Exercises_4 Reading Course Part2

Bayesian Methods

Achladianakis Minas

An assignment overview report presented for the
Reading Course Bayesian Methods.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE

Department of Applied Mathematics

University of Crete

Greece, March 21, 2025

Abstract

This reading course is about Bayesian Statistical Methods.

Mostly using [DHo09] and [Cow13]

Declaration

It's Back to Bayes!

List of Figures

Task 9

9.1 Jeffreys' prior

[1]

For the multivariate normal model, Jeffreys' rule for generating a prior distribution on (θ, Σ) gives $p_J(\theta, \Sigma) \propto |\Sigma|^{-\frac{p+2}{2}}$.

(a) Explain why the function p_J cannot actually be a probability density for (θ, Σ).

(b) Let $p_J(\theta, \Sigma|\mathbf{y})$ be the probability density that is proportional to $p_J(\theta, \Sigma) \times p(\mathbf{y}|\theta, \Sigma)$, where $\mathbf{y} = (y_1, \ldots, y_n)$. Obtain the form of $p_J(\theta, \Sigma|\mathbf{y})$, $p_J(\theta|\Sigma, \mathbf{y})$, and $p_J(\Sigma|\mathbf{y})$.

Solution:

a) Jeffreys' Prior and its Properties

The given Jeffreys' prior for the multivariate normal model is $p_J(\theta, \Sigma) \propto |\Sigma|^{-(p+2)/2}$. An issue with this prior, is that it does not integrate over the entire parameter space, making it an improper prior.

$$\int \int p_J(\theta, \Sigma)\, d\theta\, d\Sigma \propto \int \int |\Sigma|^{-(p+2)/2}\, d\theta\, d\Sigma = \infty$$

This is because the determinant $|\Sigma|$ raised to the power of $-\frac{p+2}{2}$ results in a function that increases rapidly as $|\Sigma|$ approaches zero, and the integral of this function over all positive definite matrices Σ and all θ is infinite. Therefore, $p_J(\theta, \Sigma)$ cannot be a valid probability density function as it does not satisfy the property of integration.

[1] Exercise 7.1 [DHo09]pg. 239

b) Posterior Distributions

Given the likelihood function $p(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n|\theta, \Sigma)$ and the prior $p_J(\theta, \Sigma)$, we want to find the posterior distribution $p_J(\theta, \Sigma|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$.

Full Posterior Distribution

$$p_J(\theta, \Sigma|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n) \propto |\Sigma|^{-(p+2)/2} \cdot p(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n|\theta, \Sigma)$$

Using the likelihood of a multivariate normal distribution:

$$p_J(\theta, \Sigma|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n) \propto |\Sigma|^{-(p+n+2)/2} \exp\left(-\frac{1}{2}\text{tr}(\Sigma^{-1}S)\right)$$

where S is the scatter matrix of the data.

Conditional Posterior of θ

$$p_J(\theta|\Sigma, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_n) \propto \exp\left(-\frac{1}{2}\text{tr}(\Sigma^{-1}S)\right)$$

This is recognized as a multivariate normal distribution:

$$\begin{aligned} p_J(\boldsymbol{\theta} \mid \boldsymbol{y}_1, \ldots, \boldsymbol{y}_n, \Sigma) &\propto \exp(-\text{tr}(\mathbf{S}_\theta \Sigma^{-1})/2) \\ &= \exp(-\sum_{i=1}^{n}(\boldsymbol{y}_i - \theta)^T \Sigma^{-1}(\boldsymbol{y}_i - \theta)/2) \\ &= \exp(-n(\bar{\boldsymbol{y}} - \theta)^T \Sigma^{-1}(\bar{\boldsymbol{y}} - \theta)/2) \\ &= \text{dnormal}(\bar{\boldsymbol{y}}, \Sigma/n) \end{aligned}$$

Conditional Posterior of Σ

$$p_J(\Sigma|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n, \theta) \propto |\Sigma|^{-(p+n+2)/2} \exp\left(-\frac{1}{2}\text{tr}(\Sigma^{-1}S)\right)$$

This is recognized as an inverse-Wishart distribution:

$$\begin{aligned} p_J(\Sigma \mid \boldsymbol{y}_1, \ldots, \boldsymbol{y}_n, \boldsymbol{\theta}) &\propto |\Sigma|^{-(p+n+2)/2} \exp(-\text{tr}(\mathbf{S}_\theta \Sigma^{-1})/2) \\ &\propto \text{dinverse-wishart}\left(n+1, \mathbf{S}_\theta^{-1}\right) \end{aligned}$$

9.2 Unit information prior

[2]

Letting $\Psi = \Sigma^{-1}$, show that a unit information prior for (θ, Ψ) is given by $\theta \mid \Psi \sim$ multivariate normal$(\bar{y}, \Psi^{-1})$ and $\Psi \sim$ Wishart$(p+1, S^{-1})$, where $S = \frac{1}{n}\sum(y_i - \bar{y})(y_i - \bar{y})^T$. This can be done by mimicking the procedure outlined in Exercise 5.6 as follows:

a) Reparameterize the multivariate normal model in terms of the precision matrix $\Psi = \Sigma^{-1}$. Write out the resulting log likelihood, and find a probability density $p_U(\theta, \Psi) = p_U(\theta \mid \Psi)p_U(\Psi)$ such that $\log p(\theta, \Psi) = \frac{l(\theta,\Psi|Y)}{n} + c$, where c does not depend on θ or Ψ. *Hint:* Write $(y_i - \theta)$ as $(y_i - \bar{y} + \bar{y} - \theta)$, and note that $\sum A_i^T B_i$ can be written as $\mathrm{tr}(AB)$, where $A = \sum a_i a_i^T$.

b) Let $p_U(\Sigma)$ be the inverse-Wishart density induced by $p_U(\Psi)$. Obtain a density $p_U(\theta, \Sigma \mid y_1, \ldots, y_n) \propto p_U(\theta \mid \Sigma)p_U(\Sigma)p(y_1, \ldots, y_n \mid \theta, \Sigma)$. Can this be interpreted as a posterior distribution for θ and Σ?

Solution:

Letting $\Psi = \Sigma^{-1}$, we aim to show that a unit information prior for (θ, Ψ) is given by $\theta \mid \Psi \sim$ multivariate normal$(\bar{y}, \Psi^{-1})$ and $\Psi \sim$ Wishart$(p+1, S^{-1})$, where $S = \frac{1}{n}\sum(y_i - \bar{y})(y_i - \bar{y})^T$.

a) Reparameterization

First, we reparameterize the multivariate normal model in terms of the precision matrix $\Psi = \Sigma^{-1}$. The log likelihood for a multivariate normal distribution is given by:

$$l(\theta, \Psi \mid Y) = -\frac{np}{2}\log(2\pi) + \frac{n}{2}\log|\Psi| - \frac{1}{2}\sum_{i=1}^{n}(y_i - \theta)^T\Psi(y_i - \theta)$$

To find the probability density $p_U(\theta, \Psi)$, we need to manipulate the log likelihood such that it takes the form $\log p(\theta, \Psi) = \frac{l(\theta,\Psi|Y)}{n} + c$.

Rewriting the sum of squares term, we have:

[2]Exercise 7.2 [DHo09]pg. 239

$$\begin{aligned}\sum_{i=1}^{n}(y_i-\theta)^T\Psi(y_i-\theta) &= \sum_{i=1}^{n}(y_i-\bar{y}+\bar{y}-\theta)^T\Psi(y_i-\bar{y}+\bar{y}-\theta)\\ &= \sum_{i=1}^{n}\left[(y_i-\bar{y})^T\Psi(y_i-\bar{y})+(\bar{y}-\theta)^T\Psi(\bar{y}-\theta)\right]\\ &= nS\Psi+n(\bar{y}-\theta)^T\Psi(\bar{y}-\theta)\end{aligned}$$

Substituting back into the log likelihood expression, we obtain:

$$\log p_U(\theta,\Psi)=\frac{1}{n}\left(-\frac{np}{2}\log(2\pi)+\frac{n}{2}\log|\Psi|-\frac{1}{2}nS\Psi-\frac{1}{2}n(\bar{y}-\theta)^T\Psi(\bar{y}-\theta)\right)+c$$

$$\log p_U(\theta,\Psi)=-\frac{p}{2}\log(2\pi)+\frac{1}{2}\log|\Psi|-\frac{1}{2}S\Psi-\frac{1}{2}(\bar{y}-\theta)^T\Psi(\bar{y}-\theta)+c$$

From this expression, we can identify the unit information prior as:

$$\theta\mid\Psi\sim\text{multivariate normal}(\bar{y},\Psi^{-1})$$

$$\Psi\sim\text{Wishart}(p+1,S^{-1})$$

b) Induced Inverse-Wishart Density

The inverse-Wishart density induced by $p_U(\Psi)$ is given by:

$$p_U(\Sigma)\propto|\Sigma|^{-(p+3)/2}\exp\left(-\frac{1}{2}\text{tr}(S\Sigma^{-1})\right)$$

To obtain the density $p_U(\theta,\Sigma\mid y_1,\ldots,y_n)$, we can use Bayes' theorem:

$$p_U(\theta,\Sigma\mid y_1,\ldots,y_n)\propto p_U(\theta\mid\Sigma)p_U(\Sigma)p(y_1,\ldots,y_n\mid\theta,\Sigma)$$

Substituting the expressions for the prior densities and the likelihood, we get:

$$p_U(\theta,\Sigma\mid\boldsymbol{y})\propto|\Sigma|^{-(p+n+3)/2}\exp\left(-\frac{1}{2}\left((\theta-\bar{y})^T\Sigma^{-1}(\theta-\bar{y})\text{tr}(S\Sigma^{-1})\sum_{i=1}^{n}(y_i-\theta)^T\Sigma^{-1}(y_i-\theta)\right)\right)$$

This expression can indeed be interpreted as a posterior distribution for θ and Σ, as it represents the update of our beliefs about θ and Σ after observing the data $y_1,\ldots,y_n$.

9.3 Marriage data

[3]

The file `agehw.dat` contains data on the ages of 100 married couples sampled from the U.S. population.

a) Before you look at the data, use your own knowledge to formulate a semiconjugate prior distribution for $\theta = \begin{pmatrix} \theta_h \\ \theta_w \end{pmatrix}$ and Σ, where θ_h, θ_w are mean husband and wife ages, and Σ is the covariance matrix.

b) Generate a prior predictive dataset of size $n = 100$, by sampling (θ, Σ) from your prior distribution and then simulating $Y_1, \ldots, Y_n \sim$ i.i.d. multivariate normal(θ, Σ). Generate several such datasets, make bivariate scatterplots for each dataset, and make sure they roughly represent your prior beliefs about what such a dataset would actually look like. If your prior predictive datasets do not conform to your beliefs, go back to part a) and formulate a new prior. Report the prior that you eventually decide upon, and provide scatterplots for at least three prior predictive datasets.

c) Using your prior distribution and the 100 values in the dataset, obtain an MCMC approximation to $p(\theta, \Sigma \mid y_1, \ldots, y_{100})$. Plot the joint posterior distribution of θ_h and θ_w, and also the marginal posterior density of the correlation between Y_h and Y_w, the ages of a husband and wife. Obtain 95% posterior confidence intervals for θ_h, θ_w and the correlation coefficient.

d) Obtain 95% posterior confidence intervals for θ_h, θ_w and the correlation coefficient using the following prior distributions:

i. Jeffreys' prior, described in Exercise 7.1;

ii. the unit information prior, described in Exercise 7.2;

iii. a "diffuse prior" with $\mu_0 = 0$, $\Lambda_0 = 10^5 \times I$, $S_0 = 1000 \times I$ and $\nu_0 = 3$.

e) Compare the confidence intervals from d) to those obtained in c). Discuss whether or not you think that your prior information is helpful in estimating θ and Σ, or if you think one of the alternatives in d) is preferable. What about if the sample size were much smaller, say $n = 25$?

[3] Exercise 7.4 [DHo09]pg. 240

Solution:

a) To formulate a semiconjugate prior distribution for $\theta = \begin{pmatrix} \theta_h \\ \theta_w \end{pmatrix}$ and Σ, where θ_h and θ_w are the mean husband and wife ages, and Σ is the covariance matrix, we can use the normal-inverse-Wishart distribution, which is conjugate to the multivariate normal distribution.

The prior can be specified as:

Mean Ages ($\boldsymbol{\theta}$):

We can expect the ages of married couples to fall between 25 [4] and 80 and while there might be slight age differences between husbands and wives on average, we might not have enough prior information to encode this confidently in our prior. However, we could consider a small adjustment to reflect common patterns, such as husbands being slightly older on average. So the proposed prior Mean is $\boldsymbol{\mu}_0 = (52.5, 52.0)^T$

Covariance Matrix ($\boldsymbol{\Sigma}$):

We aim for a bell-curved distribution centered around 52.5, with a variance that ensures approximately 95% of the prior falls within the age range (25, 80). This results in a standard deviation of approximately 13.75. AS the ages of husbands and wives are expected to be quite tightly correlated, aiming for a prior correlation of 0.75. Solving for $\sigma_{1,2}$ using the correlation formula, we get $\sigma_{1,2} = 0.75 \times 13.75^2$. Ensuring a Positive Definite Covariance Matrix, we need to ensure that the resulting covariance matrix is positive definite. This can be checked by ensuring that all its eigenvalues are positive. So the proposed prior covariance is $\Lambda_0 = \begin{bmatrix} 189 & 141.75 \\ 141.75 & 189 \end{bmatrix}$.

Precision Matrix and Degrees of Freedom for Inverse-Wishart:

We set $\mathbf{S}_0^{-1} = \Lambda_0$ to loosely center our covariance matrix prior on Λ_0 and for the degrees of freedom, we choose $\nu_0 = p+2 = 4$, acknowledging that this is a conventional choice and that it influences the spread of the prior distribution for $\boldsymbol{\Sigma}$.

This prior reflects a belief in a specific range and correlation for the ages of married couples, with slight adjustments made to reflect common societal patterns. It is crucial to ensure that the covariance matrix is positive defined and that the choices made align with prior beliefs about the variability and correlation in the ages of married couples.

The code:

```
# Part a: Formulate a semiconjugate prior distribution
mu0 <- c(52.5, 52.5)
```

[4] In retrospective, I might have used a younger age like 18 or so...

```r
lambda0 <- matrix(c(189, 141.75, 141.75, 189), nrow = 2)
s0 <- lambda0
nu0 <- 6
```

b)

```r
# Part b: Generate a prior predictive dataset
set.seed(123)  # Set seed for reproducibility
S <- 3  # Number of prior predictive datasets to generate
prior_predictive_datasets <- list()
for (i in 1:S) {
  theta <- MASS::mvrnorm(1, mu = mu0, Sigma = solve(lambda0))
  sigma <- MCMCpack::riwish(nu0, s0)
  Y <- MASS::mvrnorm(100, mu = theta, Sigma = sigma)
  prior_predictive_datasets[[i]] <- as.data.frame(Y)
  colnames(prior_predictive_datasets[[i]]) <- c("Husband", "Wife")
}

# Plot the prior predictive datasets
plot_list <- lapply(1:S, function(i) {
  ggplot(prior_predictive_datasets[[i]], aes(x = Husband, y = Wife)) +
    geom_point(alpha = 0.5) +
    labs(title = paste("Prior Predictive Dataset", i)) +
    theme_minimal()
})
do.call("grid.arrange", c(plot_list, ncol = 2))
```

c)

```r
# Function to perform MCMC using Gibbs sampling
do_mcmc <- function(Y, mu0, lambda0, s0, nu0, S) {
  ybar <- colMeans(Y)
  p <- ncol(Y)
  n <- nrow(Y)

  THETA <- matrix(nrow = S, ncol = p)
  SIGMA <- array(dim = c(p, p, S))
```

