Assessing convergence of Bayesian MCMC

Sebastian Duchene

May 4 2014

This is an example of MCMC runs with different proposal functions. With different proposal functions the MCMC will need to be tuned to obtain sufficient samples to estimate the parameters. The examples bellow illustrate different scenarios and suggest how to tune the analysis to improve the MCMC.

Define some code to set up the MCMC

Load the Coda package to estimate the Effective Sample Size.

```
library(coda)
```

Define some data. We will estimate the mean and standard deviation of these data which were sampled from a normal distribution with mean = 10 and sd = 2. The exact values may vary because of the nature of the simulation and because we are only using 100 samples. In practice, it is preferable to have more data so that the prior has a lower influence on the posterior.

```
set.seed(18002262)
x <- rnorm(100, 10, 2)
```

Although this example is trivial, it serves to illustrate the effectiveness of the MCMC. In particular, we can determine the true values of the parameters of interest empirically. In this case we are estimating the mean and the standard deviation. These will be the "expected" values.

```
mean(x)
```

```
## [1] 9.963
```

```
sd(x)
```

```
## [1] 1.808
```

Next, we define a likelihood function based on a model. In this case the model is a normal distribution, which matches that used to generate the data. Note that we have defined the model, and that we are interested in the parameters of the model (mean and standard deviation). We can test different models, such as different probability distributuons, but this is beyond the scope of this example.

```
likelihood <- function(param) {
    mean.model <- param[1]
    sd.model <- param[2]
    single.likelihoods <- dnorm(x, mean = mean.model, sd = sd.model, log = T)
    sum.likelihoods <- sum(single.likelihoods)
    return(sum.likelihoods)
}
```

Define the prior. We will use a prior with low information content, such as a uniform distribution for both parameters, for the mean it is bounded between 0 and 15 and for the standard deviation it is bounded at 0 and 5.

```r
prior <- function(param) {
    mean.param <- param[1]
    sd.param <- param[2]
    # Some very uninformative priors
    mean.prior <- dunif(mean.param, 0, 15, log = T)
    sd.prior <- dunif(sd.param, 0, 5, log = T)
    return(mean.prior + sd.prior)
}
```

Now define a proposal function. This determines how the MCMC will move around the parameter space. With an optimal proposal function the MCMC should converge to the true parameter values very easily, but in practice this is very difficult to define, and MCMC analyses need tuning to obtain reliable estimates.

```r
proposal.function <- function(param) {
    return(c(rnorm(1, mean = param[1], sd = 0.5), abs(rnorm(1, mean = param[2],
        sd = 0.5))))  # the abs is so that there is to avoid negative proposals for the starndard devia
}
```

The last function is the calculation of the posterior, which is the product of the likelihood and the prior. In this example we add them because we are using log transformations.

```r
posterior <- function(param) {
    return(likelihood(param) + prior(param))
}
```

Finally, we define a simple MCMC with the functions above, using the Metropolis-Hastings algorithm. This MCMC samples every step, but it could be modified to reduce the sampling frequency.

```r
mcmc_bayesian <- function(startvals, iterations = 50000) {
    chain <- matrix(NA, nrow = iterations, ncol = 5)
    colnames(chain) <- c("mean", "sd", "likelihood", "posterior", "prior")
    chain[1, ] <- c(startvals, likelihood(startvals), posterior(startvals),
        prior(startvals))
    for (i in 2:iterations) {
        # Uncomment the line bellow to print the progress of the MCMC on screen
        # print(paste('iteration', i , 'of', iterations, 'the parameters are',
        # paste(chain[i-1, ], collapse = ' ')))
        proposal <- proposal.function(chain[i - 1, 1:2])
        prob.accept <- exp(posterior(proposal) - posterior(chain[i - 1, 1:2]))
        if (runif(1) < prob.accept) {
            chain[i, ] <- c(proposal, likelihood(proposal), posterior(proposal),
                prior(proposal))
        } else {
            chain[i, ] <- chain[i - 1, ]
        }
    }
    return(chain)
}
```

Example 1

In this example we use the original proposal function and run the MCMC for 10000 steps, as before.

