STAT7630: Bayesian Statistics
Lecture Slides # 13

Bayesian Logistic Regression Models
Chapter 13 (Logistic Regression)

Elvan Ceyhan
Department of Mathematics & Statistics
Auburn University

Fall 2024,
Updated: November, 2024

Outline

Bayesian Logistic Regression

- Prediction and Classification
- Evaluating Model Performance
- Bayesian Logistic Regression with Multiple Predictors

Regression for Binary Data

- Consider a regression framework where the response variable Y is binary, taking exactly two values (e.g., Pass/Fail, Survive/Die, Win/Loss), typically encoded as 0 or 1.
- Traditional models, such as the Normal or Poisson regression, are not suitable for this type of response variable due to the binary nature of Y.
- When Y is binary, the expected value $\mathbf{E}(Y)$ corresponds to the probability $P(Y = 1)$.
- The model will establish a relationship between $\mathbf{E}(Y)$ and a predictor X, or a set of predictors $X_1, X_2, \ldots, X_p$, to capture the underlying dependency structure.

Review: Odds and Probability

- Recall that for an event with probability π, the odds of the event are defined as $\frac{\pi}{1-\pi}$.
- Since the probability π ranges from 0 to 1, the odds span values from 0 to ∞.
- The odds are:
  - Less than 1 if and only if $\pi < 0.5$.
  - Equal to 1 if and only if $\pi = 0.5$.
  - Greater than 1 if and only if $\pi > 0.5$.

Real Data Example: Logistic Regression Model

- Consider a dataset of senior citizens where two variables are measured:
  - A binary response variable Y.
  - An (approximately) continuous predictor variable X.
- The response variable Y indicates senility status:
  - $Y = 0$: No senility present.
  - $Y = 1$: Senility present.
- The predictor variable X represents the individual's score on a subset of the Wechsler Adult Intelligence Scale (WAIS) exam.

Real Data Example: Logistic Regression Model

- Recall that for a binary response Y_i, the expected value is $\mathbf{E}(Y_i) = P(Y_i = 1)$.
- We model $\mathbf{E}(Y_i) = \pi_i$ as a function of X_i, the WAIS score for the individual.
- The mean response given the predictors follows:

$$Y_i \mid \beta_0, \beta_1 \overset{\text{ind}}{\sim} \text{Bernoulli}(\pi_i), \quad \text{where } \log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \beta_1 X_i.$$

- The "linear predictor" $\beta_0 + \beta_1 X_i$ is related to the log-odds that $Y_i = 1$.
- The model equation can also be expressed in terms of the odds or probability:

$$\frac{\pi_i}{1-\pi_i} = e^{\beta_0 + \beta_1 X_i} \quad \text{and} \quad \pi_i = \frac{e^{\beta_0 + \beta_1 X_i}}{1 + e^{\beta_0 + \beta_1 X_i}}.$$

General Form of the Logistic Regression Model

- For a logistic regression model with multiple predictors, the log-odds is modeled as:

$$\log(\text{odds}) = \log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p.$$

- Interpretation of β_1:
  - Let odds_x be the odds that $Y = 1$ when $X_1 = x$, and let odds_{x+1} be the odds that $Y = 1$ when $X_1 = x + 1$ (a one-unit increase in X_1).
  - Holding all other predictors $X_2, \ldots, X_p$ constant:
    - β_1 represents the expected change in log-odds:

$$\beta_1 = \log(\text{odds}_{x+1}) - \log(\text{odds}_x).$$

    - e^{β_1} represents the expected multiplicative change in odds:

$$e^{\beta_1} = \frac{\text{odds}_{x+1}}{\text{odds}_x}.$$

Priors in the Logistic Regression Model

- To perform Bayesian logistic regression, we must specify priors on the coefficients $\beta_0, \beta_1, \ldots, \beta_p$.
- A common choice is to use normal priors for these coefficients:

$$\beta_j \sim N(\mu_j, \sigma_j^2), \quad j = 0, 1, \ldots, p.$$

- For objective Bayesian analysis, we can set prior means $\mu_j = 0$ for all coefficients.
- Posterior estimation involves sampling techniques, such as:
    - Implementing the Metropolis-Hastings algorithm manually.
    - Using high-level tools like `stan_glm` in the `rstanarm` package.
- **Example:** Applying "noninformative" priors in `R` to the WAIS senility dataset demonstrates this approach.

