Bayesian Lab2

Xuan Wang & Lepeng Zhang

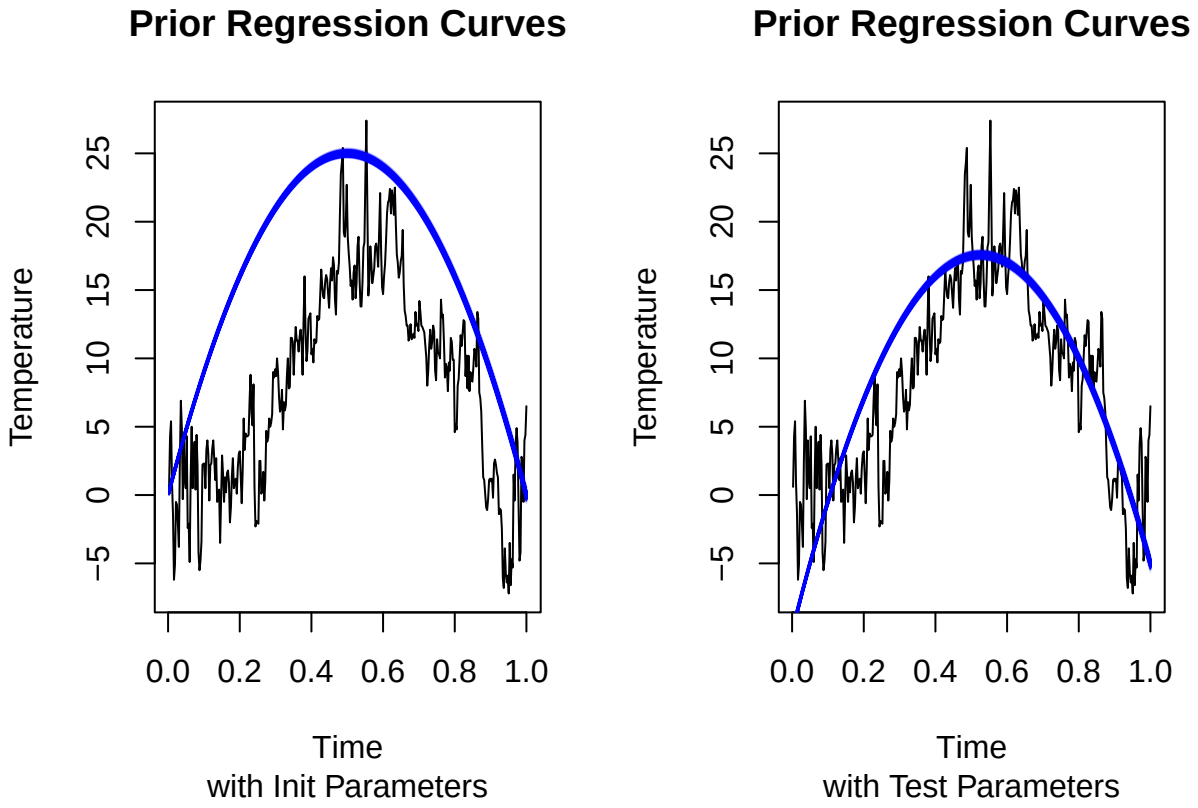
2024-04-25

Question 1 Linear and polynomial regression

1a)

```r
# Load necessary libraries
library(readxl)
library(mvtnorm)
set.seed(12345)

# Import the dataset
data <- read_xlsx("Linkoping2022.xlsx")

# Create time variable
data$time <- seq_along(data$temp) / 365

# Define prior hyperparameters
mu0 <- c(0, 100, -100)
Omega0 <- 0.01 * diag(3)
nu0 <- 1
sigma02 <- 1

# Simulate draws from the joint prior
simulate_prior <- function(mu0, Omega0, nu0, signma02, n) {
  beta <- rmvnorm(n, mu0, Omega0)
  sigma2 <- 1 / rchisq(n, nu0, ncp = nu0 * sigma02)
  cbind(beta, sigma2)
}

# Compute regression curves for each draw from the prior
compute_regression_curve <- function(draw) {
  beta0 <- draw[1]
  beta1 <- draw[2]
  beta2 <- draw[3]
  beta0 + beta1 * data$time + beta2 * data$time^2
}

prior_draws_init <- simulate_prior(mu0, Omega0, nu0, signma02, 1000)

# Define test prior hyperparameters
mu0_test <- c(-10, 105, -100)
Omega0_test <- 0.01 * diag(3)
nu0_test <- 4
```

```r
sigma02_test <- 0.1

prior_draws_test <- simulate_prior(mu0_test, Omega0_test, nu0_test, signma02_test, 1000)

#Plot the prior regression curve
par(mfrow = c(1, 2))
plot(data$time, data$temp, type = "l", main = "Prior Regression Curves", sub ="with Init Parameters", xl
for (i in 1:nrow(prior_draws_init)) {
  lines(data$time, compute_regression_curve(prior_draws_init[i,]), col = rgb(0, 0, 1, alpha = 0.1))
