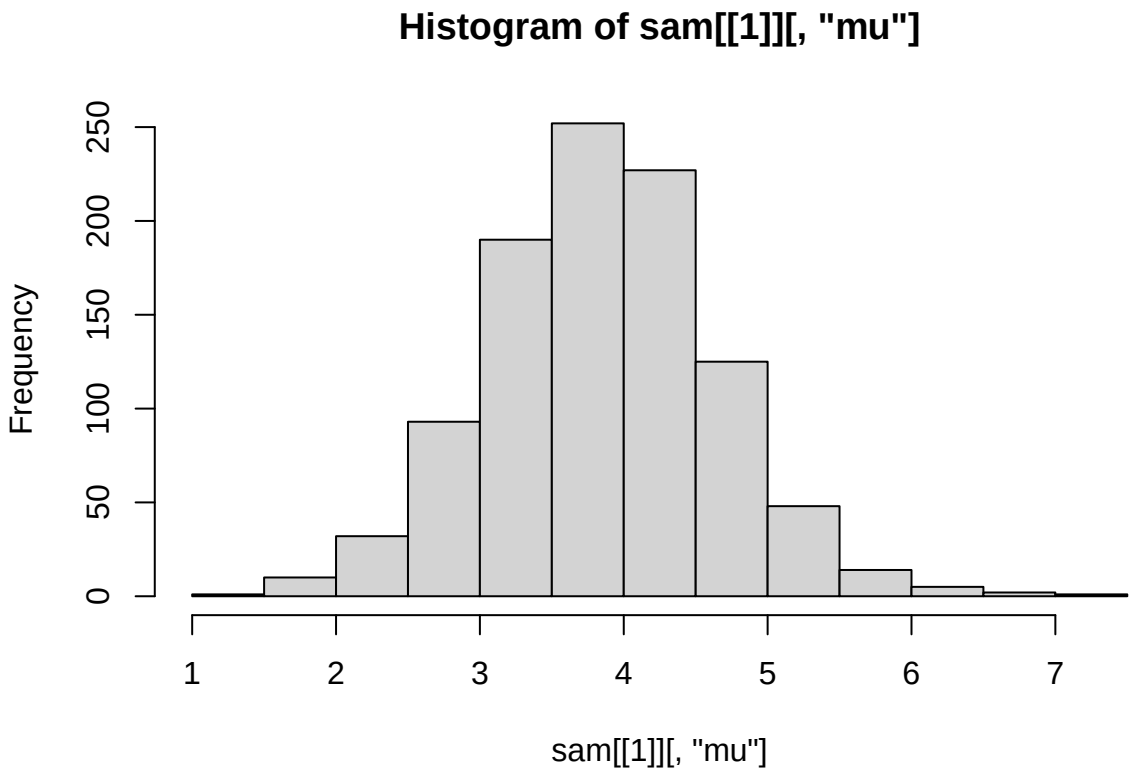
```r
n <- length(y)
ybar <- mean(y)
a2 <- (n/2) + a
sigma2 <- numeric(length = niter)
mu <- numeric(length = niter)
# draw from prior
sigma2[1] <- 1
mean <- 0
ratio <- sigma2[1]/(sigma2[1] + n * psi2)
sd <- sqrt(psi2 * ratio)
mu[1] <- rnorm(1, mean = mean,sd = sd)
for (i in 2:niter){
  ratio <- sigma2[i - 1]/(sigma2[i - 1] + n * psi2)
  mean <- ratio * mu0 + (1 - ratio) * ybar
  sd <- sqrt(psi2 * ratio)
  mu[i] <- rnorm(1, mean = mean,sd = sd)
  b2 <- (1/2)*(sum((y - mu[i])^2)) + b
  sigma2[i] <- 1/rgamma(1, shape = a2, rate = b2)
}

summary(mu)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.3385  3.3705  3.8721  3.8625  4.3821  6.5238
```

