STAT 656: HW4

Rohan Dekate

Gibbs sampler for the probit model

1. Derive and write down the conditional distribution $Z|X, Y, \beta$.

Using Bayes's rule

$$p(Z|X, Y, \beta) \propto p(Y|Z, X, \beta) \cdot p(Z|X, \beta)$$

We know that $z_i \sim \mathcal{N}(x_i\beta, 1)$.

$$p(Z|X, \beta) = \prod_{i=1}^{N} p(z_i|x_i, \beta) = \prod_{i=1}^{N} \mathcal{N}(z_i|x_i\beta, 1)$$

We are also given $y_i = \mathbb{I}(z_i > 0)$. Thus, if we know z_i then y_i is known without any other dependency. So $p(Y|Z, X, \beta) = p(Y|Z)$.

$$p(Y|Z) = \prod_{i=1}^{N} p(y_i|z_i)$$

$$p(y_i|z_i) = \mathbb{I}(z_i > 0)^{y_i} \cdot \mathbb{I}(z_i \leq 0)^{1-y_i}$$

$$p(Z|X, Y, \beta) \propto \left[\prod_{i=1}^{N} p(y_i|z_i)\right] \cdot \left[\prod_{i=1}^{N} p(z_i|x_i, \beta)\right]$$

$$p(Z|X, Y, \beta) \propto \prod_{i=1}^{N} p(y_i|z_i) \cdot p(z_i|x_i, \beta)$$

$$p(Z|X, Y, \beta) = \prod_{i=1}^{N} p(z_i|x_i, y_i, \beta)$$

$$p(z_i|x_i, y_i, \beta) \propto p(y_i|z_i) \cdot p(z_i|x_i, \beta)$$

Case 1:$y_i = 1$

$$p(z_i|x_i, y_i, \beta) \propto p(y_i = 1|z_i) \cdot \mathcal{N}(z_i|x_i\beta, 1)$$

$$p(z_i|x_i, y_i, \beta) \propto \mathbb{I}(z_i > 0) \cdot \mathcal{N}(z_i|x_i\beta, 1)$$

Case 2:$y_i = 0$

$$p(z_i|x_i, y_i, \beta) \propto p(y_i = 0|z_i) \cdot \mathcal{N}(z_i|x_i\beta, 1)$$

$$p(z_i|x_i, y_i, \beta) \propto \mathbb{I}(z_i \leq 0) \cdot \mathcal{N}(z_i|x_i\beta, 1)$$

2. Derive and write down the conditional distribution $p(\beta|X, Y, Z)$?

$p(\beta|X, Y, Z) = p(\beta|X, Z)$ as once Z is known we don't need Y to determine β. Applying Bayes' rule:

$$p(\beta|X, Z) \propto p(Z|X, \beta) \cdot p(\beta)$$

$$p(Z|X, \beta) = \mathcal{N}(Z|X\beta, I_N)$$

$$p(Z|X, \beta) \propto \exp\left(-\frac{1}{2}(Z - X\beta)^T I_N^{-1}(Z - X\beta)\right)$$

$$p(Z|X, \beta) \propto \exp\left(-\frac{1}{2}(Z - X\beta)^T (Z - X\beta)\right)$$

$$p(\beta) \propto \exp\left(-\frac{1}{2}(\beta - 0)^T \Sigma_b^{-1}(\beta - 0)\right)$$

$$p(\beta) \propto \exp\left(-\frac{1}{2}\beta^T \Sigma_b^{-1}\beta\right)$$

$$p(\beta|X, Z) \propto \exp\left(-\frac{1}{2}(Z - X\beta)^T (Z - X\beta)\right) \cdot \exp\left(-\frac{1}{2}\beta^T \Sigma_b^{-1}\beta\right)$$

$$p(\beta|X, Z) \propto \exp\left(-\frac{1}{2}\left[(Z - X\beta)^T (Z - X\beta) + \beta^T \Sigma_b^{-1}\beta\right]\right)$$

$(Z - X\beta)^T (Z - X\beta) = (Z^T - \beta^T X^T)(Z - X\beta) = Z^T Z - Z^T X\beta - \beta^T X^T Z + \beta^T X^T X\beta = Z^T Z - 2\beta^T X^T Z + \beta^T X^T X\beta$

