

Slide 13 - R Output

Elvan Ceyhan

11/17/2024

```
#Set Working Directory to Source File Location  
library("rstudioapi") # Load rstudioapi package  
#setwd(dirname(getActiveDocumentContext()$path)) # Set working directory to source file location  
#getwd()
```

Some required packages:

```
# Load packages  
library(rstan)  
library(mvtnorm) # For multivariate normal sampling  
library(ggplot2)
```

Binary Response Model

This code implements logistic regression with a Bayesian perspective, adapted from an example at: <https://brunaw.com/phd/mcmc/report.pdf>

We define binary response variable:

- $Y = 0$: Individual has no senility
- $Y = 1$: Individual has senility present
- $X = wais$: Scores on a subset of the Wechsler Adult Intelligence Scale (WAIS)

```
# Load required library  
library(mvtnorm)  
  
# Response variable (y=senility): 14 with senility (1), 40 without senility (0)  
senility <- c(rep(1, times = 14), rep(0, times = 40))  
  
# Predictor variable (x): WAIS scores  
wais <- c(9, 13, 6, 8, 10, 4, 14, 8, 11, 7, 9, 7, 5, 14, 13, 16, 10, 12, 11, 14,  
          15, 18, 7, 16, 9, 9, 11, 13, 15, 13, 10, 11, 6, 17, 14, 19, 9, 11, 14,  
          10, 16, 10, 16, 14, 13, 13, 9, 15, 10, 11, 12, 4, 14, 20)  
  
# Fit logistic regression model using the logit link  
log_reg_out <- glm(senility ~ wais, family = binomial(logit))  
summary(log_reg_out)
```

##

```
## Call:
## glm(formula = senility ~ wais, family = binomial(logit))
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   2.4040     1.1918   2.017  0.04369 *
## wais          -0.3235     0.1140  -2.838  0.00453 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##    Null deviance: 61.806  on 53  degrees of freedom
## Residual deviance: 51.017  on 52  degrees of freedom
## AIC: 55.017
##
## Number of Fisher Scoring iterations: 5
```

```
# Prepare design matrix for Bayesian inference
# X includes an intercept (column of 1s) and the predictor variable (x1)
X <- cbind(1, wais) # Automatically handles intercept addition

# Define prior distribution parameters
beta_pri_mean <- c(0, -0.35) # Prior mean for beta (intercept, slope)
beta_pri_cov <- diag(c(100, 0.175^2)) # Prior covariance matrix

# Scale factor for the proposal covariance matrix
# Adjust `k` to control acceptance rate in MCMC
k <- 1

# Compute proposal covariance matrix based on Fisher Information
pro_cov_mat <- k * solve(t(X) %*% diag(fitted(log_reg_out)) %*% X)
# Uncomment the following if you want to use a diagonal proposal covariance matrix:
# pro_cov_mat <- k * diag(ncol(X))
```

MCMC for Logistic Regression Coefficients

```
# Initial setup
betas_curr <- beta_pri_mean # Initial values for beta
V <- pro_cov_mat # Proposal covariance matrix
mu <- beta_pri_mean # Mean of prior distribution
Sig_inv <- solve(beta_pri_cov) # Inverse of prior covariance matrix

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

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

```

# Calculate Metropolis ratio
log_r <- (
  -k * sum(log(1 + exp(X %>% betas_pro))) + sum(senility * X %>% betas_pro) -
    0.5 * t(betas_pro - mu) %>% Sig_inv %>% (betas_pro - mu)
) - (
  -k * sum(log(1 + exp(X %>% betas_curr))) + sum(senility * X %>% betas_curr) -
    0.5 * t(betas_curr - mu) %>% Sig_inv %>% (betas_curr - mu)
)

