Task 5 - MCMC for Bayesian Inference

MCMC for Bayesian Inference

Setting up Parameters

```
years_exp <- salary_df$YearsExperience
salary <- salary_df$Salary
n <- length(salary)

#years_exp_std <- scale(years_exp)[,1]
X <- cbind(1, years_exp)

beta0_prior_mean <- mean(salary)
beta0_prior_var <- (15000)^2
beta1_prior_mean <- 0
beta1_prior_var <- (1500)^2
sigma2_prior_a <- 2
sigma2_prior_b <- 10000

log_posterior <- function(params, y, X) {
  beta0 <- params[1]
  beta1 <- params[2]
  sigma2 <- params[3]

  if (sigma2 <= 0) return(-Inf)

  mu <- X %*% c(beta0, beta1)
  log_lik <- sum(dnorm(y, mu, sqrt(sigma2), log = TRUE))

  # Priors
  log_prior_beta0 <- dnorm(beta0, beta0_prior_mean, sqrt(beta0_prior_var), log = TRUE)
  log_prior_beta1 <- dnorm(beta1, beta1_prior_mean, sqrt(beta1_prior_var), log = TRUE)
  log_prior_sigma2 <- dgamma(1/sigma2, sigma2_prior_a, sigma2_prior_b, log = TRUE) - 2*log(sigma2)

  return(log_lik + log_prior_beta0 + log_prior_beta1 + log_prior_sigma2)
}

metropolis_hastings <- function(n_iter, y, X, initial_values, proposal_sd) {
  n_params <- length(initial_values)
  samples <- matrix(0, n_iter, n_params)
  current <- initial_values
  n_accepted <- 0

  for (i in 1:n_iter) {
    proposal <- current + rnorm(n_params, 0, proposal_sd)

    log_ratio <- log_posterior(proposal, y, X) - log_posterior(current, y, X)
```

```
    if (log(runif(1)) < log_ratio) {
      current <- proposal
      n_accepted <- n_accepted + 1
    }

    samples[i, ] <- current
  }

  cat("Metropolis-Hastings acceptance rate:", n_accepted / n_iter, "\n")
  return(samples)
}
```

Running Metropolis-Hastings MCMC

```
n_iter <- 10000
burn_in <- 2000
initial_values <- c(mean(salary), 0, var(salary))
proposal_sd <- c(10000, 2000, 500)

mh_samples <- metropolis_hastings(n_iter, salary, X, initial_values, proposal_sd)
```

```
## Metropolis-Hastings acceptance rate: 0.2671
```

```
mh_samples_post_burnin <- mh_samples[(burn_in+1):n_iter, ]
mh_samples_post_burnin[, 3] <- sqrt(mh_samples_post_burnin[, 3])
```

```
mh_mcmc <- mcmc(mh_samples_post_burnin)
summary(mh_mcmc)
```

```
##
## Iterations = 1:8000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 8000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##       Mean        SD  Naive SE Time-series SE
## [1,] 59968 6.959e+03 77.806432       286.7887
## [2,]  3352 1.073e+03 11.993650        40.7051
## [3,] 27414 9.372e-02  0.001048         0.0394
##
## 2. Quantiles for each variable:
##
##       2.5%   25%   50%   75% 97.5%
## var1 46113 55416 59959 64503 73502
## var2  1253  2625  3363  4089  5378
## var3 27414 27414 27414 27414 27415
```

```
cat("\nEffective Sample Size for each variable\n")
```

