PLSC 40502: Problem Set 2

Solutions

January 23, 2022

This problem set is due at **11:59 pm on Friday, February 3rd**.

Please upload your solutions as a .pdf file saved as "Yourlastname_Yourfirstinitial_pset2.pdf"). In addition, an electronic copy of your .Rmd file (saved as "Yourlastname_Yourfirstinitial_pset2.Rmd") must be submitted to the course website at the same time. We should be able to run your code without error messages. In addition to your solutions, please submit an annotated version of this `.rmd` file saved as "Yourlastname_Yourfirstinitial_pset2_feedback.rmd", noting the problems where you needed to consult the solutions and why along with any remaining questions or concerns about the material. In order to receive credit, homework submissions must be substantially started and all work must be shown.

Problem 1

The **Congressional Election Study** is an annual, large, nationally representative survey of the American population designed to understand how Americans view their representatives. We will be using this dataset to predict the level of support for U.S. trade policy, particularly the Section 232 Steel and Aluminum tariffs implemented by the Trump administration and currently in place under the Biden administration.

You will find the `CCES Guide 2020.pdf` very useful for understanding the definition and structure of different variables (see after page 27, the "common content") and this problem will assume that you are able to use this reference to identify relevant outcomes and covariates. Note that the numeric values in the CSV dataset below correspond in order to the categories listed in the PDF. For example, the Guide lists two responses for `gender`: Male and Female. Therefore, "Male" is coded as a 1 and Female is coded as a "2"

Be very careful in reading the variable names and definitions!

The code below will load in the Common Content from the 2020 CES. Please download the file directly from `https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi%3A10.7910/DVN/E9N6PH` and place it into your data folder:

```
ces <- read_csv("data/CES20_Common_OUTPUT_vv.csv")
```

Our main outcome of interest is `CC20_338b`, whether the respondent supports a policy of "25% tariffs on imported steel and 10% on imported aluminum, EXCEPT from Canada and Mexico." (1 = Yes, 2 = No)

We'll be predicting this using data on age, sex, race, education and income. Start by cleaning the data and setting up the preliminaries

Part A

We'll need to coarsen our covariates into a higher level of aggregation since some bins are very, very small. Specifically, we want to match the bins for sex, race, age, education and income used to generate the national vote breakdown in the CCES Guide (pages 25/26). Using the definitions given by those bins, and the following variables, generate a factor variable for gender, race, age and education (you should have four factor variables in total):

- `birthyr` Year of birth (also remember that this survey was conducted in 2020)
- `gender` - Gender of respondent (1 = Male, 2 = Female)
- `race` - "What racial or ethnic group best describes you?"
- `hispanic` - "Are you of Spanish, Latino, or Hispanic origin or descent?" (Note for the 'hispanic' category in the 5-category race variable, you should code respondents who answer yes to this question *or* "hispanic" to `race`)
- `educ` - "What is the highest level of education you have completed?"

You can validate that your coarsenings are correct using the "% of electorate" column in the national vote breakdown from pages 25/26 of the CCES Guide. Subset the data to respondents without missing `vvweight_post` (matched to voter file + filled out both waves) and without missing `CL_2020gvm` and take weighted averages using the sampling weights.

Generate a dataset that contains an indicator variable for whether the respondent supports the tariff `CC20_338b == 1` along with your four predictors and remove all missing data. Set the baselines for gender to male, age to "18-29", race to White and education to "High school or less".

To check your work, your final dataset should contain `60398` observations.

Start with some data cleaning using `mutate`

```
# Outcome
ces <- ces %>% mutate(tariff = case_when(CC20_338b == "1" ~ 1,
                                         CC20_338b == "2" ~ 0,
                                         TRUE ~ NA_real_))

# Covariates
# Age
ces <- ces %>% mutate(age = 2020 - birthyr)
ces <- ces %>% mutate(age_bin = case_when(age>=18&age<=29 ~ "18-29",
                                          age>=30&age<=44 ~ "30-44",
                                          age>=45&age<=64 ~ "45-64",
                                          age>=65 ~ "65+"))

# Gender
ces <- ces %>% mutate(gender_bin = case_when(gender == 1 ~ "Male",
                                             gender == 2 ~ "Female"))

# Race
ces <- ces %>% mutate(race_bin = case_when(race==3|hispanic==1 ~ "Hispanic",
                                           race==1 ~ "White",
                                           race ==2 ~ "Black",
                                           race == 4  ~ "Asian",
                                           TRUE ~ "Other"))

# Education
ces <- ces %>% mutate(educ_bin = case_when(educ>=1&educ<=2 ~ "H.S. or less",
                                           educ>=3&educ<=4 ~ "Some college",
                                           educ==5 ~ "College degree",
                                           educ==6 ~ "Postgraduate"))
```

