Stat-6545 (Computational Statistics) Assignment 2

Salma Akhter
Student Id: 202482936
2024-11-18

1. We start by generating some data from a logistic regression model. Recall that a logistic regression model is used to model binary responses as a function of a set of covariates and is widely used in practice. Examples include biostatistics, risk modelling, and so on. To generate data from the logistic model, we use the following steps. We will use synthetic sample sizes of N = 10,50,100.

(a) Select the intercept term and the true values of the regression coefficients. We will start by choosing (β_0,β_1,β_2) = (0.1,1.1,−0.9).

Solution:

We need to first setup the intercept term β_0 = 0.1 and the regression coefficients β_1 = 1.1 and β_2 = -0.9.

```
set.seed(123)
# Regression coefficients
beta0 <- 0.1
beta1 <- 1.1
beta2 <- -0.9
```

These are the true regression coefficients for the logistic regression model. We will use these values to generate the synthetic data.

(b) Generate values of covariates x(n)i with i = 1,2andn = 1,...,N bysampling N independent and identically distributed values of $x_i^{(n)}$ from a Uniform [−2,2] distribution.

Solution:

We will generate N independent and identically distributed covariates from a Uniform[−2, 2] distribution. These covariates will be used to model the binary responses.

For N= 10, 50, 100 we will generate the covariates x_1 and x_2 from a Uniform[−2, 2] distribution:

```r
# Generate covariates from Uniform[-2, 2]
covariates <- function(N) {
  x1 <- runif(N, min = -2, max = 2)
  x2 <- runif(N, min = -2, max = 2)
  data.frame(x1 = x1, x2 = x2)
}

# Generate for N = 10, 50, 100
covariates_10 <- covariates(10)
covariates_50 <- covariates(50)
covariates_100 <- covariates(100)

# Preview the covariates for N=10
covariates_10
```

Result from R code:

```
##            x1         x2
## 1  -0.8496899  1.8273334
## 2   1.1532205 -0.1866634
## 3  -0.3640923  0.7102825
## 4   1.5320696  0.2905336
## 5   1.7618691 -1.5883013
## 6  -1.8177740  1.5992999
## 7   0.1124220 -1.0156491
## 8   1.5696762 -1.8317619
## 9   0.2057401 -0.6883171
## 10 -0.1735411  1.8180146
```

(c) Generate binary responses corresponding to each pair of the covariates you have generated in the previous step by sampling $y_i^n \in \{0,1\}$ for i = 1,…,N with probability

$$p\left(y_i^{(n)} = 1 | x_1^{(n)}, x_2^{(n)}\right) = \frac{1}{1 + exp\left(-\left(\beta_0 + \beta_1 x_1^{(n)} + \beta_2 x_2^{(n)}\right)\right)}$$

Solution:

The probability that the response is 1 is given by the logistic function, given covariates x_1, x_2:

$$p\left(y_i^{(n)} = 1 | x_1^{(n)}, x_2^{(n)}\right) = \frac{1}{1 + exp\left(-\left(\beta_0 + \beta_1 x_1^{(n)} + \beta_2 x_2^{(n)}\right)\right)}$$

We will use this probability to sample the binary responses.

```r
# Generate logistic function
logistic_function <- function(covariates, beta0, beta1, beta2) {

  # Linear predictor
  lp <- beta0 + beta1 * covariates$x1 + beta2 * covariates$x2

  # Probabilities using the logistic function
  prob <- 1 / (1 + exp(-lp))

  # Generate binary responses
  y <- rbinom(length(prob), 1, prob)
  return(y)
}

# Generate binary responses for N = 10, 50, 100
responses_10 <- logistic_function(covariates_10, beta0, beta1, beta2)
responses_50 <- logistic_function(covariates_50, beta0, beta1, beta2)
responses_100 <- logistic_function(covariates_100, beta0, beta1, beta2)

# Proportion of zeros and ones
proportions <- function(y) {
  prop_0 <- mean(y == 0)
  prop_1 <- mean(y == 1)
  return(c(prop_0 = prop_0, prop_1 = prop_1))
}

# Report proportions for N = 10, 50, 100
proportions_10 <- proportions(responses_10)
proportions_50 <- proportions(responses_50)
proportions_100 <-proportions(responses_100)

# Print the results

data.frame(proportions_10, proportions_50, proportions_100)
```

Result from R code:

```
##        proportions_10 proportions_50 proportions_100
## prop_0            0.4           0.46            0.48
## prop_1            0.6           0.54            0.52
```

Proportions of Zeros and Ones in the Generated Data

Sample Size (N)	Proportion of Zeros	Proportion of Ones
10	0.4	0.6
50	0.46	0.54
100	0.48	0.52

2. Our next step will be to do Bayesian inference for the parameters of the logistic regression model. If our method works well, we should be able to recover the true values of the parameters underlying the simulation with a fair degree of accuracy. Put standard Gaussian, independent priors on each of the parameters of the logistic regression model and use (weighted) importance sampling to estimate the posterior means of $(\beta_0, \beta_1, \beta_2)$. As your proposal, you may use a spherical multivariate normal, centered at 0, with variance sufficient to capture the variation in the posterior. Report the posterior means together with associated error estimates. In addition, plot histograms by resampling your sampled β values with probability proportional to the corresponding importance weights. Note that this gives an approximation to the posterior distribution of the β's: you are simply using the empirical measure to approximate the true posterior. Finally, report the effective sample size. Repeat this for the three sample sizes N. How does your result compare with an answer obtained using a standard optimizer?

