Week 5: More Multilevel Models

PLSC 40502 - Statistical Models

Review

Previously

- Stan
  - **Hamiltonian Monte Carlo** - use the gradient of the likelihood $\times$ prior to get good M-H proposals
  - Avoids proposals that are either too auto-correlated and too likely to be rejected
  - Commmon structure to statistical modeling: `data`, `parameters`, `model`
  - Frequently used models have canned routines in `rstanarm`
- **Diagnosing model fit**
  - Posterior predictive checks
  - **Information criteria** vs. **Cross-validation**
  - log-predictive density as a measure of prediction quality (how much probability mass do we place on the "correct" outcome)

This week

- **Multilevel regression**
  - Extending the Bayesian GLM by adding additional **grouping** and structure
  - "Random effects" models/"Partial pooling"
- **Post-stratifiation**
  - Adjusting non-representative surveys to match population targets
  - Combining with multilevel regression: "MrP"
- **Dynamic regression**
  - Modeling dependence in the parameters over time
  - Autoregressive models
  - Latent factor models

Multilevel Models

Hierarchical Models

- In our simple normal regression from before, we assumed both a common mean and intercept for the data.
  - The underlying model is the same regardless of state.
- We also assumed that observations were completely exchangeable
  - $Y_i \sim \text{Normal}(X_i'\beta, \sigma^2)$ implies no residual correlation in the outcomes across **all** counties.
- This may be a poor modeling choice if states are very heterogeneous and information about the state in which a county is in would provide additional predictive power beyond knowing the 2016 presidential vote.
  - Additionally, **exchangeability** of observations may be violated -- unobserved factors may lead observations within a particular state to be correlated (e.g. if all counties in a region are exposed to some common shock).

Hierarchical Models

- **Hierarchical** or **Multilevel** linear models try to address this by incorporating additional structure on the regression parameters.
  - Suppose observations belong to a group $j \in \{1, 2, 3, \dots, J\}$ (e.g. states, time periods, schools, etc...)
  - We observe Y_{ij} as the outcome for unit i in grouping j.
- We can incorporate group membership into the data-generating process.

- Instead of a **common prior** on $\beta \sim \text{Normal}(b_0, B_0^{-1})$, we may do something like:

$$\beta_j \sim \text{Normal}(\mu_\beta, \Sigma_\beta)$$

$$Y_{ij} \sim \text{Normal}(X_{ij}'\beta_j, \sigma^2)$$

  - This model assumes that each group has it's own set of group-level coefficients β_j that are drawn from a common distribution centered at β.
- The choice of prior on β then dictates the degree of **pooling** across groupings.
  - Flat prior: **no pooling** - β_j are estimated entirely separately
  - Zero variance prior: complete pooling (all β_j are equal to β)

Hierarchical Models

- There are lots of choices about how we want to incorporate hierarchy into the model the coefficients across groupings
  - What levels of aggregation and how many?
  - Which parameters should be allowed to vary (all of the coefficients or just the intercepts)?
  - Where does the structure enter into the model (the means? the variances? the covariances?)

Revisiting: Predicting Elections

```
# Load the data
elections <- read_csv("data/us-house-wide.csv")

# Aggregate the house data to counties
elections_county <- elections %>% group_by(fipscode) %>% summarize(state=state[1], county=count
                                                                   total.votes = sum(total.vote
                                                                   dem = sum(dem))

# Merge in 2015 Presidential
pres_2016 <- read_csv("data/clinton_2016_vote.csv")
elections_county <- elections_county %>% left_join(pres_2016 %>% dplyr::select(county_fips, car
                                                   by=c(fipscode="county_fips"))

# Generate vote shares
elections_county$dem2018 <- elections_county$dem/elections_county$total.votes
elections_county$dem2016 <- elections_county$candidatevotes/elections_county$totalvotes

# Drop missing
elections_county <- elections_county %>% filter(!is.na(dem2018)&!is.na(dem2016))
```

```
X_mat <- model.matrix(dem2018 ~ dem2016, data=elections_county)
Y <- elections_county$dem2018
K <- ncol(X_mat) # Number of beta parameters
```

Application: Elections

- Let's implement the "Normal Regression" model in Stan

```
model_structure <- "
data {
  int N; // number of observations
  int K; // number of covariates
  matrix[N, K] X; //covariate matrix
  vector[N] y; //outcome vector
}
parameters {
  vector[K] beta; //regression coefficients
  real<lower = 0> sigma; // standard deviation
}
model {
  beta ~ normal(0, 3); // multivariate normal prior
  sigma ~ inv_gamma(0.001/2, 0.001/2); // inverse gamma
  y ~ normal(X * beta, sigma); // * is matrix multiplication if terms are matrices
}
generated quantities {
  array[N] real y_rep = normal_rng(X * beta, sigma);
  vector[N] log_lik;
  for (n in 1:N) log_lik[n] = normal_lpdf(y[n] | X[n, ] * beta, sigma);
}
"
```

Application: Elections

- Load the relevant packages

```
library(rstan)
```

- Pass the actual parameters as a list

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

Application: Elections

- Run Stan!

```
model_fit <- stan(
  model_code = model_structure,  # Stan code
  data = data_source,    # named list of data
  chains = 4,              # number of Markov chains
  warmup = 50,          # 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
  refresh = 0,          # no progress shown
  seed = 60637
  )
```

Leave-one-out CV

- Once specified, we can obtain an estimate of the expected log predictive accuracy

```
library(loo)
```

```
loo_reg <- loo(model_fit, pars="log_lik")
loo_reg$estimates
```

```
##           Estimate    SE
## elpd_loo     4254 115.1
## p_loo         220  66.8
## looic       -8507 230.1
```

Varying intercepts model

- Our regression from before is an example of a fully pooled regression.
  - Within individual states though, the regression line may be a poor predictor
  - The simplest fix is to allow the "intercept" to shift across units.
- Assume:

$$Y_{ij} \sim \text{Normal}(X_{ij}'\beta + \alpha_j, \sigma^2)$$

$$\alpha_j \sim \text{Normal}(0, \sigma_\alpha^2)$$

And keep the same priors as before on the β and σ_j^2 parameters. Here, we now omit the intercept from the betas.

- α_j can be interpreted as the group-specific "shift" in the intercept from the "grand" intercept β_0
  - An equivalent parameterization would be to remove the intercept from β and write $\alpha_j \sim Normal(\mu_\alpha, \Sigma_\alpha)$

Varying intercepts model

- Let's implement this in Stan - first, our `data` block

```
data{
  int N; // number of observations
  int J; // Number of groups
  int K; // number of covariates
  array[N] int J_i; // group membership indicator
  matrix[N, K] X; //matrix of covariates
  vector[N] y; //outcome
}
```

Varying intercepts model

- Next, our parameters.

```
parameters{
  vector[K] beta; // beta coefficients
  vector[J] alpha; // random intercepts
  real<lower = 0> sigma; // variance of outcome
  real<lower=0> sigma_a; // variance of intercepts
}
```

Varying intercepts model

- Finally, our model

```
model{
  beta ~ normal(0, 3); // normal prior on coefficients
  alpha ~ normal(0, sigma_a); // normal distribution on random intercepts
  sigma ~ inv_gamma(0.001/2, 0.001/2);
  sigma_a ~ inv_gamma(0.001/2, 0.001/2);
  y ~ normal(alpha[J_i] + X*beta, sigma);
}
```

Varying intercepts model

- Put it into Stan!

```
elections_county$state_num <- as.numeric(as.factor(elections_county$state))
varying_intercepts_data <- list(N = nrow(X_mat), J = length(unique(elections_county$state_num)),
                    J_i = elections_county$state_num,
                    K = ncol(X_mat), X=X_mat, y=Y)
```

```
model_var_intercept <- stan(
  model_code = varying_intercepts_model,  # Stan code
  data = varying_intercepts_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 (this is much slower if = 1)
  refresh = 0,           # no progress shown
  seed = 60637           # random seed
  )
```

