A report for the exercise 10.2 of [DHo09]

Bayesian Methods

Achladianakis Minas

Presented for the Reading Course.

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE

Department of Applied Mathematics
University of Crete

Greece, April 2, 2025

List of Figures

Review of ch10

Nonconjugate priors and Metropolis-Hastings algorithms

Introduction

Chapter 10 of Peter D. Hoff's book covers Nonconjugate Priors and Metropolis-Hastings Algorithms. The chapter begins by discussing the context where conjugate priors are not suitable, leading to the introduction of the Metropolis-Hastings algorithm. This algorithm is presented as a general method for approximating posterior distributions.

The chapter is divided into several sections, starting with a detailed look at generalized linear models, specifically through an example involving song sparrow reproductive success. The discussion then transitions to the Metropolis algorithm, including a comprehensive explanation of the algorithm's steps, its rationale, and examples illustrating its application. The chapter continues by applying the Metropolis algorithm to Poisson regression, offering insights into the implementation challenges and strategies.

Next, the Metropolis-Hastings algorithm is introduced as an extension of the Metropolis algorithm, providing a more generalized framework for sampling from posterior distributions. This section includes technical details about proposal distributions and acceptance ratios, and it explains how the algorithm adapts to different scenarios.

The chapter concludes with a section combining the Metropolis and Gibbs algorithms, demonstrating how they can be integrated in complex models where some parameters have known conditional distributions while others do not.

Throughout the chapter, mathematical formulas, such as expressions for the acceptance ratio in the Metropolis-Hastings algorithm and the Poisson regression model, are presented to illustrate the concepts. The chapter emphasizes the practical

application of these algorithms in Bayesian statistical methods, supported by real-world examples and theoretical explanations.

10.1 Generalized Linear Models

In the first subsection, Hoff introduces the Generalized Linear Models (GLMs), expanding upon traditional linear models to accommodate various types of response distributions using the equation:

$$Y_i \sim f(\cdot|\theta_i), \quad g(\theta_i) = X_i^T \beta,$$

, where Y_i s the response variable, f s the probability distribution function, θ_i is a parameter related to the mean of Y_i, X_i is the vector of predictors, β is the vector of coefficients and $g(\dot{})$ is the link function.

10.2 The Metropolis Algorithm

This subsection delves into the Metropolis algorithm, a Markov Chain method for obtaining a sequence of random samples from a probability distribution for which direct sampling is difficult. The algorithm's steps are described as:

- Given $\theta^{(t)}$, propose $\theta' \sim q(\theta'|\theta^{(t)})$
- Compute acceptance ratio: $r = \frac{p(\theta'|y)q(\theta^{(t)}|\theta')}{p(\theta^{(t)}|y)q(\theta'|\theta^{(t)})}$
- Set $\theta^{(t+1)} = \theta'$ or $\theta^{(t)}$ with probability $\min(1, r)$ and $\max(0, 1-r)$ respectively.

where $\theta^{(t)}$ and $\theta^{'}$ are the current and proposed values of the parameter, $p(\theta|y)$ is the posterior distribution, and $q(\theta^{'}|\theta^{(t)})$ is the proposal distribution.

The rest of the chapter builds upon these foundations, discussing the specifics of implementing these models and algorithms in various contexts.

10.3 The Metropolis algorithm for Poisson regression

This section provides advanced methodologies in the Metropolis algorithm. The focus is on enhancing the efficiency of the algorithm through various optimization techniques.

- The section begins with an introduction to the importance of proposal distribution in the acceptance rate of the Metropolis algorithm.

- It then delves into the strategies for tuning the proposal distribution to maintain a balance between exploration and exploitation.

- Subsequent parts address the convergence diagnostics, emphasizing the necessity to assess the mixing and stationarity of the Markov chain.

- The subchapter concludes by presenting a case study that applies these methods to a complex statistical model, demonstrating the practical utility of the discussed techniques.

10.4 Metropolis, Metropolis-Hastings and Gibbs

Subsection 10.4 delves into the Metropolis, Metropolis-Hastings, and Gibbs sampling algorithms, highlighting their roles as foundational methods in the field of Markov Chain Monte Carlo (MCMC).

- The subsection begins by outlining the original Metropolis algorithm, emphasizing its significance in initiating the exploration of MCMC methods for sampling from complex distributions.

- It then transitions to the Metropolis-Hastings algorithm, a generalization of the Metropolis algorithm, which broadens the applicability to a wider range of proposal distributions and target densities.

- The discussion extends to the Gibbs sampling method, identified as a special case of the Metropolis-Hastings algorithm, particularly effective when direct sampling from the conditional distributions of a multivariate distribution is feasible.

- An in-depth analysis of adaptive Metropolis (AM) algorithms is provided, focusing on their dynamic parameter adjustment to enhance sampling efficiency for high-dimensional target distributions.

- The section also presents theoretical insights on the convergence properties of these adaptive algorithms, ensuring adaptivity does not compromise the chain's ergodicity.

- Practical considerations for implementing adaptive Metropolis-Hastings algorithms are explored, including strategies for maintaining theoretical guarantees while improving sampling efficiency.

Exercise 10.2

2.1 Nesting Success of Young Male Sparrows

[1] Younger male sparrows may or may not nest during a mating season, perhaps depending on their physical characteristics. Researchers have recorded the nesting success of 43 young male sparrows of the same age, as well as their wingspan, and the data appear in the file msparrownest.dat. Let Y_i be the binary indicator that sparrow i successfully nests, and let x_i denote their wingspan. Our model for Y_i is logit $\Pr(Y_i = 1|\alpha, \beta, x_i) = \alpha + \beta x_i$, where the logit function is given by $\text{logit}\,\theta = \log\left[\frac{\theta}{1-\theta}\right]$.

a) Write out the joint sampling distribution $\prod_{i=1}^{n} p(y_i|\alpha, \beta, x_i)$. and simplify as much as possible.

b) Formulate a prior probability distribution over α and β, by considering the range of $\Pr(Y = 1|\alpha, \beta, x)$ as x ranges over 10 to 15, the approximate range of the observed wingspans.

c) Implement a Metropolis algorithm that approximates $p(\alpha, \beta|y, x)$. Adjust the proposal distribution to achieve a reasonable acceptance rate, and run the algorithm long enough so that the effective sample size is at least $1,000$ for each parameter.

d) Compare posterior densities of α and β to their prior densities.

e) Using output from the Metropolis algorithm, come up with a way to make a confidence band for the following function $f_{\alpha\beta}(x)$ of wingspan:

$$f_{\alpha\beta}(x) = \frac{e^{\alpha+\beta x}}{1 + e^{\alpha+\beta x}}.$$

where α and β are the parameters in your sampling model. Make a plot of such a band.

