Assignment Part III

Silpa Soni Nallacheruvu (19980824-5287) Hernan Aldana (20000526-4999)

2024-10-23

set.seed(980824)

Summary

This report covers three main tasks under a Bayesian framework applied to logistic regression modeling. Task 1 focuses on validating AIC calculations and leave-one-out cross-validation to assess model accuracy. Task 2 involves implementing a function to compute the posterior density of regression parameters, enhancing the model's estimation under Bayesian principles. Task 3 is dedicated to applying the Metropolis-Hastings algorithm for posterior sampling, aiming to simulate from the posterior distribution of parameters, assess convergence, and estimate credibility intervals and predictive probabilities for new observations.

Task 1

Approach :

To calculate the AIC, the formula

$2 * k - 2 * l(\hat{\theta_{ml}})$

was applied where k is the number of parameters in the theta. To calculate the leave-one-out cross-validation, the formula

$\frac{\sum_{i=1}^{n} l_i(\hat{theta_i})}{n}$

was applied where $\hat{\theta_i}$ is $\hat{\theta_{ml}}(X_{-i})$, (X_{-i}) is one observation (X_i) left out of X, $l_i(\hat{theta_i})$ is the log likelihood for the i-th left out observation and n is the total number of observations. These values were then compared with the AIC from R in the provided summary.

Code :

```
# ---- Task_1 ----

# Compute AIC = 2k - 2l(theta_ml)
n <- nrow(X)
theta0 = rep(0, ncol(X))
k <- length(theta0)
theta_estimate <- NR(theta0, 3, y, X)
log_likelihood <- l(theta_estimate, y, X)
aic_computed <- 2*k - 2*log_likelihood

#AIC output from R summary
r_summary_aic <- summary(modell)$aic
```

```
# Compute k_cv =sum(l_i(theta_i))/n
# Here, theta_i = theta_ml(X_-i)
nk_cv <- 0
for(i in 1:n) {
  X_minus_i <- X[-i, , drop=FALSE]
  y_minus_i <- y[-i, , drop=FALSE]
  X_i <- X[i, , drop=FALSE]
  y_i <- y[i, , drop=FALSE]
  theta_i <- NR(theta0, 3, y_minus_i, X_minus_i)
  # log likelihood for i-th observation
  log_likelihood_theta_i <- l(theta_i, y_i, X_i)
  nk_cv <- nk_cv + log_likelihood_theta_i
}

k_cv <- nk_cv/n

# Creating a comparison data frame
comparison_aic_values <- data.frame(
  "AIC_R_model" = r_summary_aic,
  "AIC_computed" = aic_computed,
  "2*nK_CV_computed" = 2*nk_cv
)
```

Output :

```
comparison_aic_values
```

```
##   AIC_R_model AIC_computed X2.nK_CV_computed
## 1    1302.397     1302.397         -1302.367
```

Observation :

The computed AIC value and the AIC value from the R summary coincide in value. The computed K_CV is in the same magnitude as $AIC/(-2n)$ as expected.

Task 2

Approach :

The a posteriori density combines the a priori and the likelihood of the data. For the logistic regression, we have y data, X and the parameter vector θ. so the a posteriori density is proportional to: $P(\theta|y, X) \propto P(y|X, \theta)P(\theta)$ where $P(\theta)$ is the a prior density and $P(y|X, \theta)$ is. the likelihood from the logistic regression model.

for a binary outcome, the likelihood for the logistic regression is given by the following equation:

$P(y_i|X_i, \theta) = \frac{1}{(1+\exp(-X_i\theta)}$

The a priori is Gaussian: $\theta \sim N(0, 100I)$ so the a prior density is the following: $P(\theta) \propto exp(-\frac{1}{2}\theta^T(100I)^{-1}\theta)$

Code :

```
# ---- Task_2 ----
post <- function(theta, y, X) {
  eta <- X %*% theta # Logistic regression likelihood
```

```
  likelihood <- prod(plogis(eta)^y * (1 - plogis(eta))^(1 - y))
  prior <- exp(-0.5 * sum(theta^2 / 100)) # a priori with theta ~ N(0, 100 * I)
  posterior <- likelihood * prior # posteriori is proportional to a priori * likelihood
  return(posterior)
}
```

Output :

```
Xtest <- cbind(1, 18:25, rep(c(0, 1), 4), rep(c(1, 1, 0, 0), 2))
ytest <- c(rep(TRUE, 4), rep(FALSE, 4))
testing<-post(c(260, -10, 10, -20), ytest, Xtest) / post(c(270, -15, 15, -25), ytest , Xtest)
testing
```

```
## [1] 3.707556e+25
```

Observation :

Given the results obtained by the test the function works correctly.

