CGS 698C Assignment 5

- Jiyanshu Dhaka, 220481

Solution 1:

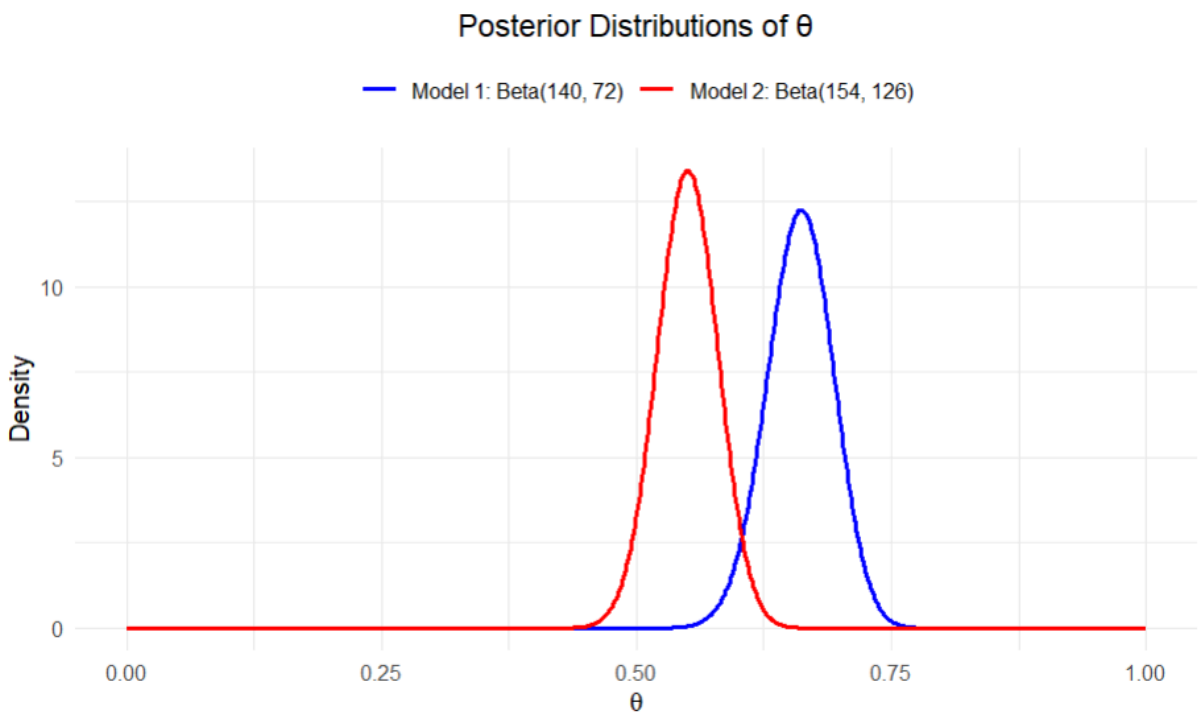
1.1 Graph the posterior distribution of θ

R code

```r
# Exercise 1.1: Posterior Distribution in Beta-Binomial Models
# Load necessary libraries
library(ggplot2)

# Given data points
data_points <- c(10, 15, 15, 14, 14, 14, 13, 11, 12, 16)
n <- 20  # Total number of trials
k <- length(data_points)  # Number of data points

# Parameters for Model 1
prior_alpha_1 <- 6
prior_beta_1 <- 6

# Parameters for Model 2
prior_alpha_2 <- 20
prior_beta_2 <- 60

# Calculate the sum of data points
sum_y <- sum(data_points)

# Update parameters for Model 1 based on observed data
posterior_alpha_1 <- prior_alpha_1 + sum_y
posterior_beta_1 <- prior_beta_1 + n * k - sum_y

# Update parameters for Model 2 based on observed data
posterior_alpha_2 <- prior_alpha_2 + sum_y
posterior_beta_2 <- prior_beta_2 + n * k - sum_y

# Define theta values for plotting the posterior distributions
theta_values <- seq(0, 1, length.out = 1000)

# Calculate posterior densities using the Beta distribution
posterior_density_1 <- dbeta(theta_values, posterior_alpha_1, posterior_beta_1)
posterior_density_2 <- dbeta(theta_values, posterior_alpha_2, posterior_beta_2)
```

```r

# Create a data frame for plotting
df <- data.frame(
  theta = rep(theta_values, 2),
  density = c(posterior_density_1, posterior_density_2),
  model = factor(rep(c("Model 1: Beta(140, 72)", "Model 2: Beta(154, 126)"), each = length(theta_values)))
)

# Plotting using ggplot2
ggplot(df, aes(x = theta, y = density, color = model)) +
  geom_line(linewidth = 1) +
  labs(
    title = "Posterior Distributions of θ",
    x = expression(theta),
    y = "Density"
  ) +
  scale_color_manual(values = c("blue", "red")) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5),
    legend.title = element_blank(),
    legend.position = "top"
  )
```