Specifying Subjective Priors in the Logistic Regression Model

- A structured process for eliciting prior information can be particularly effective when using `stan_glm`.
- In `stan_glm`, the prior is placed on the *centered* intercept (β_0^*), distinct from the β_0 in the model.
- **Example:** Prior elicitation for senility probability.
    - Assume a "typical" subject has a probability of senility between 0.2 and 0.6.
    - Corresponding log-odds range:

$$\log\left(\frac{0.2}{0.8}\right) = -1.4 \quad \text{to} \quad \log\left(\frac{0.6}{0.4}\right) = 0.4.$$

    - Set the prior mean for the **CENTERED** β_0^* to the midpoint:

$$\mu_{\beta_0^*} = \frac{-1.4 + 0.4}{2} = -0.5.$$

    - Set the prior standard deviation to half the range:

$$\sigma_{\beta_0^*} = \frac{0.4 - (-1.4)}{2} = 0.45.$$

More on Specifying Subjective Priors in the Logistic Regression Model

Example: Specifying the prior mean and standard deviation for β_1.

- **Belief:** For a one-unit increase in WAIS score, the odds of senility are expected to fall between 0.5 and 1 (i.e., reduced to half or remain the same).
- Corresponding range for β_1 (log-odds):

$$\beta_1 \in [\log(0.5), \log(1)] = [-0.69, 0]\,.$$

- **Prior elicitation:**
  - Set the prior mean for β_1 to the midpoint of the range:

$$\mu_{\beta_1} = \frac{-0.69 + 0}{2} = -0.35.$$

  - Set the prior standard deviation to half the range:

$$\sigma_{\beta_1} = \frac{0 - (-0.69)}{2} = 0.175.$$

- This prior reflects expert knowledge about the expected effect of WAIS scores on senility odds.

Fitting the Logistic Regression Model

- Priors can be specified, and the `stan_glm` function in the `rstanarm` package automates the Metropolis-Hastings algorithm for posterior sampling.
- Perform standard MCMC diagnostics to ensure model convergence and reliability:
  - Check trace plots, R-hat statistics, and effective sample sizes.
  - Take remedial actions if diagnostics indicate convergence issues.
- Summaries of the posterior distributions for model coefficients can be obtained using:
  - The `summary()` function for detailed statistical summaries.
  - The `tidy()` function for a cleaner, formatted output.
- Refer to `R` examples for implementing and fitting the logistic regression model using Bayesian methods.

Interpretations of Estimated Parameters

- Posterior estimate for β_1:
  - Approximate value: $\widehat{\beta}_1 \approx -0.3$.
  - The exact estimate may vary slightly depending on the choice of priors and the specifics of the MCMC run.
- Interpretation of β_1:
  - The odds of senility decrease by a factor of $e^{-0.3} \approx 0.74$ for each one-point increase in WAIS score.
  - This corresponds to a 26% reduction in the odds of senility per unit increase in WAIS score.
- Credible interval for β_1:
  - 95% credible interval: $(-0.498, -0.142)$.
  - High posterior probability exists that higher WAIS scores are associated with lower odds of senility.

Outline

Bayesian Logistic Regression

Prediction and Classification

Evaluating Model Performance

Bayesian Logistic Regression with Multiple Predictors

Using the Logistic Regression Model for Prediction

- A primary application of the logistic regression model is predicting the binary response Y for new observations.
- Example:
  - For a new senior citizen with a WAIS score of $X = 10$, predict whether the individual is senile.
- Prediction approach:
  - Plug $X = 10$ into the estimated logistic regression model to compute:

    $$\widehat{\mathbf{E}}(Y \mid X = 10),$$

    which is the estimated probability $\widehat{\pi}$ that the person is senile.
  - Decision rule:
    - If $\widehat{\pi} > 0.5$, predict $Y = 1$ (senile).
    - If $\widehat{\pi} \leq 0.5$, predict $Y = 0$ (not senile).
- Note: A cutoff $c \neq 0.5$ can be used to adjust the sensitivity and specificity of the predictions.

Defining a Classification Rule

- Logistic regression can be used to classify an individual into one of two groups: $Y = 0$ or $Y = 1$.
- Classification rule:
  - For a given predictor value x (or a set of predictors $x_1, x_2, \ldots, x_p$), generate a large number of posterior predictions for Y.
  - Let p denote the proportion of posterior predictions where $Y = 1$.
  - Select a classification cutoff value $c \in [0, 1]$.
  - Decision rule:
    - If $p \geq c$, classify the individual into the $Y = 1$ group.
    - If $p < c$, classify the individual into the $Y = 0$ group.
- This approach allows for flexible classification thresholds based on the specific context or desired trade-off between sensitivity and specificity.

Choice of Classification Cutoff Value

- The default classification cutoff value is $c = 0.5$, which is commonly used.
- However, in certain scenarios, a different cutoff value may be more appropriate:
  - Especially when the cost of one type of misclassification error significantly outweighs the cost of the other.
- Example from the book:
  - $Y = 1$: Predicting rain (carry an umbrella).
  - $Y = 0$: Predicting no rain (no umbrella).
  - Decision trade-off:
    - Is it worse to carry an umbrella unnecessarily or to forgo the umbrella and get wet?
  - To minimize the risk of getting wet, we might choose a smaller cutoff, such as $c = 0.25$, thereby predicting rain more often and playing it safe.
- Adjusting c allows for flexibility to match the classification strategy to the specific context and error costs.