Task 3

Approach :

Implemented the Metropolis-Hastings algorithm over 10000 iterations while checking on the condition if the acceptance probability is greater than the acceptance ratio which is generated randomly as $ratio \sim U(0, 1)$. We accepted the theta proposal if the condition was true.

Code :

```
# ---- Task_3 ----

mh_algo <- function(theta_estimate, y, X) {
  N <- 10000
  theta <- matrix(nrow = N, ncol = 4)
  # initial value as the calculated theta_estimate
  theta[1,] <- theta_estimate
  # here, suggested sigma as standard error from part II
  sigma <- standard_error(theta_estimate, y, X)
  for (i in 2:N) {
    theta_star <- theta[i-1,] + rnorm(4) * sigma
    # check if the acceptance probability is greater than the acceptance ratio
    posterior_theta_star <- post(theta_star, y, X)
    posterior_theta <- post(theta[i-1,], y, X)
    ratio <- runif(1)
    if (posterior_theta_star/ posterior_theta > ratio) {
      theta[i,] <- theta_star  # Accept the proposal
    } else {
      theta[i,] <- theta[i-1,]  # Reject the proposal
    }
  }
  return(theta)
}
```

```r
theta_sample <- mh_algo(theta_estimate, y, X)
```

Output :

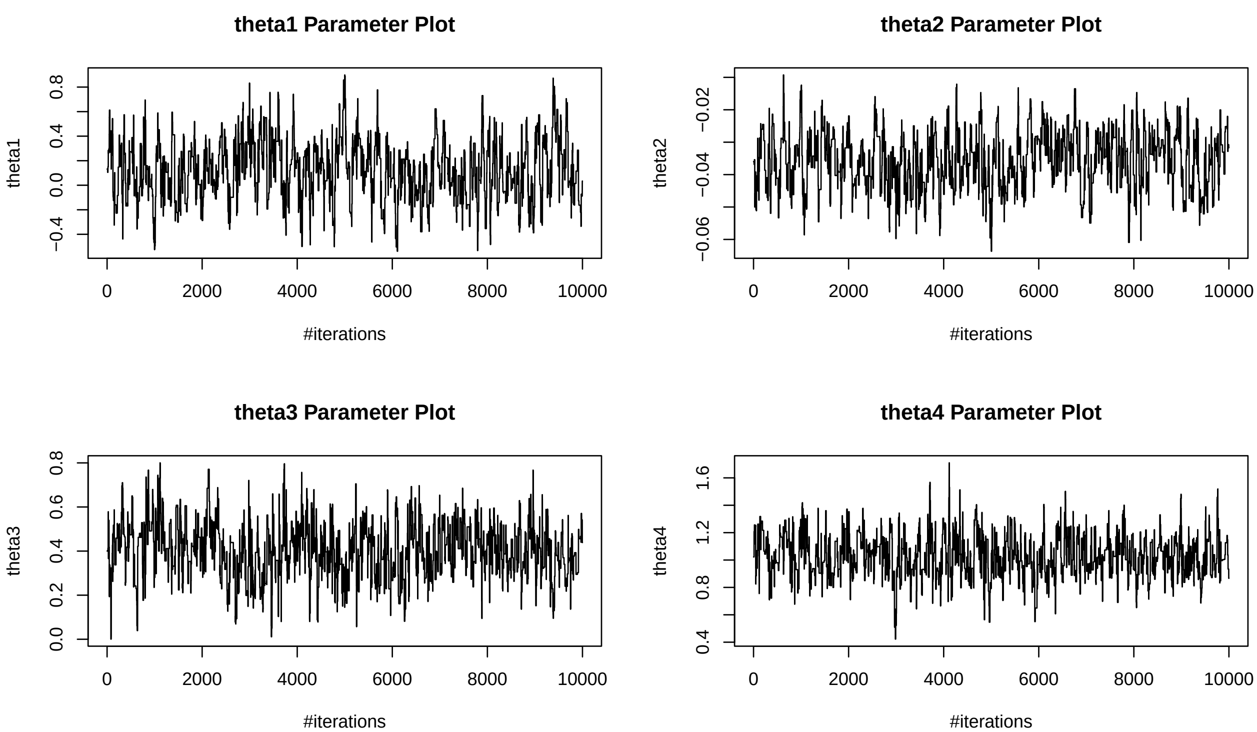
Parameter Plots

```r
par(mfrow=c(2,2))
plot(x = c(1:10000), y = theta_sample[,1], type = "l", col = "black",
main="theta1 Parameter Plot", xlab= "#iterations", ylab = "theta1")

plot(x = c(1:10000), y = theta_sample[,2], type = "l", col = "black",
main="theta2 Parameter Plot", xlab= "#iterations", ylab = "theta2")

plot(x = c(1:10000), y = theta_sample[,3], type = "l", col = "black",
main="theta3 Parameter Plot", xlab= "#iterations", ylab = "theta3")

plot(x = c(1:10000), y = theta_sample[,4], type = "l", col = "black",
main="theta4 Parameter Plot", xlab= "#iterations", ylab = "theta4")
```



