GR5065 Assignment 5

Juan Lopez Martin

1 American Family Survey

1.1 Adjusting for Party Identification or Related Variables

Our base regression will try to predict presidential approval based on age, race, gender, and education level. Note that the baseline levels for the categorical variables (race and gender) will be white and male.

```
library(haven)
AFS <- as_factor(read_dta("Data for Release 2018.DTA"))
```

```
AFS$age <- 2018 - AFS$birthyr
AFS$presvote16post <- as.character(AFS$presvote16post)
AFS$presvote16post <- ifelse(AFS$presvote16post %in% c("Gary Johnson", "Jill Stein",
                                                       "Evan McMullin", "Other"),
                             "Other", AFS$presvote16post)

AFS_clean <- AFS %>%
  select(age, race, gender, educ, app_dtrmp, pid7, presvote16post, inputstate) %>%
  mutate(age = as.numeric(replace(age, age %in% c("skipped", "not asked"), NA)),
         race = factor(replace(race, race %in% c("skipped", "not asked"), NA)),
         gender = factor(replace(gender, gender %in% c("skipped", "not asked"), NA)),
         educ = as.integer(factor(educ, ordered = TRUE, levels = c("No HS",
                                                                   "HS",
                                                                   "High school graduate",
                                                                   "Some college",
                                                                   "2-year",
                                                                   "4-year",
                                                                   "Post-grad"))),
         pid = as.integer(factor(pid7, ordered = TRUE, levels = c("Strong Democrat",
                                                                  "Not very strong Democrat",
                                                                  "Lean Democrat",
                                                                  "Independent",
                                                                  "Lean Republican",
                                                                  "Not very strong Republican",
                                                                  "Strong Republican"))),
         trump = factor(app_dtrmp, ordered = TRUE, levels = c("Strongly disapprove",
                                                              "Somewhat disapprove",
                                                              "Somewhat approve",
                                                              "Strongly approve"))) %>%
  na.omit()
```

The first simple model does not include any multilevel structure or party ID variable.

```
fit1 <- brm(trump ~ 1 + age + race + gender + educ,
            data = AFS_clean, family = cumulative(link = "logit"),
```

```
            prior = prior(normal(0, 2), class = "b") +
            prior(normal(0, 10), class = "Intercept"))
print(fixef(fit1), digits = 1)
```

```
##                    Estimate Est.Error   Q2.5 Q97.5
## Intercept[1]          -0.51     0.162 -0.829 -0.19
## Intercept[2]          -0.08     0.162 -0.391  0.24
## Intercept[3]           0.71     0.163  0.385  1.04
## age                    0.01     0.002  0.009  0.02
## raceBlack             -1.36     0.137 -1.626 -1.09
## raceHispanic          -0.42     0.114 -0.647 -0.20
## raceAsian             -0.26     0.210 -0.665  0.14
## raceNativeAmerican     0.51     0.424 -0.311  1.35
## raceMixed             -0.76     0.224 -1.201 -0.33
## raceOther              0.51     0.296 -0.086  1.09
## raceMiddleEastern     -1.06     0.979 -3.153  0.74
## genderFemale          -0.47     0.071 -0.617 -0.33
## educ                  -0.14     0.024 -0.185 -0.09
```

The second model allows the cutpoints and the effect of age, race, gender, and education to vary by state.

```
fit2 <- brm(trump ~ 1 + age + race + gender + educ +
              (1 + age + race + gender + educ | inputstate),
        data = AFS_clean, family = cumulative(link = "logit"),
        prior = prior(normal(0, 2), class = "b") + prior(normal(0, 10), class = "Intercept") +
          prior(exponential(0.5), class = "sd") + prior(lkj_corr_cholesky(1), class = "L"),
        chains = 6, cores = 6, iter = 2000, control = list(adapt_delta = 0.9))
print(fixef(fit2), digits = 1)
```

```
##                    Estimate Est.Error   Q2.5 Q97.5
## Intercept[1]          -0.51     0.164 -0.837 -0.20
## Intercept[2]          -0.07     0.164 -0.391  0.24
## Intercept[3]           0.73     0.166  0.404  1.05
## age                    0.01     0.002  0.009  0.02
## raceBlack             -1.41     0.152 -1.708 -1.12
## raceHispanic          -0.45     0.142 -0.740 -0.18
## raceAsian             -0.26     0.241 -0.768  0.20
## raceNativeAmerican     0.50     0.746 -1.029  1.97
## raceMixed             -0.79     0.262 -1.325 -0.30
## raceOther              0.59     0.372 -0.117  1.35
## raceMiddleEastern     -1.13     1.350 -3.921  1.51
## genderFemale          -0.47     0.087 -0.642 -0.30
## educ                  -0.14     0.024 -0.188 -0.09
```