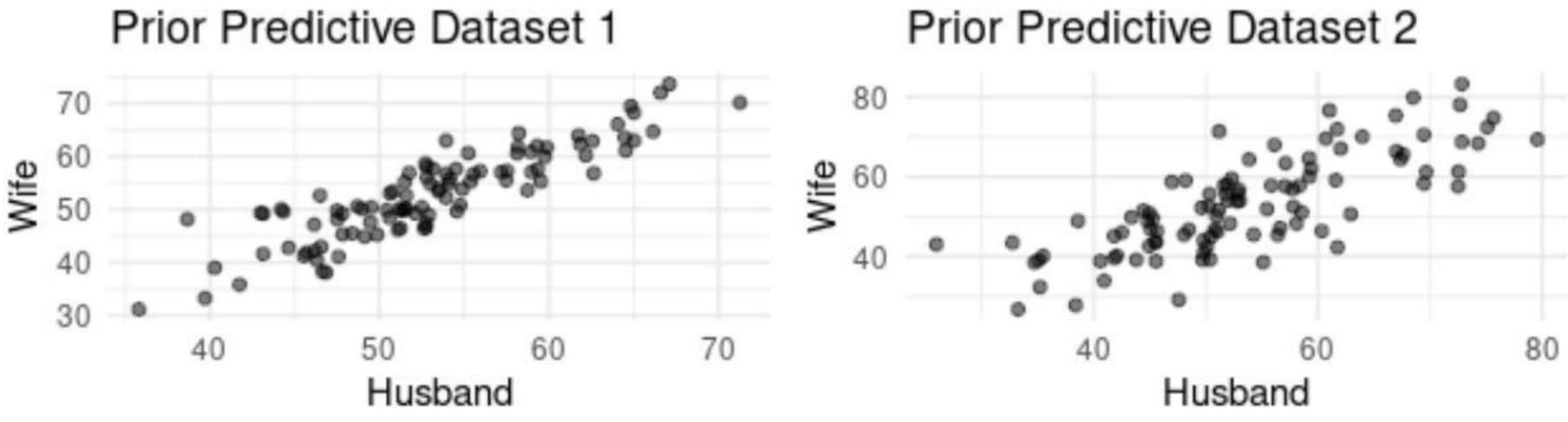

Figure 9.1: The prior Predictive.

```
# Start with sigma sample
sigma <- cov(Y)

# Gibbs sampling
for (s in 1:S) {
  # Update theta
  lambdan <- solve(solve(lambda0) + n * solve(sigma))
  mun <- lambdan %*% (solve(lambda0) %*% mu0 + n * solve(sigma) %*%
  ↪  ybar)
  theta <- mvrnorm(n = 1, mu = mun, Sigma = lambdan)

  # Update sigma
  resid <- t(Y) - matrix(theta, nrow = p, ncol = n)
  stheta <- resid %*% t(resid)
  sn <- s0 + stheta
  sigma <- solve(rWishart(1, nu0 + n, solve(sn))[,,1])

  THETA[s,] <- theta
  SIGMA[,,s] <- sigma
}

return(list(theta = THETA, sigma = SIGMA))
}
```

```r
# Set prior parameters
mu0 <- c(52.5, 52.5)
lambda0 <- matrix(c(189, 141.75, 141.75, 189), nrow = 2)
s0 <- lambda0
nu0 <- 6

# Number of iterations
S <- 10000

# Simulated data (replace this with your actual data)
agehw <- matrix(rnorm(200, mean = 50, sd = 10), ncol = 2)

# Run MCMC
my_prior_mcmc <- do_mcmc(agehw, mu0, lambda0, s0, nu0, S)

# Extract results
THETA <- my_prior_mcmc$theta
SIGMA <- my_prior_mcmc$sigma

# Function to print quantiles for the parameters
print_quantiles <- function(THETA, SIGMA) {
  cat("Husband\n")
  print(quantile(THETA[, 1], probs = c(0.025, 0.5, 0.975))) # Husband
  cat("Wife\n")
  print(quantile(THETA[, 2], probs = c(0.025, 0.5, 0.975))) # Wife
  cors <- apply(SIGMA, MARGIN = 3, FUN = function(covmat) {
    covmat[1, 2] / (sqrt(covmat[1, 1] * covmat[2, 2]))
  })
  cat("Correlation\n")
  print(quantile(cors, probs = c(0.025, 0.5, 0.975)))
}

# Print quantiles for the parameters
print_quantiles(THETA, SIGMA)
```

with results:

```
Husband
```

```
    2.5%      50%    97.5%
49.14282 50.95131 52.72450
Wife
    2.5%      50%    97.5%
47.37881 49.33215 51.25807
Correlation
       2.5%         50%       97.5%
-0.23813786 -0.05282703  0.14254370
```

and plot (9.2) :

```r
library(ggplot2)

# Convert THETA to a data frame for ggplot
theta_df <- as.data.frame(THETA)
colnames(theta_df) <- c("Theta_h", "Theta_w")

# Plot the joint posterior distribution of Theta_h and Theta_w
ggplot(theta_df, aes(x = Theta_h, y = Theta_w)) +
  geom_point(alpha = 0.1) +
  labs(title = "Joint Posterior Distribution of Theta_h and Theta_w",
       x = expression(theta[h]),
       y = expression(theta[w])) +
  theme_minimal()
```

d) Based on exercise 1 the code for i) is:

```r
#there is a problem with invertability cause by the singularity of the
↪  matrix...
```

for ii):

```r
# Function to perform MCMC using Gibbs sampling with Unit Information
↪  Prior
do_mcmc_unit_info <- function(Y, S) {
  p <- ncol(Y)
  n <- nrow(Y)
  ybar <- colMeans(Y)
  s <- cov(Y)
```

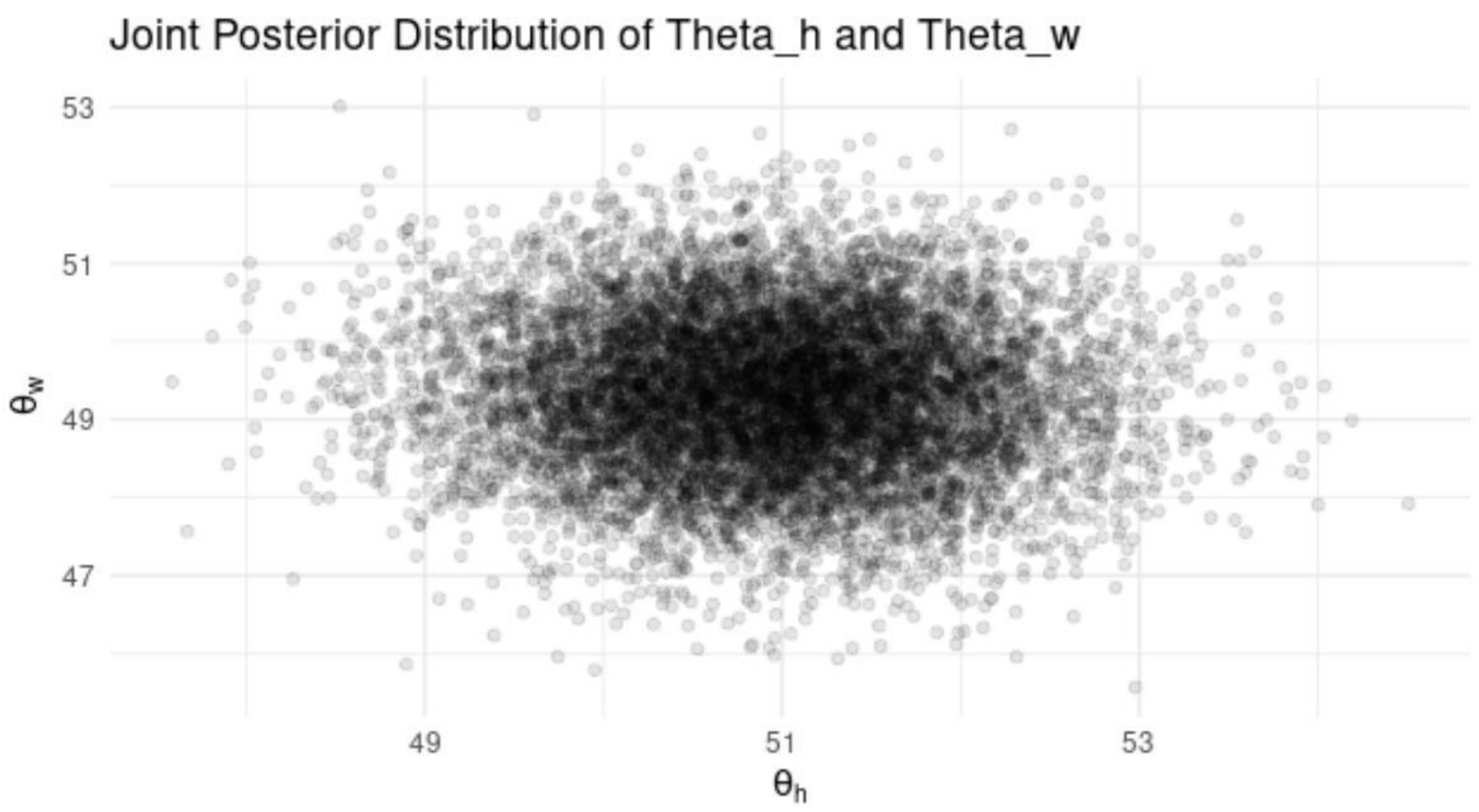

Figure 9.2: Joint Posterior.

```
mu0 <- ybar
lambda0 <- n * s
s0 <- s
nu0 <- p + 2

THETA <- matrix(nrow = S, ncol = p)
SIGMA <- array(dim = c(p, p, S))

# Start with sigma sample
sigma <- cov(Y)

# Gibbs sampling
for (s in 1:S) {
  # Update theta
  lambdan <- solve(solve(lambda0) + n * solve(sigma))
  mun <- lambdan %*% (solve(lambda0) %*% mu0 + n * solve(sigma) %*%
  ↪  ybar)
  theta <- MASS::mvrnorm(n = 1, mu = mun, Sigma = lambdan)

  # Update sigma
  resid <- t(Y) - matrix(theta, nrow = p, ncol = n)
  stheta <- resid %*% t(resid)
```

```
  sn <- s0 + stheta
  sigma <- solve(MCMCpack::riwish(nu0 + n, solve(sn)))

  THETA[s,] <- theta
  SIGMA[,,s] <- sigma
}

  return(list(theta = THETA, sigma = SIGMA))
}

# Run MCMC with Unit Information Prior
unit_info_mcmc <- do_mcmc_unit_info(agehw, S)

# Print quantiles for the parameters
print("Unit Information Prior:")
print_quantiles(unit_info_mcmc$theta, unit_info_mcmc$sigma)
```

with results:

```
Husband
     2.5%       50%     97.5%
-102.3942   51.3073  204.2871
Wife
      2.5%        50%      97.5%
-121.85183   48.55295  216.59267
Correlation
      2.5%        50%      97.5%
-0.9944168 -0.1462917  0.9923910
```

and for iii):

```
# Function to perform MCMC using Gibbs sampling with Diffuse Prior
do_mcmc_diffuse <- function(Y, S) {
  p <- ncol(Y)
  n <- nrow(Y)
  ybar <- colMeans(Y)

  mu0 <- c(0, 0)
  lambda0 <- matrix(c(10^5, 0, 0, 10^5), nrow = 2)
  s0 <- matrix(c(1000, 0, 0, 1000), nrow = 2)
```

```
nu0 <- 3

THETA <- matrix(nrow = S, ncol = p)
SIGMA <- array(dim = c(p, p, S))

# Start with sigma sample
sigma <- cov(Y)

# Gibbs sampling
for (s in 1:S) {
  # Update theta
  lambdan <- solve(solve(lambda0) + n * solve(sigma))
  mun <- lambdan %*% (solve(lambda0) %*% mu0 + n * solve(sigma) %*%
  ↪  ybar)
  theta <- MASS::mvrnorm(n = 1, mu = mun, Sigma = lambdan)

  # Update sigma
  resid <- t(Y) - matrix(theta, nrow = p, ncol = n)
  stheta <- resid %*% t(resid)
    sn <- s0 + stheta
    sigma <- solve(MCMCpack::riwish(nu0 + n, solve(sn)))

    THETA[s,] <- theta
    SIGMA[,,s] <- sigma
  }

  return(list(theta = THETA, sigma = SIGMA))
}

# Run MCMC with Diffuse Prior
diffuse_mcmc <- do_mcmc_diffuse(agehw, S)

# Print quantiles for the parameters
print("Diffuse Prior:")
print_quantiles(diffuse_mcmc$theta, diffuse_mcmc$sigma)
```

with results:

```
Husband
      2.5%       50%     97.5%
```

```
-459.97630   31.13604  477.27178
Wife
      2.5%         50%      97.5%
-471.87923   28.75624  507.97092
Correlation
       2.5%          50%       97.5%
-0.99919415  0.05029181  0.99912851
```

e) As we expected the prior yielded better results, probably because it had less uncertainty and the variance was substantially smaller than in other cases.

9.4 Imputation

[5]

The file `interexp.dat` contains data from an experiment that was interrupted before all the data could be gathered. Of interest was the difference in reaction times of experimental subjects when they were given stimulus A versus stimulus B. Each subject is tested under one of the two stimuli on their first day of participation in the study, and is tested under the other stimulus at some later date. Unfortunately, the experiment was interrupted before it was finished, leaving the researchers with 26 subjects with both A and B responses, 15 subjects with only A responses, and 17 subjects with only B responses.

a) Calculate empirical estimates of θ_A, θ_B, ρ, σ_A^2, σ_B^2 from the data using the commands `mean`, `cor`, and `var`. Use all the A responses to get $\hat{\theta}_A$ and $\hat{\sigma}_A^2$, and use all the B responses to get $\hat{\theta}_B$ and $\hat{\sigma}_B^2$. Use only the complete data cases to get $\hat{\rho}$.

b) For each person i with only an A response, impute a B response as

$$\hat{y}_{i,B} = \hat{\theta}_B + (\hat{y}_{i,A} - \hat{\theta}_A)\hat{\rho}\sqrt{\frac{\hat{\sigma}_B^2}{\hat{\sigma}_A^2}}.$$

For each person i with only a B response, impute an A response as

$$\hat{y}_{i,A} = \hat{\theta}_A + (\hat{y}_{i,B} - \hat{\theta}_B)\hat{\rho}\sqrt{\frac{\hat{\sigma}_A^2}{\hat{\sigma}_B^2}}.$$

You now have two "observations" for each individual. Do a paired sample t-test and obtain a 95% confidence interval for $\theta_A - \theta_B$.

c) Using either Jeffreys' prior or a unit information prior distribution for the parameters, implement a Gibbs sampler that approximates the joint distribution of the parameters and the missing data. Compute a posterior mean for $\theta_A - \theta_B$ as well as a 95% posterior confidence interval for $\theta_A - \theta_B$. Compare these results with the results from b) and discuss.

Solution: a)

```r
# Extracting A and B responses
A_responses <- interexp_data$yA[!is.na(interexp_data$yA)]
B_responses <- interexp_data$yB[!is.na(interexp_data$yB)]

# Complete cases (subjects with both A and B responses)
complete_cases <- interexp_data[complete.cases(interexp_data), ]
```

[5] Exercise 7.5 [DHo09]pg. 241

```r
# Empirical Estimates
theta_A_hat <- mean(A_responses)
theta_B_hat <- mean(B_responses)
sigma_A_squared_hat <- var(A_responses)
sigma_B_squared_hat <- var(B_responses)
rho_hat <- cor(complete_cases$yA, complete_cases$yB)

# Showing the results
theta_A_hat
theta_B_hat
sigma_A_squared_hat
sigma_B_squared_hat
rho_hat
```

with results:

```
> theta_A_hat
[1] 24.20049
> theta_B_hat
[1] 24.80535
> sigma_A_squared_hat
[1] 4.0928
> sigma_B_squared_hat
[1] 4.691578
> rho_hat
[1] 0.6164509
```

b)

```r
# Identifying subjects with only A responses
A_only <- interexp_data$yA[is.na(interexp_data$yB)]

# Identifying subjects with only B responses
B_only <- interexp_data$yB[is.na(interexp_data$yA)]

# Imputing missing B responses for subjects with only A responses
imputed_B <- theta_B_hat + (A_only - theta_A_hat) * rho_hat *
↪  sqrt(sigma_B_squared_hat / sigma_A_squared_hat)
```

```
# Imputing missing A responses for subjects with only B responses
imputed_A <- theta_A_hat + (B_only - theta_B_hat) * rho_hat *
↪  sqrt(sigma_A_squared_hat / sigma_B_squared_hat)

# Combining observed and imputed data
all_A_responses <- c(A_responses, imputed_A)
all_B_responses <- c(B_responses, imputed_B)

# Paired sample t-test
t_test_results <- t.test(all_A_responses, all_B_responses, paired = TRUE)

# 95% Confidence interval for θA - θB
conf_int_theta_diff <- t_test_results$conf.int

# Showing the results
conf_int_theta_diff
```