}

plot(data$time, data$temp, type = "l", main = "Prior Regression Curves", sub = "with Test Parameters", x
for (i in 1:nrow(prior_draws_test)) {
  lines(data$time, compute_regression_curve(prior_draws_test[i,]), col = rgb(0, 0, 1, alpha = 0.1))
}
```




```r
par(mfrow = c(1, 1)) #Reset plot layout
```

1b)

The joint posterior distribution of the parameters in the Bayesian linear regression model is updated from the prior distribution using the observed data. The hyperparameters of the posterior distribution, `Omega_n`, `mu_n`, `nu_n`, and `sigma02_n`, are calculated as follows:

- `Omega_n` is the precision matrix (the inverse of the covariance matrix) of the multivariate normal prior for the regression coefficients (0, 1, 2). It is updated based on the prior precision matrix (Omega0) and the data matrix X (which contains the time and time^2 variables). The formula to update Omega_n is:

$$\Omega_n = (\Omega_0^{-1} + X^T X)^{-1}$$

- `mu_n` is the mean vector of the multivariate normal prior for the regression coefficients. It is updated based on the prior mean vector (mu0), the prior precision matrix (Omega0), and the response variable (temp). The formula to update mu_n is:

$$\mu_n = \Omega_n(\Omega_0^{-1}\mu_0 + X^T temp)$$

- `nu_n` is the degrees of freedom of the inverse chi-square prior for the variance parameter (^2). It is updated based on the prior degrees of freedom (nu0) and the number of observations(n). The formula to update nu_n is:

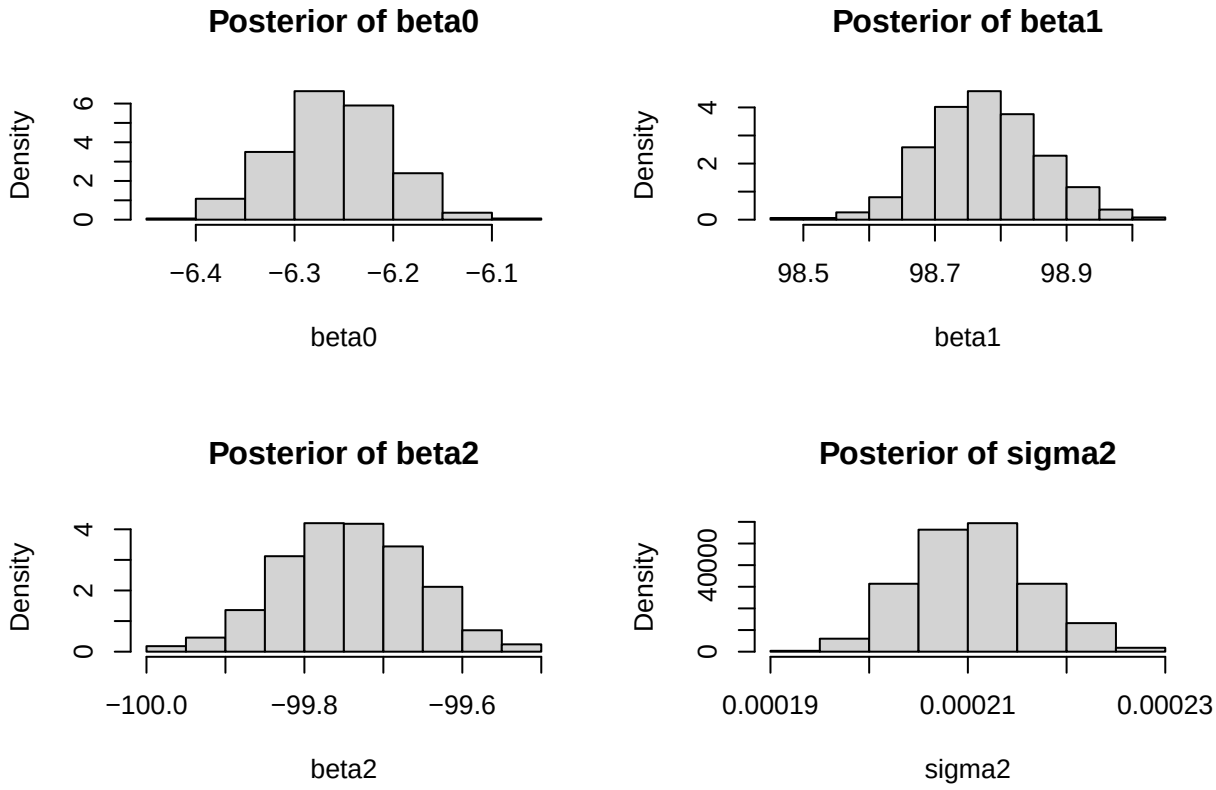
$$\nu_n = \nu_0 + n$$

- `sigma_n^2` is the scale parameter of the inverse chi-square prior for the variance parameter. It is updated based on the prior degrees of freedom (nu0), the prior scale parameter (sigma02), the response variable (temp), the data matrix X, and the prior and updated mean vectors (mu0 and mu_n). The formula to update sigma_n2 is:

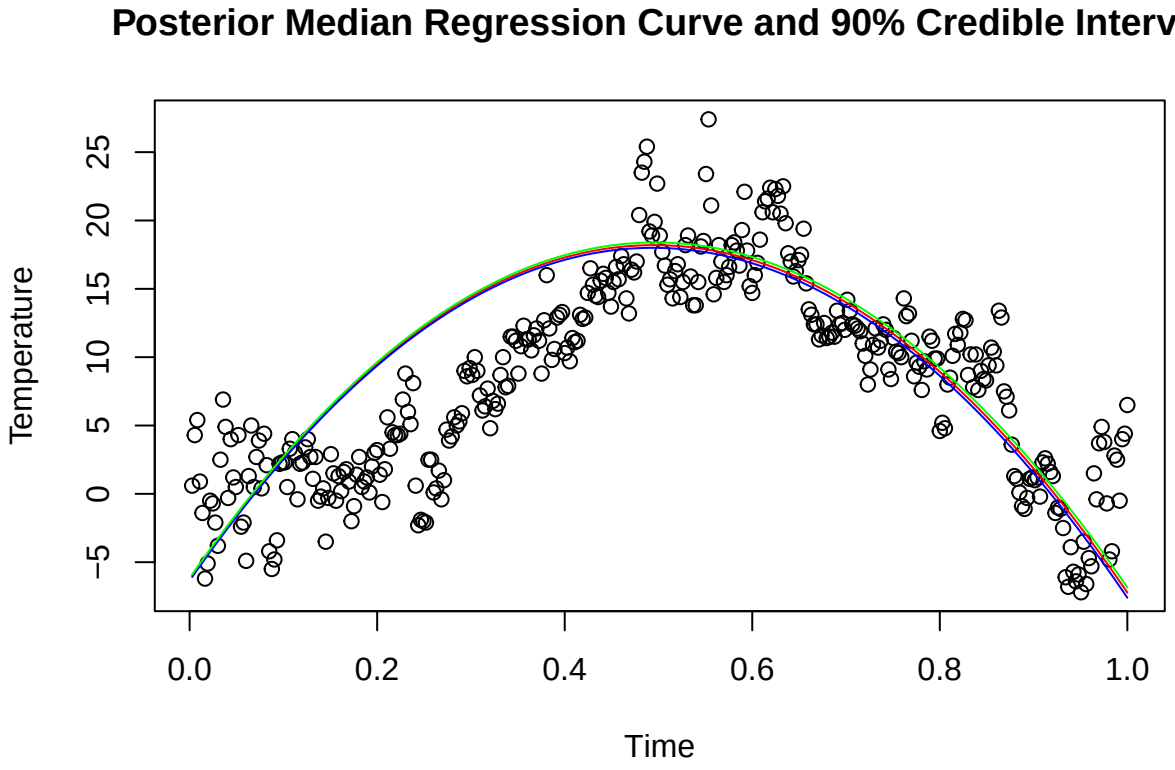
$$\sigma_n^2 = \frac{\nu_0 \cdot \sigma_0^2 + (y - X \cdot \mu_0)^T \cdot (y - X \cdot \mu_0) + \mu_0^T \cdot \Omega_0^{-1} \cdot \mu_0 - \mu_n^T \cdot \Omega_n^{-1} \cdot \mu_n}{\nu_n}$$

```r
set.seed(12345)

simulate_posterior <- function(n, data, mu0, Omega0, nu0, sigma02) {
  # Placeholder for the posterior draws
  posterior_draws <- matrix(0, nrow = n, ncol = 4)

  X <- cbind(1, data$time, data$time^2)

  for (i in 1:n) {
    # Update the hyperparameters based on the data and the prior
    Omega_n <- solve(solve(Omega0) + t(X) %*% X)
    mu_n <- Omega_n %*% (solve(Omega0) %*% mu0 + t(X) %*% data$temp)
    #mu_n <- Omega_n %*% (t(X) %*% data$temp + Omega0_inv %*% mu0)

    nu_n <- nu0 + length(data$temp)
    sigma_n2 <- (nu0 * sigma02 + t(data$temp - X %*% mu0) %*% (data$temp - X %*% mu0) + t(mu0) %*% solv

    # Draw from the posterior distribution
    beta <- rmvnorm(1, mu_n, Omega_n)
    sigma2 <- 1 / rchisq(1, nu_n, ncp = nu_n * sigma_n2)

    # Store the draw
    posterior_draws[i,] <- c(beta, sigma2)
  }

  return(posterior_draws)
}

#mu0 <- c(-10, 105, -100)
#Omega0 <- 0.01 * diag(3)
#nu0 <- 4
```

```r
#sigma02 <- 0.1
posterior_draws <- simulate_posterior(1000, data, mu0, Omega0, nu0, sigma02)


# Plot a histogram for each marginal posterior of the parameters
par(mfrow = c(2, 2))
hist(posterior_draws[,1], freq = FALSE, main = "Posterior of beta0", xlab = "beta0")
hist(posterior_draws[,2], freq = FALSE, main = "Posterior of beta1", xlab = "beta1")
hist(posterior_draws[,3], freq = FALSE, main = "Posterior of beta2", xlab = "beta2")
hist(posterior_draws[,4], freq = FALSE, main = "Posterior of sigma2", xlab = "sigma2")
```




```r
par(mfrow = c(1, 1))  # Reset the plot layout


# Make a scatter plot of the temperature data and overlay a curve for the posterior median of the regre
plot(data$time, data$temp, main = "Posterior Median Regression Curve and 90% Credible Interval", xlab =
lines(data$time, t(apply(posterior_draws[,1:3], 2, median)) %*% t(cbind(1, data$time, data$time^2)), col
lines(data$time, t(apply(posterior_draws[,1:3], 2, quantile, probs = 0.05)) %*% t(cbind(1, data$time, da
lines(data$time, t(apply(posterior_draws[,1:3], 2, quantile, probs = 0.95)) %*% t(cbind(1, data$time, da
```