The third model includes the same predictors as the second, adding the self-reported party leaning. For simplicity, we do not allow this coefficient to vary by state.

```
fit3 <- brm(trump ~ 1 + age + race + gender + educ +
              (1 + age + race + gender + educ | inputstate) + pid,
            data = AFS_clean, family = cumulative(link = "logit"),
            prior = prior(normal(0, 2), class = "b") + prior(normal(0, 10), class = "Intercept") +
              prior(exponential(0.5), class = "sd") + prior(lkj_corr_cholesky(1), class = "L"),
            chains = 6, cores = 6, iter = 2000, control = list(adapt_delta = 0.9))
print(fixef(fit3), digits = 1)
```

```
##                    Estimate Est.Error   Q2.5 Q97.5
```

```
## Intercept[1]          3.099     0.220  2.671  3.53
## Intercept[2]          3.877     0.225  3.440  4.30
## Intercept[3]          5.192     0.240  4.726  5.65
## age                   0.013     0.003  0.008  0.02
## raceBlack            -0.166     0.169 -0.500  0.16
## raceHispanic         -0.009     0.167 -0.333  0.32
## raceAsian            -0.256     0.267 -0.786  0.25
## raceNativeAmerican    1.243     0.828 -0.452  2.93
## raceMixed            -0.435     0.269 -0.978  0.06
## raceOther             0.830     0.359  0.135  1.55
## raceMiddleEastern    -0.545     1.496 -3.702  2.25
## genderFemale         -0.312     0.092 -0.490 -0.13
## educ                 -0.092     0.028 -0.147 -0.04
## pid                   0.868     0.027  0.816  0.92
```

The fourth model is the same as the third, but including the self-reported vote in the previous presidential election instead of party leaning. Note that the baseline category is no vote in the previous election.

```
fit4 <- brm(trump ~ 1 + age + race + gender + educ +
              (1 + age + race + gender + educ | inputstate) + presvote16post,
            data = AFS_clean, family = cumulative(link = "logit"),
            prior = prior(normal(0, 2), class = "b") + prior(normal(0, 10), class = "Intercept") +
              prior(exponential(0.5), class = "sd") + prior(lkj_corr_cholesky(1), class = "L"),
            chains = 6, cores = 6, iter = 2000, control = list(adapt_delta = 0.9))
print(fixef(fit4), digits = 1)
```

```
##                              Estimate Est.Error   Q2.5  Q97.5
## Intercept[1]                   -0.420     0.195 -8e-01 -0.040
## Intercept[2]                    0.441     0.195  6e-02  0.817
## Intercept[3]                    1.909     0.203  2e+00  2.309
## age                             0.006     0.003  6e-04  0.012
## raceBlack                      -0.418     0.194 -8e-01 -0.045
## raceHispanic                    0.133     0.172 -2e-01  0.488
## raceAsian                      -0.181     0.364 -9e-01  0.504
## raceNativeAmerican              0.894     0.579 -2e-01  2.052
## raceMixed                      -0.478     0.292 -1e+00  0.084
## raceOther                       0.270     0.386 -5e-01  1.037
## raceMiddleEastern              -0.315     1.503 -3e+00  2.505
## genderFemale                   -0.273     0.100 -5e-01 -0.072
## educ                           -0.052     0.030 -1e-01  0.008
## presvote16postDonaldTrump       2.577     0.131  2e+00  2.838
## presvote16postHillaryClinton   -2.298     0.137 -3e+00 -2.027
## presvote16postOther            -0.365     0.171 -7e-01 -0.027
```