Varying intercepts model

- Summarize the fit

```
print(model_var_intercept, pars = c("beta","sigma","alpha[1]"))
```

```
## 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% 97.5% n_eff Rhat
## beta[1]  0.04       0 0.01 0.03 0.04 0.04 0.04  0.05   524 1.01
## beta[2]  1.07       0 0.01 1.05 1.06 1.07 1.07  1.08  8641 1.00
## sigma    0.05       0 0.00 0.05 0.05 0.05 0.05  0.05  8763 1.00
## alpha[1] 0.03       0 0.01 0.01 0.02 0.03 0.03  0.04   940 1.00
##
## Samples were drawn using NUTS(diag_e) at Mon Feb  3 20:58:38 2025.
## 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).
```

Posterior predictive check

- Pull the `y_rep` from the model (in the form of a list)

```
y_ppc_varint <- rstan::extract(model_var_intercept)$y_rep
```

- Calculate the upper and lower credible intervals

```
y_ppc_ci_varint <- t(apply(y_ppc_varint, 2, function(x) quantile(x, c(.025, .975))))
```

- What's the share that cover the truth?

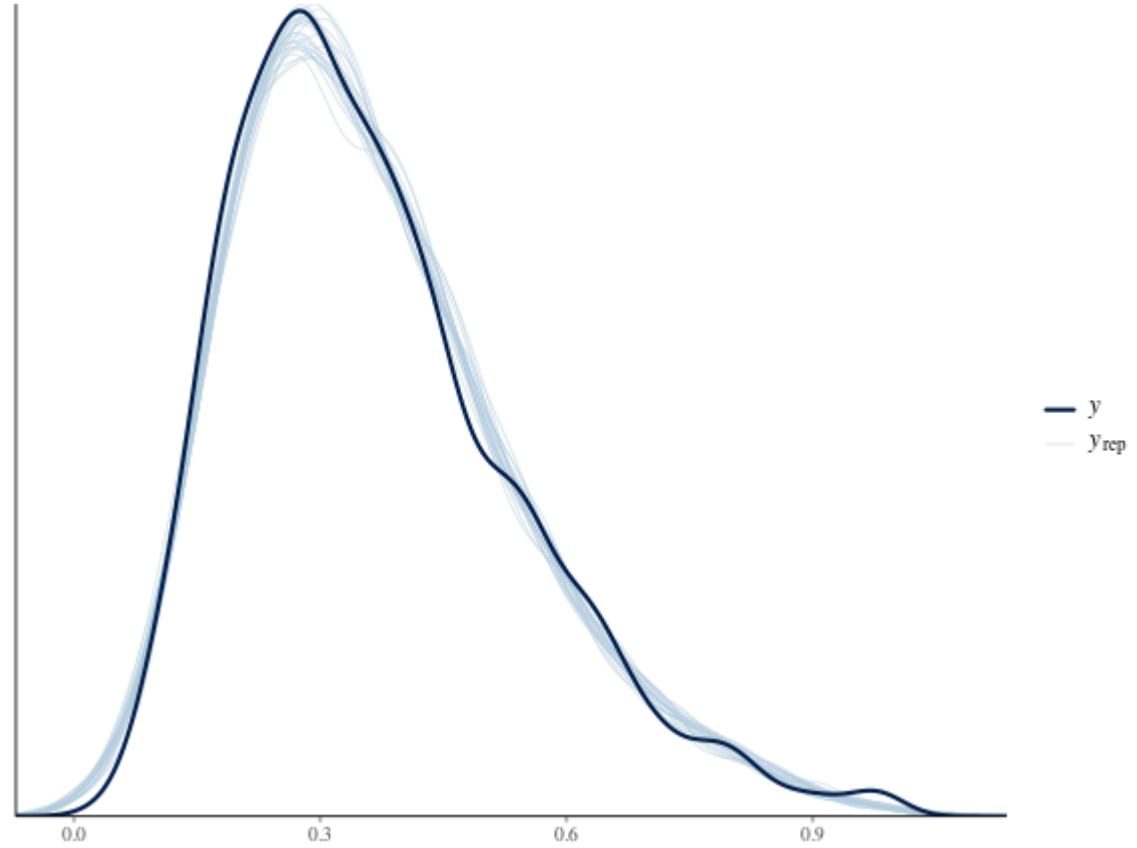
```
cover_95_varint <- y_ppc_ci_varint[,1]<Y&Y<y_ppc_ci_varint[,2]
mean(cover_95_varint)
```

```
## [1] 0.963
```

Posterior predictive check

- Empirical density vs. predicted

```
bayesplot::ppc_dens_overlay(y = elections_county$dem2018, yrep = y_ppc_varint[1:25,])
```



Posterior predictive check

- What's the root mean squared deviation from the truth?

```
sqd_error_varint <- apply((Y - t(y_ppc_varint))^2, 2, mean)
print(sqrt(mean(sqd_error_varint)))
```