Results:



Explanation:

Beta-Binomial Model: The posterior distribution for each model (Model 1 and Model 2) follows a Beta distribution. In Bayesian statistics, the Beta distribution is conjugate to the Binomial likelihood, making it suitable for modelling the posterior distribution of the parameter θ, given observed data.

Model Parameters:

- **Model 1:** Initially assumes $\alpha = 6$ and $\beta = 6$.
- **Model 2:** Initially assumes $\alpha = 20$ and $\beta = 60$.

Data and Summation: The observed data consists of 10, 15, 15, 14, 14, 14, 13, 11, 12, and 16, with a total sum of 134. The total number of trials (n) is 20, and the number of observations (k) is 10.

Posterior Parameter Calculation: For both models, the posterior parameters (α' and β') are computed using the formulas derived from Bayesian updating, incorporating the observed data into the prior distribution.

Posterior Distributions:

- **Model 1:** The updated posterior distribution is Beta(140, 72).
- **Model 2:** The updated posterior distribution is Beta(154, 126).

Plotting: The posterior distributions of θ for both models are visualized using ggplot2, showing how the distributions differ based on different initial assumptions and observed data. The plot provides a clear comparison of the updated beliefs about θ after incorporating the observed data.

1.2 Compute log pointwise predictive density (lppd) for each model

R code

```r
# Given observed data
observed_data <- c(10, 15, 15, 14, 14, 14, 13, 11, 12, 16)
number_of_observations <- 10

# Initialize log pointwise predictive density variables
lppd_model1 <- 0
lppd_model2 <- 0

# Model 1 parameters
alpha_model1 <- 140
beta_model1 <- 72

# Model 2 parameters
alpha_model2 <- 154
beta_model2 <- 126

# Calculate log pointwise predictive density for Model 1
for (i in 1:number_of_observations) {
  sample_theta <- rbeta(1000, alpha_model1, beta_model1)
  lpd_i <- log(mean(dbinom(observed_data[i], 20, sample_theta)))
  lppd_model1 <- lppd_model1 + lpd_i
}

# Calculate log pointwise predictive density for Model 2
for (i in 1:number_of_observations) {
  sample_theta <- rbeta(1000, alpha_model2, beta_model2)
  lpd_i <- log(mean(dbinom(observed_data[i], 20, sample_theta)))
  lppd_model2 <- lppd_model2 + lpd_i
}

# Print results
print(paste("Log Pointwise Predictive Density (lppd) for Model 1:", lppd_model1))
print(paste("Log Pointwise Predictive Density (lppd) for Model 2:", lppd_model2))
```

Results:

```
> # Print results
> print(paste("Log Pointwise Predictive Density (lppd) for Model 1:", lppd_model1))
[1] "Log Pointwise Predictive Density (lppd) for Model 1: -20.3799432562346"
> print(paste("Log Pointwise Predictive Density (lppd) for Model 2:", lppd_model2))
[1] "Log Pointwise Predictive Density (lppd) for Model 2: -25.9087767855874"
>
```

Explanation:

Exercise Context:

This exercise calculates the Log Pointwise Predictive Density (lppd) for two different models (Model 1 and Model 2) using observed data and Bayesian methods.