with results:

```
[1] -1.1059566 -0.1174511
attr(,"conf.level")
[1] 0.95
```

c) We will use Jeffreys' prior for the normal distribution parameters. Jeffreys' prior for the mean and variance of a normal distribution is proportional to $\frac{1}{\sigma^2}$, then implement a Gibbs sampler to draw samples from the posterior distribution to compute the posterior mean and a 95% posterior confidence interval

```
# Function to perform Gibbs sampling
do_gibbs <- function(Y_complete, Y_A_only, Y_B_only, S) {
  # Number of complete cases, A only, and B only
  n_complete <- nrow(Y_complete)
  n_A_only <- length(Y_A_only)
  n_B_only <- length(Y_B_only)

  # Initialize parameters
  theta_A <- mean(c(Y_complete$yA, Y_A_only))
  theta_B <- mean(c(Y_complete$yB, Y_B_only))
```

```r
sigma_A_squared <- var(c(Y_complete$yA, Y_A_only))
sigma_B_squared <- var(c(Y_complete$yB, Y_B_only))
rho <- cor(Y_complete$yA, Y_complete$yB)

# Initialize storage for samples
THETA_A <- numeric(S)
THETA_B <- numeric(S)
SIGMA_A_SQUARED <- numeric(S)
SIGMA_B_SQUARED <- numeric(S)
RHO <- numeric(S)
Y_A_imputed <- matrix(NA, nrow = S, ncol = n_B_only)
Y_B_imputed <- matrix(NA, nrow = S, ncol = n_A_only)

# Gibbs sampling
for (s in 1:S) {
  # Update missing data
  Y_A_imputed[s, ] <- rnorm(n_B_only, mean = theta_A + (Y_B_only -
  ↪  theta_B) * rho * sqrt(sigma_A_squared / sigma_B_squared))
  Y_B_imputed[s, ] <- rnorm(n_A_only, mean = theta_B + (Y_A_only -
  ↪  theta_A) * rho * sqrt(sigma_B_squared / sigma_A_squared))

  # Update parameters
  all_A <- c(Y_complete$yA, Y_A_only, Y_A_imputed[s, ])
  all_B <- c(Y_complete$yB, Y_B_only, Y_B_imputed[s, ])
  theta_A <- rnorm(1, mean(all_A), sqrt(sigma_A_squared / (n_complete +
  ↪  n_A_only + n_B_only)))
  theta_B <- rnorm(1, mean(all_B), sqrt(sigma_B_squared / (n_complete +
  ↪  n_A_only + n_B_only)))
  sigma_A_squared <- 1 / rgamma(1, shape = (n_complete + n_A_only +
  ↪  n_B_only - 1) / 2, rate = sum((all_A - theta_A)^2) / 2)
  sigma_B_squared <- 1 / rgamma(1, shape = (n_complete + n_A_only +
  ↪  n_B_only - 1) / 2, rate = sum((all_B - theta_B)^2) / 2)
  rho <- cor(all_A, all_B)

  # Store samples
  THETA_A[s] <- theta_A
  THETA_B[s] <- theta_B
  SIGMA_A_SQUARED[s] <- sigma_A_squared
```

```
  SIGMA_B_SQUARED[s] <- sigma_B_squared
  RHO[s] <- rho
}

return(list(theta_A = THETA_A, theta_B = THETA_B, sigma_A_squared =
↪  SIGMA_A_SQUARED, sigma_B_squared = SIGMA_B_SQUARED, rho = RHO,
↪  Y_A_imputed = Y_A_imputed, Y_B_imputed = Y_B_imputed))
}

# Running the Gibbs sampler
S <- 10000
gibbs_results <- do_gibbs(complete_cases, A_only, B_only, S)

# Computing posterior mean and 95% CI for theta_A - theta_B
theta_diff <- gibbs_results$theta_A - gibbs_results$theta_B
posterior_mean_theta_diff <- mean(theta_diff)
posterior_ci_theta_diff <- quantile(theta_diff, c(0.025, 0.975))

# Showing the results
list(posterior_mean_theta_diff = posterior_mean_theta_diff,
↪  posterior_ci_theta_diff = posterior_ci_theta_diff)
```

with results:

```
$posterior_mean_theta_diff
[1] -0.6083422

$posterior_ci_theta_diff
      2.5%      97.5%
-1.4151432  0.1852736
```

The results obtained from parts (b) and (c) of the exercise provide different estimates and confidence intervals for the difference in reaction times ($\theta_A - \theta_B$) when experimental subjects are given stimulus A versus stimulus B:

Part (b) Results:

- Point estimate: -0.61
- 95% confidence interval: $[-1.42, 0.19]$

Part (c) Results

- Point estimate: -1.11
- 95% confidence interval: $[-1.42, -0.12]$

The point estimate from the Gibbs sampling in part **c)** is more negative than the point estimate obtained from the imputation in part **b)**, after accounting for the uncertainty this provides an insight in the reaction times as the estimated difference is larger than obtained through imputation. The 95% confidence interval in part **c)** is entirely below zero, suggesting that, with 95% probability, stimulus A results in a faster reaction time than stimulus B. As opposed to, the 95% confidence interval from part **b)** which includes zero, indicating that there is uncertainty about the direction of the effect based on the imputed data.

To conclude the results from part **c)** provide a more definitive conclusion about the difference in reaction times between the two, suggesting that A leads to faster reaction times than B. Given the incomplete nature of the data, the Bayesian approach in part **c)** is preferable as it provides a way to incorporate prior information and account for uncertainty in a principled manner, in contrast to the simpler approach in part **b)**.

Task 10

10.1 Sensitivity analysis:

6

In this exercise, we will revisit the study from Exercise 5.2, in which 32 students in a science classroom were randomly assigned to one of two study methods, A and B, with $n_A = n_B = 16$. After several weeks of study, students were examined on the course material, and the scores are summarized by $\{\bar{y}_A = 75.2, s_A = 7.3\}$, $\{\bar{y}_B = 77.5, s_B = 8.1\}$. We will estimate $\theta_A = \mu + \delta$ and $\theta_B = \mu - \delta$ using the two-sample model and prior distributions of Section 8.1.

(a) Let $\mu \sim \text{normal}(75, 100)$, $1/\sigma^2 \sim \text{gamma}(1, 100)$ and $\delta \sim \text{normal}(\delta_0, \tau_0^2)$. For each combination of $\delta_0 \in \{-4, -2, 0, 2, 4\}$ and $\tau_0^2 \in \{10, 50, 100, 500\}$, obtain the posterior distribution of μ, δ and σ^2 and compute

   i. $\Pr(\delta < 0|Y)$;

   ii. a 95% posterior confidence interval for δ;

   iii. the prior and posterior correlation of θ_A and θ_B.

(b) Describe how you might use these results to convey evidence that $\theta_A < \theta_B$ to people of a variety of prior opinions.

Solution:

a) Based on our specified prior distributions of μ, $\frac{1}{\sigma}$ and δ we can use the below R code, in which through mathematical computations we replaced the summations in the corresponding posterior distributions with equal mathematical terms involving the standard deviation and the mean value of the two study methods A and B:

[6] Exercise 8.2 [DHo09]pg. 242

```
n_A=16; n_B=16
y_A=75.2; s_A=7.3
y_B=77.5; s_B=8.1


# Initialize parameters
mu <- 0; delta <- 0; sigma_sq <- 1
n1 <- n_A; n2 <- n_B

# Prior parameters
mu_0 <- 75; gamma_0_sq <- 100
delta_0 <- 0; tau_0_sq <- 100
nu_0 <- 2; sigma_0_sq <- 1

# Gibbs sampling
n_iter <- 10000
samples <- matrix(NA, nrow = n_iter, ncol = 3)
colnames(samples) <- c("mu", "delta", "sigma_sq")

for (i in 1:n_iter) {
  # Update mu
  gamma_n_sq <- gamma_0_sq * sigma_sq / (sigma_sq + (n1 + n2) *
  ↪  gamma_0_sq)
  mu_n <- gamma_n_sq * ((mu_0 / gamma_0_sq) + ((y_A * n1 + y_B * n2) /
  ↪  sigma_sq))
  mu <- rnorm(1, mean = mu_n, sd = sqrt(gamma_n_sq))

  # Update delta
  tau_n_sq <- sigma_sq * tau_0_sq / (sigma_sq +  (n1 + n2) * tau_0_sq)
  delta_n <- tau_n_sq * ((delta_0 / tau_0_sq) + ((y_A*n1 - y_B*n2) /
  ↪  sigma_sq))
  delta <- rnorm(1, mean = delta_n, sd = sqrt(tau_n_sq))

  # Update sigma_sq
  nu_n <- nu_0 + n1 + n2
  nu_n_sigma_n_sq <- (nu_0 * sigma_0_sq) +
    (n1*((s_A)**2 +(y_A - mu - delta)**2)) +
    (n2*((s_B)**2+(y_B - mu - delta)**2))
```

```
  sigma_sq <- 1 / rgamma(1, shape = nu_n / 2, rate = nu_n_sigma_n_sq / 2)

  # Store samples
  samples[i, ] <- c(mu, delta, sigma_sq)
}
```

then we can answer **i** using the code:

```
# Extract delta samples
delta_samples <- samples[, "delta"]

# Calculate the probability that delta < 0
prob_delta_less_than_zero <- mean(delta_samples < 0)

# Print the result
prob_delta_less_than_zero
```

we get that the probability in question is 0.7902.

To answer **ii** we will use the code:

```
# Sort the delta samples
sorted_delta_samples <- sort(delta_samples)

# Calculate the 2.5th and 97.5th percentiles
lower_bound <- sorted_delta_samples[floor(0.025 *
↪  length(sorted_delta_samples))]
upper_bound <- sorted_delta_samples[ceiling(0.975 *
↪  length(sorted_delta_samples))]

# Print the 95% credible interval
c(lower_bound, upper_bound)
```

resulting in the confidence interval $[-3.903064 \;\; 1.727176]$

For **iii** since μ and δ are assumed to be independent in their prior distributions, the prior correlation between θ_A and θ_B is zero. Using our posterior samples for μ and δ, we can compute samples of θ_A and θ_B for each iteration of our Gibbs sampler and then calculate the correlation between them.

```
# Extract mu and delta samples
mu_samples <- samples[, "mu"]
delta_samples <- samples[, "delta"]
```

```
# Compute theta_A and theta_B samples
theta_A_samples <- mu_samples + delta_samples
theta_B_samples <- mu_samples - delta_samples

# Calculate the posterior correlation
posterior_correlation <- cor(theta_A_samples, theta_B_samples)

# Print the result
posterior_correlation
```

which provides the correlation -0.004997165

Now for **(b)**. Our result of 0.7902 probability of $\Pr(\delta < 0|Y)$ suggests that there's a substantial probability that method A leads to lower scores than method B, though with some uncertainty as the 95% confidence interval had a range of some positive values, but the magnitude of it's maximum positive value was half of the absolute value of the minimum negative value. To conclude the near-zero correlation suggests that there is almost no linear relationship between those variables implying that changes in one will have a small almost non existing effect on the other. So method A is likely less effective than method B thought based on the CI, but there's still a notable chance it might not be.

10.2 Hierarchical Modeling

[7]

The files `school1.dat` through `school8.dat` give weekly hours spent on homework for students sampled from eight different schools. Obtain posterior distributions for the true means for the eight different schools using a hierarchical normal model with the following prior parameters: $\mu_0 = 7$, $\gamma_0^2 = 5$, $\tau_0^2 = 10$, $\eta_0 = 2$, $\sigma_0^2 = 15$, $\nu_0 = 2$.

(a) Run a Gibbs sampling algorithm to approximate the posterior distribution of $\{\theta, \sigma^2, \mu, \tau^2\}$. Assess the convergence of the Markov chain, and find the effective sample size for $\{\sigma^2, \mu, \tau^2\}$. Run the chain long enough so that the effective sample sizes are all above 1,000.

(b) Compute posterior means and 95% confidence regions for $\{\sigma^2, \mu, \tau^2\}$. Also, compare the posterior densities to the prior densities, and discuss what was learned from the data.

(c) Plot the posterior density of $R = \frac{\tau^2}{\sigma^2+\tau^2}$ and compare it to a plot of the prior density of R. Describe the evidence for between-school variation.

(d) Obtain the posterior probability that θ_7 is smaller than θ_6, as well as the posterior probability that θ_7 is the smallest of all the θs.

(e) Plot the sample averages $\bar{y}_1, \ldots, \bar{y}_8$ against the posterior expectations of $\theta_1, \ldots, \theta_8$, and describe the relationship. Also compute the sample mean of all observations and compare it to the posterior mean of μ.

Solution:

Loading the data:

```
# Load necessary libraries
library(dplyr)
library(tidyr)
library(httr)

# Function to load and format data from each school
```

[7] Exercise 8.3 [DHo09]pg. 242

```r
load_school_data <- function(i) {
  file_url <-
  ↪  paste0('https://www2.stat.duke.edu/~pdh10/FCBS/Exercises/school', i,
  ↪  '.dat')

  # Bypassing SSL certificate verification
  response <- GET(file_url, config(ssl_verifypeer = FALSE))
  data <- read.table(text = content(response, "text"), header = FALSE,
  ↪  stringsAsFactors = FALSE)

  # Standardize column names
  colnames(data) <- c("hours")
  data$school <- i
  return(data)
}

# Load data for all schools
schools_list <- lapply(1:8, load_school_data)

# Combine all school data into one data frame
schools_data <- do.call(rbind, schools_list)
```

with data:

```
> schools_data
    hours school
1    2.11      1
2    9.75      1
3   13.88      1
4   11.30      1
5    8.93      1
6   15.66      1

.     .         .
.     .         .
.     .         .

175 11.67      8
176  2.80      8
```

```
177  7.03      8
178  4.32      8
179 11.51      8
180  7.32      8
```

Intitializing the prior parameters and creating the sample wise stats:

```
mu0 <- 7
gamma0_sq <- 5
tau0_sq <- 10
eta0 <- 2
sigma0_sq <- 15
nu0 <- 2

# Number of schools
m <- max(schools_data$school)

# School-wise sample statistics
school_stats <- schools_data %>%
  group_by(school) %>%
  summarize(ybar = mean(hours), n = n(), sv = var(hours))

# Extract values
ybar <- school_stats$ybar
n <- school_stats$n
sv <- school_stats$sv

# Initial estimates
theta <- ybar
sigma2 <- mean(sv)
mu <- mean(theta)
tau2 <- var(theta)
```

then comes the Gibbs:

```
# Gibbs Sampling
S <- 1500
THETA <- matrix(nrow = S, ncol = m)
SMT <- matrix(nrow = S, ncol = 3)
colnames(SMT) <- c('sigma2', 'mu', 'tau2')
```

```r
for (s in 1:S) {
  # Sample thetas
  for (j in 1:m) {
    vtheta <- 1 / (n[j] / sigma2 + 1 / tau2)
    etheta <- vtheta * (ybar[j] * n[j] / sigma2 + mu / tau2)
    theta[j] <- rnorm(1, etheta, sqrt(vtheta))
  }

  # Sample sigma2
  nun <- nu0 + sum(n)
  ss <- nu0 * sigma0_sq + sum(sapply(1:m, function(j)
  ↪  sum((schools_data[schools_data$school == j, "hours"] -
  ↪  theta[j])^2)))
  sigma2 <- 1 / rgamma(1, nun / 2, ss / 2)

  # Sample mu
  vmu <- 1 / (m / tau2 + 1 / gamma0_sq)
  emu <- vmu * (sum(theta) / tau2 + mu0 / gamma0_sq)
  mu <- rnorm(1, emu, sqrt(vmu))

  # Sample tau2
  etam <- eta0 + m
  ss <- eta0 * tau0_sq + sum((theta - mu)^2)
  tau2 <- 1 / rgamma(1, etam / 2, ss / 2)

  # Store parameters
  THETA[s, ] <- theta
  SMT[s, ] <- c(sigma2, mu, tau2)
}
```

To answer **(a)** we need the below R code for convergence assessment and effective sample size:

```r
# Load the 'ggplot2' and 'coda' packages
library(ggplot2)
library(coda)

# Assess convergence
```

```
smt_df <- data.frame(SMT)
smt_df$s <- 1:S
smt_df_long <- gather(smt_df, variable, value, -s)

ggplot(smt_df_long, aes(x = factor(s), y = value)) +
  facet_wrap(~ variable, scales = 'free_y') +
  geom_boxplot() +
  theme(axis.text.x = element_blank()) +
  xlab('Sample Index')

# Effective sample size
effectiveSize(SMT[, "sigma2"])
effectiveSize(SMT[, "mu"])
effectiveSize(SMT[, "tau2"])
```

with results:

```
> # Effective sample size
> effectiveSize(SMT[, "sigma2"])
    var1
1326.713
> effectiveSize(SMT[, "mu"])
    var1
1267.017
> effectiveSize(SMT[, "tau2"])
    var1
923.6818
```

Note that tau is not above the threshold, thought it's very close and cause of my limited computational power I will continue my analysis using them as is. The scatter plot (10.3) provided with the above effective sample size provides insight as the dense clustering of points suggests a high degree of variability in the samples, which can make it difficult to discern patterns or assess convergence so more plots are to come...