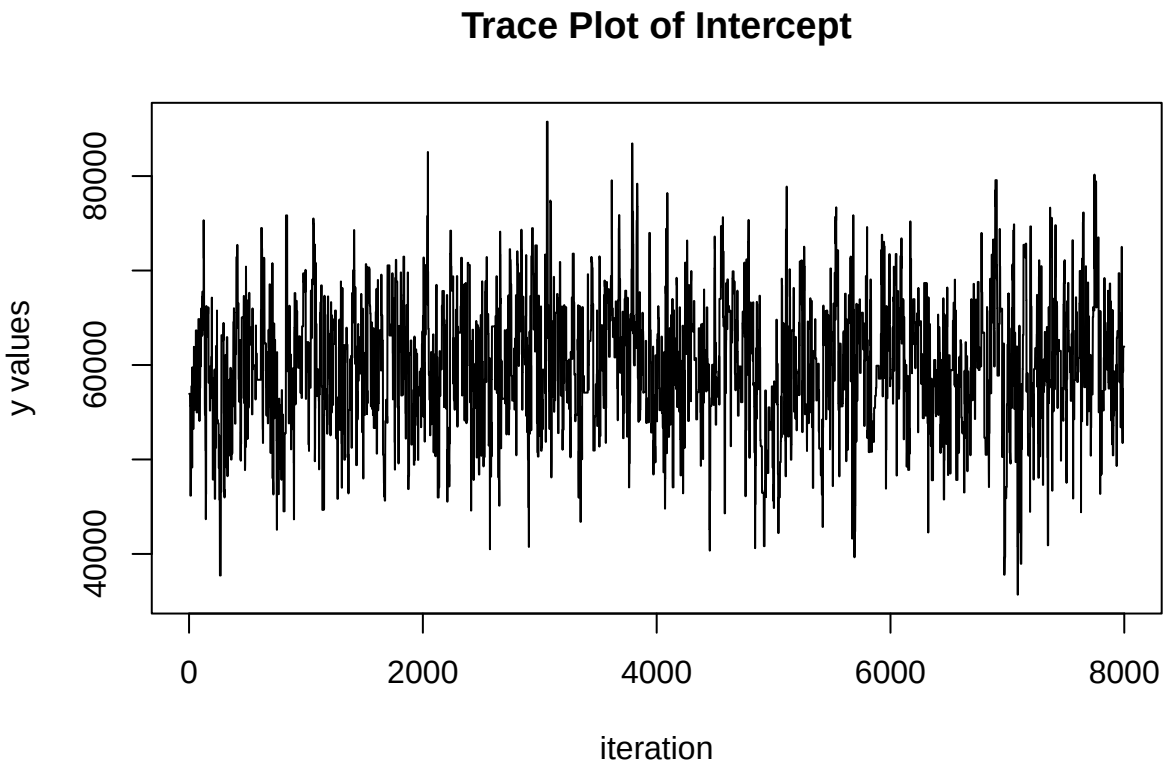
```
##
## Effective Sample Size for each variable
```

```r
effectiveSize(mh_mcmc)
```