1c)

Given the quadratic regression model

$$\text{temp} = \beta_0 + \beta_1 \cdot \text{time} + \beta_2 \cdot \text{time}^2 + \epsilon$$

where

$$\epsilon \sim N(0, \sigma^2)$$

, the expected temperature at time t is given by

$$f(t) = E[\text{temp}|\text{time} = t] = \beta_0 + \beta_1 \cdot t + \beta_2 \cdot t^2$$

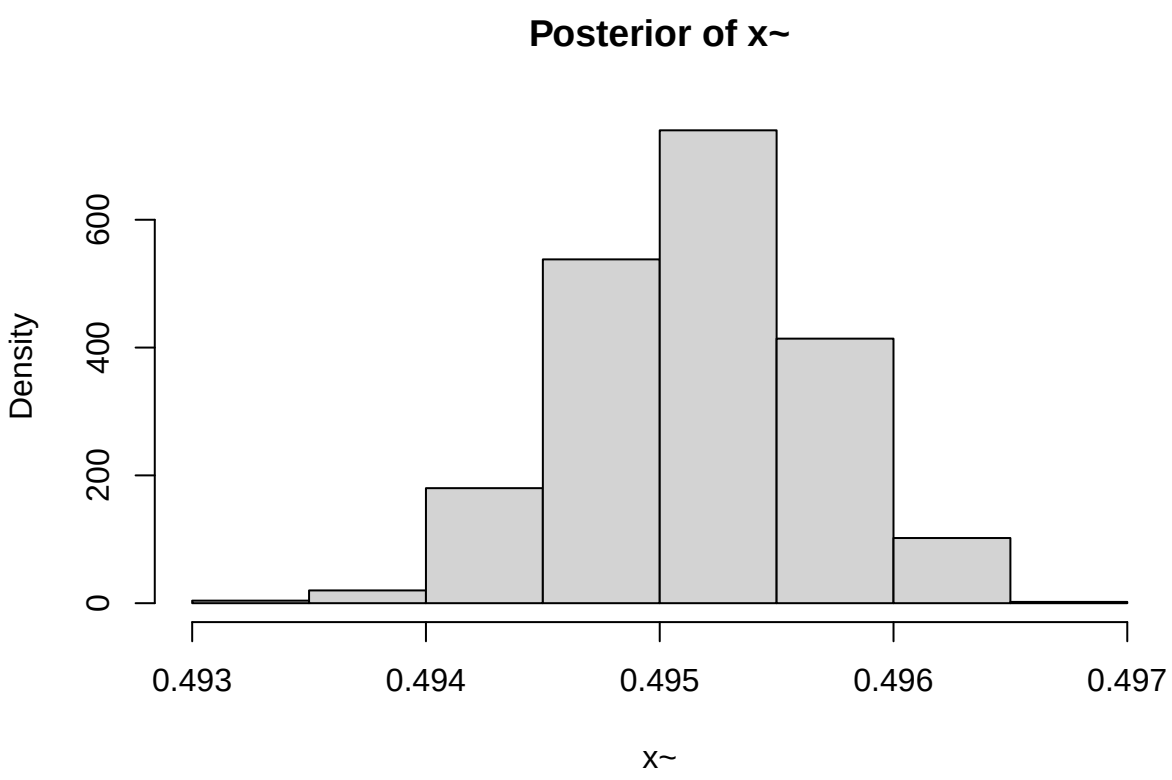
The time $\tilde{x}$ that maximizes this expected temperature can be found by setting the derivative of $f(t)$ with respect to t equal to zero and solving for t, which gives