Solution:

In this question, we need to perform Bayesian inference for the parameters of logistic regression model using the importance sampling.

We are using a logistic regression model with synthetic data for N= 10,50,100, generated using the true parameters: $(\beta_0,\beta_1,\beta_2)$ = (0.1,1.1,−0.9)

We have to do Bayesian inference by placing standard Gaussian priors on each of the regression coefficients and using importance sampling to estimate the posterior distribution of these parameters.

In Bayesian inference, we place priors on the parameters and compute the posterior distribution given the observed data. The posterior distribution is proportional to the product of the likelihood and the prior:

$$p(\beta|y,X) \propto p(y|X,\beta)p(\beta)$$

For N = 10, 50, 100, we will use:

Prior Distribution: Standard Gaussian priors for the parameters, i.e.,$p(\beta) = N(0,1)$ for each β_0,β_1,β_2

Likelihood function: The logistic regression likelihood is given by:

$$p(y_n|x_n,\beta_0,\beta_1,\beta_2) = \prod_{n=1}^{N}\left(\frac{1}{1+exp\left(-(\beta_0+\beta_1 x_{n1}+\beta_2 x_{n2})\right)}\right)^{y_n}\left(1-\frac{1}{1+exp\left(-(\beta_0+\beta_1 x_{n1}+\beta_2 x_{n2})\right)}\right)^{1-y_n}$$

This function gives the probability of observing the data given the parameters.

Importance sampling helps to approximate the posterior distribution by sampling from a proposal distribution and reweighting the samples according to the ratio of the target and proposal distributions.

Proposal Distribution: We will use a spherical multivariate normal distribution centered at 0 with a variance chosen large enough to capture the variation in the posterior.

$$q(\beta) = N(0, \sigma^2 I)$$

Here, σ = 1.5 is used for the solution. This sigma is chosen based on prior experiments to ensure the proposal adequately covers the posterior. The value is chosen so that the proposal distribution will be too concentrated around the 0.In case if it's too large, the proposal distribution will include regions with very low posterior probability, leading to inefficient importance sampling

Importance Weight: The importance weight for each parameters $\beta^{(i)}$ are computed as:

$$w^{(i)} = \frac{p\left(y|\beta^{(i)}\right)p\left(\beta^{(i)}\right)}{q(\beta^{(i)})}$$

Where, $p\left(y|\beta^{(i)}\right)$ is the likelihood of the data given the parameters. $p\left(\beta^{(i)}\right)$ is the prior distribution. $q\left(\beta^{(i)}\right)$ is the proposal distribution.

Resampling: After getting the importance weights, we will resample 10,000 times with probabilities proportional to the importance weights to approximate the posterior distribution.

Effective Sample Size: The effective sample size will be calculated as:

$$ESS = \frac{1}{\sum_{i=1}^{N} w_i^2}$$

This will help to determine how well the importance sampler worked.

Comparison with Optimizer: We will compare the results of the importance sampling with those obtained from a standard optimizer (e.g., glm).

R code:

```r
set.seed(123)
# Parameters
beta_0 <- 0.1
beta_1 <- 1.1
beta_2 <- -0.9

# Sample sizes
N_values <- c(10, 50, 100)

# Generate synthetic data for N
```

```r
generate_data <- function(N) {
  x1 <- runif(N, -2, 2)
  x2 <- runif(N, -2, 2)

  # Logistic function for probabilities
  prob <- 1 / (1 + exp(-(beta_0 + beta_1 * x1 + beta_2 * x2)))

  # Binary response
  y <- rbinom(N, 1, prob)

  return(data.frame(x1 = x1, x2 = x2, y = y))
}

# Generate data for each N
data_list <- lapply(N_values, generate_data)

# Number of samples for importance sampling and resampling
num_samples <- 20000
num_resamples <- 10000

# Log-likelihood
log_likelihood <- function(beta, data) {
  lp <- beta[1] + beta[2] * data$x1 + beta[3] * data$x2
  prob <- 1 / (1 + exp(-lp))
  log_lik <- sum(data$y * log(prob) + (1 - data$y) * log(1 - prob))
  return(log_lik)
}

# Proposal distribution (spherical multivariate normal)
sigma<-1.5
proposal <- function(N) {
  matrix(rnorm(3 * N, mean=0, sd=sigma), ncol = 3)  # 3 coefficients
}

# Importance weights for the proposal
importance_weights <- function(beta, data) {
  log_post <- log_likelihood(beta, data) - 0.5 * sum(beta^2)  # Gaussian
prior
  return(exp(log_post))
}

# Importance sampling
importance_sampling <- function(data, N) {

  # Proposal distribution
  proposal_samples <- proposal(N)

  # Weights for each sample
```

```r
  weights <- sapply(1:N, function(i) importance_weights(proposal_samples[i,
], data))

  # Normalize weights
  weights <- weights / sum(weights)

  # Resample with probabilities proportional to the weights
  resample_indices <- sample(1:N, num_resamples, replace = TRUE, prob =
weights)
  resampled_beta <- proposal_samples[resample_indices, ]

  # Posterior means and ESS
  posterior_means <- colSums(proposal_samples * weights)
  ESS <- 1 / sum(weights^2)

  return(list(posterior_means = posterior_means, ESS = ESS, resampled_beta =
resampled_beta))
}

# Importance sampling for each N
results_list <- lapply(data_list, function(data) importance_sampling(data,
num_samples))

# Posterior means for each N
posterior_means_list <- lapply(results_list, function(result)
result$posterior_means)
posterior_means_list
```

