

# Gibbs sampler and MCMC (R scripts)

Jingchen (Monika) Hu

MATH 347 Bayesian Statistics

## Installing the necessary packages

```
install.packages("devtools")
require(devtools)
devtools::install_github("bayesball/ProbBayes")

require(ggplot2)
require(gridExtra)
require(ProbBayes)
require(tidyverse)
crcblue <- "#2905a1"
```

## Example: Expenditures in the Consumer Expenditure Surveys

The TOTEXPPQ variable in the CE sample

```
CEsample <- read_csv("CEsample1.csv")
```

```
summary(CEsample$TotalExpLastQ)
```

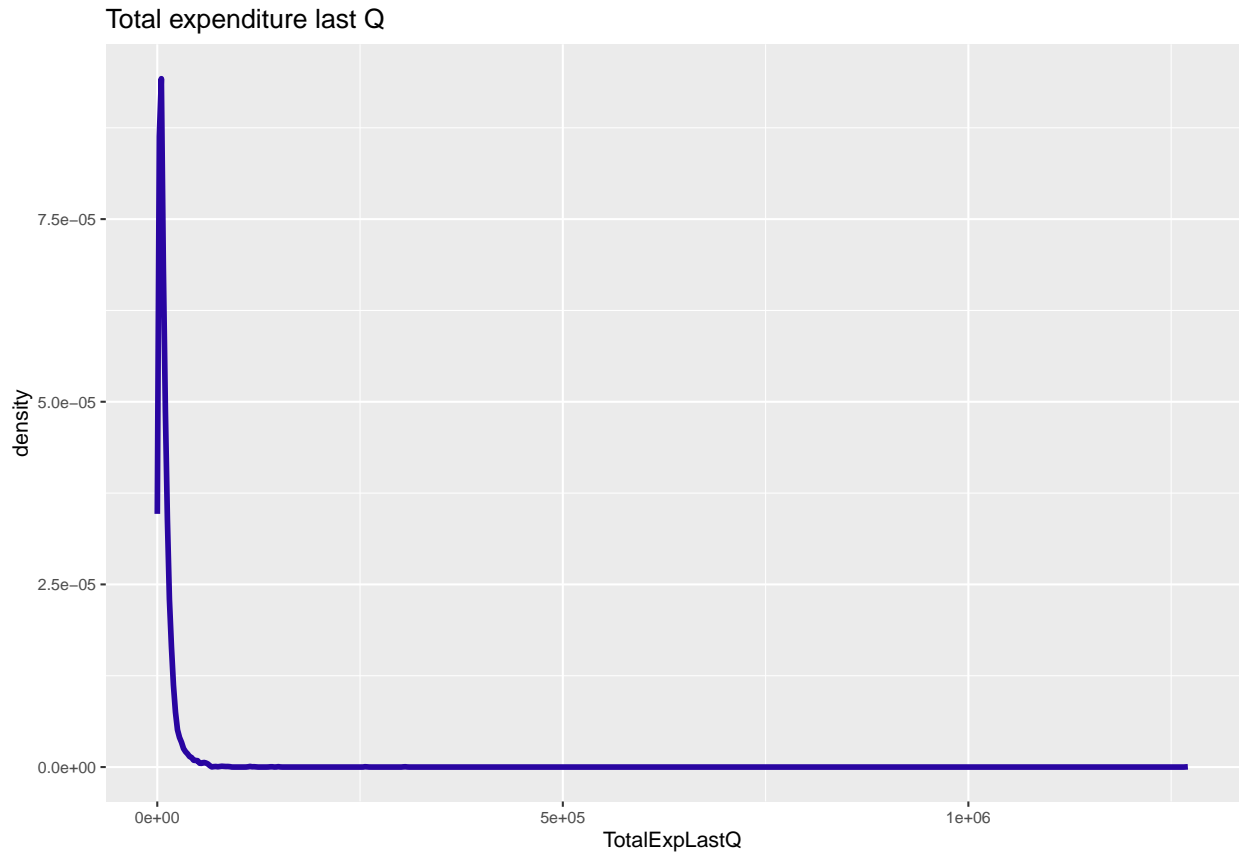
```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       30   3522    6417    9513   11450 1270598
```

```
sd(CEsample$TotalExpLastQ)
```

```
## [1] 19341.25
```