The fifth and last model includes all the predictors mentioned previously. Again, the party leaning and past voting behavior are not allowed to vary by state.

```
fit5 <- brm(trump ~ 1 + age + race + gender + educ +
              (1 + age + race + gender + educ | inputstate) + pid + presvote16post,
            data = AFS_clean, family = cumulative(link = "logit"),
            prior = prior(normal(0, 2), class = "b") + prior(normal(0, 10), class = "Intercept") +
              prior(exponential(0.5), class = "sd") + prior(lkj_corr_cholesky(1), class = "L"),
            chains = 6, cores = 6, iter = 2000, control = list(adapt_delta = 0.9))
print(fixef(fit5), digits = 1)
```

```
##                              Estimate Est.Error   Q2.5 Q97.5
```

```
## Intercept[1]                    1.688     0.232  1.237  2.15
## Intercept[2]                    2.683     0.237  2.210  3.15
## Intercept[3]                    4.341     0.251  3.847  4.83
## age                             0.007     0.003  0.001  0.01
## raceBlack                       0.073     0.183 -0.292  0.43
## raceHispanic                    0.247     0.170 -0.084  0.59
## raceAsian                      -0.139     0.339 -0.837  0.51
## raceNativeAmerican              1.247     0.614 -0.017  2.46
## raceMixed                      -0.336     0.288 -0.908  0.21
## raceOther                       0.569     0.399 -0.182  1.38
## raceMiddleEastern              -0.186     1.547 -3.497  2.69
## genderFemale                   -0.278     0.106 -0.480 -0.06
## educ                           -0.048     0.031 -0.109  0.01
## pid                             0.552     0.030  0.492  0.61
## presvote16postDonaldTrump       1.818     0.133  1.554  2.07
## presvote16postHillaryClinton   -1.623     0.142 -1.905 -1.34
## presvote16postOther            -0.662     0.172 -0.995 -0.33
```

```
loo1 <- loo(fit1, cores = 6)
loo2 <- loo(fit2, cores = 6)
loo3 <- loo(fit3, cores = 6)
loo4 <- loo(fit4, cores = 6)
loo5 <- loo(fit5, cores = 6)
```

The ELPD for the five models make intuitive sense:

```
loo_compare(loo1, loo2, loo3, loo4, loo5)
```

```
##      elpd_diff se_diff
## fit5     0.0       0.0
## fit4  -177.1      19.8
## fit3  -279.1      23.3
## fit1 -1041.0      34.8
## fit2 -1041.4      35.1
```

As expected, the fifth model is the one with the best ELPD. This is not surprising given that presidental vote in the last elections and current political leaning should be strongly connected with approval of the current administration. However, we should remember that the estimated ELPD is only a measure of out-of-sample prediction accuracy; that is, it reflects how good the model will do when trying to predict the outcome variable in a new dataset. From a prediction perspective, this (or any other out-of-sample accuracy metric) is the only thing we should care about, as often happens in a machine learning context. However, in an inferential context this is not the only element we must consider. In experiments it is often said that adjusting for post-treatment covariates can poison causal estimates. That is: if including a predictor that refers to something that have been influenced by the treatment, the estimate of the treatment effect can be biased even with random assignment. This is because variance that should be attributed to the treatment can be attributed to this post-treatment or intermediate variable.

This is not an experiment, but the same reasoning applies. If we do not include a predictor that corresponds to ideological leaning or past voting behavior, the estimated coefficient for a given predictors in linear regression can be interpreted as the expected difference under the model between two people that differ in that variable but are equal in all the other covariates. For instance, the coefficient for black people should be interpreted as "comparing one black and one white person that have the same age, gender, and education, the black one has a X more/less probability of supporting Donald Trump than the white one". Of course, the model we are using is a ordinal logistic regression and thus the interpretation would be a bit more challenging due to the ordinality and non-linearity of the logit function, but the idea is the same. This should be used, for instance, if we wanted to get a general overview of the different factors that influence Smerican voters for choosing one

candidate over the other.