Outline

Bayesian Logistic Regression

Prediction and Classification

Evaluating Model Performance

Bayesian Logistic Regression with Multiple Predictors

Assessing Model Quality

- The posterior predictive distribution can be used to evaluate model quality.
- Approach:
    - Use the `pp_check` function as a shortcut to generate numerous posterior-simulated datasets.
    - For each simulated dataset, calculate the count of $Y = 1$ values.
    - Visualize these counts using a histogram.
- Model evaluation:
    - Compare the actual count of $Y = 1$ values from the observed data to the distribution of simulated counts.
    - If the observed count falls near the center of the simulated distribution, it indicates that the model fits well.
- **Example:** See `R` implementation for the WAIS dataset.

Measuring Classification Accuracy

- Classification accuracy evaluates the performance of the logistic regression model in correctly classifying binary observations.
- A common approach is to use a **confusion matrix**: Compare the actual binary values (Y) with the predicted binary values ($\widehat{Y}$) based on the chosen classification rule.
- For a sample of n individuals:
  - Let Y_i denote the actual binary outcome for observation i, where $i = 1, \ldots, n$.
  - Compute $\widehat{Y}_i$, the predicted binary outcome, using the fitted logistic regression model and the chosen classification cutoff.
- The confusion matrix summarizes the counts of:
  - True Positives (correctly predicted $Y = 1,\ \widehat{Y} = 1$)
  - True Negatives (correctly predicted $Y = 0,\ \widehat{Y} = 0$).
  - False Positives (incorrectly predicted $Y = 0,\ \widehat{Y} = 1$).
  - False Negatives (incorrectly predicted $Y = 1,\ \widehat{Y} = 0$).

Confusion Matrix

- The confusion matrix summarizes the classification results in a 2×2 format with entries a, b, c, and d:

	$\widehat{Y} = 0$	$\widehat{Y} = 1$
$Y = 0$	a	b
$Y = 1$	c	d

- **Definitions:**
  - a: True Negatives (correctly predicted $Y = 0,\ \widehat{Y} = 0$).
  - b: False Positives (incorrectly predicted $Y = 0,\ \widehat{Y} = 1$).
  - c: False Negatives (incorrectly predicted $Y = 1,\ \widehat{Y} = 0$).
  - d: True Positives (correctly predicted $Y = 1,\ \widehat{Y} = 1$).

Confusion Matrix

- This framework provides metrics such as accuracy, precision, recall, and $F1$-score to assess classification performance.
- Metrics for model performance:
  - **Overall Accuracy**: Proportion of all observations correctly classified:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d}.$$

  - **Misclassification Rate**: Proportion of incorrectly classified observations:

$$\text{Misclassification Rate} = 1 - \text{Accuracy} = \frac{b+c}{a+b+c+d}.$$

Sensitivity and Specificity

- **Sensitivity** (True Positive Rate): Proportion of $Y = 1$ observations correctly classified.

$$\text{Sensitivity} = \frac{d}{c + d}.$$

- **Specificity** (True Negative Rate): Proportion of $Y = 0$ observations correctly classified.

$$\text{Specificity} = \frac{a}{a + b}.$$

- Interpretation:
  - **Sensitivity** measures how well the model identifies true positives.
  - **Specificity** measures how well the model identifies true negatives.
- These metrics are crucial for evaluating model performance, especially when the costs of false positives and false negatives differ.

Aims of Sensitivity and Specificity

- Ideally, both **sensitivity** and **specificity** should be high to ensure robust model performance.
- However, practical applications often dictate prioritizing one over the other.
- **Example:** Medical testing for a potentially deadly disease (e.g., breast cancer).
  - **High Sensitivity**:
    - Ensures that most true cases of the disease ($Y = 1$) are detected.
    - Reduces the risk of a true cancer going undiagnosed, preventing untreated conditions.
  - **Lower Specificity**:
    - May result in some healthy individuals ($Y = 0$) being misclassified as sick.
    - This could lead to wasted time and resources but does not carry deadly consequences.
- In this context, high sensitivity is often more critical to minimize life-threatening errors.

Tuning the Classification Rule Based on Sensitivity and Specificity

- The classification rule is determined by the cutoff value c.
- To optimize the model's performance:
  - Experiment with various values of c.
  - For each c, compute the **in-sample** sensitivity and specificity using the resulting confusion matrix.
  - Alternatively, use **cross-validation** to estimate sensitivity and specificity for **out-of-sample** predictions.
- Trade-off between sensitivity and specificity:
  - **Lower** c: Increases sensitivity but decreases specificity.
  - **Higher** c: Increases specificity but decreases sensitivity.
- This trade-off should be considered in the context of the application to balance the costs of false positives and false negatives.