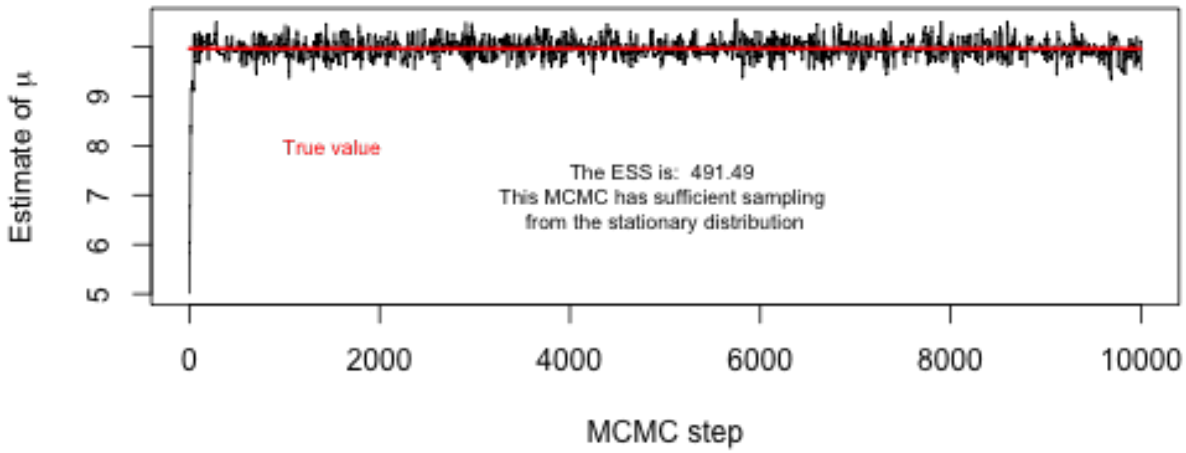
```
ch1 <- mcmc_bayesian(startvals = c(5, 3), iterations = 10000)
```

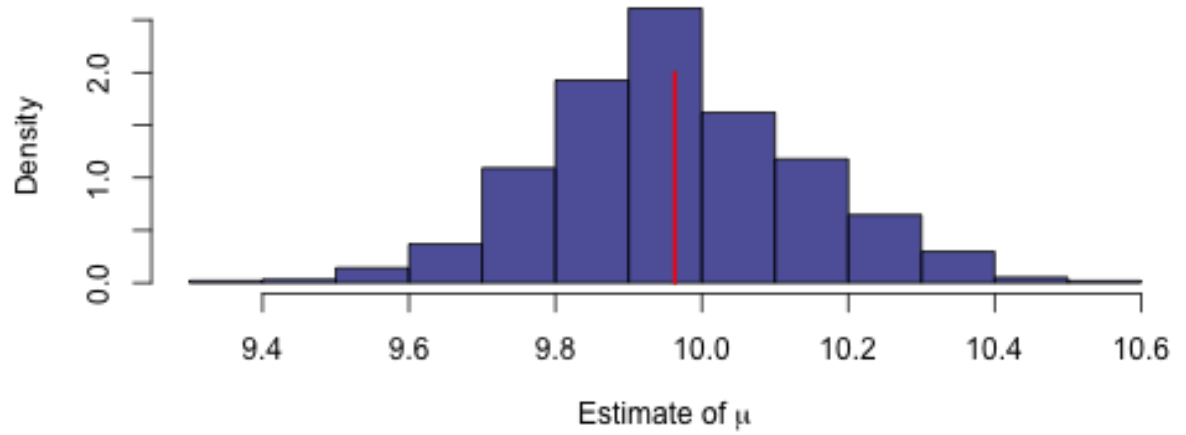
Plot some diagnostics of *Example 1*

```
rem_burnin <- 100:nrow(ch1)
par(mfrow = c(2, 1))
plot(ch1[, 1], type = "l", xlab = "MCMC step", ylab = expression(paste("Estimate of ",
    mu)), main = "Fig 1A.")
lines(x = 1:10000, y = rep(mean(x), 10000), col = "red", lwd = 2)
text(5000, 7, labels = paste("The ESS is: ", round(effectiveSize(mcmc(ch1[,
    1])), 2), "\nThis MCMC has sufficient sampling \nfrom the stationary distribution"),
    cex = 0.7)
text(1500, 8, labels = "True value", col = "red", cex = 0.7)

hist(ch1[rem_burnin, 1], col = rgb(0, 0, 0.5, 0.7), xlab = expression(paste("Estimate of ",
    mu)), main = "", freq = F)
lines(x = c(mean(x), mean(x)), y = c(0, 2), col = "red", lwd = 2)
```

Fig 1A.



