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

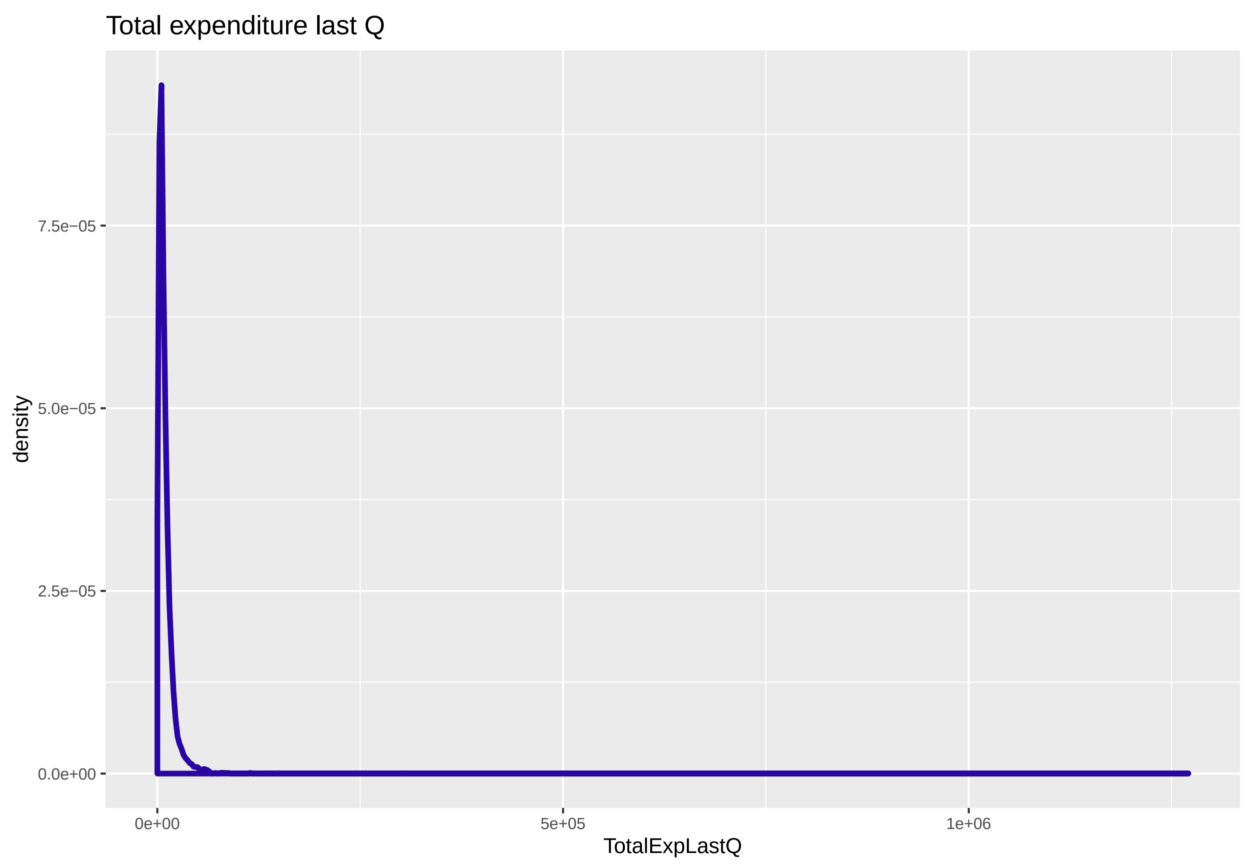
```
CEsample <- read_csv("CEsample1.csv")

summary(CEsample$TotalExpLastQ)
```

