

Multilevel Regression and Post-stratification

Douglas Rivers

5/1/2018

Course Schedule

| Time | Topics |
|-------------|--------------------------------|
| 2:30 - 3:15 | Overview and Examples |
| 3:15 - 4:00 | Post-stratification |
| 4:00 - 4:45 | Multilevel Regression |
| 4:45 - 5:30 | Bayesian inference |
| 5:30 - 6:00 | Advanced topics and discussion |

Course materials and basic R setup

Slides, code, and datasets can be downloaded from Github:

<https://github.com/rdrivers/mrp-aapor>

If you wish to run the examples shown in class, you will need to

1. Download R from one of the sites listed at <https://cran.r-project.org/mirrors.html> and install. R is an open source programming environment for statistical computing.
2. (Optional) Download RStudio Desktop from [LINK]. RStudio is an integrated development environment for R. The free open source version is recommended.
3. Start R or RStudio and enter the following at the R command prompt:

```
install.packages(c("tidyverse", "lme4", "survey", "arm", "maps", "mapproj",  
  "gridExtra"))
```

1. Overview and Examples

What is MRP?



Kristen Soltis Anderson ✓

@KSoltisAnderson

Following



Most popular at [#AAPOR](#): some guy named Mr. P and some other guy named Stan

2:59 PM - 13 May 2016

Formally, **M**ultilevel **R**egression and **P**ost-stratification

Informally, **Mr. P**

Who is the ubiquitous Mr. P?



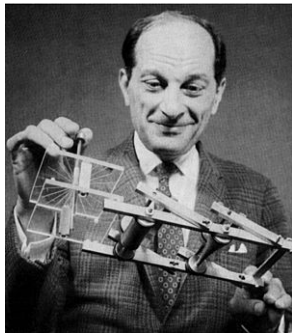
Andrew Gelman

Gelman is a prolific statistician, political scientist and blogger at Columbia University.

Developed MRP in a series of papers starting in 1997. Adapts and extends some techniques for *small area estimation* from survey statistics.

Ideas (post-stratification, multilevel models, hierarchical Bayesian estimation) were well known, but the combination has been very effective in polling applications.

. . . and his mysterious friend Stan?



Stanislaw Ulam
(1909-1984)

Ulam was a physicist who invented the *Monte Carlo method* while working on the Manhattan Project.

Stan is an open source software project developing *Markov chain Monte Carlo* (MCMC) software for statistical inference and other applications.

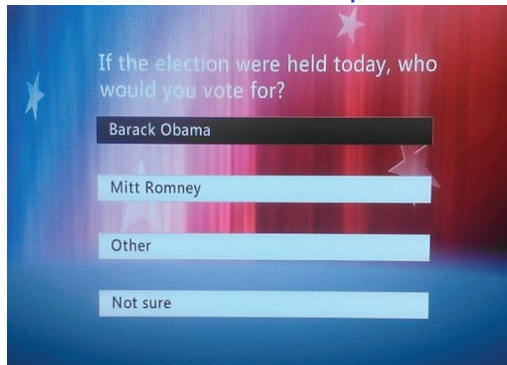


<https://mc-stan.org>

What problems does MRP address?

- ▶ **Selection bias.** Flexible and robust method for correcting for imbalances in sample composition, even when these are severe and can involve a large number of variables.
- ▶ **Small area estimation.** Can provide good estimates for sub-national units (such as states, Congressional Districts, counties, *etc.*)
- ▶ **Trend analysis.** Estimate means of survey variables over time with a set of rolling cross-sections

Selection bias: the Xbox panel



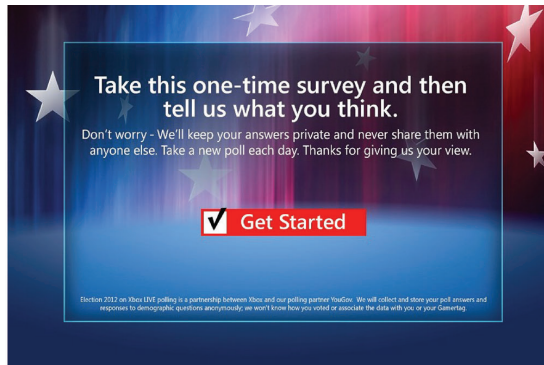
If the election were held today, who would you vote for?

Barack Obama

Mitt Romney

Other

Not sure



Take this one-time survey and then tell us what you think.

Don't worry - We'll keep your answers private and never share them with anyone else. Take a new poll each day. Thanks for giving us your view.

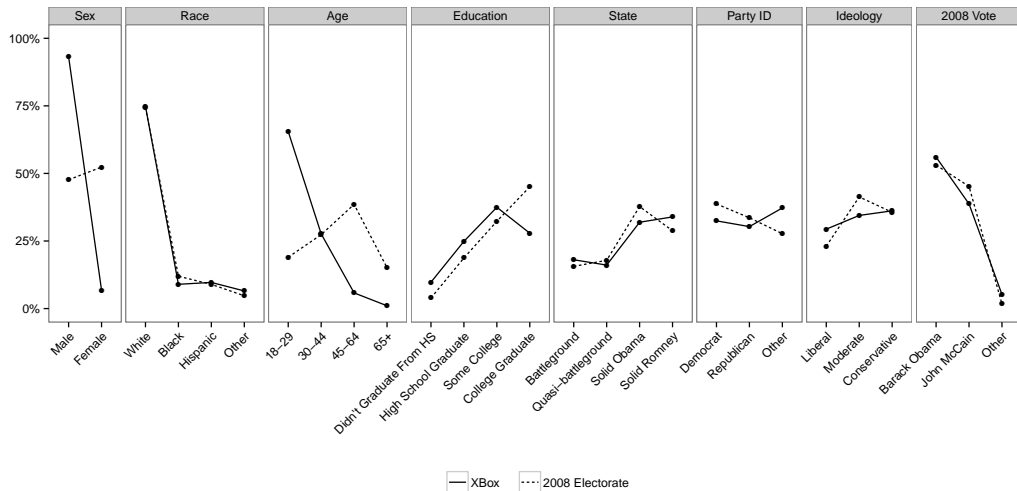
☒ Get Started

Election 2012 on Xbox LIVE polling is a partnership between Xbox and our polling partner YouGov. We will collect and store your poll answers and responses to demographic questions anonymously; we won't know how you voted or associate the data with you or your Gamertag.

- ▶ 750,148 interviews with 345,858 Xbox users during the 2012 election campaign
- ▶ However, the sample had some problems. . . .