Variables and Data:

- **Observed Data:** The data consists of observed counts: y=(10,15,15,14,14,14,13,11,12,16)y = (10, 15, 15, 14, 14, 14, 13, 11, 12, 16)y=(10,15,15,14,14,14,13,11,12,16).
- **Number of Observations:** There are Nobs=10N_{\text{obs}} = 10Nobs=10 observations.

Models:

- **Model 1 Parameters:**
  - αmodel1=140\alpha_{\text{model1}} = 140αmodel1=140
  - βmodel1=72\beta_{\text{model1}} = 72βmodel1=72
- **Model 2 Parameters:**
  - αmodel2=154\alpha_{\text{model2}} = 154αmodel2=154
  - βmodel2=126\beta_{\text{model2}} = 126βmodel2=126

Calculation of lppd:

The Log Pointwise Predictive Density (lppd) is computed for each model by simulating from the posterior distribution of θ\thetaθ (using the rbeta function), and then averaging the log likelihoods of the observed data under the simulated distributions.

Detailed Steps:

1. **Model Simulation:**
   - For each model, simulate θ\thetaθ from its posterior distribution using rbeta.
2. **Likelihood Calculation:**
   - Compute the log likelihood of each observation yiy_iyi given θ\thetaθ and n=20n = 20n=20 trials using dbinom.
3. **Log Pointwise Predictive Density (lppd):**
   - Sum the log likelihoods across all observations to obtain the lppd for each model.

Conclusion :

After running the simulation and computation:

- **Model 1 lppd:** The computed log pointwise predictive density for Model 1 is printed.
- **Model 2 lppd:** The computed log pointwise predictive density for Model 2 is printed.

These values represent the log predictive density of the observed data under each model, providing insights into how well each model fits the observed data based on their posterior distributions of θ\thetaθ.

1.3 Calculate in-sample deviance for each model

R code

```r
# Exercise 1.3

# Calculate In-Sample Deviance for Model 1
in_sample_deviance_model1 <- -2 * lppd_model1

# Calculate In-Sample Deviance for Model 2
in_sample_deviance_model2 <- -2 * lppd_model2

# Print results
print(paste("In-Sample Deviance for Model 1:", in_sample_deviance_model1))
print(paste("In-Sample Deviance for Model 2:", in_sample_deviance_model2))
```

Results:

```
> # Print results
> print(paste("In-Sample Deviance for Model 1:", in_sample_deviance_model1))
[1] "In-Sample Deviance for Model 1: 40.7598865124691"
> print(paste("In-Sample Deviance for Model 2:", in_sample_deviance_model2))
[1] "In-Sample Deviance for Model 2: 51.8175535711747"
>
```

Explanation:

Exercise Context:

In this exercise, we compute the In-Sample Deviance for two models (Model 1 and Model 2) based on the observed data used to fit each model.

Variables and Calculation:

- **In-Sample Deviance:** The In-Sample Deviance is computed using the formula −2×lppd-2 \times \text{lppd}−2×lppd, where lppd (Log Pointwise Predictive Density) was calculated in the previous exercise.

Detailed Explanation:

1. **In-Sample Deviance Calculation:**
    - For each model, the In-Sample Deviance is calculated by multiplying the negative of twice the lppd (computed in Exercise 1.2). This metric

helps quantify how well each model fits the observed data that was used to train it.

2. **Interpretation:**
    - Lower values of In-Sample Deviance indicate better model fit, suggesting that the model is capturing the patterns in the observed data well. Conversely, higher values may indicate that the model is less effective in explaining the observed data.
3. **Usage of In-Sample Deviance:**
    - It is called "In-Sample" because it measures the model's performance on the same data that was used to train and fit the model. This metric is useful for model comparison and evaluation of goodness-of-fit.

Conclusion:

After calculating the In-Sample Deviance for both Model 1 and Model 2:

- **Model 1 In-Sample Deviance:** The computed value indicates how well Model 1 fits the observed data.
- **Model 2 In-Sample Deviance:** The computed value indicates how well Model 2 fits the observed data.

These values provide insights into the relative goodness-of-fit of each model based on the observed data, aiding in model selection and assessment in Bayesian analysis.