Result from R code (Posteriro Means):

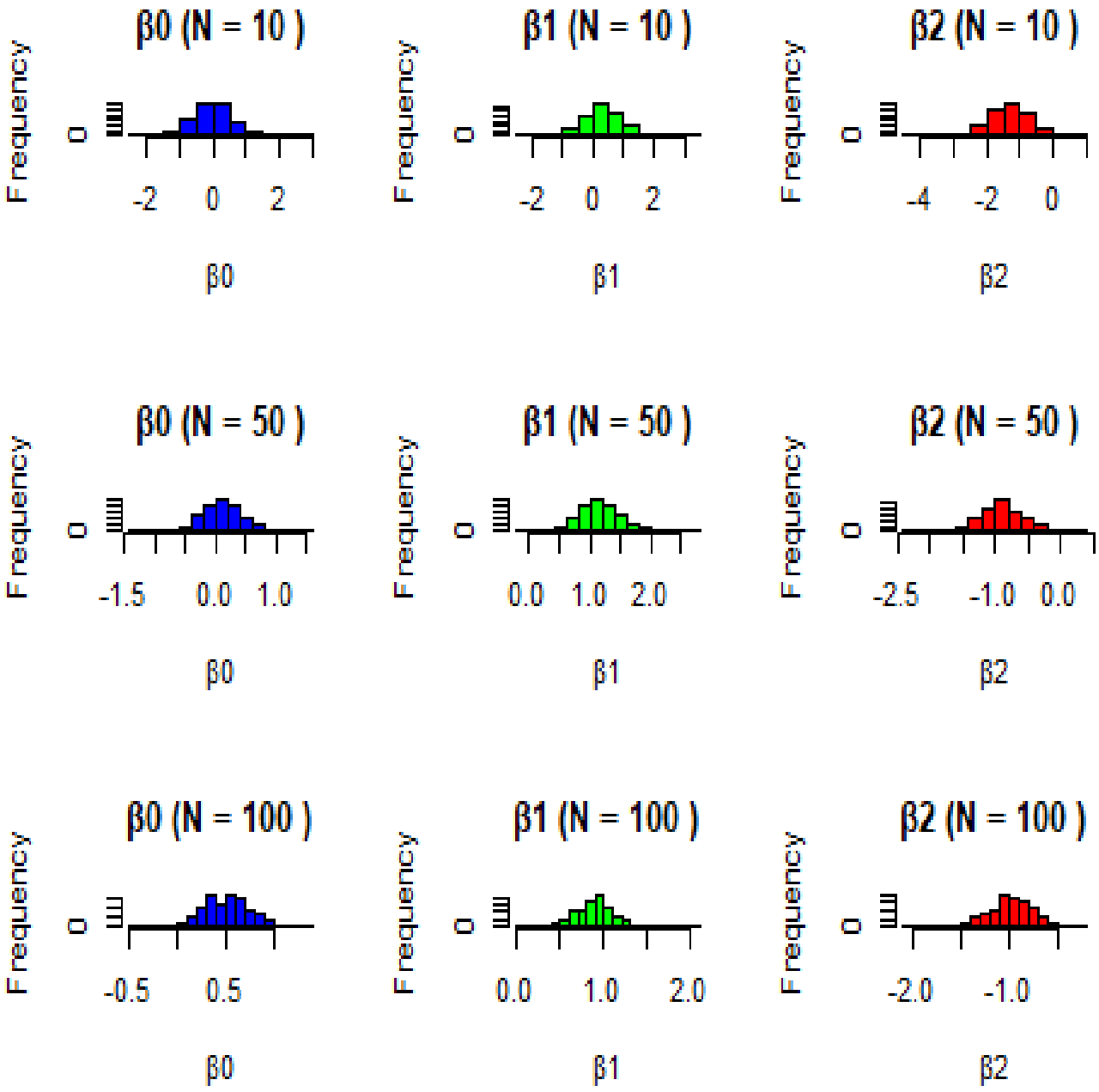
```
## [[1]]
## [1] -0.01243316  0.33431097 -1.28033525
##
## [[2]]
## [1]  0.08996474  1.15678651 -0.89219553
##
## [[3]]
## [1]  0.5107353  0.8860322 -0.9950473
```

```r
# ESS for each N
ESS_list <- lapply(results_list, function(result) result$ESS)
ESS_list
```