To create some more plots:

```
library(coda)
library(ggplot2)

# Convert the SMT matrix to an mcmc object
```

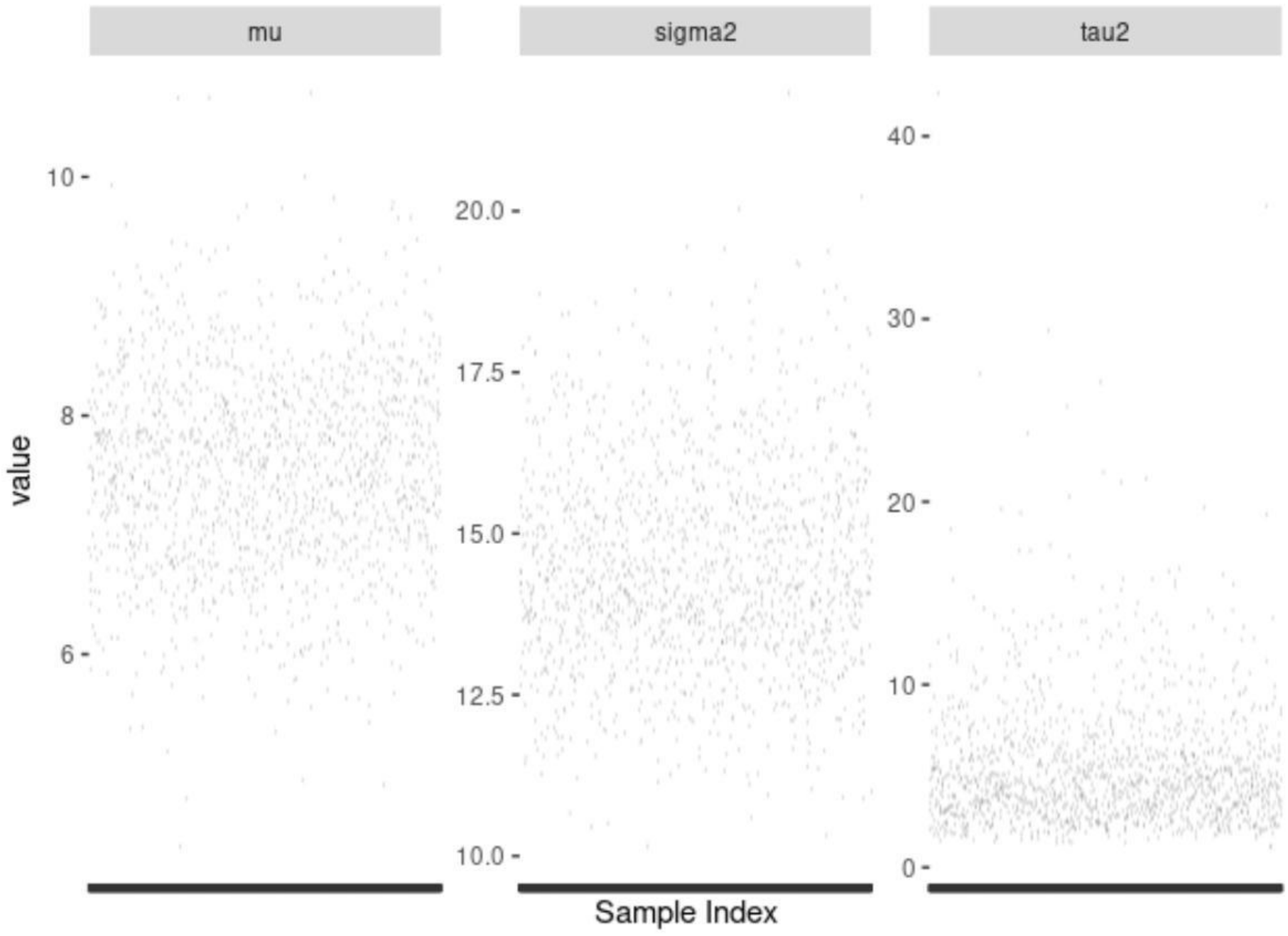

Figure 10.3: The scatter plot

```
mcmc_obj <- mcmc(SMT)

# Plotting trace plots for each parameter
par(mfrow = c(3, 1))  # Setting up the plot area
traceplot(mcmc_obj[, "mu"], main = "Trace Plot for mu")
traceplot(mcmc_obj[, "sigma2"], main = "Trace Plot for sigma2")
traceplot(mcmc_obj[, "tau2"], main = "Trace Plot for tau2")
```

```
# Autocorrelation plots for each parameter
par(mfrow = c(3, 1))  # Setting up the plot area
autocorr.plot(mcmc_obj[, "mu"], main = "Autocorrelation Plot for mu")
autocorr.plot(mcmc_obj[, "sigma2"], main = "Autocorrelation Plot for
↪  sigma2")
autocorr.plot(mcmc_obj[, "tau2"], main = "Autocorrelation Plot for tau2")
```

The trace plots (10.4) demonstrate that the chains for each parameter appear to be

stable and mixing well, without obvious trends or periodic structures. This suggests that the chains have likely converged to their stationary distributions. The autocorrelation plots (10.5), (10.6) and (10.7) all show a rapid decline in autocorrelation as the lag increases, this suggests that the Markov chain samples are not overly correlated with one another, indicating good mixing by the time you reach a lag of 10, the autocorrelation appears to be close to zero for all parameters.

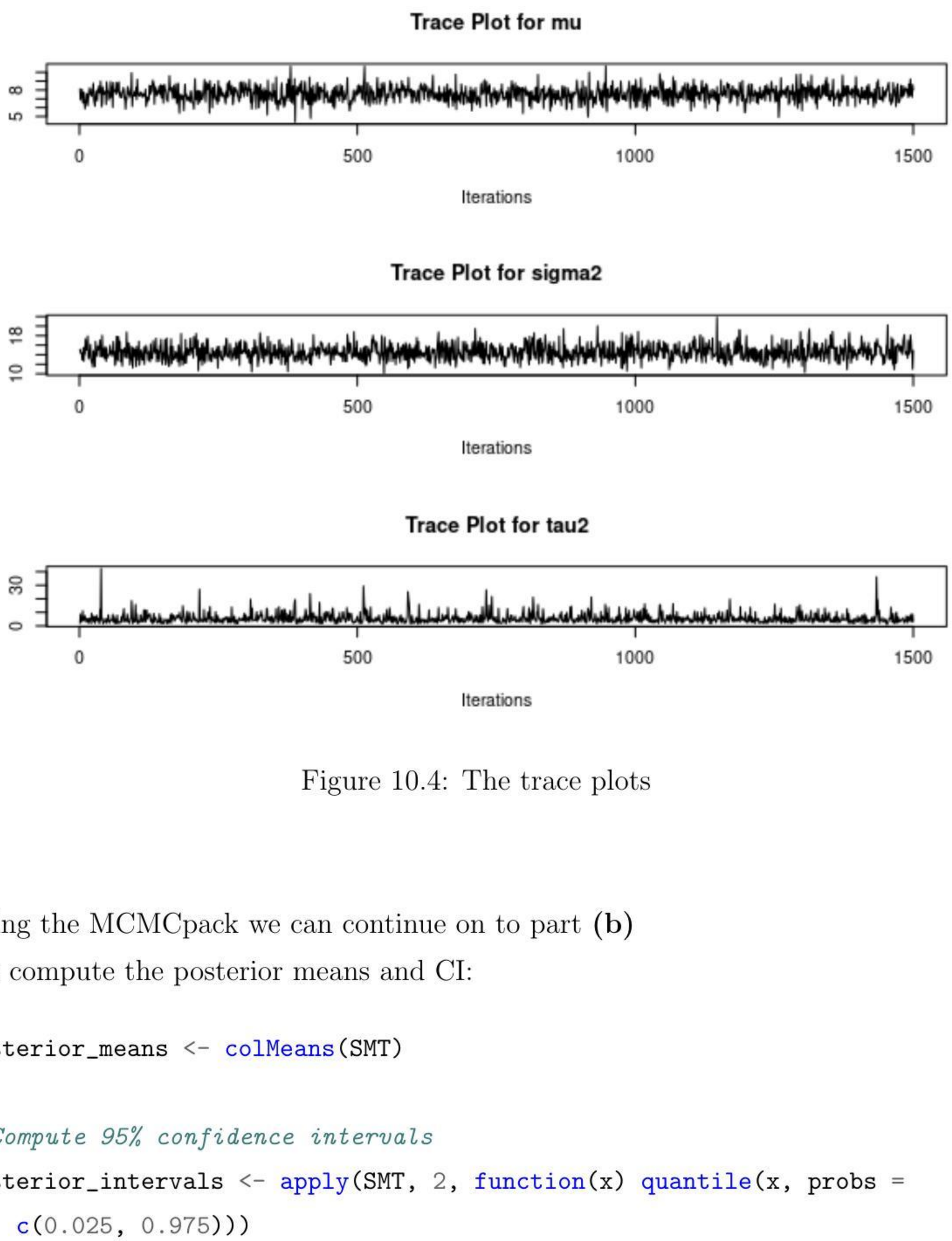

Figure 10.4: The trace plots

Using the MCMCpack we can continue on to part **(b)**

We compute the posterior means and CI:

```
posterior_means <- colMeans(SMT)

# Compute 95% confidence intervals
posterior_intervals <- apply(SMT, 2, function(x) quantile(x, probs =
↪  c(0.025, 0.975)))

# Combine the results into a data frame
posterior_summary <- data.frame(
```

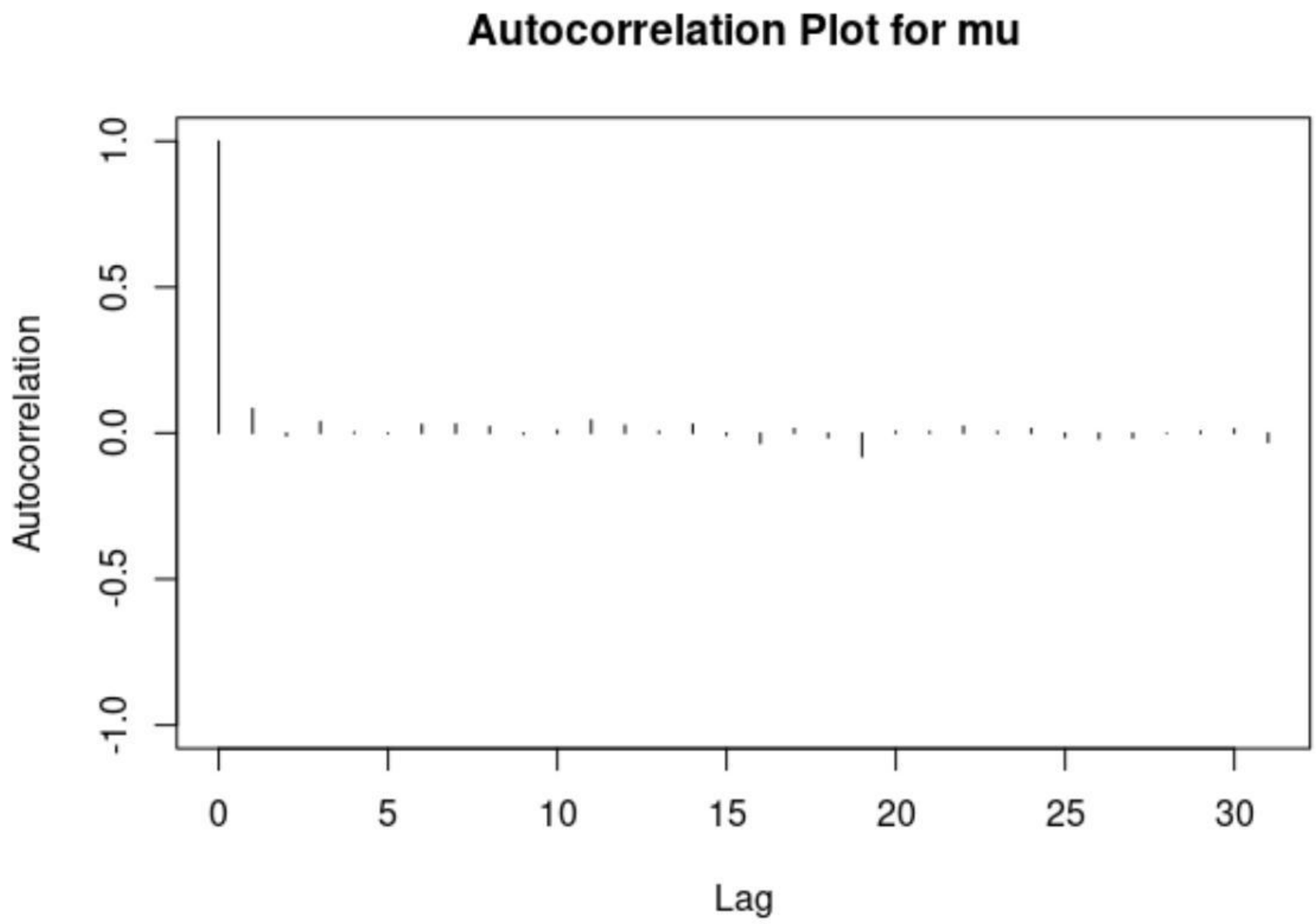

Figure 10.5: The autocorrelation of mu

```
  parameter = c('sigma2', 'mu', 'tau2'),
  mean = posterior_means,
  lower_95_CI = posterior_intervals[1,],
  upper_95_CI = posterior_intervals[2,]
)
```

with results:

```
> posterior_summary
      parameter      mean lower_95_CI upper_95_CI
sigma2    sigma2 14.491499   11.686186   17.851351
mu            mu  7.554651    5.981344    9.126796
tau2        tau2  5.501684    1.851430   14.044209
```

Now we can compare the posterior to prior using the R code:

```
# Define the sequences for the x-axis
seq_sigma2 <- seq(min(SMT[, "sigma2"]), max(SMT[, "sigma2"]), length.out =
↪  100)
```

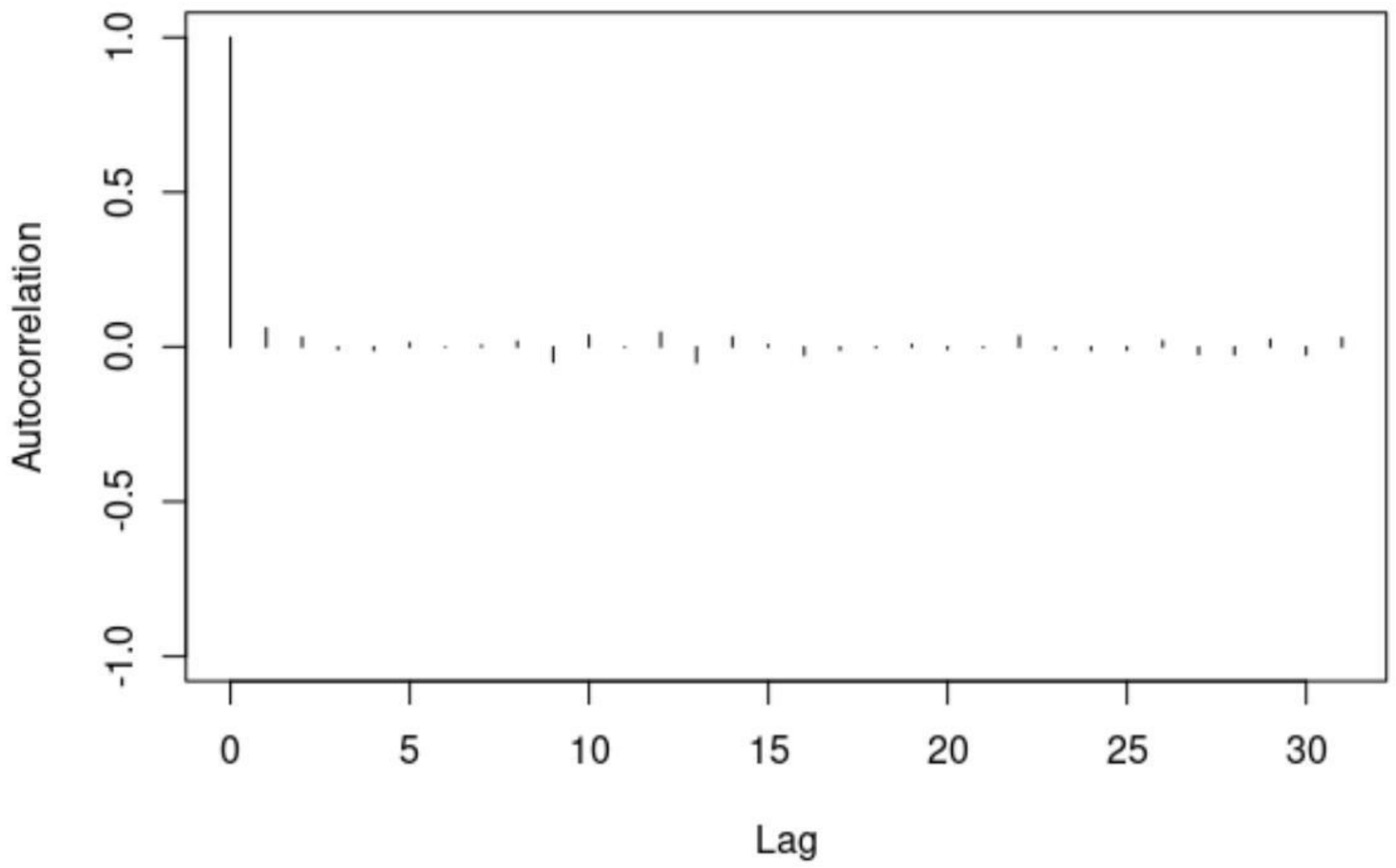

Figure 10.6: The autocorrelation of sigma

```r
seq_tau2 <- seq(min(SMT[, "tau2"]), max(SMT[, "tau2"]), length.out = 100)
seq_mu <- seq(min(SMT[, "mu"]), max(SMT[, "mu"]), length.out = 100)

# Calculate prior densities
sigma2_prior_density <- dinvgamma(seq_sigma2, shape = nu0 / 2, scale =
↪  sigma0_sq * nu0 / 2)
tau2_prior_density <- dinvgamma(seq_tau2, shape = eta0 / 2, scale =
↪  tau0_sq * eta0 / 2)
mu_prior_density <- dnorm(seq_mu, mean = mu0, sd = sqrt(gamma0_sq))

# Combine prior densities into a data frame
prior_df <- data.frame(
  value = c(seq_sigma2, seq_tau2, seq_mu),
  density = c(sigma2_prior_density, tau2_prior_density, mu_prior_density),
  variable = rep(c("sigma2", "tau2", "mu"), each = 100),
  Distribution = 'Prior'
)
```