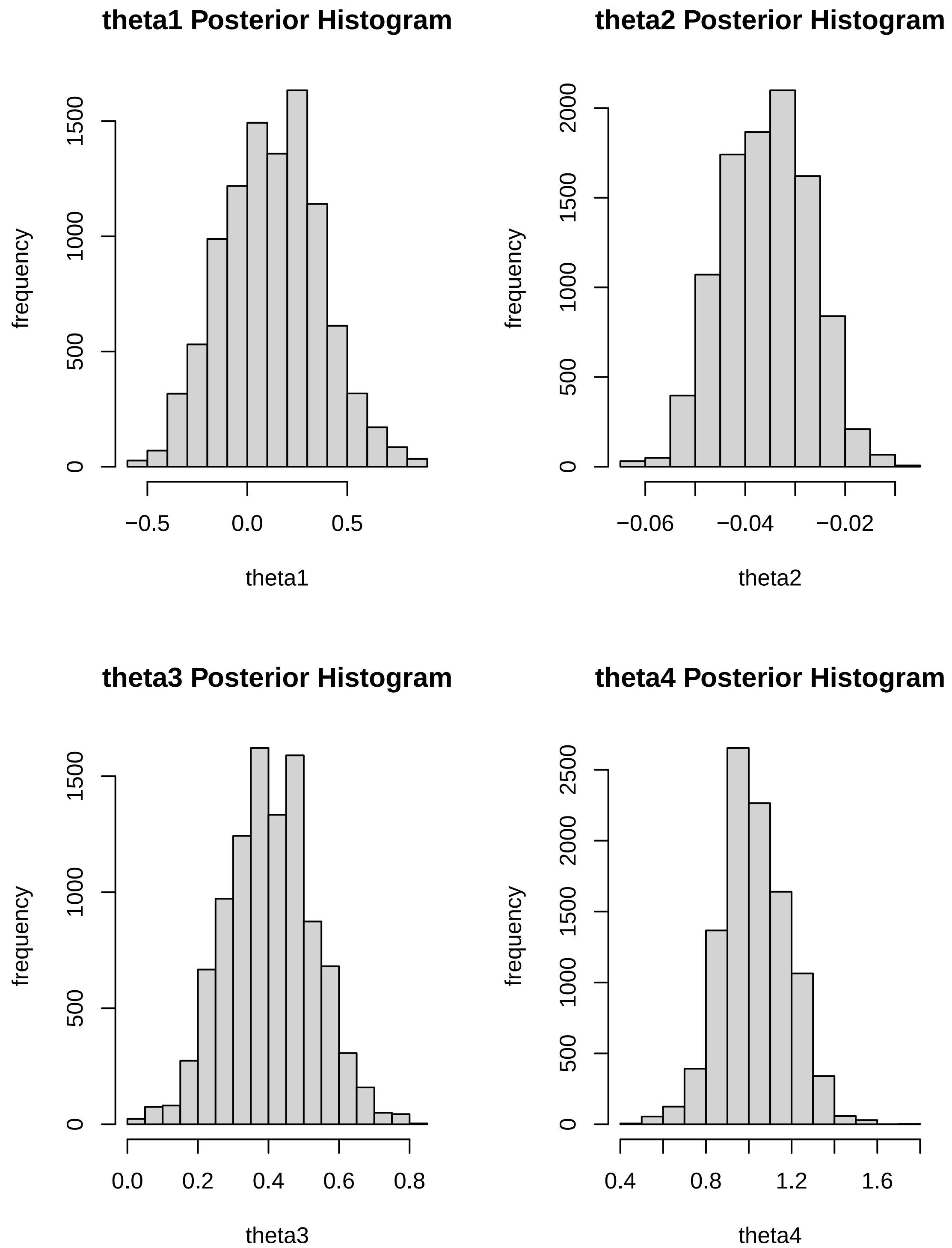
Parameter Posteriors Histograms

```r
par(mfrow=c(2,2))
hist(x = theta_sample[(1:10000), 1], col = "lightgray",
main="theta1 Posterior Histogram", xlab= "theta1", ylab = "frequency")

hist(x = theta_sample[(1:10000), 2],  col = "lightgray",
main="theta2 Posterior Histogram", xlab= "theta2", ylab = "frequency")
```

```
hist(x = theta_sample[(1:10000), 3], col = "lightgray",
main="theta3 Posterior Histogram", xlab= "theta3", ylab = "frequency")

hist(x = theta_sample[(1:10000), 4], col = "lightgray",
main="theta4 Posterior Histogram", xlab= "theta4", ylab = "frequency")
```