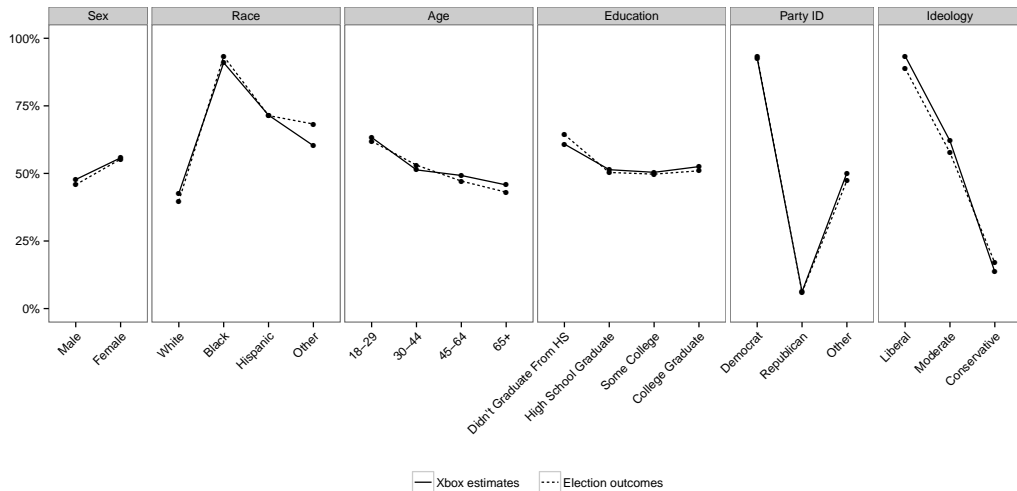
(Gelman, Goel, Rivers, and Rothschild, "The Mythical Swing Voter", *Quarterly Journal of Political Science*, 2016)

Xbox panel demographics



93% were male, but there were still over 5,000 women in the sample.

MRP estimates of 2012 voting from Xbox panel

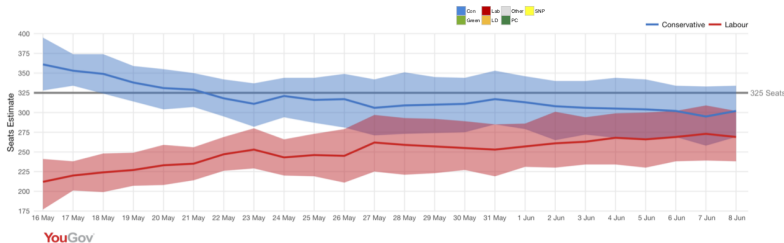
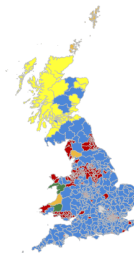


75,096 cells: 2 gender x 4 race x 4 age x 4 education x 4 party x 3 ideology x 50 states

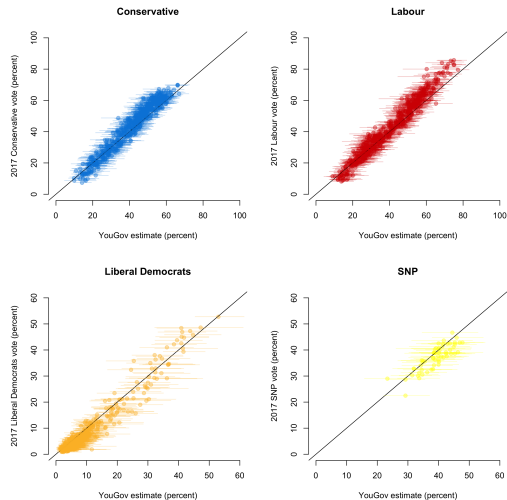
Lesson 1

Under some circumstances, **big data** (large sample sizes) and **modelling** can work with severe selection bias and empty cells.

Small area estimation: the 2017 UK general election



Comparison of MRP forecasts and outcomes

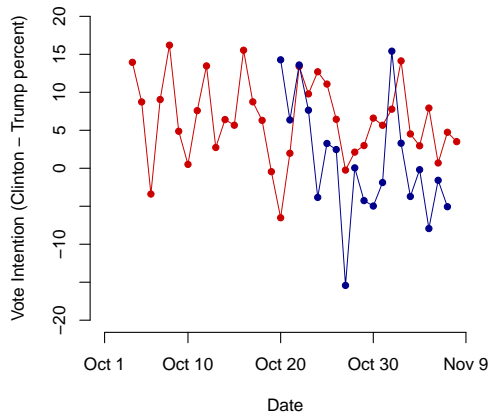


Lesson 2

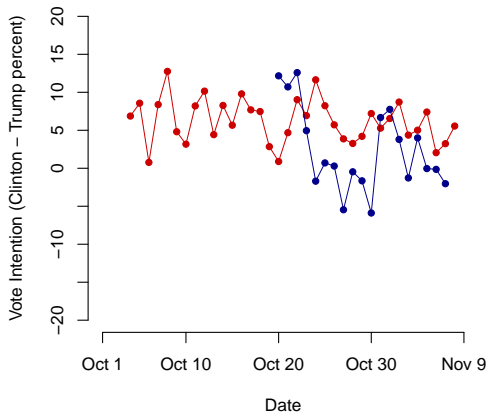
It's possible to get pretty good estimates of **small areas** (parliamentary constituencies, congressional districts, even precincts) with effective predictors and multilevel models.

Why do polls bounce around so much?

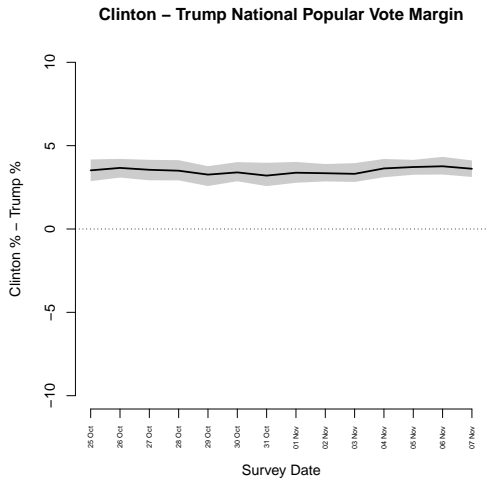
Using survey weights



Reweighting by Party ID



Trend analysis: the 2016 US presidential election



Large swings in the 2012 and 2016 pre-election polls were methodological artifacts due to inadequate **post-stratification** and **modeling**.

Ingredients for MRP and running example

Survey Pew Research Organization's *October 2016 Political Survey* (2,583 interviews, conducted October 20-25, 2016.)