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

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

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

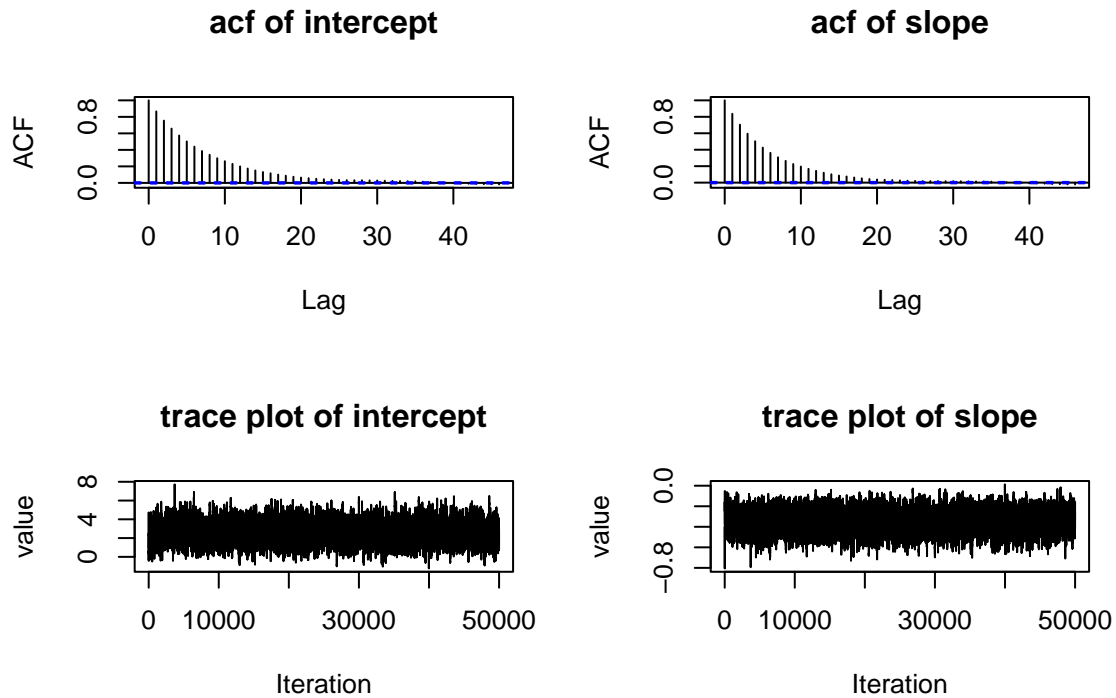
```

```
## Acceptance Rate: 0.61576
```

```

# Diagnostic plots of the original MCMC sample
par(mfrow = c(2, 2)) # 2x2 plotting grid
acf(beta_vals[, 1], main = "acf of intercept") # Autocorrelation for intercept
acf(beta_vals[, 2], main = "acf of slope") # Autocorrelation for slope
plot(beta_vals[, 1], type = 'l',
      main = "trace plot of intercept", xlab = "Iteration", ylab = "value")
plot(beta_vals[, 2], type = 'l',
      main = "trace plot of slope", xlab = "Iteration", ylab = "value")

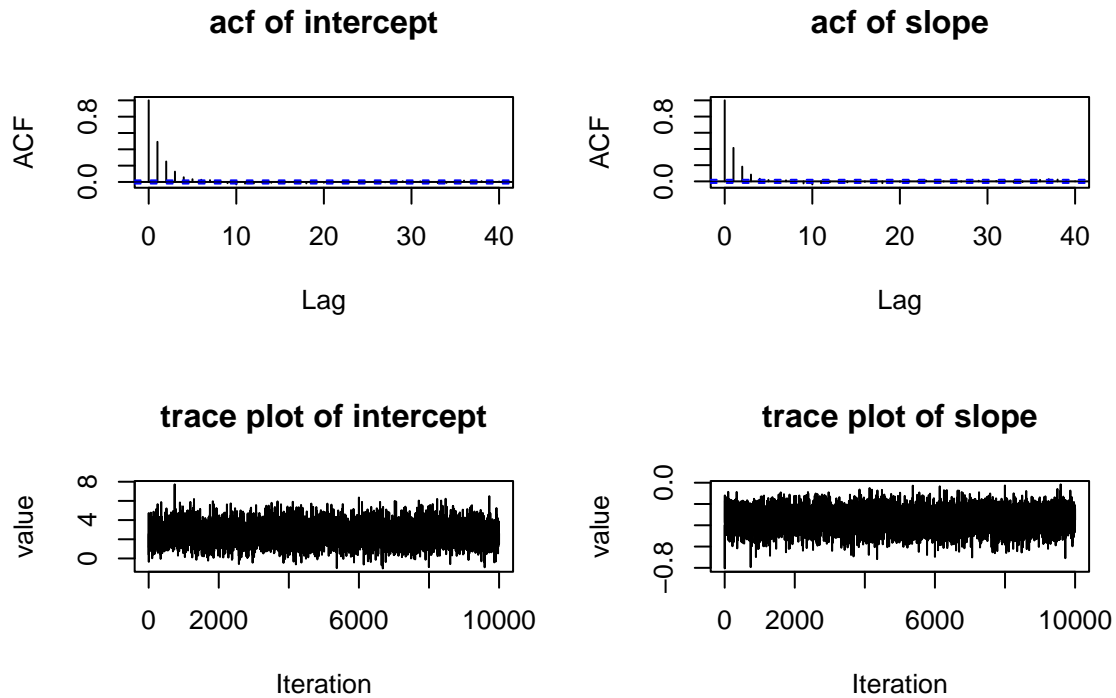
```



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

# Thinning results (every 5th value)
thin <- 5
beta_vals_thin <- beta_vals[seq(1, Niter, by = thin), ]

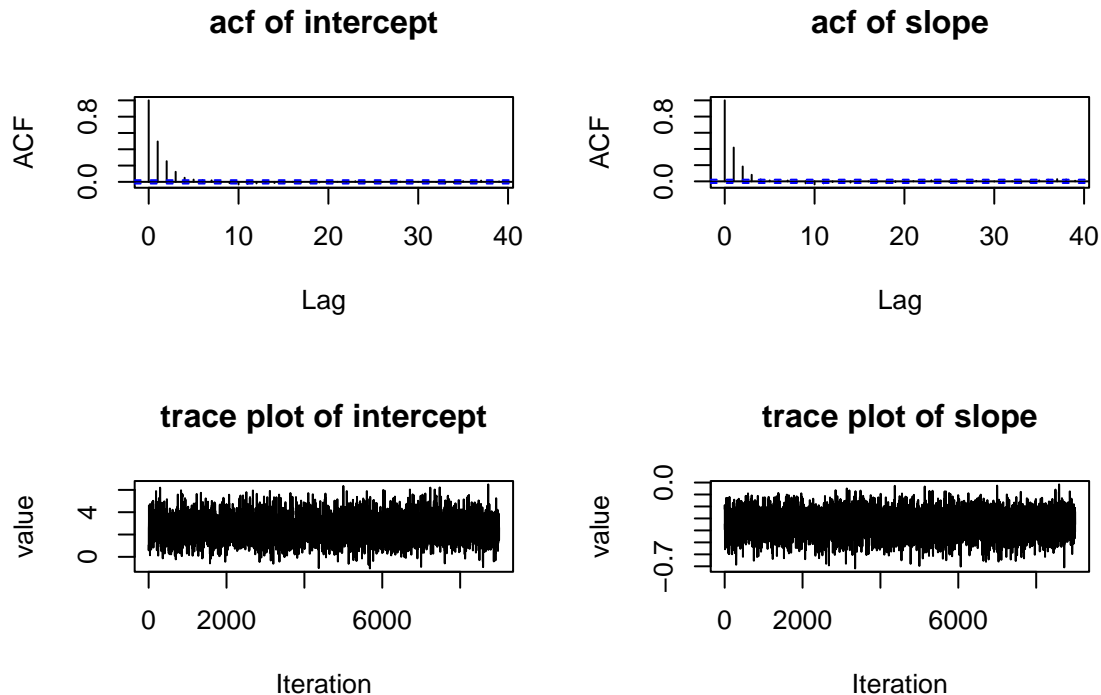
# Diagnostic plots of the original MCMC sample
par(mfrow = c(2, 2)) # 2x2 plotting grid
acf(beta_vals_thin[, 1], main = "acf of intercept") # Autocorrelation for intercept
acf(beta_vals_thin[, 2], main = "acf of slope") # Autocorrelation for slope
plot(beta_vals_thin[, 1], type = 'l',
     main = "trace plot of intercept", xlab = "Iteration", ylab = "value")
plot(beta_vals_thin[, 2], type = 'l',
     main = "trace plot of slope", xlab = "Iteration", ylab = "value")
```