95% credibility intervals

```r
c_i <- apply(theta_sample, 2, quantile, probs = c(0.025, 0.975))
print(t(c_i))
```

```
##              2.5%       97.5%
## [1,] -0.33585603  0.61317680
## [2,] -0.05211341 -0.01987185
## [3,]  0.16101927  0.65019500
## [4,]  0.74633908  1.33013667
```

```r
# frequentist 95% ci:
frequentist_ci <- cbind(theta_estimate- 1.96 * standard_error(theta_estimate, y, X),
                        theta_estimate + 1.96 * standard_error(theta_estimate, y, X))
print(frequentist_ci)
```

```
##                               [,1]        [,2]
## (Intercept)            -0.35134563  0.61547550
## Alder                  -0.05326235 -0.01871364
## KonMan                  0.13284303  0.66733331
## UtbildareTrafikskola  0.71235944  1.33278775
```

Future Test

We calculate the conditional probability of success as $(P(Y^* = 1 \mid y) = E(p(x^*) \mid y))$, where (x^*) represents the features of the specific observation as mentioned.

Here, $p(x^*)$ is a function of random variable θ. Used the logistic formula : $p(x^*, \theta) = \frac{1}{1+\exp(-\theta^T x^*)}$

Hence, $E(p(x^*, \theta) \mid y))$ is calculated using Monte Carlo Integration by the sample mean $\frac{1}{N}\sum_{i=1}^{N} p(\theta_i, x^*)$ for N = 10000 samples and $theta_i$ as i-th sample from theta estimate of MH algorithm.

```r
# a privately educated subject of your own sex and age.
# intercept = 1
# Alder = 26    # Own age
# KonMan = 0    # Female
# Utbildare = 0 # Privately educated
X_star <- c(1, 26, 0, 0)
# model the probability using regression framework
p_x <- function(theta, x) {
  return (1 / (1 + exp(-t(theta)*x)))
}
# compute the probability of pass wrt X_star
prob_y <- rep(0, 10000)
for(i in 1:10000) {
  theta <- theta_sample[i, ]
  prob_y[i] <- p_x(theta, X_star)
}
expected_prob_y <- mean(prob_y)
cat("expected probability of y* : ", expected_prob_y)
```

```
## expected probability of y* :  0.5300772
```