[1] Exercise 10.2 form [DHo09] pg.245

A solution

a) Simplify the joint sampling distribution as much as possible

Given the Y_i binary results its probability distribution for each observation i is a Bernoulli distribution. Thus the optimal simplification for the joint sampling distribution is:

$$\prod_{i=1}^{n} p(y_i|\alpha, \beta, x_i) = \prod_{i=1}^{n} \frac{e^{(\alpha+\beta x_i)y_i}}{1 + e^{\alpha+\beta x_i}}$$

The analytical derivation of the equation can be found in the Appendix.

b)Formulate a prior probability distribution considering the probability of nest in the given range of x

To do so, we first need to plot the graph for such x's using a variety of α's and β's to help us decide on a prior:

```
library(ggplot2)

# Calculating probability
calculate_probability <- function(alpha, beta, x) {
  exp(alpha + beta * x) / (1 + exp(alpha + beta * x))
}

# Creating the x sequence of wingspan
wingspan_values <- seq(10, 15, by = 0.1)

# Trying some values for alpha and beta
alpha_values <- c(-10, -5, 0, 5, 10)
beta_values <- c(-2, -1, 0, 1, 2)

# The data frame for plotting
plot_data <- expand.grid(wingspan = wingspan_values, alpha = alpha_values,
↪  beta = beta_values)
plot_data$probability <- with(plot_data, calculate_probability(alpha,
↪  beta, wingspan))
```

```
# The plot
ggplot(plot_data, aes(x = wingspan, y = probability, color = factor(beta),
↪  group = interaction(alpha, beta))) +
  geom_line() +
  facet_wrap(~ alpha, scales = "free_y") +
  labs(title = "Probability of Nesting Success vs Wingspan",
       x = "Wingspan",
       y = "Probability of Nesting Success",
       color = "Beta Value") +
  theme_minimal()
```

The created plot 2.1 presents the probability in question given a variety of α's and β's. In the top-left subplot $\alpha = -10$ and as the wingspan increases from 10 to 15, the probability of nesting success increases for the positive values of β, while it remains zero for the null β value and the negative ones. The top-middle with $\alpha = -5$ yields similar plots to the first thought somewhat more extreme as depending on β the probability is either too close to 1 or too close to 0. In the top-right subplot $\alpha = 0$ and the probability of nesting is 0.5 for $\beta = 0$ 1 for the positive values and 0 for the negative. The bottom left plot with $\alpha = 5$ yields the same results as the top-middle where $\alpha = -5$ and the last bottom-right for $\alpha = 10$ provides the symmetric results of the top-left where $\alpha = -10$.

For the prior distribution, we need to select ranges for α and β that reflect our belief about the probability of nesting success before seeing the data. From the graph, we can observe that if α is too negative, it suggests that sparrows with a typical wingspan are unlikely to nest successfully, which may not align with our prior beliefs. Symmetrically if α is too positive, it suggests that almost all sparrows will nest successfully regardless of wingspan, which also seems unrealistic. Therefore, we might choose a prior distribution for α and β that centers around the values where the probability of nesting success for the range of wingspan values is neither too close to 0 nor too close to 1. A reasonable prior could be a normal distribution centered around 0 for both α and β and a standard deviation indicating that we believe most sparrows have a reasonable chance of nesting success, but we allow for a wide range of possible effects of wingspan on this probability. Given the magnitude of the wingspan I choose the standard deviation of β to be equal to 1, ($sd_\beta = 1$), since the value 1 provides little uncertainty, our uncertainty should be reflected in the variance of α which I choose it to be 3, as we want to avoid values that are too

positive or too negative it will be ill-advised to choose a standard deviation greater than that. Thus, I declare the priors for $\alpha \sim \text{Normal}(0, 3^2)$ and $\beta \sim \text{Normal}(0, 1^2)$.