Sanity check to see if the marginal distributions are correct compared to the CCES documents (optional!)

```
ces_voted <- ces %>% filter(!is.na(vvweight_post)&!is.na(CL_2020gvm))
```

```
# Gender
ces_voted %>% group_by(gender_bin) %>% summarize(n= sum(vvweight_post)) %>% ungroup() %>% mutate(prop =
```

```
## # A tibble: 2 x 3
##   gender_bin      n  prop
##   <chr>       <dbl> <dbl>
## 1 Female     18552. 0.526
## 2 Male       16740. 0.474
```

```
# Race
ces_voted %>% group_by(age_bin) %>% summarize(n= sum(vvweight_post)) %>% ungroup() %>% mutate(prop = n/s
```

```
## # A tibble: 4 x 3
##   age_bin      n  prop
##   <chr>    <dbl> <dbl>
## 1 18-29    5466. 0.155
## 2 30-44    8112. 0.230
## 3 45-64   12311. 0.349
## 4 65+      9403. 0.266
```

```
# Race
ces_voted %>% group_by(race_bin) %>% summarize(n= sum(vvweight_post)) %>% ungroup() %>% mutate(prop = n/
```

```
## # A tibble: 5 x 3
##   race_bin      n   prop
##   <chr>     <dbl>  <dbl>
## 1 Asian     1248. 0.0354
## 2 Black     3849. 0.109
## 3 Hispanic  3703. 0.105
## 4 Other      917. 0.0260
## 5 White    25575. 0.725
```

```
# College
ces_voted %>% group_by(educ_bin) %>% summarize(n= sum(vvweight_post)) %>% ungroup() %>% mutate(prop = n/
```

```
## # A tibble: 4 x 3
##   educ_bin            n  prop
##   <chr>           <dbl> <dbl>
## 1 College degree  9276. 0.263
## 2 H.S. or less   10077. 0.286
## 3 Postgraduate    5465. 0.155
## 4 Some college   10474. 0.297
```

Make the variables factors with the correct baselines

```
ces$gender_bin <- relevel(as.factor(ces$gender_bin), "Male")
ces$age_bin <- relevel(as.factor(ces$age_bin), "18-29")
ces$educ_bin <- relevel(as.factor(ces$educ_bin), "H.S. or less")
ces$race_bin <- relevel(as.factor(ces$race_bin), "White")
```

Subset down to the non-missing data

```
ces_full <- ces %>% filter(!is.na(tariff)&!is.na(age_bin)&!is.na(race_bin)&!is.na(gender_bin)&!is.na(edu
nrow(ces_full) # sanity check row number
```

```
## [1] 60398
```

Part B

Next, let's write down a model. We'll assume that the data-generating process for the indicator $Y_i \in \{0, 1\}$ of whether the respondent supports the tariffs has the following DGP:

$$Y_i = \mathbb{1}(Y_i^* > 0)$$

where $\mathbb{1}()$ denotes the indicator function. That is, when Y_i^* is greater than 0, $Y_i = 1$ and when Y_i^* is less than or equal to 0, $Y_i = 0$.

$$Y_i^* \sim \text{Normal}(X_i'\beta, 1)$$

and

$$\beta \sim \text{Normal}(b_0, B_0^{-1})$$

Start by classifying each of the variables in the DGP. What is the **observed data**? What is a **latent variable** and what is a **known constant**?

- **observed data**: Y_i
- **latent variable**: Y_i^*, β
- **known constant**: X_i, b_0, B_0 (the latter two are the **hyperparameters**)

Part C

Our goal is to do inference on the posterior distribution

$$f(\beta|\mathbf{Y}, \mathbf{X}) \propto f(\mathbf{Y}|\beta, \mathbf{X})f(\beta|\mathbf{X}) \equiv f(\mathbf{Y}|\beta, \mathbf{X})f(\beta)$$

However, this is difficult. Instead, we simulate from the joint posterior that includes the $\mathbf{Y}^* = \{Y_1^*, Y_2^*, Y_3^*, \ldots, Y_N^*\}$:

$$f(\beta, \mathbf{Y}^*|\mathbf{Y}, \mathbf{X})$$

We will then marginalize over this joint distribution by keeping the draws of β.

Use Bayes' rule to write this posterior distribution as proportional to a **likelihood** multiplied by a **prior**. Factor the density as much as possible and use the independencies and conditional independencies implied by the model to eliminate terms from the conditioning set of some of the densities. You should have three main component densities: one related to the prior on β and two likelihood terms.

Apply Bayes' rule

$$f(\beta, \mathbf{Y}^*|\mathbf{Y}, \mathbf{X}) \propto f(\mathbf{Y}|\beta, \mathbf{Y}^*, \mathbf{X})f(\mathbf{Y}^*, \beta|\mathbf{X})$$

If we know $\mathbf{Y}^*$, we know $\mathbf{Y}$ deterministically – $\mathbf{X}$ and β don't provide any additional information, so by conditional independence, we have

$$f(\beta, \mathbf{Y}^*|\mathbf{Y}, \mathbf{X}) \propto f(\mathbf{Y}|\mathbf{Y}^*)f(\mathbf{Y}^*, \beta|\mathbf{X})$$

Next, factor the prior

$$f(\beta, \mathbf{Y}^*|\mathbf{Y}, \mathbf{X}) \propto f(\mathbf{Y}|\mathbf{Y}^*) \times f(\mathbf{Y}^*|\beta, \mathbf{X}) \times f(\beta|\mathbf{X})$$

By independence, we can factor the likelihood

$$f(\beta, \mathbf{Y}^*|\mathbf{Y}, \mathbf{X}) \propto \left[\prod_{i=1}^{N} f(Y_i|Y_i^*) \times f(Y_i^*|\beta, X_i)\right] \times f(\beta|\mathbf{X})$$

We have two "likelihood-style" terms: $f(Y_i|Y_i^*)$ and $f(Y_i^*|\beta, X_i)$ multiplied by the prior on β $f(\beta|\mathbf{X}) = f(\beta)$

$$f(\beta, \mathbf{Y}^*|\mathbf{Y}, \mathbf{X}) \propto \left[\prod_{i=1}^{N} f(Y_i|Y_i^*) \times f(Y_i^*|\beta, X_i)\right] \times f(\beta)$$

Part D

What is the conditional distribution of $Y_i^*|Y_i, \beta, X_i$?

