SDS_459_HW2

2025-04-05

Aidan Kardan Bayesian Statistics HW 2

Question 2,3,4, and 5

```r
# 1) Simulate data
set.seed(123)
n <- 30
x <- rgamma(n, shape = 2.5, rate = 1.5)
```

```r
# 2) Log-posterior = log-likelihood + log-prior
log_post <- function(alpha, beta) {
  # Gamma log-likelihood
  ll <- n * alpha * log(beta) - n * lgamma(alpha) +
        (alpha - 1) * sum(log(x)) - beta * sum(x)
  # Jeffreys prior (in log form)
  lp <- -log(beta) + 0.5 * log(alpha * trigamma(alpha) - 1)
  ll + lp
}
```

```r
# 3) Random-walk MH sampler
mh_sampler <- function(iters, a0, b0, sa, sb) {
  # Create vectors to store the sampled values of alpha and beta
  chain_alpha <- numeric(iters)
  chain_beta  <- numeric(iters)
  # Initialize the chain at starting values
  chain_alpha[1] <- a0
  chain_beta[1]  <- b0

  # Run the sampler for n - 1 steps
  for (t in 1:(iters - 1)) {
    # Current values
    ca <- chain_alpha[t]
    cb <- chain_beta[t]

    # Propose new values from normal distributions centered at current values
    pa <- rnorm(1, ca, sa)
    pb <- rnorm(1, cb, sb)

    # Accept/reject step (only compute if both proposals are positive)
    if (pa > 0 && pb > 0) {
      logR <- log_post(pa, pb) - log_post(ca, cb)  # log of acceptance ratio

      # Accept if log(U) < logR, otherwise keep current values
      if (log(runif(1)) < logR) {
        chain_alpha[t + 1] <- pa
        chain_beta[t + 1]  <- pb
```

```
      } else {
        chain_alpha[t + 1] <- ca
        chain_beta[t + 1]  <- cb
      }
    } else {
      # Reject negative proposals by default
      chain_alpha[t + 1] <- ca
      chain_beta[t + 1]  <- cb
    }
  }

  # Return both chains as a matrix with column names
  cbind(alpha = chain_alpha, beta = chain_beta)
}
```

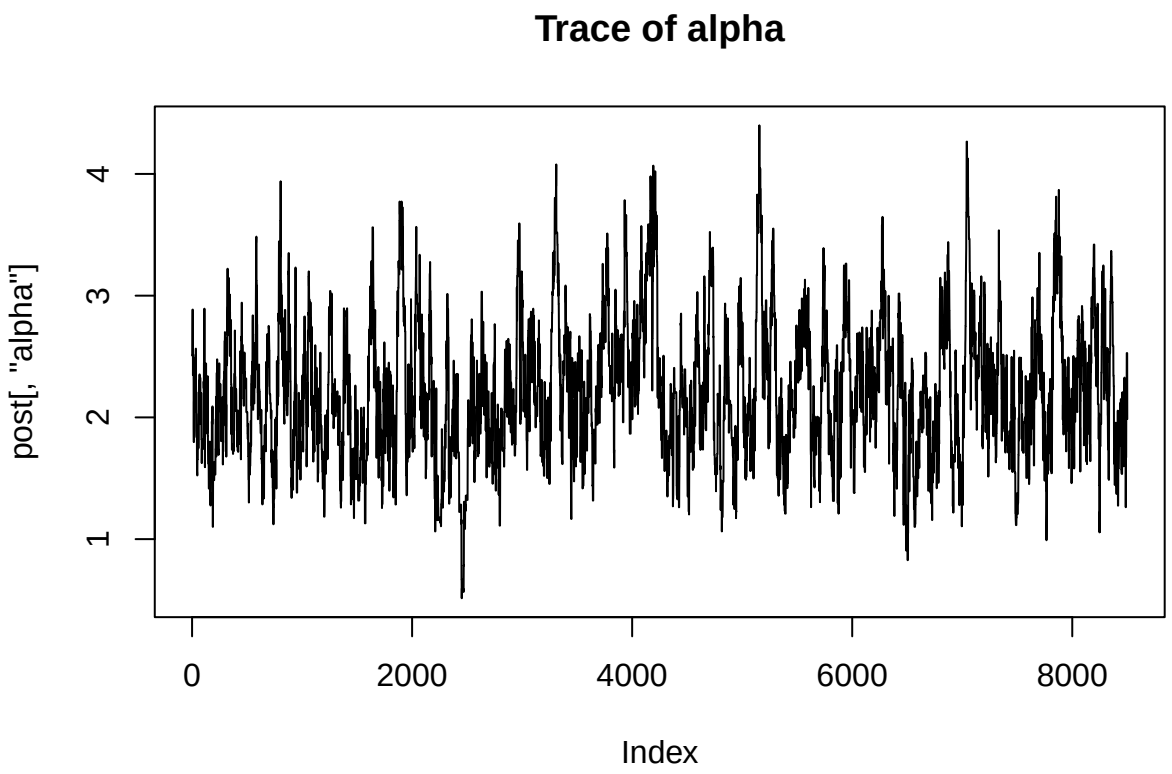
```
# Run sampler and drop burn-in
set.seed(123)
chain <- mh_sampler(10000, 1, 1, 0.3, 0.3)
burn <- 1500
post <- chain[(burn + 1):10000, ]

# Posterior summaries
cat("Mean alpha:", mean(post[, "alpha"]), "\n")
```

```
## Mean alpha: 2.20565
```