$p(\beta|X, Z) \propto \exp\left(-\frac{1}{2}\left[Z^T Z - 2\beta^T X^T Z + \beta^T X^T X\beta + \beta^T \Sigma_b^{-1}\beta\right]\right)$

$$p(\beta|X, Z) \propto \exp\left(-\frac{1}{2}\left[Z^T Z - 2\beta^T X^T Z + \beta^T X^T X\beta + \beta^T \Sigma_b^{-1}\beta\right]\right)$$

$$p(\beta|X, Z) \propto -2\beta^T X^T Z + \beta^T X^T X\beta + \beta^T \Sigma_b^{-1}\beta$$

$$p(\beta|X, Z) \propto \beta^T (X^T X + \Sigma_b^{-1})\beta - 2\beta^T (X^T Z)$$

$$\beta|X,Y,Z \sim \mathcal{N}(\mu_{\text{post}}, \Sigma_{\text{post}})$$

$$\mu_{\text{post}} = \Sigma_{\text{post}}(X^T Z)$$

$$\Sigma_{\text{post}}^{-1} = X^T X + \Sigma_b^{-1}$$

$$\Sigma_{\text{post}} = (X^T X + \Sigma_b^{-1})^{-1}$$

3. Describe the overall Gibbs sampling algorithm, and how you would calculate the posterior mean and covariance of β.

1. Choose a starting value for β. Pre-compute the constant posterior covariance matrix $\Sigma_{\text{post}} = (X^T X + \Sigma_b^{-1})^{-1}$.
2. Iterate for $t = 1, \cdots, T$:
    1. Sample $Z^{(t)}$ given $\beta^{(t-1)}$: For each $i = 1, \cdots, N$, draw $z_i^{(t)}$ from a truncated normal distribution with mean $x_i \beta^{(t-1)}$ and variance 1.
        - If $y_i = 1$, truncate to $(0, \infty)$.
        - If $y_i = 0$, truncate to $(-\infty, 0]$.
    2. Sample $\beta^{(t)}$ given $Z^{(t)}$: First calculate $\mu_{\text{post}}^{(t)} = \Sigma_{\text{post}}(X^T Z^{(t)})$. Then draw $\beta^{(t)}$ from the multivariate normal distribution $\mathcal{N}(\mu_{\text{post}}^{(t)}, \Sigma_{\text{post}})$.
3. The sequence $\{\beta^{(1)}, \beta^{(2)}, \cdots, \beta^{(T)}\}$ are the posterior draws from $p(\beta|X,Y)$

The posterior mean is estimated by the sample mean of M samples remaining after burn-in.

$$\hat{\mu}_\beta = \frac{1}{M} \sum_{t=B+1}^{T} \beta^{(t)}$$

The posterior covariance is estimated by the sample covariance matrix.

$$\hat{\Sigma}_\beta = \frac{1}{M-1} \sum_{t=B+1}^{T} (\beta^{(t)} - \hat{\mu}_\beta)(\beta^{(t)} - \hat{\mu_\beta})^T$$

4. Write an R function to implement the Gibbs sampler above. This should accept as input a dataset (X, Y) and return samples from the posterior distributions over β and the z's. One of the steps will involve sampling from a truncated normal distribution: you can either do this by rejection sampling, or by using a function from a package.

```r
probit_gibbs <- function(y, X, prior_mean, prior_precision, n_iter, burn_in) {

  N <- nrow(X) # Number of observations
  d <- ncol(X) # Number of predictors

  # Calculate number of samples to store
```

```r
n_samples <- n_iter - burn_in

# Create storage matrices for posterior samples
beta_samples <- matrix(NA, nrow = n_samples, ncol = d)
z_samples <- matrix(NA, nrow = n_samples, ncol = N)

# Initialize beta
beta_current <- as.numeric(prior_mean)

# Initialize Z
z_current <- ifelse(y == 0, -0.5, 0.5)

# This is the posterior precision for p(beta | Z, X, Y)
XtX <- t(X) %*% X
post_precision_beta <- XtX + prior_precision

# This is the posterior covariance
post_cov_beta <- solve(post_precision_beta)

# Pre-calculate (Sigma_b^-1 * mu_b) for the mean calculation
prior_prec_mean <- prior_precision %*% prior_mean

for (t in 1:n_iter) {

  # Calculate the mean for z_i's
  mu_z <- X %*% beta_current

  # Set truncation bounds based on y
  # If y_i = 1, z_i > 0.  If y_i = 0, z_i <= 0.
  lower_bounds <- ifelse(y == 1, 0, -Inf)
  upper_bounds <- ifelse(y == 1, Inf, 0)

  # Sample Z from the truncated normal distribution
  z_current <- truncnorm::rtruncnorm(
    n = N,
    a = lower_bounds,
    b = upper_bounds,
    mean = as.numeric(mu_z),
    sd = 1
  )

  # p(beta | ...) ~ N(mu_post, Sigma_post)

  # Calculate the mean for beta
  mu_beta <- post_cov_beta %*% (t(X) %*% z_current + prior_prec_mean)

  # Sample beta from the multivariate normal distribution
  beta_current <- as.numeric(mvtnorm::rmvnorm(n = 1,
                                              mean = mu_beta,
                                              sigma = post_cov_beta))

  if (t > burn_in) {
    store_index <- t - burn_in
```

```
      beta_samples[store_index, ] <- beta_current
      z_samples[store_index, ] <- z_current
    }

  }

  return(list(beta_samples = beta_samples, z_samples = z_samples))
}
```