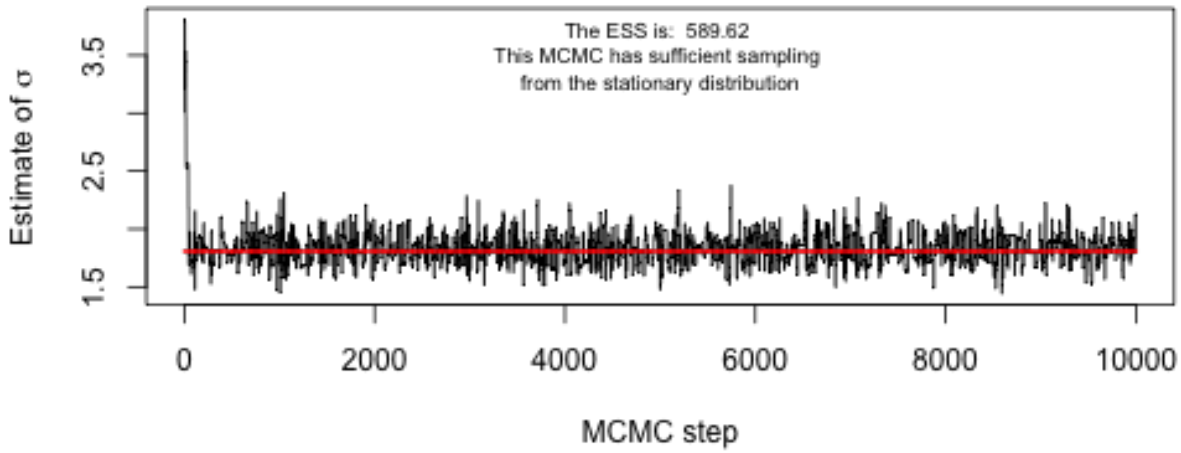


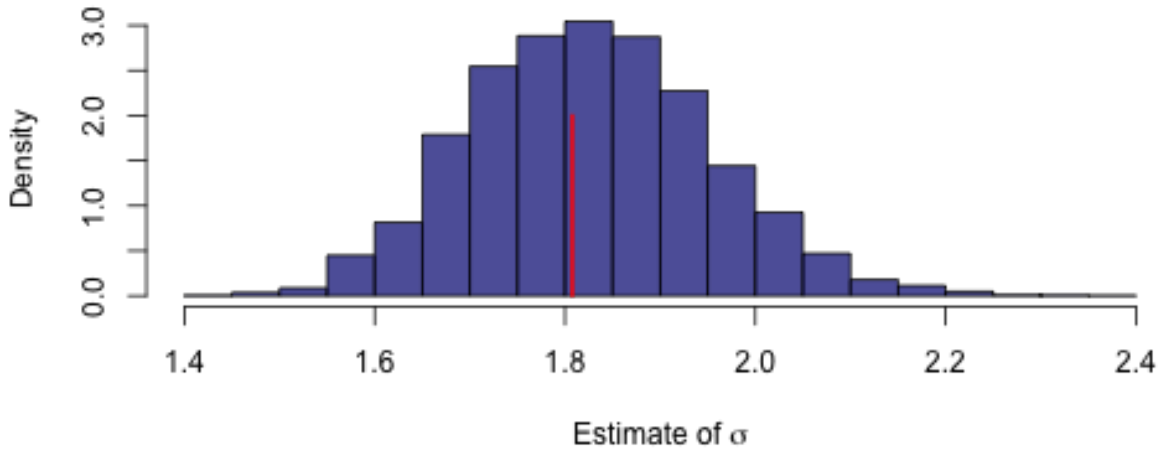

```
par(mfrow = c(2, 1))
plot(ch1[, 2], type = "l", xlab = "MCMC step", ylab = expression(paste("Estimate of ",
    sigma)), main = "Fig 1B.")
lines(x = 1:10000, y = rep(sd(x), 10000), col = "red", lwd = 2)

text(5000, 3.5, labels = paste("The ESS is: ", round(effectiveSize(mcmc(ch1[,
    2])), 2), "\nThis MCMC has sufficient sampling \nfrom the stationary distribution"),
    cex = 0.7)

hist(ch1[rem_burnin, 2], col = rgb(0, 0, 0.5, 0.7), xlab = expression(paste("Estimate of ",
    sigma)), main = "", freq = F)
lines(x = c(sd(x), sd(x)), y = c(0, 2), col = "red", lwd = 2)
```