Meanwhile, including party identification or similar variables will make this new predictor capture a lot of the variance previously attributed to other predictors, also making the prediction much more accurate. Importantly, this implies the coefficient is interpreted with the party ID (or similar variable) held constant. That is, the coefficient for blacks will be interpreted as "comparing one black person and one white person that have the same age, gender, education, and party ID, the black one will have a X more/less probability of supporting Donald Trump than the white one". Note that the comparison is for two people with the same Party ID (or past voting behavior or any other similar variable). Therefore, this is more suited if we were interested in understanding the more specific dynamics of a particular election between certain candidates.

In this case, we are concerned about making predictions for the different states and therefore we will select the fifth model.

Lastly, note that the state-level heterogeneity for the five different models, although there does not seem to be a consistent pattern.

```
columnstoselect <- paste0("sd_inputstate__", c("age", "raceBlack", "raceHispanic","raceAsian",
                                               "raceNativeAmerican", "raceMixed", "raceOther",
                                               "raceMiddleEastern", "genderFemale", "educ"))

l2 <- colMeans(as.matrix(fit2)[,columnstoselect])
l3 <- colMeans(as.matrix(fit3)[,columnstoselect])
l4 <- colMeans(as.matrix(fit4)[,columnstoselect])
l5 <- colMeans(as.matrix(fit5)[,columnstoselect])
round(data.frame("Fit2" = l2, "Fit3" = l3, "Fit4" = l4, "Fit5" = l5), 3)
```

```
##                                   Fit2  Fit3  Fit4  Fit5
## sd_inputstate__age                0.003 0.002 0.002 0.002
## sd_inputstate__raceBlack          0.219 0.205 0.414 0.257
## sd_inputstate__raceHispanic       0.270 0.308 0.350 0.322
## sd_inputstate__raceAsian          0.304 0.359 1.039 0.801
## sd_inputstate__raceNativeAmerican 1.786 1.940 0.872 1.018
## sd_inputstate__raceMixed          0.390 0.317 0.365 0.319
## sd_inputstate__raceOther          0.721 0.448 0.527 0.489
## sd_inputstate__raceMiddleEastern  2.553 3.489 3.635 3.822
## sd_inputstate__genderFemale       0.200 0.167 0.245 0.231
## sd_inputstate__educ               0.026 0.027 0.030 0.032
```

1.2 Binary or Ordinal

```
AFS_clean$trump_binary <- (AFS_clean$trump %in% c("Strongly approve", "Somewhat approve")) * 1
```

We start fitting a Bernoulli model with the same predictors as in the fifth model in the previous section.

```
fit_b <- brm(trump_binary ~ 1 + age + race + gender + educ +
               (1 + age + race + gender + educ | inputstate) + pid + presvote16post,
             data = AFS_clean, family = bernoulli,
             prior = prior(normal(0, 2), class = "b") + prior(normal(0, 10), class = "Intercept")+
               prior(exponential(0.5), class = "sd") + prior(lkj_corr_cholesky(1), class = "L"),
             chains = 6, cores = 6, iter = 2000, control = list(adapt_delta = 0.9))

loob <- loo(fit_b)
```

I understand the code in section 1.2 of the assignment was incorrect, at least for my data. To get the correct loo estimate for the ordinal regression in the binary classification task we need to transpose the matrix that results from the pp_expect function which goes to the prob argument of dbinom. Then, it has to be

transposed again to be fed into loo (which takes a S by N matrix). I provided a minimal working example of this problem in thread #501 on CampusWire, but the main problem is that dbinom applies, by default, the transformation columnwise instead of rowise.

```
Pr <- pp_expect(fit5)
# 3 and 4 instead of 1 and 2 because I reversed the variable so higher numbers mean higher approval
lmatrix <- apply(Pr, MARGIN = 1:2, FUN = function(p) p[3] + p[4])
ll <- t(dbinom(x = AFS_clean$trump_binary, size = 1, log = TRUE, prob = t(lmatrix)))

looll <- loo(ll, cores = 6)
```

```
loo_compare(looll, loob)
```

```
##        elpd_diff se_diff
## model1  0.0       0.0
## fit_b  -5.6       4.0
```