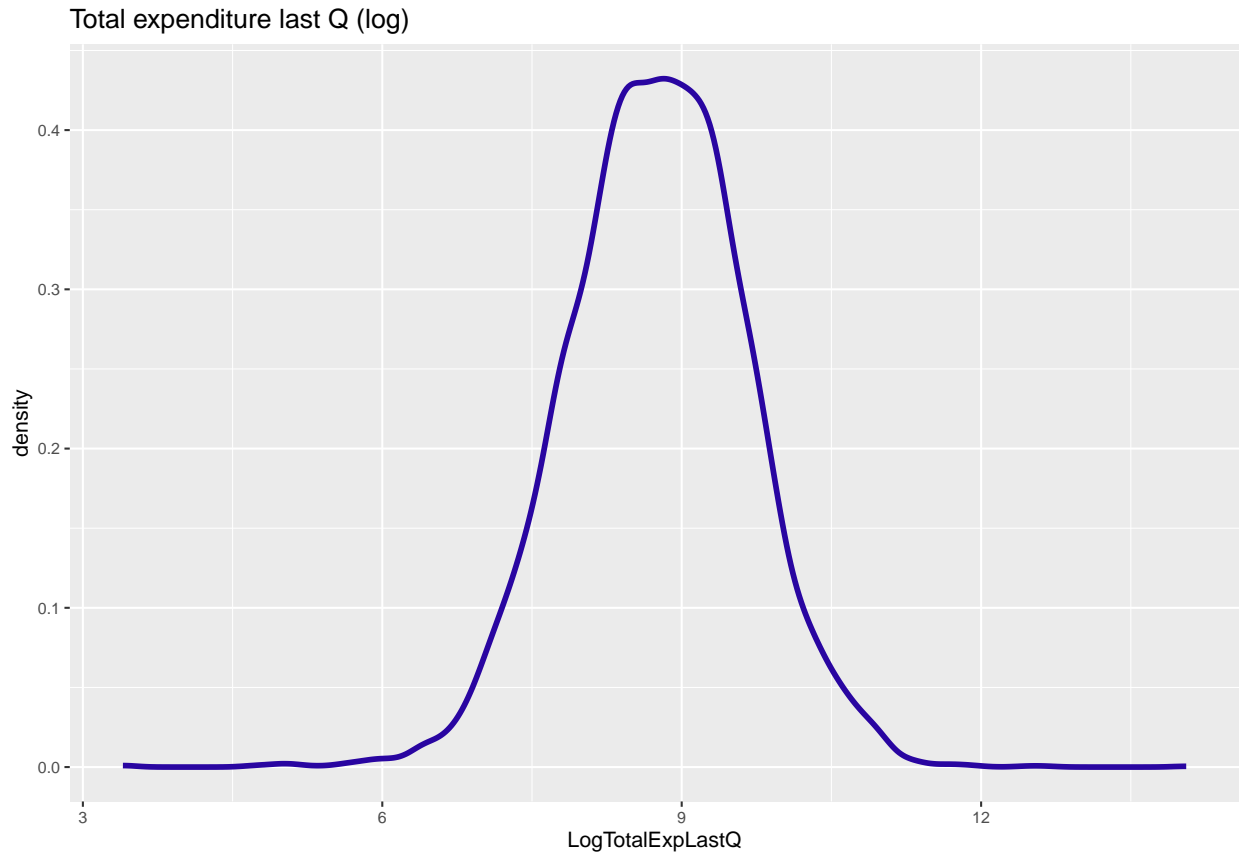
```
ggplot(data = CEsample, aes(TotalExpLastQ)) +
  geom_density(color = crcblue, size = 1) +
  labs(title = "Total expenditure last Q") +
  theme_grey(base_size = 8, base_family = "")
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



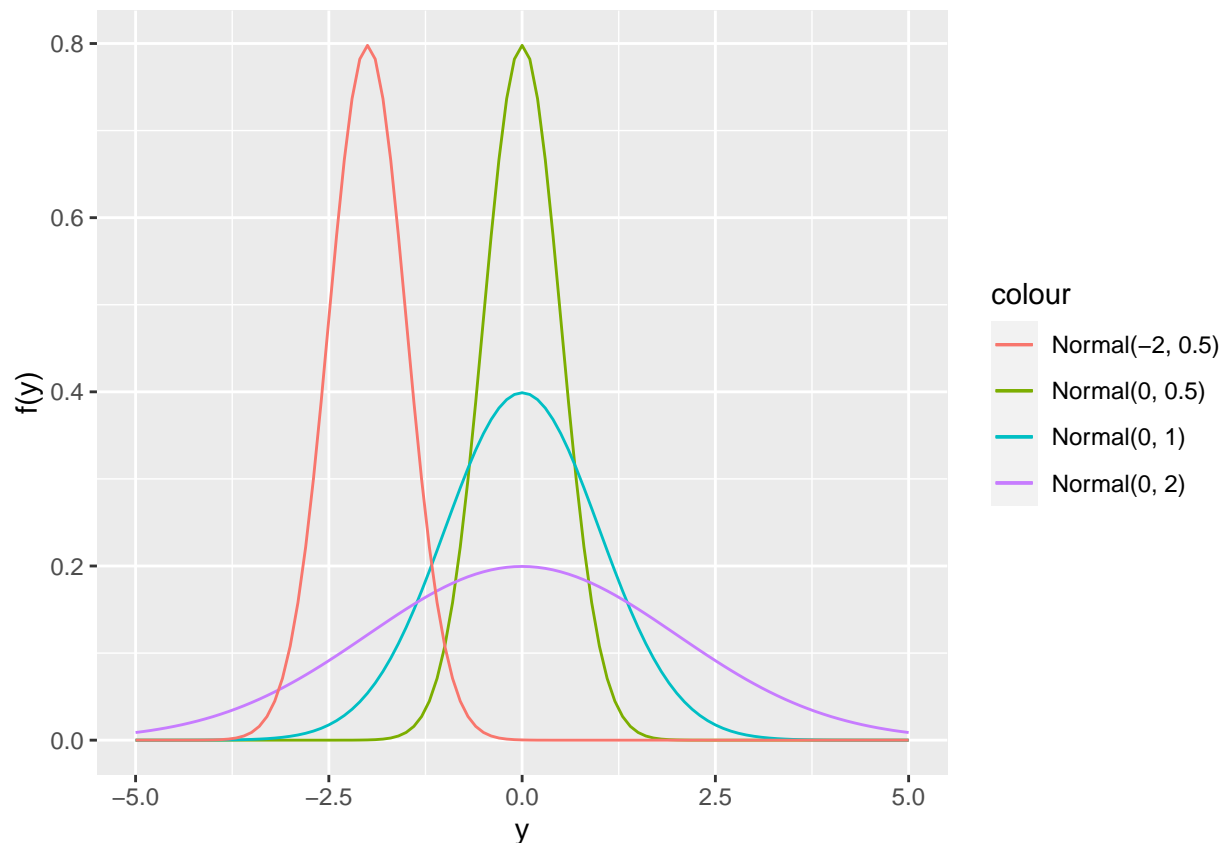
### Log transformation of the TOTEXPPQ variable

```
CEsample$LogTotalExpLastQ <- log(CEsample$TotalExpLastQ)
ggplot(data = CEsample, aes(LogTotalExpLastQ)) +
  geom_density(color = crcblue, size = 1) +
  labs(title = "Total expenditure last Q (log)") +
  theme_grey(base_size = 8, base_family = "")
```



## The Normal distribution