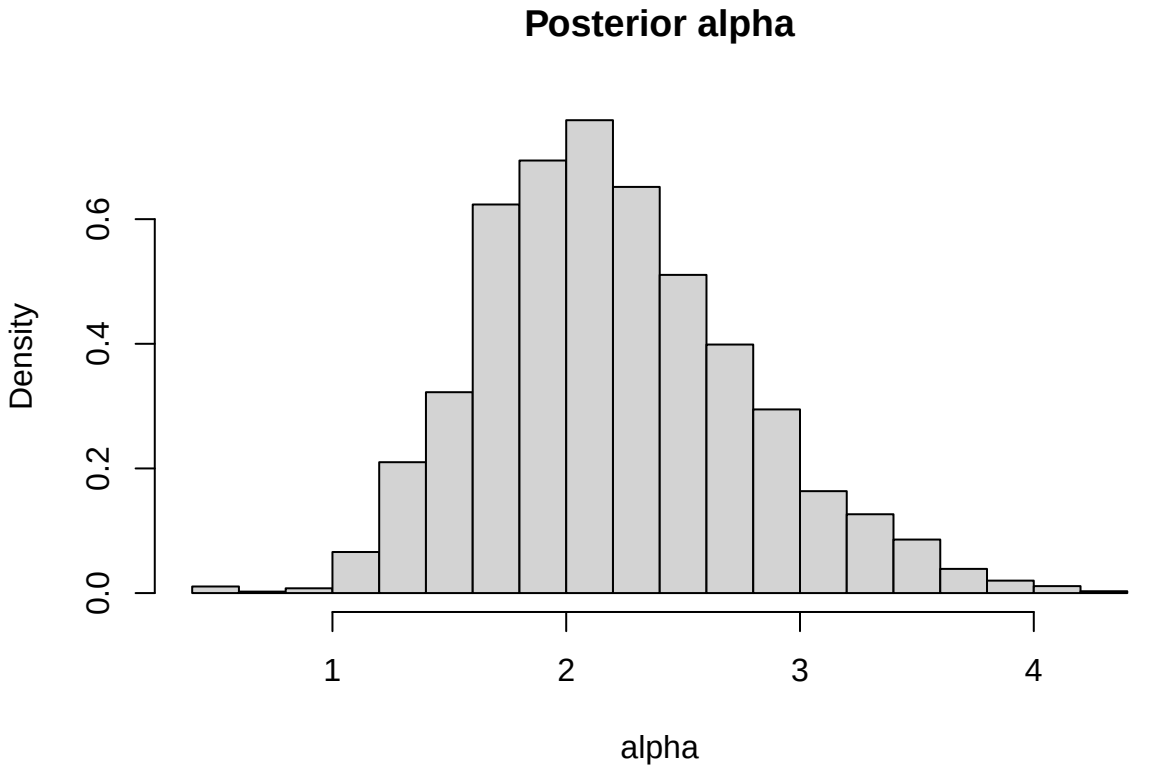
```
cat("Mean beta :", mean(post[, "beta"]),  "\n")
```

```
## Mean beta : 1.584053
```

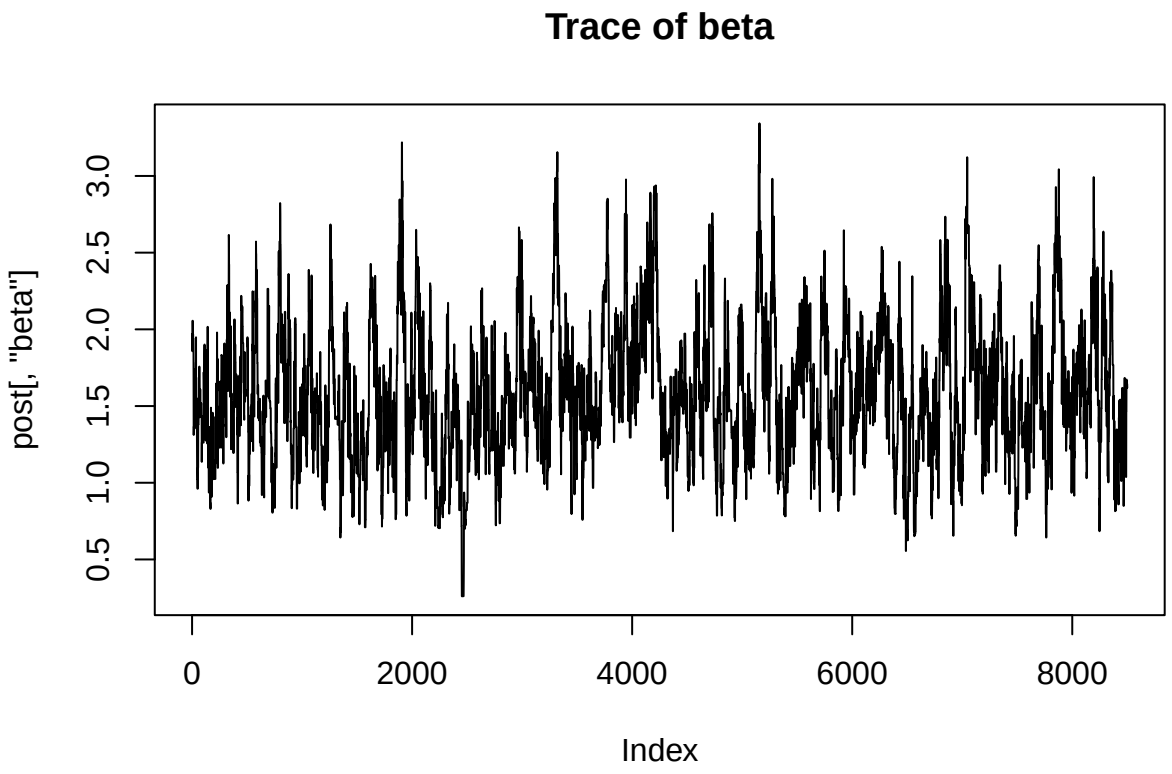
```
# Summary for alpha
plot(post[, "alpha"], type = 'l', main = 'Trace of alpha')
```