Hint: What do we already know is the conditional distribution of $Y_i^*|\beta, X_i$? What information does adding Y_i provide?

The conditional is proportional to the joint distribution, which we've already factored from above

$$f(Y_i^*|Y_i, \beta, X_i) \propto f(Y_i|Y_i^*) \times f(Y_i^*|\beta, X_i)$$

The second term is normal, mean $X_i'\beta$, variance 1 by the definition of our model from above

The first term is just an indicator – it takes on a value of 1 if $Y_i = 1$ and Y_i^* is positive or if $Y_i = 0$ and Y_i^* is negative.

Therefore, the conditional distribution of Y_i^* given Y_i is a **truncated normal distribution**. If $Y_i = 1$, Y_i^* is truncated normal on $(0, \infty)$ and if $Y_i = 0$, Y_i^* is truncated normal on $(-\infty, 0]$

Intuitively, the observed data constrain the latent variable to be in either the positive or negative reals but provide no additional information.

Part E

What is the form of the conditional distribution of $\beta|Y_i^*, Y_i, X_i$ (just the form/type of the distribution, you don't need to derive its mean/variance though this is a standard result)?

Hint 1: What can we get rid of on the right-hand side of the conditioning bar (using conditional independence)?
Hint 2: Once we've gotten rid of that variable, what familiar posterior distribution from a different model does this remind you of?

Conditional on the latent variable Y_i^*, Y_i provides no additional information so the conditional distribution can be written as $f(\beta|Y_i^*, X_i)$.

Next, by Bayes' rule

$$f(\beta|Y_i^*, X_i) \propto f(Y_i^*|\beta, X_i) \times f(\beta|X_i)$$

And by our assumed prior on β,

$$f(\beta|Y_i^*, X_i) \propto f(Y_i^*|\beta, X_i) \times f(\beta)$$

We know $f(\beta)$ is the normal density and so is $f(Y_i^*|\beta, X_i)$ and the product of two normal densities is also a normal density. Hence the conditional distribution $\beta|Y_i^*, Y_i, X_i$ is normal.

Essentially, this is the same as Bayesian linear regression with Y_i^* as the "outcome" and a known outcome variance of 1.

Part F

Let's estimate the probit model to predict the probability of supporting the tariff conditional on age, sex, race and education level. For now, include the covariates additively, omitting any interactions (e.g. education-race). Use `model.matrix()` to generate the regression matrix $\mathbf{X}$.

Implement a Gibbs sampler to obtain draws from the joint posterior distribution. Use a fairly diffuse prior for β that sets $b_0 = 0$ and $B_0 = \frac{1}{10}\mathbf{I}$ where $\mathbf{I}$ is the identity matrix.

Pick some reasonable starting values for β and start by sampling from the conditional distribution of Y_i^* to generate "starting" values of Y_i^*. Iterate between sampling from the conditional distribution of β and the conditional distribution of the Y_i^*. You only need to store all of the samples from β (this should save on memory as you do not need to store all 2000 iterations of the $\approx 60,000$ draws from Y_i^*).

Set the starting seed to `60637` and run the sampler for `2000` iterations after discarding the first `500` as a "burn-in" (so you should run it for 2,500 iterations in total). My implementation took about 1.5 minutes to run on my desktop, so be aware that you may want to test your code with fewer iterations before launching the full chain.

Hint: You can efficiently sample from the distribution you found in Part D using the `truncnorm` R package.

Hint 2: Using well-known results, the distribution from Part E has variance-covariance matrix

$$\mathbf{V} = (B_0 + \mathbf{X}'\mathbf{X})^{-1}$$

and mean

$$\mathbf{M} = \mathbf{V}(B_0 b_0 + \mathbf{X}'\mathbf{Y}^*)$$

Start with the preliminaries - make the model matrix

```
X_mat <- model.matrix(tariff ~ age_bin + gender_bin + educ_bin + race_bin, data=ces_full)
Y <- ces_full$tariff
```

Next placeholders for the samples (we just need the betas)

```
n_iter <- 2000
burnin <- 500
beta_mcmc <- matrix(nrow=n_iter+burnin, ncol=ncol(X_mat))
beta_mcmc[1,] <- rep(0, ncol(X_mat)) # Starting values are zeroes
```

Finally, define the normal prior

```
b_0 <- rep(0, ncol(X_mat))
B_0 <- diag(rep(1/10, ncol(X_mat)))
```