```
ggplot(data = data.frame(y = c(-5, 5)), aes(y)) +  
  stat_function(fun = dnorm, args = list(mean = 0, sd = 0.5), aes(color = "Normal(0, 0.5)")) +  
  stat_function(fun = dnorm, args = list(mean = 0, sd = 1), aes(color = "Normal(0, 1)")) +  
  stat_function(fun = dnorm, args = list(mean = 0, sd = 2), aes(color = "Normal(0, 2)")) +  
  stat_function(fun = dnorm, args = list(mean = -2, sd = 0.5), aes(color = "Normal(-2, 0.5)"))  
ylab("f(y)")
```



## Prior and posterior distributions for mean AND standard deviation

### Some derivations

Notice that we have unknown  $\mu$  and  $\sigma$  here. They are **both** unknown. And we derive the full conditional posterior distributions (conditioning on all of the data and every other parameters, even though they are unknown).

e.g.

- $\mu|y_1, \dots, y_n, \phi$
- $\phi|y_1, \dots, y_n, \mu$

Assuming independence, joint density = product of marginal density

$$\pi(\mu, \sigma) = \pi_1(\mu)\pi_2(\sigma) \quad (1)$$

$$L(\mu, \sigma) = f(y_1, \dots, y_n|\mu, \sigma) \quad (2)$$

$$\implies \pi(\mu, \sigma|y_1, \dots, y_n) \propto \pi(\mu, \sigma)L(\mu, \sigma) \quad (3)$$

Then we need to derive the probability distribution of  $\mu|y_1, \dots, y_n, \phi$  and  $\phi|y_1, \dots, y_n, \mu$

$$\pi(\mu, \sigma) = \pi_1(\mu)\pi_2(\sigma) \quad (4)$$

$$= \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{(\mu - \mu_0)^2}{2\sigma_0^2}\right) \frac{\beta^\alpha}{\Gamma(\alpha)} \left(\frac{1}{\sigma^2}\right)^{\alpha-1} \exp\left(-\frac{\beta}{\sigma^2}\right) \quad (5)$$

$$L(\mu, \sigma) = f(y_1, \dots, y_n | \mu, \sigma) \quad (6)$$

$$= \prod_{i=1}^n \left[ \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mu)^2}{2\sigma^2}\right) \right] \quad (7)$$

$$= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left(-\frac{\sum (y_i - \mu)^2}{2\sigma^2}\right) \quad (8)$$

Since

$$\pi(\mu, \sigma | y_1, \dots, y_n) \propto \pi(\mu, \sigma) L(\mu, \sigma) \quad (9)$$

so we have

$$\pi(\mu | y_1, \dots, y_n, \sigma) \propto \exp\left(-\frac{(\mu - \mu_0)^2}{2\sigma_0^2}\right) \exp\left(-\frac{\sum (y_i - \mu)^2}{2\sigma^2}\right) \quad (10)$$

$$\pi\left(\frac{1}{\sigma^2} | y_1, \dots, y_n, \mu\right) \propto \left(\frac{1}{\sigma^2}\right)^{\alpha-1} \exp\left(-\frac{\beta}{\sigma^2}\right) \left(\frac{1}{\sigma^2}\right)^{\frac{n}{2}} \exp\left(-\frac{\sum (y_i - \mu)^2}{2\sigma^2}\right) \quad (11)$$

In general, let's say we have 4 unknown parameters  $\theta_1, \theta_2, \theta_3, \theta_4$  and the joint prior  $\pi(\theta_1, \theta_2, \theta_3, \theta_4)$  and the likelihood  $L(\theta_1, \theta_2, \theta_3, \theta_4)$ . Then we have  $\pi(\theta_1, \theta_2, \theta_3, \theta_4 | y_1, \dots, y_n) \propto \pi(\theta_1, \theta_2, \theta_3, \theta_4) L(\theta_1, \theta_2, \theta_3, \theta_4)$ .

And then, we get

- $\pi(\theta_1 | -) \propto \dots$
- $\pi(\theta_2 | -) \propto \dots$
- $\pi(\theta_3 | -) \propto \dots$
- $\pi(\theta_4 | -) \propto \dots$

**Use R/RStudio to run a Gibbs sampler**

```
gibbs_normal <- function(input, S, seed){
  set.seed(seed)
  ybar <- mean(input$y)
  n <- length(input$y)
  para <- matrix(0, S, 2) # Sx2 zero matrix
```

```

phi <- input$phi_init
for(s in 1:S){
  mu1 <- (input$mu_0/input$sigma_0^2 + n*phi*ybar)/
  (1/input$sigma_0^2 + n*phi)
  sigma1 <- sqrt(1/(1/input$sigma_0^2 + n*phi))
  mu <- rnorm(1, mean = mu1, sd = sigma1)
  alpha1 <- input$alpha + n/2
  beta1 <- input$beta + sum((input$y - mu)^2)/2
  phi <- rgamma(1, shape = alpha1, rate = beta1)
  para[s, ] <- c(mu, phi)
}
para }

```

- Run the Gibbs sampler:

```

input <- list(y = CEsample$LogTotalExpLastQ, mu_0 = 5, sigma_0 = 1,
alpha = 1, beta = 1, phi_init = 1)
output <- gibbs_normal(input, S = 10000, seed = 123)

```

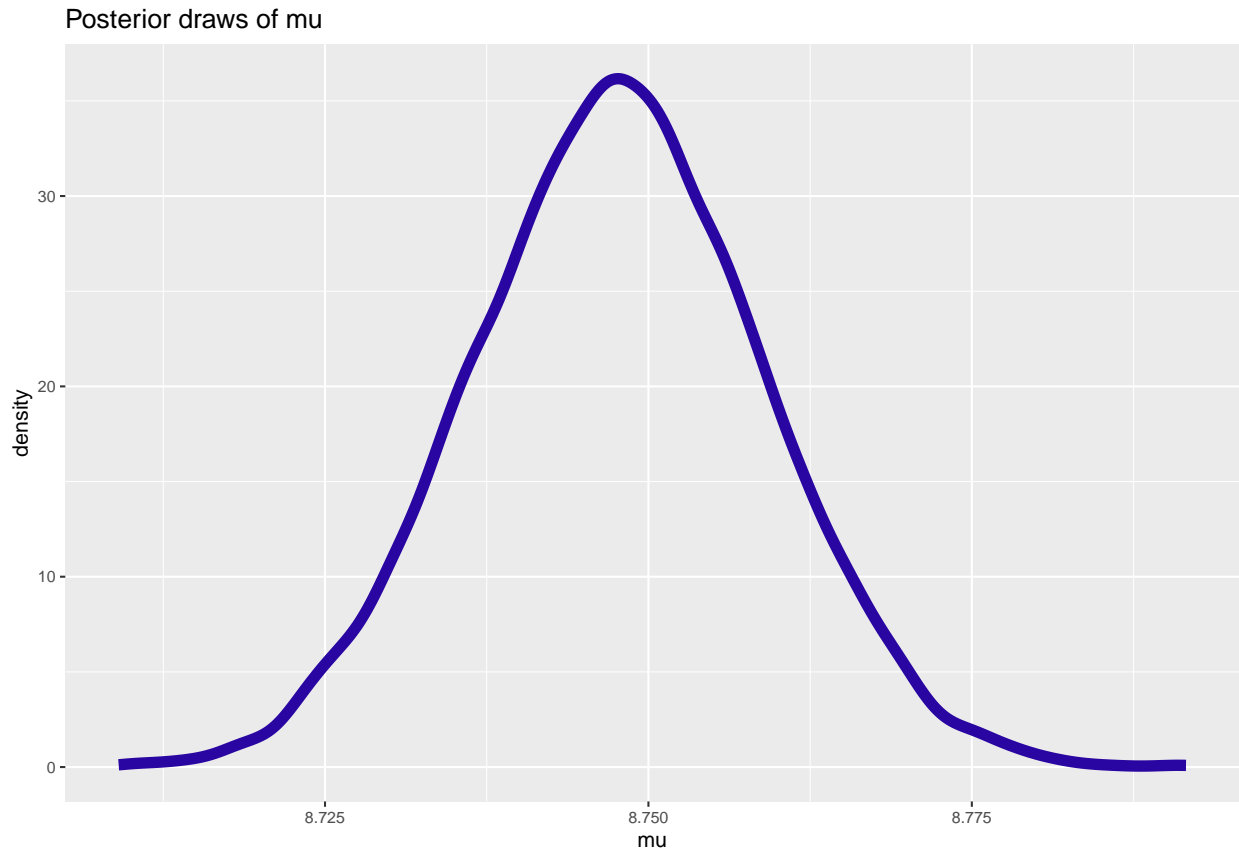
- Extract posterior draws of mu and phi from the Gibbs sampler output:

```

para_post <- as.data.frame(output)
names(para_post) <- c("mu", "phi")

ggplot(para_post, aes(mu)) +
  geom_density(size = 2, color = crcblue) +
  labs(title = "Posterior draws of mu") +
  theme_grey(base_size = 8,
base_family = "")

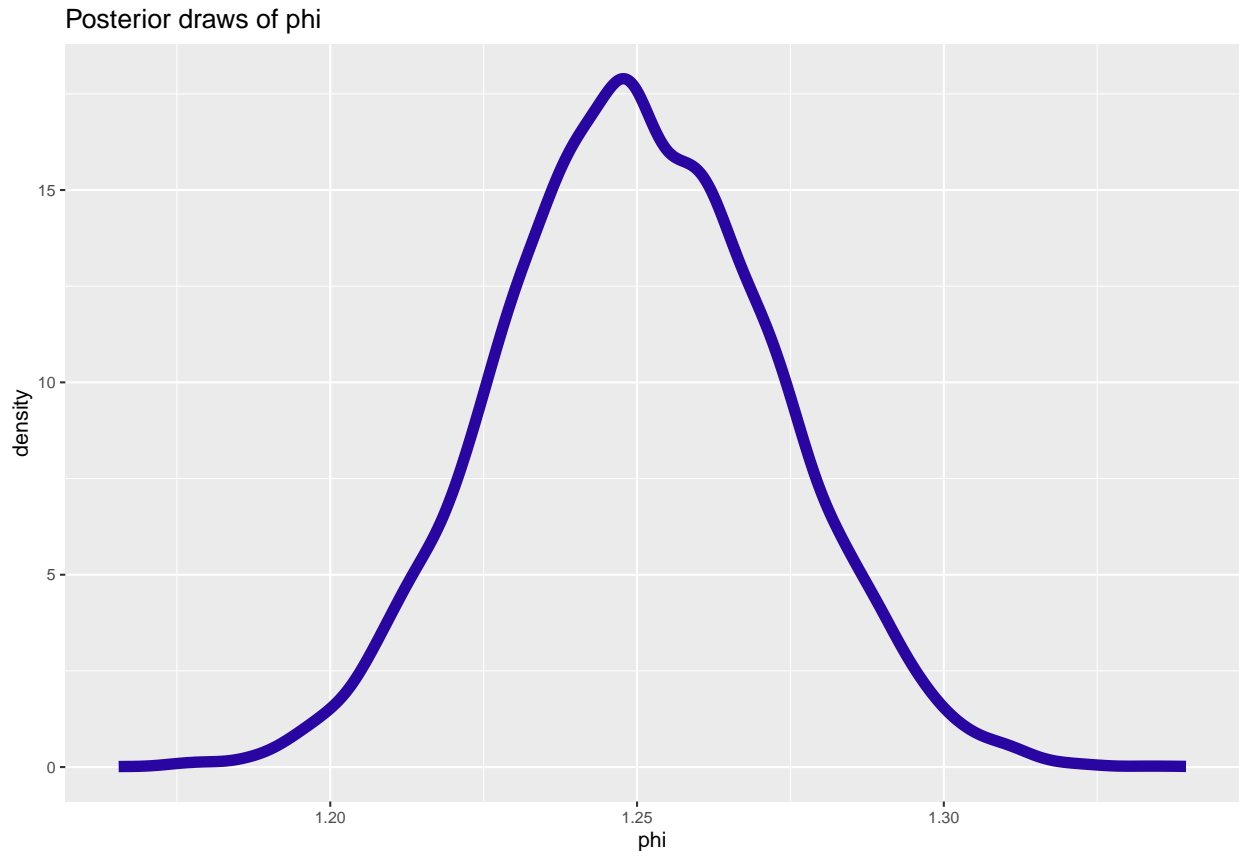
```



```
quantile(para_post$mu, c(0.025,0.975))
```

```
##      2.5%    97.5%  
## 8.725482 8.769543
```

```
ggplot(para_post, aes(phi)) +  
  geom_density(size = 2, color = crcblue) +  
  labs(title = "Posterior draws of phi") +  
  theme_grey(base_size = 8,  
    base_family = "")
```



```
quantile(para_post$phi, c(0.025,0.975))
```

```
##      2.5%    97.5%
## 1.206482 1.294191
```

- Exercise 1: Update the Gibbs sampler to initiate it with `mu_init`. Compare your results to the existing results. You should get similar results meaning that it doesn't matter you start with  $\mu$  or  $\phi$  if indeed the Gibbs sampler has converted to the true posterior, you can even try different initial values.