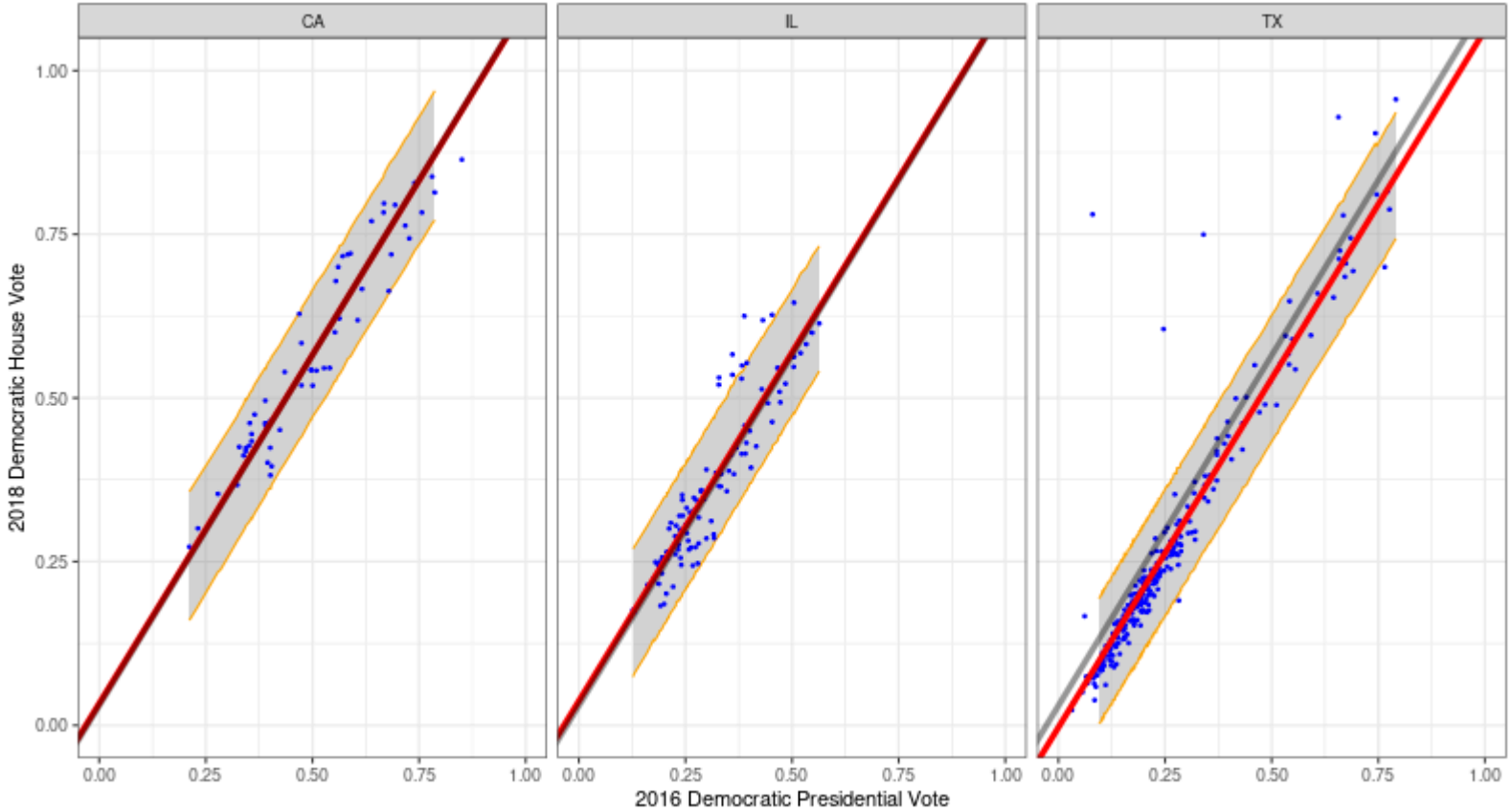
```
## [1] 0.0693
```

- Make a dataframe for the predictive "ribbon" plot

```
elections_county$lowerppdVI <- y_ppc_ci_varint[,1]
elections_county$upperppdVI <- y_ppc_ci_varint[,2]
```

Outcome plots

- Compare the two regression lines



Leave-one-out statistic

- Calculate the leave-one-out statistic

```
library(loo)
```

```
loo_varint <- loo(model_var_intercept, pars="log_lik")
loo_varint$estimates
```

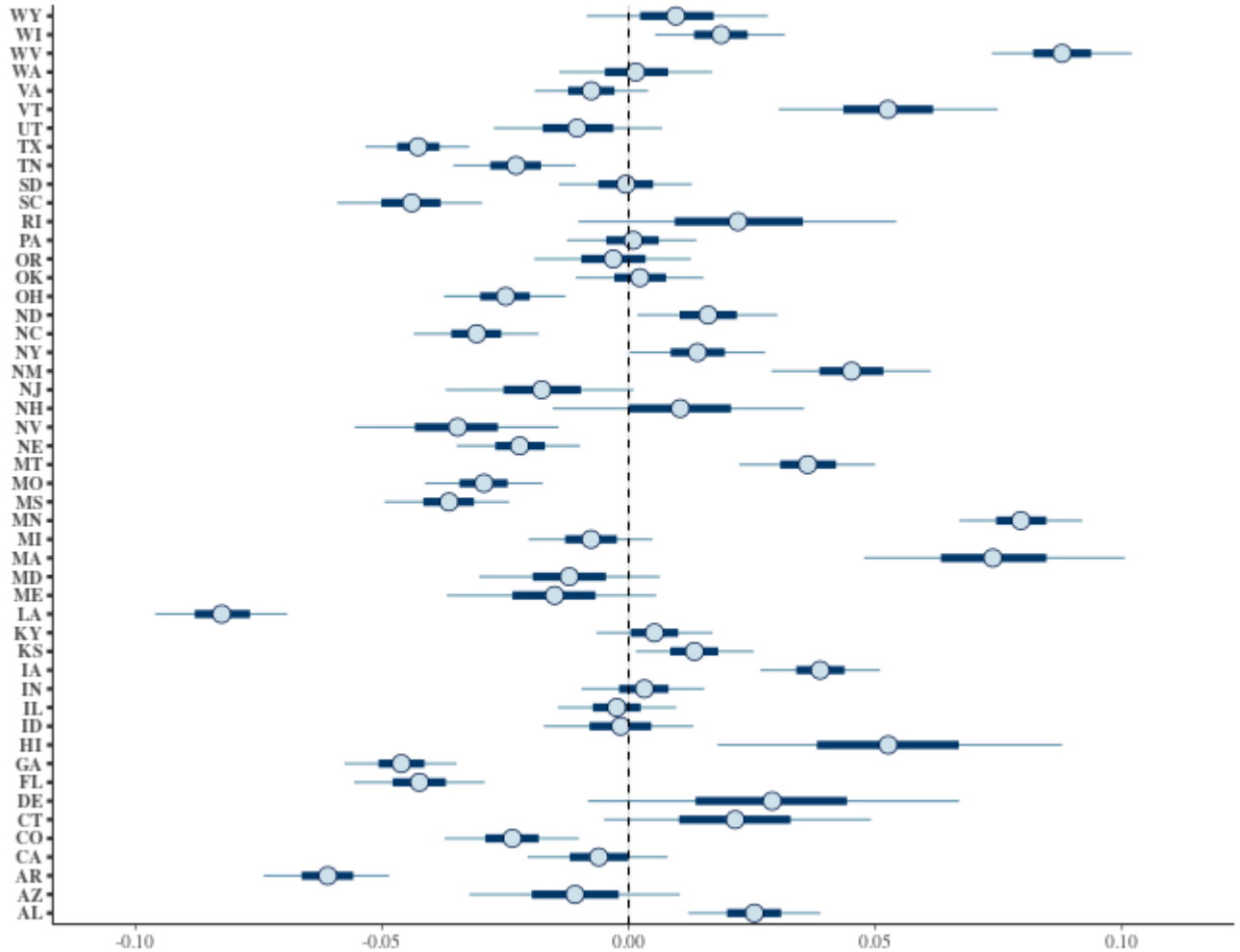
```
##           Estimate    SE
## elpd_loo   4854.7 162.4
## p_loo        61.4  10.1
## looic     -9709.3 324.7
```

```
loo_compare(loo_varint, loo_reg)
```

```
##        elpd_diff se_diff
## model1    0.0       0.0
## model2 -601.0      63.2
```

Varying intercepts model

- Plot the deviations from the "grand mean" intercept by state



Varying slopes model

- What if we allow all of the parameters to vary by group?

$$\beta_j \sim \text{Normal}(\mu_\beta, \Sigma_\beta)$$

$$Y_{ij} \sim \text{Normal}(X'_{ij}\beta_j, \sigma^2)$$

- This is a more general and flexible hierarchical model
  - But introduces a slight complication -- we now need a prior distribution on an entire **matrix** Σ_β (the variances and covariances of the parameters).
- The conjugate prior is the inverse-Wishart, the multivariate extension of the inverse-gamma.
  - An alternative non-conjugate prior (that appears to have better performance) is a scaled correlation matrix with an LKJ prior on the correlation matrix (see Stan documentation for more).
  - The latter is now preferred (especially by the Stan programmers)

Varying slopes model

- Our prior on Σ_β starts by decomposing Σ_β into a diagonal scaling matrix $\mathbf{diag}(\tau)$ and a correlation matrix Ω

$$\Sigma_\beta = \mathbf{diag}(\tau)\Omega\mathbf{diag}(\tau)$$

- For each τ_k we'll use a Half-Cauchy distribution (Cauchy with the constraint $\tau_k > 0$)
- For Ω, we'll use the Lewandowski-Kurowicka-Joe (LKJ) distribution which defines a distribution over symmetric positive definite matrices
  - **Intuition**: Similar to a beta distribution in multiple dimensions.