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

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

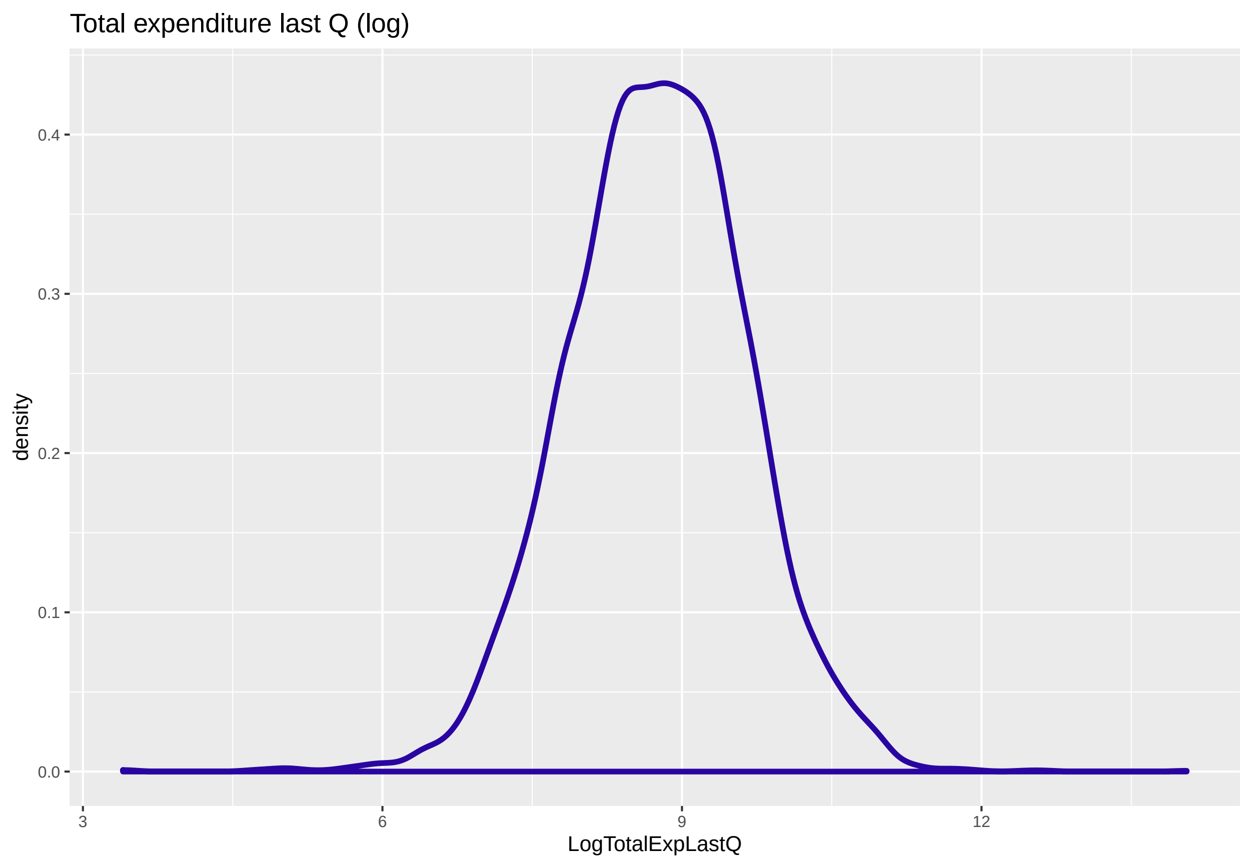
```
## [1] 19341.25
```

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



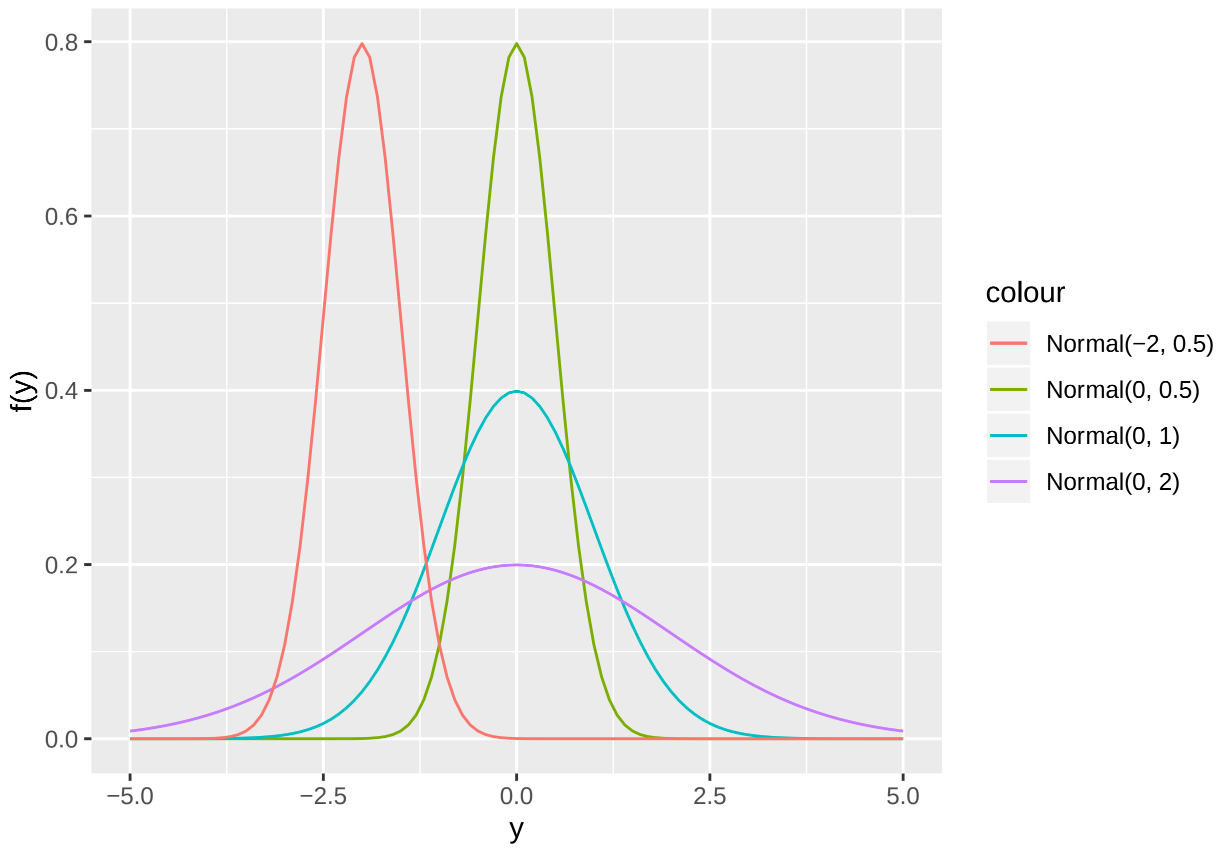
Log transformation of the TOTEXPPQ variable

```r
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