$$\tilde{x} = -\frac{\beta_1}{2\beta_2}$$

In the `simulate_x_tilde` function, we extract the parameters β_1 and β_2 from each draw of the posterior distribution and compute $\tilde{x}$ using the formula above.

```
simulate_x_tilde <- function(posterior_draws) {
  # Placeholder for the simulations
  x_tilde_simulations <- numeric(nrow(posterior_draws))

  for (i in 1:nrow(posterior_draws)) {
    # Extract the parameters
    beta1 <- posterior_draws[i, 2]
```

```r
    beta2 <- posterior_draws[i, 3]

    # Compute the time that maximizes the expected temperature
    x_tilde <- -beta1 / (2 * beta2)

    # Store the simulation
    x_tilde_simulations[i] <- x_tilde
  }

  return(x_tilde_simulations)
}

x_tilde_simulations <- simulate_x_tilde(posterior_draws)
hist(x_tilde_simulations, freq = FALSE, main = "Posterior of x~", xlab = "x~")
```



1d)

```r
# Define the order of the polynomial
p <- 10

# Define mu0
mu0 <- c(0, rep(100, p - 1), rep(0, p - 1))

# Define Omega0
Omega0 <- diag(c(0.01, rep(1, p - 1), rep(0.01, p - 1)))
```

Question 2 Posterior approximation for classification with logistic regression

2a)

```r
# Load necessary libraries
library(MASS)
library(mvtnorm)
set.seed(12345)

# Load the data
data <- read.table("WomenAtWork.dat", header = TRUE)

# Define the logistic function
logistic <- function(x) {
  return(exp(x) / (1 + exp(x)))
}

# Define the log posterior
log_posterior <- function(beta, data, tau) {
  y <- data$Work
  X <- as.matrix(data[,-1])
  eta <- X %*% beta
  p <- logistic(eta)
  log_likelihood <- sum(y * log(p) + (1 - y) * log(1 - p))
  log_prior <- - sum(beta^2) / (2 * tau^2)
  #log_prior <- sum(dnorm(beta, mean = 0, sd = tau, log = TRUE))
  return(log_likelihood + log_prior)
}

# Optimize the log posterior
start <- rep(0, ncol(data) - 1)  # starting values
tau <- 2  # prior standard deviation
result <- optim(start, log_posterior, data = data, tau = tau, control = list(fnscale = -1), hessian = T

# Extract the posterior mode and observed information matrix
beta_hat <- result$par
obs_hessian <- -solve(result$hessian)

# Compute the 95% equal tail posterior probability interval for NSmallChild
index <- which(colnames(data) == "NSmallChild")
ci <- qnorm(c(0.025, 0.975), beta_hat[index], sqrt(obs_hessian[index, index]))

print(ci)
```

```
## [1] -0.2904429  0.3553803
```

```r
# Fit a logistic regression model
glmModel <- glm(Work ~ 0 + ., data = data, family = binomial)

# Print the coefficients
print(coef(glmModel))
```

```
##     Constant   HusbandInc    EducYears     ExpYears          Age NSmallChild
```

```
##  0.02262929 -0.03796308  0.18447411  0.12131763 -0.04858167 -1.56485140
##   NBigChild
## -0.02526059
```

```
# Compare with the posterior mode
print(beta_hat)
```

```
## [1]  0.32692525 -0.03860880  0.15652049  0.11410605 -0.04942491 -1.51956866
## [7]  0.03246872
```

Given the result of ci [-0.2904429,0.3553803], this interval represents the range in which we are expecting the value of the NSmallChild coefficient to lie with 95% probability, since it includes 0, it suggests that there is possibility that NSmallChild has no effect on the probability that a woman works. It simply means that given the data and the model, the number of small children a woman has may not be a significant factor in determining whether she works or not.