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

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

# Diagnostic plots
par(mfrow = c(2, 2)) # 2x2 plotting grid
acf(beta_vals_thin_b[, 1], main = "acf of intercept") # Autocorrelation for intercept
acf(beta_vals_thin_b[, 2], main = "acf of slope") # Autocorrelation for slope
plot(beta_vals_thin_b[, 1], type = 'l',
     main = "trace plot of intercept", xlab = "Iteration", ylab = "value")
plot(beta_vals_thin_b[, 2], type = 'l',
     main = "trace plot of slope", xlab = "Iteration", ylab = "value")
```



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

# Posterior summary
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 summary into a data frame
names_preds <- c("wais") # Predictor names (intercept + predictors)
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 posterior summary
print(beta_post_summ)
```

```
##           X0.025.Quantile X0.5.Quantile X0.975.Quantile
## Intercept      0.5540770      2.5338294      4.7376081
## wais          -0.5406337     -0.3399216     -0.1607656
```

Bayesian Logistic Regression using rstanarm

```
# Load required packages
library(bayesrules) # For Bayesian methods and tools
library(rstanarm)  # For Bayesian regression models
```

```

library(bayesplot)      # For visualizing MCMC diagnostics
library(tidyverse)      # For data manipulation and visualization
library(tidybayes)      # For working with Bayesian models
library(broom.mixed)    # For generating tidy summaries of Bayesian models

# Create the dataset
wais_data <- data.frame(senility, wais)

# Fit the Bayesian logistic regression model
wais_mod <- stan_glm(
  senility ~ wais,
  data = wais_data,
  family = binomial,
  prior_intercept = normal(-0.5, 0.45),
  prior = normal(-0.35, 0.175),
  chains = 4,
  iter = 10000,
  prior_PD = FALSE
)

```

Logistic regression model
Prior for intercept
Prior for slope
Number of MCMC chains
Total iterations per chain (post-warmup = 5000)
Include likelihood in posterior estimation

Prior justification:

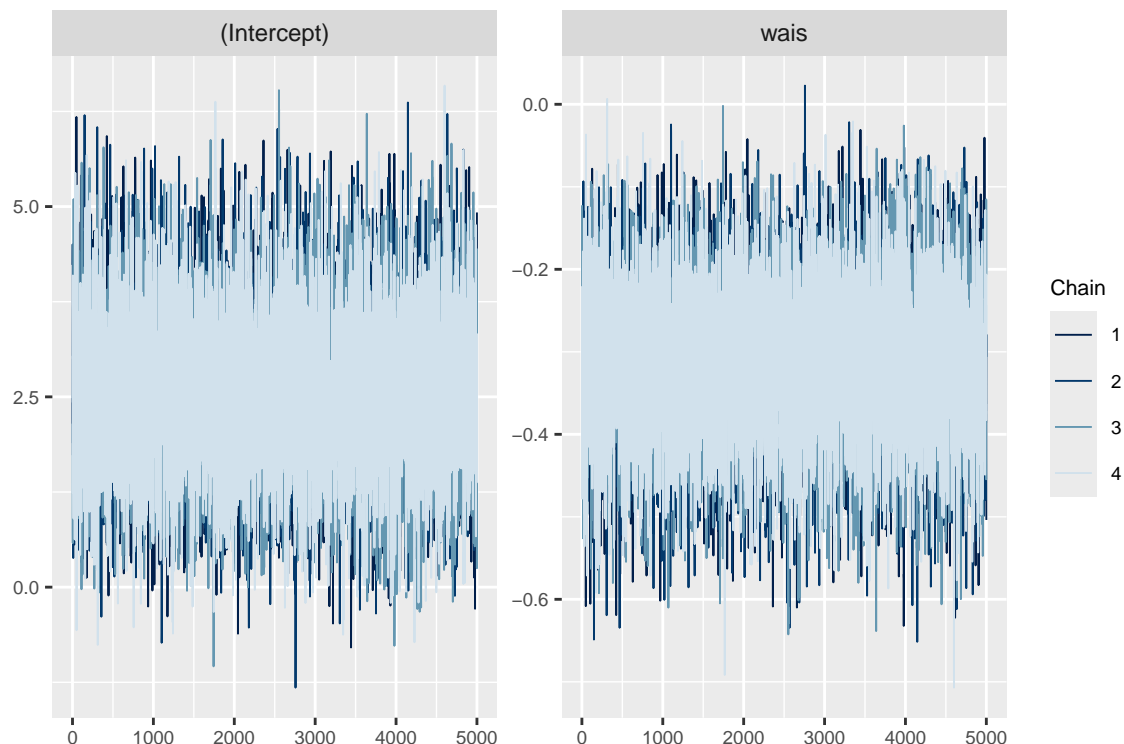
- Intercept prior: The log-odds of senility ($y = 1$) for a “typical” subject are between -1.4 and 0.4. Prior mean is -0.5, and prior SD is 0.45.
- Slope prior: For a one-unit increase in WAIS score, the odds of senility change by 0.5x to 1x. Prior mean is -0.35, and prior SD is 0.175.

```

# MCMC diagnostics
mcmc_trace(wais_mod)

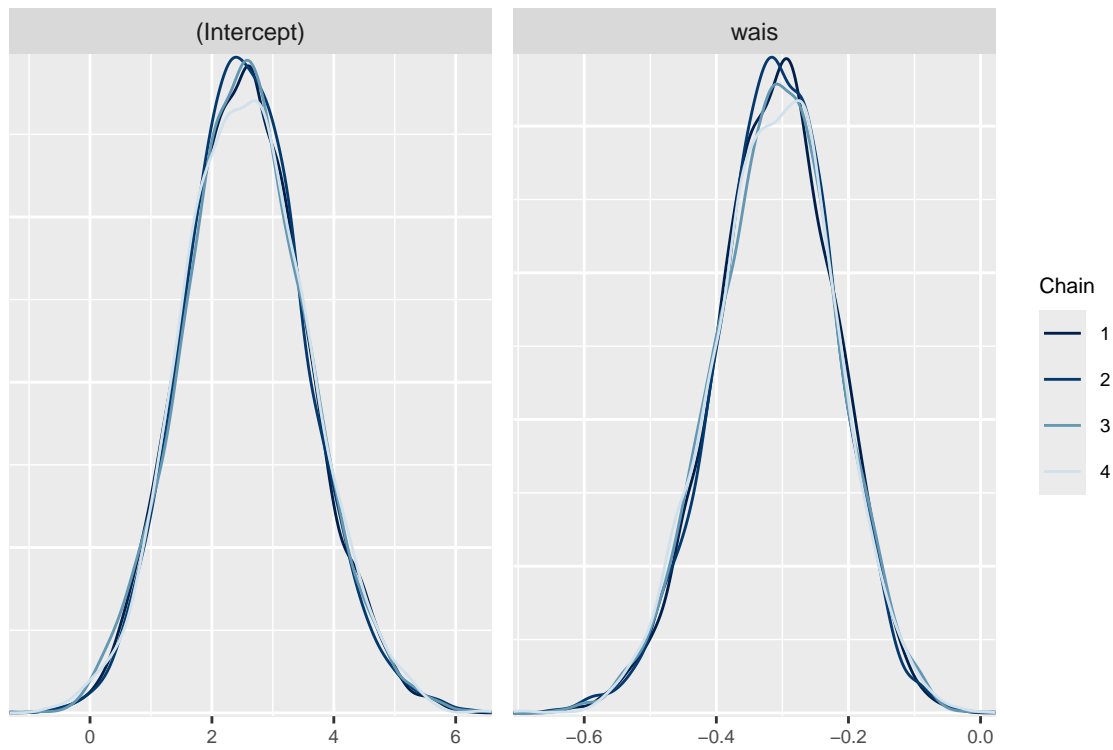
```