Varying slopes model

- Our `data` block

```
data{
  int N; // number of observations
  int J; // Number of groups
  int K; // number of covariates
  array[N] int J_i; // group membership indicator
  matrix[N, K] X; //matrix of covariates
  vector[N] y; //outcome
}
```

Varying slopes model

- Our `parameters` field

```
parameters{
  corr_matrix[K] Omega; //prior correlation
  vector<lower=0>[K] tau; //prior scale
  vector[K] mu_beta;
  array[J] vector[K] beta; //group-level coefficients
  real<lower = 0> sigma; // variance parameters
}
```

Varying slopes model

- And our `model`

```
model{
  tau ~ cauchy(0, 2.5);
  Omega ~ lkj_corr(2);
  sigma ~ inv_gamma(0.001/2, 0.001/2); //Inverse-gamma
  mu_beta ~ normal(0, 3);
  beta ~ multi_normal(mu_beta, quad_form_diag(Omega, tau));
  for (n in 1:N) {
    y[n] ~ normal(X[n] * beta[J_i[n]], sigma);
  }
}
```

Varying slopes model

- Run it in Stan

```
model_var_slope <- stan(
  model_code = varying_slopes,  # Stan code
  data = varying_intercepts_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,
  refresh=0,          # no progress shown
  seed = 60637          # random seed
  )
```

Varying slopes model

- Summarize the fit

```
print(model_var_slope, pars=c("mu_beta"))
```

```
## 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% 97.5% n_eff Rhat
## mu_beta[1] 0.04       0 0.01 0.03 0.04 0.04 0.05  0.06  7722    1
## mu_beta[2] 1.06       0 0.02 1.02 1.04 1.06 1.07  1.09  7447    1
##
## Samples were drawn using NUTS(diag_e) at Mon Feb  3 21:03:12 2025.
## 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).
```

Varying slopes model

- Pull the `y_rep` from the model (in the form of a list)

```
y_ppc_varslope <- rstan::extract(model_var_slope)$y_rep
```

- Calculate the upper and lower credible intervals

```
y_ppc_ci_varslope <- t(apply(y_ppc_varslope, 2, function(x) quantile(x, c(.025, .975))))
```

- What's the share that cover the truth?

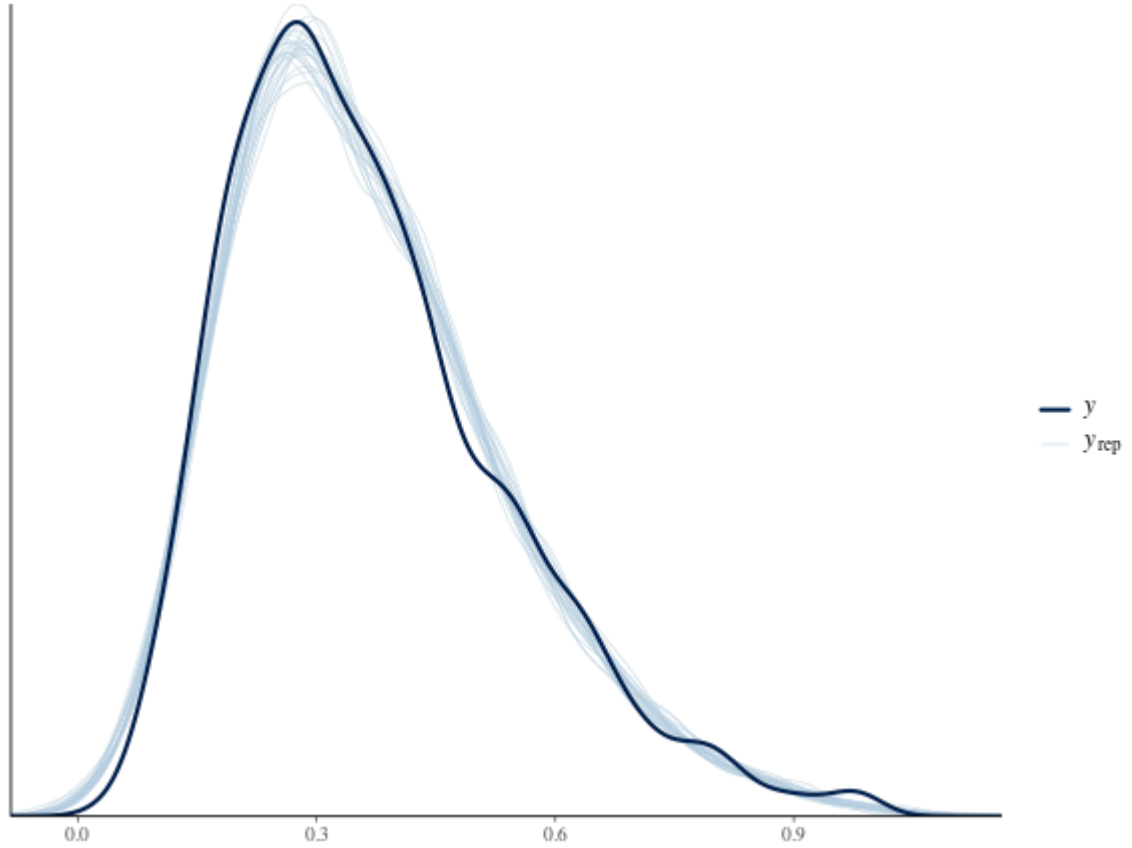
```
cover_95_varint <- y_ppc_ci_varslope[,1]<Y&Y<y_ppc_ci_varslope[,2]
mean(cover_95_varint)
```

```
## [1] 0.967
```

Empirical density

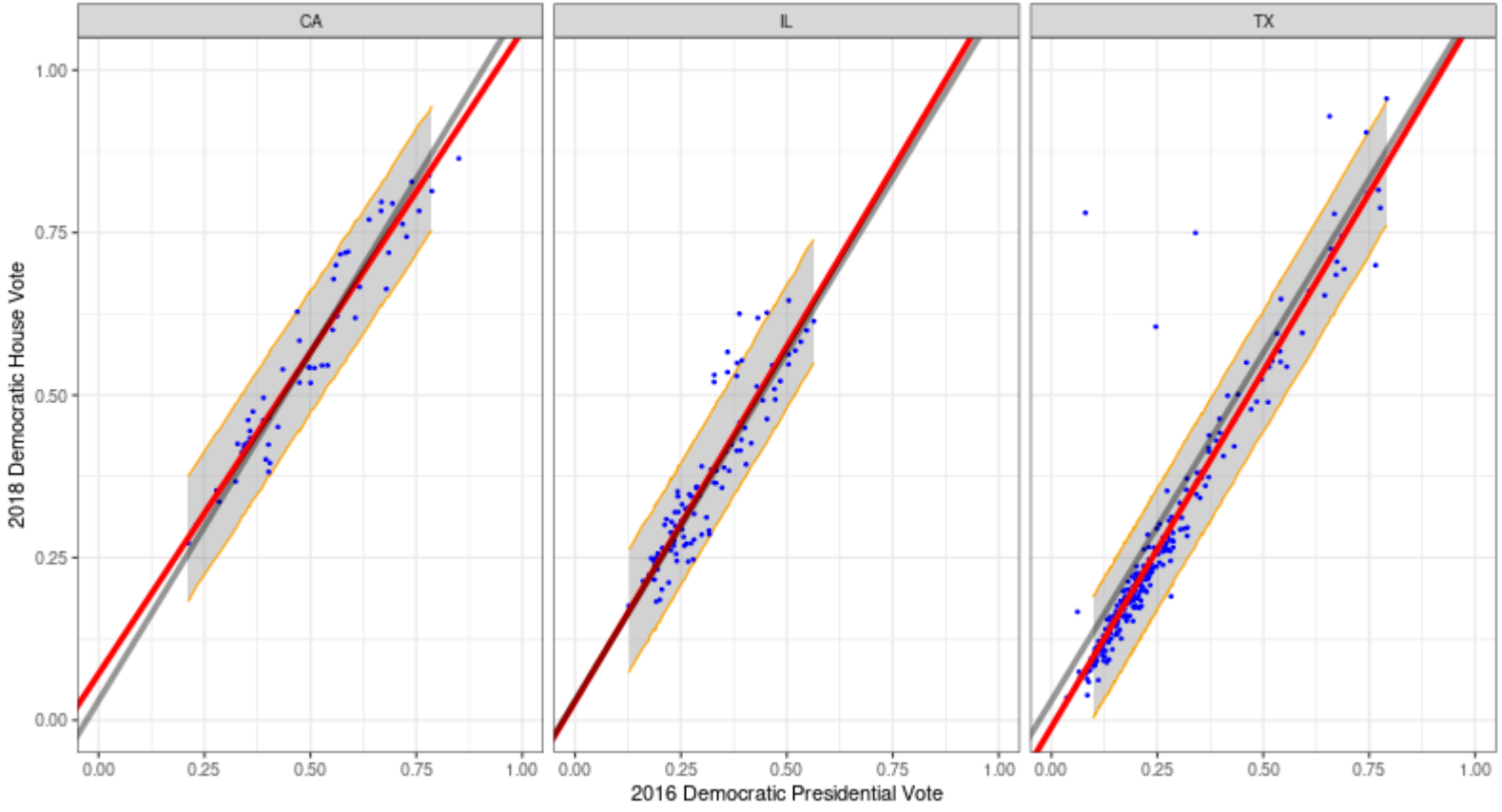
- Posterior predictions vs. empirical density

```
bayesplot::ppc_dens_overlay(y = elections_county$dem2018, yrep = y_ppc_varslope[1:25,])
```