Effect of computer-generated reminders

1. Apply your function from the previous question on the data provided in `flu_data.txt`. Describe how you initialized the algorithm, the number of iterations that you ran it for, and the burn-in period. Plot the posterior distribution over β as well as a few of the z_i's. Based on your posterior approximation, what do you conclude about the effect of the treatment on patient vaccination?

```
flu_data <- read.table("flu_data1.txt", header = TRUE, na.strings = ".")
head(flu_data)
```

```
##   treatment vaccinated age copd heartd renal liverd
## 1         1          1  73    0      1     0      0
## 2         0          1  65    0      0     0      0
## 3         0          1  77    1      1     0      0
## 4         1          1  68    0      1     0      0
## 5         1          0  68    0      1     0      0
## 6         1          0  66    0      0     0      0
```

```
# Report on missing data
n_total <- nrow(flu_data)
flu_data_clean <- na.omit(flu_data)
n_clean <- nrow(flu_data_clean)
n_dropped <- n_total - n_clean

cat(paste("Loaded", n_total, "observations.\n"))
```

```
## Loaded 2901 observations.
```

```
cat(paste("Dropped", n_dropped, "observations with missing covariates.\n"))
```

```
## Dropped 8 observations with missing covariates.
```

```
cat(paste("Analyzing", n_clean, "complete observations.\n\n"))
```

```
## Analyzing 2893 complete observations.
```

```
y_response <- flu_data_clean$vaccinated

X_design <- model.matrix(
  vaccinated ~ treatment + scale(age) + copd + heartd + renal + liverd,
  data = flu_data_clean
)

# Get dimensions
N <- nrow(X_design)
```

```r
d <- ncol(X_design)
cat(paste("Model has", N, "observations and", d, "beta coefficients (incl. intercept).\n"))
```

```
## Model has 2893 observations and 7 beta coefficients (incl. intercept).
```

```r
print("Predictors (X columns):")
```

```
## [1] "Predictors (X columns):"
```

```r
print(colnames(X_design))
```

```
## [1] "(Intercept)" "treatment"   "scale(age)"  "copd"        "heartd"
## [6] "renal"       "liverd"
```

We initialize the algorithm by setting the coefficient vector $\beta^{(0)} = 0$. Run the Gibbs sampler for a total of 5000 iterations with a burn-in of 1000 iterations.

```r
# Set a weakly informative N(0, 10*I) prior for beta
prior_mean_val <- rep(0, d)
prior_precision_val <- diag(d) * 0.1

# Set MCMC parameters
n_iter_total <- 5000
burn_in_period <- 1000

# Run the sampler
set.seed(123) # for reproducible results
gibbs_results <- probit_gibbs(
  y = y_response,
  X = X_design,
  prior_mean = prior_mean_val,
  prior_precision = prior_precision_val,
  n_iter = n_iter_total,
  burn_in = burn_in_period
)
```

```r
# Add column names to the beta samples matrix for easy analysis
colnames(gibbs_results$beta_samples) <- colnames(X_design)

# Calculate posterior summaries (Mean, SD, 95% Credible Interval)
post_summary <- data.frame(
  Mean = colMeans(gibbs_results$beta_samples),
  SD = apply(gibbs_results$beta_samples, 2, sd),
  Q2.5 = apply(gibbs_results$beta_samples, 2, quantile, probs = 0.025),
  Q97.5 = apply(gibbs_results$beta_samples, 2, quantile, probs = 0.975)
)
# Posterior summary
print(round(post_summary, 4))
```