It seems that the Bernoulli model performed slightly worse than the ordinal logistic regression on the binary classification task. This is not particularly surprising, considering that the ordinal logistic regression was based on a more detailed outcome (i.e. whether the responded approved/disapproved to a greater or smaller extent) that was simplified for the binary logistic regression. If the extra information is relevant it should performance,as it appears to happen in this case to a small degree. The tendency to remove (i.e. dichotomize) relevant information to choose a simpler model (i.e. logistic regression) is, in most cases, a bad idea that leads to worse models. Therefore, in this context I would prefer to model a more complex outcome variable such as this four-level approval or a feeling thermometer over the simpler dichotomous variable.

1.3 Predicting States

```
states <- unique(AFS_clean$inputstate)

fill1 <- data.frame(state = states, n = rep(NA, length(states)),
                    Pr_Model = rep(NA, length(states)))

for(i in 1:length(states)){
  rowinfo <- ((lmatrix>0.5)*1)[,AFS_clean$inputstate==states[i]]
  fill1$n[i] <- ncol(rowinfo)
  fill1$Pr_Model[i] <- round(sum(rowMeans(rowinfo)>0.4999)/6000, 2)
}

fill2 <- AFS_clean %>% group_by(inputstate) %>%
  summarise(Vote_Share_Data = round(mean(trump_binary), 2)) %>% select(Vote_Share_Data)

fill <- cbind(fill1, fill2)
```

Note that the predictions are quite imprecise, something that particularly happens in states with a very low sample size.

```
knitr::kable(fill)
```

state	n	Pr_Model	Vote_Share_Data
California	291	0.00	0.43
Wisconsin	54	0.00	0.67
New York	168	0.00	0.42
Florida	209	0.00	0.74
Ohio	100	0.00	0.36
North Carolina	71	0.00	0.38

state	n	Pr_Model	Vote_Share_Data
Washington	59	0.00	0.38
Georgia	83	0.00	0.45
Nebraska	17	0.01	0.50
Illinois	108	0.00	0.46
Missouri	57	0.98	0.41
Arizona	67	0.00	0.23
Indiana	53	0.00	0.56
Oregon	50	0.00	0.31
Nevada	30	0.00	0.34
Virginia	68	0.00	0.38
Louisiana	38	0.00	0.40
Tennessee	47	0.13	0.55
Iowa	29	0.00	0.39
Maine	9	0.16	0.78
Arkansas	19	1.00	0.36
South Dakota	7	1.00	0.27
Kentucky	51	0.93	0.45
South Carolina	44	0.00	0.45
Alabama	47	0.09	0.57
Michigan	58	0.00	0.47
Idaho	18	0.29	0.18
Maryland	44	0.00	0.29
Wyoming	11	0.05	0.40
Texas	192	0.00	0.38
Colorado	58	0.00	0.44
Massachusetts	48	0.00	0.50
Delaware	11	0.00	0.38
Pennsylvania	141	0.00	0.42
New Jersey	77	0.00	1.00
Montana	11	0.00	0.42
Utah	17	0.70	0.53
District of Columbia	12	0.00	0.32
New Mexico	22	1.00	0.40
Minnesota	53	0.04	0.50
Mississippi	21	0.02	0.34
Kansas	20	0.00	0.57
Connecticut	21	0.00	0.55
Vermont	5	0.00	0.40
Oklahoma	30	1.00	0.53
Hawaii	13	0.03	0.00
New Hampshire	8	0.02	0.34
West Virginia	20	0.71	0.42
Rhode Island	10	0.00	0.55
Alaska	3	1.00	0.48
North Dakota	2	1.00	0.45

For using the posterior predictions instead of the actual reported vote share we need to trust our model to a certain extent (i.e. a very simplistic model will probably not work). However, if the model is relatively good it should offer a clear improvement over the raw data. Note that the advantage of this model-based approach is that we can incorporate much more relevant information, such as past voting behavior, party ID, or the effects of multiple demographics. The use of a multilevel model should be able to also improve the predictions as it is able to partially pool relevant information on the effect of a certain predictor across

states. This property makes a better a more effective use of the information in the data than the alternative approach of fitting a separate model for each state – something that will not even be possible to do in this case given that some states have very few data points.