Outline

Bayesian Logistic Regression

Prediction and Classification

Evaluating Model Performance

Bayesian Logistic Regression with Multiple Predictors

Bayesian Logistic Regression with Multiple Predictors

- The logistic regression model can be extended to include multiple predictors $X_1, X_2, \ldots, X_p$.
- **Example:** Predicting whether it rains tomorrow in Perth, Australia (Y is binary).
  - **Predictors:**
    - X_1: Humidity at 9 a.m. today.
    - X_2: Humidity at 3 p.m. today.
    - X_3: Whether it rains today (binary).
- Model equation for the mean response:

$$\pi_i = \mathbf{E}(Y_i \mid \mathbf{x}) = \frac{\exp(\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \beta_3 X_{i3})}{1 + \exp(\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \beta_3 X_{i3})}.$$

- **Bayesian framework:**
  - Priors are specified for $\beta_0, \beta_1, \beta_2, \beta_3$.
  - Posterior inference is conducted to estimate parameters and make predictions.

Fitting Bayesian Multiple Logistic Regression

- **Priors:** Normal priors for the coefficients $\beta_0, \beta_1, \beta_2, \beta_3$ are specified as usual.
- Posterior simulation conducted via:
  - Direct Metropolis-Hastings.
  - Automated tools like `stan_glm` from the `rstanarm` package.
- Example of estimated coefficients:
  $\widehat{\beta}_1 = -0.007$, $\widehat{\beta}_2 = 0.08$, $\widehat{\beta}_3 = 1.15$. (Estimates may vary depending on prior specifications and MCMC runs.)
- **Inference:**
  - The 95% credible interval for β_1 includes 0 suggesting that "humidity at 9 a.m. today" may not be necessary as a predictor.
  - Strong association between predictors (X_1, X_2, X_3) might explain why not all predictors are required.
- **Interpretation:** Model refinement can be based on credible intervals and collinearity considerations.

Model Selection in Bayesian Multiple Logistic Regression

- Model selection involves comparing different sets of predictors using standard criteria:
  - **Cross-Validation (CV) Accuracy**: Measures prediction performance on held-out data.
  - **Expected Log Predictive Density (ELPD)**: Reflects the model's fit to future data.
  - **Bayesian Information Criterion (BIC)**: Balances model fit and complexity.

Model Selection in Bayesian Multiple Logistic Regression

- **Example:** Predicting rain with multiple predictors.
  - Compare:
    - Model with 3 predictors (X_1, X_2, X_3).
    - Model with only X_1.
  - **Results:**
    - The 3-predictor model shows better CV accuracy, higher ELPD, and lower BIC.
    - Therefore, the 3-predictor model is preferred over the single-predictor model.
  - Refinement:
    - A model with only X_2 and X_3 (excluding X_1) slightly outperforms the 3-predictor model based on these criteria.
- **Conclusion:**
  - Model selection criteria help identify a balance between complexity and predictive performance.

```
    confusion <- matrix(0, nrow = 2, ncol = 2, dimnames = list(c("0", "1"), c("0", "1")))
  }

  # Compute accuracy
  accuracy <- sum(diag(confusion)) / sum(confusion)
  return(accuracy)
}
```

```
library(rstanarm)
library(tidyverse)

# Perform k-fold cross-validation
CV_acc_vals <- sapply(
  folds,
  function(test_ind) {
    train_ind <- setdiff(seq_len(nrow(wais_data)), test_ind)
    comp_CV_acc(train_ind, test_ind, wais_mod, wais_data, cutoff = 0.5)
  }
)
```

```
# Calculate mean cross-validated accuracy
CV_acc <- mean(CV_acc_vals)
cat("Cross-Validated Classification Accuracy:", CV_acc, "\n")
```

```
## Cross-Validated Classification Accuracy: 0.8266667
```

Bayesian Logistic Regression with Multiple Predictors

```
# Rain example from the book

# Load required libraries
library(rstanarm)      # For Bayesian logistic regression
library(tidyverse)     # For data manipulation and visualization
library(broom.mixed)   # For tidy summaries of Bayesian models
library(bayesrules)    # For Bayesian tools
library(caret)         # For stratified cross-validation

# Load and process the data
data(weather_perth)  # Assuming `weather_perth` is preloaded
weather <- weather_perth %>%
  select(day_of_year, raintomorrow, humidity9am, humidity3pm, raintoday)
```