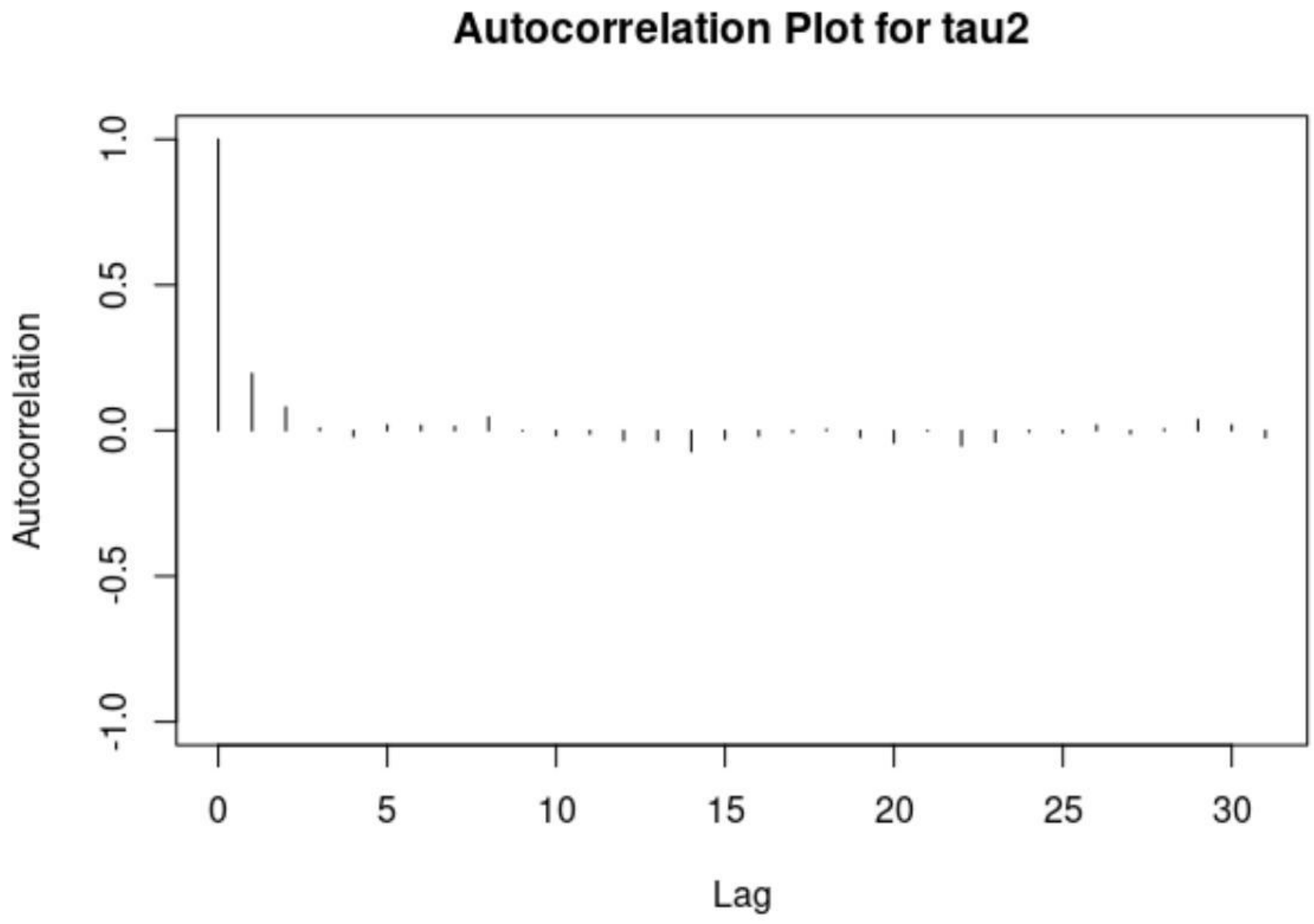

Figure 10.7: The autocorrelation of tao

```r
# Calculate posterior densities
posterior_densities_sigma2 <- density(SMT[, "sigma2"])
posterior_densities_tau2 <- density(SMT[, "tau2"])
posterior_densities_mu <- density(SMT[, "mu"])

# Create a combined data frame for posterior densities
posterior_df <- data.frame(
  value = c(posterior_densities_sigma2$x,
            posterior_densities_tau2$x,
            posterior_densities_mu$x),
  density = c(posterior_densities_sigma2$y,
              posterior_densities_tau2$y,
              posterior_densities_mu$y),
  variable = rep(c("sigma2", "tau2", "mu"),
                 each = length(posterior_densities_sigma2$y))
)

# Add a column to indicate that these are posterior densities
```

```r
posterior_df$Distribution <- 'Posterior'

# Combine prior and posterior densities
combined_df <- rbind(prior_df, posterior_df)

# Plotting prior and posterior densities
ggplot(combined_df, aes(x = value, y = density, color = Distribution)) +
  geom_line() +
  facet_wrap(~ variable, scales = 'free') +
  labs(color = "Distribution", x = "Value", y = "Density") +
  theme_minimal()
```

Resulting in the plot (10.8). From which we can observe that for μ the posterior distribution is tighter and more peaked than the prior, suggesting that the data provided substantial information about the true mean. The center of the posterior distribution is shifted slightly to the right of the prior mean, indicating that the average hours across schools are likely higher than the prior expectation of 7 hours. For σ^2 is also tighter than the prior distribution, indicating that the data helped to reduce uncertainty, the peak of the posterior distribution is also shifted to the right , suggesting that the variability within schools may be higher than initially expected. For τ^2 the posterior distribution shows a similar pattern of being more concentrated with the peak located to the right of the prior peak again, implying that the data provided evidence that there is more variation between the schools' average homework hours.

In Conclusion the data has had a significant impact on our beliefs about the parameters. The posterior distributions are more precise, reflecting increased certainty about the parameter values after observing the data. The reduction in the spread of the posterior distributions compared to the priors indicates a reduction in uncertainty about the parameters due to the data.The shifts in the peaks of the posterior distributions relative to the prior means indicate that the data has led to a re-evaluation of the parameter estimates suggesting that the actual homework time patterns in the sampled schools differ from the initial expectations. Finally he pronounced peak in the posterior distribution for τ^2 suggests strong evidence for between-school variation

For **(c)** we start by computing the prior density of R as:

```r
# Simulate values from the prior distributions
```

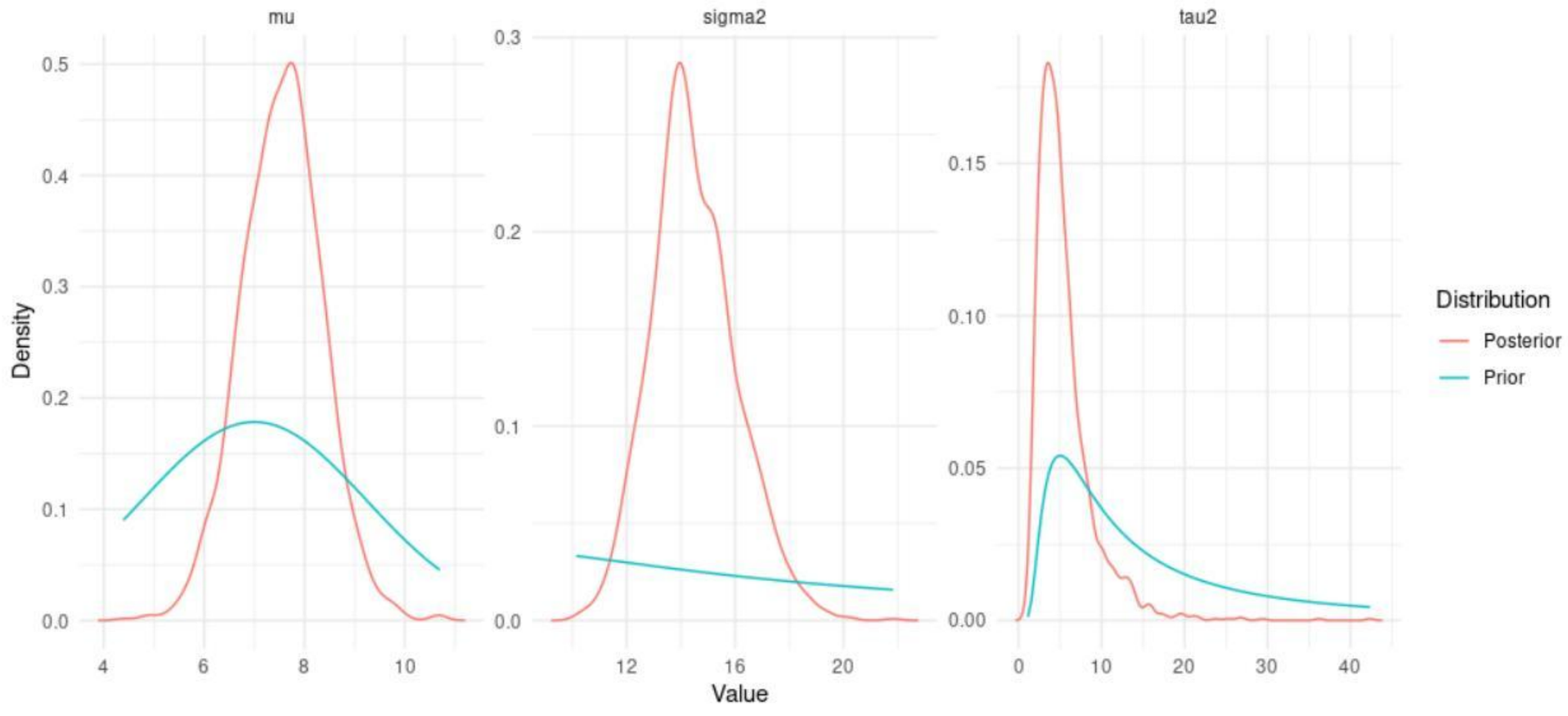

Figure 10.8: The Prior vs Posterior Plot

```
sigma2_prior_sim <- rinvgamma(10000, shape = nu0 / 2, scale = sigma0_sq *
↪  nu0 / 2)
tau2_prior_sim <- rinvgamma(10000, shape = eta0 / 2, scale = tau0_sq *
↪  eta0 / 2)
```

```
# Calculate R from the simulated prior values
R_prior_sim <- tau2_prior_sim / (sigma2_prior_sim + tau2_prior_sim)

# Calculate the density of R from the prior simulation
R_prior_density <- density(R_prior_sim)
```

and then the posterior:

```
# Calculate R from the posterior samples
R_posterior <- SMT[, "tau2"] / (SMT[, "sigma2"] + SMT[, "tau2"])

# Calculate the density of R from the posterior samples
R_posterior_density <- density(R_posterior)
```

and finally the plot (10.9):

```
# Create a data frame for plotting
R_density_df <- data.frame(
  value = c(R_prior_density$x, R_posterior_density$x),
  density = c(R_prior_density$y, R_posterior_density$y),
```

```
  Distribution = rep(c("Prior", "Posterior"), each =
  ↪  length(R_prior_density$x))
)

# Plot the densities
ggplot(R_density_df, aes(x = value, y = density, color = Distribution)) +
  geom_line() +
  labs(title = "Density of R", x = "Value of R", y = "Density", color =
  ↪  "Distribution") +
  theme_minimal()
```

R represents the proportion of the total variance attributed to between-school variability and based on the plot there is substantial evidence from the data to suggest that between-school variation is an important factor in the total variability based on the posterior distribution differing so much from the prior distribution

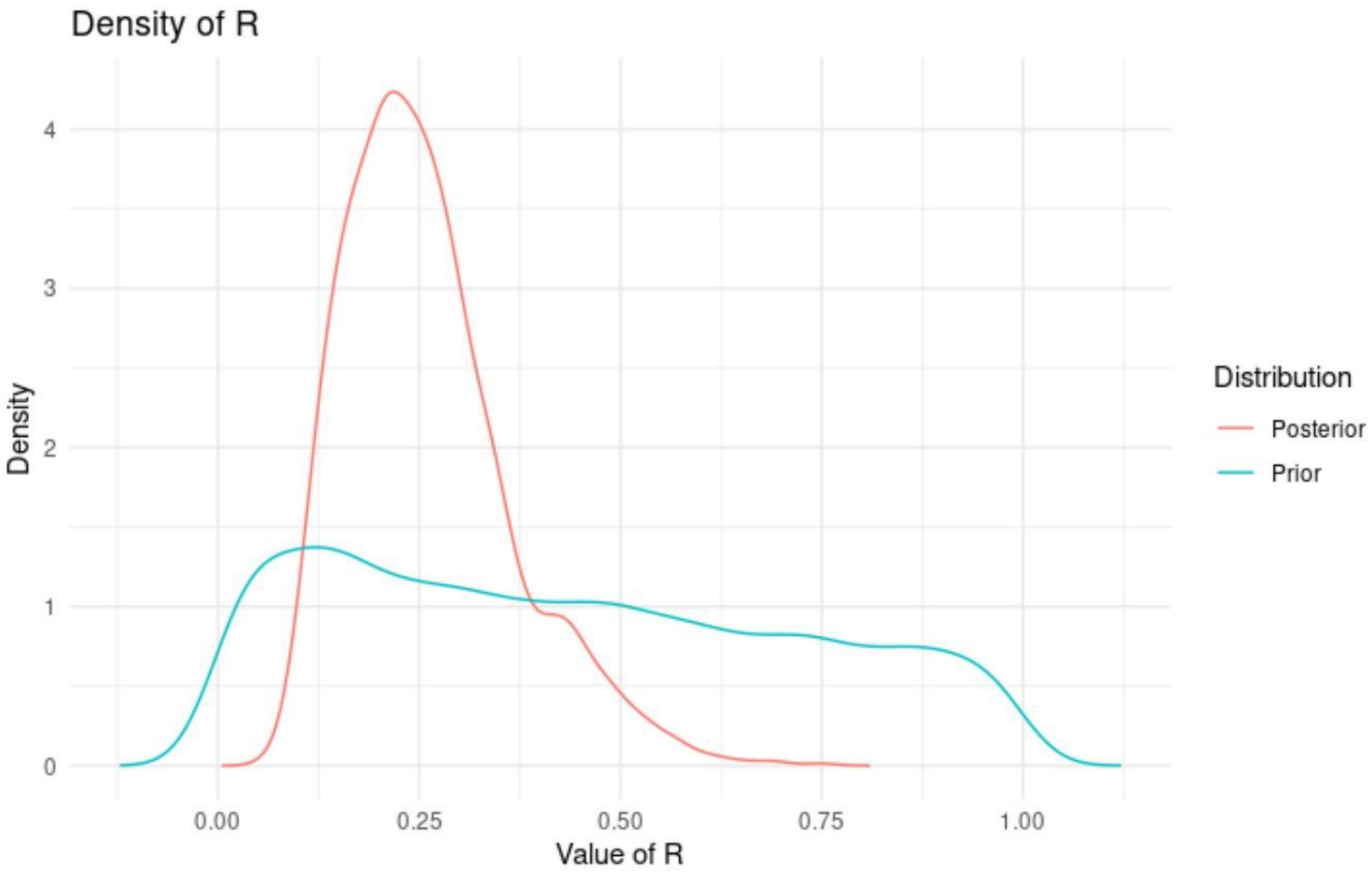

Figure 10.9: The R Densities

(d): The posterior probability that $\theta_7 < \theta_6$ is 0.552

```
> mean(THETA[, 7] < THETA[, 6])
[1] 0.552
```

and for the probability of θ_7 being the smallest:

```
theta7_smallest <- apply(THETA, 1, function(row) all(row[7] < row[-7]))
> mean(theta7_smallest)
[1] 0.3293333
```

The plot (10.10) is provided by the R code:

```
# Calculate the posterior means for theta for each school
posterior_means_theta <- colMeans(THETA)

# Create a data frame for plotting
comparison_df <- data.frame(sample_average = ybar, posterior_mean =
↪  posterior_means_theta, school = 1:m)

# Create the plot
ggplot(comparison_df, aes(x = sample_average, y = posterior_mean)) +
  geom_point() +  # Add the points
  geom_text(aes(label = school), vjust = -1) +  # Annotate points with
  ↪  school numbers
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color =
  ↪  "blue") +  # Add y=x reference line
  xlab("Sample Average") +
  ylab("Posterior Expectation of Theta") +
  ggtitle("Comparison of Sample Averages and Posterior Expectations") +
  theme_minimal()
```

The plot (10.10) suggests that the hierarchical model provides estimates that are informed by both the individual school data and the data from all schools combined resulting in a balance between the sample mean and the overall mean, offering potentially more robust estimates, especially for schools with less data. As all points are close to the line, we can see the shrinkage effect of the hierarchical model as schools with very high or very low sample averages have their posterior expectations adjusted toward the mean. Simply pout the hierarchical model has taken individual school data and adjusted the estimates toward a common group mean. The schools have sample averages close to the overall mean so the data provides strong evidence that the sample mean is close to the true mean and for the hours that means the sample mean of all observations will likely give an estimate of the overall homework hours that is less extreme than the sample means of individual schools. In conclusion,

the hierarchical model helps to stabilize the estimates of homework hours for each school by pooling information. It suggests that while there may be differences in how much homework is assigned across schools, once we account for the overall variability, individual school estimates are brought closer to a common mean. This is particularly beneficial for making inferences about schools with smaller sample sizes or more extreme sample averages.