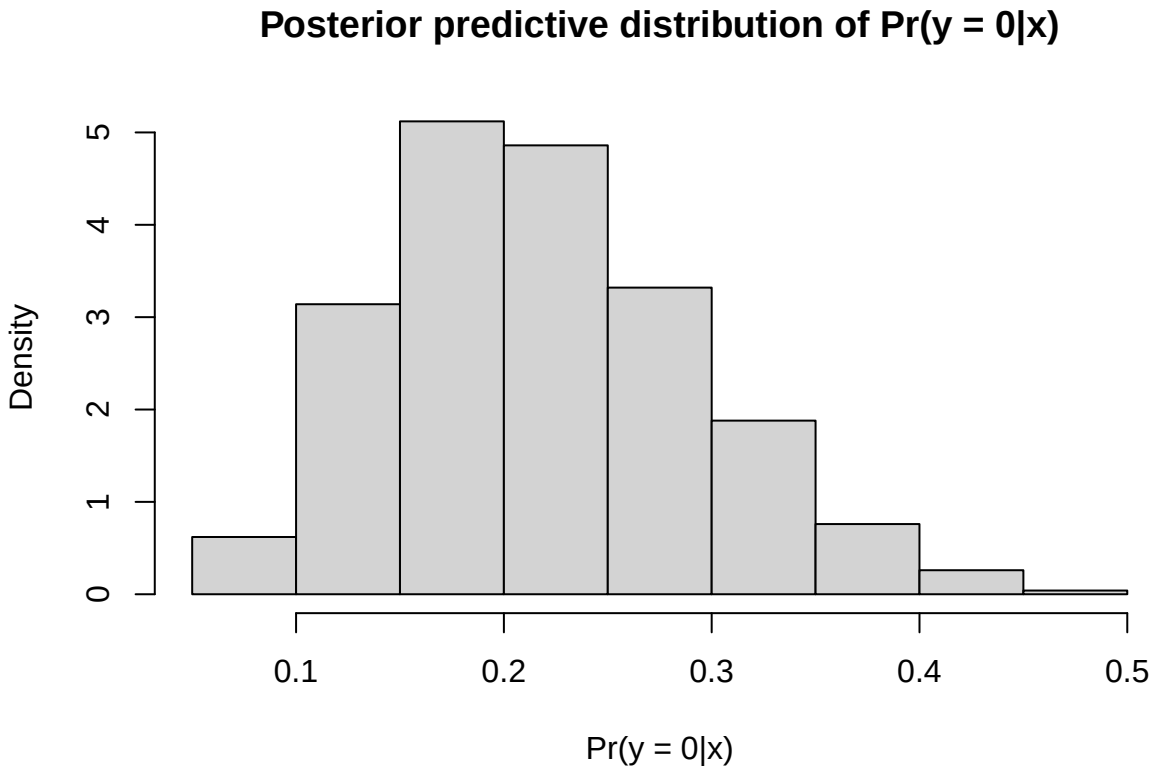
When comparing the bayesian mode to glm model, it could see that both models are quite close for all variables, which suggest that the bayesian logistic regression mode with a normal prior in this data is providing reasonable estimates. However, there are some differences.

1) the Constant variable is larger in the Bayesian model compared to the glm model. This could be due to the influence of the prior in the Bayesian mode which pulls the estimates towards zero.

2) the NBigChild coefficient is positive in the Bayesian mode while it is negative in the glm model. This could suggest that the influnce of having more children older than 6 years on the probability of a woman working is uncertain, and could be positive or negative depending on the data and mode assumptions.