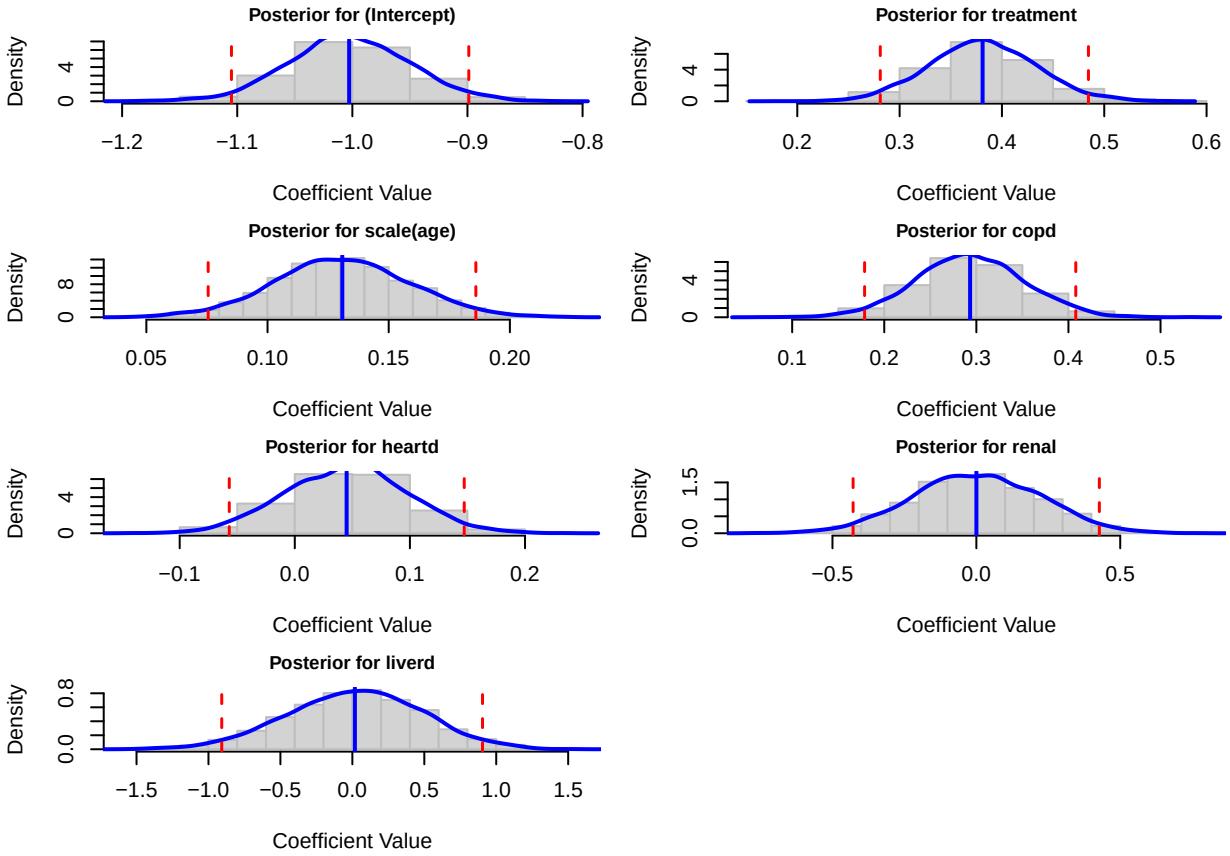
```
##                Mean     SD    Q2.5   Q97.5
## (Intercept) -1.0028 0.0524 -1.1050 -0.8988
## treatment    0.3811 0.0519  0.2811  0.4846
## scale(age)   0.1309 0.0280  0.0755  0.1860
## copd         0.2932 0.0587  0.1787  0.4080
## heartd       0.0450 0.0528 -0.0569  0.1472
## renal        0.0003 0.2215 -0.4283  0.4274
## liverd       0.0176 0.4652 -0.9077  0.9047
```

```r
treatment_effect <- post_summary["treatment", ]
print(treatment_effect)
```

```
##                Mean         SD      Q2.5    Q97.5
## treatment 0.3811121 0.05185462 0.2811155 0.484551
```