Trace plots for convergence diagnostics



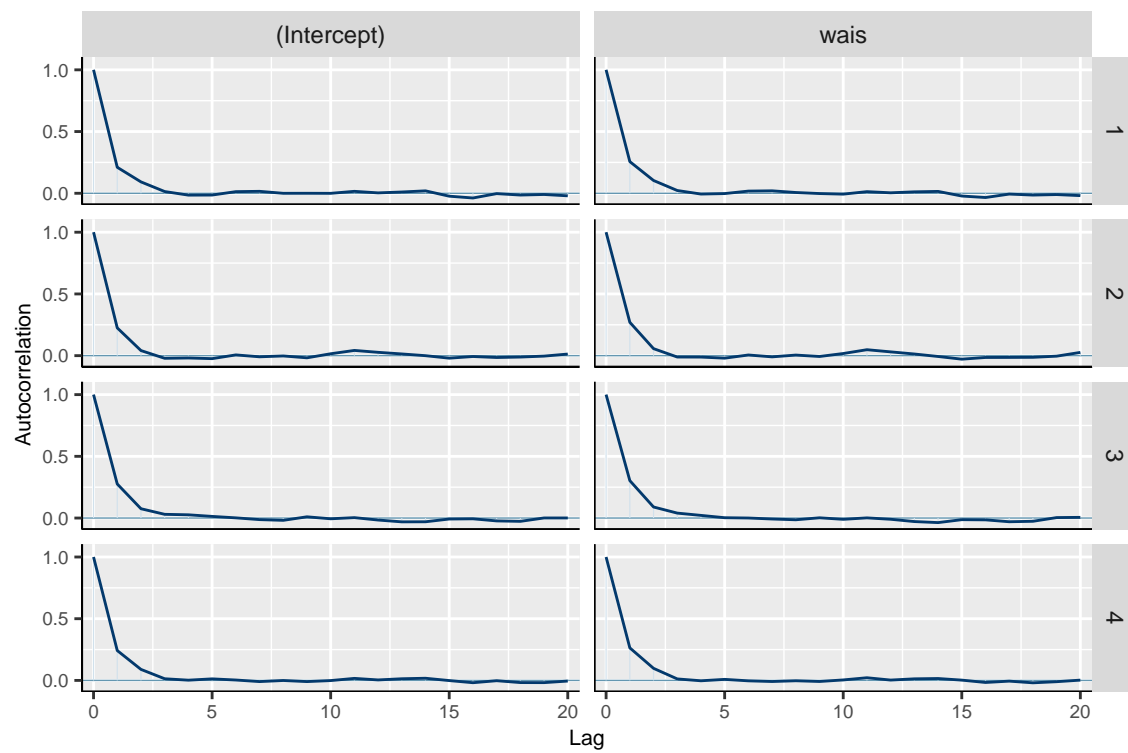
```
mcmc_dens_overlay(wais_mod)
```

```
# Overlay density plots for posterior distributions
```



```
mcmc_acf(wais_mod)
```

```
# Autocorrelation plots for MCMC chains
```

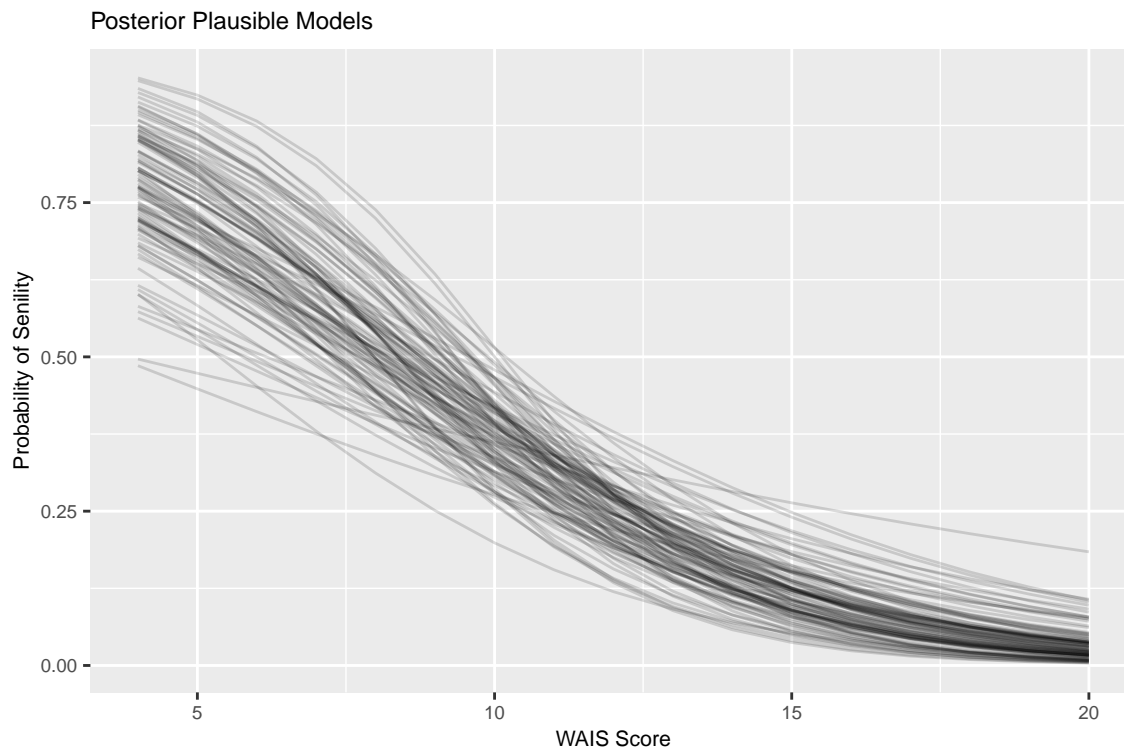



```

# Note: Slight autocorrelation is acceptable, but chains otherwise show good convergence.

# Visualizing posterior plausible models
wais_data %>%
  add_epred_draws(wais_mod, ndraws = 100) %>% # Use updated function for posterior draws
  ggplot(aes(x = wais, y = senility)) +
  geom_line(aes(y = .epred, group = .draw), alpha = 0.15) + # Use `.epred` for expected values
  labs(
    y = "Probability of Senility",
    x = "WAIS Score",
    title = "Posterior Plausible Models"
  )