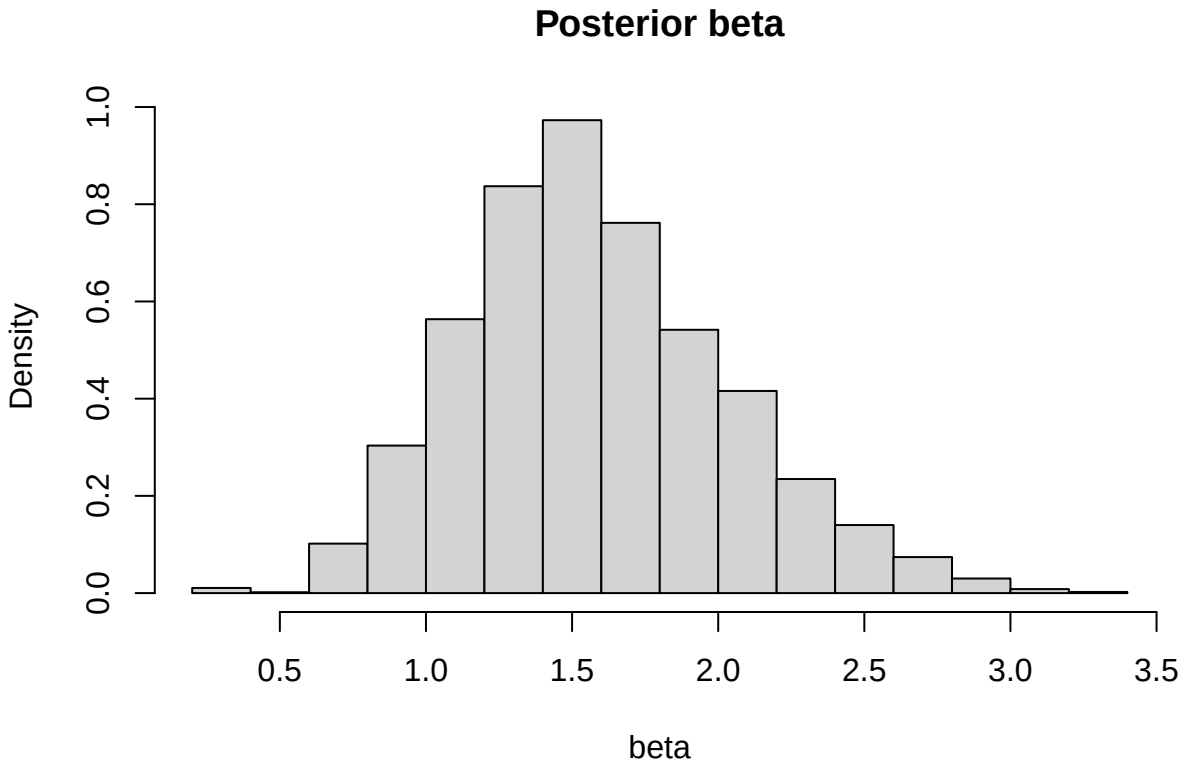

```r
hist(post[, "alpha"], main = 'Posterior alpha', xlab = 'alpha', freq = FALSE)
```




```r
# Summary for beta
# Summary for alpha
plot(post[, "beta"], type = 'l', main = 'Trace of beta')
```

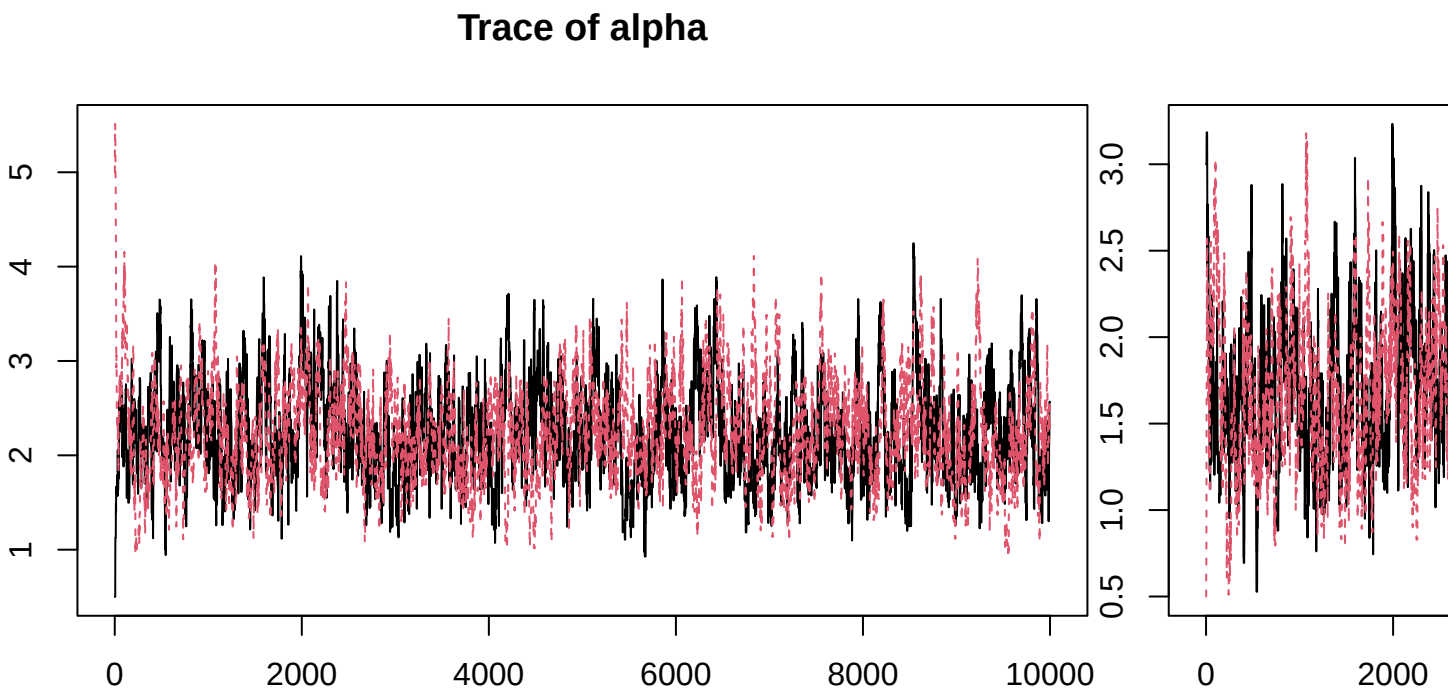



```r
hist(post[, "beta"], main = 'Posterior beta', xlab = 'beta', freq = FALSE)
```



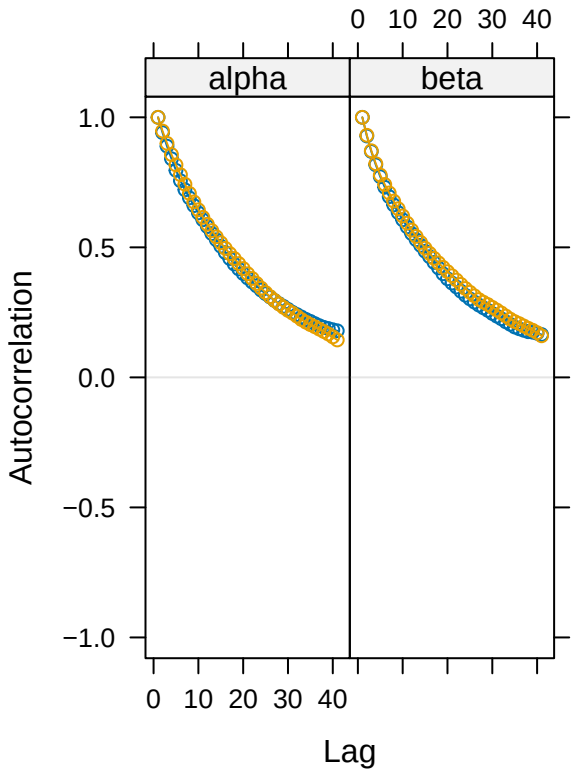
This highlights the strength of MCMC: even with poor starting values and a small sample size, the sampler converges to a posterior centered near the true parameters.

```r
library(coda)

# Reuse mh_sampler and log_post from Q2

# Run two chains from very different initial values
set.seed(123)
chain1 <- mh_sampler(10000, a0 = 0.5, b0 = 3, sa = 0.3, sb = 0.3)
chain2 <- mh_sampler(10000, a0 = 5,   b0 = 0.5, sa = 0.3, sb = 0.3)

# Convert to coda mcmc objects
mcmc1 <- mcmc(chain1)
mcmc2 <- mcmc(chain2)
mcmc_list <- mcmc.list(mcmc1, mcmc2)

# Traceplots for visual check
traceplot(mcmc_list)
```