Fig 1B.





This is an ideal run. The MCMC has converged to the stationary distribution, there is low autocorrelation for MCMC samples, the parameter estimates appear accurate and precise, and we have ESS > 200. Importantly, the true value is contained within the parameter estimates (this is unknown in real data sets).

Example 2

We can use a sloppy proposal function that will make large moves around the parameter space, resulting in inefficent sampling. This can be typically overcome by running longer chains.

```
proposal.function <- function(param) {
    return(c(rnorm(1, mean = param[1], sd = 5), abs(rnorm(1, mean = param[2],
        sd = 5))))  # the abs is so that there is to avoid negative proposals for the starndard deviati
}
```

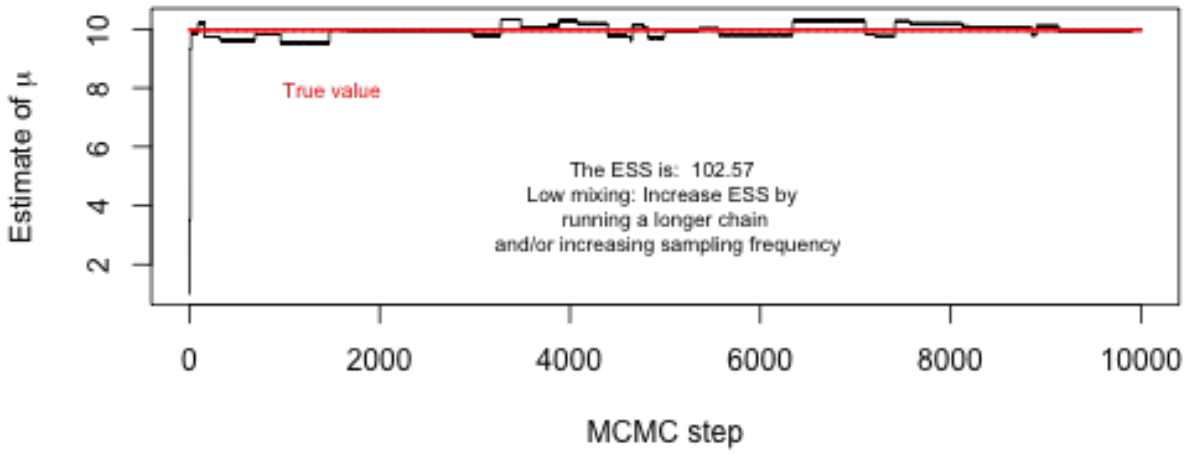
Run the MCMC and plot the run.