Survey variable 2016 Presidential voting intention

Covariates Individual characteristics (from the survey) and group level predictors (2012 state vote)

Post-strata Age \times Gender \times Race \times Education \times State

Stratum counts from the November 2016 Voting and Registration Supplement to the *Current Population Survey*

Data sources

The file `cleaned.RData` contains four R dataframes:

- ▶ `pew` - Pew Research Organization's **October 2016 Political Survey**. The original data can be found at <http://www.people-press.org/dataset/october-2016-political-survey/>.
- ▶ `cps` - the November 2016 Voting and Registration Supplement to the **Current Population Survey**. The full dataset can be downloaded from <http://www.nber.org/cps/>.
- ▶ `votes12` and `votes16` - votes cast for major presidential candidates, turnout, and voting age population by state. **Vote counts** are from <https://uselectionatlas.org/> and **population counts** are from <https://www2.census.gov/programs-surveys/cps/>.

Code `clean-data.R` for creating `cleaned.RData` is on the course Github site.

How MRP works

Step 1: multilevel regression Fit a model relating the survey variable (vote) to individual and group level covariates (age, gender, race, education, state, 2012 vote).

Step 2: imputation Impute the survey variable (proportion voting Democratic) for all combinations of age \times gender \times race \times education \times state. each cell of post-stratification (2 age \times 2 gender \times 3 race \times 4 education \times 51 state).

Step 3: post-stratification Weight the imputed values by the population count in each of the $2 \times 2 \times 3 \times 4 \times 51 = 2248$ cells.

Some important concepts we will cover

- ▶ Fit **multilevel** regression models for survey variables which are “richly parameterized” and can flexibly adjust for high dimensional selection bias.
- ▶ Estimate models using **hierarchical Bayesian priors** to avoid over-fitting.
- ▶ Impute using the **posterior predictive distribution** (not the sample average).
- ▶ Use **multiple imputations** to accurately reflect the sources of uncertainty in the imputation process.

Don't worry if you don't understand the terminology – details will follow – but it's just **regression** and **imputation**.

2. Post-stratification

Review of stratified sampling

In *stratified sampling*, we divide the population into **strata** (e.g., regions) and draw samples within each stratum.

The **stratified estimator** is obtained by weighting the stratum estimates by the population size in each.

A stratified design can be more **efficient** than a simple random sample of the same size because

- ▶ Eliminates variability in the proportion of sample coming from each stratum
- ▶ Larger sample sizes allocated to more heterogeneous strata, less to homogeneous strata

Notation for stratified sampling

Population $U = U_1 \cup \dots \cup U_H$ partitioned into H strata

Sample $S = S_1 \cup \dots \cup S_H$ where $S_h \subset U_h$.

Population sizes $N = N_1 + \dots + N_H$

Sample sizes $n = n_1 + \dots + n_H$

Sample means $\bar{y}_1, \dots, \bar{y}_H$

Stratified estimator $\bar{y}_{ST} = \frac{N_1}{N} \bar{y}_1 + \dots + \frac{N_H}{N} \bar{y}_H$

The stratified estimator is a **weighted** combination of the stratum means. The weights are **known** – sampling variation comes from errors in estimating the stratum means.

Stratification weights

An equivalent way of computing the stratified estimator is to define the **stratum weights**

$$w_h = \frac{\text{population proportion in stratum}}{\text{sample proportion in stratum}} = \frac{N_h/N}{n_h/n}$$

Each respondent gets the weight associated with their stratum:

$$\bar{y}_{ST} = \text{weighted average of sample values} = \frac{w_{h_1}y_1 + \cdots + w_{h_n}y_n}{w_{h_1} + \cdots + w_{h_n}}$$

where h_i and w_{h_i} are the **stratum and weight****, respectively, associated with respondent i .

Variance of the stratified estimator

$$V(\bar{y}_{ST}) = \sum_{h=1}^H \left(\frac{N_h}{N} \right)^2 \left(1 - \frac{n_h}{N_h} \right) \frac{S_h^2}{n_h} = \frac{1}{n} \sum_{i=1}^n w_{hi}^2 (1 - f_h) \frac{S_{hi}^2}{n}$$

There are three factors that determine the variance of the stratified estimator:

- ▶ Size of the **sampling fractions** $f_h = n_h/N_h$ in each stratum. Usually these are close to zero and can be ignored.
- ▶ The **within stratum variances** $\sum_{i \in U_h} (y_i - \bar{y}_h)^2 / (N_h - 1)$.
- ▶ Variability of the **stratum weights** w_h .

S_h^2 can be substantially less than the total variance of y , so there are potential large gains in efficiency from stratifying, compared to SRSWOR.

Variation in the weights tends to *increase* the standard error of the estimator, unless it's possible to allocate sample size n_h proportional to S_h^2 .

Post-stratification

When the sample design is **not** stratified, it may still be feasible to use the stratified estimator if the sample can be divided into groups (called **post-strata**) whose population sizes are known.

The sample sizes in the post-strata are not fixed by design, so the variance formula only holds conditional upon the realized sample sizes n_1, \dots, n_H . (See Holt and Smith, “Post-stratification”, *Journal of the Royal Statistical Society, Series A*, 1979.)

Even though the sample may have started with equal selection probabilities, often the proportions in the post-strata differ substantially from the population proportions. Failure to post-stratify results in large bias, but large weights may result in unstable estimates.

Usual advice is to collapse post-strata to have at least 20 respondents per cell.

Post-stratification in R

Step 1 Recode survey and population data (usually from a *public use microdata* file) so that the post-stratifying variables are consistent.

Step 2 Choose the post-strata

Step 3 Collapse categories to eliminate empty cells

Step 4 Compute the post-stratification weights

Step 5 Estimate means and proportions (with standard errors)

Step 1: recode Pew data...

Variables should be factors (R's version of categorical variables) with the same levels (categories) *in the same order*.

```
suppressMessages(library("tidyverse"))
load("data/cleaned.RData")
pew <- pew %>%
  filter(
    complete.cases(age, raceeth, gender, educ, vote16),
    vote16 != "nonvoter") %>%
  mutate(
    demvote = ifelse(vote16 == "clinton", 1, 0),
    age4 = factor(case_when(age < 30 ~ "18-29",
      age < 45 ~ "30-44", age < 65 ~ "45-64",
      TRUE ~ "65+")),
    race3 = fct_collapse(raceeth,
      white = c("white", "other")),
    educ4 = fct_collapse(educ,
      "hs" = c("grades 1-8", "hs dropout", "hs grad"),
      "some col" = c("some col", "assoc")))
```

...then do the same for CPS

```
cps <- cps %>%  
  filter(  
    complete.cases(age_top_codes,  
      raceeth, gender, educ, turnout),  
    turnout == "yes") %>%  
  mutate(  
    age4 = factor(case_when(  
      age_top_codes == "<80" & age < 30 ~ "18-29",  
      age_top_codes == "<80" & age < 45 ~ "30-44",  
      age_top_codes == "<80" & age < 65 ~ "45-64",  
      TRUE ~ "65+")),  
    race3 = fct_collapse(raceeth,  
      white = c("white", "other")),  
    educ4 = fct_collapse(educ,  
      "hs" = c("grades 1-8", "hs dropout", "hs grad"),  
      "some col" = c("some col", "assoc")))
```

Check that the datasets are consistent – mistakes will be made!