```



```

# Summary of posterior estimates (point estimates and credible intervals)
post_summ <- tidy(wais_mod, conf.int = TRUE, conf.level = 0.95)
print(post_summ)

```

```

## # A tibble: 2 x 5
##   term      estimate std.error conf.low conf.high
##   <chr>      <dbl>     <dbl>   <dbl>   <dbl>
## 1 (Intercept)  2.54      1.01    0.605   4.66
## 2 wais       -0.308    0.0907 -0.497  -0.139

```

```

# Comparison with frequentist maximum likelihood estimates (MLEs)
log_reg_out <- glm(senility ~ wais, family = binomial(logit))
summary(log_reg_out)

```

```
##
```

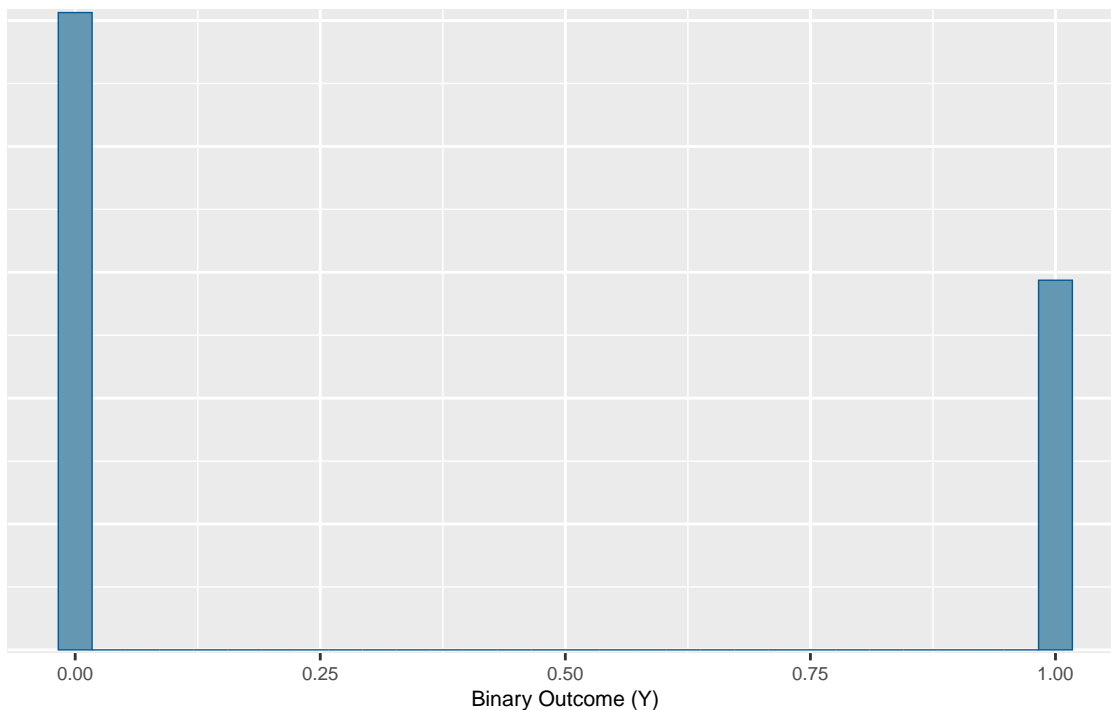
```
## Call:
## glm(formula = senility ~ wais, family = binomial(logit))
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   2.4040     1.1918   2.017  0.04369 *
## wais          -0.3235     0.1140  -2.838  0.00453 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##    Null deviance: 61.806  on 53  degrees of freedom
## Residual deviance: 51.017  on 52  degrees of freedom
## AIC: 55.017
##
## Number of Fisher Scoring iterations: 5

# Posterior predictions for a new observation (WAIS score = 10)
bin_pred <- posterior_predict(wais_mod, newdata = data.frame(wais = 10))

# Plot posterior predictive distribution
mcmc_hist(bin_pred) +
  labs(
    x = "Binary Outcome (Y)",
    title = "Posterior Predictive Distribution for WAIS Score = 10"
  )

## 'stat_bin()' using 'bins = 30'. Pick better value with 'binwidth'.
```