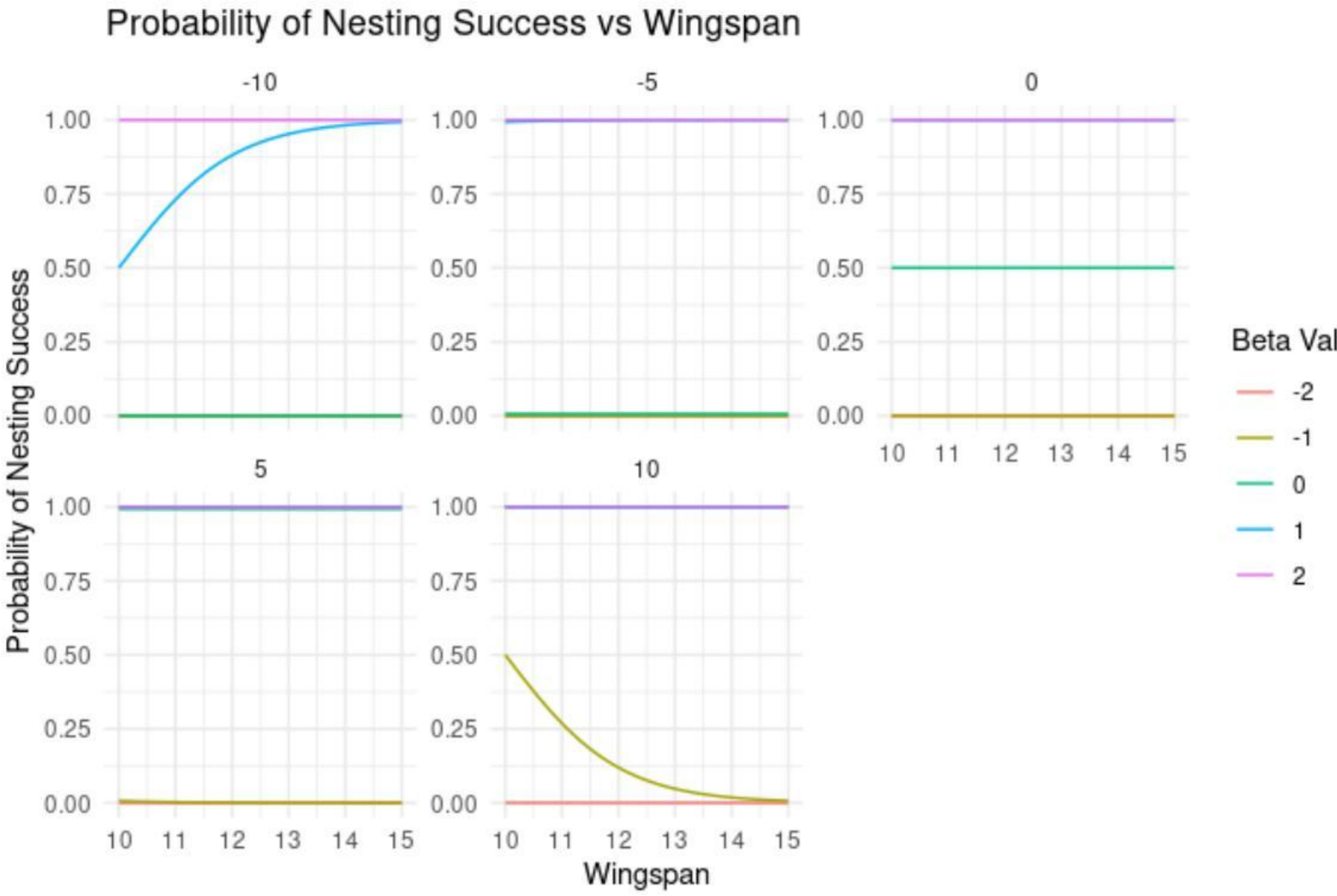

Figure 2.1: Some plots to help us decide prior

2.1.1 c) The Metropolis Algorithm

To run the Metropolis Algorithm we first need to upload our data:

```
success <- c(
  0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
  0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0,
  0, 1, 1
)
```

```
wingspan <- c(
  13.03, 13.69, 12.62, 11.70, 12.39, 12.44, 12.22, 13.65, 14.30, 12.39,
  13.51, 13.21, 13.02, 12.89, 14.79, 11.57, 14.26, 13.94, 13.85, 11.34,
  12.41, 13.88, 11.98, 11.18, 13.17, 14.62, 15.41, 14.70, 13.45, 12.66,
  12.73, 12.36, 10.59, 12.02, 13.52, 12.97, 12.96, 11.42, 13.12, 13.37,
  13.11, 13.10, 11.83
)
```

```
y<-success
x<-wingspan
```

Then we create the likelihood function:

```
# The logistic function
logistic <- function(alpha, beta, x) {
  1 / (1 + exp(-(alpha + beta * x)))
}

# The likelihood function
likelihood <- function(alpha, beta, y, x) {
  p <- logistic(alpha, beta, x)
  prod(p^y * (1 - p)^(1 - y))
}
```

We introduce the priors we decided on step b)

```
# The prior functions
prior <- function(alpha, beta, alpha_mean = 0, alpha_sd = 3, beta_mean =
↪  0, beta_sd = 1) {
  dnorm(alpha, alpha_mean, alpha_sd) * dnorm(beta, beta_mean, beta_sd)
}
```

Finally we run the Metropolis Algorithm:

```
metropolis <- function(y, x, iterations, start = c(alpha = 0, beta = 0)) {
  chain <- matrix(NA, nrow = iterations, ncol = 2)
  colnames(chain) <- c("alpha", "beta")
  chain[1, ] <- start

  for (i in 2:iterations) {
    # The Proposal distribution
    proposal <- rnorm(2, mean = chain[i - 1, ], sd = c(0.1, 0.1)) # sd can
    ↪  be adjusted

    # The acceptance ratio
    current_like <- likelihood(chain[i - 1, "alpha"], chain[i - 1,
    ↪  "beta"], y, x)
    proposal_like <- likelihood(proposal[1], proposal[2], y, x)
    current_prior <- prior(chain[i - 1, "alpha"], chain[i - 1, "beta"])
```

```
proposal_prior <- prior(proposal[1], proposal[2])

acceptance_ratio <- (proposal_like * proposal_prior) / (current_like *
↪  current_prior)

# Accept or reject the proposal
if (runif(1) < acceptance_ratio) {
    chain[i, ] <- proposal
  } else {
    chain[i, ] <- chain[i - 1, ]
  }
 }

 return(chain)
}
```

As you can see from the code the Metropolis algorithm function returns the whole chain on which we can use the library's "coda" "effectiveSize" build-in function to get our effective sample size. I run the algorithm for 3 different interetional values 10000, 100000 and 1000000, found the effective sample size and plotted the chain after separating their values using mcmc build-in "coda" function. The resulting plots 2.2, 2.3, 2.4 include, insightful trace plots, for the convergence of the algorithm and density plots which can help us in the discussion of part d).

These results were provided by running the following code:

```
set.seed(123)
chain <- metropolis(y, x, iterations = 1000000)

library(coda)
chain_mcmc <- mcmc(chain)
effectiveSize(chain_mcmc)
plot(chain_mcmc)
```

and the resulting effective sample sizes were:

```
# for 10000 iterations
   alpha     beta
3.137940 4.022989
```

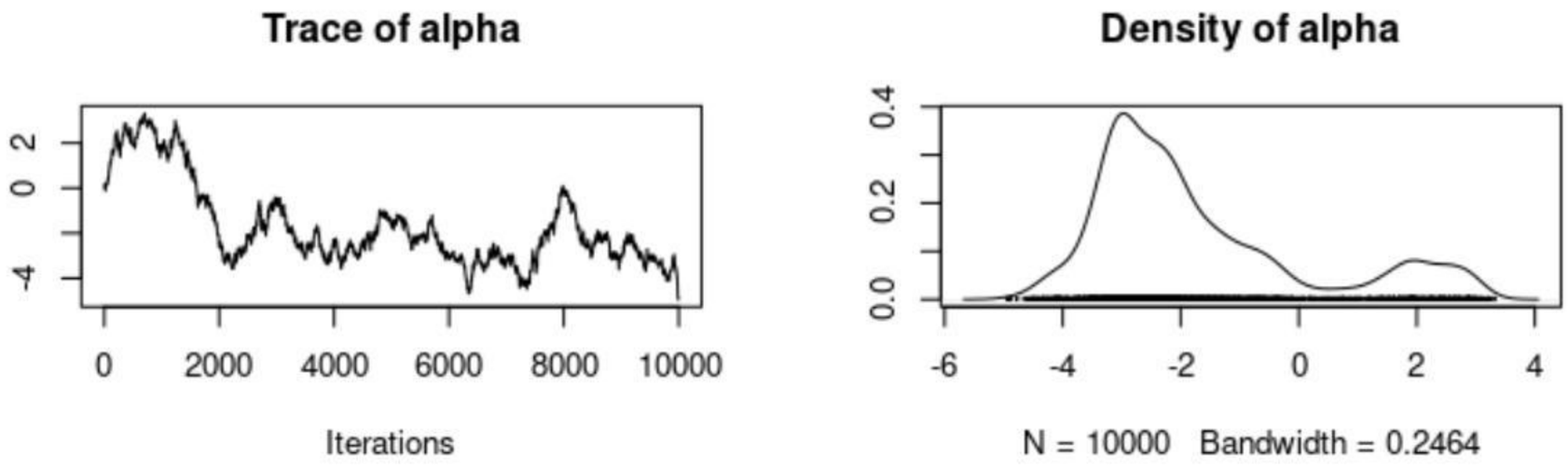

Figure 2.2: Trace and Density plots for 10000 iterations

```
#For 100000 iterations
   alpha     beta
17.76347 19.86714

For 1000000 iterations
   alpha     beta
127.6889 134.4572
```

So the 10000 iterations were sufficient for the exercise as their effective sample size was more than 1000 for each parameter. The reason I ran it for more was that the trace plots for that many iterations (also for 100000) yielded troubling trace plots, in the sense that they provided doughty of convergence as they seemed to visualize patterns in the graph. though in the 1000000 iterations, we can see that based on the trace plots there are no apparent patterns thus there is not any indication that the algorithm did not converge.