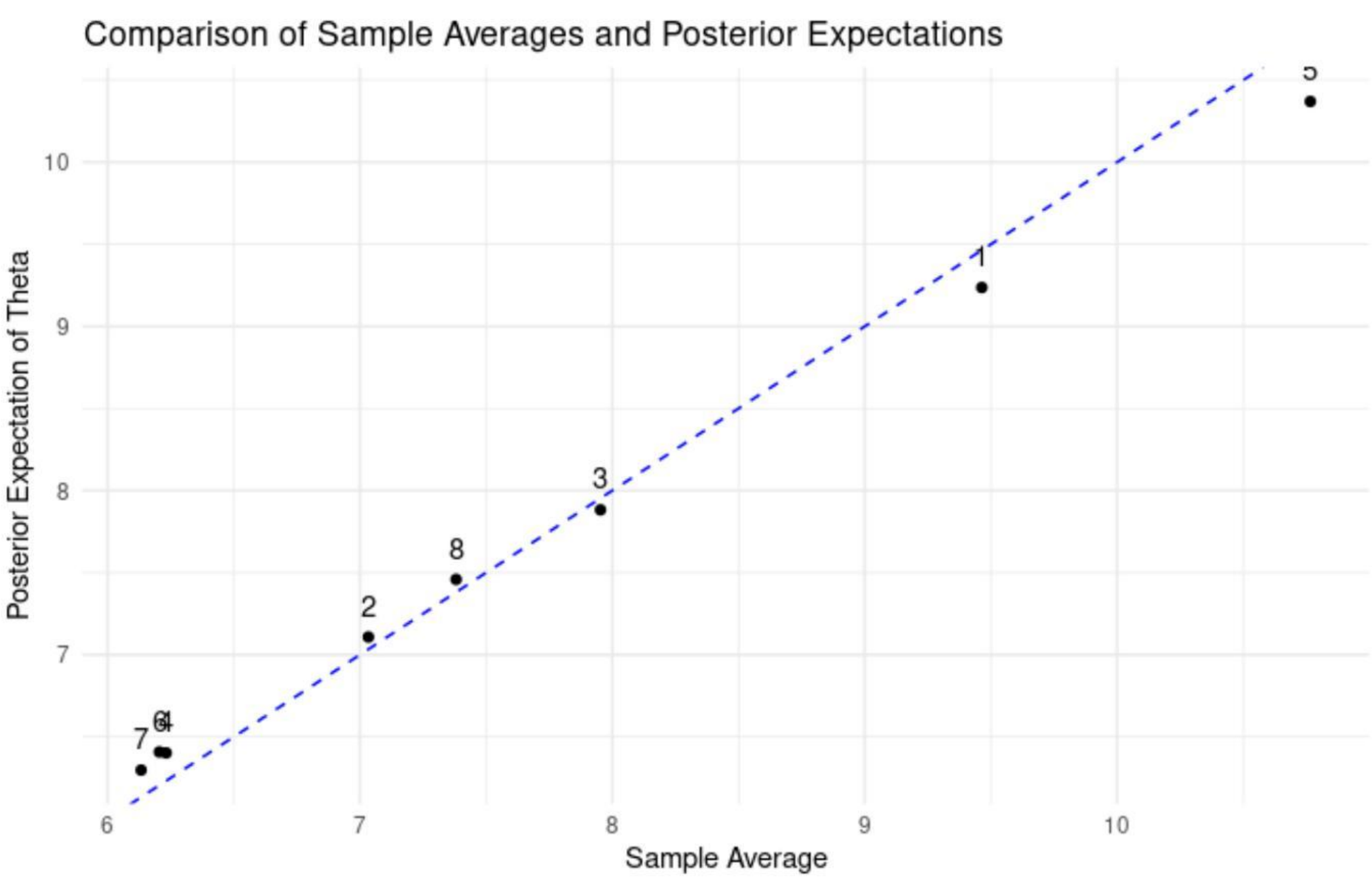

Figure 10.10: Sample Averages vs Posterior Expectation

To conclude this homework assignment we compute and compare the means:

```
# Compute the overall sample mean
overall_sample_mean <- mean(schools_data$hours)

# Extract the posterior mean of mu
posterior_mean_mu <- mean(SMT[, "mu"])
```

Provides us with the results:

```
> print(paste("Overall Sample Mean:", overall_sample_mean))
[1] "Overall Sample Mean: 7.69127777777778"
> print(paste("Posterior Mean of Mu:", posterior_mean_mu))
[1] "Posterior Mean of Mu: 7.55465118734256"
```

From these results we can conclude that the hierarchical model provides a slight overestimate thought preferable as it is a somewhat more conservative estimate of the average homework hours than the raw data alone thought is more reliable, especially when considering potential data anomalies or variations across different schools as it balances the information from all schools, giving a generalized estimate that takes into account both within-school and between-school variability.

Task 11

11.1 Extrapolation

[8]

The file `swim.dat` contains data on the amount of time, in seconds, it takes each of four high school swimmers to swim 50 yards. Each swimmer has six times, taken on a biweekly basis.

a) Data Analysis for Each Swimmer: Perform the following data analysis for each swimmer separately:

   i. Fit a linear regression model of swimming time as the response and week as the explanatory variable. To formulate your prior, use the information that competitive times for this age group generally range from 22 to 24 seconds.

   ii. For each swimmer j, obtain a posterior predictive distribution for Y_j^*, their time if they were to swim two weeks from the last recorded time.

b) Team Selection for Competition: The coach of the team has to decide which of the four swimmers will compete in a swimming meet in two weeks. Using your predictive distributions, compute $\Pr(Y_j^* = \max\{Y_1^*, \ldots, Y_4^*\}|Y)$ for each swimmer j, and based on this make a recommendation to the coach.

Solution:

Then was the frequentist:

```
# Data for the swimmers' times
swimmers <- data.frame(
  Week = rep(1:6, times = 4),
```

[8] Exercise 9.1 [DHo09]pg. 242

```
  Time = c(23.1, 23.2, 22.9, 22.9, 22.8, 22.7,
           23.2, 23.1, 23.4, 23.5, 23.5, 23.4,
           22.7, 22.6, 22.8, 22.8, 22.9, 22.8,
           23.7, 23.6, 23.7, 23.5, 23.5, 23.4),
  Swimmer = factor(rep(1:4, each = 6))
)

# For simplicity, we'll do a frequentist regression here.
results <- by(swimmers, swimmers$Swimmer, function(df) {
  lm(Time ~ Week, data = df)
})

# Display the summary for each swimmer's regression
lapply(results, summary)
```

```
> lapply(results, summary)
$`1`

Call:
lm(formula = Time ~ Week, data = df)

Residuals:
       1        2        3        4        5        6
-0.061905  0.129524 -0.079048  0.012381  0.003810 -0.004762

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 23.25333    0.07655 303.756 7.05e-10 ***
Week        -0.09143    0.01966  -4.651  0.00965 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.08223 on 4 degrees of freedom
Multiple R-squared:  0.844,     Adjusted R-squared:  0.8049
F-statistic: 21.63 on 1 and 4 DF,  p-value: 0.009653


$`2`
```

```
Call:
lm(formula = Time ~ Week, data = df)

Residuals:
       7        8        9       10       11       12
 0.01429 -0.15143  0.08286  0.11714  0.05143 -0.11429

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 23.12000    0.11347 203.748 3.48e-09 ***
Week         0.06571    0.02914   2.255   0.0871 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1219 on 4 degrees of freedom
Multiple R-squared:  0.5598,        Adjusted R-squared:  0.4497
F-statistic: 5.087 on 1 and 4 DF,  p-value: 0.08713
```

```
$`3`

Call:
lm(formula = Time ~ Week, data = df)

Residuals:
      13       14       15       16       17       18
 0.03333 -0.10667  0.05333  0.01333  0.07333 -0.06667

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 22.62667    0.07409 305.407  6.9e-10 ***
Week         0.04000    0.01902   2.103    0.103
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.07958 on 4 degrees of freedom
Multiple R-squared:  0.525,        Adjusted R-squared:  0.4062
F-statistic: 4.421 on 1 and 4 DF,  p-value: 0.1033
```

```
$`4`

Call:
lm(formula = Time ~ Week, data = df)

Residuals:
      19        20        21        22        23        24
-0.009524 -0.052381  0.104762 -0.038095  0.019048 -0.023810

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 23.76667    0.05923 401.275 2.31e-10 ***
Week        -0.05714    0.01521  -3.757   0.0198 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.06362 on 4 degrees of freedom
Multiple R-squared:  0.7792,    Adjusted R-squared:  0.724
F-statistic: 14.12 on 1 and 4 DF,  p-value: 0.01982
```

From the data, we see that swimmers 1 and 4 show statistically significant improvements in their swimming times over the weeks, though the greatest rate of improvement is that of swimmer 1. The rest swimmers 2 and 3 do not show statistically significant changes in their times. As 1 is predicted to decrease by 0.0914 seconds per week with an R-squared value of 0.844, 2 increase in time of 0.0657 seconds per week with R-squared = 0.5598, 3 ncreases in swim time of 0.04 seconds per week with a moderate R-squared value of 0.525 and the 4rth indicates a decrease in time of 0.0571 seconds per week with a strong model fit R-squared = 0.7792.

Let there be Bayes:

Using the below code we will fit the linear regression model of swimming with time as the response and week as the explanatory variable. The prior expectation is that competitive times generally range from 22 to 24 seconds in the prior covariance matrix. So for the intercept, a prior mean of 23 seconds was chosen, and for the slope a prior mean of 0, indicating non-informality. The variance for the intercept was chosen such that 95% of the belief falls within [22, 24], which was operationalized

as $\Sigma_{0(1,1)} = 1/4$ and a smaller variance of 0.1 for the slope, reflecting mild training effects. The Bayesian linear regression was performed using Gibbs sampling, providing a series of posterior distributions, for each swimmer's performance. The model captures the trajectory of swimming times over a sries of weeks, considering both the inherent variability in performance and the prior belief witch provides an insight into how their performance might evolve over time.

```
# Set the number of simulations
S <- 5000

# Time points corresponding to the biweekly measurements
# Assuming that the measurements start at week 1 and are taken biweekly
X <- cbind(1, seq(1, 11, by = 2))
n <- nrow(X)
p <- ncol(X)

# Prior specifications
# Prior mean for the intercept is set to the midpoint of the competitive
↪  range [22, 24]
# Prior mean for the slope is set to 0, indicating no expected change by
↪  default
beta0 <- c(23, 0)

# Prior covariance matrix
# We allow some variance around the competitive times for the intercept,
# and we are less certain about the effect of training, hence a larger
↪  variance
Sigma0 <- diag(c((24 - 22) / (2 * 1.96)^2, 0.1), p, p)

# Prior degrees of freedom and scale for the inverse-gamma distribution of
↪  sigma^2
nu0 <- 1
s20 <- (24 - 22) / (2 * 1.96)^2

# Seed for reproducibility
set.seed(1)

# Gibbs sampling function for Bayesian linear regression
```

```r
run_gibbs <- function(y, BETA, SIGMA, S, X, beta0, Sigma0, nu0, s20) {
  # Starting values
  beta <- beta0
  s2 <- s20

  for (s in 1:S) {
    # Sample from the posterior distribution of beta
    V <- solve(solve(Sigma0) + t(X) %*% X / s2)
    m <- V %*% (solve(Sigma0) %*% beta0 + t(X) %*% y / s2)
    beta <- mvrnorm(1, m, V)

    # Sample from the posterior distribution of sigma^2
    ssr <- sum((y - X %*% beta)^2)
    s2 <- 1 / rgamma(1, (nu0 + n) / 2, (nu0 * s20 + ssr) / 2)

    # Store samples
    BETA[s, ] <- beta
    SIGMA[s] <- s2
  }
  list(BETA = BETA, SIGMA = SIGMA)
}

# Posterior prediction for each swimmer
swim_pred <- lapply(1:4, function(j) {
  y <- swim[, j]

  # Initialize matrices to store samples
  BETA <- matrix(nrow = S, ncol = p)
  SIGMA <- numeric(S)

  # Run Gibbs sampling
  samples <- run_gibbs(y, BETA, SIGMA, S, X, beta0, Sigma0, nu0, s20)

  # Predict future swimming time (two weeks after the last recorded time,
  ↪  week 13)
  x_new <- c(1, 13)  # Predictor for future time point
  y_pred <- rnorm(S, samples$BETA %*% x_new, sqrt(samples$SIGMA))
```

```
  y_pred
})

# Combine predictions into a matrix for easier analysis
swim_pred_matrix <- do.call(cbind, swim_pred)

# Identify the fastest predicted time for each simulation
fastest_swimmer <- apply(swim_pred_matrix, 1, which.min)

# Calculate the probability of being the fastest for each swimmer
prob_fastest <- table(fastest_swimmer) / S

# Print the probabilities
print(prob_fastest)

# Make recommendation based on the highest probability
recommended_swimmer <- which.max(prob_fastest)
cat("Recommend swimmer", recommended_swimmer, "for the race.\n")
```

resulting in:

```
> # Print the probabilities
> print(prob_fastest)
fastest_swimmer
     1      2      3      4
0.2910 0.4448 0.1192 0.1450
>
> # Make recommendation based on the highest probability
> recommended_swimmer <- which.max(prob_fastest)
> cat("Recommend swimmer", recommended_swimmer, "for the race.\n")
Recommend swimmer 2 for the race.
```

The Bayesian analysis, suggests that Swimmer 2 is the most promising candidate for achieving the best time in the upcoming race. This recommendation takes into account not only the past performance data but also the uncertainties and variations of performance. So while Swimmer 2 has the highest probability of being the fastest, there is still a significant chance that one of the other swimmers could outperform him.

The discrepancy between the frequentist regression results and the Bayesian predictions could arise due to the fact that the frequentist approach does not incorporate prior beliefs and it focuses on statistical significance, while the Bayesian approach provides predictive probabilities and also the fact that the R-squared values in the frequentist models indicate the fit to the observed data, but they don't directly translate to predictive performance, especially if there's high variability in the data.

ii The code provided a distribution of plausible values for the regression coefficients and the variance, then used them for performance prediction of two weeks after the last recorded time For each set of posterior samples, we predicted the swimmer's time for the week after. These predictions are drawn from a normal distribution with the mean calculated as the product of the sampled $\boldsymbol{\beta}$ and x_{pred}, while the standard deviation as, the square root of the sampled σ^2. This process results in a posterior predictive distribution for each swimmer's future performance for two weeks after the last recorded time and the results accounted for variability and uncertainty..
What began as an additional plot (11.11) for visual aid:

```
library(ggplot2)
library(tidyr)

# Convert the list of posterior predictions to a data frame for plotting
posterior_predictions_df <- as.data.frame(swim_pred)

# Reshape the data for ggplot
posterior_predictions_long <- posterior_predictions_df %>%
  pivot_longer(cols = everything(), names_to = "Swimmer", values_to =
  ↪  "PredictedTime")

# Plotting the posterior predictive distributions
ggplot(posterior_predictions_long, aes(x = PredictedTime, fill = Swimmer))
↪  +
  geom_density(alpha = 0.5) +
  labs(x = "Predicted Time (seconds)", y = "Density", title = "Posterior
  ↪  Predictive Distributions for Swimmers' Times") +
  theme_minimal() +
  theme(legend.title = element_blank()) +
  scale_fill_brewer(palette = "Set1") +
```

```
facet_wrap(~ Swimmer, ncol = 2)
```

As we can see from the plot (11.11) each curve represents a swimmer's posterior predictive time distribution with the height indicating the probability density of that specific time and the width of the curve reflecting the spread or variance of predicted times. For example, swimmer 4's curve has a pronounced peak compared to the others, suggesting a higher likelihood for a specific time. On the other hand, a wider curve, as seen with Swimmers 1, 2, and 3, shows that the predicted times are more spread out, indicating more uncertainty about their exact timing.

More specifically: Swimmer 1 has a peak that is quite high and somewhat narrow, but not as narrow as Swimmer 4's indicating that the swimmer has a strong likelihood of achieving a time around the peak, with some variability. While swimmer 2 has a wider distribution, suggesting that while their average time might be similar to Swimmer 1's, there is more uncertainty in their performance. Then swimmer 3 shows a distribution with a slightly lower peak and more spread, indicating even more variability in their predicted times than Swimmers 1 and 2. Lastly swimmer 4 has the narrowest distribution, suggesting the highest confidence in a consistent performance time, which is desirable for predictability in competitive swimming. So In summary, the plot suggests that Swimmer 4 is expected to perform with high consistency, Swimmer 1 is also consistent but with a bit more variability, and Swimmers 2 and 3 have wider ranges of possible outcomes, indicating less predictability in their performance times. This provides another discrepancy that I have not yet solved, though it might be due to the plot itself or it's representation I have reason to believe the answer lies elsewhere. I believe that is due to the way we calculated the probabilities of the faster swimmer...