Prior belief for logistic regression: On a "typical" day, the chance of rain is 20% (0.2). The prior mean on the CENTERED beta_0 (intercept) is log(0.2/(1 - 0.2)) = -1.4. A prior SD of 0.7 implies a 95% chance the log-odds are between -2.8 and 0. This corresponds to odds of 0.06 to 1, or probabilities between 0.057 and 0.5.

```
# Fit a Bayesian logistic regression model with multiple predictors
rain_stanglm2 <- stan_glm(
```

```
  raintomorrow ~ humidity9am + humidity3pm + raintoday,
  data = weather,
  family = binomial,                                # Logistic regression
  prior_intercept = normal(-1.4, 0.7),             # Prior for intercept
  prior = normal(0, 2.5, autoscale = TRUE),        # Weakly informative prior for coefficients
  chains = 4,                                      # Number of MCMC chains
  iter = 10000                                     # Number of iterations (post-warmup = 5000)
)

# Summarize posterior estimates with confidence intervals
rain_stanglm2_summ <- tidy(rain_stanglm2, effects = "fixed", conf.int = TRUE, conf.level = 0.95)
print(rain_stanglm2_summ)

# Model comparison: Fit a simpler model with a single predictor
rain_stanglm1 <- stan_glm(
  raintomorrow ~ humidity9am,
  data = weather,
  family = binomial,
  prior_intercept = normal(-1.4, 0.7),             # Same prior for intercept
  prior = normal(0.07, 0.035),                    # Prior for slope (adjusted based on context)
  chains = 4,
  iter = 10000,
  prior_PD = FALSE                                # Use posterior data
)
```

```
# Compare classification accuracy using k-fold cross-validation
# The book suggests c = 0.2 as a reasonable cutoff, but feel free to explore others
set.seed(123)  # For reproducibility

# Cross-validation for rain_stanglm1
CV_acc_1 <- classification_summary_cv(
  model = rain_stanglm1,
  data = weather,
  cutoff = 0.2,
  k = 10  # 10-fold cross-validation
)

# Cross-validation for rain_stanglm2
CV_acc_2 <- classification_summary_cv(
  model = rain_stanglm2,
  data = weather,
  cutoff = 0.2,
  k = 10
)

# Print cross-validated classification accuracy for both models
cat("Cross-Validated Accuracy for Model 1 (Single Predictor):\n")
```

```
## Cross-Validated Accuracy for Model 1 (Single Predictor):
```

```
CV_acc_1$cv
```

```
##   sensitivity specificity overall_accuracy
```

```
## 1   0.6353766   0.7156357            0.701
```

```r
cat("Cross-Validated Accuracy for Model 2 (Multiple Predictors):\n")
```

```
## Cross-Validated Accuracy for Model 2 (Multiple Predictors):
```

```r
CV_acc_2$cv
```

```
##   sensitivity specificity overall_accuracy
## 1   0.7555544   0.8143257            0.802
```

```r
# One approach to model selection:
# LOO for rain_stanglm1 (Single Predictor)
loo1 <- loo(rain_stanglm1)
cat("LOO Estimates for Model 1 (Single Predictor):\n")
```

```
## LOO Estimates for Model 1 (Single Predictor):
```

```r
print(loo1$estimates)
```

```
##             Estimate         SE
## elpd_loo -437.076458 19.0039708
## p_loo       2.217106  0.1798331
## looic     874.152915 38.0079416
```

```r
# LOO for rain_stanglm2 (Multiple Predictors)
loo2 <- loo(rain_stanglm2)
cat("LOO Estimates for Model 2 (Multiple Predictors):\n")
```

```
## LOO Estimates for Model 2 (Multiple Predictors):
```

```r
print(loo2$estimates)
```

```
##            Estimate         SE
## elpd_loo -356.91119 20.8083907
## p_loo       4.23658  0.3494051
## looic     713.82239 41.6167814
```

```r
# Comparing LOO-CV:
cat("\nModel Comparison using LOO:\n")
```

```
##
## Model Comparison using LOO:
```

```r
loo_comp <- loo_compare(loo1, loo2)
print(loo_comp)
```

```
##               elpd_diff se_diff
## rain_stanglm2   0.0       0.0
## rain_stanglm1 -80.2      13.5
```

```r
# Frequentist approach: Use Bayesian Information Criterion (BIC)
# BIC does not incorporate prior information, allowing direct comparison of models

# Fit frequentist logistic regression models
rain_glm1 <- glm(
  raintomorrow ~ humidity9am,
  data = weather,
  family = binomial(logit)
)

rain_glm2 <- glm(
  raintomorrow ~ humidity9am + humidity3pm + raintoday,
  data = weather,
  family = binomial(logit)
)

# Calculate BIC for both models
BIC1 <- BIC(rain_glm1)
BIC2 <- BIC(rain_glm2)

cat("\nBIC Comparison:\n")
```

```
##
## BIC Comparison:
```

```r
cat("Model 1 (Single Predictor): BIC =", BIC1, "\n")
```

```
## Model 1 (Single Predictor): BIC = 883.7219
```

```r
cat("Model 2 (Multiple Predictors): BIC =", BIC2, "\n")
```

```
## Model 2 (Multiple Predictors): BIC = 733.167
```

```r
# Interpretation:
# Lower BIC indicates better model fit while penalizing for model complexity.
# Compare BIC values to decide which model is more appropriate.
```