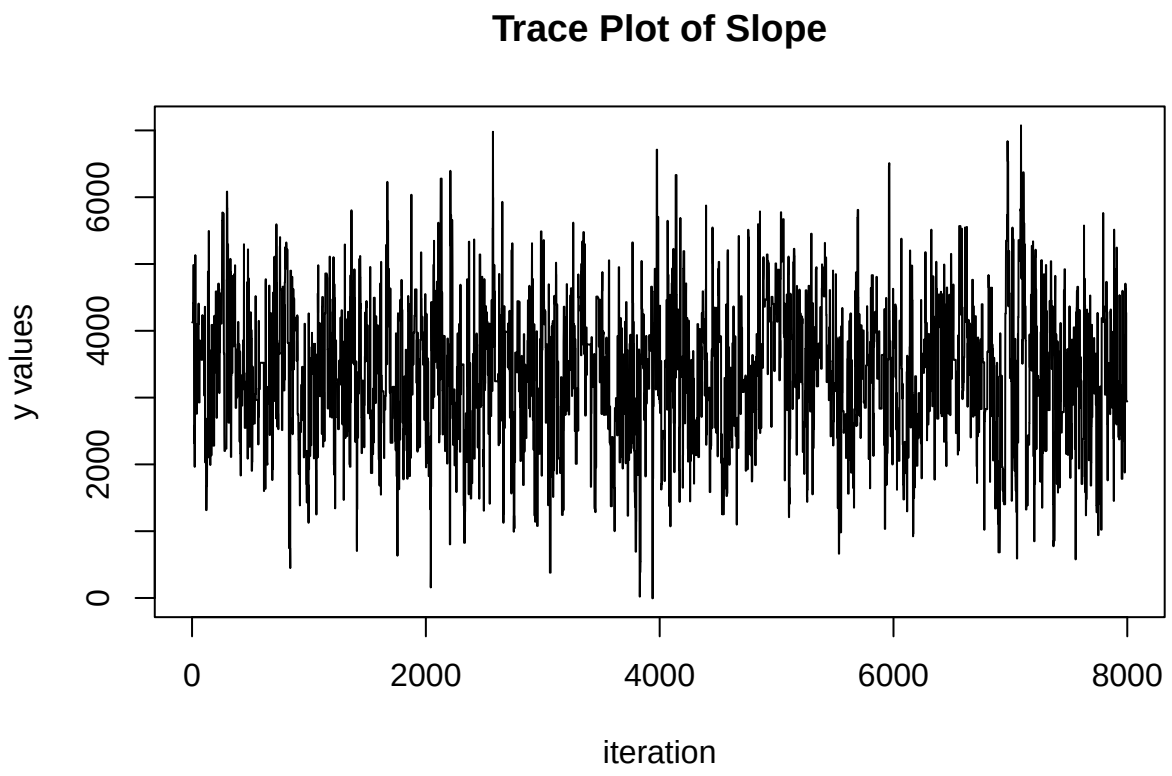
```
##       var1       var2       var3
## 588.839374 694.536530   5.657413
```

Plots

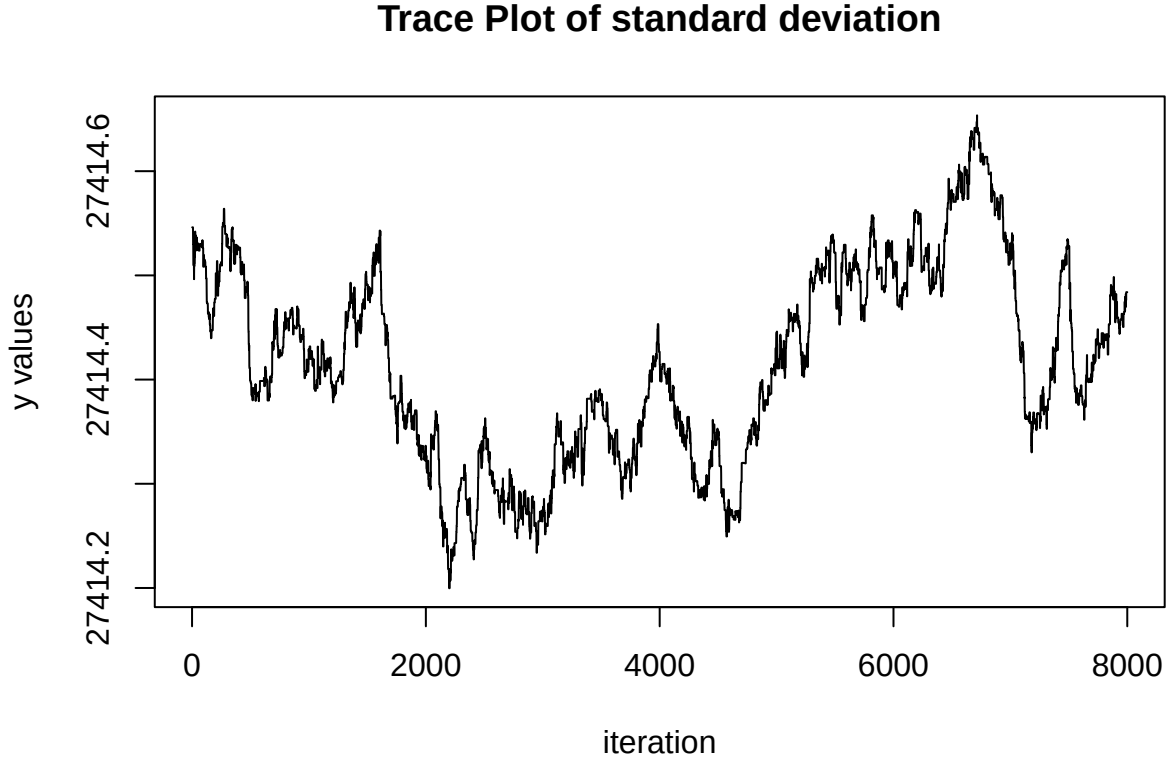
```r
plot(1:length(mh_samples_post_burnin[,1]), mh_samples_post_burnin[,1], type = "l", ylab="y values", xlab
```




```r
plot(1:length(mh_samples_post_burnin[,2]), mh_samples_post_burnin[,2], type = "l", ylab="y values", xlab
```




```
plot(1:length(mh_samples_post_burnin[,3]), mh_samples_post_burnin[,3], type = "l", ylab="y values", xlab
```