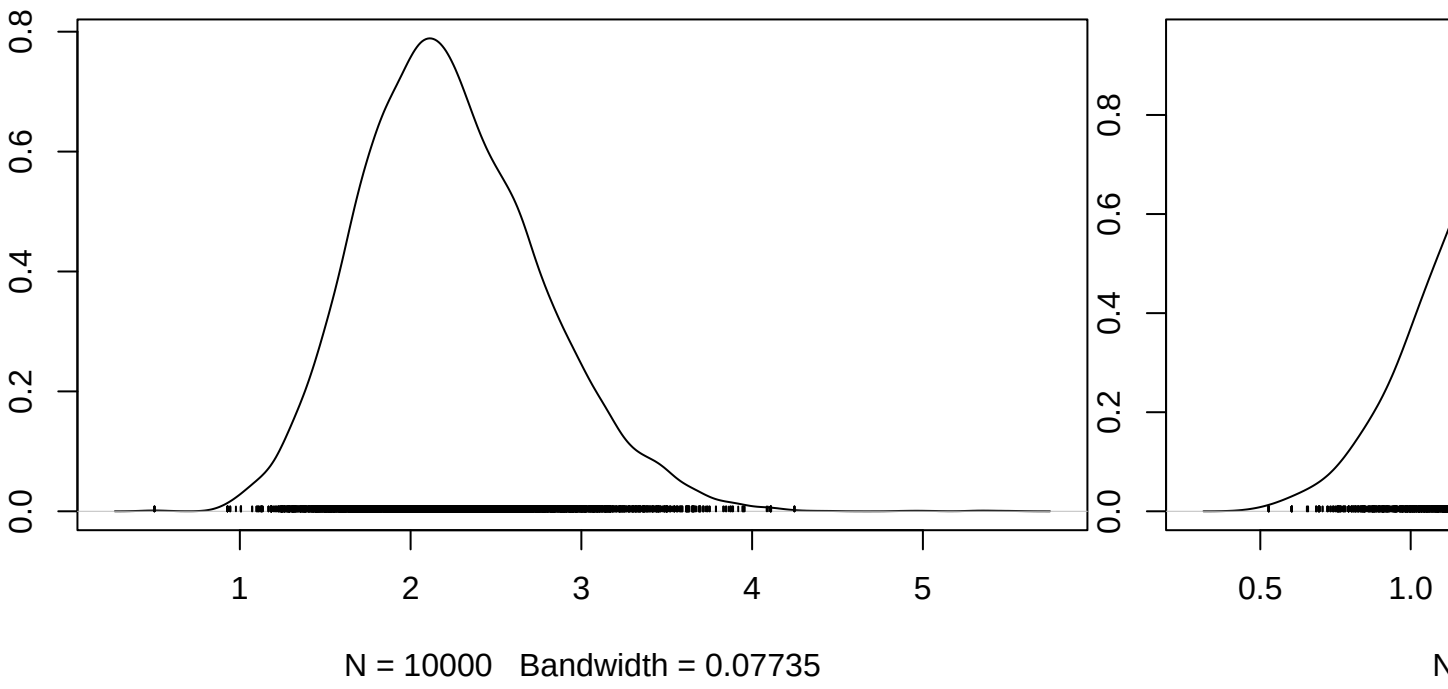
The traceplots for alpha and beta show that the two chains start from very different initial values, as expected from our overdispersed starting points. Early in the simulation, there is some visible discrepancy between the chains. However, both chains quickly converge and begin overlapping consistently, indicating good mixing and that stationarity was reached. This validates our choice of burn-in and confirms that both chains are sampling from the same posterior distribution.

```r
# Autocorrelation plots
acfplot(mcmc_list)
```



The autocorrelation plots show that both alpha and beta exhibit high autocorrelation at low lags, which gradually declines toward zero. This is expected for a random-walk Metropolis-Hastings sampler, where each sample is dependent on the previous one. The slow decay confirms some stickiness in the chain, but not enough to be problematic — it's a natural outcome of using correlated proposals and a relatively small step size.

```
# Density plots
densplot(mcmc_list)
```



The posterior density plots for both alpha and beta appear approximately normal and are centered near the true values (2.5 and 1.5, respectively), suggesting good mixing and accurate inference from the MCMC sampling.

```
# Gelman-Rubin diagnostic (PSRF close to 1 means convergence)
gelman_diag <- gelman.diag(mcmc_list)
print(gelman_diag)
```

```
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## alpha       1.01       1.03
## beta        1.01       1.02
##
## Multivariate psrf
##
## 1.01
```

The potential scale reduction factor was close to 1 for both parameters, suggesting that the chains have mixed well and likely converged to the target posterior.

```
# Geweke diagnostic (z-scores near 0 suggest convergence)
geweke_diag <- geweke.diag(mcmc_list)
print(geweke_diag)
```

```
## [[1]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##  alpha   beta
## 0.8022 1.1691
```

```
##
##
## [[2]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##   alpha    beta
## -0.1460 -0.2317
```

The z-scores for both chains are comfortably within the range (-2,2), with most values close to 0, suggesting that the early and late parts of the chains are sampling from the same distribution, indicating convergence.

```
# Raftery diagnostic (estimates how many more samples are needed)
raftery_diag <- raftery.diag(mcmc_list)
print(raftery_diag)
```

```
## [[1]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##        Burn-in  Total Lower bound  Dependence
##        (M)      (N)   (Nmin)       factor (I)
##  alpha 32       33090 3746         8.83
##  beta  29       30992 3746         8.27
##
##
## [[2]]
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##        Burn-in  Total Lower bound  Dependence
##        (M)      (N)   (Nmin)       factor (I)
##  alpha 42       46021 3746         12.30
##  beta  29       30453 3746          8.13
```

For both chains, the total number of samples needed is around 30,000 to 46,000, which is greater than the current post-burn sample size (8500).

