# Stats 451 Final Project

Alyssa Yang

2024-04-30

## Load in libraries

```
library(readxl)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(rstan)
```

```
## Loading required package: StanHeaders

##
## rstan version 2.32.3 (Stan version 2.26.1)

## 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)
```

```
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
```

## Read in data, remove NA values

```r
data <- read_xlsx("combined_michigan_law.xlsx")
data <- na.omit(data)
# View(data)
nrow(data)
```

```
## [1] 6609
```

## Find the mean LSAT score and mean GPA of those who were accepted

```r
accepted <- data[data$Status == "Accepted",]

mean_lsat = mean(accepted$LSAT)
mean_lsat
```

```
## [1] 170.1432
```

```r
mean_gpa = mean(accepted$GPA)
mean_gpa
```

```
## [1] 3.71689
```

## Select the first 200 datapoints from each year

```r
data <- data %>%
  group_by(Year) %>%
  # Select the first 200 rows within each group
  slice_head(n = 200) %>%
  ungroup()
```

## STAN model

```r
model_string <- "
data {
  int<lower=0> N;                   // Number of observations
  int<lower=0,upper=1> status[N];  // Accepted (1) or Rejected (0)
  vector[N] lsat;                   // LSAT
  vector[N] gpa;                    // GPA
  int<lower=0,upper=1> urm[N];      // URM indicator
  int<lower=0,upper=1> intl[N];     // International indicator
}
```

```
parameters {
  real alpha;               // Intercept
  vector[4] beta;           // Coefficients for predictors
}

model {
  alpha ~ normal(0.284, 100);         // Prior for intercept
  beta ~ normal(0, 100);           // Prior for coefficients

  // Likelihood
  for (i in 1:N) {
    real p;
    p = inv_logit(alpha + beta[1] * lsat[i] + beta[2] * gpa[i] + beta[3] * urm[i] + beta[4] * intl[i]);
    status[i] ~ binomial(1, p); // Likelihood
  }
}
"

# Prepare data
data_list <- list(
  N = nrow(data),
  status = as.integer(data$Status == "Accepted"),
  lsat = data$LSAT,
  gpa = data$GPA,
  urm = as.integer(data$URM),
  intl = as.integer(data$Intl)
)

# Compiling and producing posterior samples from the model.
fit <- stan(model_code = model_string, data = data_list)
```

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

## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/inclu
## In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
##                 ^
##                 ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/4.2/Resources/library/RcppEigen/include/Eigen/Core:96:10: fa
## #include <complex>
##          ^~~~~~~~
## 3 errors generated.
```