Bayesian Logistic Regression Model with Simplified Predictors — only humidity3pm & raintoday as Predictors

```r
rain_stanglm_simp <- stan_glm(
  raintomorrow ~ humidity3pm + raintoday,
  data = weather,
  family = binomial,                              # Logistic regression
  prior_intercept = normal(-1.4, 0.7),           # Prior for intercept
  prior = normal(0, 2.5, autoscale = TRUE),      # Weakly informative priors
  chains = 1,                                    # Single MCMC chain for simplicity
  iter = 10000                                   # Total iterations (post-warmup = 5000)
)
```

```r
# Cross-validated classification accuracy (cutoff = 0.2, 10-fold CV)
CV_acc_simp <- classification_summary_cv(
  model = rain_stanglm_simp,
  data = weather,
  cutoff = 0.2,
  k = 10
)

# Print cross-validated accuracy
cat("Cross-Validated Classification Accuracy (Simplified Model):\n")
```

```
## Cross-Validated Classification Accuracy (Simplified Model):
```

```r
print(CV_acc_simp$cv)
```

```
##   sensitivity specificity overall_accuracy
## 1    0.759782   0.8097739              0.8
```

```r
# Evaluate model using Leave-One-Out Cross-Validation (LOO)
loo_simp <- loo(rain_stanglm_simp)
cat("\nLOO Estimates for Simplified Model:\n")
```

```
##
## LOO Estimates for Simplified Model:
```

```r
print(loo_simp$estimates)
```

```
##             Estimate         SE
## elpd_loo -356.229479 20.7998838
## p_loo       3.122271  0.2752114
## looic     712.458958 41.5997676
```

```r
# Frequentist logistic regression model with the same predictors
rain_glm_simp <- glm(
  raintomorrow ~ humidity3pm + raintoday,
  data = weather,
  family = binomial(logit)
)

# Calculate BIC for the simplified model
BIC_simp <- BIC(rain_glm_simp)
cat("\nBIC for Simplified Model (Frequentist):\n", BIC_simp, "\n")
```

```
##
## BIC for Simplified Model (Frequentist):
##  727.1193
```

```r
# Comparison Notes:
# - LOO: Lower `elpd_loo` values indicate a better Bayesian model fit.
# - BIC: Lower BIC indicates a better frequentist model fit while penalizing complexity.
```

Multiple Logistic Regression Model using Base R

```r
# Load required package
library(mvtnorm)  # For multivariate normal distributions

# Extract variables from the weather dataset
rain_tom <- weather$raintomorrow
humid9am <- weather$humidity9am
humid3pm <- weather$humidity3pm
rain_today <- weather$raintoday

# Convert variables to numeric for calculations
rain_tom <- as.numeric(rain_tom) - 1  # Convert rain_tom to 0's and 1's
rain_today <- as.numeric(rain_today)    # Ensure rain_today is numeric

# Construct the design matrix (X) with an intercept
X <- cbind(rep(1, times = length(humid9am)), humid9am, humid3pm, rain_today)

# Prior specifications
beta_pri_mean <- c(0, -0.35, -0.2, 0.5)  # Prior means for beta parameters
# Prior reflects:
# - No specific belief about beta_0 (intercept)
# - Moderate beliefs about the slopes for humid9am, humid3pm, and rain_today
beta_pri_cov <- diag(c(100, 40, 40, 40))  # Prior covariance matrix

# Scaling factor for proposal covariance matrix (can be tuned for MCMC acceptance rates)
k <- 1

# Frequentist Logistic Regression Model (using glm)
log_reg_out <- glm(rain_tom ~ humid9am + humid3pm + rain_today, family = binomial(logit))

# Summary of the logistic regression model
cat("\nSummary of the Logistic Regression Model:\n")
```

```
##
## Summary of the Logistic Regression Model:
```

```r
summary(log_reg_out)
```