Outcome plots

- Compare the regression lines



Leave-one-out statistic

- Calculate the leave-one-out statistic

```
library(loo)
```

```
loo_varslope <- loo(model_var_slope, pars="log_lik")
loo_varslope$estimates
```

```
##           Estimate    SE
## elpd_loo     4932 169.9
## p_loo         102  15.2
## looic       -9863 339.9
```

```
loo_compare(loo_varslope ,loo_varint, loo_reg)
```

```
##        elpd_diff se_diff
## model1    0.0       0.0
## model2  -77.0      29.0
## model3 -678.1      76.2
```

Post-stratification

Surveys

- Historically, two approaches to survey sampling design
  - **Quota sampling** - Define a set of known demographic targets and recruit respondents to match.
  - **Probability sampling** - Select respondents from a **sampling frame** at random with a known probability.
- Dominance of **probability sampling** in the late 20th century
  - Random Digit Dialing allowed for (near)-simple random samples from the U.S. adult population
  - High response rates

Decline of pure probability samples

- Two big factors have lead to the decline of exclusively probability-based sampling approaches
  - Decline in population coverage - fewer individuals have landlines!
  - Extremely high non-response rates.
- Non-random non-response can bias our estimates
  - Non-responders have different characteristics to the responders that may be correlated with the target quantity of interest.
  - Huge concern in recent efforts to poll elections (e.g. "shy tories")
- Modern polling
  - Combine probability and quota approaches
  - Weighting ex-post to match population targets.

Survey inference

- Our goal is to estimate some parameter θ_y related to a population outcome variable y (e.g. a proportion, mean, median, etc...)
- In addition to the outcome of interest y, we observe **auxiliary** variables $\mathbf{x}$
  - Units are sampled from some unknown target population $f_{\mathcal{U}}(y, \mathbf{x})$
  - The in-sample distribution of the **responders** is denoted $g_{\mathcal{R}}(y, \mathbf{x})$
- Our auxiliary information involves some **known features** of the target distribution: $\breve{I}_{\mathbf{x}}$
  - We'll use this to construct **targets** for the population distribution $\tilde{T}_{\mathbf{x}} = \{\tilde{T}_{\mathbf{x}1}, \ldots, \tilde{T}_{\mathbf{x}M}\}$
  - For example, suppose we know the full population distribution of age, gender, education, income, and party ID
- In some settings, the auxiliary distribution maps easily to the target - but in other settings we have to use the auxiliary information to **estimate** our target
  - (e.g) we might have features of our target distribution but we don't know who will turn out to **vote** before the election
  - "Likely voter models" are a **target estimation** problem

Inference with known sampling weights

- If we know the probability that a unit is selected into the sample from the sample frame π_i, it is straightforward to construct an estimator $\hat{\theta}_y$ of the population mean θ_y.
- The **Horvitz-Thompson** estimator weights each unit by $d_i = \frac{1}{\pi_i}$, the inverse probability of being selected into the sample

$$\hat{\theta_y}^{\text{HT}} = \frac{\sum_{i=1}^{N} d_i Y_i}{\mathbb{E}[\sum_{i=1}^{n} d_i]}$$

- When sampling probabilities are equivalent, this reduces to the sample mean.
- More commonly, rather than using the expectation of the weights in the denominator, we'll use the actual observed sum of the weights, giving the Hajek estimator

$$\hat{\theta_y}^{\text{H}} = \frac{\sum_{i=1}^{N} d_i Y_i}{\sum_{i=1}^{n} d_i}$$

Unknown sampling weights

- When d_i is not known, we will need to estimate **adjustment weights** using a combination of modeling assumptions and auxiliary data
  - Even when d_i is known, if non-response is high, we still don't know the probability of selection into the **observed** data ρ_i
- With adjustment weights $\tilde{w}_i$, our Hajek estimator becomes

$$\hat{\theta_y}^{\text{W}} = \frac{\sum_{i=1}^{N} \tilde{w}_i Y_i}{\sum_{i=1}^{n} \tilde{w}_i}$$

- Now the weights are not necessarily known but must be obtained from our population targets $\tilde{T}_{\mathbf{x}}$

Post-stratification

- The easiest approach to adjusting a non-representative survey is to weight to match the **known joint** distribution of x in the population $f_{\mathcal{U}}(\mathbf{x})$
  - This requires a lot of auxiliary information about the target $f_{\mathcal{U}}(\mathbf{x})$ - typically obtained from high-quality census data
  - (e.g.) U.S. Census Public-Use Microdata: What is the share of Black, college educated, 30-45 year olds in Massachusetts?
- In post-stratification, we divide our sample up into C **cells** c that are mutually exclusive and exhaustive.
  - $\tilde{T}_{\mathbf{x}} = \{\tilde{P}_1, \tilde{P}_2, \ldots, \tilde{P}_C\}$ is our population distribution of these cells
- Our **sampling/response model** assumes that **within** each of these cells we have a simple-random sample from that particular stratum of the population
  - There may be variation in non-response or over/under-sampling, but only **across** cells
  - "post"-stratification because the intuition is akin to a design where we actually *did* stratify ex-ante
- Our **measurement model** assumes we observe the joint distribution of auxiliary variables $\mathbf{x}$ in the target population
  - Difficult in many cases!

Post-stratification

- Two ways to think of post-stratification:
- **First** - Let $\hat{\theta}_y^c$ denote our estimator for the population mean **within** cell c (possibly using design weights d_i)
  - Then, our post-stratification estimator $\hat{\theta}_y^{PS}$ is:

$$\hat{\theta}_y^{PS} = \sum_{c=1}^{C} \tilde{P}_c \hat{\theta}_y^c$$

- Alternatively, we'll sometimes write $\tilde{P}_c = \frac{\tilde{N}_c}{\tilde{N}}$ where $\tilde{N}$ is the size of the population and $\tilde{N}_c$ is the number of units in that cell.

$$\hat{\theta}_y^{PS} = \sum_{c=1}^{C} \frac{\tilde{N}_c}{\tilde{N}} \hat{\theta}_y^c$$

- Fixed "constant" weights on the within-cell estimators.

Post-stratification

- **Second** - We can think of it as weighting individual observations using our adjustment weights $\tilde{w}_i^{PS}$
- Let $c(i)$ denote the class to which unit i belongs. Then the post-stratification weights are

$$\tilde{w}_i^{PS} = (\tilde{P}_{c(i)} / \hat{P}_{c(i)}^{S}) \times d_i$$

where $\hat{P}_{c(i)}^{S}$ is the estimated **in-sample** proportion of observations in class c

- When design weights are constant, $\hat{P}_{c(i)}^{S}$ is just the sample mean of the indicator of class membership $\mathbf{1}_{i \in c}$
  - More generally, you can write it as $\hat{P}_{c(i)}^{S} = (\sum_{i=1}^{N} d_i \mathbf{1}_{i \in c}) / (\sum_{i=1}^{N} d_i)$
- **Intuition**
  - The weights **up-weight** observations that are under-represented in the sample relative to the target population
  - The weights **down-weight** observations that are over-represented in the sample relative to the target population.