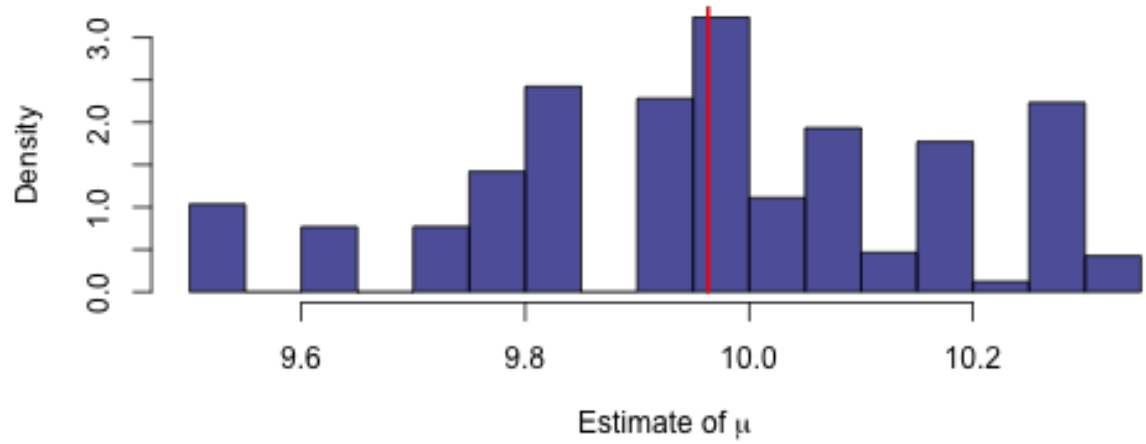
```
ch2 <- mcmc_bayesian(startvals = c(1, 3), iterations = 10000)
```

```
par(mfrow = c(2, 1))
rem_burnin <- 100:nrow(ch2)
plot(ch2[, 1], type = "l", xlab = "MCMC step", ylab = expression(paste("Estimate of ",
    mu)), main = "Fig 2A.")
lines(x = 1:10000, y = rep(mean(x), 10000), col = "red", lwd = 2)
text(5000, 4, labels = paste("The ESS is: ", round(effectiveSize(mcmc(ch2[,
    1])), 2), "\nLow mixing: Increase ESS by \nrunning a longer chain\n and/or increasing sampling freq
    cex = 0.7)
text(1500, 8, labels = "True value", col = "red", cex = 0.7)

hist(ch2[rem_burnin, 1], col = rgb(0, 0, 0.5, 0.7), xlab = expression(paste("Estimate of ",
    mu)), main = "", freq = F)
lines(x = c(mean(x), mean(x)), y = c(0, 4), col = "red", lwd = 2)
```

Fig 2A.