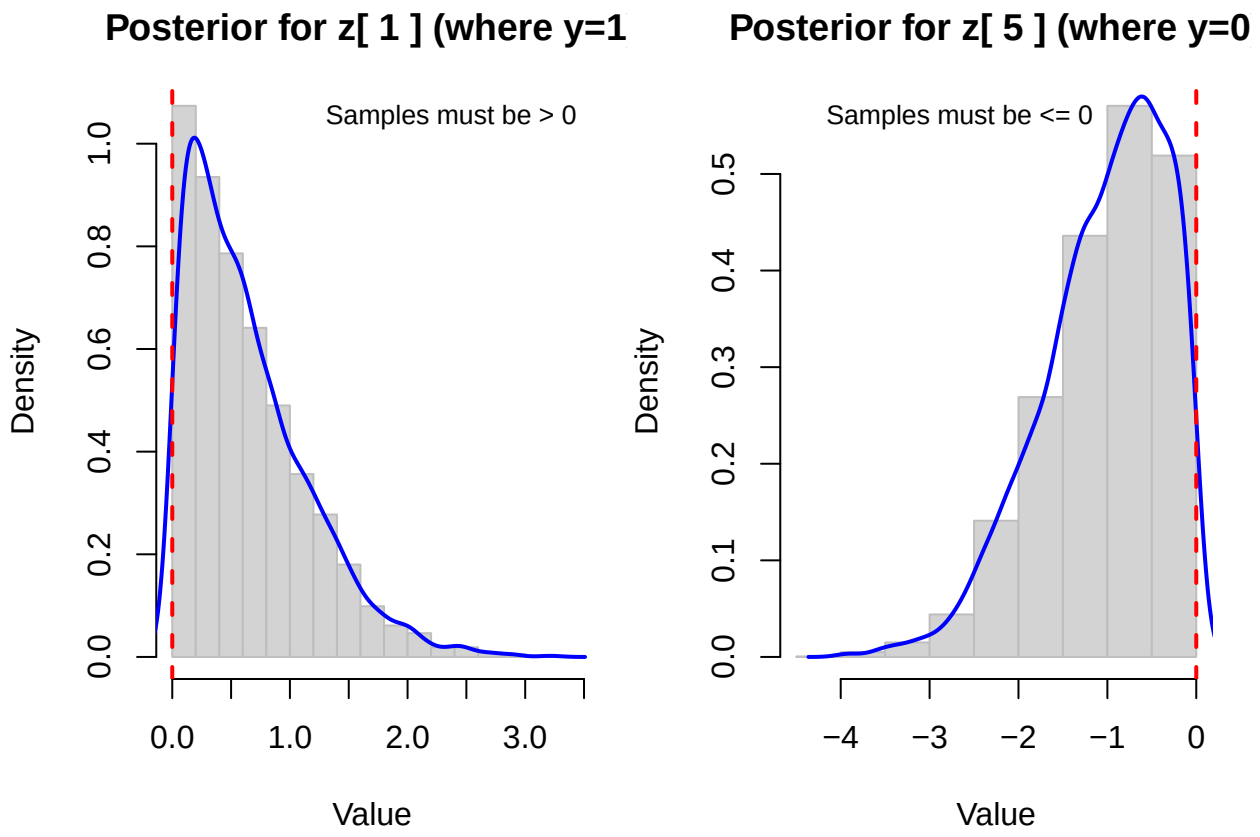
The posterior mean for the treatment coefficient is positive, indicating that the treatment increases the likelihood of vaccination.

```r
# Get the beta samples and their names
beta_samps <- gibbs_results$beta_samples
beta_names <- colnames(beta_samps)
d <- ncol(beta_samps)

# 4x2 Subplots
par(mfrow = c(4, 2), mar = c(4, 4, 2, 1))

# Loop through each beta coefficient
for (i in 1:d) {
  hist(beta_samps[, i],
       main = paste("Posterior for", beta_names[i]),
       xlab = "Coefficient Value",
       freq = FALSE,
       border = "gray",
       col = "lightgray",
       cex.main = 0.9
  )
  # Add density line
  lines(density(beta_samps[, i]), col = "blue", lwd = 2)
  # Add posterior mean line
  abline(v = mean(beta_samps[, i]), col = "blue", lty = 1, lwd = 2)
  # Add 95% CI lines
  ci <- quantile(beta_samps[, i], probs = c(0.025, 0.975))
  abline(v = ci[1], col = "red", lty = 2, lwd = 1.5)
  abline(v = ci[2], col = "red", lty = 2, lwd = 1.5)
}

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




```r
# Get the Z samples
z_samps <- gibbs_results$z_samples

# Find the index of the first patient with y=1
idx_y1 <- which(y_response == 1)[1]
# Find the index of the first patient with y=0
idx_y0 <- which(y_response == 0)[1]

# Set up a 1x2 panel plot
par(mfrow = c(1, 2), mar = c(4, 4, 3, 1))

# Plot for z_i where y_i = 1
hist(z_samps[, idx_y1],
     main = paste("Posterior for z[", idx_y1, "] (where y=1)"),
     xlab = "Value",
     freq = FALSE,
     border = "gray",
     col = "lightgray"
)
lines(density(z_samps[, idx_y1]), col = "blue", lwd = 2)
abline(v = 0, col = "red", lty = 2, lwd = 2) # Show truncation point
legend("topright", "Samples must be > 0", bty="n", cex=0.8)