Now to run the MCMC sampler

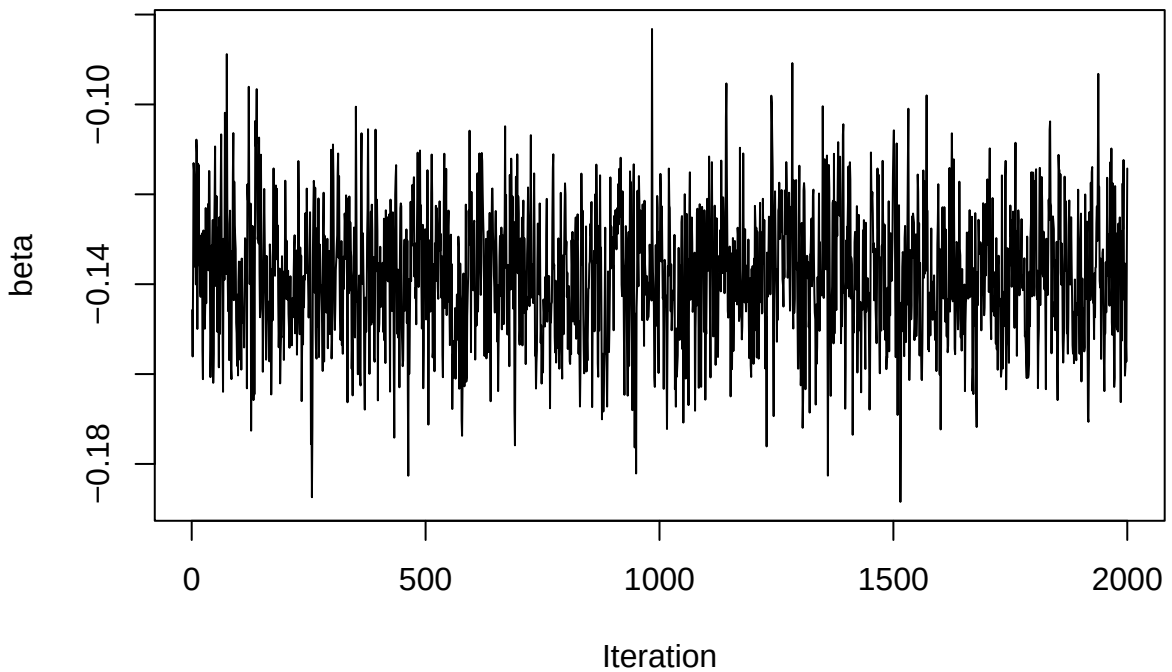
```r
set.seed(60637)
for(i in 1:(n_iter + burnin - 1)){
  # Step 1 - sample the latent variables Y^*
  linpred <- X_mat%*%beta_mcmc[i,] # Linear predictor at the current value of the betas
  Y_star <- rep(NA, length(Y))
  Y_star[Y==1] <- truncnorm::rtruncnorm(n=sum(Y==1), a=0, b=Inf, mean=linpred[Y==1], sd=1)
  Y_star[Y==0] <- truncnorm::rtruncnorm(n=sum(Y==0), a=-Inf, b=0, mean=linpred[Y==0], sd=1)

  # Step 2 - sample the betas given Y^*
  V = solve(B_0 + t(X_mat)%*%X_mat) # Bayesian regression variance
  M = V%*%(B_0%*%b_0 + t(X_mat)%*%Y_star) # Bayesian regression mean
  beta_mcmc[i+1,] <- MASS::mvrnorm(1, mu=M, Sigma=V) # Multivariate normal
}
# Throw out the burn-in iterations
beta_mcmc <- beta_mcmc[(burnin+1):(burnin+n_iter),]

# Name the columns of our betas
colnames(beta_mcmc) <- colnames(X_mat)
```

Part F

Make some traceplots of your coefficients - does it look like the Gibbs Sampling chain has sufficiently "mixed"?