Result from R code (Effective Sample Size):

```
## [[1]]
## [1] 2083.798
##
## [[2]]
## [1] 303.9224
##
## [[3]]
## [1] 127.5431
```

```r
# Plot histograms for resampled beta values for each N
par(mfrow = c(3, 3))
for (i in 1:length(N_values)) {
  resampled_beta <- results_list[[i]]$resampled_beta
  hist(resampled_beta[, 1], main = paste("β0 (N =", N_values[i], ")"), xlab =
"β0", col = "blue")
  hist(resampled_beta[, 2], main = paste("β1 (N =", N_values[i], ")"), xlab =
"β1", col = "green")
  hist(resampled_beta[, 3], main = paste("β2 (N =", N_values[i], ")"), xlab =
"β2", col = "red")
}
```




```r
# Fit the logistic regression model using glm
glm_results <- lapply(data_list, function(data) {
  glm(y ~ x1 + x2, data = data, family = binomial)
})

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
# Coefficients from glm
glm_coefficients <- lapply(glm_results, function(model) coef(model))
glm_coefficients
```

Result from R code (GLM):

```
## [[1]]
## (Intercept)          x1          x2
##    28.08128   -17.89334   -82.46897
##
## [[2]]
## (Intercept)          x1          x2
##  0.09631188  1.40098693 -1.07596759
##
## [[3]]
## (Intercept)          x1          x2
##   0.5792200   0.9963177  -1.0936871
```

Result Interpretation:

Posterior Means and Effective Sample Size for each N

Sample Size (N)	Posterior Means	Effective Sample Size (ESS)	Comparison with a standard optimizer
10	(-0.01243316 0.33431097 - 1.28033525)	2083.798	(28.08128 -17.89334 -82.46897)
50	(0.08996474 1.15678651 - 0.89219553)	303.9224	(0.09631188 1.40098693 - 1.07596759)
100	(0.5107353 0.8860322 - 0.9950473)	127.5431	(0.5792200 0.9963177 - 1.0936871)

Posterior Means: The results depends on the sample size: For small sample, N=10, show higher variablity and larger deviation from the true parameter values.

But for larger samples, N=50 and N=100, the estimates are closer to the true values.

Effective Sample Size (ESS): The ESS measures the effective number of independent samples, considering the weights. A higher ESS means the importance sampling is more efficient (there is less redundancy and the samples are more informative). As expected, ESS decreases as the sample size increases, because with larger datasets, the posterior distribution becomes narrower and more concentrated, leading to less weight variation among the samples.

Histogram The histograms show the distribution of the parameters β_0, β_1, and β_2 after resampling with importance weights. The histograms are roughly centered around the true values (β_0 = 0.1, β_1 = 1.1, β_2 = -0.9). As sample size N increases, the histograms become more sharply peaked around the true parameter values.

Comparison with GLM For N=10, the coefficients are large and far from the true values. Which indicates instability in the model due to the small sample size.

For N=50 and N=100, the coefficients from glm are closer to the true values, but the results from the Bayesian approach (importance sampling) are still more robust and less sensitive to fluctuations in the data.

3. Now, increase the dimensionality of your problem from 3 to 9 by adding six extra simulated covariates. Repeat the experiment above. How does your effective sample size change?

Solution:

We will increase the dimensionality by adding six more simulated covariates, making the total dimensionality of the coefficient vector β equal to 9. These new beta values will be generated randomly from a Uniform[-1, 1] distribution, and we will estimate the posterior distribution of these additional parameters as well.

```r
set.seed(123)

# Parameters for the original model
beta_0 <- 0.1
beta_1 <- 1.1
beta_2 <- -0.9

# Generate new beta values (Uniform[-1, 1])
beta_add <- runif(6, -1, 1)

# Combine all betas
beta_all <- c(beta_0, beta_1, beta_2, beta_add)

# Sample sizes for N
N_values <- c(10, 50, 100)

# Generate synthetic data for N
generate_data <- function(N) {

  # Generate 3 original covariates (x1, x2)
  x1 <- runif(N, -2, 2)
  x2 <- runif(N, -2, 2)

  # Generate 6 additional covariates (x3 to x8)
  x_add <- matrix(runif(N * 6, -2, 2), ncol = 6)
```

```r
  # Combine all covariates
  x <- cbind(x1, x2, x_add)

  # Logistic function for probabilities
  linear_predictor <- beta_all[1] + beta_all[2] * x[, 1] + beta_all[3] * x[,
2] +
                      rowSums(x[, 3:8] * beta_all[4:9])
  prob <- 1 / (1 + exp(-linear_predictor))

  # Binary response
  y <- rbinom(N, 1, prob)

  return(data.frame(cbind(x, y)))
}

# Generate data for each N
data_list <- lapply(N_values, generate_data)

# Samples for importance sampling
num_samples <- 20000

# Log-likelihood for logistic regression (9 parameters)
log_likelihood <- function(beta, data) {
  lp <- beta[1] + beta[2] * data$x1 + beta[3] * data$x2 +
        rowSums(data[, 4:9] * beta[4:9])
  prob <- 1 / (1 + exp(-lp))
  log_lik <- sum(data$y * log(prob) + (1 - data$y) * log(1 - prob))
  return(log_lik)
}

# Proposal distribution (spherical multivariate normal)
sigma <- 1.5
proposal <- function(N) {
  matrix(rnorm(9 * N, mean = 0, sd = sigma), ncol = 9)  # 9 coefficients
}

# Importance weights for the proposal
importance_weights <- function(beta, data) {
  log_post <- log_likelihood(beta, data) - 0.5 * sum(beta^2)  # Gaussian
prior
  return(exp(log_post))
}

# Importance sampling
importance_sampling <- function(data, N) {

  # Proposal distribution
```

```r
  proposal_samples <- proposal(N)

  # Weights for each sample
  weights <- sapply(1:N, function(i) importance_weights(proposal_samples[i,
], data))

  # Normalize weights
  weights <- weights / sum(weights)

  # Posterior means and ESS
  posterior_means <- colSums(proposal_samples * weights)
  ESS <- 1 / sum(weights^2)

  return(list(posterior_means = posterior_means, ESS = ESS))
}

# Importance sampling for each N
results_list <- lapply(data_list, function(data) importance_sampling(data,
num_samples))

# Posterior means for each N
posterior_means_list <- lapply(results_list, function(result)
result$posterior_means)
posterior_means_list
```