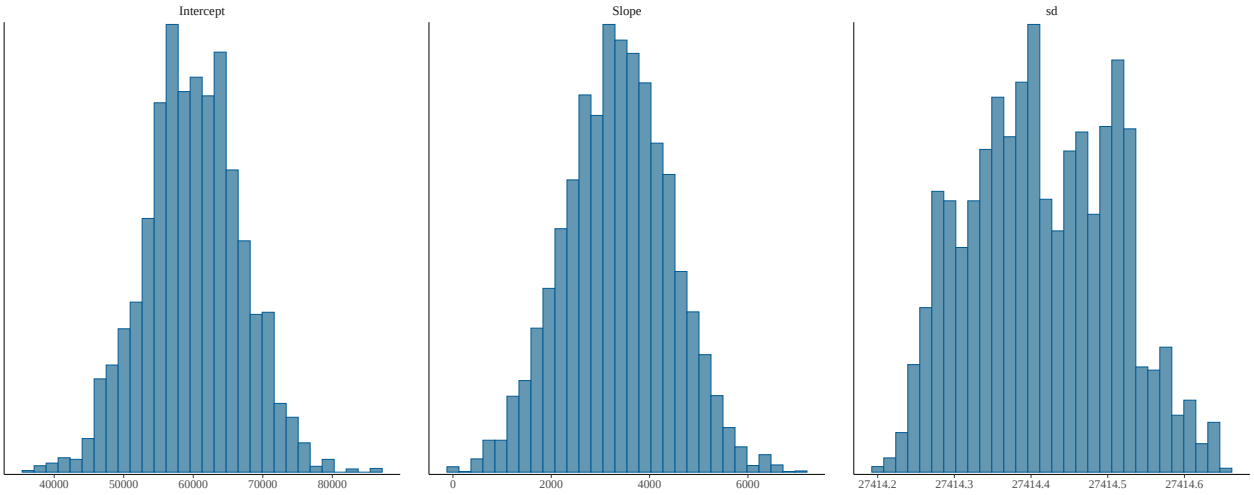



```
mh_bayes_samples_fin <- as.array(mh_samples_post_burnin)

dimnames(mh_bayes_samples_fin) <- list(1:nrow(mh_bayes_samples_fin), c("Intercept", "Slope", "sd"))
```

```
mcmc_hist(mh_bayes_samples_fin, pars = c("Intercept", "Slope", "sd"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# bivariate marginal posterior distribution

if (requireNamespace("hexbin", quietly = TRUE)) {
  mcmc_hex(mh_bayes_samples_fin, pars = c("Intercept", "Slope"))
}
```