Example: CCES 2020

- Let's dive in to the 2020 CCES.

```
library(survey)
cces <- read_csv("data/CCES_subset.csv") %>% filter(!is.na(trumpApprove))
cces$educ_bin[cces$educ_bin == "College degree"] <- "College graduate" # Fix a naming inconsist
```

- We'll be using a subset of the outcome data, looking at the share of respondents who state that they strongly or somewhat approve of then-President Donald Trump.
- Start by making a **`svydesign`** object - we'll pretend that the sampling weights don't exist for now

```
cces_surv_unwt <- svydesign(~1, weights = ~1, data=cces)
```

- In-sample, what's the proportion of respondents who approve of Trump?

```
svymean(~trumpApprove, design=cces_surv_unwt)
```

```
##                mean SE
## trumpApprove 0.383  0
```

Example: CCES 2020

- How does this compare to the properly weighted mean?

```
cces_surv_wt <- svydesign(~1, weights = ~commonweight, data=cces)
svymean(~trumpApprove, design=cces_surv_wt)
```

```
##                mean SE
## trumpApprove 0.444  0
```

- Trump's approval is a few points higher after the weighting adjustment.
- From the CCES Guide:

> the completed cases were weighted to the sampling frame using entropy balancing. The 2019 ACS was used as the frame for weighting the common content and the team samples. The CES sample was weighted to match the distributions of the 2019 ACS on gender, age, race, Hispanic origin, and education level. The moment conditions included age, gender, education, race, plus their interactions. The resultant weights were then post-stratified by age, gender, education, race, “born again" status, voter registration status, 2016 Presidential vote choice, and 2020 Presidential vote choice as needed.

Population targets

- We'll be using the 2020 ACS 5-year Public Use Microdata Sample
  - Obtain the complete **joint** distribution of age, gender and education in the U.S.

```
acs_targets <- read_csv("data/ACS_2020_microdata_3cat.csv")
```

- What's the **marginal** distribution of education in the target population?

```
acs_targets %>% group_by(educ_bin) %>% summarize(n = sum(Count)) %>% ungroup() %>% mutate(prop
```

```
## # A tibble: 4 × 3
##   educ_bin                n  prop
##   <chr>               <dbl> <dbl>
## 1 College graduate 48194135 0.190
## 2 H.S. or less     99042042 0.391
## 3 Postgraduate     28425804 0.112
## 4 Some college     77634550 0.306
```

Population targets

- How does it compare to the **unweighted** marginal distribution in the sample?

```
cces %>% group_by(educ_bin) %>% summarize(n = n()) %>% ungroup() %>% mutate(prop = n/sum(n))
```

```
## # A tibble: 4 × 3
##   educ_bin             n  prop
##   <chr>            <int> <dbl>
## 1 College graduate 14146 0.232
## 2 H.S. or less     18592 0.305
## 3 Postgraduate      8373 0.137
## 4 Some college     19857 0.326
```

Population targets

- What about the **joint** distribution of gender and education?
- In the **population**

```
acs_targets %>% group_by(educ_bin, gender_bin) %>% summarize(n = sum(Count)) %>% ungroup() %>%
```

```
## # A tibble: 8 × 4
##   educ_bin         gender_bin        n   prop
##   <chr>            <chr>         <dbl>  <dbl>
## 1 College graduate Female     25521371 0.101
## 2 College graduate Male       22672764 0.0895
## 3 H.S. or less     Female     48061006 0.190
## 4 H.S. or less     Male       50981036 0.201
## 5 Postgraduate     Female     15043514 0.0594
## 6 Postgraduate     Male       13382290 0.0528
## 7 Some college     Female     41305976 0.163
## 8 Some college     Male       36328574 0.143
```

Population targets

- In the **sample**

```
cces %>% group_by(educ_bin, gender_bin) %>% summarize(n = n()) %>% ungroup() %>% mutate(prop =
```

```
## # A tibble: 8 × 4
##   educ_bin         gender_bin     n   prop
##   <chr>            <chr>      <int>  <dbl>
## 1 College graduate Female      7478 0.123
## 2 College graduate Male        6668 0.109
## 3 H.S. or less     Female     12021 0.197
## 4 H.S. or less     Male        6571 0.108
## 5 Postgraduate     Female      4153 0.0681
## 6 Postgraduate     Male        4220 0.0692
## 7 Some college     Female     11536 0.189
## 8 Some college     Male        8321 0.136
```

Post-stratification

- We can construct the post-stratification weights manually
  - Start by calculating the population proportions in each bin

```
acs_targets <- acs_targets %>% mutate(strata = str_c(gender_bin, educ_bin, age_bin, sep="-"),
                                      pop_proportion = Count/sum(Count))
```

- Do the same for the sample

```
cces <- cces %>% mutate(strata = str_c(gender_bin, educ_bin, age_bin, sep="-"))
cces_strat <- cces %>% group_by(strata) %>% summarize(n=n()) %>% ungroup() %>% mutate(samp_prop
```

- Join the datasets

```
cces <- cces %>% left_join(acs_targets%>% select(strata, pop_proportion), by="strata")
cces <- cces %>% left_join(cces_strat %>% select(strata, samp_proportion), by="strata")
```

Post-stratification

- Construct the post-stratification weights

```
cces <- cces %>% mutate(postStratWt = pop_proportion/samp_proportion)
```

- Take the weighted average to estimate Trump Approval

```
weighted.mean(cces$trumpApprove, cces$postStratWt)
```

```
## [1] 0.397
```

Population targets

- Did the weights equalize the distributions? Let's look again at the joint distribution of gender and education
- In the **population**

```
acs_targets %>% group_by(educ_bin, gender_bin) %>% summarize(n = sum(Count)) %>% ungroup() %>%
```

```
## # A tibble: 8 × 4
##   educ_bin         gender_bin        n   prop
##   <chr>            <chr>         <dbl>  <dbl>
## 1 College graduate Female     25521371 0.101
## 2 College graduate Male       22672764 0.0895
## 3 H.S. or less     Female     48061006 0.190
## 4 H.S. or less     Male       50981036 0.201
## 5 Postgraduate     Female     15043514 0.0594
## 6 Postgraduate     Male       13382290 0.0528
## 7 Some college     Female     41305976 0.163
## 8 Some college     Male       36328574 0.143
```

Population targets

- In the re-weighted **sample**

```
cces %>% group_by(educ_bin, gender_bin) %>% summarize(n = sum(postStratWt)) %>% ungroup() %>% m
```

```
## # A tibble: 8 × 4
##   educ_bin         gender_bin      n   prop
##   <chr>            <chr>       <dbl>  <dbl>
## 1 College graduate Female      6143. 0.101
## 2 College graduate Male        5457. 0.0895
## 3 H.S. or less     Female     11568. 0.190
## 4 H.S. or less     Male       12271. 0.201
## 5 Postgraduate     Female      3621. 0.0594
## 6 Postgraduate     Male        3221. 0.0528
## 7 Some college     Female      9942. 0.163
## 8 Some college     Male        8744. 0.143
```

Post-stratification.