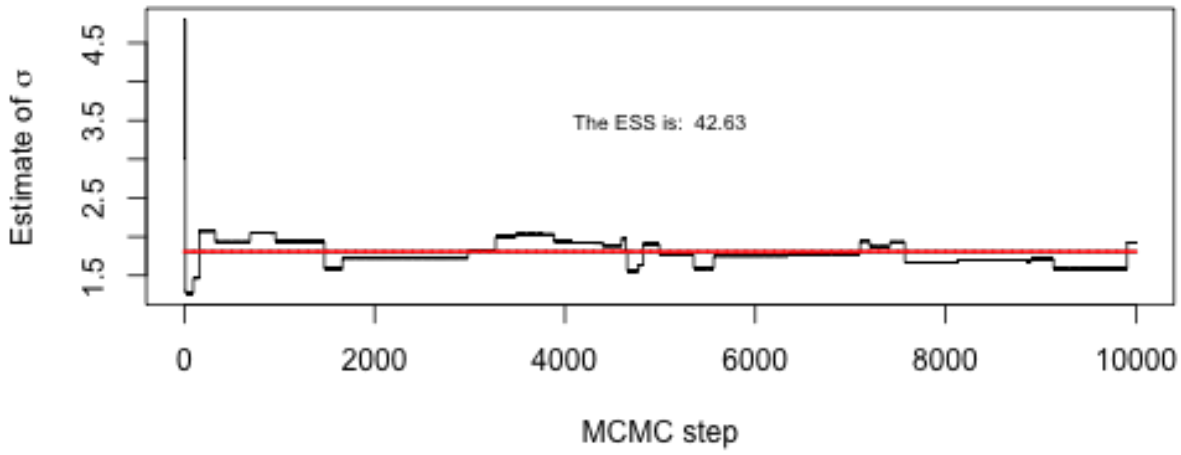


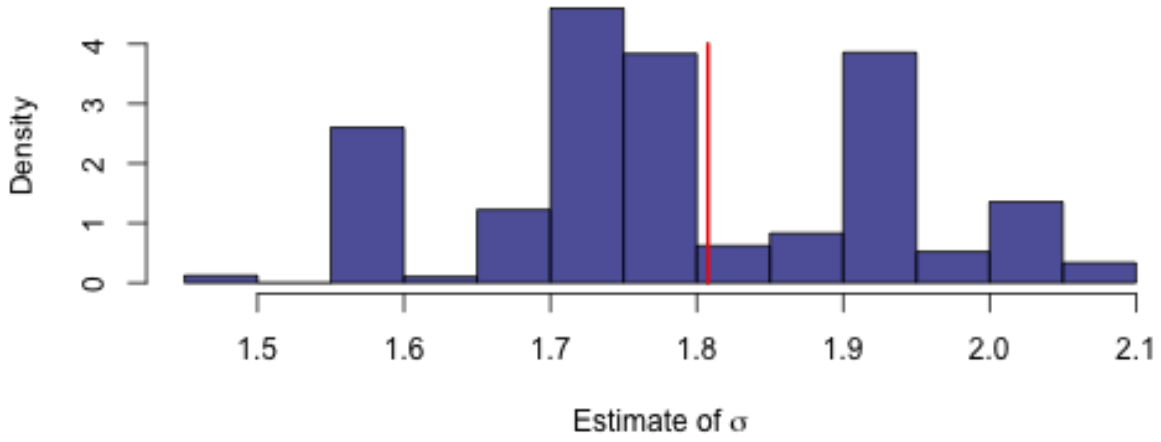

```
par(mfrow = c(2, 1))
plot(ch2[, 2], type = "l", xlab = "MCMC step", ylab = expression(paste("Estimate of ",
    sigma)), main = "Fig 2B.")
lines(x = 1:10000, y = rep(sd(x), 10000), col = "red", lwd = 2)

text(5000, 3.5, labels = paste("The ESS is: ", round(effectiveSize(mcmc(ch2[,
    2])), 2)), cex = 0.7)

hist(ch2[rem_burnin, 2], col = rgb(0, 0, 0.5, 0.7), xlab = expression(paste("Estimate of ",
    sigma)), main = "", freq = F)
lines(x = c(sd(x), sd(x)), y = c(0, 4), col = "red", lwd = 2)
```

Fig 2B.





It appears that the MCMC has reached the stationary distribution, but it is not moving eficiently around the parameter space, resulting in low ESS. This can be easily solved by increasing the chain length. Sometimes it is also helpful to increase the sampling frequency in analyses where this can be modified.

Example 3

Now we will modify the proposal function to make very small moves. This will result in inefficient sampling, and difficulty in finding the stationary distribution.

```
proposal.function <- function(param) {
    return(c(rnorm(1, mean = param[1], sd = 0.01), abs(rnorm(1, mean = param[2],
        sd = 0.01))))  # the abs is so that there is to avoid negative proposals for the starndard devi
}
```

As before, run the MCMC:

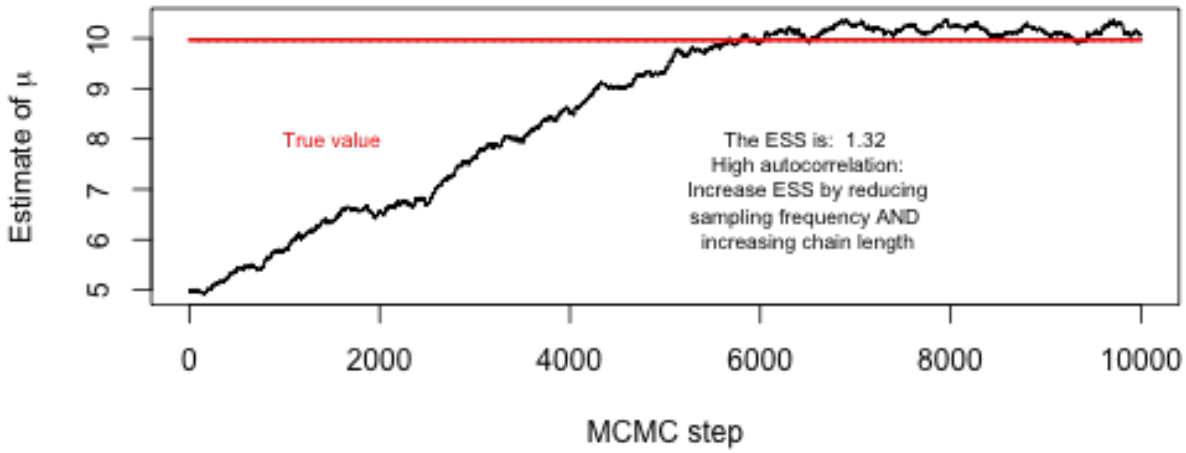
```
ch3 <- mcmc_bayesian(startvals = c(5, 3), iterations = 10000)
```