# Plot for z_i where y_i = 0
hist(z_samps[, idx_y0],
     main = paste("Posterior for z[", idx_y0, "] (where y=0)"),
     xlab = "Value",
```

```
     freq = FALSE,
     border = "gray",
     col = "lightgray"
)
lines(density(z_samps[, idx_y0]), col = "blue", lwd = 2)
abline(v = 0, col = "red", lty = 2, lwd = 2) # Show truncation point
legend("topleft", "Samples must be <= 0", bty="n", cex=0.8)
```




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

Because the 95% credible interval is entirely positive and does not contain zero, we have strong statistical evidence to conclude that the **computer-generated reminder (the treatment) had a positive effect on the likelihood of patient vaccination**, even after accounting for the patient's age and other health conditions.

2. Implement the probit model in Stan, and produce an MCMC approximation to the posterior. Your results here should agree with those from the previous question. Verify this, describing how you did this. Also, compare the mixing of the two MCMC samplers, and comment on this.

```
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
##
## rstan version 2.32.7 (Stan version 2.32.2)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)
```

```
library(parallel)

# Set Stan to run on multiple cores
options(mc.cores = parallel::detectCores())

stan_model_code <- "
data {
  int<lower=0> N;          // Number of observations
  int<lower=0> d;          // Number of predictors (including intercept)
  matrix[N, d] X;          // Design matrix
  int<lower=0, upper=1> y[N]; // Binary response vector
}
parameters {
  vector[d] beta;          // Vector of coefficients
}
model {
  // Define the linear predictor
  vector[N] eta = X * beta;

  // Prior: beta ~ N(0, 10*I)
  beta ~ normal(0, sqrt(10));

  // Likelihood:
  y ~ bernoulli(Phi(eta));
}
"
```

```
# Create the data list for Stan
stan_data_list <- list(
  N = N,
  d = d,
  X = X_design,
  y = y_response
)

# Compile Stan model
stan_model <- stan_model(model_code = stan_model_code)
```

```
## Trying to compile a simple C file
```

```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## using C compiler: 'Apple clang version 17.0.0 (clang-1700.4.4.1)'
## using SDK: 'MacOSX26.1.sdk'
## clang -arch arm64 -std=gnu2x -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Li
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/RcppEigen
## In file included from /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/RcppEigen
```

```
## /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/RcppEigen/include/Eigen/src/Cor
##   679 | #include <cmath>
##       |          ^~~~~~~
## 1 error generated.
## make: *** [foo.o] Error 1
```

```
# Run NUTS sampler
stan_fit <- sampling(
  stan_model,
  data = stan_data_list,
  iter = 2000,      # 1000 warmup + 1000 post-warmup
  warmup = 1000,
  chains = 4,       # 4 chains * 1000 = 4000 total samples
  seed = 42
)
```

```
# Summarize posterior
stan_summary <- summary(stan_fit, pars = "beta",
                        probs = c(0.025, 0.975))$summary