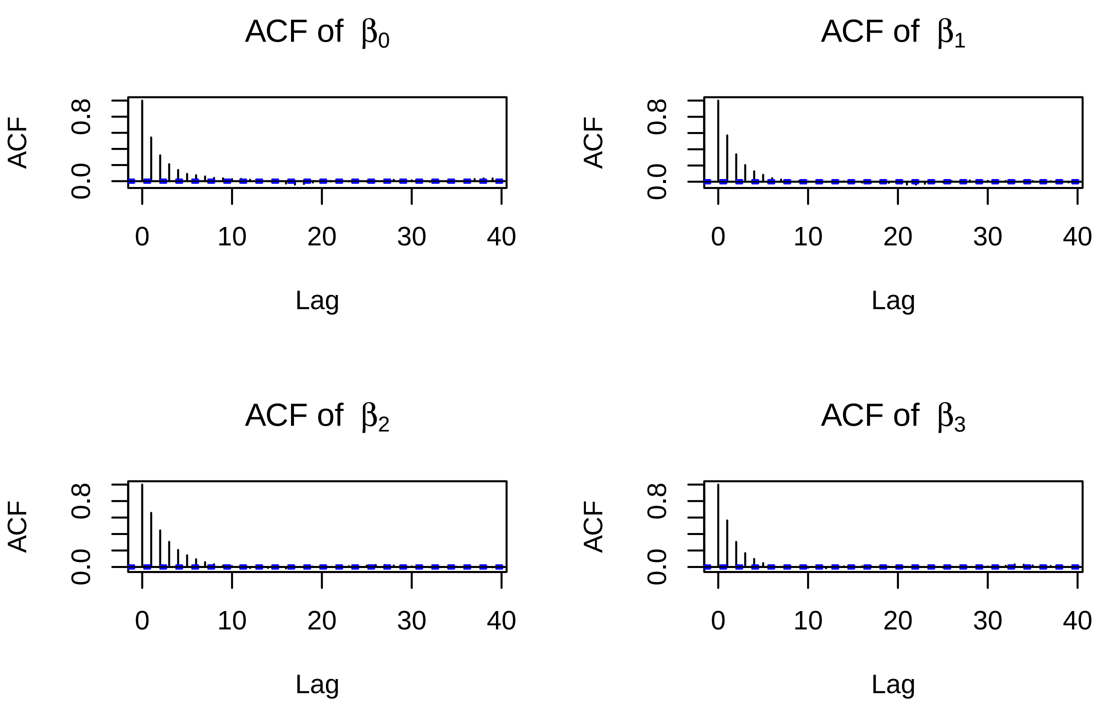
```
##
## Call:
## glm(formula = rain_tom ~ humid9am + humid3pm + rain_today, family = binomial(logit))
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.639653   0.488528 -13.591  < 2e-16 ***
## humid9am    -0.006850   0.007407  -0.925    0.355
## humid3pm     0.079831   0.008576   9.309  < 2e-16 ***
## rain_today   1.155394   0.216950   5.326 1.01e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 960.74  on 999  degrees of freedom
## Residual deviance: 705.54  on 996  degrees of freedom
## AIC: 713.54
##
## Number of Fisher Scoring iterations: 5
```