The dependence factor ranges from 8 to 12, indicating some autocorrelation, which is common in MH samples.

Run of 10,000 iterations may be a bit short for highly accurate quantile estimation at the 2.5% tail.

```
# Heidelberger-Welch diagnostic (checks stationarity and convergence)
heidel_diag <- heidel.diag(mcmc_list)
print(heidel_diag)
```

```
## [[1]]
##
##       Stationarity start     p-value
##       test         iteration
## alpha passed       1         0.469
## beta  passed       1         0.274
```

```
##
##       Halfwidth Mean Halfwidth
##       test
## alpha passed    2.23 0.0660
## beta  passed    1.61 0.0512
##
## [[2]]
##
##       Stationarity start     p-value
##       test         iteration
## alpha passed       1         0.789
## beta  passed       1         0.728
##
##       Halfwidth Mean Halfwidth
##       test
## alpha passed    2.24 0.0679
## beta  passed    1.61 0.0533
```

The Heidelberger-Welch diagnostic passed both the stationary and halfwidth tests for all parameters in both chains. This confirms that the chains have converged to a stable distribution and that the posterior means are estimated with sufficient accuracy.

```
# Posterior summaries after burn-in
burn <- 2000
post1 <- mcmc(chain1[(burn+1):10000, ])
post2 <- mcmc(chain2[(burn+1):10000, ])

# Highest Posterior Density Intervals (95%)
print(HPDinterval(post1))
```

```
##           lower    upper
## alpha 1.2296771 3.205034
## beta  0.7936674 2.415598
## attr(,"Probability")
## [1] 0.95
```

```
print(HPDinterval(post2))
```

```
##          lower    upper
## alpha 1.205538 3.242571
## beta  0.827363 2.455671
## attr(,"Probability")
## [1] 0.95
```

The intervals are consistent between the two chains, and contain the true parameter values, alpha = 2.5, beta = 1.5. This reinforces convergence and confirms the posterior distributions are centered correctly.

```
# Load library and data
library(MCMCpack)
```

```
## Loading required package: MASS
```

```
## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)
```

```
## ## Copyright (C) 2003-2025 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park
```

```
## ##
## ## Support provided by the U.S. National Science Foundation
```

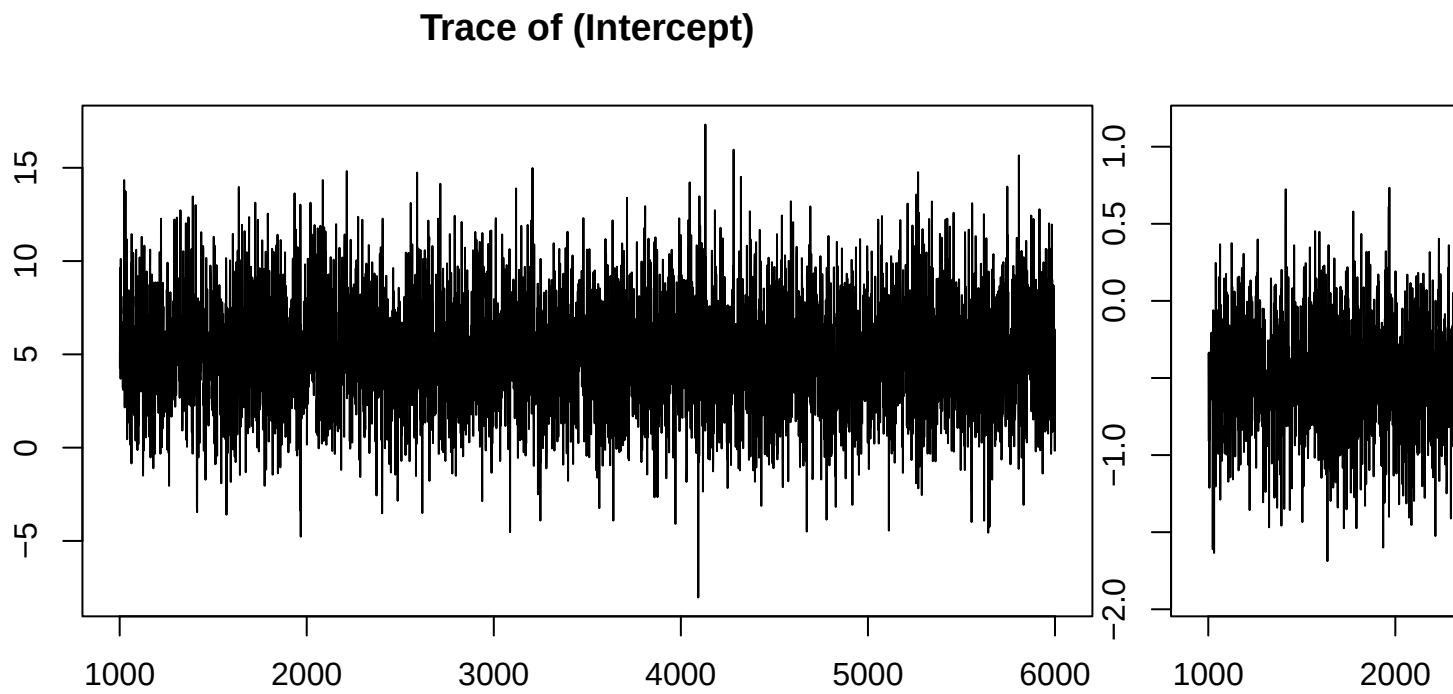
```
## ## (Grants SES-0350646 and SES-0350613)
## ##
```