Time spent cleaning the data at this stage is well spent.

```
compare_distributions <- function(var, data1, data2, wgt1, wgt2, digits = 1) {  
  stopifnot(all(levels(data1[[var]]) == levels(data2[[var]])))  
  formula1 <- as.formula(paste(wgt1, "~", var))  
  formula2 <- as.formula(paste(wgt2, "~", var))  
  tbl <- rbind(round(100 * prop.table(xtabs(formula1, data1)), digits),  
              round(100 * prop.table(xtabs(formula2, data2)), digits))  
  row.names(tbl) <- c(substitute(data1), substitute(data2))  
  tbl  
}  
compare_distributions("race3", pew, cps, "", "weight")
```

```
##      white black hispanic  
## pew  83.3   8.9      7.8  
## cps  78.9  11.9      9.2
```


Compare variables in pew and cps

```
compare_distributions("educ4", pew, cps, "", "weight")
```

```
##      hs some col col grad postgrad
## pew 22.0    26.9    29.7    21.3
## cps 29.6    30.8    25.0    14.6
```

```
compare_distributions("age4", pew, cps, "", "weight")
```

```
##      18-29 30-44 45-64 65+
## pew  12.8  19.3  40.6 27.3
## cps  15.7  22.5  37.6 24.2
```

```
compare_distributions("gender", pew, cps, "", "weight")
```

```
##      male female
## pew 53.5    46.5
## cps 46.4    53.6
```

Step 2: create post-strata

The survey package contains useful functions for post-stratification, including postStratify, rake, and nonresponse.

```
suppressMessages(library(survey))
pop.counts <- xtabs(weight ~ age4 + gender + race3 + educ4, data = cps)
sample.counts <- xtabs(~ age4 + gender + race3 + educ4, data = pew)
pew <- mutate(pew,
  weight0 = sum(pop.counts) / sum(sample.counts))
sample.weights <- xtabs(weight0 ~ age4 + gender + race3 +
  educ4, data = pew)
nr <- nonresponse(sample.weights, sample.counts, pop.counts)
```

nr is an object which keeps track of the cells (post-strata).

Check for empty cells and/or large weights

```
sparseCells(nr, nrweight = 4)
```

```
## sparseCells(nr, nrweight = 4)
## Cells: 14 22 44 72 81 85 92 96
## Indices:
##   age4   gender  race3   educ4
## 14 "30-44" "female" "black"  "hs"
## 22 "30-44" "female" "hispanic" "hs"
## 44 "65+"   "male"   "hispanic" "some col"
## 72 "65+"   "female" "hispanic" "col grad"
## 81 "18-29" "male"   "black"   "postgrad"
## 85 "18-29" "female" "black"   "postgrad"
## 92 "65+"   "male"   "hispanic" "postgrad"
## 96 "65+"   "female" "hispanic" "postgrad"
## Summary:
##   NRwt    wt  n
## 14 4.61 374000 2
## 22  Inf    Inf  0
## 44  Inf    Inf  0
## 72  Inf    Inf  0
## 81  Inf    Inf  0
## 85  Inf    Inf  0
## 92  Inf    Inf  0
## 96  Inf    Inf  0
```

Look for categories adjacent to empty cells

```
neighbours(14, nr) # use nr$index to get cell index
```

```
## `[.nonresponse`(object, nbour.index)
## Cells: 15 22 38 13 10 6
## Indices:
##   age4   gender  race3    educ4
## 15 "45-64" "female" "black"   "hs"
## 22 "30-44" "female" "hispanic" "hs"
## 38 "30-44" "female" "black"   "some col"
## 13 "18-29" "female" "black"   "hs"
## 10 "30-44" "male"   "black"   "hs"
## 6  "30-44" "female" "white"   "hs"
## Summary:
##   NRwt    wt  n
## 15 2.07 168000 8
## 22  Inf   Inf  0
## 38 1.22  98400 8
## 13 2.71 220000 3
## 10 3.18 258000 2
## 6  2.09 169000 11
```

Step 3: collapse cells

Cells 10, 11, 14 and 15 contain blacks aged 30-64 with a high school education.

```
nr$index[,,"black","hs"]
```

```
##           gender
## age4      male female
## 18-29      9      13
## 30-44     10      14
## 45-64     11      15
## 65+       12      16
```

```
nr <- joinCells(nr, 10, 11, 14, 15) # update the nr object
nr$index[,,"black","hs"]
```

```
##           gender
## age4      male female
## 18-29      9      13
## 30-44     10      10
## 45-64     10      10
## 65+       12      16
```

Eliminate remaining empty cells

Combine males and females and collapse age and education categories for minorities. Each call to `joinCells` collapses some cells and stores the result in `nr`.

```
nr <- joinCells(nr, 18, 19, 21, 22) # hisp 30-64 hs
nr <- joinCells(nr, 44, 48, 68, 72, 92, 96) # hisp 65+ >hs
nr <- joinCells(nr, 57, 61, 81, 85) # black 18-29 col+
sparseCells(nr, nrweight = 4) # no more bad cells
```

```
## NULL
```

NULL means that, after collapsing, there are no empty cells or weights > 4 .

Step 4: compute weight and add to dataframe

`weights(nr)` is a four dimensional (age x gender x race x education) array of **stratum weights**. I created `get_weights` to convert the stratum weights into a vector of **individual weights**.

```
get_weights <- function(data, nr) {  
  wgt_arr <- weights(nr)  
  var.names <- names(dimnames(wgt_arr))  
  indexes <- data %>%  
    select(var.names) %>%  
    mutate_all(as.integer) %>%  
    as.matrix()  
  wgt_arr[indexes]  
}  
pew$ps.weight <- get_weights(pew, nr)
```

Check that the post-stratification worked

```
compare_distributions("race3", pew, cps, "ps.weight", "weight")
```

```
##      white black hispanic
## pew  78.9  11.9      9.2
## cps  78.9  11.9      9.2
```

```
compare_distributions("educ4", pew, cps, "ps.weight", "weight")
```

```
##      hs some col col grad postgrad
## pew  29.6    30.8    25.2    14.4
## cps  29.6    30.8    25.0    14.6
```

```
compare_distributions("age4", pew, cps, "ps.weight", "weight")
```

```
##      18-29 30-44 45-64 65+
## pew  15.7  22.0  38.1 24.2
## cps  15.7  22.5  37.6 24.2
```