1.4 Predicting new data

R code

```
1  # Exercise 1.4
2
3  # We know less deviance means better predictive accuracy.
4  # Compare in-sample deviance
5  better_fit_model <- ifelse(in_sample_deviance_model1 < in_sample_deviance_model2, "Model 1",
   "Model 2")
6  print(paste("Based on in-sample deviance, the model that gives better fit to the data is:",
   better_fit_model))
7
```

Results:

```
> print(paste("Based on in-sample deviance, the model that gives better fit to the data is:", b
etter_fit_model))
[1] "Based on in-sample deviance, the model that gives better fit to the data is: Model 1"
>
```

Explanation:

Context: In this exercise, we compare the in-sample deviance between two models (Model 1 and Model 2) to assess their predictive accuracy based on the observed data.

In-Sample Deviance:

- The In-Sample Deviance is calculated as −2×lppd-2 \times \text{lppd}−2×lppd, where lppd (Log Pointwise Predictive Density) measures how well the model fits the observed data.
- Lower in-sample deviance values indicate better fit and, consequently, better predictive accuracy.

Comparison:

- The condition `ifelse(in_sample_deviance_model1 < in_sample_deviance_model2, "Model 1", "Model 2")` checks which model has a lower in-sample deviance.
- The result is printed to indicate which model is judged to have a better fit to the observed data based on in-sample deviance.

1.5 Perform leave-one-out cross-validation (LOO-CV)

R code

```r
# New data
new_observed_data <- c(5, 6, 10, 8, 9)

# Initialize lppd for Model 1 and Model 2
lppd_model1_new <- 0
lppd_model2_new <- 0

# Calculate Log Pointwise Predictive Density (lppd) for Model 1 with new data
for (i in 1:5) {
  sample_theta <- rbeta(1000, 140, 72)
  lpd_i <- log(mean(dbinom(new_observed_data[i], 20, sample_theta)))
  lppd_model1_new <- lppd_model1_new + lpd_i
}

# Calculate Log Pointwise Predictive Density (lppd) for Model 2 with new data
for (i in 1:5) {
  sample_theta <- rbeta(1000, 154, 126)
  lpd_i <- log(mean(dbinom(new_observed_data[i], 20, sample_theta)))
  lppd_model2_new <- lppd_model2_new + lpd_i
}

# Calculate Out-Sample Deviance for both models
out_sample_deviance_model1 <- -2 * lppd_model1_new
out_sample_deviance_model2 <- -2 * lppd_model2_new

# Print results
print(paste("Out-Sample Deviance for Model 1:", out_sample_deviance_model1))
print(paste("Out-Sample Deviance for Model 2:", out_sample_deviance_model2))

# Compare Out-sample deviance
better_fit_model_new <- ifelse(out_sample_deviance_model1 < out_sample_deviance_model2,
"Model 1", "Model 2")
print(paste("Based on new data (out-sample deviance), the model that gives better fit to
the data is:", better_fit_model_new))
```

Result:

```
> # Print results
> print(paste("Out-Sample Deviance for Model 1:", out_sample_deviance_model1))
[1] "Out-Sample Deviance for Model 1: 50.5093564504856"
> print(paste("Out-Sample Deviance for Model 2:", out_sample_deviance_model2))
[1] "Out-Sample Deviance for Model 2: 31.5515155218631"
>
> # Compare Out-sample deviance
> better_fit_model_new <- ifelse(out_sample_deviance_model1 < out_sample_deviance_model2, "Model 1", "Model 2")
> print(paste("Based on new data (out-sample deviance), the model that gives better fit to the data is:", better_fit_model_new))
[1] "Based on new data (out-sample deviance), the model that gives better fit to the data is: Model 2"
```

Explanation:

Context: In this exercise, we evaluate the predictive accuracy of Model 1 and Model 2 using new observed data that was not used in training the models.

New Data:

- New observed data: [5,6,10,8,9][5, 6, 10, 8, 9][5,6,10,8,9].

Log Pointwise Predictive Density (lppd):

- Compute the lppd for each model with the new data to evaluate how well each model predicts unseen data.