To calculate the acceptance rate for the implementation, one can introduce some counters in the Metropolis Algorithm like so:

```
#The Metropolis algorithm with counters
metropolis <- function(y, x, iterations, start = c(alpha = 0, beta = 0)) {
  chain <- matrix(NA, nrow = iterations, ncol = 2)
```

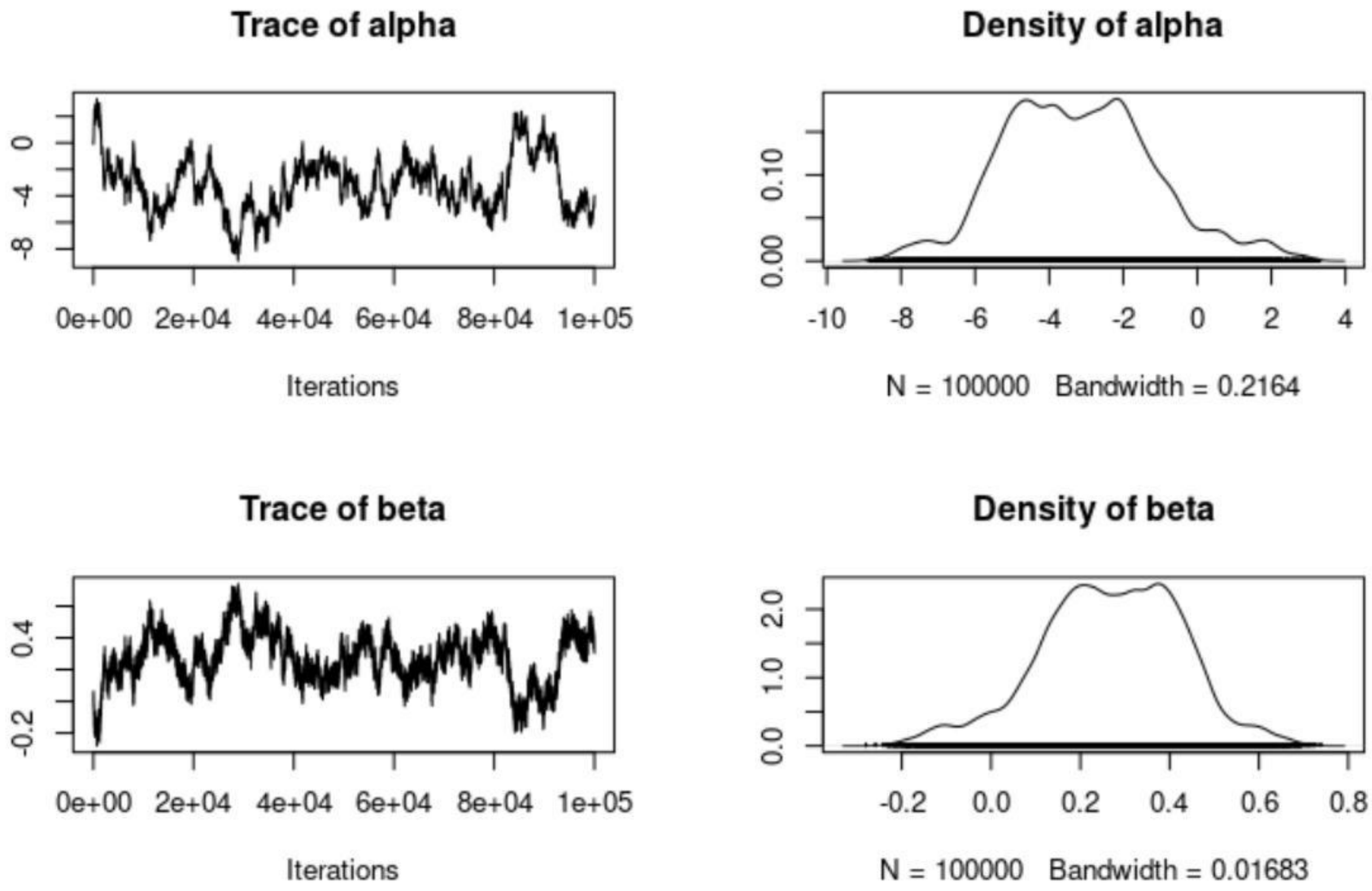

Figure 2.3: Trace and Density plots for 10000 iterations

```r
colnames(chain) <- c("alpha", "beta")
chain[1, ] <- start

accepted <- 0  # Counter for accepted proposals

for (i in 2:iterations) {
  # The Proposal distribution
  proposal <- rnorm(2, mean = chain[i - 1, ], sd = c(0.1, 0.1)) # Adjust
  ↪  sd later

  # The acceptance ratio
  current_like <- likelihood(chain[i - 1, "alpha"], chain[i - 1,
  ↪  "beta"], y, x)
  proposal_like <- likelihood(proposal[1], proposal[2], y, x)
  current_prior <- prior(chain[i - 1, "alpha"], chain[i - 1, "beta"])
  proposal_prior <- prior(proposal[1], proposal[2])

  acceptance_ratio <- (proposal_like * proposal_prior) / (current_like *
  ↪  current_prior)
```