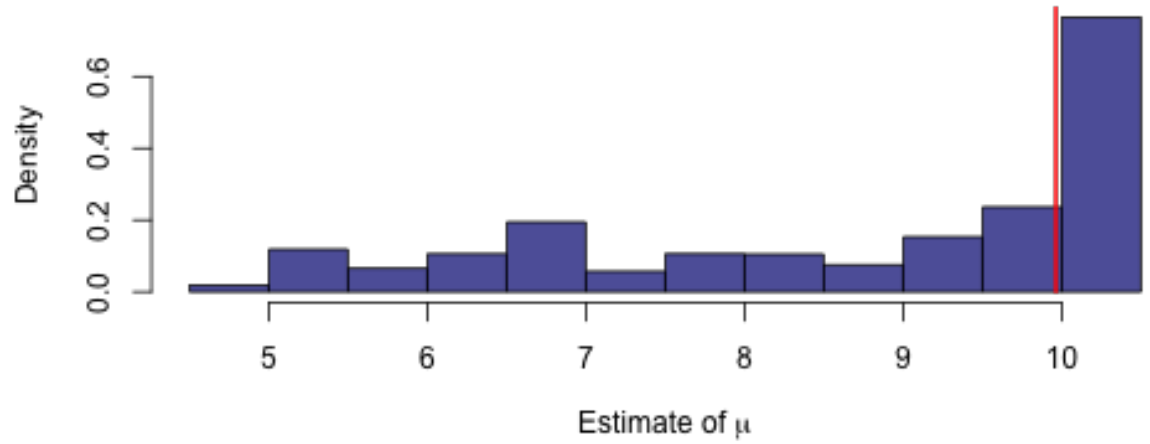
Produce similar plots to those of the previous examples:

```
par(mfrow = c(2, 1))
rem_burnin <- 100:nrow(ch3)
plot(ch3[, 1], type = "l", xlab = "MCMC step", ylab = expression(paste("Estimate of ",
    mu)), main = "Fig 3A.")
lines(x = 1:10000, y = rep(mean(x), 10000), col = "red", lwd = 2)
text(6500, 7, labels = paste("The ESS is: ", round(effectiveSize(mcmc(ch3[,
    1])), 2), "\nHigh autocorrelation:\n Increase ESS by reducing \nsampling frequency AND \nincreasing
    cex = 0.7)
text(1500, 8, labels = "True value", col = "red", cex = 0.7)

hist(ch3[rem_burnin, 1], col = rgb(0, 0, 0.5, 0.7), xlab = expression(paste("Estimate of ",
    mu)), main = "", freq = F)
lines(x = c(mean(x), mean(x)), y = c(0, 4), col = "red", lwd = 2)
```

Fig 3A.