Step 5: compute estimates using the new weight

Use either xtabs (in base R) or the survey package

```
round(100 * prop.table(xtabs(ps.weight ~ vote16, pew)), 1)
```

```
## vote16
##  clinton    trump    other nonvoter
##    49.1     41.2     9.7      0.0
```

```
design <- svydesign(ids = ~ 1, weights = ~ ps.weight, data = pew)
round(100 * prop.table(svytable(~ vote16, design)), 1)
```

```
## vote16
##  clinton    trump    other nonvoter
##    49.1     41.2     9.7      0.0
```

```
cv <- function(x) sd(x) / mean(x) # coefficient of variation
cv(pew$ps.weight)^2 # weighting loss
```

```
## [1] 0.2154227
```

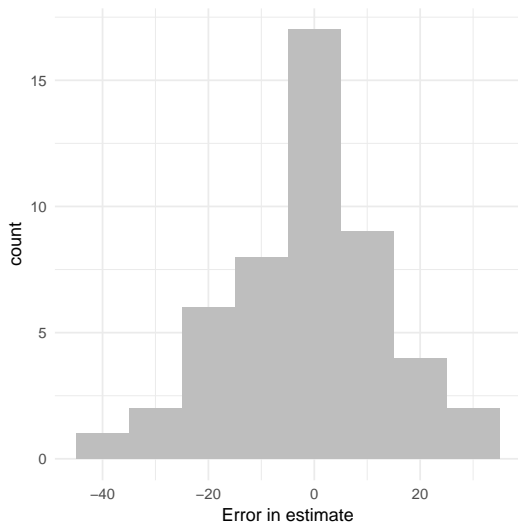
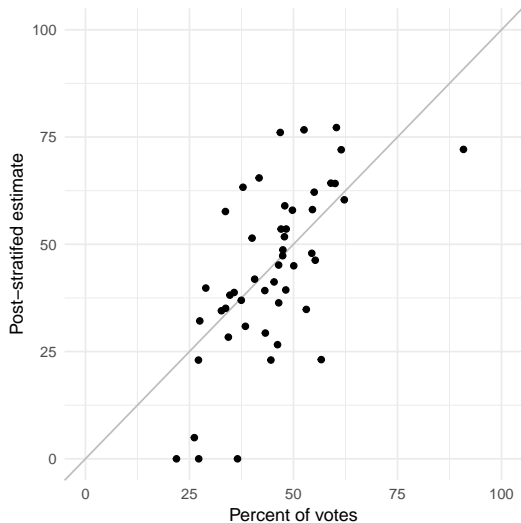
State estimates

```
tbl <- 100 * prop.table(xtabs(ps.weight ~ state + vote16, data = pew), 1)
round(tbl, 1)[1:10,]
```

| ## | vote16 | | | | |
|----|--------|---------|-------|-------|----------|
| ## | state | clinton | trump | other | nonvoter |
| ## | AK | 0.0 | 86.9 | 13.1 | 0.0 |
| ## | AL | 28.4 | 60.3 | 11.3 | 0.0 |
| ## | AR | 57.6 | 42.4 | 0.0 | 0.0 |
| ## | AZ | 23.0 | 62.3 | 14.7 | 0.0 |
| ## | CA | 72.0 | 19.9 | 8.0 | 0.0 |
| ## | CO | 39.4 | 44.1 | 16.5 | 0.0 |
| ## | CT | 58.1 | 33.0 | 8.9 | 0.0 |
| ## | DC | 72.1 | 6.3 | 21.6 | 0.0 |
| ## | DE | 34.8 | 65.2 | 0.0 | 0.0 |
| ## | FL | 47.3 | 45.3 | 7.4 | 0.0 |

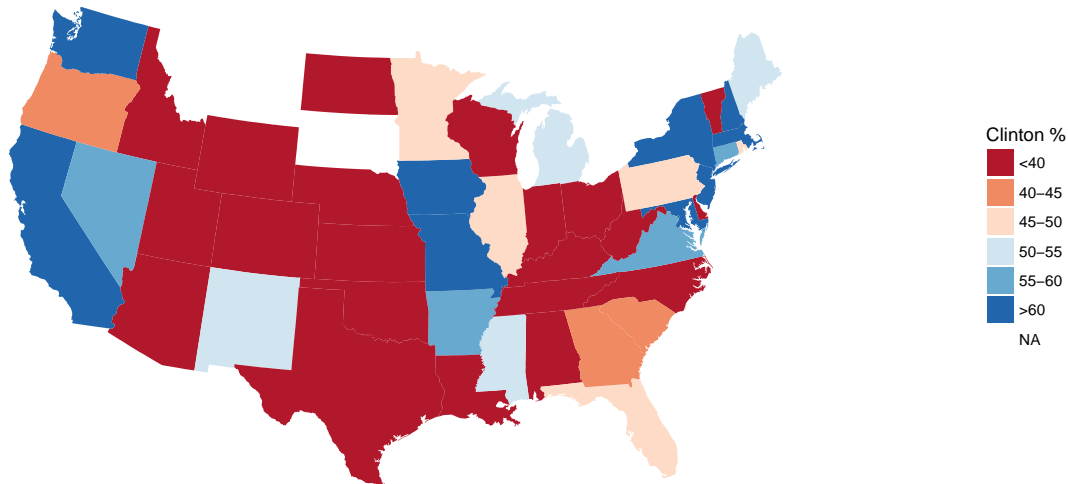
Plotting the estimates

State estimates are nearly useless, even if approximately unbiased.



Mapping the estimates

Probably not something you'd want to share with the world.



Problems with post-stratification

Empty cells

- ▶ Collapsing cells increases bias
- ▶ Alternative is *raking* (works if no empty cells in marginals)

Small cells

- ▶ Even with non-empty cells, mean estimates will be noisy if the cell size is small.
- ▶ Recommended cell size is at least 20 – suggest collapsing cells

Large weights

- ▶ Common practice is to *trim* large weights.
- ▶ Solves the wrong problem: population cell count is ok, sample cell estimate is noisy.