# Add coefficient names
rownames(stan_summary) <- colnames(X_design)

print(round(stan_summary, 4))
```

```
##                mean se_mean     sd    2.5%   97.5%    n_eff   Rhat
## (Intercept) -1.0005  0.0010 0.0504 -1.1005 -0.9027 2457.234 1.0014
## treatment    0.3816  0.0010 0.0505  0.2834  0.4827 2816.906 1.0000
## scale(age)   0.1318  0.0004 0.0278  0.0768  0.1868 4082.453 1.0004
## copd         0.2908  0.0009 0.0562  0.1803  0.4001 4128.665 1.0011
## heartd       0.0432  0.0009 0.0519 -0.0594  0.1439 3400.585 1.0001
## renal       -0.0135  0.0032 0.2232 -0.4668  0.4068 4947.265 1.0008
## liverd       0.0105  0.0073 0.4620 -0.9567  0.8618 3961.050 1.0002
```

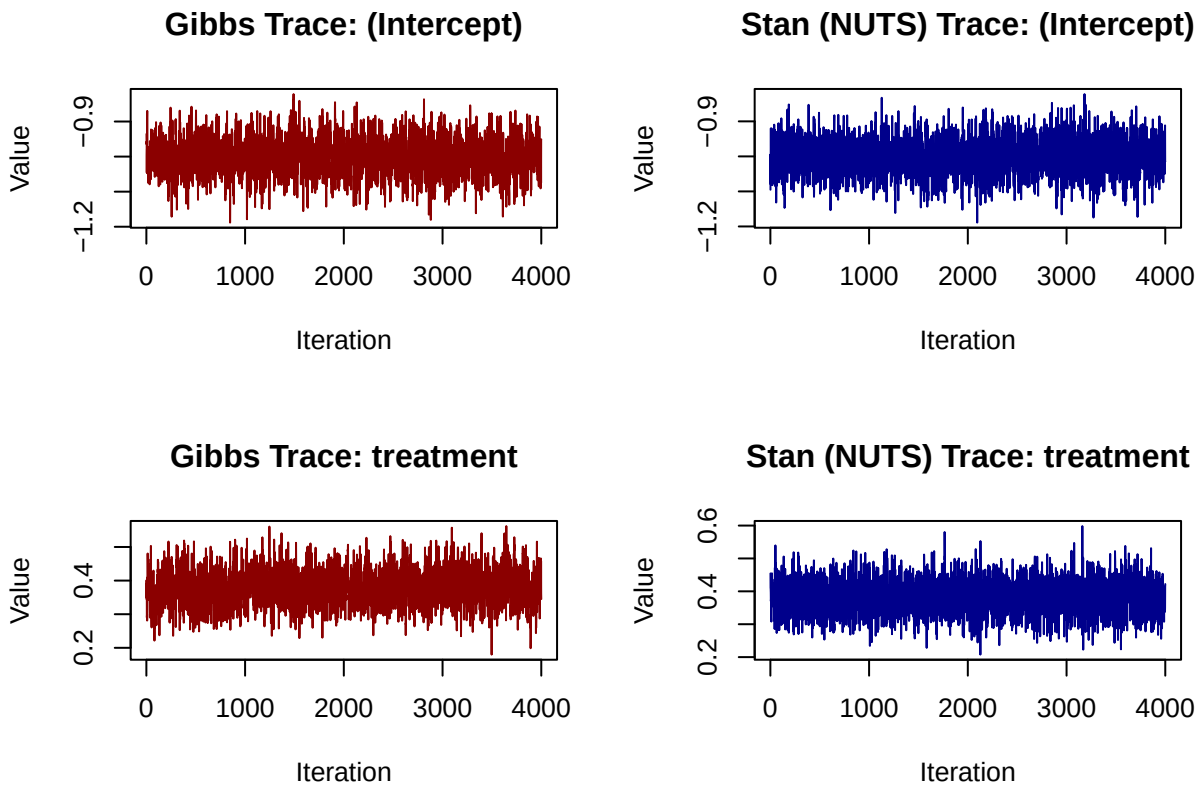
```
print(round(post_summary, 4))
```

```
##                Mean     SD    Q2.5   Q97.5
## (Intercept) -1.0028 0.0524 -1.1050 -0.8988
## treatment    0.3811 0.0519  0.2811  0.4846
## scale(age)   0.1309 0.0280  0.0755  0.1860
## copd         0.2932 0.0587  0.1787  0.4080
## heartd       0.0450 0.0528 -0.0569  0.1472
## renal        0.0003 0.2215 -0.4283  0.4274
## liverd       0.0176 0.4652 -0.9077  0.9047
```

The statistical summary between the NUTS and Gibbs sampler are in close agreement with each other.

```
stan_traces <- as.data.frame(stan_fit, pars = "beta")

# Set up a 2x2 plot window
par(mfrow = c(2, 2))

# Plot 1: Gibbs Trace for (Intercept)
plot(gibbs_results$beta_samples[, "(Intercept)"], type = "l",
       col = "darkred", main = "Gibbs Trace: (Intercept)",
       xlab = "Iteration", ylab = "Value")
```

```r
# Plot 2: Stan/NUTS Trace for (Intercept)
plot(stan_traces$`beta[1]`, type = "l",
       col = "darkblue", main = "Stan (NUTS) Trace: (Intercept)",
       xlab = "Iteration", ylab = "Value")

# Plot 3: Gibbs Trace for treatment
plot(gibbs_results$beta_samples[, "treatment"], type = "l",
       col = "darkred", main = "Gibbs Trace: treatment",
       xlab = "Iteration", ylab = "Value")

# Plot 4: Stan/NUTS Trace for treatment
plot(stan_traces$`beta[2]`, type = "l",
       col = "darkblue", main = "Stan (NUTS) Trace: treatment",
       xlab = "Iteration", ylab = "Value")
```




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

```r
library(coda)
```

```
##
## Attaching package: 'coda'
```

```
## The following object is masked from 'package:rstan':
##
##     traceplot
```

```r
gibbs_mcmc <- as.mcmc(
  gibbs_results$beta_samples,
  start = burn_in_period + 1,
```

```
  end = n_iter_total,
  thin = 1
)

n_eff <- effectiveSize(gibbs_mcmc)
print(round(n_eff, 0))
```

```
## (Intercept)   treatment  scale(age)        copd      heartd       renal
##        1303        1356        1175        1411        1398        1497
##      liverd
##        1464
```

The Stan model with NUTS sampler has higher effective sample size compared to the Gibbs sampler. Thus the NUTS sampler is more efficient than Gibbs sampler providing better mixing.

3. Implement the logistic regression model in Stan, and produce an MCMC approximation to the posterior. Comment on any difference between this as the results from the probit model.