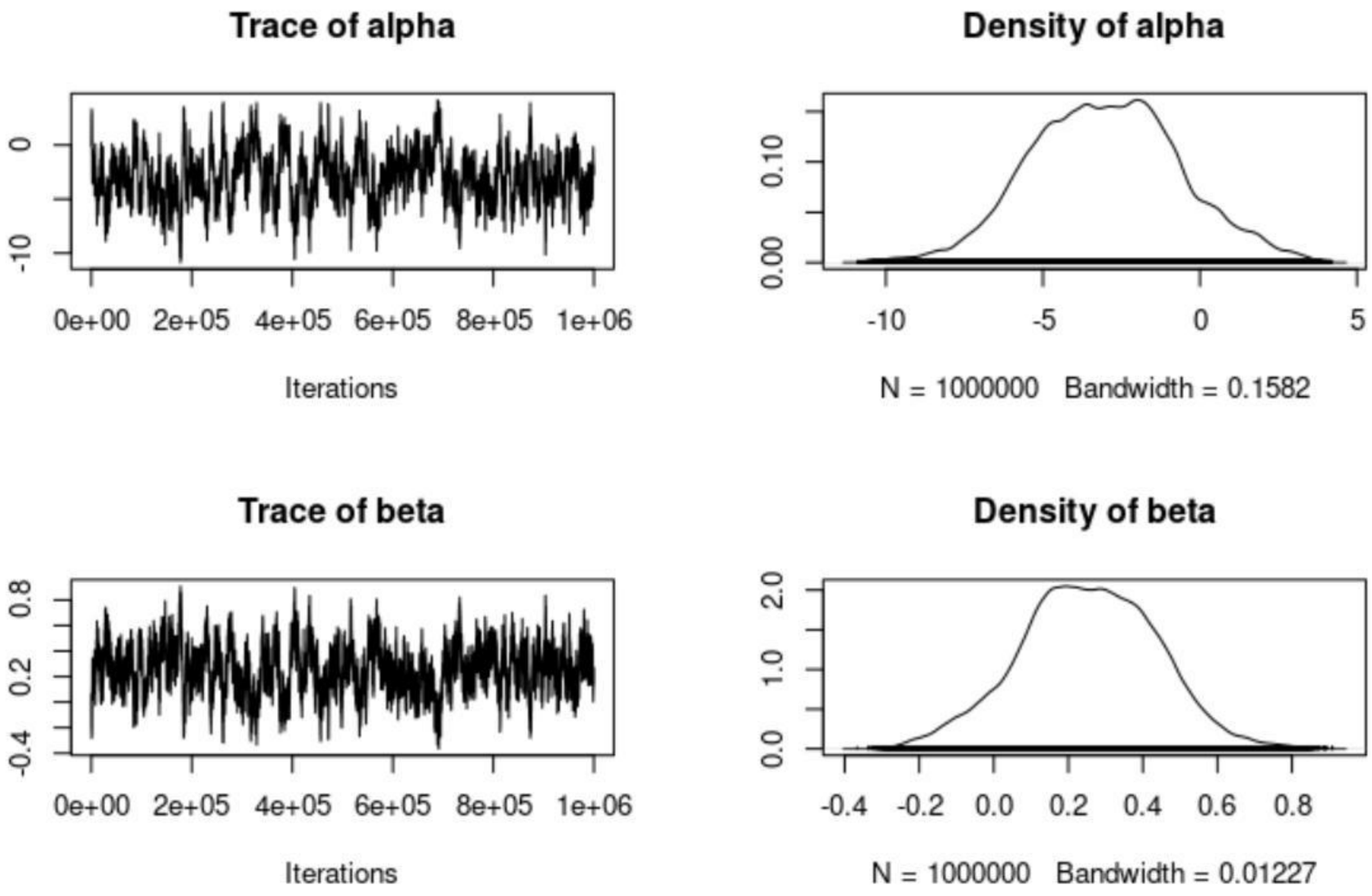

Figure 2.4: Trace and Density plots for 1000000 iterations

```
  # Accept or reject the proposal
    if (runif(1) < acceptance_ratio) {
      chain[i, ] <- proposal
      accepted <- accepted + 1  # The accepted counter increment
    } else {
      chain[i, ] <- chain[i - 1, ]
    }
  }
  # The acceptance rate calculation
  acceptance_rate <- accepted / (iterations - 1)
  cat("Acceptance rate:", acceptance_rate, "\n")
  return(chain)
}
```

The resulting acceptance rate is shown below and it's in the reasonable range (over 20% and less than 50):

```
> chain <- metropolis(y, x, iterations = 1000000)
Acceptance rate: 0.2867663
```

Here I would like to note that we can adjust the proposal's distribution standard deviation according to our needs (to better fit the data), but first let's move on to

d).

2.1.2 d) Comparing the prior to the posterior densities

Given the resulting densities of the Metropolis algorithm produced chains for 1000000 iterations we can say that the posterior densities are close to normal in shape, with mean and standard deviation not that far from our initial normal priors (intuitive guess). The joint distribution affected the prior in a way that the posterior mean of α drifted to negative values, with an (optically) approximating mean value of -3 and sd=2.5 (based on the 2.4 where the -2sd seems to be around 8), while the posterior of β resembles a normal distribution with mean 0.2 and sd =0.2. To further examine the relation of the prior and posterior (mostly to enhance our reasoning) we can change the standard deviation of the proposal distribution to better fit our data based on the results we got from the previous run we analyzed:

```
proposal <- rnorm(2, mean = chain[i - 1, ], sd = c(0.5, 0.1))
```

This code implementation resulted in a larger effective sample size for both variables and smother Density plots:

```
> chain <- metropolis(y, x, iterations = 1000000)
Acceptance rate: 0.2660543
> effectiveSize(chain_mcmc)
   alpha     beta
2483.206 2561.123
```

We can see from the plot 2.5, that this small adjustment from the initial (0.1,0.1) to (0.5,0.1) provided better trace plots with less possible patterns than the previous one and smoother better concentrated posterior densities. Some additional tests could be made by implementing the code for a variety of normally distributed priors with different means and variances and more fine-tuning on the sd of the proposal distribution, though I believe they would only enhance our prior analysis.

2.1.3 e) Creating a band and plotting

To find the band we can take that mcmc output from the Metropolis algorithm, extract the α's and β's to compute our function using each pair (α, β) from the posterior samples, for the originally asked grid of x values (10-15). Then we can simply calculate the quantiles, the 2.5%, and the 97.5% for each x to create the lower and upper bounds of the confidence band. This band would indicate the