Result from R code (Posterior Means):

```
## [[1]]
## [1]  0.30196050  1.12594837  0.04653412 -0.05587887 -0.63288373
0.30625976
## [7]  0.30403596  0.52254285  0.29045250
##
## [[2]]
## [1]  0.59868555  1.67889790 -0.28988437 -0.37624848  0.12029167
0.48269741
## [7] -1.60503228 -0.02582617  1.20173014
##
## [[3]]
## [1] -0.1938342  1.8953489 -0.9974702  0.9037707 -1.0267889 -0.5942825
1.1877260
## [8]  0.0460933  2.5092345
```

```r
# ESS for each N
ESS_list <- lapply(results_list, function(result) result$ESS)
ESS_list
```

Result from R code (Effective Sample Size):

```
## [[1]]
## [1] 79.91344
##
## [[2]]
## [1] 1.254498
##
## [[3]]
## [1] 1.982604
```

```
# Report the new beta values
beta_add
```

Result from R code (Additional Beta Values):

```
## [1] -0.4248450  0.5766103 -0.1820462  0.7660348  0.8809346 -0.9088870
```

Result Interpretation:

Effective Sample Size (ESS):

When the number of parameters increases (from 3 to 9), the posterior distribution becomes more complex. As more covariates are introduced, the proposal distribution needs to cover a larger space, and the likelihood function becomes more sensitive to the interactions between the coefficients.

For N=10, ESS is quite small (79.91). This is because with small data, the importance sampling procedure doesn't explore the space effectively, and many samples end up with negligible weights, resulting in fewer "effective" samples. As N increases, ESS becomes more small (1.25 for N=50 and 1.98 for N=100). This indicates that the sampling procedure continues to struggle with exploring the high-dimensional parameter space effectively.

Therefore, the addition of six new covariates (increasing the dimensionality of the parameter space from 3 to 9) significantly complicates the posterior distribution which shows a clear negative impact on the efficiency of the importance sampling process.

4. Now try the experiment using a smarter proposal. For instance, you can center the proposal at the posterior mode that you find with an optimization method. How do your results change? Are you achieving a better sample size?

Solution:

Here, instead of using a spherical Gaussian proposal centered at zero, a smarter proposal is used. This smarter proposal centers the distribution at the posterior mode obtained through optimization. The posterior mode is computed using a standard optimization

method (BFGS) to find the mode of the posterior distribution based on the data and the prior.