Posterior Predictive Distribution for WAIS Score = 10



```

# Summarize posterior predictions as a frequency table
bin_tab <- table(bin_pred)
print(bin_tab)

## bin_pred
##      0      1
## 12658  7342

# Compute posterior predictive mean (probability of senility for WAIS score = 10)
post_mean <- colMeans(bin_pred)
cat("Posterior Predictive Probability of Senility (WAIS = 10):", post_mean, "\n")

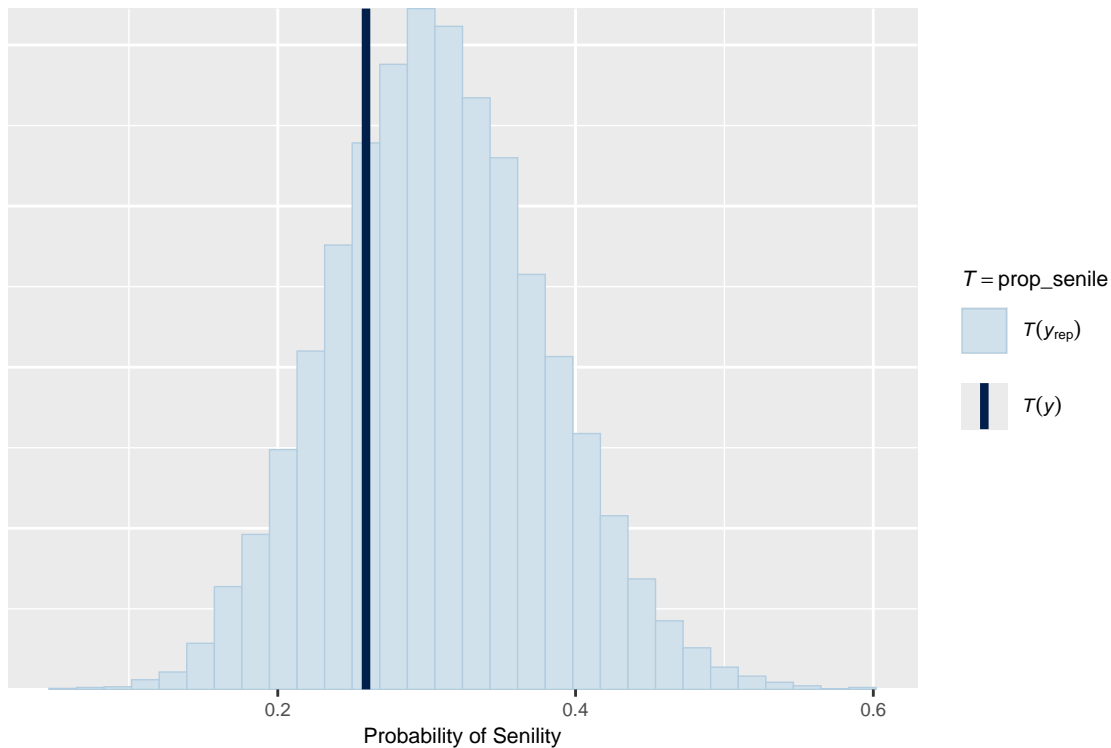
## Posterior Predictive Probability of Senility (WAIS = 10): 0.3671

# Check posterior predictive distribution
prop_senile <- function(x){mean(x == 1)}

pp_check(
  wais_mod,
  plotfun = "stat",
  stat = "prop_senile"
) + xlab("Probability of Senility")

## 'stat_bin()' using 'bins = 30'. Pick better value with 'binwidth'.

```



Note: Observed values being near the center of the posterior predictive distribution indicate good model fit.

```
# Classification accuracy and confusion matrix for cutoff = 0.5
classification_summary(
  model = wais_mod,
  data = wais_data,
  cutoff = 0.5
)
```

```
## $confusion_matrix
## y 0 1
## 0 37 3
## 1 7 7
##
## $accuracy_rates
##
## sensitivity      0.5000000
## specificity      0.9250000
## overall_accuracy 0.8148148
```

Cross-Validated Classification Accuracy Using k-fold Cross-Validation

```
# Load required library
library(caret)
library(rstanarm)
library(tidyverse)

# Create stratified folds
set.seed(123) # For reproducibility
folds <- createFolds(wais_data$senility, k = 10, list = TRUE, returnTrain = FALSE)

# Function to compute classification accuracy for each fold
comp_CV_acc <- function(train_ind, test_ind, model, data, cutoff) {
  train_data <- data[train_ind, ]
  test_data <- data[test_ind, ]

  # Refit the model on the training data
  fold_mod <- update(model, data = train_data)

  # Predict probabilities on the test data
  pred_probs <- posterior_epred(fold_mod, newdata = test_data) # Updated function
  pred_classes <- ifelse(colMeans(pred_probs) > cutoff, 1, 0)

  # Compute confusion matrix
  confusion <- table(
    Predicted = factor(pred_classes, levels = c(0, 1)),
    Actual = factor(test_data$senility, levels = c(0, 1))
  )

  # Handle missing classes in the confusion matrix
  if (nrow(confusion) < 2 || ncol(confusion) < 2) {
    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
```

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

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

```
CV_acc_2$cv
```

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

```
# One approach to model selection:
```

```
# L00 for rain_stanglm1 (Single Predictor)
```

```
loo1 <- loo(rain_stanglm1)
```

```
cat("L00 Estimates for Model 1 (Single Predictor):\n")
```

```
## L00 Estimates for Model 1 (Single Predictor):
```

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

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

```
# L00 for rain_stanglm2 (Multiple Predictors)
```

```
loo2 <- loo(rain_stanglm2)
```

```
cat("L00 Estimates for Model 2 (Multiple Predictors):\n")
```

```
## L00 Estimates for Model 2 (Multiple Predictors):
```

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

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

```
# Comparing L00-CV:
```

```
cat("\nModel Comparison using L00:\n")
```

```
##
```

```
## Model Comparison using L00:
```

```
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
```

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:
```

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

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

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

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

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 with humidity3pm & raintoday as the Only Predictors

```
rain_stanglm_simp <- stan_glm(
  raintomorrow ~ humidity3pm + raintoday,
  data = weather,
  family = binomial,
  prior_intercept = normal(-1.4, 0.7),
  prior = normal(0, 2.5, autoscale = TRUE),
  chains = 1,
  iter = 10000
)