```
## make: *** [foo.o] Error 1
```

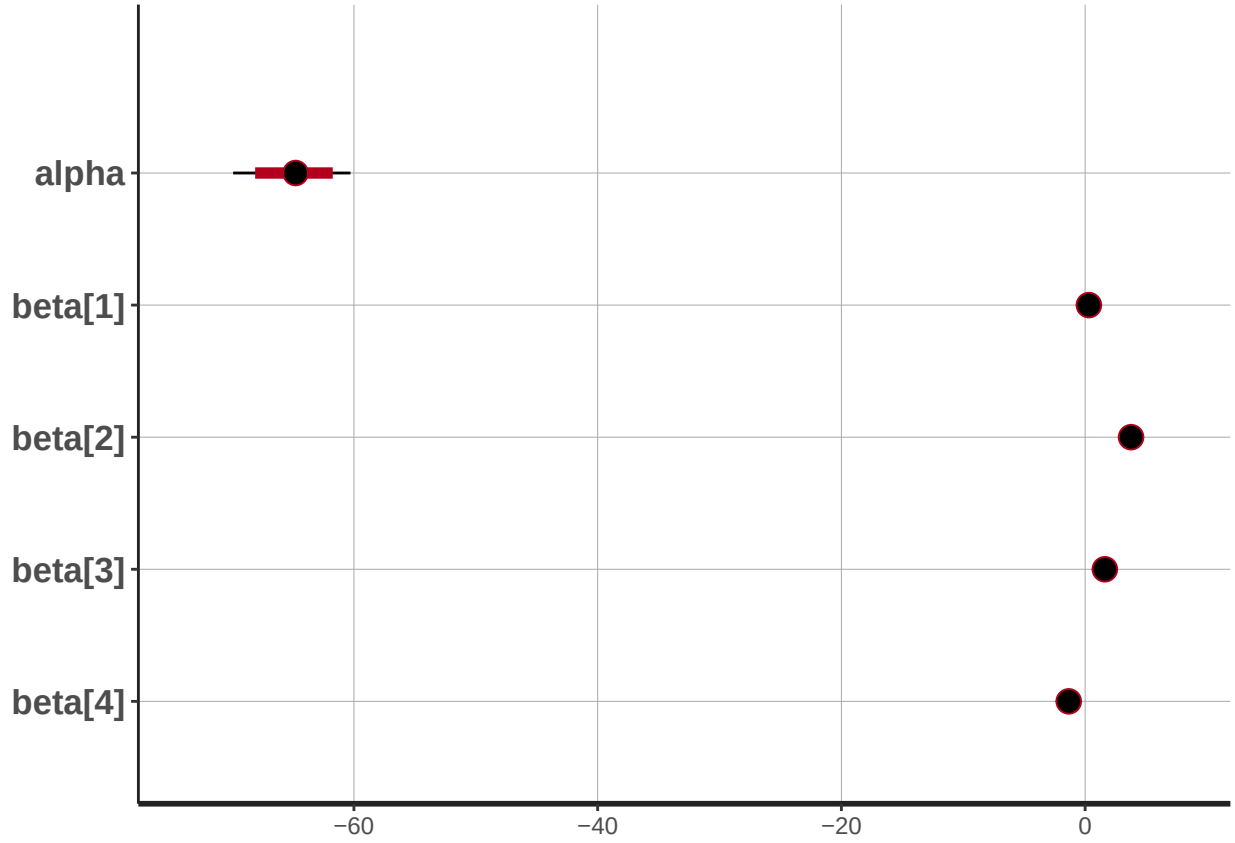
```
# Plotting and summarizing the posterior distribution
fit
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##             mean se_mean   sd     2.5%      25%      50%      75%    97.5%
## alpha     -64.88    0.07 2.48   -69.91   -66.51   -64.78   -63.17   -60.28
## beta[1]     0.31    0.00 0.01     0.28     0.30     0.31     0.32     0.33
## beta[2]     3.77    0.00 0.18     3.43     3.65     3.77     3.89     4.14
## beta[3]     1.62    0.00 0.15     1.33     1.52     1.62     1.72     1.92
## beta[4]    -1.35    0.00 0.26    -1.86    -1.53    -1.34    -1.17    -0.84
## lp__    -1471.47    0.04 1.48 -1475.07 -1472.25 -1471.18 -1470.37 -1469.49
##         n_eff Rhat
## alpha    1404    1
## beta[1]  1481    1
## beta[2]  2341    1
## beta[3]  1866    1
## beta[4]  2854    1
## lp__     1597    1
##
## Samples were drawn using NUTS(diag_e) at Tue Apr 30 15:23:16 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
plot(fit)
```

```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```




## Find mean of posterior alpha and beta

```
# Extract posterior samples
posterior_samples <- as.data.frame(fit)

# Extract alpha and beta from posterior samples
alpha_samples <- posterior_samples$alpha
beta_samples <- posterior_samples[, paste0("beta[", 1:4, "]")]

# Calculate summary statistics for alpha and beta
summary_alpha <- summary(alpha_samples)
summary_beta <- summary(beta_samples)

# Print summary statistics
print(summary_alpha)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -74.97  -66.51  -64.78  -64.88  -63.17  -57.46
```

```
print(summary_beta)
```

```
##     beta[1]          beta[2]          beta[3]          beta[4]
```

```
##  Min.   :0.2700   Min.   :3.148   Min.   :1.077   Min.   :-2.2921
##  1st Qu.:0.3000   1st Qu.:3.652   1st Qu.:1.519   1st Qu.:-1.5284
##  Median :0.3085   Median :3.770   Median :1.619   Median :-1.3441
##  Mean   :0.3089   Mean   :3.774   Mean   :1.621   Mean   :-1.3469
##  3rd Qu.:0.3172   3rd Qu.:3.894   3rd Qu.:1.720   3rd Qu.:-1.1707
##  Max.   :0.3589   Max.   :4.465   Max.   :2.209   Max.   :-0.3712
```

## obtaining posterior probability of acceptance for fake person

```
# Find means of posterior alpha and betas
mean_alpha = mean(alpha_samples)
mean_beta1 = mean(beta_samples$`beta[1]`)
mean_beta2 = mean(beta_samples$`beta[2]`)
mean_beta3 = mean(beta_samples$`beta[3]`)
mean_beta4 = mean(beta_samples$`beta[4]`)

# Data for fake person
x1 = 173
x2 = 3.703
x3 = 0
x4 = 1

# Find probability
val = mean_alpha + mean_beta1 * x1 + mean_beta2 * x2 + mean_beta3 * x3 + mean_beta4 * x4

prob <- exp(val) / (1 + exp(val))
prob
```

```
## [1] 0.7662082
```