Let's look at the one we care about, College Graduate

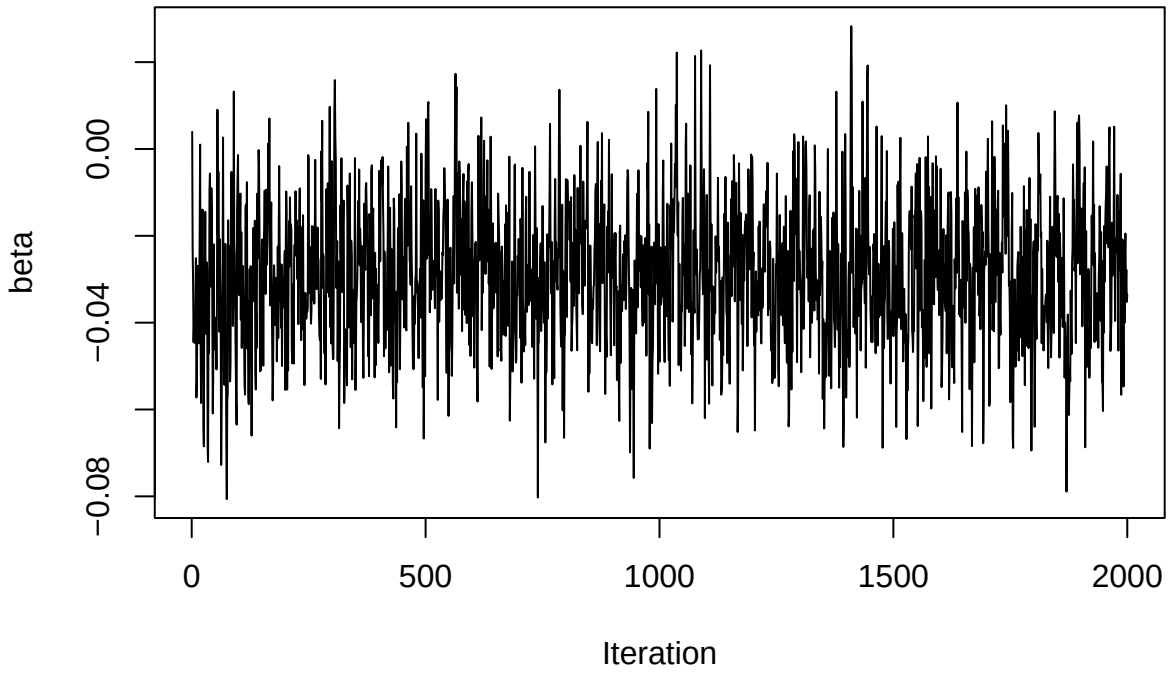
```r
plot(x=1:n_iter, y=beta_mcmc[,"educ_binCollege degree"], type="l", xlab="Iteration", ylab="beta")
```



Looks reasonable - some autocorrelation in samples, but there's no clear and obvious **trend** suggesting the markov chain has reached the stationary distribution.

How about the intercept?

```
plot(x=1:n_iter, y=beta_mcmc[,"(Intercept)"], type="l", xlab="Iteration", ylab="beta")
```



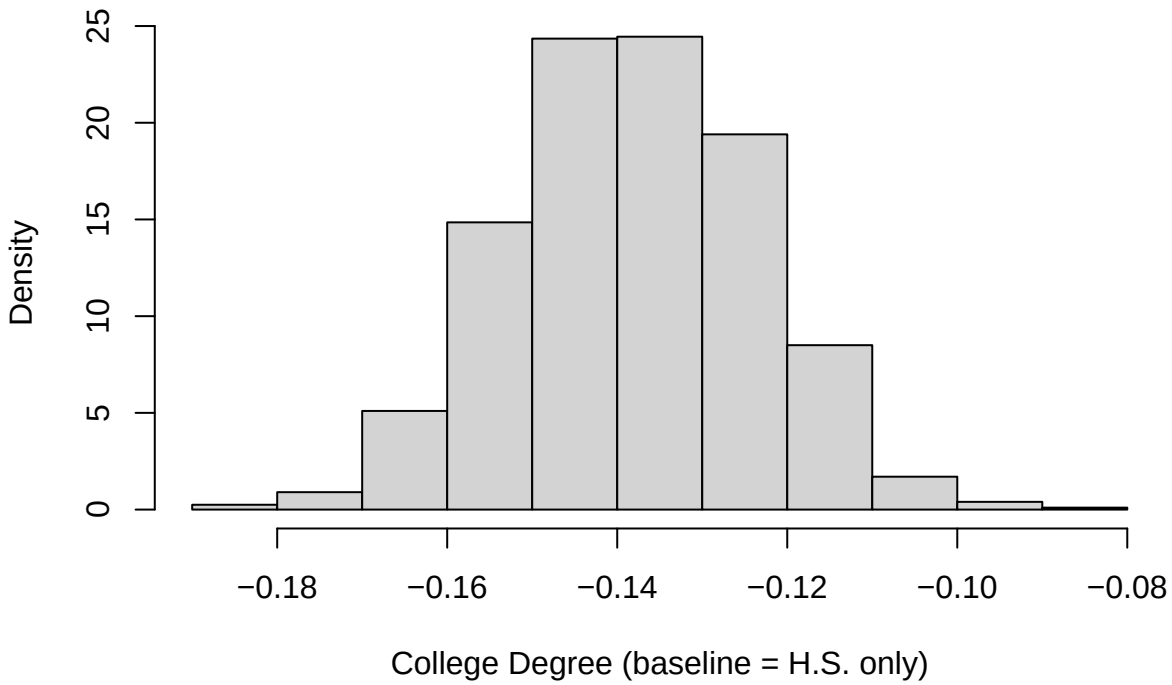
Similar patterns - looks like we're sampling from the target posterior.

Part G

Plot the posterior distribution of the coefficient on "College Graduate". Obtain a posterior mean estimate along with a 95% credible interval. Interpret your results substantively.

The posterior distribution of the coefficient on "College Graduate" is

```
hist(beta_mcmc[,"educ_binCollege degree"], xlab="College Degree (baseline = H.S. only)", main="", freq=
```




```
pm_probit <- colMeans(beta_mcmc)
pm_probit["educ_binCollege degree"]
```

```
## educ_binCollege degree
##                 -0.138
```

```
ci_probit_95 <- apply(beta_mcmc, 2, function(x) quantile(x, c(.025, .975)))
ci_probit_95[,"educ_binCollege degree"]
```

```
##   2.5%  97.5%
## -0.166 -0.111
```

The posterior mean estimate of the coefficient on "College Degree" is -0.138 with a 95% equal-tailed credible interval of $(-0.166, -0.111)$. We find strong evidence that college graduates are on average less likely to support the tariff than individuals with only a high school diploma or less.