A further improvement, apart from adding other relevant predictors, will be to include postratification. That is, instead of assuming that the AFS survey is representative for each state, we could use census data to get more realistic estimates.

2 Discrimination in Police Stops

2.1 Prior preditive distribution

The model is summarized here for simplicity, and implemented in NC_rng.stan. The code is also shown below.

```
north_carolina <- readRDS("north_carolina.rds")
```

$R_{rd} \sim Binom(q_{rd}, S_{rd})$

- $q_{rd} = \phi_{rd} \frac{1-\text{dbeta}(t_{rd},\phi_{rd},\lambda_{rd}+1)}{\text{dbeta}(t_{rd},\phi_{rd},\lambda_{rd}}$

$S_{rd} \sim Binom(p_{rd}, n_{rd})$

- $p_{rd} = 1 - \text{dbeta}(t_{rd}, \phi_{rd}, \lambda_{rd})$
  - $t_{rd} \sim logit^{-1}(N(\mu_{tr}, \sigma_{tr}))$
    - $\mu_{tr} \sim N(0, 2)$
    - $\sigma_{tr} \sim N_+(0, 2)$
  - $\phi_{rd} = \text{logit}^{-1}(\phi_r + \phi_d)$
    - $\phi_r \sim N(\mu_{\phi_r}, \sigma_{\phi_r})$
      - $\mu_{\phi_r} \sim N(0, 2)$
      - $\sigma_{\phi_r} \sim N_+(0, 2)$
    - $\phi_d \sim N(\mu_{\phi_d}, \sigma_{\phi_d})$ or $\phi_d = 0$ when $d = 1$
      - $\mu_{\phi_d} \sim N(0, 2)$
      - $\sigma_{\phi_d} \sim N_+(0, 2)$
  - $\lambda_{rd} \sim \exp(\lambda_r + \lambda_d)$
    - $\lambda_r \sim N(\mu_{\lambda_r}, \sigma_{\lambda_r})$
      - $\mu_{\lambda_r} \sim N(0, 2)$
      - $\sigma_{\lambda_r} \sim N_+(0, 2)$
    - $\lambda_d \sim N(\mu_{\lambda_d}, \sigma_{\lambda_d})$ or $\lambda_d = 0$ when $d = 1$
      - $\mu_{\lambda_d} \sim N(0, 2)$
      - $\sigma_{\lambda_d} \sim N_+(0, 2)$

