STA365 Homework 4

Yun-Hsiang Chan

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.5
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(rstan)
```

```
## Loading required package: StanHeaders

## rstan (Version 2.19.2, GitRev: 2e1f913d3ca3)

## 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 improved execution time, we recommend calling
## Sys.setenv(LOCAL_CPPFLAGS = '-march=native')
## although this causes Stan to throw an error on a few processors.

##
## Attaching package: 'rstan'

## The following object is masked from 'package:tidyr':
##
##     extract
```

```
survey <- read_csv("survey.csv")
```

```
## Parsed with column specification:
## cols(
##   cat_pref = col_double(),
```

```
##   male = col_double(),
##   age = col_double(),
##   eth = col_double(),
##   income = col_double(),
##   state = col_double(),
##   id = col_double()
## )
```

```
poststrat <- read_csv("poststrat.csv")
```

```
## Parsed with column specification:
## cols(
##   male = col_double(),
##   eth = col_double(),
##   age = col_double(),
##   income = col_double(),
##   state = col_double(),
##   N = col_double()
## )
```

```
head(survey)
```

```
## # A tibble: 6 x 7
##   cat_pref  male   age   eth income state    id
##      <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1        1     0     7     3      1    13     1
## 2        1     0     7     2      1    37     2
## 3        1     1     5     3      2    45     3
## 4        1     1     7     1      1     1     4
## 5        0     1     5     1      3    12     5
## 6        1     0     6     3      3    14     6
```

```
head(poststrat)
```

```
## # A tibble: 6 x 6
##    male   eth   age income state      N
##   <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>
## 1     0     1     1      1     1  94877
## 2     0     1     1      1     2 156645
## 3     0     1     1      1     3 137803
## 4     0     1     1      1     4 141987
## 5     0     1     1      1     5 121577
## 6     0     1     1      1     6  93574
```

```
data {
  int<lower = 0> n_survey;
  int<lower = 0> n_age;
  int<lower = 0> n_eth;
  int<lower = 0> n_state;
  int yes[n_survey];
  vector[n_survey] male;
```

```
  int age[n_survey];
  int eth[n_survey];
  int state[n_survey];

  // Bit for poststratification
  int<lower = 0> n_pred;
  vector[n_pred] male_pred;
  int age_pred[n_pred];
  int eth_pred[n_pred];
  int state_pred[n_pred];
  int N_in_cell_pred[n_pred];
}
parameters {
  real mu;
  real beta;
  vector[n_age] z_age;
  vector[n_eth] z_eth;
  vector[n_state] z_state;
  real<lower = 0> tau_age;
  real<lower = 0> tau_eth;
  real<lower = 0> tau_state;
}
transformed parameters {
  vector[n_age] alpha_age = tau_age * z_age;
  vector[n_eth] alpha_eth = tau_eth * z_eth;
  vector[n_state] alpha_state = tau_state * z_state;
}
model {
  yes ~ binomial_logit(1, mu + beta*male + alpha_age[age] + alpha_eth[eth] + alpha_state[state]);
  z_age ~ normal(0,1);
  z_eth ~ normal(0,1);
  z_state ~ normal(0, 1);
  tau_age ~ normal(0,1);
  tau_eth ~ normal(0,1);
  tau_state ~ normal(0,1);
  mu ~ normal(0,1);
  beta ~ normal(0,1);
}
generated quantities {
  int yes_pred[n_pred];
  for (n in 1:n_pred) {
    yes_pred[n] = binomial_rng(N_in_cell_pred[n],
   1.0/(1.0 + exp(-(mu + beta*male_pred[n] + alpha_age[age_pred[n]] + alpha_eth[eth_pred[n]] + alpha_sta
  }
}
```

```r
stan_data <- list(
  n_survey = length(survey$cat_pref),
  n_age = length(unique(poststrat$age)),
  n_eth = length(unique(poststrat$eth)),
  n_state = length(unique(poststrat$state)),
  yes = survey$cat_pref,
  male = survey$male,
  age = survey$age,
```

```
  eth = survey$eth,
  state = survey$state,
  n_pred = length(poststrat$male),
  male_pred = poststrat$male,
  age_pred = poststrat$age,
  eth_pred = poststrat$eth,
  state_pred = poststrat$state,
  N_in_cell_pred = poststrat$N
)

fit <- sampling(model_code, data = stan_data)
```

```
##
## SAMPLING FOR MODEL '92149621de19ac7bb3f6179047ab0e0b' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.001 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 10 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 16.256 seconds (Warm-up)
## Chain 1:                13.5 seconds (Sampling)
## Chain 1:                29.756 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '92149621de19ac7bb3f6179047ab0e0b' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
```

```
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 16.458 seconds (Warm-up)
## Chain 2:                13.325 seconds (Sampling)
## Chain 2:                29.783 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '92149621de19ac7bb3f6179047ab0e0b' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.001 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 10 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 16.256 seconds (Warm-up)
## Chain 3:                13.893 seconds (Sampling)
## Chain 3:                30.149 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '92149621de19ac7bb3f6179047ab0e0b' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.001 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 10 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
```

```
## Chain 4:
## Chain 4:  Elapsed Time: 16.809 seconds (Warm-up)
## Chain 4:                13.553 seconds (Sampling)
## Chain 4:                30.362 seconds (Total)
## Chain 4:
```

```
## Warning: There were 4 divergent transitions after warmup. Increasing adapt_delta above 0.8 may help.
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```r
## Proportion of people saying yes
n_pop <- sum(poststrat$N)
yes <- rstan::extract(fit, "yes_pred")
#This gives a 4000*6300 matrix so each row is a sample from the posterior predictive!
prop <- rowSums(yes$yes_pred) / n_pop
hist(prop, breaks = 30)
```