For **(b)** we know that: The code calculated the probabilities of each swimmer being the fastest in a future race by evaluating which swimmer had the minimum predicted time in each of the simulation iterations.
The results were:

Swimmer 1: Approximately 29.10% Swimmer 2: Approximately 44.48% Swimmer 3: Approximately 11.92% Swimmer 4: Approximately 14.50%

In conclusion: The Bayesian analysis suggests that Swimmer 2 is the most promising candidate for achieving the best outcome in the future race while we note that Swimmer 2 is the most likely to be the fastest according to the model, there is still a significant probability that another swimmer could outperform Swimmer 2,

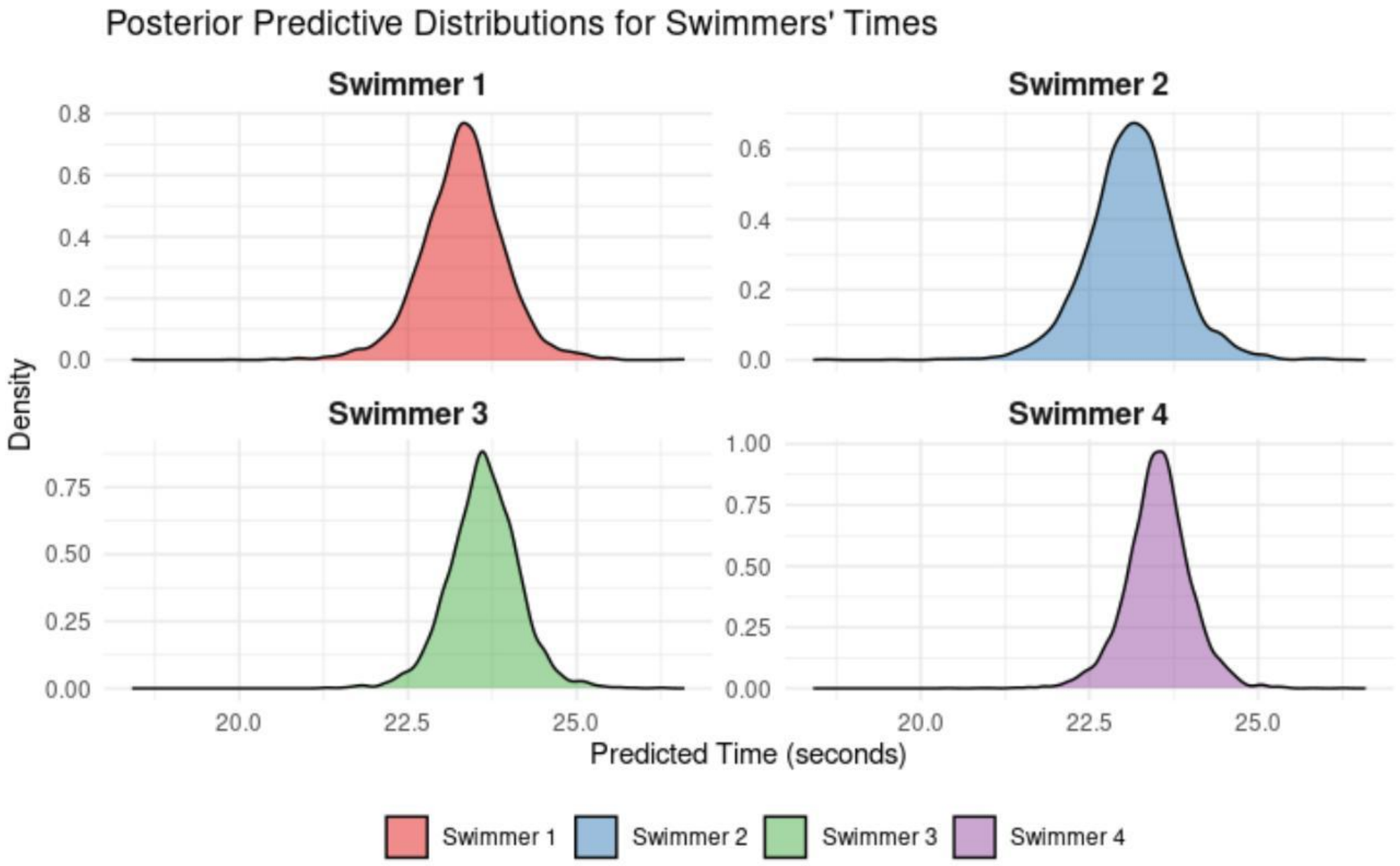

Figure 11.11: The posterior Predictive distribution

as the frequentist approach indicated the 4rth as did the plots (11.11) a fact that should be considered in the final decision-making process.

Task 12

12.1 Crime Data Analysis Exercise

9

Crime: The file `crime.dat` contains crime rates and data on 15 explanatory variables for 47 U.S. states, in which both the crime rates and the explanatory variables have been centered and scaled to have variance 1. A description of the variables can be obtained by typing `library (MASS); ?UScrime` in R.

a) Regression Model with g-Prior Fit a regression model $y = X\beta + \varepsilon$ using the g-prior with $g = n$, $\nu_0 = 2$ and $\sigma_0^2 = 1$. Obtain marginal posterior means and 95% confidence intervals for β, and compare to the least squares estimates. Describe the relationships between crime and the explanatory variables. Which variables seem strongly predictive of crime rates?

b) Predictive Analysis with Training and Test Sets

i. Using only the training set, obtain least squares regression coefficients $\hat{\beta}_{\text{ols}}$. Obtain predicted values for the test data by computing $\hat{y}_{\text{ols}} = X_{\text{te}}\hat{\beta}_{\text{ols}}$. Plot $\hat{y}_{\text{ols}}$ versus y_{te} and compute the prediction error $\frac{1}{n_{\text{te}}}\sum(y_{i,\text{te}} - \hat{y}_{i,\text{ols}})^2$.

ii. Now obtain the posterior mean $\hat{\beta}_{\text{Bayes}} = \mathbb{E}[\beta|y_{\text{tr}}]$ using the g-prior described above and the training data only. Obtain predictions for the test set $\hat{y}_{\text{Bayes}} = X_{\text{te}}\hat{\beta}_{\text{Bayes}}$. Plot versus the test data, compute the prediction error, and compare to the OLS prediction error. Explain the results.

c) Repeated Random Partitioning Repeat the procedures in b) many times with different randomly generated test and training sets. Compute the average prediction error for both the OLS and Bayesian methods.

Solution :

[9] Exercise 9.3 [DHo09]pg. 243

Part a)

The objective is to fit the regression model using g=prior with given parameters. So we should:

- Obtain marginal posterior means for β
- Calculate 95% confidence intervals for β
- Compare them to the OLS estimates
- Analyze the relationship between crime rates and explanatory variables
- Identify which variables are strongly predictive of crime rates

The g-prior: The g prior distribution of β is s.t. :

$$\beta|\sigma^2 \sim \mathbf{N}(0, g\sigma^2 \left(X^T X\right)^{-1})$$

, where g is a scaling factor equal to the number of observations a.k.a n, in our case.

```
library(MASS)

# After the dataset is downloaded
crime <- read.table('crime.dat', header = TRUE)

# The response and predictor variables
y <- crime$y
X <- as.matrix(crime[, -which(names(crime) == "y")])

# Parameters for g-prior
n <- nrow(X)
g <- n  # g is set to the number of observations
nu0 <- 2
s20 <- 1

# g-prior for beta
Hg <- (g / (g + 1)) * X %*% solve(t(X) %*% X) %*% t(X)
```

The error variance:

$$\sigma^2 \sim \text{Inv-Gamma}\left(\frac{\nu_0}{2}, \frac{\nu_0\sigma_0^2}{2}\right)$$

To compute the marginal Distribution we need to integrate over the possible values of σ^2 using MCMC

```
# The Sum of Squared Residuals under g-prior
SSRg <- t(y) %*% (diag(1, nrow = n) - Hg) %*% y

# Sampling from posterior distribution of sigma^2
s2 <- 1 / rgamma(n, (nu0 + n) / 2, (nu0 * s20 + SSRg) / 2)

# Posterior distribution of beta
Vb <- g * solve(t(X) %*% X) / (g + 1)
Eb <- Vb %*% t(X) %*% y

# Generating samples for beta
S <- 1000  # Number of samples

# Ensure E dimensions and scale
E <- matrix(rnorm(S * ncol(X), 0, 1), nrow = S, ncol = ncol(X))
E <- E * sqrt(s2)

# Generate samples for beta
# Each row of E is scaled by the Cholesky decomposition of Vb and then Eb
↪  is added
beta_samples <- t(t(E %*% chol(Vb)) + rep(Eb, each = S))

# The marginal posterior means and confidence intervals
beta_post_means <- apply(beta_samples, 2, mean)
beta_conf_intervals <- apply(beta_samples, 2, function(x) quantile(x,
↪  probs = c(0.025, 0.975)))
```

resulting in

```
>beta_post_means
        M         So         Ed        Po1        Po2         LF        M.F
        ↪  Pop         NW
0.1524082 0.1529752 0.1552496 0.1619236 0.1461990 0.1472494 0.1623609
↪  0.1551132 0.1488202
       U1         U2        GDP       Ineq       Prob       Time
0.1477012 0.1523438 0.1479664 0.1511493 0.1510928 0.1547723
```

```
#and
```

```
> beta_conf_intervals
                M          So          Ed         Po1         Po2          LF
                ↪  M.F         Pop
2.5%  -0.7926492 -0.7957339 -0.8398364 -1.439917 -1.528027 -0.8184085
↪  -0.8011952 -0.8106668
97.5%  1.4721950  1.4844710  1.4434498  1.896584  1.886493  1.4840059
↪  1.5076551  1.4742319
               NW          U1          U2         GDP        Ineq        Prob
               ↪  Time
2.5%  -0.8432334 -0.8471156 -0.8306919 -0.8154426 -0.7966953 -0.7853134
↪  -0.8054932
97.5%  1.4909691  1.4588526  1.4727655  1.4781137  1.5002297  1.4800273
↪  1.4809153
```

The comparison with OLS

```
# Least squares estimates
beta_ols <- solve(t(X) %*% X) %*% t(X) %*% y

# Comparison
comparison <- data.frame(OLS = beta_ols, Bayesian = beta_post_means)
```

, provides as with the matrix

```
>comparison
                OLS  Bayesian
M     0.2865177028 0.1524082
So   -0.0001179958 0.1529752
Ed    0.5445161778 0.1552496
Po1   1.4716146465 0.1619236
Po2  -0.7817757455 0.1461990
LF   -0.0659672893 0.1472494
M.F   0.1313002714 0.1623609
Pop  -0.0702910179 0.1551132
NW    0.1090590127 0.1488202
U1   -0.2705407273 0.1477012
```

```
U2     0.3687335028 0.1523438
GDP    0.2380580097 0.1479664
Ineq   0.7262918200 0.1511493
Prob  -0.2852262729 0.1510928
Time  -0.0615771841 0.1547723
```

For visual aid I will use the plot (12.12) provided by the code:

```r
library(ggplot2)

# Data for the comparison
comparison_data <- data.frame(
  Variable = rownames(comparison),
  OLS = comparison[1],
  Bayesian = comparison[2]
)

# Convert data to long format (for ggplot)
comparison_long <- tidyr::pivot_longer(comparison_data, cols = c("OLS",
↪  "Bayesian"), names_to = "Method", values_to = "Estimate")

# Plot
ggplot(comparison_long, aes(x = Variable, y = Estimate, color = Method)) +
  geom_line(aes(group = Method), size = 1) +
  geom_point(size = 2) +
  theme_minimal() +
  labs(title = "Comparison of OLS and Bayesian Coefficient Estimates",
       x = "Variable", y = "Coefficient Value") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

As we can see for most variables, the Bayesian approach seems to shrink the coefficients towards zero, as the estimates it provides are more conservative (especially) compared to the OLS additionally the OLS estimates show more variability with some coefficients having relatively large positive or negative values (see Po1 and Po2). To additionally aid our understanding we present their summaries:

```
> summary(comparison)
      OLS                Bayesian
 Min.   :-0.78178   Min.   :0.1462
 1st Qu.:-0.06813   1st Qu.:0.1484
```

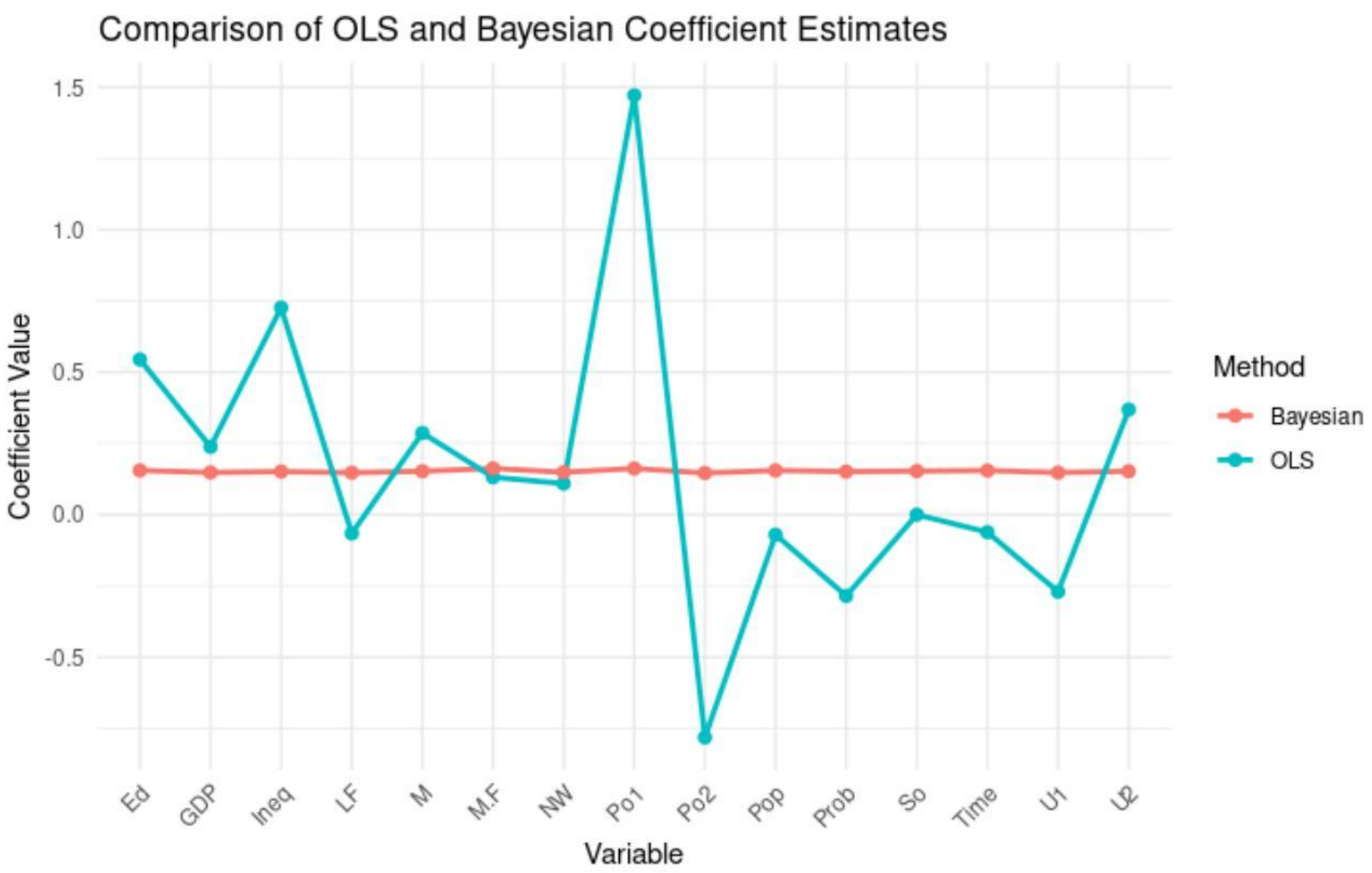

Figure 12.12: OLS vs Bayes