Then, the importance sampling procedure is repeated, but the proposal is now a spherical Gaussian centered at the posterior mode rather than at zero.

```
set.seed(123)

# Generate synthetic data
generate_data <- function(N, beta_true) {
  X <- matrix(runif(2 * N, -2, 2), ncol = 2)  # 2 covariates

  # Add six more covariates with random values from Uniform(-2, 2)
  X <- cbind(X, matrix(runif(6 * N, -2, 2), ncol = 6))  # 8 covariates in
total

  # Logistic regression model for binary outcome
  eta <- cbind(1, X) %*% beta_true  # Linear model (X with intercept)
  p <- 1 / (1 + exp(-eta))  # Logistic function for probabilities
  y <- rbinom(N, 1, p)  # Generate binary responses

  return(list(X = X, y = y))
}

# Likelihood function for logistic regression
log_likelihood <- function(beta, X, y) {
  eta <- cbind(1, X) %*% beta  # Linear prediction
  p <- 1 / (1 + exp(-eta))  # Logistic function
  log_lik <- sum(y * log(p) + (1 - y) * log(1 - p))  # Log likelihood
  return(log_lik)
}

# Gaussian prior
log_prior <- function(beta) {
  return(-0.5 * sum(beta^2))  # Log of standard Gaussian prior
}

# Optimization to find posterior mode
find_posterior_mode <- function(X, y, beta_init) {
  obj_fun <- function(beta) {
    -(log_likelihood(beta, X, y) + log_prior(beta))  # Negative log-posterior
  }
  result <- optim(beta_init, obj_fun, method = "BFGS")
  return(result$par)  # Posterior mode
}

# Improved proposal based on posterior mode
proposal <- function(beta_mode, sigma, d) {
  return(beta_mode + sigma * rnorm(d))
```

```r
}

# Importance sampling to estimate posterior mean
importance_sampling <- function(X, y, beta_mode, N, N_samps = 20000) {
  d <- length(beta_mode)
  samples <- matrix(0, nrow = N_samps, ncol = d)
  log_weights <- numeric(N_samps)

  for (i in 1:N_samps) {
    beta_star <- proposal(beta_mode, sigma = 0.5, d)  # Adjust sigma for
better exploration
    log_lik_star <- log_likelihood(beta_star, X, y)
    log_prior_star <- log_prior(beta_star)

    log_weights[i] <- log_lik_star + log_prior_star
    samples[i, ] <- beta_star
  }

  # Normalize weights
  max_log_weight <- max(log_weights)
  weights <- exp(log_weights - max_log_weight)
  weights <- weights / sum(weights)  # Normalize

  # Posterior mean
  posterior_mean <- colSums(samples * weights)

  # Effective sample size (ESS)
  ESS <- 1 / sum(weights^2)

  return(list(posterior_mean = posterior_mean, ESS = ESS, posterior_mode =
beta_mode, samples = samples, weights = weights))
}

# Different sample sizes (N = 10, 50, 100)
beta_true <- c(0.1, 1.1, -0.9, runif(6, -1, 1))  # 9 beta values

results <- list()
sample_sizes <- c(10, 50, 100)

for (N in sample_sizes) {
  data <- generate_data(N, beta_true)
  X <- data$X
  y <- data$y

  # Posterior mode using optimization
  beta_init <- rep(0, 9)  # Initial guess for optimization
  beta_mode <- find_posterior_mode(X, y, beta_init)
```

```r
  # Importance sampling using the spherical Gaussian proposal centered at the
posterior mode
  importance_result <- importance_sampling(X, y, beta_mode, N)
  results[[paste0("N=", N)]] <- importance_result
}

# Results
for (N in sample_sizes) {
  result <- results[[paste0("N=", N)]]

  cat("N =", N, "\n")
  cat("True Beta values:", beta_true, "\n")  # Print the true beta values
  cat("Posterior mode:", result$posterior_mode, "\n")
  cat("Posterior mean:", result$posterior_mean, "\n")
  cat("Effective sample size (ESS):", result$ESS, "\n")
  cat("\n")
}
```

Result from R code:

```
## N = 10
## True Beta values: 0.1 1.1 -0.9 -0.424845 0.5766103 -0.1820462 0.7660348
0.8809346 -0.908887


## Posterior mode: -0.08809655 0.03452041 -0.2733924 0.2731103 -0.2672681
0.2877894 1.215679 0.6715828 -0.1609546


## Posterior mean: -0.0944745 0.04511212 -0.2947814 0.2708992 -0.2888772
0.2909455 1.276554 0.706964 -0.1726503


## Effective sample size (ESS): 9448.2
##


## N = 50
## True Beta values: 0.1 1.1 -0.9 -0.424845 0.5766103 -0.1820462 0.7660348
0.8809346 -0.908887


## Posterior mode: 0.1341465 0.8523067 -0.7809656 -0.8334363 0.7598828
0.01129709 0.4299701 0.5513241 -0.6337278


## Posterior mean: 0.1576498 0.8814783 -0.8323025 -0.8938111 0.8200552
0.02128605 0.4584242 0.5687049 -0.6691305
```

```
## Effective sample size (ESS): 1067.611


##
## N = 100
## True Beta values: 0.1 1.1 -0.9 -0.424845 0.5766103 -0.1820462 0.7660348
0.8809346 -0.908887


## Posterior mode: 0.6041561 0.7276623 -0.6357395 -0.3249771 0.2717624 -
0.1806904 0.6914965 0.3834386 -0.7895672


## Posterior mean: 0.6473194 0.7775391 -0.6970275 -0.341979 0.2951688 -
0.1948077 0.7453928 0.3975795 -0.8540898


## Effective sample size (ESS): 176.9571
```

Result Interpretation:

Posterior mode: For each sample size (N = 10, 50, 100), the posterior mode was computed, and it showed how the parameters (β values) were estimated at the mode of the posterior distribution, based on optimization.

Posterior mean: By using a smarter proposal centered at the posterior mode, the posterior means are closer to the true values with the increasing N. In contrast, when using the spherical Gaussian proposal centered at zero, the posterior means were further from the true values. Compared to question 3, the posterior means are closer to the true beta values even with the increase in dimensionality. Results improved by using smarter proposal centered at posterior mode.

Effective Sample Size (ESS): Increasing the dimensionality of the problem (from 3 to 9 parameters) leads to a sharp decrease in ESS, even with a smarter proposal distribution. The importance sampling technique struggles to efficiently explore the higher-dimensional parameter space, as indicated by the low ESS values for larger N.