- We can also use the `survey` package - create the post-stratification weights using `postStratify` in `survey`

```
cces_postStrat <- postStratify(cces_surv_unwt, strata=~gender_bin + age_bin + educ_bin,
                               population = acs_targets %>% select(gender_bin, age_bin, educ_bi
```

- Then use `svymean`

```
svymean(~trumpApprove, cces_postStrat)
```

```
##                mean SE
## trumpApprove 0.397  0
```

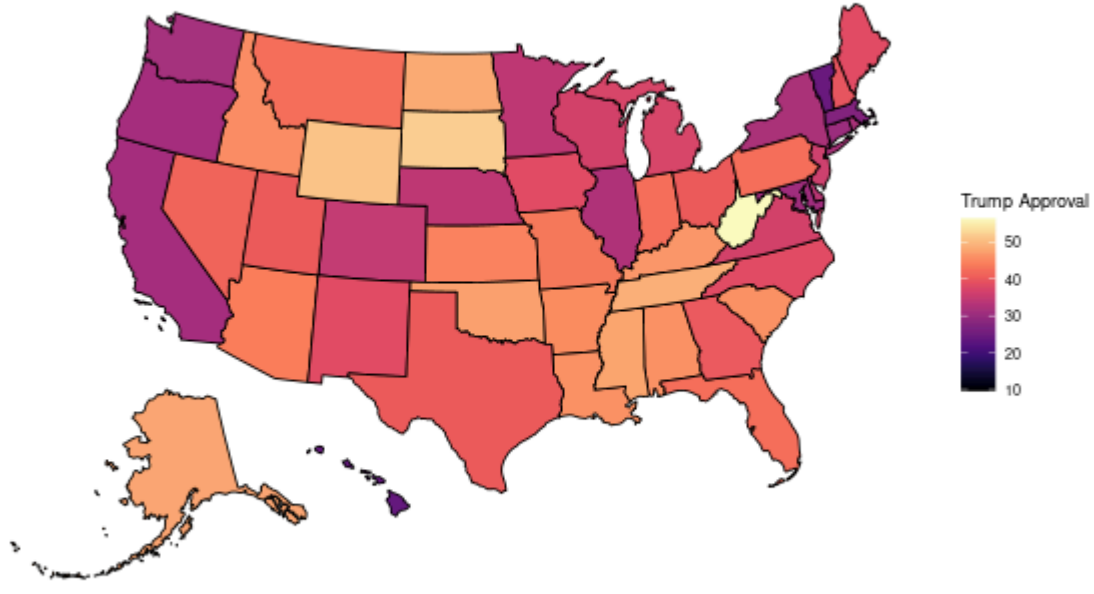
MrP

Multilevel Regression and Post-stratification

- Often we're interested in estimating political attitudes at very small areas
  - Within a nationally-representative sample, we may only have a minor fraction of respondents who are even in a particular region.
- Beyond that, our sample may not be demographically representative for a given region
  - Can we do better than just using the sub-group means for our outcome of interest?

CCES 2020

- Our **unweighted** average Trump support for each state:



MrP

- **Multilevel Regression** and **Post-stratification** combines two well-known tools to try to accomplish this task
  1. **Multilevel regression**: Using the entire dataset, specify a flexible model for the outcome of interest
  2. **Post-stratification**: Using known population characteristics, generate predictions for each covariate "cell" in each region

- The national-level regression allows us to improve **precision** in small-area estimation

  - e.g. if some of the variability between regions is driven by individual or group-level characteristics, we can model the relationship between those characteristics and the outcome using **all** the data.

- The population-level data allows us to address non-response/sampling bias

  - Ideally all of the variables that would go in the survey weighting model also go in the MrP model (so in principle we don't need to include survey weights in the analysis)

MrP Workflow

1. Get survey data that contains individual and group-level predictors (e.g. demographic characteristics + "small area" of residence)
2. Get information on the "small areas" themselves
3. Estimate a multilevel regression model using these two sets of data
4. Construct a post-stratification frame that has the joint distribution of our demographics for each "small area"
5. Predict using the model for each cell in the post-stratification frame
6. Aggregate to get predictions for each small area

Example: CCES 2020

- Let's fit a regression model
  - We'll use the `brms` package which has pre-existing implementations of popular models like the random slopes/random intercepts generalized linear mixed model
- We'll fit a simple logistic regression with coefficients on gender, age and education as well as a random intercept for state

```
library(brms)
```

```
state_mlm <- brm(trumpApprove ~ factor(gender_bin) + factor(age_bin) + factor(educ_bin) +  (1 |
                           family = bernoulli(), data=cces, cores=4, warmup=1000, iter=2000)
```

Model diagnostics

- Summarize the results

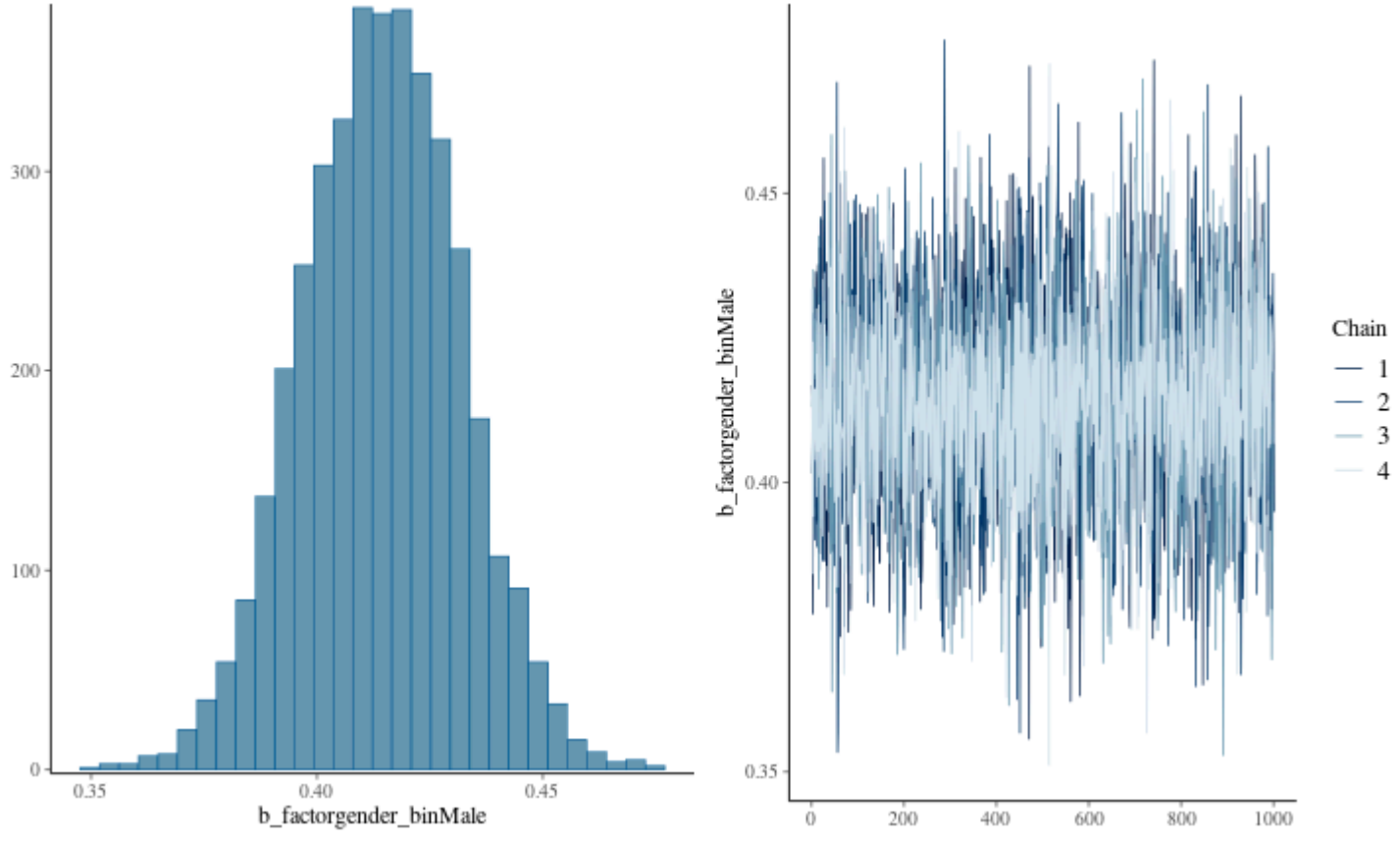
```
summary(state_mlm)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: trumpApprove ~ factor(gender_bin) + factor(age_bin) + factor(educ_bin) + (1 | inputstate)
##    Data: cces (Number of observations: 60968)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Multilevel Hyperparameters:
## ~inputstate (Number of levels: 51)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     0.33      0.04     0.26     0.42 1.00      560      943
##
## Regression Coefficients:
##                            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept                     -1.31      0.06    -1.43    -1.20 1.02      344
## factorgender_binMale           0.41      0.02     0.38     0.45 1.00     3399
## factorage_bin30M44             0.23      0.03     0.17     0.28 1.00     2193
## factorage_bin45M64             0.58      0.03     0.53     0.63 1.00     2144
## factorage_bin65P               0.76      0.03     0.70     0.82 1.00     2134
## factoreduc_binH.S.orless       0.61      0.02     0.56     0.66 1.00     1927
## factoreduc_binPostgraduate    -0.37      0.03    -0.44    -0.31 1.00     2308
## factoreduc_binSomecollege      0.30      0.02     0.25     0.34 1.00     2001
```