```r
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

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)
  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 }
```

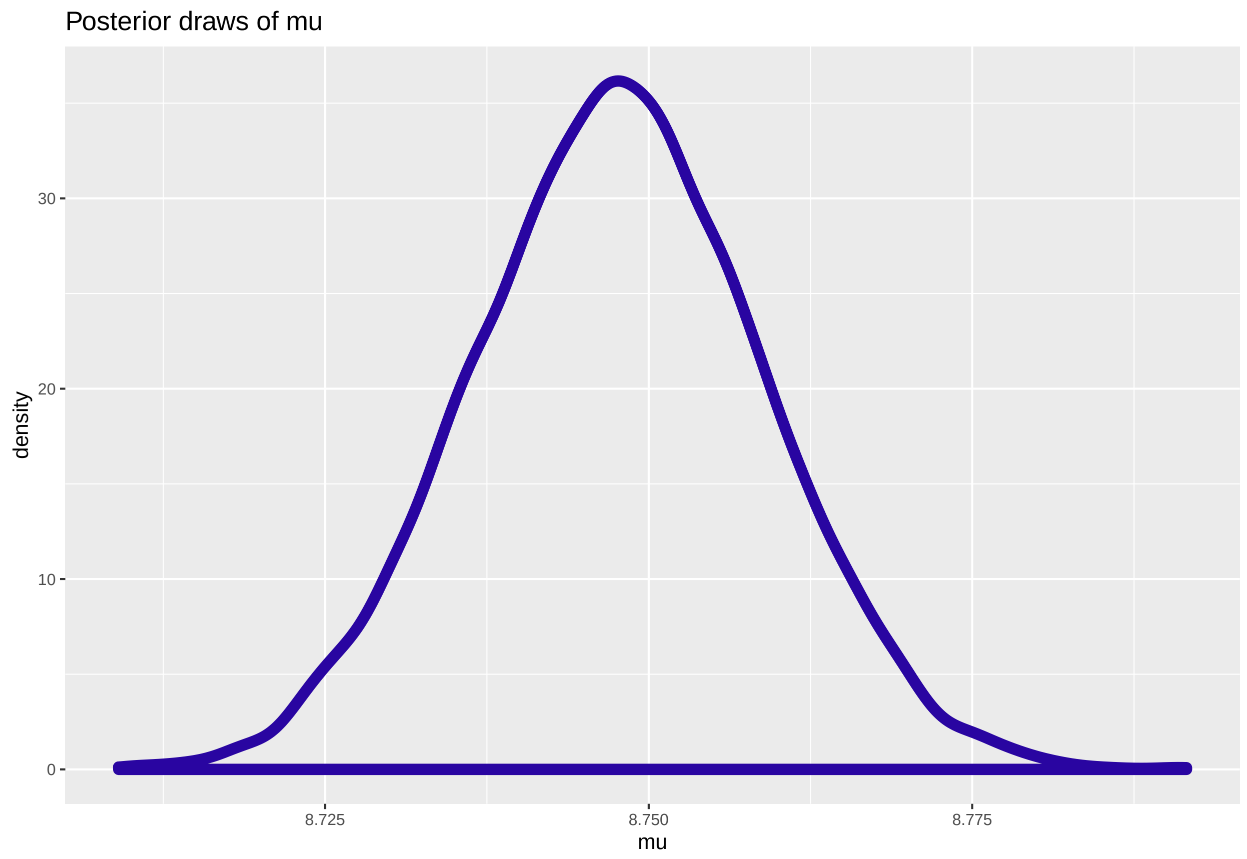
- Run the Gibbs sampler:

```
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 = "")
```

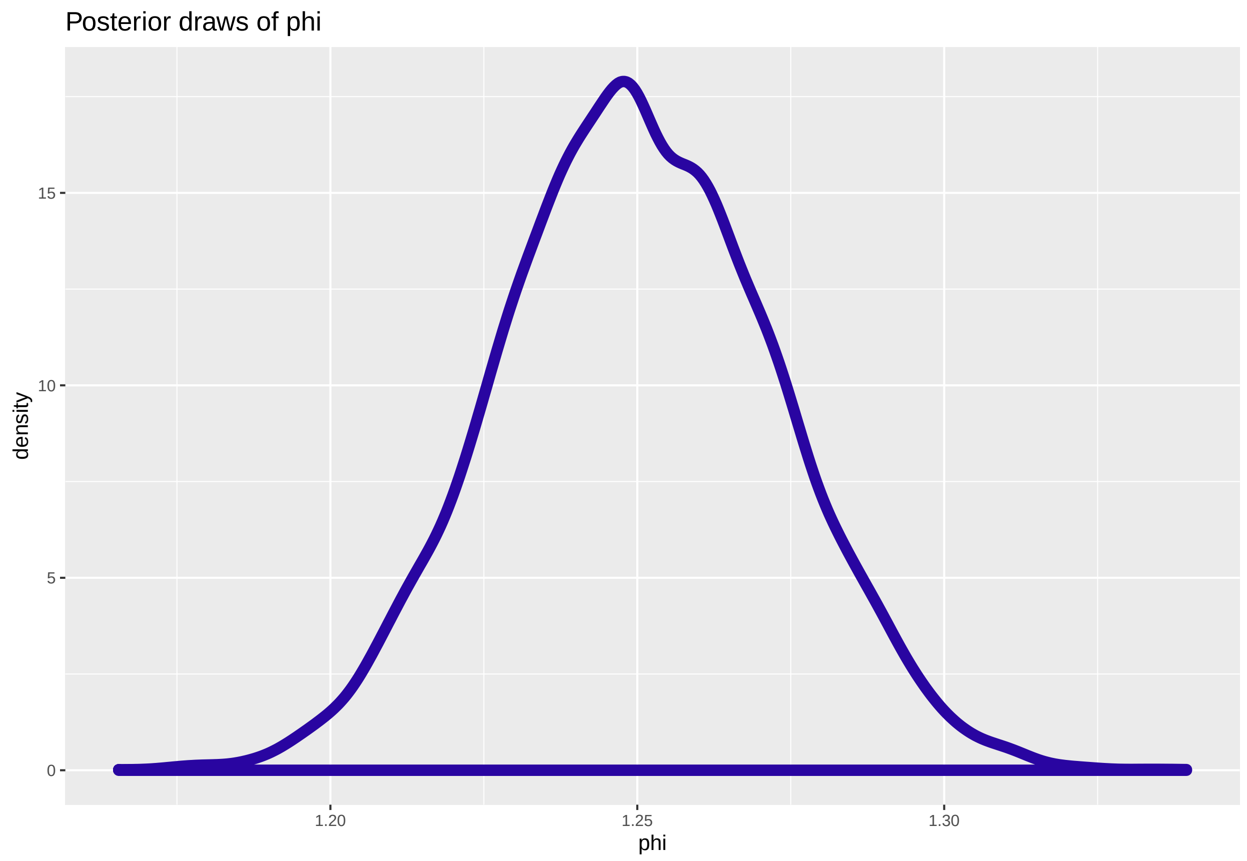


```
quantile(para_post$mu, c(0.025,0.975))
```

```
##     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") +
```

```r
  theme_grey(base_size = 8,
  base_family = "")
```



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

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

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

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

```r
library(runjags)
```

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

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

  - The prior distributions:

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

```r
modelString <- "
model{
for (i in 1:N) {
y[i] ~ dnorm(mu, phi)
}
mu ~ dnorm(mu_0, phi_0)
phi ~ dgamma(alpha, beta)
}
"
```

- Pass the data and hyperparameter values to JAGS:

```r
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:

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

```