```
gibbs_normal <- function(input, S, seed){
  set.seed(seed)
  ybar <- mean(input$y)
  n <- length(input$y)
  para <- matrix(0, S, 2) # Sx2 zero matrix
  mu <- input$mu_init # change here
  for(s in 1:S){
    # start with phi first
    alpha1 <- input$alpha + n/2
    beta1 <- input$beta + sum((input$y - mu)^2)/2
    phi <- rgamma(1, shape = alpha1, rate = beta1)

    mu1 <- (input$mu_0/input$sigma_0^2 + n*phi*ybar)/
      (1/input$sigma_0^2 + n*phi)
  }
}
```



```

    sigma1 <- sqrt(1/(1/input$sigma_0^2 + n*phi))
    mu <- rnorm(1, mean = mu1, sd = sigma1)
    para[s, ] <- c(mu, phi)
  }
para }

```

- Exercise 2: Update the Gibbs sampler to initiate it with `mu_init` and `phi_init`. Compare your results to the existing results.

If it converges, eventually the results should be the same.

## Use JAGS (Just Another Gibbs Sampler) and Bayesian inferences

### JAGS for unknown mean and standard deviation case

- R package `runjags` to run Markov chain Monte Carlo simulations.
- Descriptive of the sampling model and the prior.
- Installing JAGS software and `runjags` R package
  - Download JAGS at this link
  - Install and load `runjags` R package

```
#install.packages("runjags")
```

```
library(runjags)
```

- Only need to focus on the sampling density and the prior:
  - The sampling density:

$$y_1, \dots, y_n \mid \mu, \sigma \stackrel{i.i.d.}{\sim} \text{Normal}(\mu, \sigma).$$

- The prior distributions:

$$\begin{aligned} \mu &\sim \text{Normal}(\mu_0, \sigma_0), \\ 1/\sigma^2 = \phi &\sim \text{Gamma}(\alpha, \beta). \end{aligned}$$

```

modelString <- "
model{

# The sampling density
for (i in 1:N) {
y[i] ~ dnorm(mu, phi)
}

# The prior distributions
mu ~ dnorm(mu_0, phi_0)
phi ~ dgamma(alpha, beta)

}
"

```

- Pass the data and hyperparameter values to JAGS:

```
y <- CEsample$LogTotalExpLastQ
N <- length(y)
the_data <- list("y" = y, "N" = N, "mu_0" = 5, "phi_0" = 1/1^2,
"alpha" = 1, "beta" = 1)
```

- Run the JAGS code for this model:

```
posterior <- run.jags(modelString,
  data = the_data,
  monitor = c("mu", "phi"),
  n.chains = 1,
  adapt = 1000,
  burnin = 2000,
  sample = 5000,
  thin = 1)
```

```
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Note: the model did not require adaptation
## Burning in the model for 2000 iterations...
## Running the model for 5000 iterations...
## Simulation complete
## Calculating summary statistics...

## Warning: Convergence cannot be assessed with only 1 chain
## Finished running the simulation
```

- Obtain posterior summaries of mu and phi:

```
summary(posterior)
```

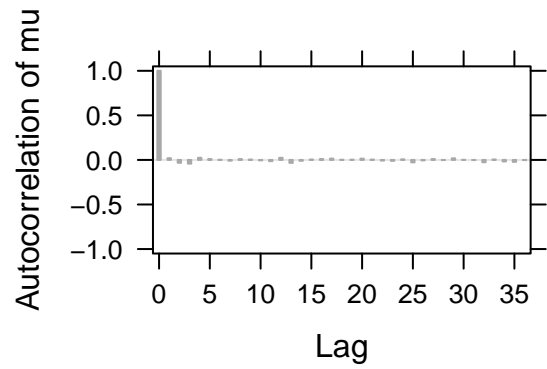
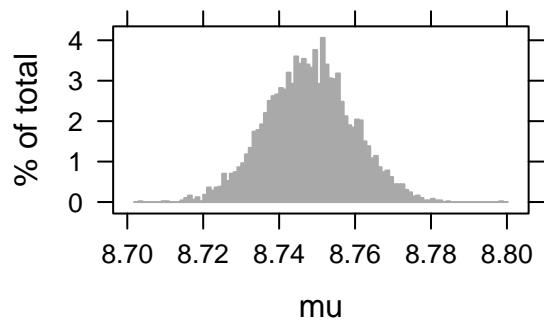
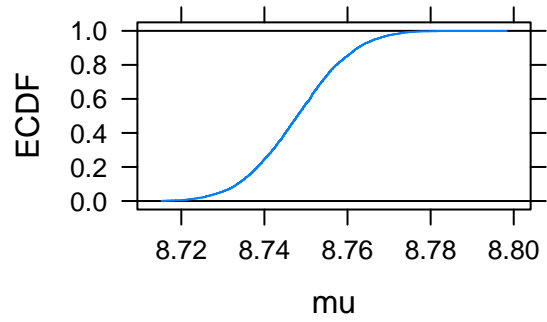
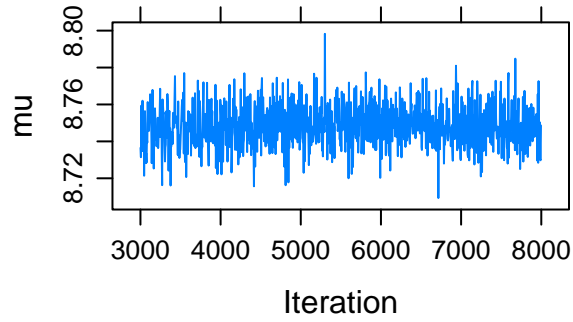
```
##      Lower95   Median Upper95      Mean      SD Mode      MCerr MC%ofSD
## mu  8.725792 8.747913 8.770037 8.747900 0.01129891   NA 0.0001552814    1.4
## phi 1.204248 1.250242 1.291397 1.250135 0.02229710   NA 0.0003153286    1.4
##      SSeff      AC.10 psrf
## mu   5295 -0.004944652   NA
## phi   5000  0.013941605   NA
```

## MCMC diagnostics

### Trace plots example