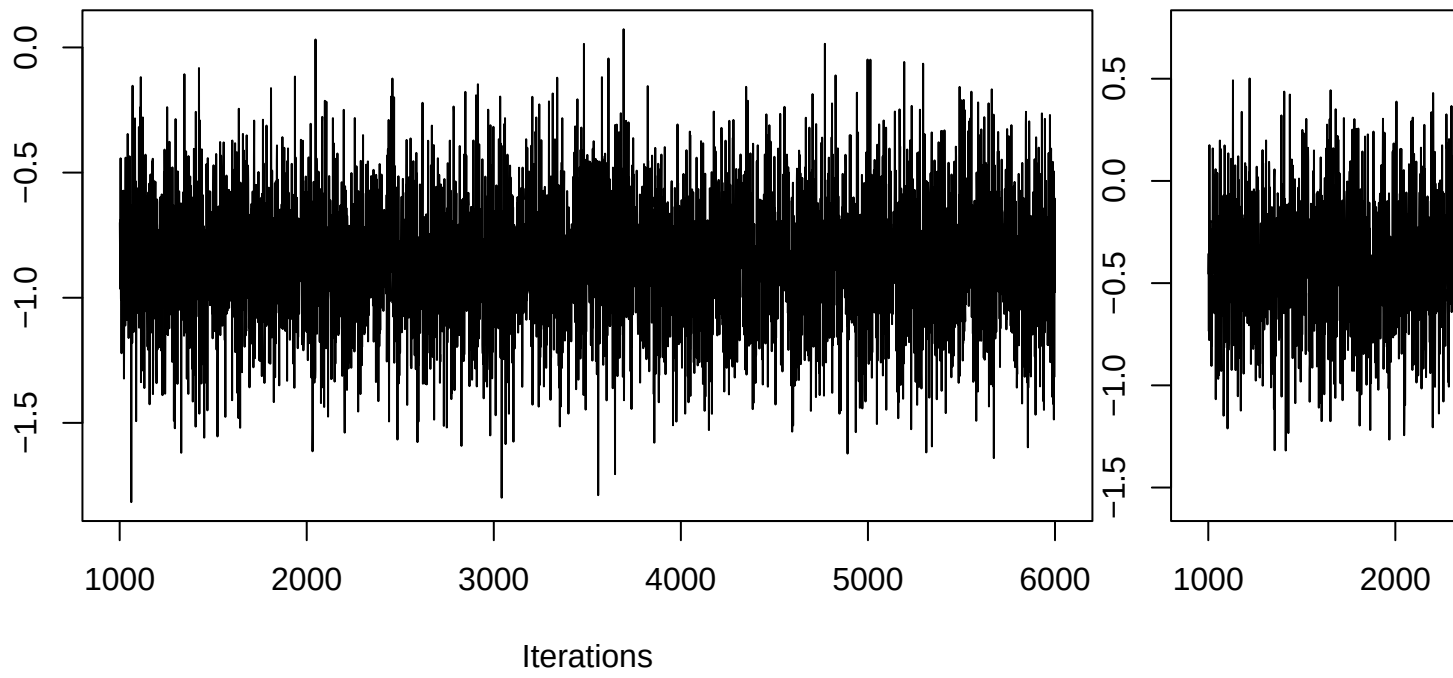
```
data(PErisk)

# Convert predictors to numeric (they may be ordered factors)
PErisk$prsexp2 <- as.numeric(as.character(PErisk$prsexp2))
PErisk$prscorr2 <- as.numeric(as.character(PErisk$prscorr2))

# Bayesian linear regression using MCMC
model_mcmc <- MCMCregress(
  formula = barb2 ~ gdpw2 + prsexp2 + prscorr2,
  data = PErisk,
  burnin = 1000,         # Discard early draws (burn-in)
  mcmc = 5000,           # Number of MCMC samples to keep
  thin = 1,              # Keep every sample
  b0 = 0, B0 = 0.001,    # Weak prior for regression coefficients
  c0 = 0.001, d0 = 0.001  # Weak prior for variance
)

# Posterior summary (means, SDs, quantiles)
summary(model_mcmc)
```

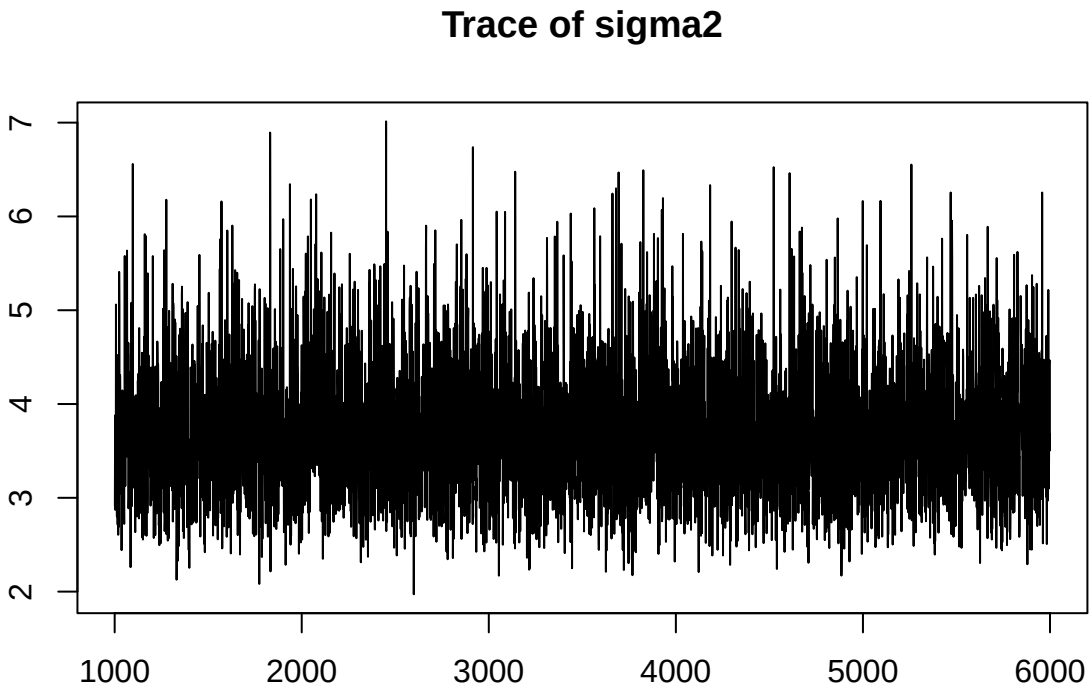
```
##
## Iterations = 1001:6000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean     SD Naive SE Time-series SE
## (Intercept)  5.0897 3.0201 0.042711       0.042257
## gdpw2       -0.4611 0.3731 0.005276       0.005276
## prsexp2     -0.8550 0.2562 0.003623       0.003507
## prscorr2    -0.4123 0.3001 0.004244       0.003743
## sigma2       3.7146 0.7160 0.010126       0.011014
##
## 2. Quantiles for each variable:
##
##                2.5%     25%     50%     75%   97.5%
## (Intercept) -0.6403  3.0517  5.0985  7.0813 11.1791
## gdpw2       -1.2097 -0.7041 -0.4588 -0.2124  0.2473
## prsexp2     -1.3616 -1.0201 -0.8560 -0.6877 -0.3409
## prscorr2    -1.0102 -0.6125 -0.4122 -0.2133  0.1744
## sigma2       2.5608  3.1935  3.6319  4.1464  5.3047
```

The MCMC regression suggests that gdp2, prsexp2, and prscorr2 are all negatively associated with barb2. Only prsexp2 has a credible interval that does not contain 0, suggesting strong evidence that it is negatively associated with barb2. The other predictors have wider intervals that include 0, implying more posterior uncertainty about their effect. The posterior mean for sigma^2 is approximately 3.71, indicating the estimated variance of the error term.