The difference in probabilities depends on the values of the other covariates, but we can compare the differences in predicted probability holding those other covariates fixed at some level.

For example, our posterior mean predicted probability that a White, 30-44, Male respondent with a college degree will support the tariff is 48.6%

```
cov_1 <- c(1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0)
pnorm(cov_1%*%pm_probit)
```

```
##       [,1]
## [1,] 0.486
```

An individual with the same characteristics but **no** degree has a 54% chance of supporting the tariff.

```
cov_2 <- c(1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
pnorm(cov_2%*%pm_probit)
```

```
##       [,1]
## [1,] 0.541
```

Part H

Carry out a posterior predictive check. For each observation in the data, predict $\tilde{Y}_i$ given β across all of the sampled values of β (derive the probability that $Y_i = 1$ given the coefficients and the functional form and simulate from the bernoulli distribution). How well does the model predict the outcomes on average?

For each iteration, obtain the predicted probabilities.

Recall that for the probit

$$Pr(Y_i = 1|X_i, \beta) = \Phi(X_i'\beta)$$

where Φ is the standard normal CDF (implemented in `pnorm`)

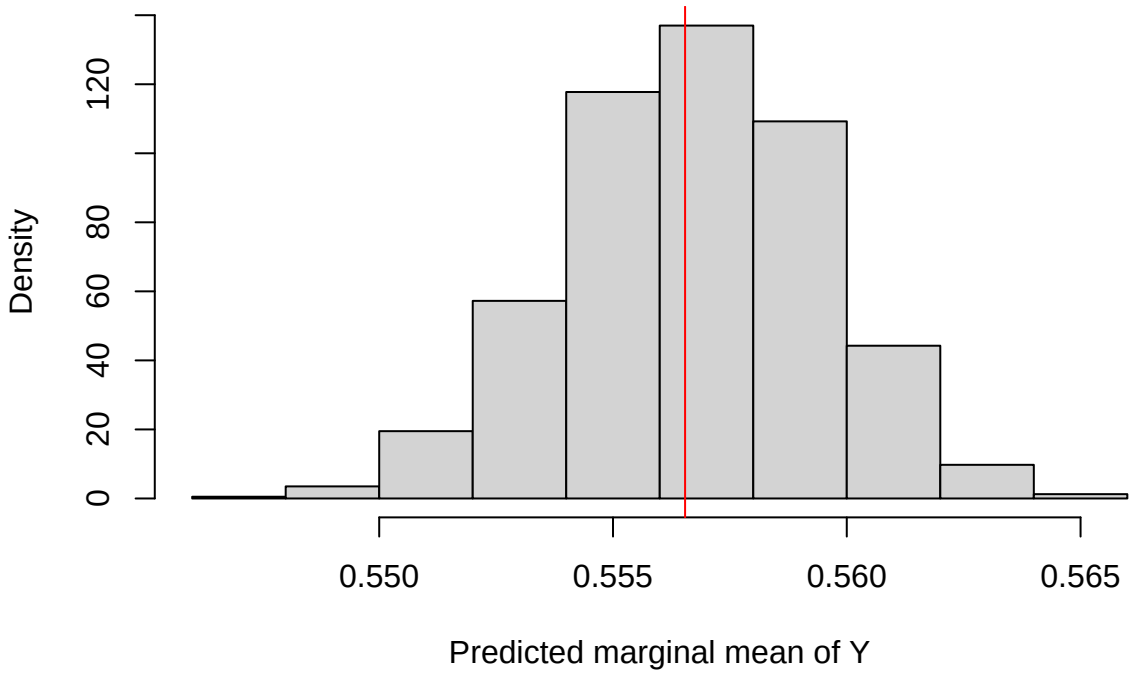
```
pred_prob <- apply(beta_mcmc, 1, function(x) pnorm(X_mat%*%x))
```

For each column of predicted probabilities, flip coins

```
predicted_y <- apply(pred_prob, 2, function(x) rbinom(length(x), 1, x))
```

Now here there are a few ways we can examine the model (so here you have a lot of options and can play around with different diagnostics). But a quick one would be to see if we replicate the distribution of Y correctly. For a binary indicator, the outcome distribution is just the share of 1s in-sample - so the analogue to our density plot figure would be something like this:

```
hist(colMeans(predicted_y), xlab="Predicted marginal mean of Y", main="", freq=F)
abline(v=mean(Y), col="red")
```



In general, it looks like the model does predict the in-sample data reasonably well - we consistently predict a marginal probability of support around .55

You could also do something with the coverage of the 95% credible intervals, etc... In general, with posterior predictive checks you just want to make sure the model is replicating some feature of the in-sample distribution.

Part I

Implement the model above in Stan without the data augmentation step (define y conventionally using the normal CDF `Phi`). Compare your results to your Gibbs sampler. Do they line up?