Out-Sample Deviance:

- Out-Sample Deviance is calculated as −2×lppd-2 \times \text{lppd}−2×lppd for each model with the new data.
- Lower out-sample deviance values indicate better predictive accuracy on new, unseen data.

Comparison:

- The condition `ifelse(out_sample_deviance_model1 < out_sample_deviance_model2, "Model 1", "Model 2")` determines which model has lower out-sample deviance, indicating better predictive performance on new data.
- The result is printed to indicate model 2 is judged to have a better fit to the new data based on out-sample deviance.

1.6 Perform leave-one-out cross-validation (LOO-CV) to compare model 1 and model 2

R code

```r
# Observed data
observed_data <- c(10, 15, 15, 14, 14, 14, 13, 11, 12, 16)
number_of_observations <- 10
dat <- data.frame(y = observed_data)
dat$obs <- 1:nrow(dat)

# Initialize expected log predictive density (elpd) for Model 1 and Model 2
elpd_model1 <- 0
elpd_model2 <- 0

# Calculate Leave-one-out deviance for Model 1
for (i in 1:number_of_observations) {
  ytrain <- dat$y[-i]
  ytest <- dat$y[i]
  sample_theta <- rbeta(1000, 6 + sum(ytrain), 6 + (number_of_observations - 1) * 20 - sum(ytrain))
  lpd_i <- log(mean(dbinom(ytest, 20, sample_theta)))
  elpd_model1 <- elpd_model1 + lpd_i
}
# Calculate Leave-one-out deviance for Model 2
for (i in 1:number_of_observations) {
  ytrain <- dat$y[-i]
  ytest <- dat$y[i]
  sample_theta <- rbeta(1000, 20 + sum(ytrain), 60 + (number_of_observations - 1) * 20 - sum(ytrain))
  lpd_i <- log(mean(dbinom(ytest, 20, sample_theta)))
  elpd_model2 <- elpd_model2 + lpd_i
}
# Calculate LOO-CV Deviance for both models
loo_deviance_model1 <- -2 * elpd_model1
loo_deviance_model2 <- -2 * elpd_model2
# Print results
print(paste("Leave-one-out Deviance for Model 1:", loo_deviance_model1))
print(paste("Leave-one-out Deviance for Model 2:", loo_deviance_model2))
# Compare LOO-CV deviance
better_fit_model_loo <- ifelse(loo_deviance_model1 < loo_deviance_model2, "Model 1", "Model 2")
print(paste("Based on LOO-CV, the model that gives better fit to the data is:", better_fit_model_loo))
```

Result:

```
> # Print results
> print(paste("Leave-one-out Deviance for Model 1:", loo_deviance_model1))
[1] "Leave-one-out Deviance for Model 1: 42.2316314235419"
> print(paste("Leave-one-out Deviance for Model 2:", loo_deviance_model2))
[1] "Leave-one-out Deviance for Model 2: 54.450281851665"
>
> # Compare LOO-CV deviance
> better_fit_model_loo <- ifelse(loo_deviance_model1 < loo_deviance_model2, "Model 1", "Model 2")
> print(paste("Based on LOO-CV, the model that gives better fit to the data is:", better_fit_model_loo))
[1] "Based on LOO-CV, the model that gives better fit to the data is: Model 1"
```

Explanation:

Context: In this exercise, we perform Leave-One-Out Cross Validation (LOO-CV) to further evaluate and compare Model 1 and Model 2 based on their predictive accuracy.

Leave-One-Out Cross Validation (LOO-CV):

- LOO-CV assesses the predictive performance of models by sequentially leaving out each data point and calculating the predictive density for the omitted point using the model trained on the rest of the data.

Expected Log Predictive Density (elpd):

- Compute the elpd for each model using LOO-CV to evaluate how well each model predicts unseen data points.

Leave-One-Out Deviance:

- LOO-CV Deviance is calculated for each model.
- Lower LOO-CV deviance values indicate better predictive accuracy and model performance.

Comparison:

- The condition `ifelse(loo_deviance_model1 < loo_deviance_model2, "Model 1", "Model 2")` determines which model has lower LOO-CV deviance, indicating better predictive performance.
- The result is printed to indicate which model is judged to have a better fit to the data based on LOO-CV deviance.