3. Multilevel regression

Key ideas

- ▶ Use a regression model to **predict** values of the survey variable for each combination of covariates.
- ▶ Use a **richly parameterized model** for the data (the first level)
 - ▶ Allow intercepts and possibly slopes to vary across groups in the data (“deep interactions”).
 - ▶ Use variables with many categories (such as counties or states), some of which may not be present in the data.
- ▶ Introduce a second level model for the **parameters** in the first level.
 - ▶ The parameters of the second level model are sometimes called *hyperparameters*.
 - ▶ The second level model is used to avoid overfitting. Results in *shrinkage* or *regularization*.
- ▶ Also known as hierarchical models, mixed models, random effects, variance components, repeated measures, longitudinal analysis, *etc.*

Stein's paradox



Charles Stein (1920-2016)

The **mean square error** (or **risk**) of an estimator $\hat{\theta}$ of θ is

$$\text{MSE}(\hat{\theta}, \theta) = E_{\theta}(\hat{\theta} - \theta)^2$$

An estimator $\hat{\theta}$ is **inadmissible** if there exists another estimator $\tilde{\theta}$ with smaller risk for *all* θ . The estimator $\tilde{\theta}$ is said to **dominate** $\hat{\theta}$.

When estimating three or more means of independent normal distributions, Stein showed that the sample mean is **inadmissible**.

A tale of three estimators

To keep things simple, we start with models for vote by state without any covariates.

No pooling estimate the model separately for each state (by a regression on 51 state dummies)

Complete pooling estimate a single **grand mean** (equivalent to a regression with just an intercept)

Partial pooling estimate a *random effects* model (to be explained shortly – think of it as a compromise between no pooling and complete pooling)

(For pedagogical purposes, we will use *linear* instead of *logistic regression* in this section. The same concepts apply to nonlinear model.)

Fixed effects (no pooling)

We use `lm` to estimate a model with state dummies and no intercept (so the estimates are the predicted Clinton vote in each state).

```
no.pooling <- lm(demvote ~ state - 1, data = pew)
no.pooling <- data_frame(
  state = gsub("state", "", names(coef(no.pooling))),
  no.pooling = 100 * coef(no.pooling)) %>%
  mutate(state = factor(state, levels = levels(pew$state)))
head(no.pooling)
```

```
## # A tibble: 6 x 2
##   state no.pooling
##   <fct>     <dbl>
## 1 AK         0
## 2 AL        25.
## 3 AR       50.0
## 4 AZ       23.1
## 5 CA       69.4
## 6 CO       42.9
```

Grand mean (complete pooling)

The grand mean can be computed in **three equivalent ways**:

```
round(100 * mean(pew$demvote), 1) # 1. mean of dichotomous indicator
```

```
## [1] 48.4
```

```
round(100 * prop.table(xtabs(~ demvote, data = pew)), 1) # 2. cross-tabulation
```

```
## demvote  
##      0      1  
## 51.6 48.4
```

```
complete.pooling <- lm(demvote ~ 1, data = pew) # 3. intercept in regression  
complete.pooling <- 100 * coef(complete.pooling)  
round(complete.pooling, 1)
```

```
## (Intercept)  
##          48.4
```

Random effects (partial pooling)

Estimate a model with one **fixed effect** (the grand mean) plus **random effects** for each state,

$$\text{vote} = \text{grand mean} + \text{state effect} + \text{individual error}$$

or, in symbols,

$$y_i = \mu + \alpha_{j_i} + \epsilon_i \quad (j_i = \text{state of respondent})$$

where

$$\alpha_1, \dots, \alpha_J \stackrel{\text{iid}}{\sim} \text{Normal}(0, \sigma_\alpha) \quad \epsilon_1, \dots, \epsilon_n \stackrel{\text{iid}}{\sim} \text{Normal}(0, \sigma_\epsilon)$$

The grand mean μ is a **fixed effect** while the state effects α_j are **random effects** (assumed to be $\text{Normal}(0, \sigma_\alpha)$).

In classical statistics, parameters (like μ , σ_ϵ and σ_α) are **estimated** while random effects are **predicted**.

What's the difference between fixed and random effects models?

In both models,

$$y_i = \mu + \alpha_{j_i} + \epsilon_i$$

with the only difference being

- ▶ **Fixed effects** $\alpha_1, \dots, \alpha_J$ are *unknown* parameters to be estimated.
 - ▶ $E(y_i) = \mu + \alpha_{j_i}$ and $V(y_i) = \sigma_\epsilon^2$
- ▶ **Random effects** $\alpha_1, \dots, \alpha_J$ are *unknown* random variables to be predicted.
 - ▶ $E(y_i) = \mu$ and $V(y_i) = \sigma_\alpha^2 + \sigma_\epsilon^2$
 - ▶ But $E(y_i|\alpha_{j_i}) = \mu + \alpha_{j_i}$ and $V(y_i|\alpha_{j_i}) = \sigma_\epsilon^2$

If you find this confusing, you are not alone. Gelman and Hill argue that the only meaningful difference is that the α_j are modelled in one and not in the other.

Predicting random effects

The default method for R's `lmer` function (in the `lme4` package) is *REML* (residualized or restricted maximum likelihood), which is unbiased for linear models.

```
suppressMessages(library(lme4))
partial.pooling <- lmer(demvote ~ 1 + (1 | state), data = pew)
partial.pooling
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: demvote ~ 1 + (1 | state)
##   Data: pew
## REML criterion at convergence: 2415.637
## Random effects:
##   Groups   Name      Std.Dev.
##   state    (Intercept) 0.1183
##   Residual                0.4854
## Number of obs: 1698, groups:  state, 49
## Fixed Effects:
## (Intercept)
##      0.4555
```

Extracting predictions

```
fixef(partial.pooling) # grand mean
```

```
## (Intercept)  
## 0.4555195
```

```
ranef(partial.pooling)$state %>% head(4) # state effects
```

```
## (Intercept)  
## AK -0.10431698  
## AL -0.12079485  
## AR 0.02019649  
## AZ -0.13642398
```

```
coef(partial.pooling)$state %>% head(4) # state predictions
```

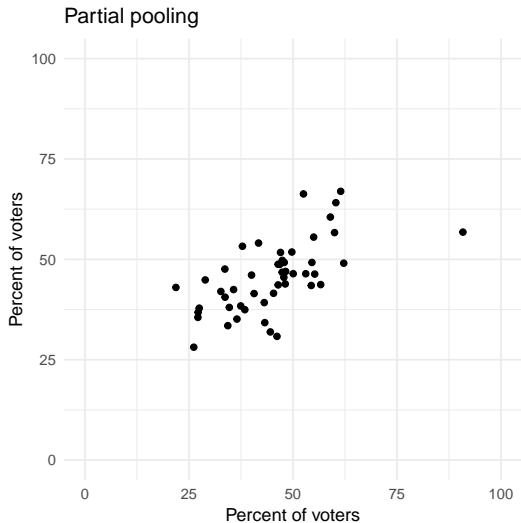
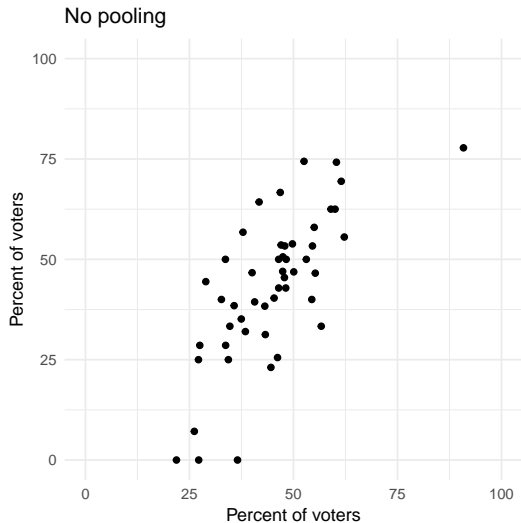
```
## (Intercept)  
## AK 0.3512025  
## AL 0.3347247  
## AR 0.4757160  
## AZ 0.3190955
```

Which is better?