## Calling the simulation...
## Welcome to JAGS 4.3.0 on Mon Sep 30 14:27:26 2019
## JAGS is free software and comes with ABSOLUTELY NO WARRANTY
## Loading module: basemod: ok
## Loading module: bugs: ok
## . . Reading data file data.txt
## . Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 6208
##    Unobserved stochastic nodes: 2
##    Total graph size: 6215
## . Initializing model
## . Adaptation skipped: model is not in adaptive mode.
## . Updating 2000
## -------------------------------------------------| 2000
## ************************************************** 100%
## . . . Updating 5000
## -------------------------------------------------| 5000
## ************************************************** 100%
## . . . . Updating 0
## . Deleting model
## .
## Note: the model did not require adaptation
## Simulation complete.  Reading coda files...
## Coda files loaded successfully
## 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)
```

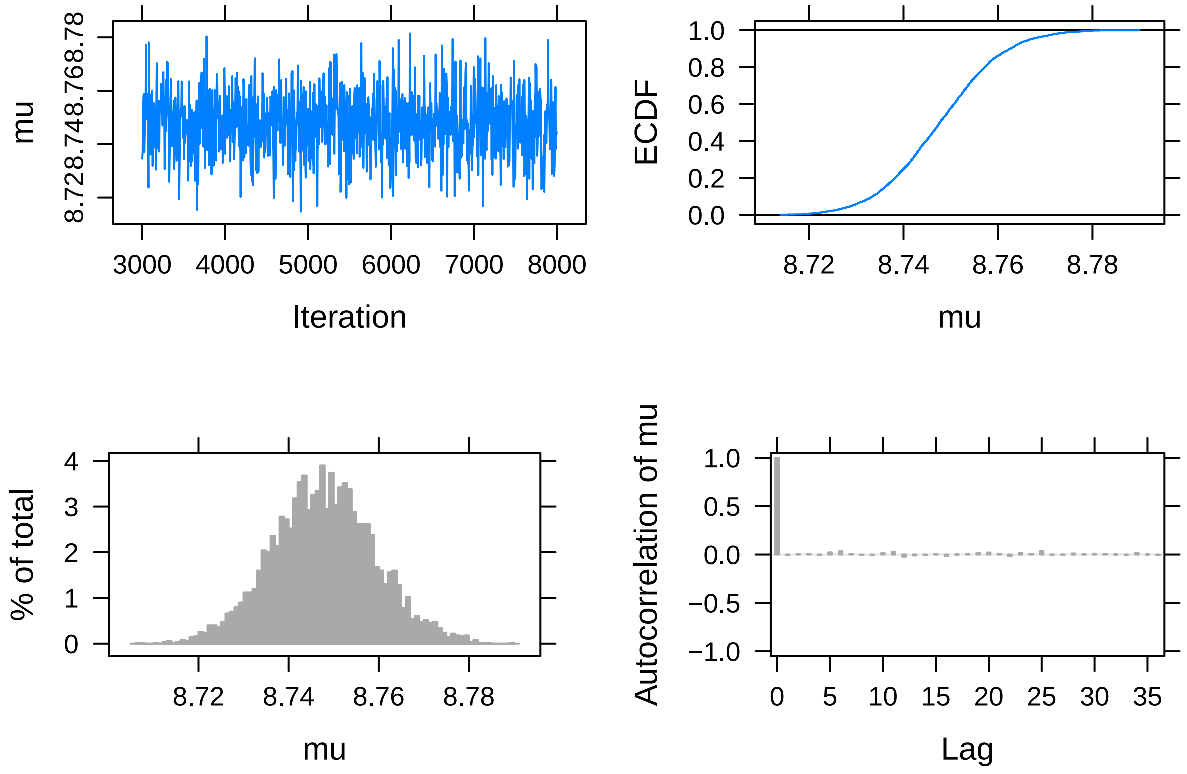
```
##     Lower95  Median Upper95     Mean         SD Mode        MCerr MC%ofSD
## mu  8.72585 8.74766 8.77133 8.747782 0.01144127   NA 0.0001618039     1.4
## phi 1.20622 1.24954 1.29270 1.250127 0.02226598   NA 0.0003148885     1.4
##     SSeff      AC.10 psrf
## mu   5000 0.01412715   NA
## phi  5000 0.01411553   NA
```