Load the package

```
library(rstan)
```

Our Stan model code

```
stan_probit <- "
data {
  int N; // number of observations
  int K; // number of covariates
  matrix[N, K] X; //covariate matrix
  array[N] int<lower=0, upper=1> y; //outcome vector
}
parameters {
  vector[K] beta; //regression coefficients
}
model {
  beta ~ normal(0, 10); // multivariate normal prior
  y ~ bernoulli(Phi(X * beta));
}
"
```

Our data list

```
stan_data <- list(N=nrow(X_mat), K=ncol(X_mat), y=Y, X=X_mat)
```

And run Stan (suppress the messages).

```
probit_hmc <- stan(
  model_code = stan_probit,  # Stan code
  data = stan_data,    # named list of data
  chains = 4,             # number of Markov chains
  warmup = 500,          # number of warmup iterations per chain
  iter = 2500,            # total number of iterations per chain
  cores = 4,  # number of cores (could use one per chain - by default uses however many you have   # no
  refresh = 0,
  seed = 60637
  )
```

Print the results.

```
print(probit_hmc)
```

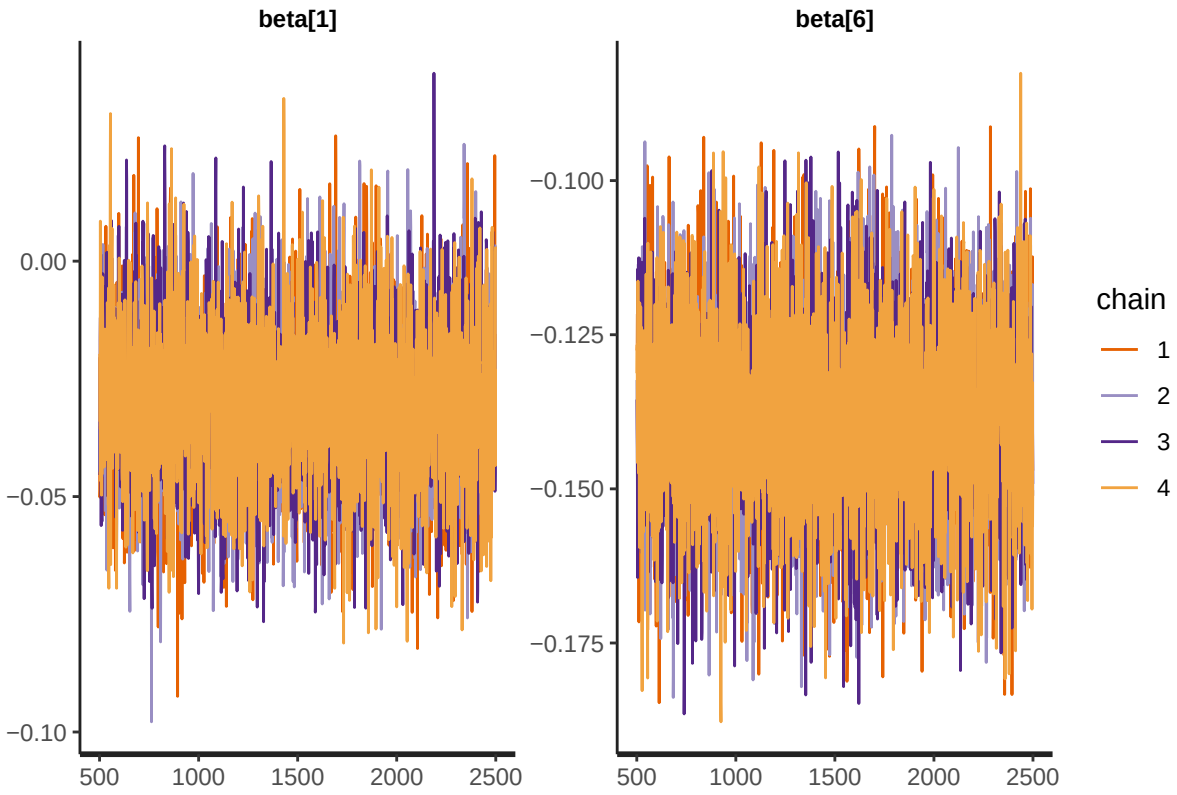
```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=2500; warmup=500; thin=1;
## post-warmup draws per chain=2000, total post-warmup draws=8000.
##
##              mean se_mean   sd      2.5%       25%       50%       75%
## beta[1]     -0.03    0.00 0.02     -0.06     -0.04     -0.03     -0.02
## beta[2]      0.13    0.00 0.02      0.10      0.12      0.13      0.14
## beta[3]      0.33    0.00 0.01      0.30      0.32      0.33      0.34
## beta[4]      0.49    0.00 0.02      0.46      0.48      0.49      0.50
```

```
## beta[5]       0.03    0.00 0.01      0.01      0.02      0.03      0.04
## beta[6]      -0.14    0.00 0.01     -0.17     -0.15     -0.14     -0.13
## beta[7]      -0.33    0.00 0.02     -0.36     -0.34     -0.33     -0.32
## beta[8]      -0.01    0.00 0.01     -0.03     -0.02     -0.01      0.00
## beta[9]      -0.10    0.00 0.03     -0.16     -0.12     -0.10     -0.08
## beta[10]     -0.12    0.00 0.02     -0.15     -0.13     -0.12     -0.11
## beta[11]      0.04    0.00 0.02      0.01      0.03      0.04      0.05
## beta[12]     -0.05    0.00 0.03     -0.10     -0.06     -0.05     -0.03
## lp__     -40663.68    0.04 2.43 -40669.20 -40665.12 -40663.33 -40661.92
##              97.5% n_eff Rhat
## beta[1]       0.00  4115    1
## beta[2]       0.16  5624    1
## beta[3]       0.36  5060    1
## beta[4]       0.52  5678    1
## beta[5]       0.05 11212    1
## beta[6]      -0.11  6865    1
## beta[7]      -0.30  7057    1
## beta[8]       0.02  6777    1
## beta[9]      -0.04 10185    1
## beta[10]     -0.09 11608    1
## beta[11]      0.07  8984    1
## beta[12]      0.01 11858    1
## lp__     -40659.90  3033    1
##
## Samples were drawn using NUTS(diag_e) at Tue Jan 31 22:10:06 2023.
## 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).
```