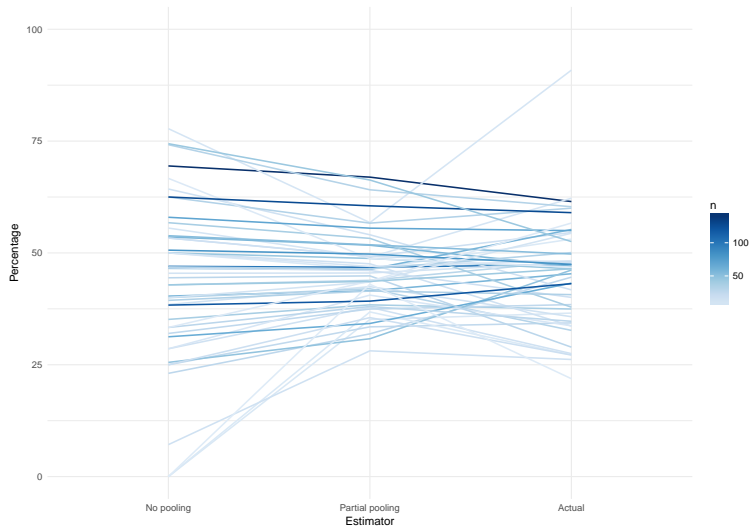
```
## # A tibble: 51 x 7
##   state name      actual post.stratified no.pooling partial.pooling    n
##   <fct> <chr>      <dbl>         <dbl>         <dbl>         <dbl> <int>
## 1 AK      Alaska      36.6           0           0           35.1     5
## 2 AL      Alabama     34.4          28.4         25.          33.5    24
## 3 AR      Arkansas    33.7          57.6         50.0         47.6    14
## 4 AZ      Arizona     44.6          23.0         23.1         31.9    26
## 5 CA      California  61.5          72.0         69.4         66.9   144
## 6 CO      Colorado    48.2          39.4         42.9         43.9    28
## 7 CT      Connecticut 54.6          58.1         53.3         49.2    15
## 8 DC      District of Columb~ 90.9          72.1         77.8         56.8     9
## 9 DE      Delaware    53.1          34.8         50.0         46.4     4
## 10 FL     Florida     47.4          47.3         47.0         46.8   100
## # ... with 41 more rows
```

The RMSE is about the same for *no pooling* (12.8%) and *complete pooling* (12.6%), but substantially smaller for *partial pooling* (9.5%).

2016 U.S. election: estimates vs. actuals



Shrinkage



What *exactly* is a random effects estimator?

The random effects/partial pooling estimate of a group mean is a weighted average of the fixed effects/no pooling estimate and the grand mean/complete pooling estimate.

$$\hat{\theta}_j^{\text{RE}} = w_j \bar{y}_j + (1 - w_j) \bar{y}$$

- ▶ The weight on the no pooling estimator \bar{y}_j is proportional to sample size n_j in group j and inversely proportional to the **within group variance**.
- ▶ The weight on the complete pooling is inversely proportional to the **between group variance**.
- ▶ Sometimes referred to as an **empirical Bayesian estimator**. The amount of shrinkage is data dependent.

Adding individual covariates: fixed effects

To get better predictions, let's add a covariate – **gender** – to the model. There are now three *fixed effects* estimators:

Complete pooling estimate a single slope and single intercept for all states

No pooling estimate separate intercepts and slopes for each state

Complete pooling of slopes, no pooling of intercepts estimate separate intercepts for each state and a common slope

(It rarely makes sense to estimate separate slopes and a common intercept, so we ignore this possibility.)

Complete pooling: common intercepts and slopes

$$y_i = \alpha + \beta x_i + \epsilon_i \quad \epsilon_i \stackrel{\text{iid}}{\sim} \text{Normal}(0, \sigma_\epsilon)$$

```
pew <- mutate(pew, female = ifelse(gender == "female", 1, 0))
fit1 <- lm(demvote ~ 1 + female, data = pew)
arm::display(fit1)
```

```
## lm(formula = demvote ~ 1 + female, data = pew)
##               coef.est coef.se
## (Intercept)  0.42      0.02
## female       0.14      0.02
## ---
## n = 1698, k = 2
## residual sd = 0.49, R-Squared = 0.02
```

$\hat{\beta} = 0.14$ implies a 14% gender gap (42% of men voted for Clinton vs. 56% of women).

No pooling: separate intercepts and slopes

$$y_i = \alpha_{j_i} + \beta_{j_i}x_i + \epsilon_i$$

```
fit2 <- lm(demvote ~ 0 + state + state:female, data = pew)
coef.fit2 <- as.matrix(coef(fit2))
round(coef(fit2)[c(1:5, 90:98)], 2)
```

```
##          stateAK          stateAL          stateAR          stateAZ          stateCA
##          0.00          0.22          0.40          0.29          0.67
## stateTN:female stateTX:female stateUT:female stateVA:female stateVT:female
##          0.00          -0.01          0.22          -0.01          -0.50
## stateWA:female stateWI:female stateWV:female stateWY:female
##          0.30          -0.04          -0.14          NA
```

It's impossible to estimate the slope in Wyoming:

```
##          state
## gender    AK AL AR AZ CA CO CT DC DE FL GA HI IA ID IL IN KS KY LA MA MD ME MI MN MO MS MT NC ND NE
## male      4  9 10 14 67 16  9  4  2 52 37  5  6  9 36 19  6 14  7 13 12 10 30 22 18  9  0 26  3  5
## female    1 15  4 12 77 12  6  5  2 48 20  4  8  5 22 18  7 16 18 19 19  1 26 22 19  6  0 21  1  2
##          state
## gender    NH NJ NM NV NY OH OK OR PA RI SC SD TN TX UT VA VT WA WI WV WY
## male      2 35  4  6 74 37 13 20 42  8 22  0 15 62 11 35  2 19 20  7  0
## female    1 34  4  9 54 27 14 12 39  2 11  0 12 58  5 17  1 24 22  7  1
```