MCMC diagnostics

Trace plots example

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

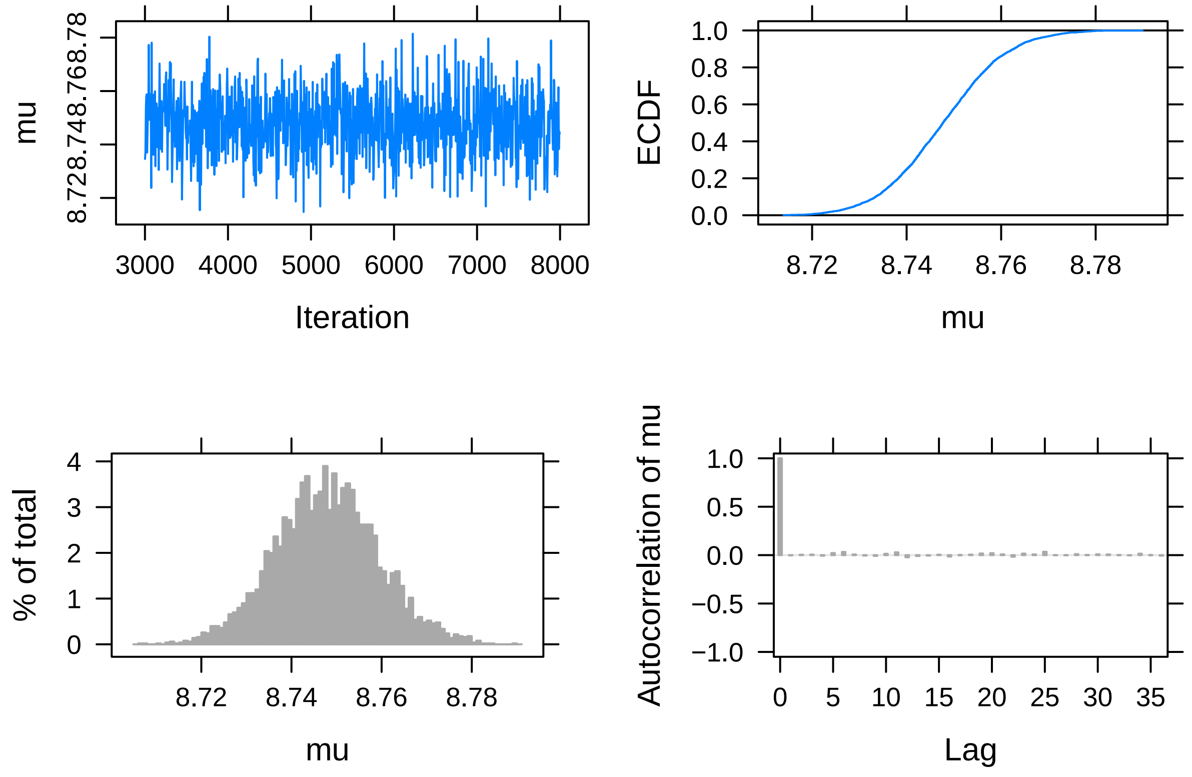
```
## Generating plots...
```



ACF plots example

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

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



Effective sample size example

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

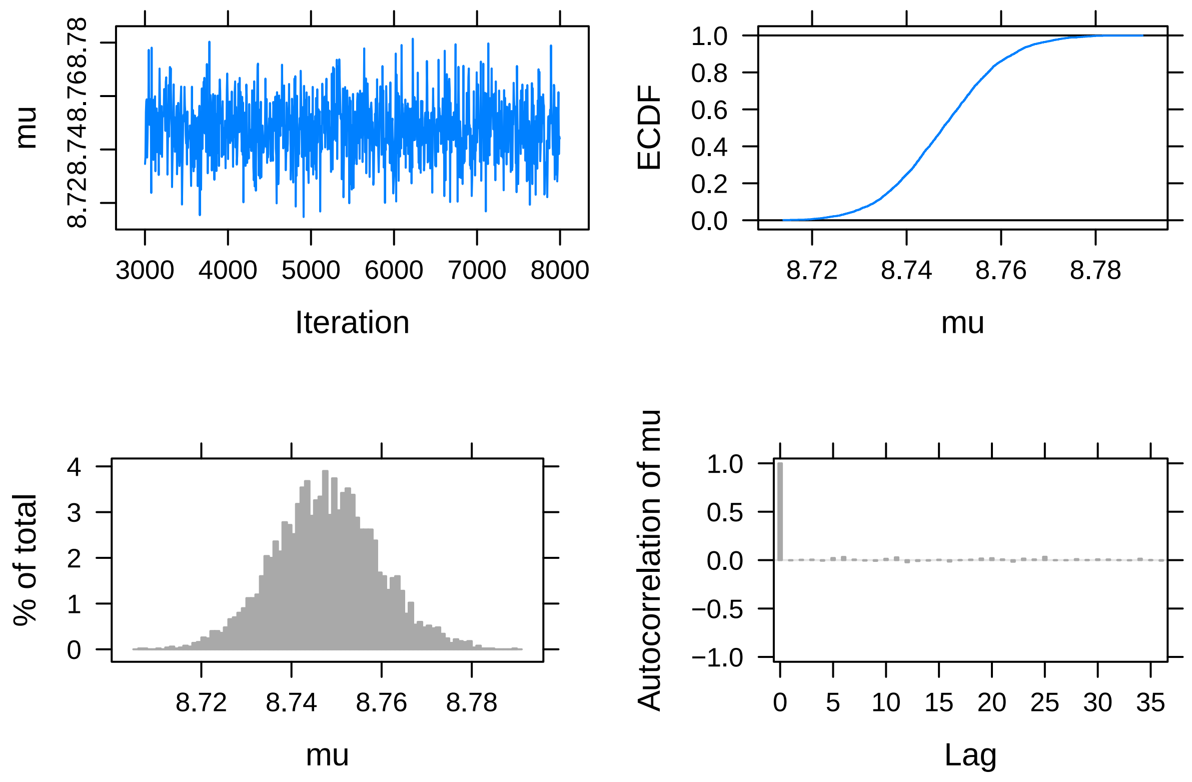
```
summary(posterior)
```

```
##     Lower95  Median Upper95     Mean         SD Mode        MCerr MC%ofSD
## mu  8.72585 8.74766 8.77133 8.747782 0.01144127   NA 0.0001618039     1.4
## phi 1.20622 1.24954 1.29270 1.250127 0.02226598   NA 0.0003148885     1.4
##     SSeff      AC.10 psrf
## mu   5000 0.01412715   NA
## phi  5000 0.01411553   NA
```