Note that the only change from the original model is that, consistent with what we have used previously in class, I'm using $abs(N(0, 2))$ instead of $N_+(0, 2))$ for the variance priors.

```
functions {
  int[ , , ] NC_rng(int D, int R, int[] Asian, int[] Black, int[] Hispanic, int[] White) {
    //matrix[D, R] data_matrix =  append_col(White, Black)
    int draws[D, R, 2] = rep_array(0, D, R, 2); // initialize with zeros
```

```
vector[R] phi_r;
vector[R] lambda_r;
vector[D] phi_d;
vector[D] lambda_d;

real mu_phi_r = normal_rng(0, 2);
real sigma_phi_r = fabs(normal_rng(0, 2));

real mu_phi_d = normal_rng(0, 2);
real sigma_phi_d = fabs(normal_rng(0, 2));

real mu_lambda_r = normal_rng(0, 2);
real sigma_lambda_r = fabs(normal_rng(0, 2));

real mu_lambda_d = normal_rng(0, 2);
real sigma_lambda_d = fabs(normal_rng(0, 2));

real mu_t_rd = normal_rng(0, 2);
real sigma_t_rd = fabs(normal_rng(0, 2));

for (r in 1:R){
  phi_r[r] = normal_rng(mu_phi_r, sigma_phi_r);
  lambda_r[r] = normal_rng(mu_lambda_r, sigma_lambda_r);
}

for (d in 1:D){
  phi_d[d] = normal_rng(mu_phi_d, sigma_phi_d);
  lambda_d[d] = normal_rng(mu_lambda_d, sigma_lambda_d);
}

phi_d[1] = 0;
lambda_d[1] = 0;

for (r in 1:R){
  for (d in 1:D){
    real phi_rd = inv_logit(phi_r[r] + phi_d[d]);
    real lambda_rd = exp(phi_r[r] + phi_d[d]);
    real t_rd = inv_logit(normal_rng(mu_t_rd, sigma_t_rd));

    real p_rd = 1 - beta_cdf(t_rd, phi_rd*lambda_rd, (1-phi_rd*lambda_rd));
    real q_rd;

    int n_rd;
    if (r==1)
        n_rd = Asian[d];
    else if (r==2)
        n_rd = Black[d];
    else if (r==3)
        n_rd = Hispanic[d];
    else if (r==4)
        n_rd = White[d];

    draws[d, r, 1] = binomial_rng(n_rd, p_rd);
```

```
        q_rd = phi_rd*(1 - beta_cdf(t_rd,
                                    phi_rd*(lambda_rd+1),
                                    (1-phi_rd*(lambda_rd+1)))) / beta_cdf(t_rd,phi_rd*lambda_rd, (1-phi
        draws[d, r, 2] = binomial_rng(draws[d, r, 1], q_rd);

      }
    }

    return draws;
  }
}
```

```
rstan::expose_stan_functions("NC_rng.stan")
PPD <- NC_rng(D = nrow(north_carolina), R = ncol(north_carolina),
              Asian = north_carolina[ , 1], Black = north_carolina[ , 2],
              Hispanic = north_carolina[ , 3], White = north_carolina[ , 4])
```

To check the implementation we can show the prior predictive distribution for the first department, the Charlotte-Mecklenburg Police Department.

```
north_carolina <- as.data.frame(north_carolina)
nc_s <- north_carolina
nc_r <- north_carolina

nc_s$Asian <- unlist(map(map(PPD, 1), 1))
nc_s$Black <- unlist(map(map(PPD, 2), 1))
nc_s$Hispanic <- unlist(map(map(PPD, 3), 1))
nc_s$White <- unlist(map(map(PPD, 4), 1))

nc_r$Asian <- unlist(map(map(PPD, 1), 2))
nc_r$Black <- unlist(map(map(PPD, 2), 2))
nc_r$Hispanic <- unlist(map(map(PPD, 3), 2))
nc_r$White <- unlist(map(map(PPD, 4), 2))

toprint <- rbind(north_carolina[1,], nc_s[1,], nc_r[1,])
titletoprint <- rownames(toprint)[1]
rownames(toprint) <- c("Total", "Searches", "Hits")
print(titletoprint)
```

```
## [1] "Charlotte-Mecklenburg Police Department"
```

```
knitr::kable(toprint)
```

	Asian	Black	Hispanic	White
Total	13599	419873	77727	282761
Searches	43	1974	536	575
Hits	1	92	8	3

2.2 Legal Analysis

If the legal claim is made against a particular police department, the defense attorney's argument is applicable. Multilevel models can, in theory, partially pool the coefficients towards the population (state) average. However, this effects tends to be small when there is a lot of data available, and in any case a model can be fit for each individual department (no pooling). If I was in this situation I will fit the no pooling models

just in case, comparing it to the multilevel model. Again, I expect that in most cases in which the data is rich-enough to make any legal claim the coefficient for the police department in the multilevel and no pooling model would be quite similar.

If the legal claim is made against the state there is no reason to worry about the information pooling across departments, as all of them are in North Carolina. Using a complete pooling model could be a simpler option, but it has the disadvantage of not accounting for the clear hierarchical structure of the data and not providing individual coefficients for each deparment.