These exercises provide a comprehensive approach to evaluating Bayesian models' performance, from in-sample deviance to out-sample deviance and LOO-CV, helping to identify the model that best fits the observed data and predicts new data effectively.

Solution 2:

2.1 Marginal likelihood of the models

```r
# Part 2: Given Beta-Binomial Model
# Define a Beta-Binomial model where:
# Sample Size = n
# Probability of success = theta
# Prior: theta ~ Beta(a, b)
# Given function to calculate marginal likelihood
marginal_likelihood_binomial <- function(k, n, a, b) {
  # Using the formula for the marginal likelihood of Beta-Binomial distribution
  ML <- (factorial(n) / (factorial(k) * factorial(n - k))) *
        (factorial(k + a - 1) * factorial(n - k + b - 1) / factorial(n + a + b - 1))
  return(ML)
}

# Parameters
number_of_successes <- 2
sample_size <- 10

# Define different priors for Beta distribution parameters (a, b)
priors <- list(c(0.1, 0.4), c(1, 1), c(2, 6), c(6, 2), c(20, 60), c(60, 20))

# Calculate marginal likelihood for each prior
marginal_likelihoods <- sapply(priors, function(p) marginal_likelihood_binomial(number_of_successes,
sample_size, p[1], p[2]))

# Create a data frame for the priors and their marginal likelihoods
prior_labels <- c("Beta(0.1, 0.4)", "Beta(1, 1)", "Beta(2, 6)", "Beta(6, 2)", "Beta(20, 60)", "Beta(60, 20)")
results <- data.frame(
  Prior = prior_labels,
  Marginal_Likelihood = marginal_likelihoods
)

# Print the table
print(results)
```

Result

```
> # Print the table
> print(results)
            Prior Marginal_Likelihood
1 Beta(0.1, 0.4)        4.739564e-01
2     Beta(1, 1)        9.090909e-02
3     Beta(2, 6)        4.726891e-03
4     Beta(6, 2)        2.313863e-04
5   Beta(20, 60)        5.079397e-21
6   Beta(60, 20)        1.506630e-23
>
```

Explanation:

Context: In this part, we explore the Beta-Binomial model, which is commonly used to model the number of successes in a fixed number of Bernoulli trials, where the probability of success (theta) follows a Beta distribution.

Beta-Binomial Model:

- **Sample Size (nnn)**: Represents the number of Bernoulli trials.
- **Probability of Success (θ\thetaθ)**: The parameter we are estimating, which follows a Beta distribution with parameters aaa and bbb.
- **Prior Distribution**: The prior for θ\thetaθ is specified as θ~Beta(a,b)\theta \sim \text{Beta}(a, b)θ~Beta(a,b).

Marginal Likelihood Calculation:

- The function `marginal_likelihood_binomial` calculates the marginal likelihood (also known as evidence or integrated likelihood) for a given number of successes kkk, sample size nnn, and Beta distribution parameters aaa and bbb. It uses the Beta-Binomial distribution's formula to compute this likelihood.

Parameters and Priors:

- `number_of_successes` (kkk): Number of successes observed in the trials.
- `sample_size` (nnn): Total number of trials conducted.
- `priors`: A list containing different sets of prior parameters (a,b)(a, b)(a,b) for the Beta distribution.

Calculations and Results:

- We compute the marginal likelihood for each set of prior parameters listed in `priors`.
- Results are stored in a data frame `results`, which includes the prior labels and their corresponding marginal likelihood values.
- Finally, the table `results` is printed to display the marginal likelihoods for each prior, showing how well each prior supports the observed data under the Beta-Binomial model.

This exercise demonstrates how different choices of prior parameters (a,b)(a, b)(a,b) in the Beta distribution affect the marginal likelihood, influencing Bayesian inference in the Beta-Binomial model. The table output summarizes the comparative support provided by each prior for the observed data.