MCMC diagnostics for the CE example

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

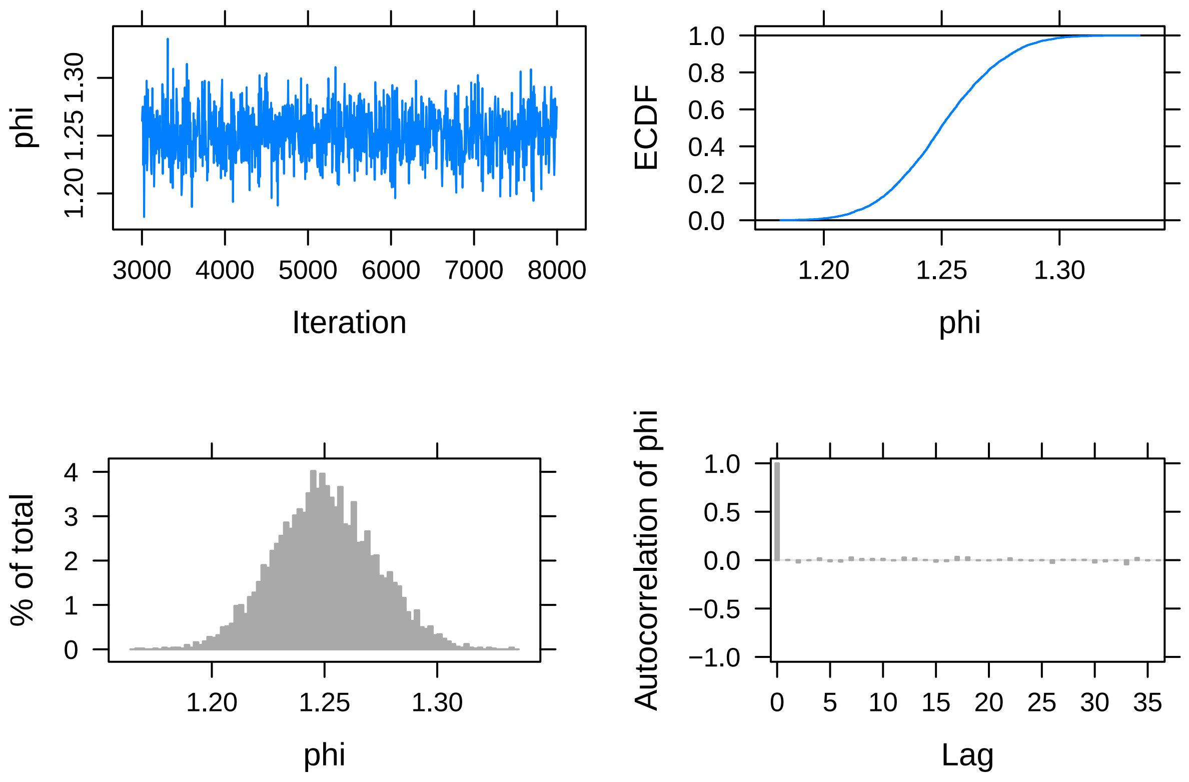
```
## Generating plots...
```



MCMC diagnostics for the CE example cont'd

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

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



Gelman-Rubin diagnostics example

- Create intinial 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)
```

```
## Calling the simulation...
## Welcome to JAGS 4.3.0 on Mon Sep 30 14:27:32 2019
## JAGS is free software and comes with ABSOLUTELY NO WARRANTY
## Loading module: basemod: ok
## Loading module: bugs: ok
## . . Reading data file data.txt
## . Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 6208
##    Unobserved stochastic nodes: 2
##    Total graph size: 6215
## . Reading parameter file inits1.txt
## . Reading parameter file inits2.txt
## . Initializing model
## . Adaptation skipped: model is not in adaptive mode.
## . Updating 2000
## -------------------------------------------------| 2000
## ************************************************** 100%
## . . . Updating 5000
## -------------------------------------------------| 5000
## ************************************************** 100%
## . . . . . Updating 0
## . Deleting model
## .
## Note: the model did not require adaptation
## Simulation complete.  Reading coda files...
## Coda files loaded successfully
## 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
```

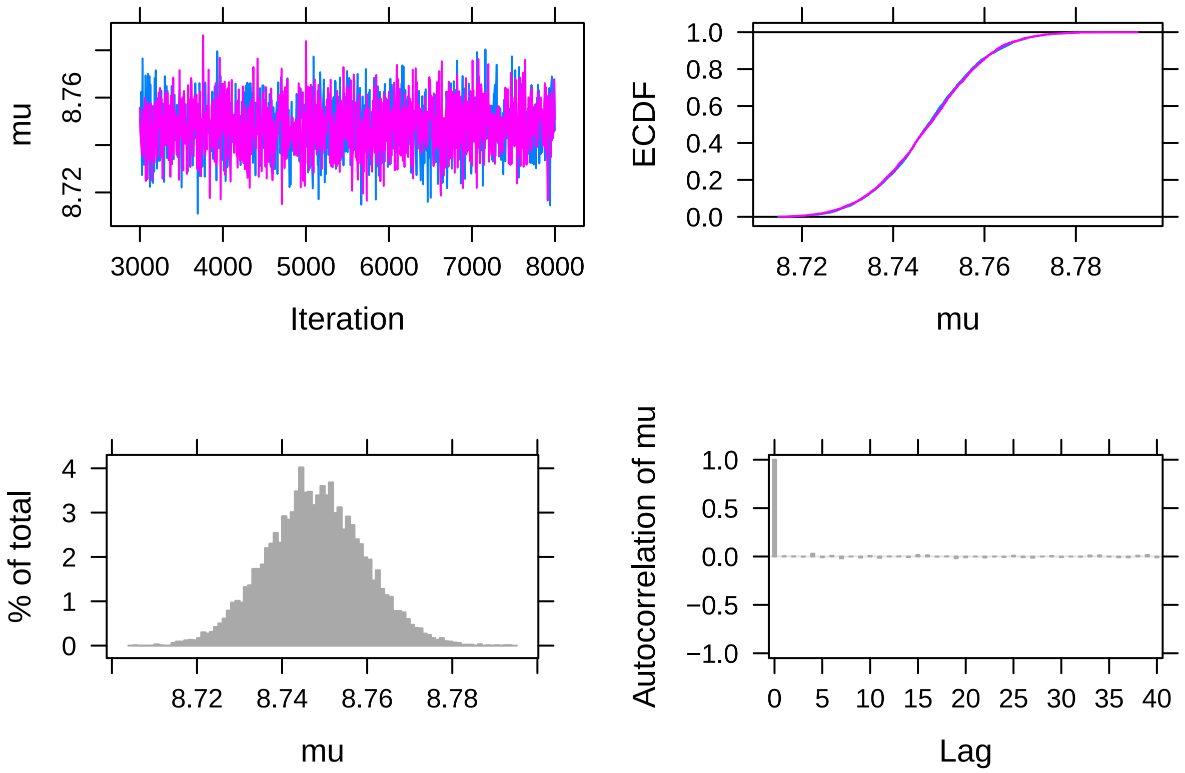
```
## Potential scale reduction factors:
##
##     Point est. Upper C.I.
## mu           1          1
## phi          1          1
##
## Multivariate psrf (for all monitored variables):
##
## 1
##
## Target psrf
##
```

```
## 1.05
```

MCMC diagnostics for the CE example, 2 chains

```r
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:

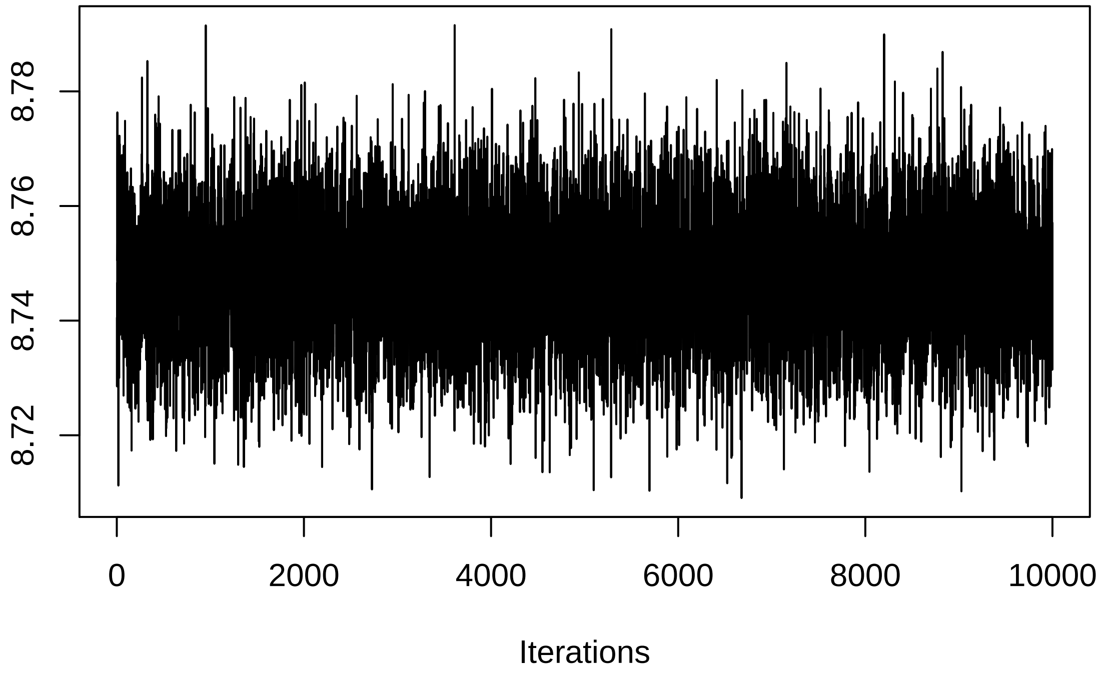
```r
install.packages("coda")
```

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

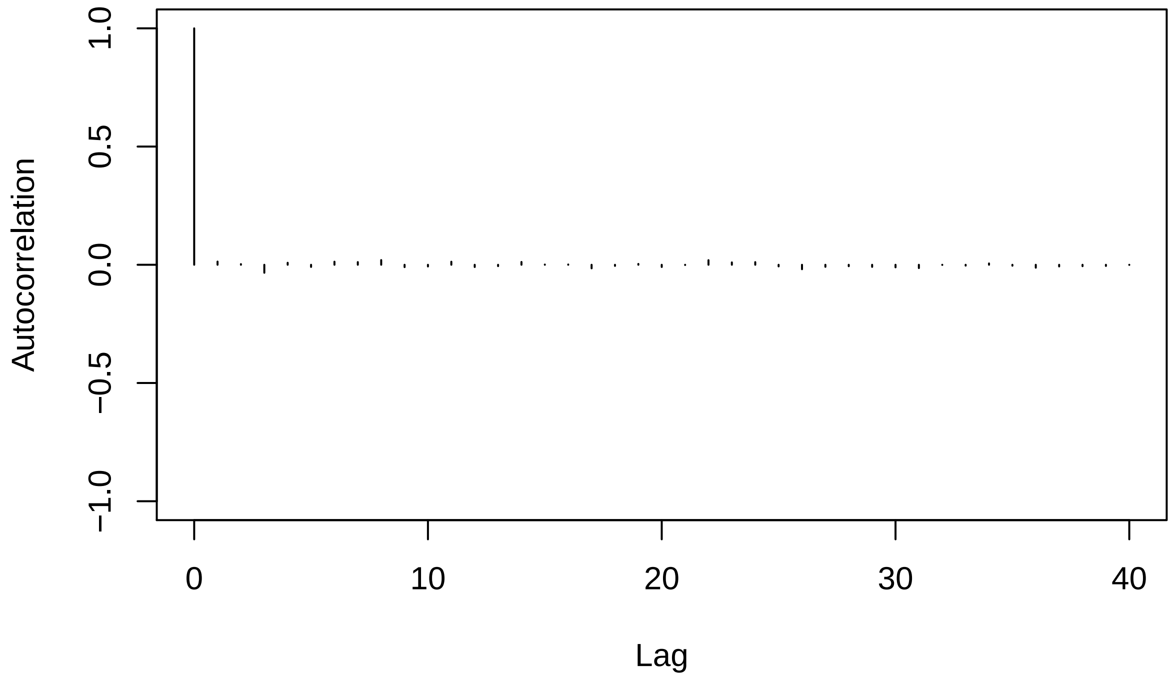
- Then one can perform MCMC diagnostics. For example:

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

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



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



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

```
##    var1
## 10356.8
```

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

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

Recap