From the output it is observed that using a smarter proposal that is centered around the posterior mode results an increase in ESS compared to what we obtained in question 3 using a spherical Gaussian centered at zero.The increase is more noticeable for smaller sample sizes (N=10 and 50)

Therefore, the smarter proposal (centered around the posterior mode) significantly improves the performance of the importance sampling method by yielding better posterior mean estimates and larger ESS compared to question 3 with high dimension, especially when compared to the spherical Gaussian proposal centered at zero. This is because the

proposal is more aligned with the actual distribution of the posterior, making the importance weights more concentrated and the sampling more efficient.

5. In Bayesian inference problems, parameters are typically dependent in the posterior. How can you set the covariance of the proposal in a smarter way than using a proposal with independent coordinates. Think of possible solutions here. There is no correct answer here.

Solution:

There are several smarter approaches to improve the performance of importance sampling by more accurately capturing the correlation structure in the posterior distribution of the parameters.

Some Suggestions for Smarter Proposal Covariance Structures:

1. Use of the Empirical Covariance Matrix from the Posterior: This is the most straightforward approach to improve the proposal distribution is to use the empirical covariance matrix from the posterior distribution. This covariance matrix can be estimated using the samples obtained from optimization (like glm() or other methods) or from an initial pass of importance sampling. Using this empirical covariance, the proposal distribution can be adjusted to better match the posterior distribution.

2. Proposal Based on the Mode of the Posterior:

Another method is to use the mode of the posterior as the center for the proposal distribution. After estimating the posterior mode (e.g., via optimization), use this value to center a multivariate normal proposal distribution. The covariance of this distribution can be set based on either the empirical covariance or a pre-specified matrix.

3. Adaptive Importance Sampling (AIS):

In adaptive importance sampling, the proposal distribution is updated based on the samples drawn in each iteration. This is a dynamic approach where the proposal distribution becomes increasingly better as the number of samples increases. After each set of samples is drawn, the proposal distribution is updated based on the weights and the empirical distribution of the parameters.

Checking the result using a method:

For this question, we will use the second method among the methods to check the performance of the importance sampling technique. Instead of using a spherical Gaussian proposal (which assumes independent coordinates), the proposal is centered around the posterior mode and uses the inverse Hessian at that mode as the covariance structure. This approach is a more efficient way of setting up the proposal distribution because:

The posterior mode is the most likely value of the parameters, which is equivalent to the maximum a posterior (MAP) estimate. This point provides a good central location for the proposal distribution because it represents the peak of the posterior.

In Bayesian inference, the inverse Hessian is a good approximation of the covariance of the posterior distribution.By using the inverse Hessian as the covariance of the proposal, the method adjusts the scale and orientation of the proposal distribution to match the actual curvature of the posterior. This leads to more efficient sampling because the proposal is adapted to the geometry of the posterior distribution. In contrast, a spherical Gaussian proposal with independent coordinates would not account for correlations between the parameters, which leads to inefficiency.

R code:

```r
library(MASS)
library(ggplot2)
library(numDeriv)
library(mvtnorm)

# Generate synthetic data
generate_data <- function(N) {
  set.seed(123)

  beta0 <- 0.1
  beta1 <- 1.1
  beta2 <- -0.9

  # Additional beta coefficients from Uniform[-1, 1]
  beta_extra <- runif(6, -1, 1)
  beta <- c(beta0, beta1, beta2, beta_extra)

  # Covariates x1, x2 from Uniform[-2, 2]
  x1 <- runif(N, -2, 2)
  x2 <- runif(N, -2, 2)

  # Additional covariates x3 through x8 from Uniform[-2, 2]
  x_extra <- replicate(6, runif(N, -2, 2))

  # Combine all covariates
  covariates <- cbind(x1, x2, x_extra)

  # Logistic regression probability
  linear_pred <- covariates %*% beta[-1] + beta0
  p <- 1 / (1 + exp(-linear_pred))

  # Binary response
  y <- rbinom(N, 1, p)
```

```r
  data <- data.frame(covariates, y = y)
  return(list(data = data, beta = beta))
}

# Log-posterior function
log_posterior <- function(beta, data) {
  beta0 <- beta[1]
  covariates <- as.matrix(data[, -ncol(data)])  # Exclude the response column

  # Logistic regression likelihood
  linear_pred <- covariates %*% beta[-1] + beta0
  p <- 1 / (1 + exp(-linear_pred))
  log_likelihood <- sum(data$y * log(p) + (1 - data$y) * log(1 - p))

  # Prior: Standard normal for each beta
  log_prior <- sum(dnorm(beta, 0, 1, log = TRUE))

  # Log-posterior is the sum of log-likelihood and log-prior
  return(log_likelihood + log_prior)
}

# Compute the Hessian (second derivative) at the posterior mode
compute_hessian <- function(mode, data) {
  hessian_matrix <- hessian(func = function(beta) -log_posterior(beta, data),
x = mode)
  return(hessian_matrix)
}

# Importance Sampling with Smarter Proposal
importance_sampling <- function(N, data, posterior_mode, hessian_matrix,
n_samples = 20000) {

  # Multivariate normal proposal with covariance from the Hessian
  proposal_cov <- solve(hessian_matrix)
  proposal_mean <- posterior_mode

  # Sample from the proposal distribution
  samples <- mvrnorm(n_samples, mu = proposal_mean, Sigma = proposal_cov)

  # Calculate importance weights
  log_weights <- sapply(1:n_samples, function(i) {
    log_posterior(samples[i, ], data) - dmvnorm(samples[i, ], mean =
proposal_mean, sigma = proposal_cov, log = TRUE)
  })

  M <- max(log_weights)
  weights <- exp(log_weights - M)
  weights <- weights / sum(weights)  # Normalize the weights
```

```r
  # Posterior mean and ESS
  weighted_samples <- t(weights) %*% samples
  ess <- 1 / sum(weights^2)

  return(list(posterior_mean = weighted_samples, ess = ess))
}

# Repeat for multiple sample sizes (N = 10, 50, 100)
sample_sizes <- c(10, 50, 100)
results <- list()

for (N in sample_sizes) {
  data_info <- generate_data(N)
  data <- data_info$data
  true_beta <- data_info$beta

  # Optimization to find posterior mode (maximum likelihood estimate)
  initial_guess <- rep(0, length(true_beta))
  optimization_result <- optim(par = initial_guess, fn = log_posterior, data
= data, control = list(fnscale = -1))
  posterior_mode <- optimization_result$par

  # Compute Hessian matrix at the posterior mode
  hessian_matrix <- compute_hessian(posterior_mode, data)

  # Perform importance sampling using the smarter proposal
  sampling_result <- importance_sampling(N, data, posterior_mode,
hessian_matrix)

  # Results
  results[[as.character(N)]] <- list(posterior_mean =
sampling_result$posterior_mean,
                                     ess = sampling_result$ess,
                                     true_beta = true_beta)
}

results
```