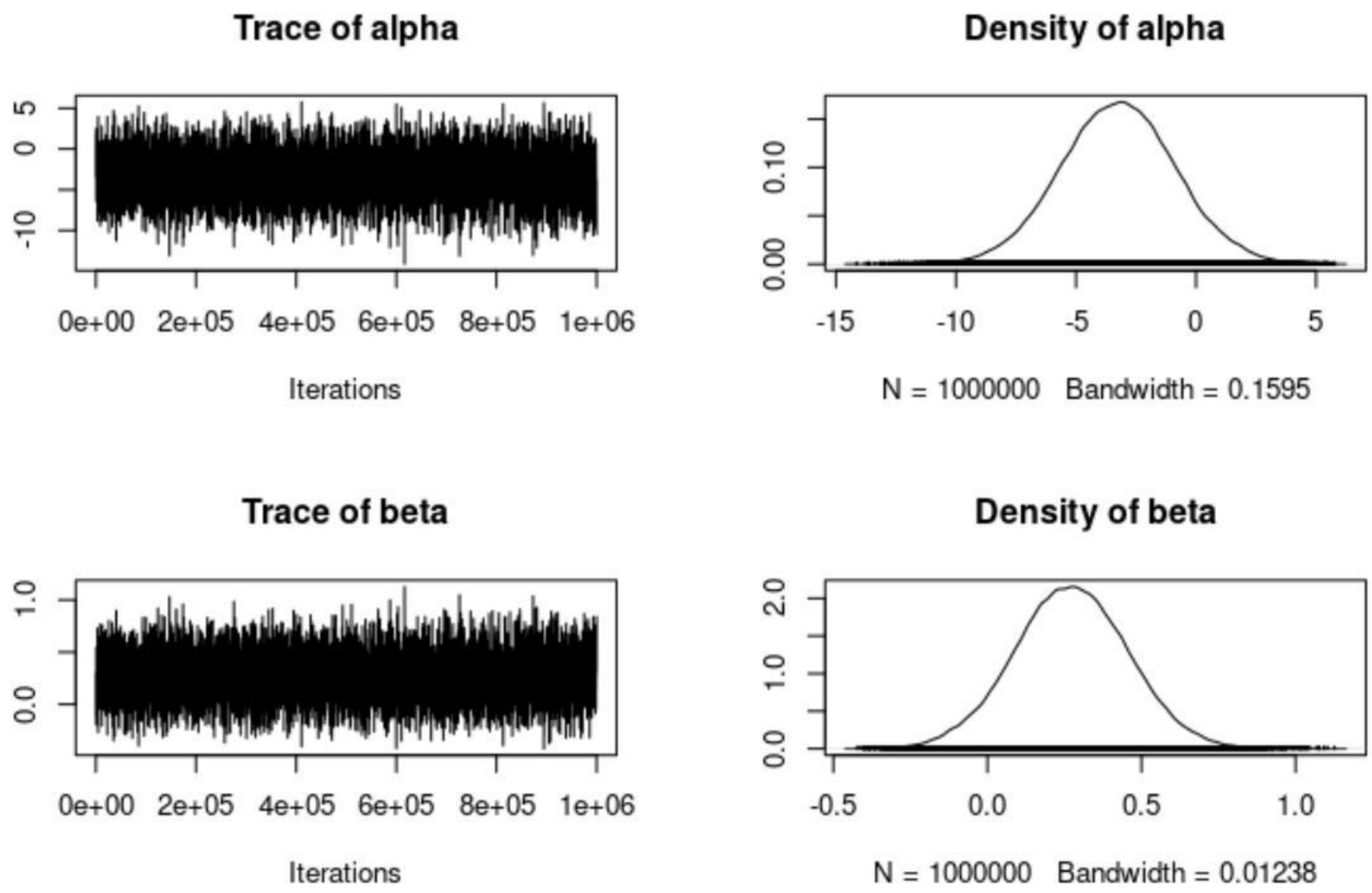

Figure 2.5: Trace and Density plots for adjusted Proposal sd

range within which the true function is expected to lie with 95% confidence level. This can be done using the following code:

```
# The logistic function
logistic_function <- function(alpha, beta, x) {
  exp(alpha + beta * x) / (1 + exp(alpha + beta * x))
}

# The original wingspan grid
wingspan_grid <- seq(10, 15, length.out = 100) # can be adjusted

# The matrix to store the evaluations
f_matrix <- matrix(NA, nrow = length(wingspan_grid), ncol =
↪  nrow(chain_mcmc))

# Calculating the function for each sample and wingspan
for (i in 1:nrow(chain_mcmc)) {
  f_matrix[, i] <- logistic_function(chain_mcmc[i, "alpha"], chain_mcmc[i,
  ↪  "beta"], wingspan_grid)
}
```

```
# The quantiles for the confidence band
lower_bound <- apply(f_matrix, 1, quantile, probs = 0.025)
upper_bound <- apply(f_matrix, 1, quantile, probs = 0.975)
median_line <- apply(f_matrix, 1, median)
```

We can produce the illustration 2.6 using the library "ggplot" and the code below

```
# The confidence band plot
plot(wingspan_grid, median_line, type = 'l', lty = 1, col = 'blue',
     ylim = range(c(lower_bound, upper_bound)), xlab = "Wingspan", ylab =
     ↪  "f_alpha_beta",
     main = "Confidence Band ")
polygon(c(wingspan_grid, rev(wingspan_grid)), c(upper_bound,
↪  rev(lower_bound)), col = 'grey70', border = NA)
lines(wingspan_grid, median_line, col = 'blue')
```

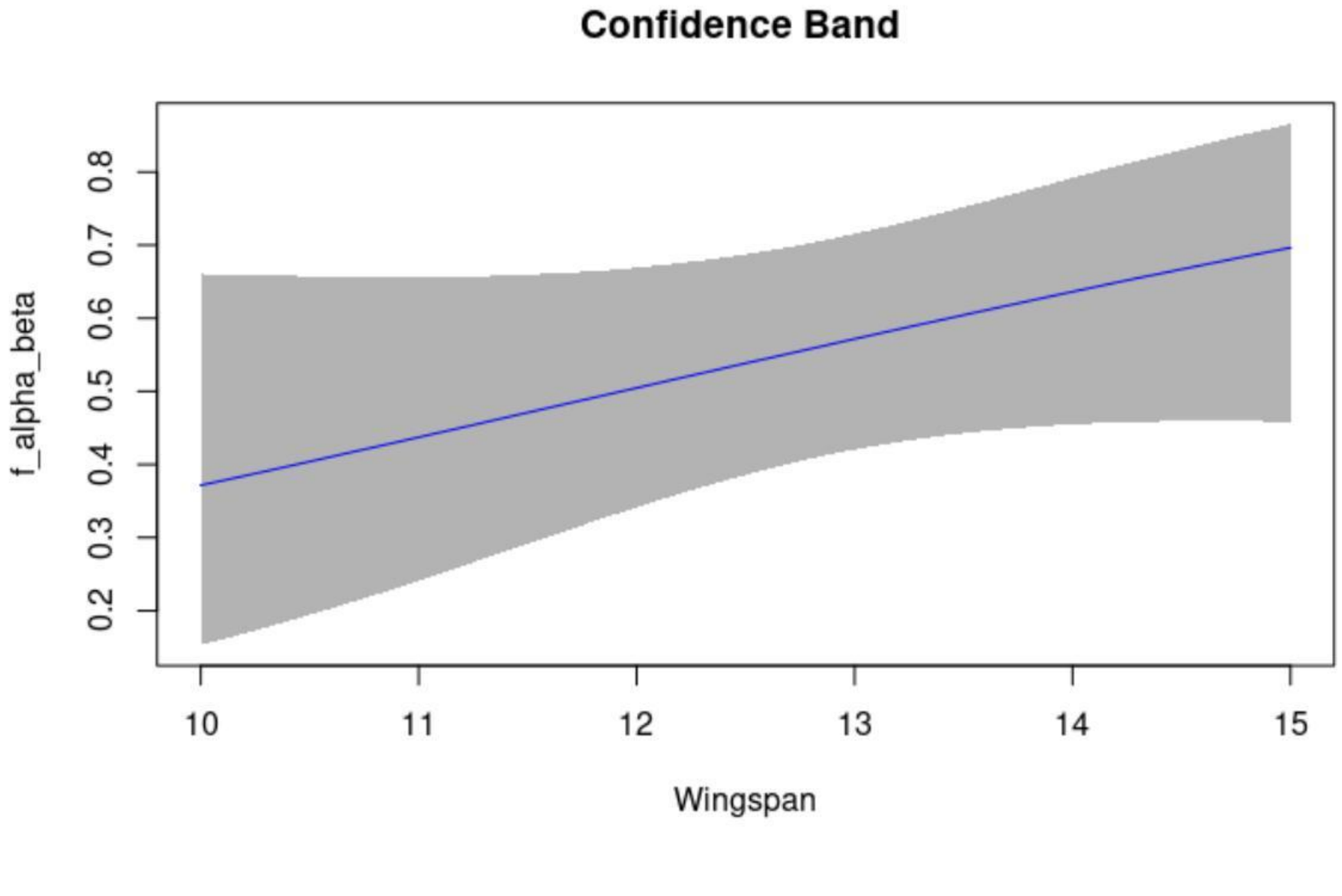

Figure 2.6: The Confidence Band

A similar and more robust band can be created by discarding the burn-in period for instance we can start by discarding the first 10000 pairs:

```
# Define the burn-in period
burn_in <- 10000
```

```
# Discarding the burn-in samples
chain_mcmc_burned_in <- chain_mcmc[(burn_in + 1):nrow(chain_mcmc), ]
```