Our posterior mean and credible interval for `beta[6]` (college education) is very similar to the results from our "by-hand" gibbs sampler. We get a posterior mean of .14 and a 95% CI of $(.17, .11)$ which is equivalent (within rounding error) to our results from above

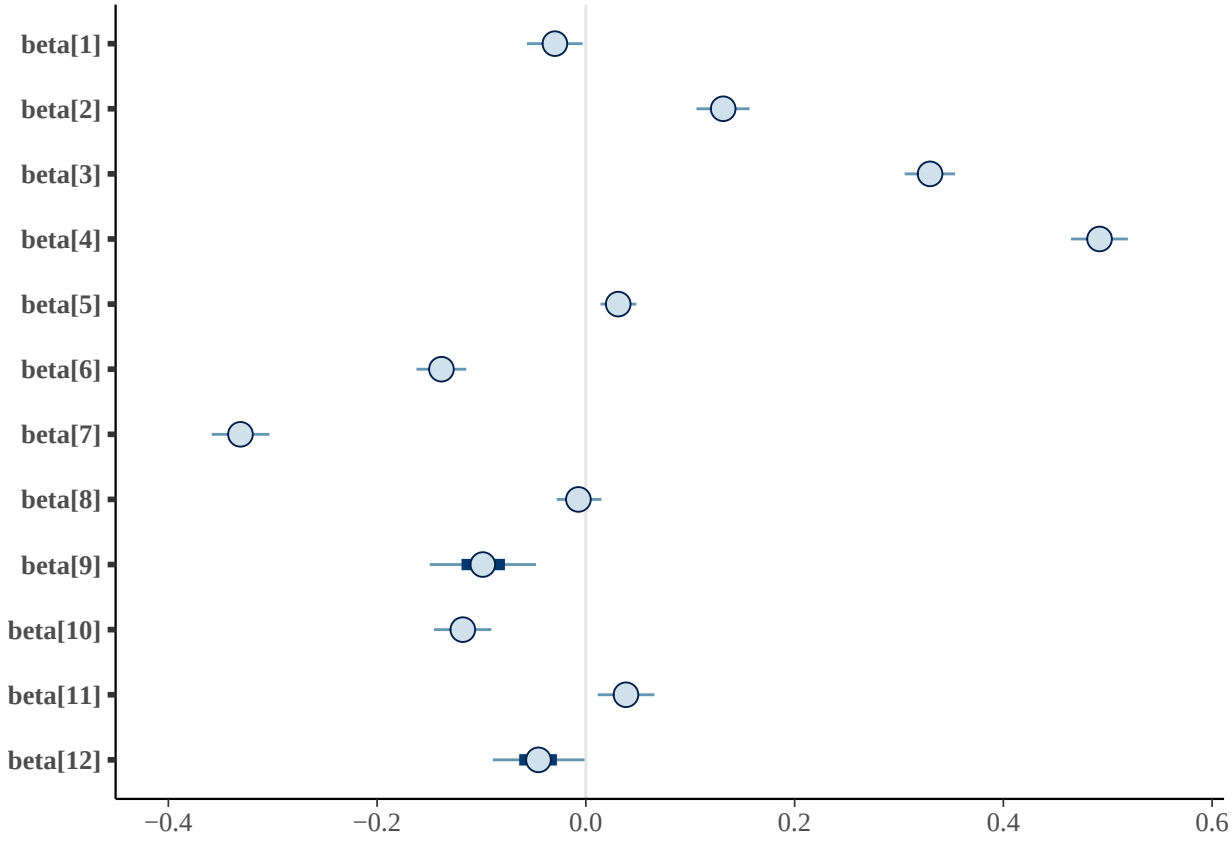
And then let's see if we've converged? Choose a few parameters to do traceplots

```
traceplot(probit_hmc, pars=c("beta[1]", "beta[6]"))
```



And let's plot the coefficients distributions!

```
bayesplot::mcmc_intervals(probit_hmc, pars = str_c("beta[", 1:12,"]"))
```


beta[1]
beta[2]
beta[3]
beta[4]
beta[5]
beta[6]
beta[7]
beta[8]
beta[9]
beta[10]
beta[11]
beta[12]
−0.4
−0.2
0.0
0.2
0.4
0.6