# Logistic regression
# Prior for intercept
# Weakly informative priors
# Single MCMC chain for simplicity
# Total iterations (post-warmup = 5000)
```



```

# 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):
```

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

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

```

# 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:
```

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

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

```

# 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
```

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

```
# 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 β_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:
```

```
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

# 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

```

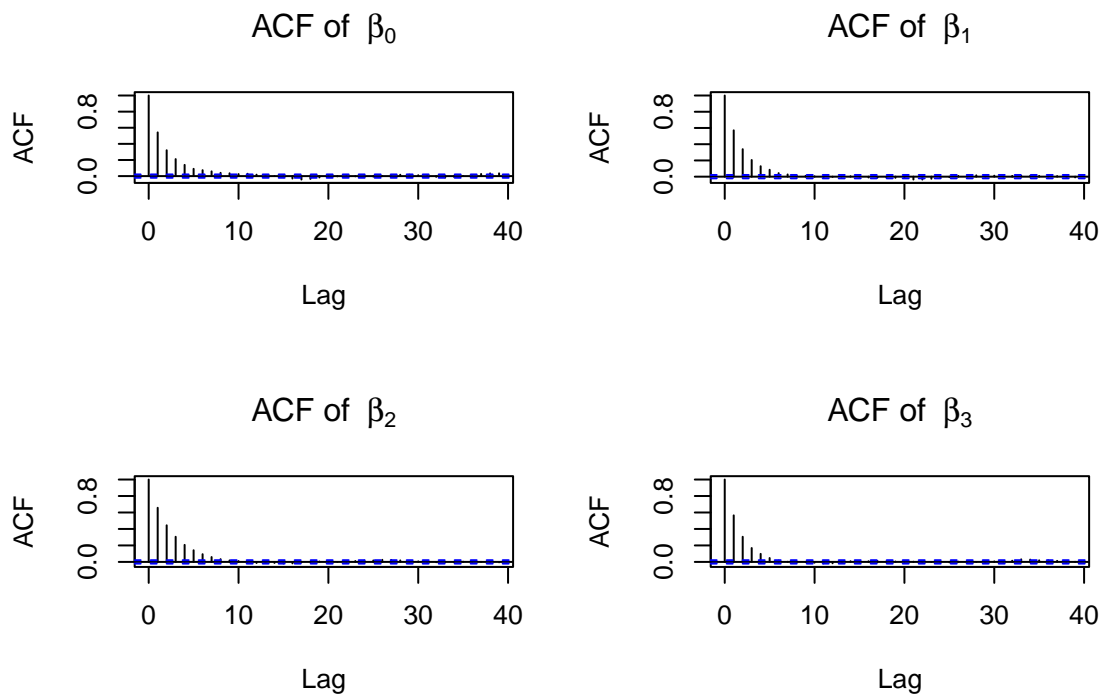
```

# 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)]))
}

```

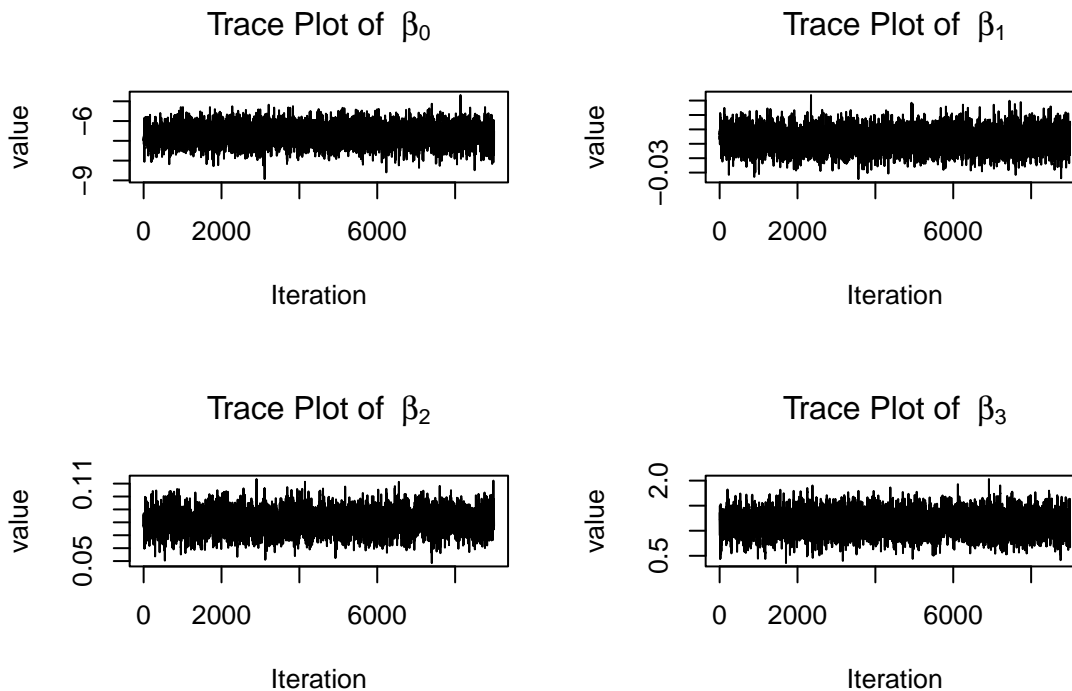


```

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")
}

```



```
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:

```
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
```