Model diagnostics

- Check convergence

```
plot(state_mlm, variable = "b_factorgender_binMale")
```



Generating the post-stratification frame

- We'll use ACS 5-year microdata data at the state level to construct our post-stratification frame for each state

```
acs_state_targets <- read_csv("data/ACS_2020_microdata_by_state.csv") %>% filter(!is.na(Gender)
fips_codes <- read_csv("data/us-state-ansi-fips.csv")
acs_state_targets <- acs_state_targets %>% left_join(fips_codes, by=c(State="stname"))
acs_state_targets$inputstate <- as.numeric(acs_state_targets$st)
acs_state_targets <- acs_state_targets %>% pivot_longer(c(`18-29`, `30-44`, `45-64`, `65+`))
acs_state_targets <- acs_state_targets %>% rename(gender_bin = Gender, educ_bin = Education, ag
acs_state_targets <- acs_state_targets %>% mutate(prop_us = Count/sum(Count))
acs_state_targets <- acs_state_targets %>% group_by(inputstate) %>% mutate(prop_state = Count/s
```

Generating the post-stratification frame

- Each row contains a row for each **state** x **age** x **gender** x **education** combination

```
acs_state_targets
```

```
## # A tibble: 1,632 × 10
##    State   gender_bin educ_bin    st    stusps inputstate age_bin  Count prop_us
##    <chr>   <chr>      <chr>       <chr> <chr>       <dbl> <chr>    <dbl>   <dbl>
##  1 Alabama Male       H.S. or le… 01    AL              1 18-29   195648 7.72e-4
##  2 Alabama Male       H.S. or le… 01    AL              1 30-44   198528 7.84e-4
##  3 Alabama Male       H.S. or le… 01    AL              1 45-64   294365 1.16e-3
##  4 Alabama Male       H.S. or le… 01    AL              1 65+     168153 6.64e-4
##  5 Alabama Male       Some colle… 01    AL              1 18-29   148968 5.88e-4
##  6 Alabama Male       Some colle… 01    AL              1 30-44   129718 5.12e-4
##  7 Alabama Male       Some colle… 01    AL              1 45-64   171277 6.76e-4
##  8 Alabama Male       Some colle… 01    AL              1 65+      91367 3.61e-4
##  9 Alabama Male       College gr… 01    AL              1 18-29    41573 1.64e-4
## 10 Alabama Male       College gr… 01    AL              1 30-44    73950 2.92e-4
## # ℹ 1,622 more rows
## # ℹ 1 more variable: prop_state <dbl>
```

Predicting with the model

- Use the model to make predictions on each row of our post-stratification frame

```
acs_state_targets$predict_mlm <- predict(state_mlm, newdata=acs_state_targets)[,1]
```

- Aggregate the predictions by the stratum shares to get state-level estimates!

```
mrp_states <- acs_state_targets %>% group_by(st) %>% summarize(trumpApproveMLM = sum(predict_ml
```

Compare

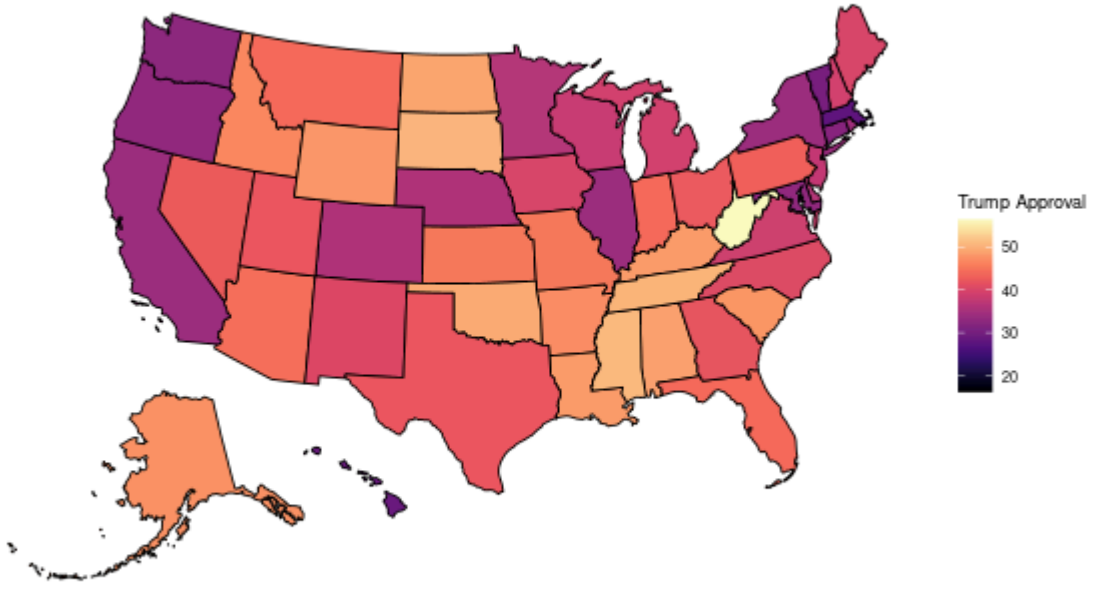
- Use the model to make predictions on each row of our post-stratification frame

```
state_summaries %>% left_join(mrp_states, by="fips")
```

```
## # A tibble: 51 × 3
##    fips  values trumpApproveMLM
##    <chr>  <dbl>           <dbl>
##  1 01     45.6             48.7
##  2 02     47.8             47.4
##  3 04     43.7             44.5
##  4 05     45.1             47.0
##  5 06     31.2             33.7
##  6 08     34.5             35.9
##  7 09     30.4             31.7
##  8 10     33.8             36.5
##  9 11      9.64            16.2
## 10 12     42.2             43.9
## # ℹ 41 more rows
```

Visualize

- Our **multilevel regression + post-stratification** estimates



How to improve this model

- We've provided a very simple example of an MrP model but there are a lot of ways it can be improved
  1. More group-level predictors (e.g. Trump state-level vote share)
  2. Individual-level interactions in the model + partial-pooling on the coefficients
  3. Actually knowing the joint distribution of the demographics in each state!
  4. Pool state-level means to regional rather than a common "grand" mean.
  5. Many more possibilities...
- More generally, we have a dataset where even within each state, we have many observations to work with, so the model isn't that necessary
  - MrP is most powerful when there are **very few** respondents from a particular small area in our data