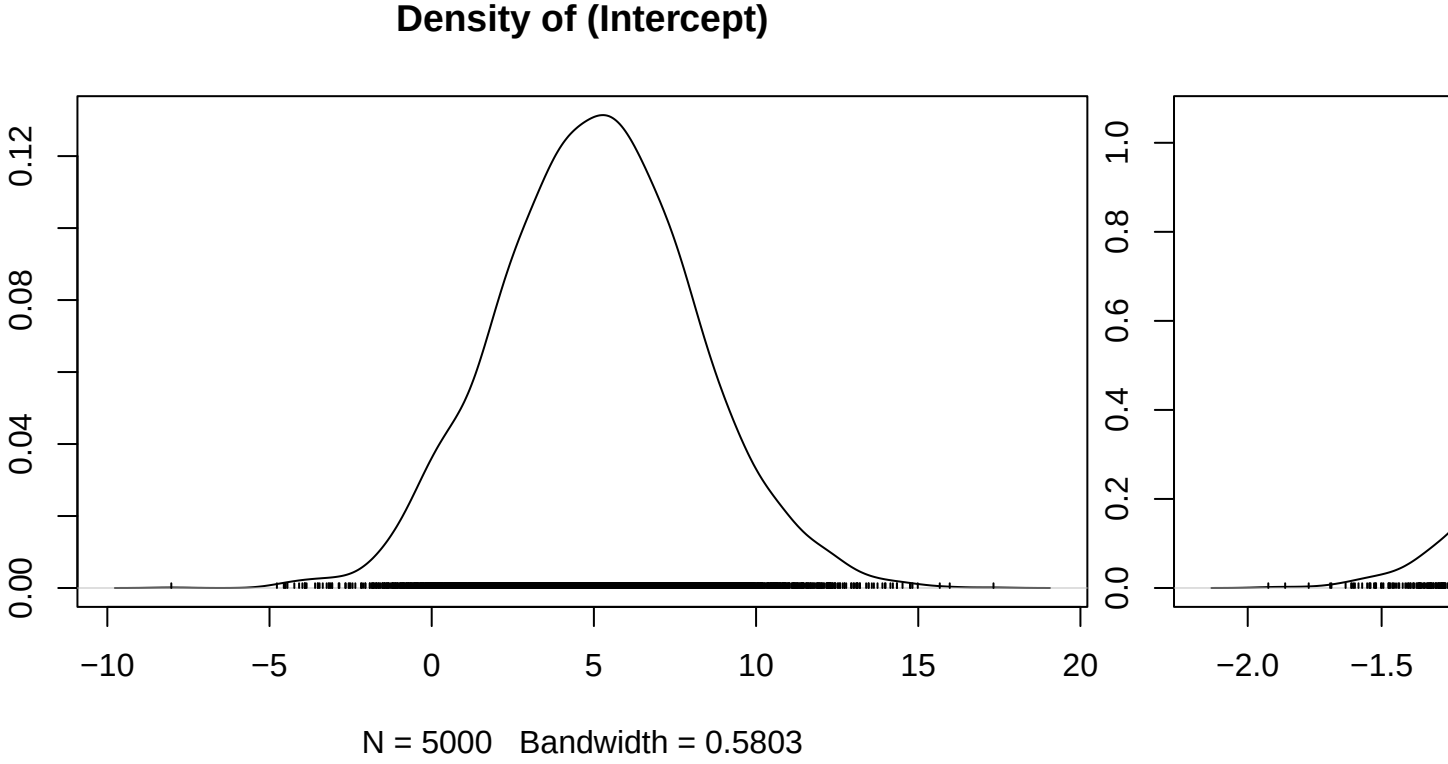
```
# Diagnostic plots
traceplot(model_mcmc)
```