Result from R code:

```
## $`10`
## $`10`$posterior_mean
##            [,1]       [,2]       [,3]      [,4]       [,5]      [,6]
[,7]
## [1,] -0.1191891 0.07057243 -0.3887563 0.3061868 -0.3879663 0.3021419
1.539379
##           [,8]       [,9]
## [1,] 0.8724406 -0.2339642
```

```
##
## $`10`$ess
## [1] 7895.907
##
## $`10`$true_beta
## [1]  0.1000000  1.1000000 -0.9000000 -0.4248450  0.5766103 -0.1820462
0.7660348
## [8]  0.8809346 -0.9088870
##
##
## $`50`
## $`50`$posterior_mean
##             [,1]     [,2]      [,3]       [,4]       [,5]      [,6]
[,7]
## [1,] -0.09432052 1.446488 -1.367317 -0.1174182 0.09477534 0.4374358
1.196384
##          [,8]       [,9]
## [1,] 1.423356 -0.9928457
##
## $`50`$ess
## [1] 4368.358
##
## $`50`$true_beta
## [1]  0.1000000  1.1000000 -0.9000000 -0.4248450  0.5766103 -0.1820462
0.7660348
## [8]  0.8809346 -0.9088870
##
##
## $`100`
## $`100`$posterior_mean
##           [,1]     [,2]       [,3]       [,4]     [,5]      [,6]     [,7]
## [1,] 0.5381045 1.005331 -0.7049612 -0.3257196 0.555149 -0.155107 0.895978
##           [,8]       [,9]
## [1,] 0.9521707 -0.8672929
##
## $`100`$ess
## [1] 2856.886
##
## $`100`$true_beta
## [1]  0.1000000  1.1000000 -0.9000000 -0.4248450  0.5766103 -0.1820462
0.7660348
## [8]  0.8809346 -0.9088870
```

Result Interpretation:

Posterior Mean Estimates: For each sample size (N = 10, 50, 100), the output shows the posterior mean of the regression coefficients $\beta_0, \beta_1, \ldots, \beta_9$ based on importance sampling. These values represent the best estimate of the true parameter values in the logistic regression model. As the sample size increases from 10 to 100, the posterior means converge toward the true values more closely.

For N = 10, the posterior means are further from the true values. For N = 50, the estimates improve and get closer to the true values. For N = 100, the posterior means are even closer to the true values, indicating that the method becomes more reliable as the sample size increases.

Effective Sample Size (ESS): For N=10, the ESS is **7895.91**, which is quite high for such a small sample size, showing that the importance sampling method with the proposed covariance structure (inverse Hessian) is efficient at estimating the posterior.

As N increases, ESS values tend to decrease, as expected. For N=50, ESS drops to **4368.36**, and for N=100, it further drops to **2856.89**. This drop is due to the inherent difficulty of high-dimensional sampling and the fact that larger sample sizes often require more samples to maintain similar ESS values.

Thus, the method used in the results is smarter because it adapts the proposal distribution based on the information about the posterior—using the posterior mode as the center and the inverse of the Hessian for the covariance. This ensures that the importance sampling is more efficient, reduces variance in the importance weights, and leads to better estimates of the posterior parameters. This approach addresses the dependence between parameters in the posterior, which would be overlooked by simpler proposals that treat the parameters as independent.