```
stan_model_code_logistic <- "
data {
  int<lower=0> N;          // Number of observations
  int<lower=0> d;          // Number of predictors (incl. intercept)
  matrix[N, d] X;          // Design matrix
  int<lower=0, upper=1> y[N]; // Binary response vector
}
parameters {
  vector[d] beta;          // Vector of coefficients
}
model {
  // Define the linear predictor
  vector[N] eta = X * beta;

  // Prior: beta ~ N(0, 10*I)
  beta ~ normal(0, sqrt(10));

  // Likelihood:
  // This is the logistic regression likelihood
  y ~ bernoulli_logit(eta);
}
"
```

```
stan_model_logistic <- stan_model(model_code = stan_model_code_logistic)
```

```
## Trying to compile a simple C file
```

```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## using C compiler: 'Apple clang version 17.0.0 (clang-1700.4.4.1)'
## using SDK: 'MacOSX26.1.sdk'
## clang -arch arm64 -std=gnu2x -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Li
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/StanHeade
## In file included from /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/RcppEigen
## In file included from /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/RcppEigen
## /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/library/RcppEigen/include/Eigen/src/Cor
```

```
##   679 | #include <cmath>
##       |          ^~~~~~
## 1 error generated.
## make: *** [foo.o] Error 1
```

```
stan_fit_logistic <- sampling(
  stan_model_logistic,
  data = stan_data_list,
  iter = 2000,      # 1000 warmup + 1000 post-warmup
  warmup = 1000,
  chains = 4,       # 4 chains * 1000 = 4000 total samples
  seed = 42
)
```

```
stan_summary_logistic <- summary(stan_fit_logistic, pars = "beta",
                                 probs = c(0.025, 0.975))$summary

# Add coefficient names
rownames(stan_summary_logistic) <- colnames(X_design)
print(round(stan_summary_logistic, 4))
```

```
##                mean se_mean     sd    2.5%   97.5%    n_eff   Rhat
## (Intercept) -1.6589  0.0017 0.0905 -1.8373 -1.4787 2712.601 1.0004
## treatment    0.6453  0.0015 0.0868  0.4714  0.8191 3271.548 1.0000
## scale(age)   0.2235  0.0007 0.0475  0.1305  0.3181 4075.605 0.9997
## copd         0.4894  0.0015 0.0937  0.3064  0.6736 4013.209 1.0002
## heartd       0.0729  0.0015 0.0901 -0.1014  0.2526 3542.381 0.9998
## renal       -0.0578  0.0059 0.3874 -0.8391  0.6729 4266.675 0.9998
## liverd      -0.0774  0.0128 0.8432 -1.9376  1.4422 4309.486 0.9999
```

We cannot compare the β coefficients between a probit and logistic model directly. The probit model uses a normal distribution while the logistic model uses a logistic distribution. We need a common scale for comparison.

```
scaling_factor <- 1.704
logit_results <- stan_summary_logistic["treatment", c("mean", "2.5%", "97.5%")]
probit_results <- stan_summary["treatment", c("mean", "2.5%", "97.5%")]
scaled_probit_results <- probit_results * scaling_factor

comparison_df <- data.frame(
  Model = c("Logistic", "Probit (Scaled by 1.704)"),
  Mean = c(logit_results[1], scaled_probit_results[1]),
  Lower_95_CI = c(logit_results[2], scaled_probit_results[2]),
  Upper_95_CI = c(logit_results[3], scaled_probit_results[3]),
  row.names = NULL
)
print(comparison_df)
```

```
##                      Model      Mean Lower_95_CI Upper_95_CI
## 1                 Logistic 0.6452683   0.4714354   0.8191252
## 2 Probit (Scaled by 1.704) 0.6502796   0.4828465   0.8225044
```

The above table indicates that the results are almost the same for the treatment variable.

The treatment has a strong, positive, and statistically significant effect on the likelihood of patient vaccination.