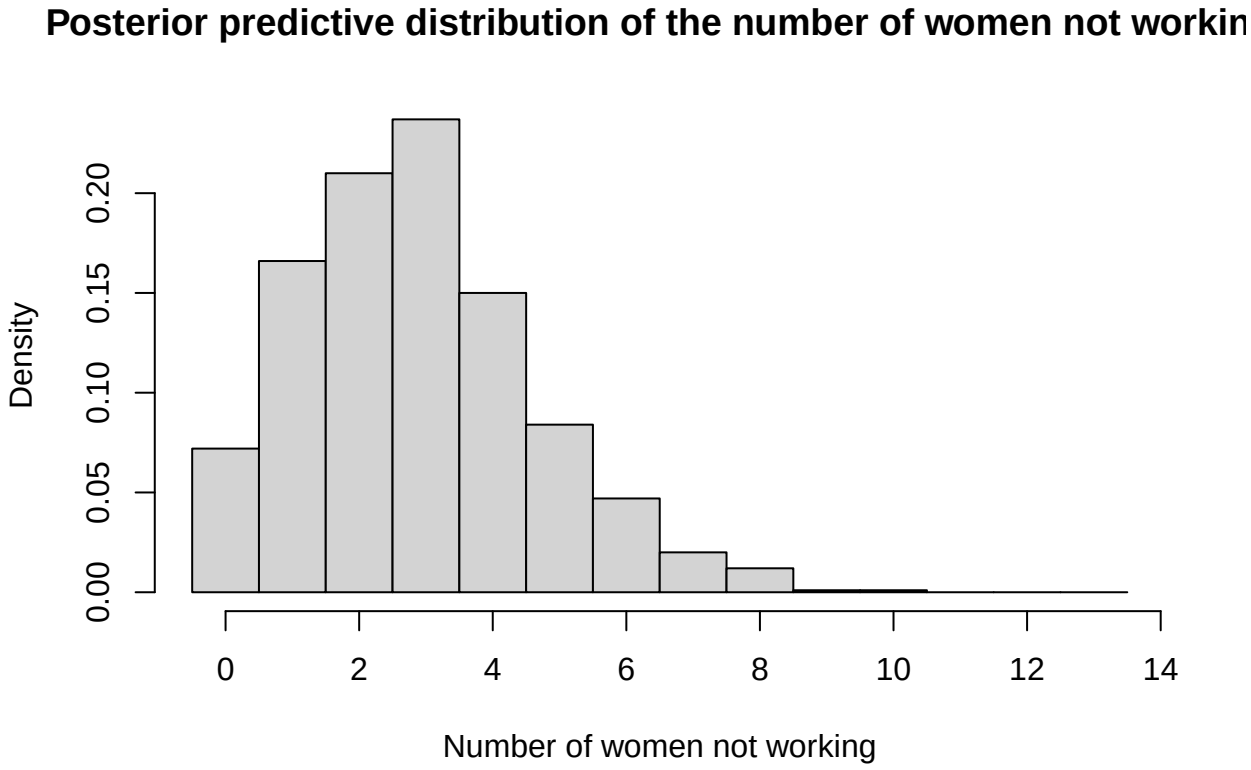
2b)

```
# Simulate from the posterior predictive distribution
simulate_posterior <- function(beta_hat, obs_hessian, x_new, n = 1000) {
  beta_draws <- rmvnorm(n, mean = beta_hat, sigma = obs_hessian)
  p_draws <- logistic(x_new %*% t(beta_draws))
  return(p_draws)
}

# Define the new woman
x_new <- c(1, 18, 11, 7, 40, 1, 1)

# Simulate from the posterior predictive distribution
p_draws <- simulate_posterior(beta_hat, obs_hessian, x_new)

# Plot the posterior predictive distribution
hist(p_draws, freq = FALSE, xlab = "Pr(y = 0|x)", main = "Posterior predictive distribution of Pr(y = 0
```



2c)

```r
# Function to simulate the number of women not working out of 13
simulate_not_working <- function(x_new, beta_hat, obs_hessian, n_women = 13, n = 1000) {
  p_draws <- simulate_posterior(beta_hat, obs_hessian, x_new, n)
  not_working_draws <- rbinom(n, size = n_women, prob = p_draws)
  return(not_working_draws)
}

# Simulate the number of women not working out of 13
not_working_draws <- simulate_not_working(x_new, beta_hat, obs_hessian)

# Plot the posterior predictive distribution
hist(not_working_draws, breaks = seq(-0.5, 13.5, 1), freq = FALSE, main = "Posterior predictive distribu
```


Posterior predictive distribution of the number of women not workin
Density
0.00
0.05
0.10
0.15
0.20
0
2
4
6
8
10
12
14
Number of women not working