We can adjust the code accordingly:

```
# The matrix to store the evaluations
f_matrix_n <- matrix(NA, nrow = length(wingspan_grid), ncol =
↪  nrow(chain_mcmc_burned_in))

# Calculating the function for each sample and wingspan
for (i in 1:nrow(chain_mcmc_burned_in)) {
  f_matrix_n[, i] <- logistic_function(chain_mcmc[i, "alpha"],
  ↪  chain_mcmc[i, "beta"], wingspan_grid)
}

# The quantiles for the confidence band
lower_bound <- apply(f_matrix_n, 1, quantile, probs = 0.025)
upper_bound <- apply(f_matrix_n, 1, quantile, probs = 0.975)
median_line <- apply(f_matrix_n, 1, median)

# The confidence band plot
plot(wingspan_grid, median_line, type = 'l', lty = 1, col = 'blue',
     ylim = range(c(lower_bound, upper_bound)), xlab = "Wingspan", ylab =
     ↪  "f_alpha_beta",
     main = "Confidence Band burn-in 10000 ")
polygon(c(wingspan_grid, rev(wingspan_grid)), c(upper_bound,
↪  rev(lower_bound)), col = 'grey70', border = NA)
lines(wingspan_grid, median_line, col = 'blue') # Overlay the median line
↪  for clarity
```

The resulting plot 2.7, provides little to no optical changes from the initial band (probably because 10000 is a number much to small for burn-in of 1000000 iterations). To optically observe a substantial difference we can re-create the experiment for 100000 burn-in pairs. By doing so, the plot 2.8 can be produced. This plot provides an optical difference to the band. As the bound is more concentrated around the median line and the median line has tilted, somewhat, to values with less distance from 0.5 (the degree of the angle from the x-axis is smaller than before), this band better represents our 95% confidence level for the area in which the true function is expected to be.