```
plot(posterior, vars = "mu")
```

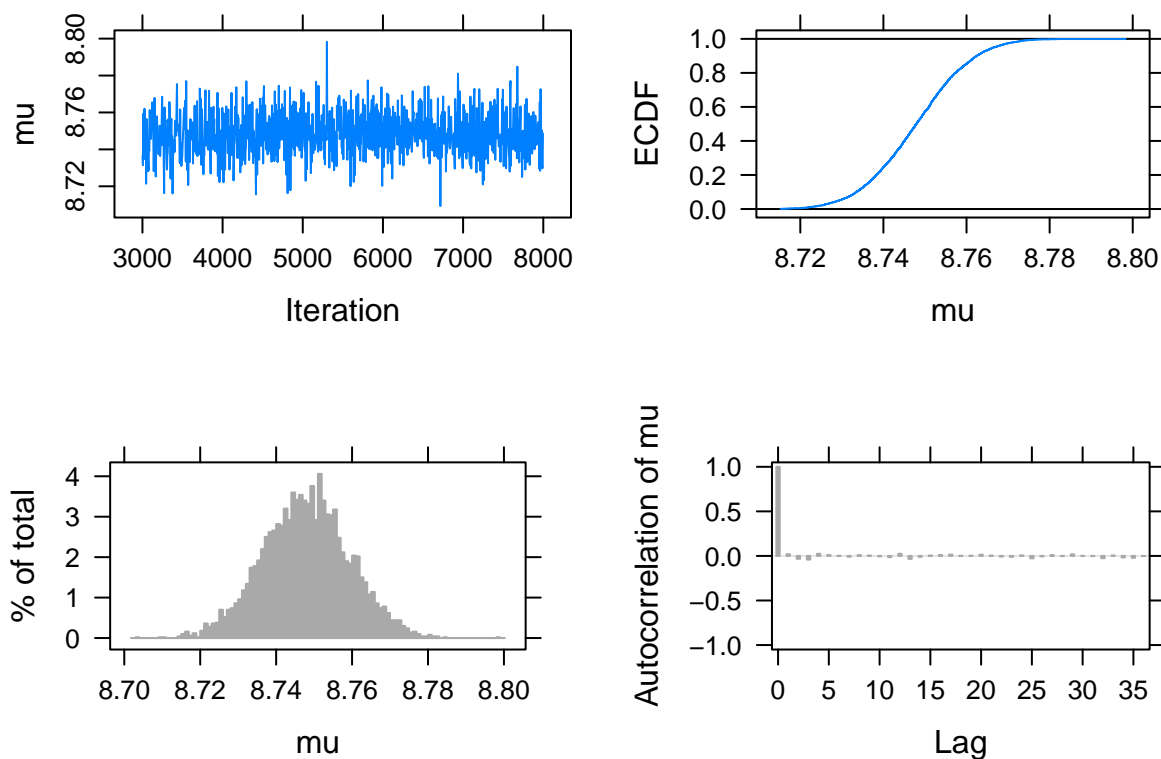
```
## Generating plots...
```



### ACF plots example

```
plot(posterior, vars = "mu")
```

```
## Generating plots...
```



### Effective sample size example

- The column of `SSEff`; recall `sample` is 5000.

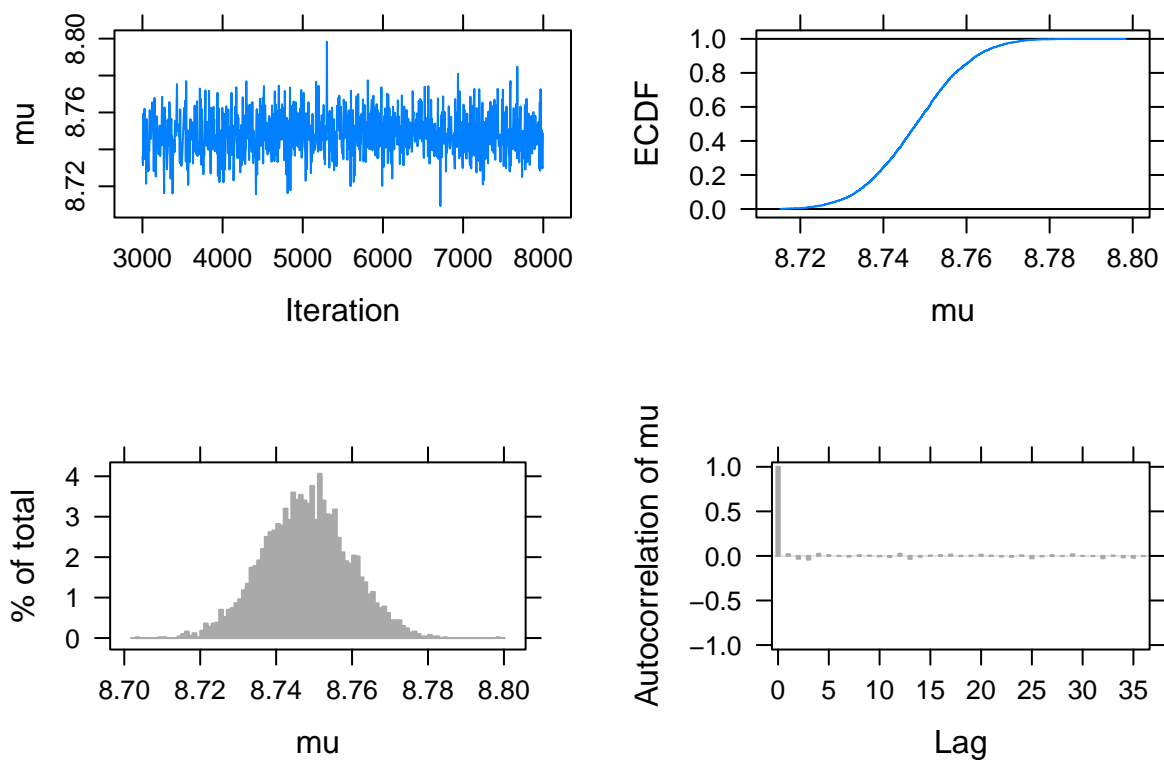
```
summary(posterior)
```

```
##      Lower95   Median Upper95    Mean      SD Mode      MCerr MC%ofSD
## mu  8.725792 8.747913 8.770037 8.747900 0.01129891   NA 0.0001552814    1.4
## phi 1.204248 1.250242 1.291397 1.250135 0.02229710   NA 0.0003153286    1.4
##      SSEff      AC.10 psrf
## mu   5295 -0.004944652   NA
## phi   5000  0.013941605   NA
```

### MCMC diagnostics for the CE example