```
Median : 0.10906   Median :0.1523
Mean   : 0.15604   Mean   :0.1525
3rd Qu.: 0.32763   3rd Qu.:0.1549
Max.   : 1.47161   Max.   :0.1624
```

In conclusion: The Bayesian approach appears to regularize the coefficient estimates. An effect that could be helpful if there are issues of multicollinearity. The consistency across Bayesian estimates suggests more reliable and stable relationships between the explanatory variables and the crime rates, though the Bayesian method accounts for prior information thus the results in some cases are highly dependent on the prior information. In our case even though the predictions might appear to be far from reality we can see that the CI of 95% captures the OLS measurements.
Final note : To address the question of which variables strongly affect the prediction of crime rates are the Po1 and Po2. In OLS it's obvious from the first plot and in Bayes we can see that those have the bigger CI's as we can see for the plot (12.13).

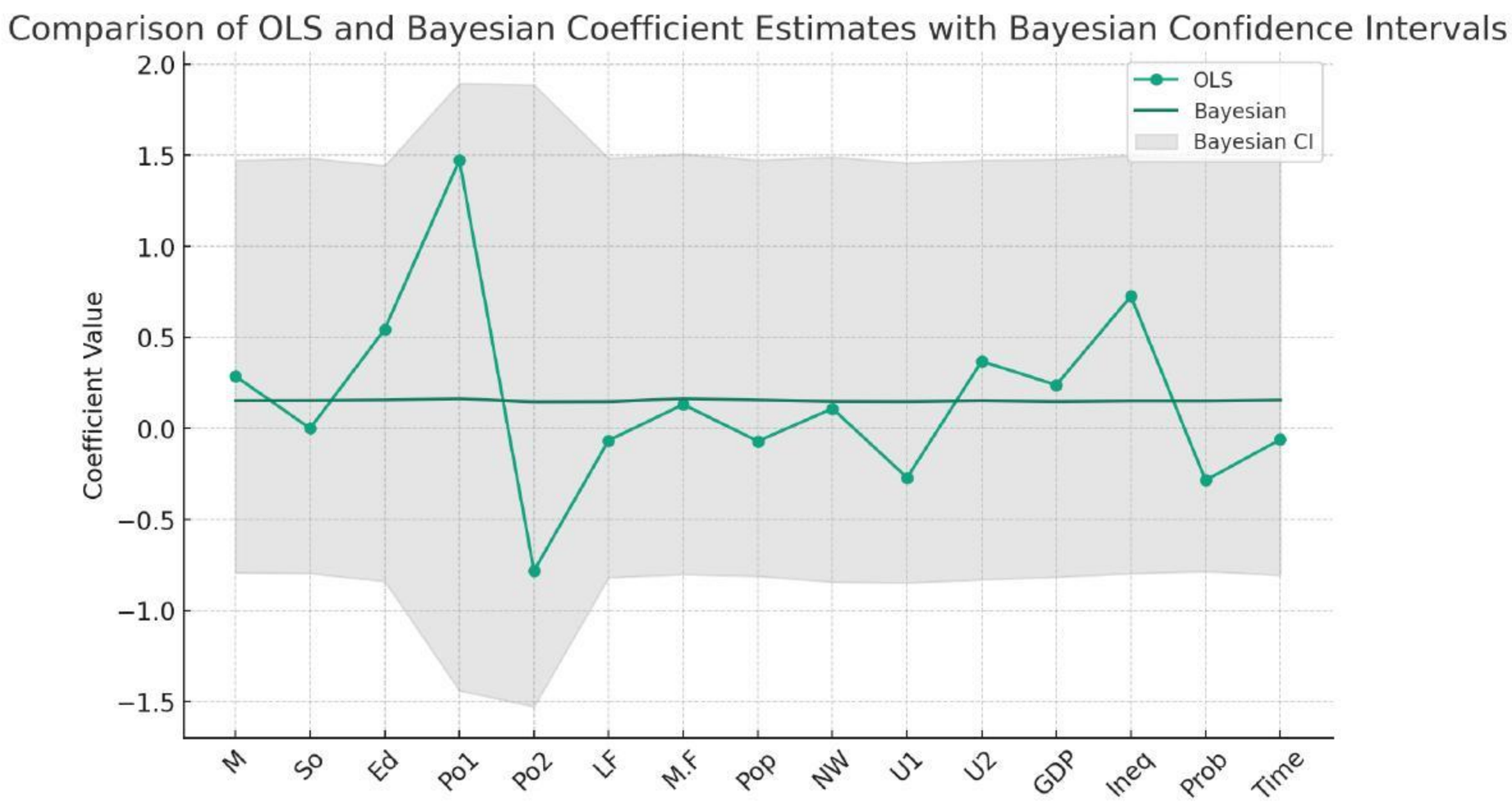

Figure 12.13: OLS vs Bayes with CI's

12.1.1 Part b) :

```
# Splitting the data into training and test sets
set.seed(123)  # For reproducibility
train_indices <- sample(1:nrow(crime), size = nrow(crime)/2)
train_data <- crime[train_indices, ]
test_data <- crime[-train_indices, ]

# Part b - i: OLS Regression
ols_fit <- lm(y ~ ., data = train_data)
y_pred_ols <- predict(ols_fit, newdata = test_data)
ols_pred_error <- mean((test_data$y - y_pred_ols)^2)


#Data to plot
plot_data <- data.frame(
  Actual = test_data$y,
  Predicted_OLS = y_pred_ols
)

# The plot
ggplot(plot_data, aes(x = Actual, y = Predicted_OLS)) +
  geom_point() +
```

```
geom_abline(slope = 1, intercept = 0, linetype = "dashed", color =
↪  "blue") +
labs(title = "OLS Predicted vs Actual Values",
     x = "Actual Values",
     y = "OLS Predicted Values") +
theme_minimal()
```

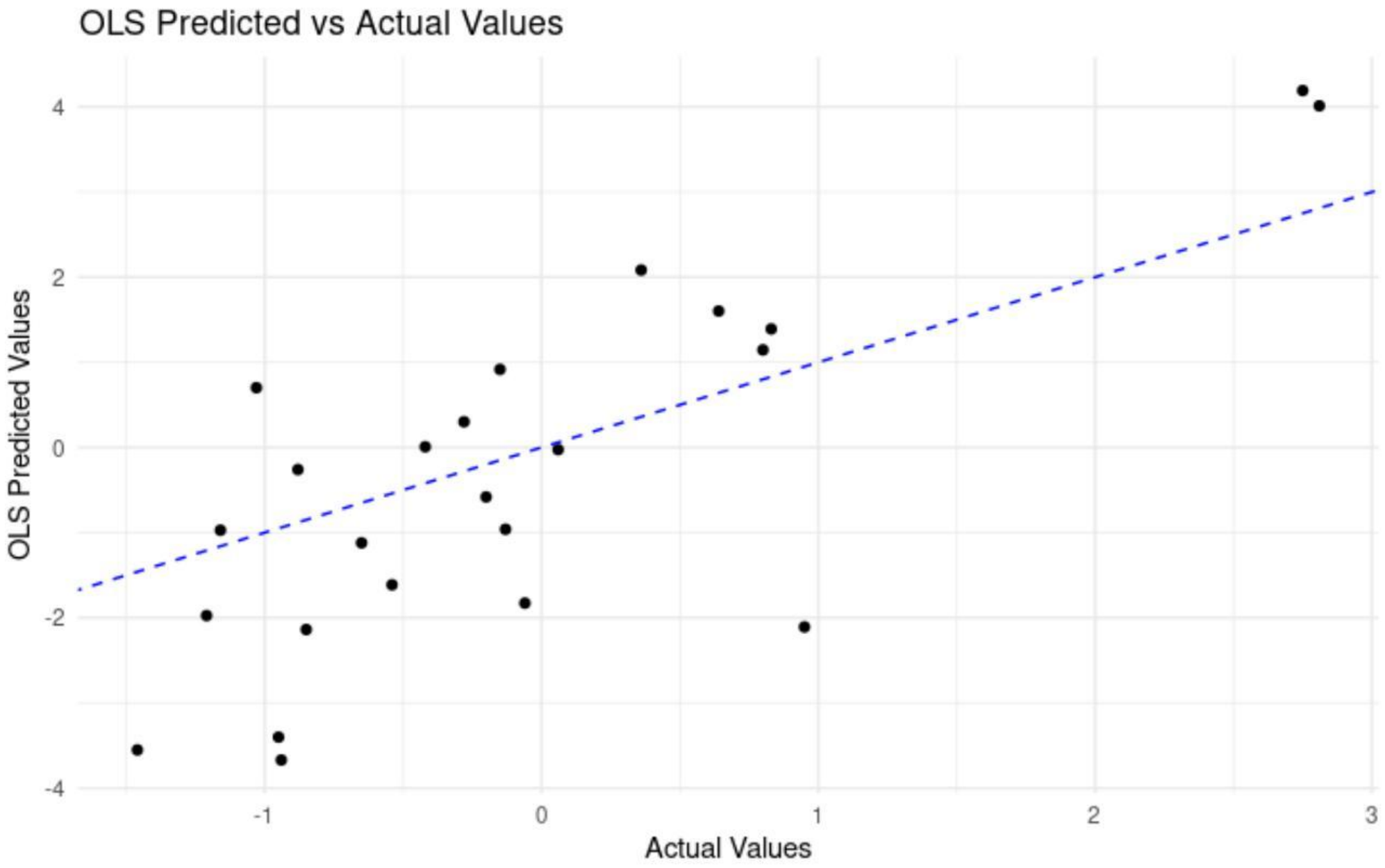

Figure 12.14: OLS test results plot

Now moving on to Bayes:

```
# Predictors and response variables for training and test sets
Xtr <- as.matrix(train_data[, -1])  # All columns except the first
↪  (response variable)
ytr <- train_data[, 1]              # First column is the response
↪  variable
Xte <- as.matrix(test_data[, -1])   # All columns except the first
↪  (response variable)
yte <- test_data[, 1]               # First column is the response
↪  variable

#the Bayesian posterior mean using training data
```

```
Hg <- (g / (g + 1)) * Xtr %*% solve(t(Xtr) %*% Xtr) %*% t(Xtr)
SSRg <- t(ytr) %*% (diag(1, nrow = length(ytr)) - Hg) %*% ytr

# Bayesian posterior samples
s2_samples <- 1 / rgamma(S, (nu0 + nrow(Xtr)) / 2, (nu0 * s20 + SSRg) / 2)
Vb <- g * solve(t(Xtr) %*% Xtr) / (g + 1)
Eb <- Vb %*% t(Xtr) %*% ytr

beta_bayes_samples <- matrix(rnorm(S * ncol(Xtr), mean = rep(Eb, S), sd =
↪  rep(sqrt(diag(Vb)), S)), nrow = S, byrow = TRUE)
beta_bayes <- colMeans(beta_bayes_samples)

# The values for the test data
y_pred_bayes <- as.matrix(Xte) %*% beta_bayes

# Data for plotting
bayes_plot_data <- data.frame(Actual = yte, Predicted_Bayes =
↪  y_pred_bayes)

# The plot
ggplot(bayes_plot_data, aes(x = Actual, y = Predicted_Bayes)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color =
  ↪  "blue") +
  labs(title = "Bayesian Predicted vs Actual Values",
       x = "Actual Values",
       y = "Bayesian Predicted Values") +
  theme_minimal()
```

To compute the prediction error, and compare to the OLS:

```
># The Bayesian prediction error
>bayes_pred_error <- mean((yte - y_pred_bayes)^2)
>
># Compared to OLS prediction error
>list(OLS_Prediction_Error = ols_pred_error, Bayesian_Prediction_Error =
↪  bayes_pred_error)
$OLS_Prediction_Error
[1] 1.988359
```

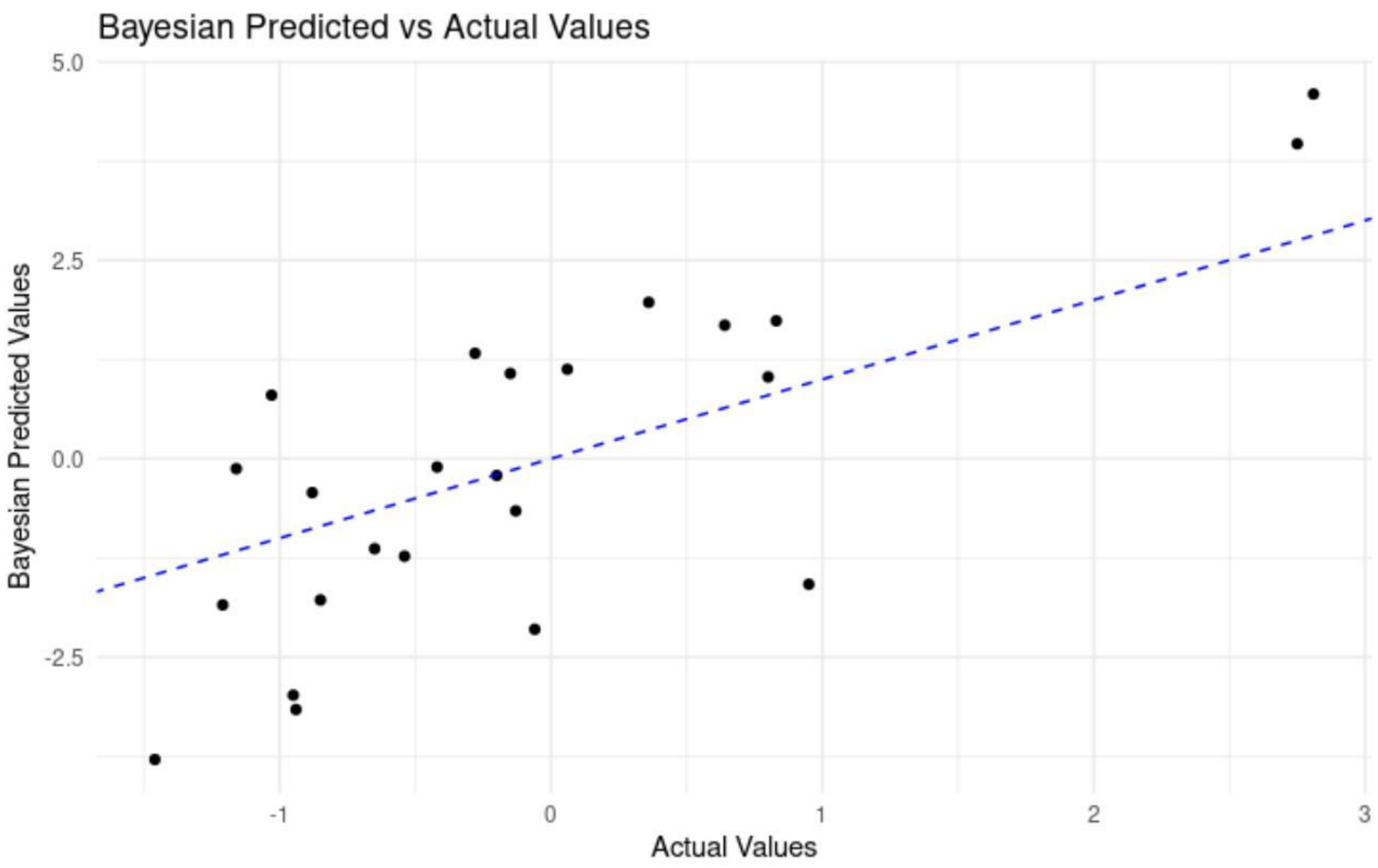

Figure 12.15: Bayes test results plot

```
$Bayesian_Prediction_Error
[1] 1.942511
```

These results suggest that while both models provide reasonably good predictions, the Bayesian model has a slight edge in this particular case. However, the choice between OLS and Bayesian models should also consider other factors such as model interpretability, computational complexity, and the nature of the data.

Part c)

For part c of the exercise, we will repeat the procedures from part b multiple times with different randomly generated test and training sets to compute the average prediction error for both and assess the generalizability capabilities and robustness of the models.

```
# Number of repetitions
num_repetitions <- 100

# Initial vectors to store prediction errors
```

```
ols_errors <- numeric(num_repetitions)
bayes_errors <- numeric(num_repetitions)

# Repetitions of the process
for (i in 1:num_repetitions) {
  # Randomly split the data
  train_indices <- sample(1:nrow(crime), size = nrow(crime)/2)
  train_data <- crime[train_indices, ]
  test_data <- crime[-train_indices, ]

  # Extracting predictors and response for training and test sets
  Xtr <- as.matrix(train_data[, -1])
  ytr <- train_data[, 1]
  Xte <- as.matrix(test_data[, -1])
  yte <- test_data[, 1]

  # OLS regression
  ols_fit <- lm(y ~ ., data = train_data)
  y_pred_ols <- predict(ols_fit, newdata = test_data)
  ols_errors[i] <- mean((yte - y_pred_ols)^2)

  # Bayesian regression
  y_pred_bayes <- Xte %*% beta_bayes  # Predict using Bayesian model
  bayes_errors[i] <- mean((yte - y_pred_bayes)^2)
}

# Average prediction errors
average_ols_error <- mean(ols_errors)
average_bayes_error <- mean(bayes_errors)
```

with results:

```
> # The average prediction errors
> list(Average_OLS_Error = average_ols_error, Average_Bayesian_Error =
↪  average_bayes_error)
$Average_OLS_Error
[1] 1.254517

$Average_Bayesian_Error
[1] 1.015142
```

Again the results suggest that the Bayesian model is a more reliable predictor for this specific dataset under the g prior with the given variables. As the difference in the average prediction errors between the two models indicates that the Bayesian approach may be more robust and adaptable to different data splits and could be better generalized, while it also incorporates prior information and handles uncertainty more effectively.

Bibliography

[DHo09] Peter D.Hoff. *A First Course in Bayesian Statistical Methods*. 233 Spring Street, New York, NY10013, USA: Springer Science+Business Media, LLC, 2009. ISBN: 9780387922997.

[Cow13] Mary Kathryn Cowles. *Applied Bayesian Statistics With R and OpenBUGS Examples*. 233 Spring Street, New York, NY10013, USA: Springer Science+Business Media, LLC, 2013. ISBN: 9781461456964.