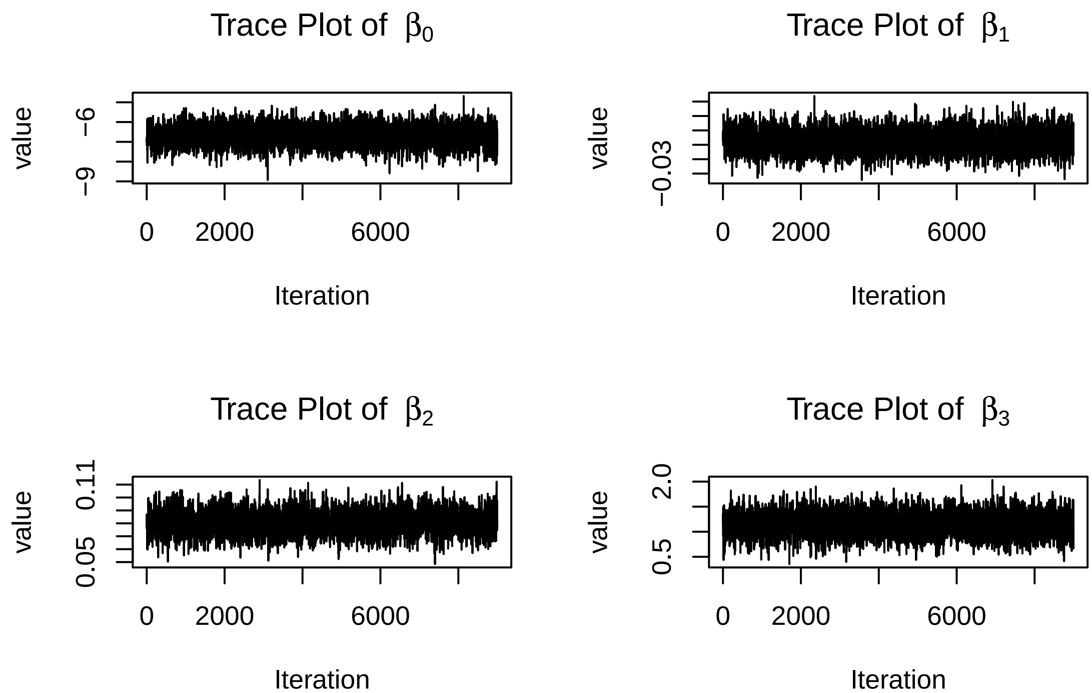
```r
# Set up the proposal covariance matrix
pro_cov_mat <- k * solve(t(X) %*% diag(fitted(log_reg_out)) %*% X)
# Alternative option for proposal covariance matrix:
# pro_cov_mat <- k * diag(ncol(X))

# Initialize MCMC parameters
betas_curr <- beta_pri_mean                # Initial parameter estimates
V <- pro_cov_mat                     # Proposal covariance matrix
mu <- beta_pri_mean                        # Prior mean vector
Sig_inv <- solve(beta_pri_cov)           # Inverse of prior covariance matrix

# MCMC settings
j <- 0                                      # Counter for accepted proposals
burn <- 1000                               # Number of burn-in iterations
Niter <- 50000                             # Total number of iterations
mcmc_res <- matrix(0, nrow = Niter, ncol = length(beta_pri_mean))  # Storage for MCMC samples

# MCMC sampling loop
for (i in 1:Niter) {
  # Generate candidate beta from proposal distribution
  betas_pro <- rmvnorm(1, betas_curr, V)
  betas_pro <- as.vector(betas_pro)

  # Calculate the Metropolis ratio
  log_ratio <- (
    -k * sum(log(1 + exp(X %*% betas_pro))) + sum(rain_tom * X %*% betas_pro) -
      0.5 * t(betas_pro - mu) %*% Sig_inv %*% (betas_pro - mu)
  ) - (
    -k * sum(log(1 + exp(X %*% betas_curr))) + sum(rain_tom * X %*% betas_curr) -
      0.5 * t(betas_curr - mu) %*% Sig_inv %*% (betas_curr - mu)
  )

  # Accept/reject step
  if (runif(1) < exp(log_ratio)) {
    betas_curr <- betas_pro  # Accept candidate
    j <- j + 1               # Increment acceptance counter
  }

  # Save the current beta values
  mcmc_res[i, ] <- betas_curr
}

# Calculate acceptance rate
acc_rate <- j / Niter
cat("Acceptance Rate:", acc_rate, "\n")
```

```
## Acceptance Rate: 0.49968
```

```r
# Thinning (everrain_tom 5th value)
thin <- 5
beta_vals_thin <- mcmc_res[seq(1, Niter, by = thin), ]

# Remove burn-in samples
beta_vals_thin_b <- beta_vals_thin[-(1:burn), ]

# Diagnostic plots: Autocorrelation for each parameter
par(mfrow = c(2, 2))  # Set up a 2humid3pm plotting grid
for (i in 1:ncol(beta_vals_thin_b)) {
  acf(beta_vals_thin_b[, i], main = bquote("ACF of " ~ beta[.(i - 1)]) )
}
```




```r
par(mfrow = c(1, 1))  # Reset plotting grid

# Diagnostic plots: Trace plots for each parameter
par(mfrow = c(2, 2))  # Set up a 2humid3pm plotting grid
for (i in 1:ncol(beta_vals_thin_b)) {
  plot(beta_vals_thin_b[, i], type = 'l',
       main = bquote("Trace Plot of " ~ beta[.(i - 1)]),
       xlab = "Iteration", ylab = "value")
}
```




```r
par(mfrow = c(1, 1))  # Reset plotting grid

# Posterior summaries
post_meds <- apply(beta_vals_thin_b, 2, median)      # Posterior medians
post_low <- apply(beta_vals_thin_b, 2, quantile, probs = 0.025)  # 2.5% quantile
post_up <- apply(beta_vals_thin_b, 2, quantile, probs = 0.975)  # 97.5% quantile

# Combine posterior summaries into a tidrain_tom data frame
names_preds <- c("humid9am", "humid3pm", "rain_today")  # Predictor names
beta_post_summ <- data.frame(
  `0.025 Quantile` = post_low,
  `0.5 Quantile` = post_meds,
  `0.975 Quantile` = post_up,
  row.names = c("Intercept", names_preds)
)

# Print the posterior Summary
cat("Posterior Summary:\n")
```

```
## Posterior Summary:
```

```r
print(beta_post_summ)
```

```
##              X0.025.Quantile X0.5.Quantile X0.975.Quantile
## Intercept        -7.66201004   -6.66399335    -5.758651522
## humid9am         -0.02158058   -0.00704039     0.007749144
## humid3pm          0.06356771    0.08053736     0.098286821
## rain_today        0.72117530    1.15555140     1.583379383
```