```
plot(posterior, vars = "mu")
```

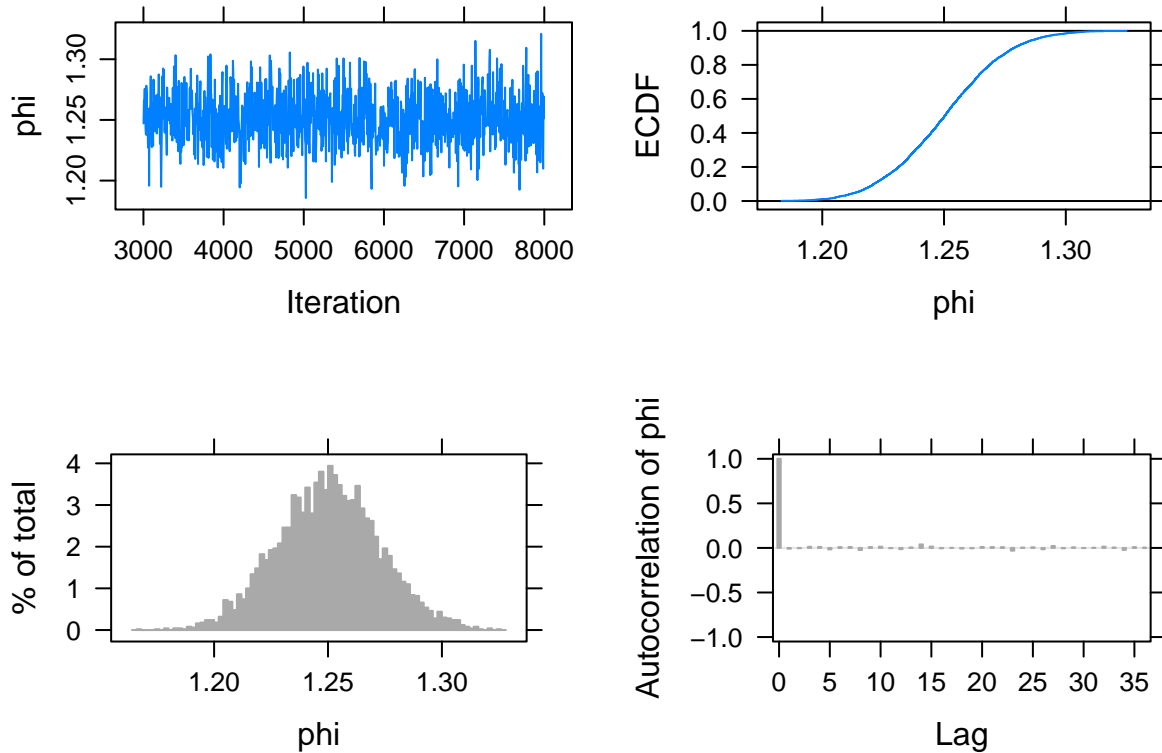
```
## Generating plots...
```



MCMC diagnostics for the CE example cont'd

```
plot(posterior, vars = "phi")
```

```
## Generating plots...
```



## Gelman-Rubin diagnostics example

- Create initial values of mu and phi:

```
inits1 <- dump.format(list(mu = 1, phi = 1,
                          .RNG.name="base::Super-Duper", .RNG.seed = 1))
inits2 <- dump.format(list(mu = 10, phi = 10,
                          .RNG.name="base::Wichmann-Hill", .RNG.seed = 2))
```

- Feed in inits1 and inits2, and let n.chains = 2:

```
posterior_2chains <- run.jags(modelString,
                             data = the_data,
                             monitor = c("mu", "phi"),
                             n.chains = 2,
                             inits=c(inits1, inits2),
                             adapt = 1000,
                             burnin = 2000,
                             sample = 5000,
                             thin = 1)
```

```
## Compiling rjags model...
## Calling the simulation using the rjags method...
```

```
## Note: the model did not require adaptation
## Burning in the model for 2000 iterations...
## Running the model for 5000 iterations...
## Simulation complete
## Calculating summary statistics...
## Calculating the Gelman-Rubin statistic for 2 variables....
## Finished running the simulation
```

## Gelman-Rubin diagnostics example cont'd

- Return `psrf` from the output, as Gelman-Rubin diagnostic results:

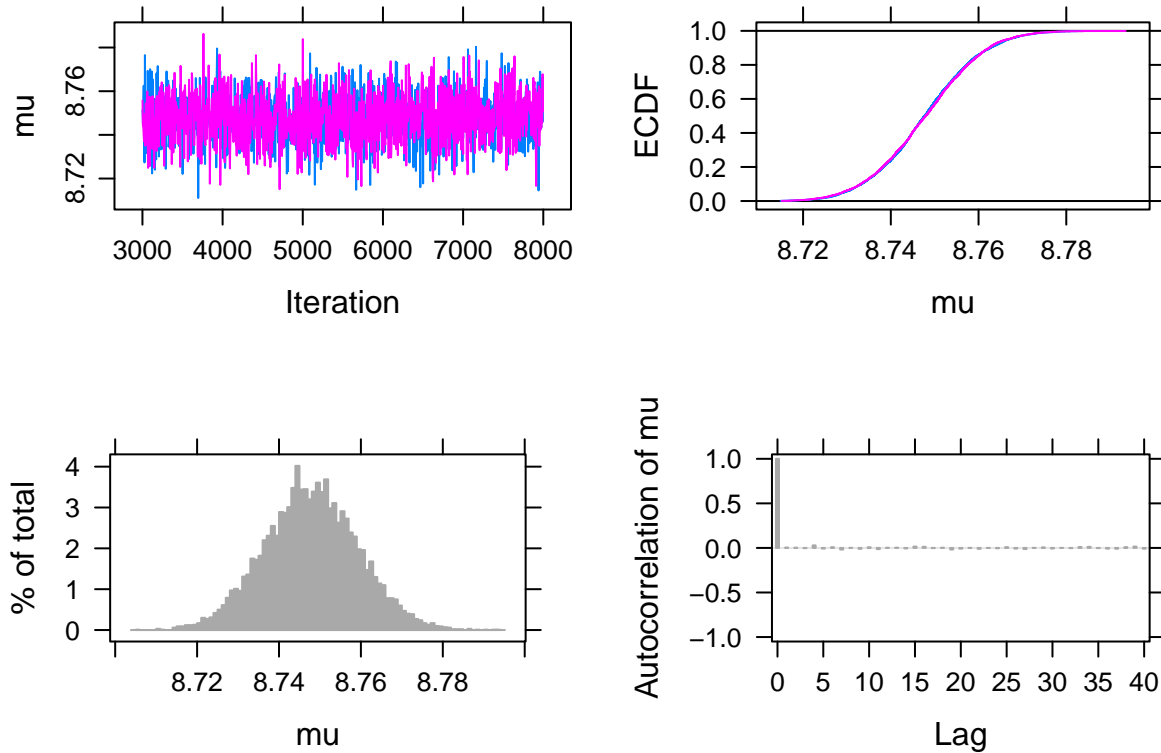
```
posterior_2chains$psrf

## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## mu           1           1
## phi          1           1
##
## Multivariate psrf (for all monitored variables):
##
## 1
##
## Target psrf
##
## 1.05
```

## MCMC diagnostics for the CE example, 2 chains

```
plot(posterior_2chains, vars = "mu")

## Generating plots...
```



### Useful diagnostics/functions in coda package

- One needs to convert parameter draws into an MCMC object. For example:

```
# install.packages("coda")
```

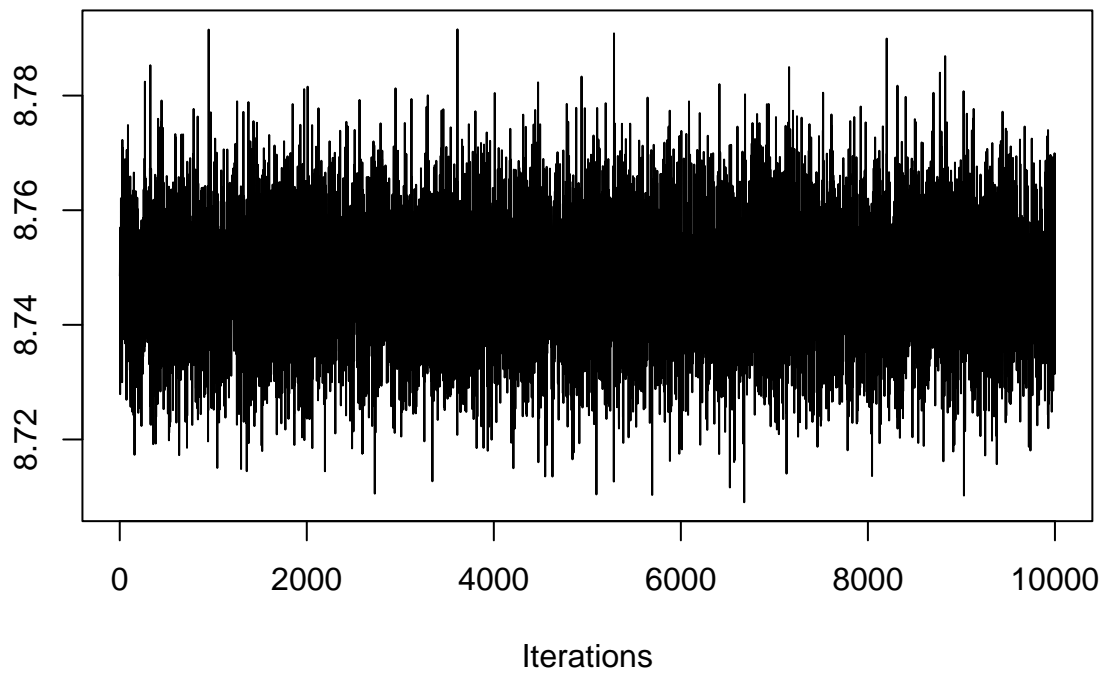
```
library(coda)
output <- gibbs_normal(input, S = 10000, seed = 123)
para_post = as.data.frame(output)
names(para_post) = c("mu", "phi")
```

- Then one can perform MCMC diagnostics. For example:

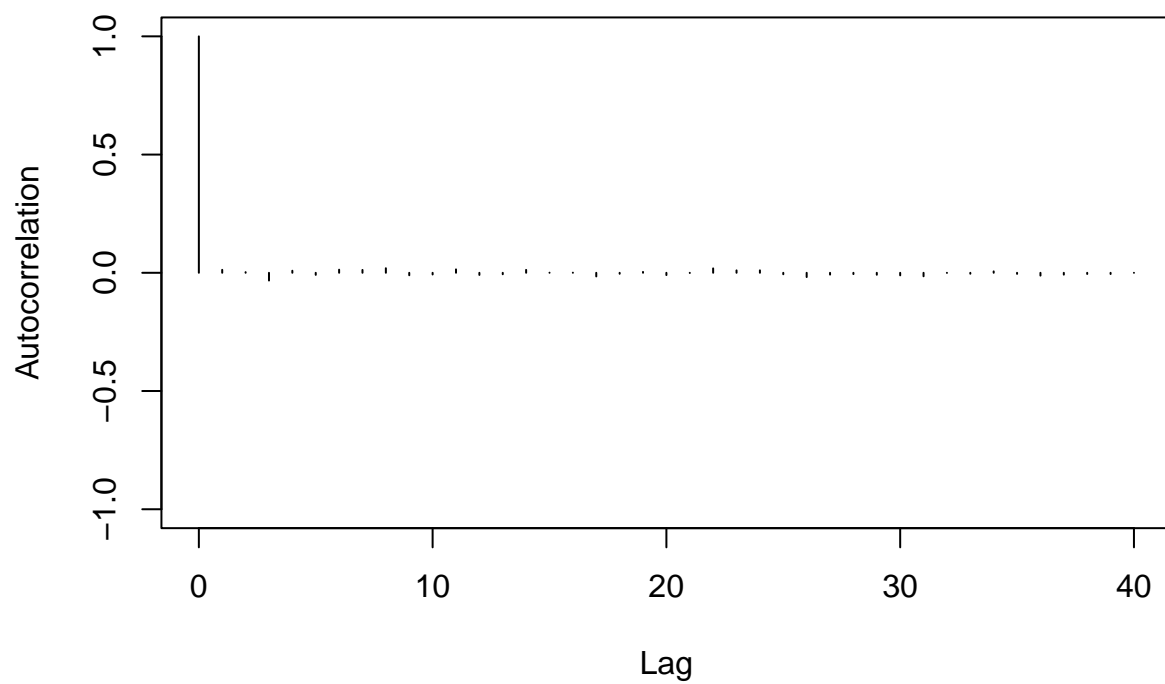
```
mu.mcmc = as.mcmc(para_post$mu)
```

```
traceplot(mu.mcmc)
```





```
autocorr.plot(mu.mcmc)
```



```
effectiveSize(mu.mcmc)
```

```
##      var1  
## 10355.44
```

```
gelman.diag(mu.mcmc)
```

Note: `gelman.diag()` needs at least 2 chains.

## Recap