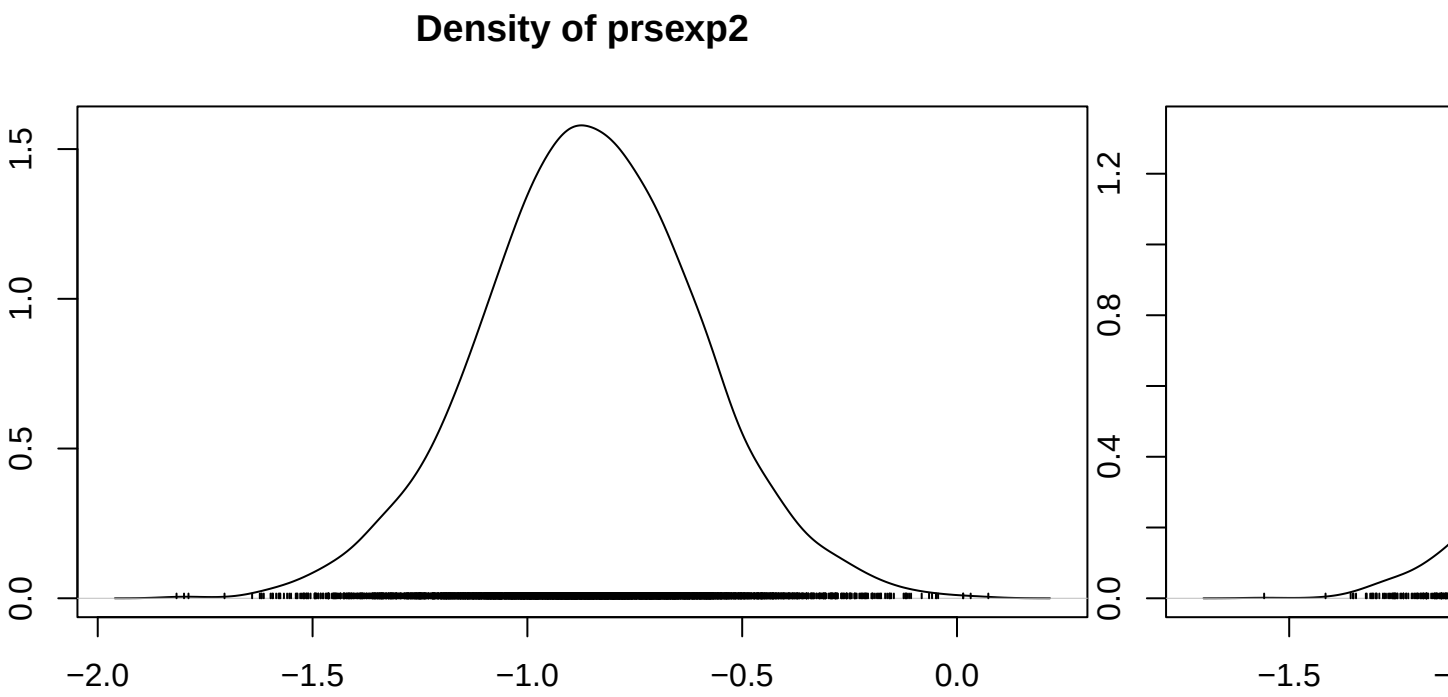

Trace of (Intercept)
15
10
5
0
−5
1.0
0.5
0.0
−1.0
−2.0
1000
2000
3000
4000
5000
6000
1000
2000
Iterations

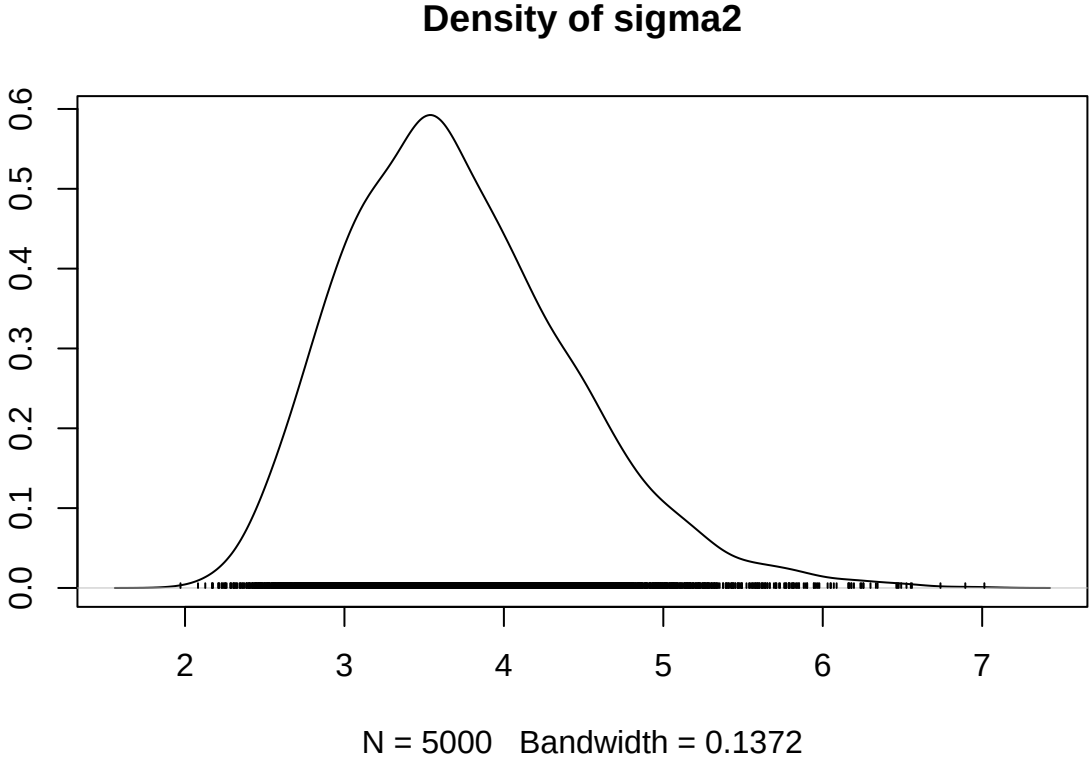

Trace of prsexp2
0.0
−0.5
−1.0
−1.5
0.5
0.0
−0.5
−1.0
−1.5
1000
2000
3000
4000
5000
6000
1000
2000
Iterations




```
densplot(model_mcmc)
```







The trace plots show good mixing after burn-in, and the density plots are fairly bell-shaped, suggesting convergence.

```
# 95% HPD intervals for each parameter
HPDinterval(model_mcmc)
```

```
##                 lower      upper
```

```
## (Intercept) -0.7733683 10.9684172
## gdpw2       -1.2113541  0.2461394
## prsexp2     -1.3654739 -0.3455490
## prscorr2    -1.0029474  0.1758189
## sigma2       2.4198275  5.0911357
## attr(,"Probability")
## [1] 0.95
```

The HPD for prsexp2 is approximately (–1.37, –0.35), meaning there's a 95% probability that the true coefficient lies within this range. Since this interval does not include zero, we can say with high confidence that prsexp2 has a negative effect on barb2.

The HPDs for gdpw2 and prscorr2 do include zero, which implies more posterior uncertainty about their effects — we cannot rule out the possibility that their impact on the outcome is negligible.

```
# Frequentist linear model for comparison
lm_fit <- lm(barb2 ~ gdpw2 + prsexp2 + prscorr2, data = PErisk)
summary(lm_fit)
```

```
##
## Call:
## lm(formula = barb2 ~ gdpw2 + prsexp2 + prscorr2, data = PErisk)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6665 -1.1611 -0.0556  1.0572  4.2043
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.1508     2.9581   1.741  0.08694 .
## gdpw2        -0.4677     0.3643  -1.284  0.20428
## prsexp2      -0.8577     0.2524  -3.398  0.00123 **
## prscorr2     -0.4104     0.2931  -1.400  0.16682
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.897 on 58 degrees of freedom
## Multiple R-squared:  0.5332, Adjusted R-squared:  0.5091
## F-statistic: 22.08 on 3 and 58 DF,  p-value: 1.157e-09
```

The frequentist model yields similar conclusions.

gdpw2: Estimate = –0.468, p = 0.204 (not significant)

prsexp2: Estimate = –0.858, p = 0.001 (significant)

prscorr2: Estimate = –0.410, p = 0.167 (not significant)

Just like in the Bayesian model, only prsexp2 is statistically significant at conventional levels.

However, the Bayesian model provides full posterior distributions for the parameters, not just point estimates and standard errors.