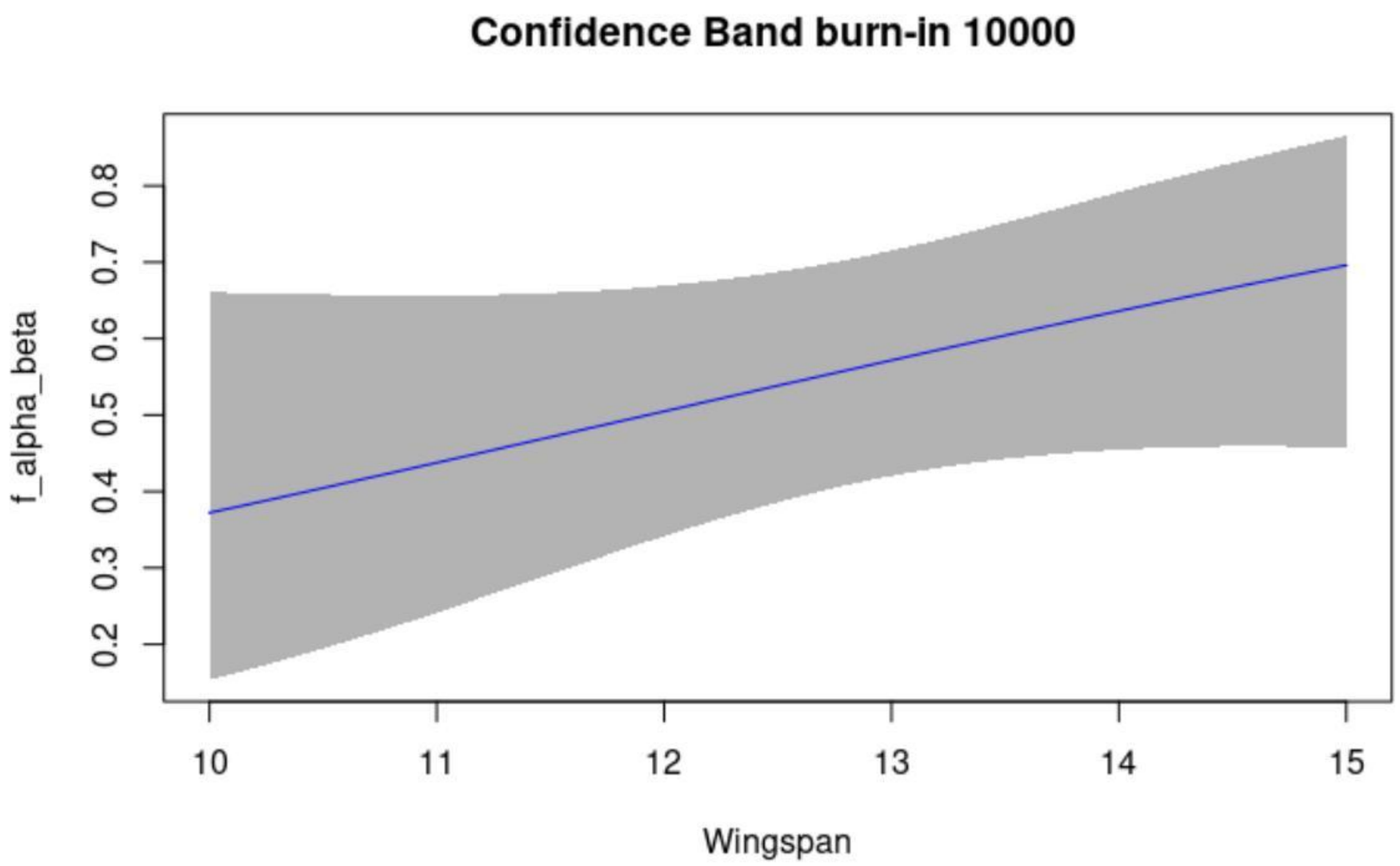

Figure 2.7: The Confidence Band burn-in 10000

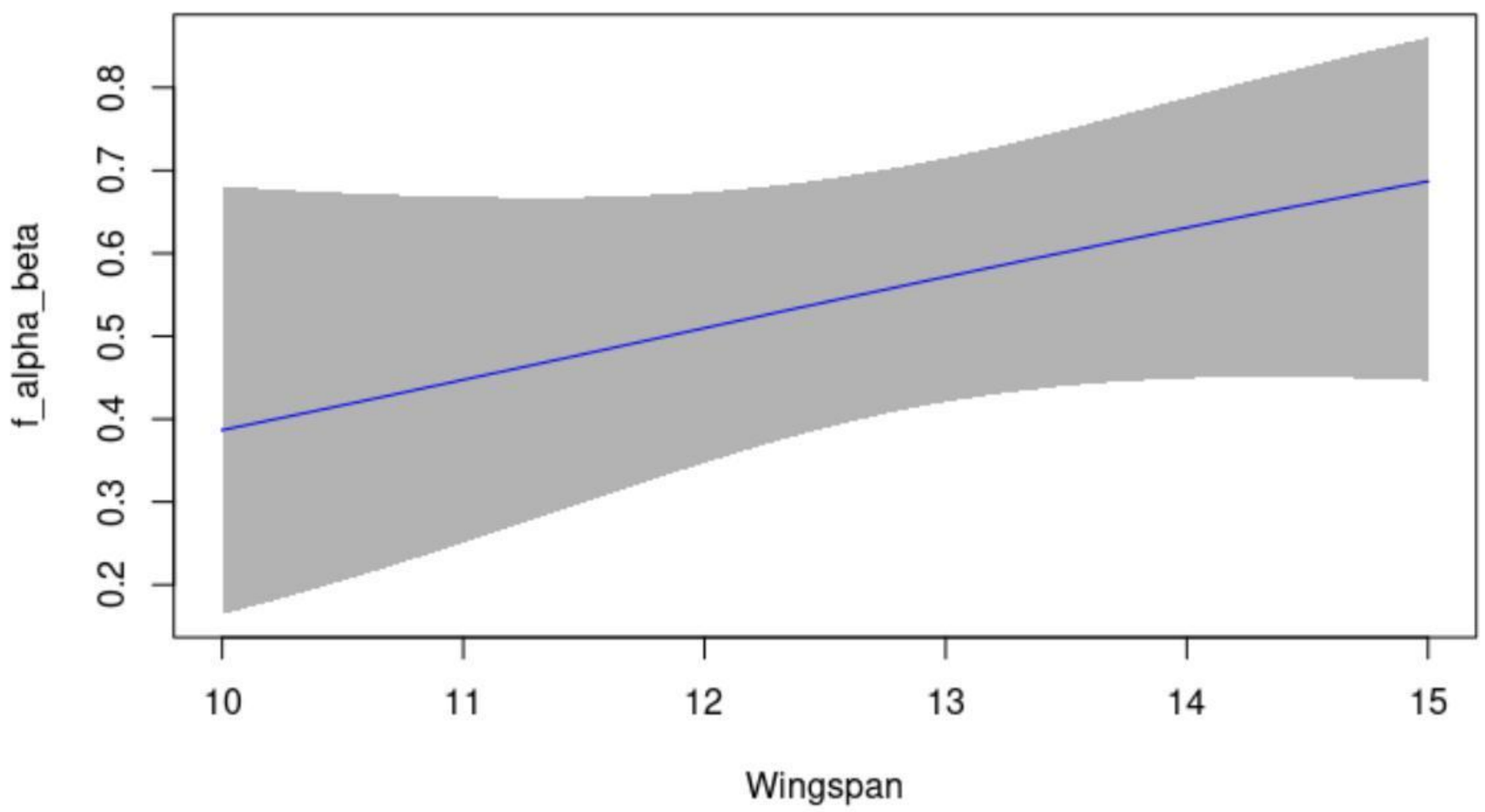

Figure 2.8: The Confidence Band burn-in 100000

Appendix A

The Analytical derivation of the simplified joint distribution

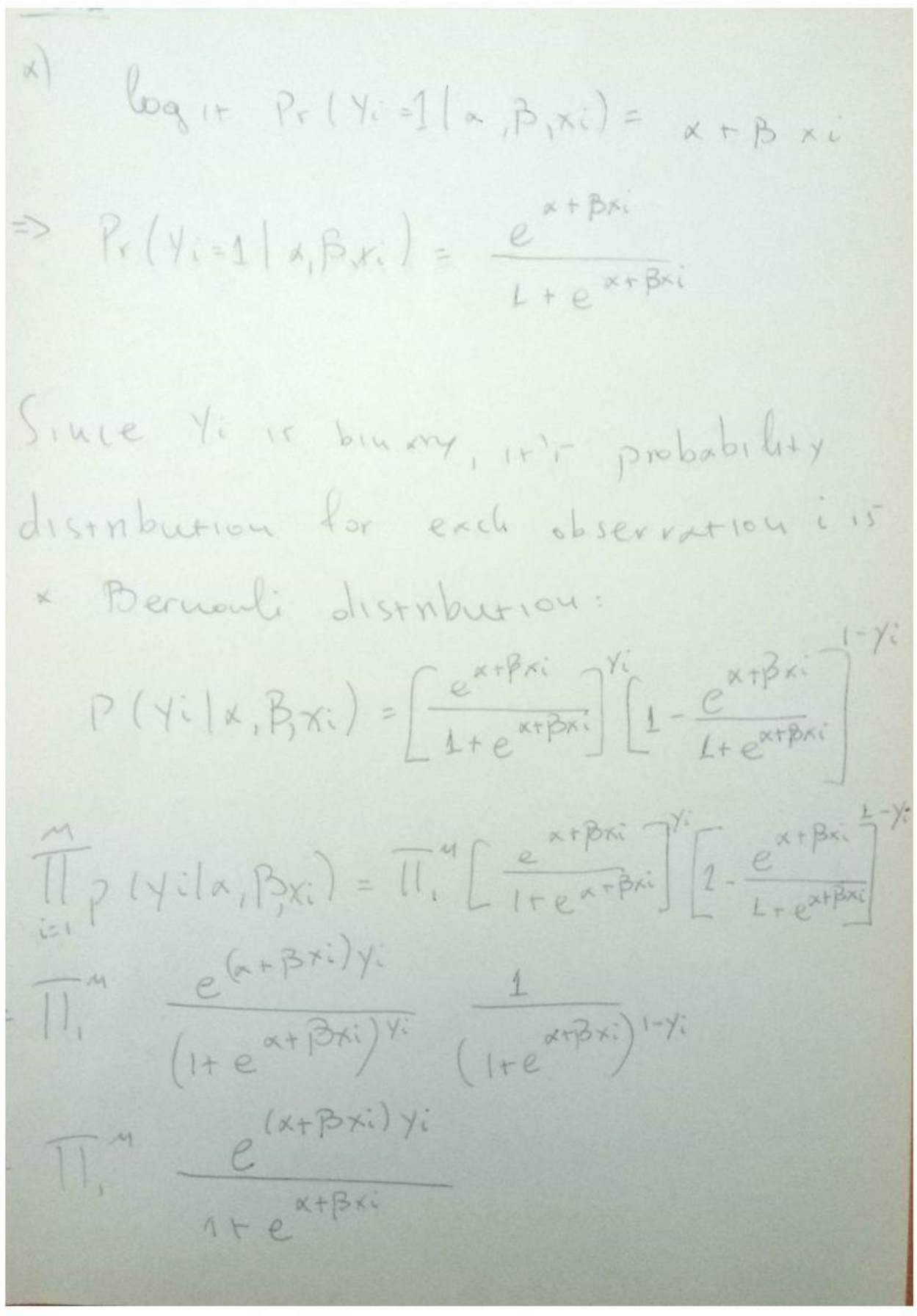

Figure A.1: Optimal simplification of the joint sampling distribution derivation.

Bibliography

[DHo09] Peter D.Hoff. *A First Course in Bayesian Statistical Methods*. 233 Spring Street, New York, NY10013, USA: Springer Science+Business Media, LLC, 2009. ISBN: 9780387922997.