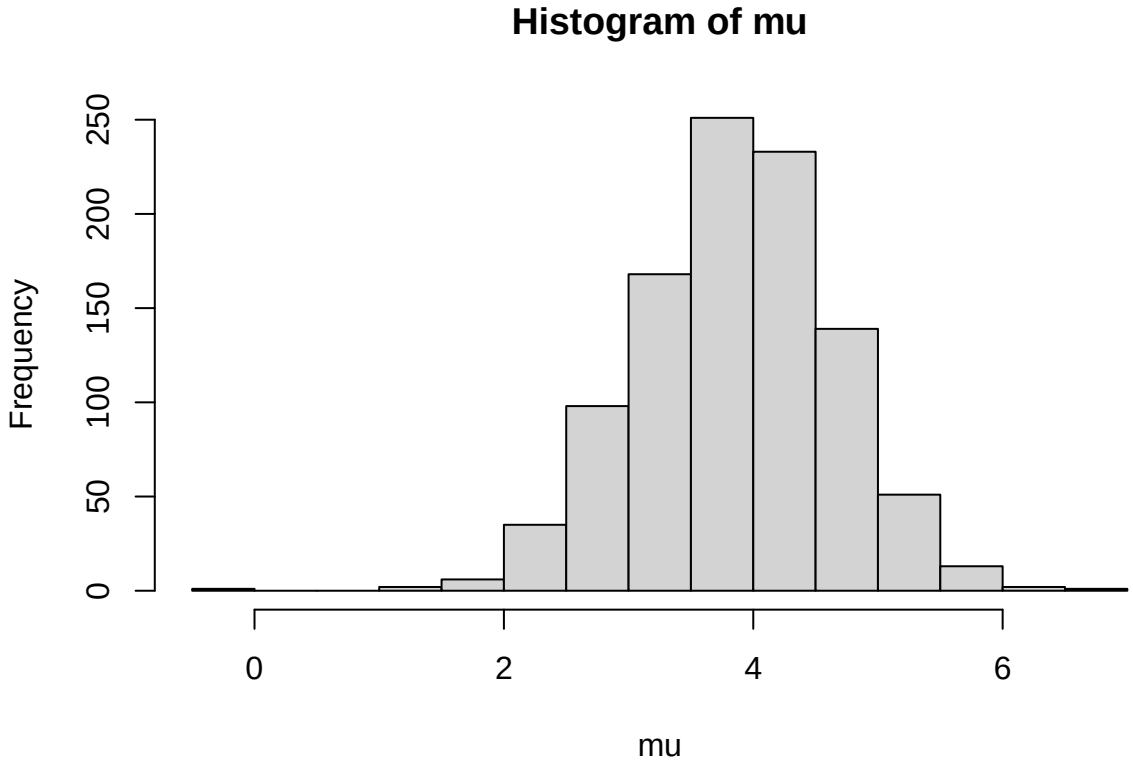
```r
summary(sigma2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8214  2.1045  2.9598  3.7137  4.3548 43.3291
```

```r
# Also implement in JAGS
norm_jagsmod <- "
model {
  for (i in 1:n){
    y[i] ~ dnorm(mu, sigma2_inv)
  }
  mu ~ dnorm(mu0, psi2_inv)
  psi2_inv <- 1/psi2
  sigma2_inv ~ dgamma(a, b)
}"
dat <- list(y = y, n = n, a = a, b = b, mu0 = mu0, psi2 = psi2)
norm_in <- list(list(mu = 0, sigma2_inv = 1),
                list(mu = 1, sigma2_inv = 2))
library(rjags)
norm_jag <- jags.model(textConnection(norm_jagsmod),
                       data=dat, inits=norm_in, n.chains=2, quiet=TRUE)
pars_basic <- c("mu", "sigma2_inv")
update(norm_jag, 1000)
sam <- coda.samples(norm_jag, c(pars_basic), n.iter=1000)

hist(sam[[1]][, "mu"])
```


Histogram of sam[[1]][, "mu"]
Frequency
0
50
100
150
200
250
1
2
3
4
5
6
7
sam[[1]][, "mu"]

```
hist(mu)
```


Histogram of mu
Frequency
0
50
100
150
200
250
0
2
4
6
mu