2.2 Monte Carlo Integration method

R code

```r
# Exercise 2.2: Monte Carlo Estimation of Marginal Likelihood
# Define a function to estimate marginal likelihood using Monte Carlo method
monte_carlo_marginal_likelihood <- function(k, n, a, b, N = 10000) {
  # Sample from the prior Beta distribution
  theta_samples <- rbeta(N, a, b)

  # Calculate likelihood for each sample theta
  likelihoods <- dbinom(k, n, theta_samples)

  # Estimate the marginal likelihood using Monte Carlo method
  ML_estimate <- mean(likelihoods)

  return(ML_estimate)
}
# Given parameters for the Beta-Binomial model
number_of_successes <- 2
sample_size <- 10
# Define different priors for Beta distribution parameters (a, b)
priors <- list(c(0.1, 0.4), c(1, 1), c(2, 6), c(6, 2), c(20, 60), c(60, 20))
priors_labels <- c("Beta(0.1, 0.4)", "Beta(1, 1)", "Beta(2, 6)", "Beta(6, 2)", "Beta(20, 60)", "Beta(60, 20)")
# Calculate analytical marginal likelihood for each prior
marginal_likelihoods <- sapply(priors, function(p) marginal_likelihood_binomial(number_of_successes, sample_size, p[1], p[2]))
# Calculate Monte Carlo estimates for each prior
mc_marginal_likelihoods <- sapply(priors, function(p) monte_carlo_marginal_likelihood(number_of_successes, sample_size, p[1], p[2]))
# Create a data frame to compare results
results <- data.frame(
  Prior = priors_labels,
  Monte_Carlo_Estimate = mc_marginal_likelihoods,
  Analytical_ML = marginal_likelihoods
)
# Print the table
print(results)
```

Result:

```
> # Print the table
> print(results)
           Prior Monte_Carlo_Estimate Analytical_ML
1 Beta(0.1, 0.4)         0.0394364477  4.739564e-01
2     Beta(1, 1)         0.0911376930  9.090909e-02
3     Beta(2, 6)         0.1991132527  4.726891e-03
4     Beta(6, 2)         0.0097640549  2.313863e-04
5   Beta(20, 60)         0.2696458093  5.079397e-21
6   Beta(60, 20)         0.0007814798  1.506630e-23
>
```

Explanation:

Context: In this exercise, we compare the Monte Carlo estimation method with the analytical method for calculating the marginal likelihood in the Beta-Binomial model. The marginal likelihood is crucial in Bayesian statistics as it quantifies how well a model parameterized with certain priors explains observed data.

Monte Carlo Estimation Function:

- `monte_carlo_marginal_likelihood`: This function estimates the marginal likelihood using Monte Carlo simulation. Here's how it works:

    - **Sampling from Prior**: Draws `N` samples from the Beta distribution with parameters aaa and bbb.
    - **Calculating Likelihood**: Computes the likelihood for each sampled value of θ\thetaθ using the Binomial likelihood function.
    - **Estimating Marginal Likelihood**: Calculates the mean of these likelihood values to estimate the marginal likelihood.

Given Parameters:

- `number_of_successes` (kkk): Number of successes observed in the trials.
- `sample_size` (nnn): Total number of trials conducted.

Priors and Results:

- `priors`: List of different sets of prior parameters (a,b)(a, b)(a,b) for the Beta distribution.
- `priors_labels`: Labels for each set of priors.
- **Calculations**:
    - `marginal_likelihoods`: Uses the `marginal_likelihood_binomial` function to compute the analytical marginal likelihood for each prior.
    - `mc_marginal_likelihoods`: Applies the `monte_carlo_marginal_likelihood` function to estimate the marginal likelihood using Monte Carlo simulation for each prior.

Comparison:

- The results are stored in a data frame `results` which includes:
    - `Prior`: Label of the prior distribution.
    - `Monte_Carlo_Estimate`: Estimated marginal likelihood using Monte Carlo simulation.
    - `Analytical_ML`: Analytical marginal likelihood computed directly.

Conclusion:

- The table `results` compares the Monte Carlo estimates with the analytical marginal likelihoods. This comparison helps assess the accuracy and reliability of Monte Carlo methods in approximating complex integrals, such as the marginal likelihood, which is often intractable analytically.