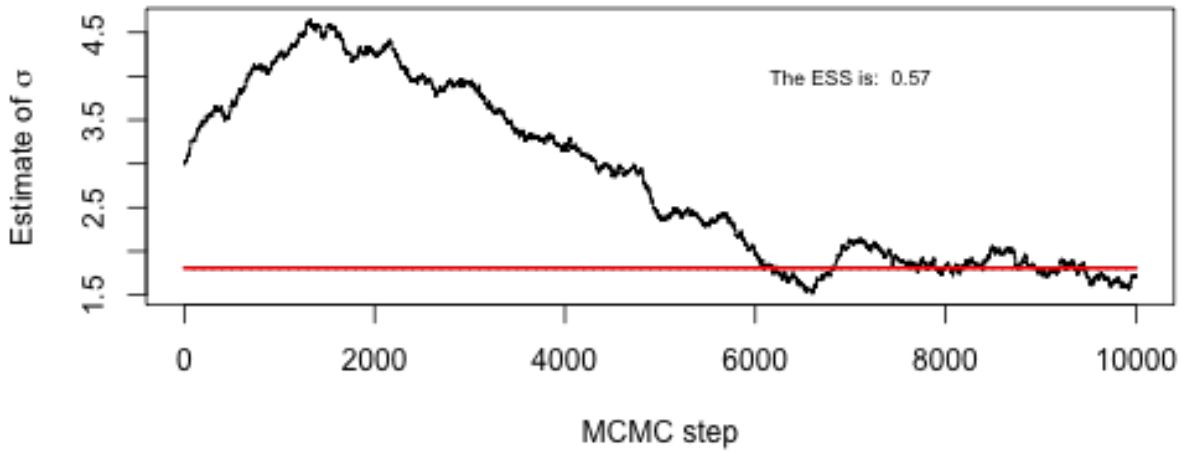


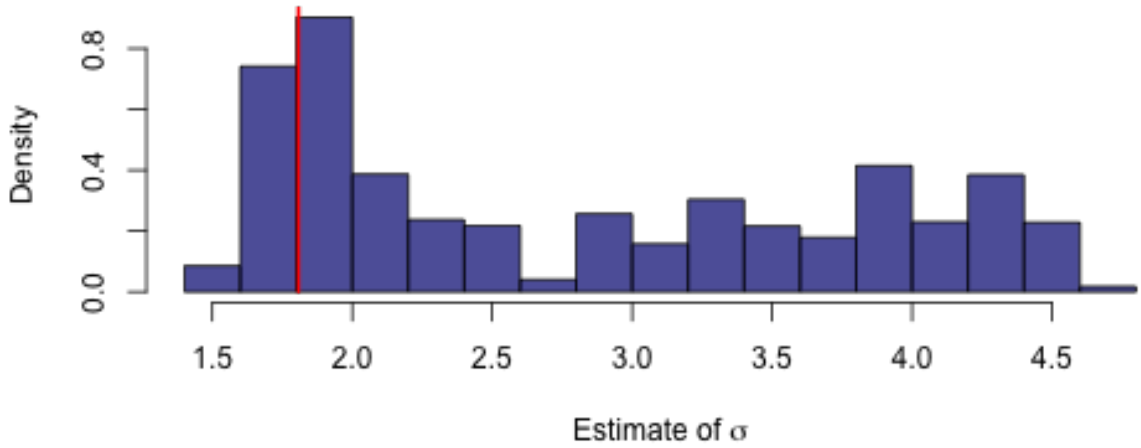

```
par(mfrow = c(2, 1))
plot(ch3[, 2], type = "l", xlab = "MCMC step", ylab = expression(paste("Estimate of ",
    sigma)), main = "Fig 3B.")
lines(x = 1:10000, y = rep(sd(x), 10000), col = "red", lwd = 2)

text(7000, 4, labels = paste("The ESS is: ", round(effectiveSize(mcmc(ch3[,
    2])), 2)), cex = 0.7)

hist(ch3[rem_burnin, 2], col = rgb(0, 0, 0.5, 0.7), xlab = expression(paste("Estimate of ",
    sigma)), main = "", freq = F)
lines(x = c(sd(x), sd(x)), y = c(0, 4), col = "red", lwd = 2)
```

Fig 3B.





The MCMC is moving frequently around the parameter space, but these moves are ineffcient in that they do not find the stationary distribution. There is a high level of autocorrelation between steps, resulting in low ESS. As in *Example 2*, the solution is to increase the MCMC chain length, but in this case it is also useful to reduce the sampling frequency to avoid very large output files.