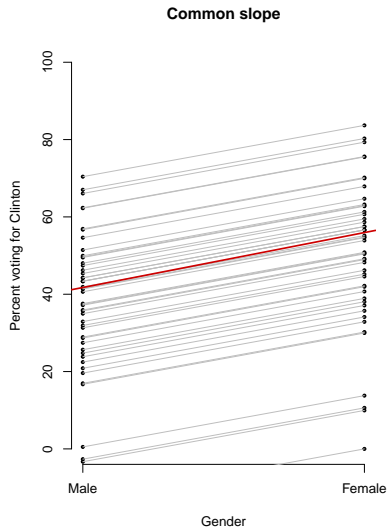
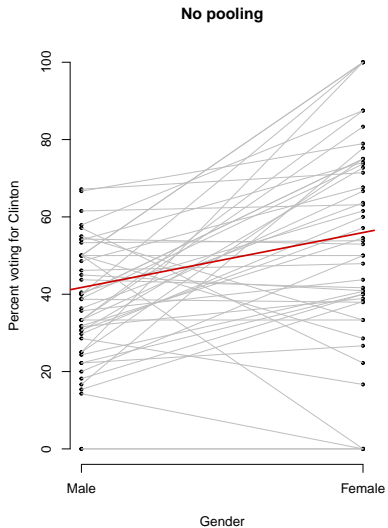
Separate intercepts with common slope

$$y_i = \alpha_{j_i} + \beta x_i + \epsilon_i$$

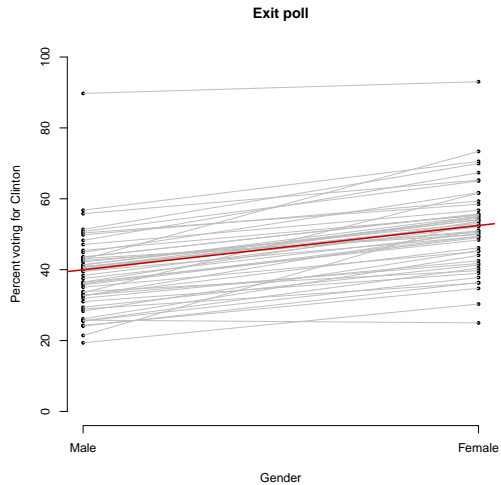
```
fit3 <- lm(demvote ~ state + female - 1, data = pew)
coef.fit3 <- coef(fit3)
names(coef.fit3) <- gsub("state", "", names(coef.fit3))
round(coef.fit3, 2)
```

```
##      AK      AL      AR      AZ      CA      CO      CT      DC      DE      FL      GA      HI
## -0.03  0.17  0.46  0.17  0.62  0.37  0.48  0.70  0.43  0.41  0.36  0.50
##      IA      ID      IL      IN      KS      KY      LA      MA      MD      ME      MI      MN
##  0.57  0.24  0.42  0.29  0.31  0.33  0.22  0.55  0.66  0.44  0.47  0.43
##      MO      MS      NC      ND      NE      NH      NJ      NM      NV      NY      OH      OK
##  0.50  0.41  0.20 -0.03  0.25  0.62  0.51  0.43  0.45  0.57  0.26  0.38
##      OR      PA      RI      SC      TN      TX      UT      VA      VT      WA      WI      WV
##  0.42  0.44  0.37  0.35  0.27  0.32  0.21  0.50  0.29  0.67  0.36  0.01
##      WY female
## -0.13  0.13
```

Which should you believe?



Gender gaps in 2016 Exit poll



Multilevel models with a single covariate

Varying intercepts, common slope

$$y_i = \mu + \alpha_{j_i} + \beta x_i + \epsilon_i$$

- ▶ The intercept for state j is $\mu + \alpha_j$.
- ▶ The slope (gender gap) is β (same for all states).

Varying intercepts and slopes

$$y_i = \mu + \alpha_{j_i} + \beta x_i + \gamma_{j_i} x_i + \epsilon_i$$

- ▶ The intercept for state j is $\mu + \alpha_j$.
- ▶ The slope (gender gap) for state j is $\beta + \gamma_j$ (varies).

Random effects: varying intercepts, common slopes

```
# Varying intercept and common slope
fit4 <- lmer(demvote ~ 1 + female + (1 | state), data = pew)
fit4
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: demvote ~ 1 + female + (1 | state)
## Data: pew
## REML criterion at convergence: 2388.584
## Random effects:
## Groups Name Std.Dev.
## state (Intercept) 0.1155
## Residual 0.4810
## Number of obs: 1698, groups: state, 49
## Fixed Effects:
## (Intercept) female
## 0.3936 0.1357
```

Random effects: varying intercepts and slopes (first try)

```
(fit5 <- lmer(demvote ~ 1 + female + (1 + female | state),  
  data = pew)) # fails to converge
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : unable  
## to evaluate scaled gradient
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model  
## failed to converge: degenerate Hessian with 1 negative eigenvalues
```

```
## Linear mixed model fit by REML ['lmerMod']  
## Formula: demvote ~ 1 + female + (1 + female | state)  
## Data: pew  
## REML criterion at convergence: 2408.288  
## Random effects:  
## Groups Name Std.Dev. Corr  
## state (Intercept) 0.0000  
## female 0.1548 NaN  
## Residual 0.4846  
## Number of obs: 1698, groups: state, 49  
## Fixed Effects:  
## (Intercept) female  
## 0.4174 0.1267  
## convergence code 0; 2 optimizer warnings; 0 lme4 warnings
```

Fixing the convergence failure (by centering covariates)

```
pew <- mutate(pew, female.c = female - 0.5)
fit6 <- lmer(demvote ~ 1 + female.c + (1 + female.c | state),
  data = pew)
fixef(fit6) # not comparable to prior models
```

```
## (Intercept)    female.c
##    0.4638900    0.1355841
```

```
head(coef(fit6)$state)
```

```
##      (Intercept)    female.c
## AK    0.3716471  0.10577811
## AL    0.3277785  0.06538934
## AR    0.5030967  0.16432114
## AZ    0.3117632  0.04284077
## CA    0.6576921  0.13720538
## CO    0.4378216  0.10209558
```

(No need to wrestle with lmer if you use Stan – which is up next.)

Unscale the estimates for comparability

```
fixef.fit6 <- fixef(fit6)
fixef.fit6[1] <- fixef.fit6[1] - 0.5 * fixef.fit6[2]
coef.fit6 <- coef(fit6)$state
coef.fit6[[1]] <- coef.fit6[[1]] - 0.5 * coef.fit6[[2]]
fixef.fit6
```

```
## (Intercept)    female.c
##    0.3960980    0.1355841
```

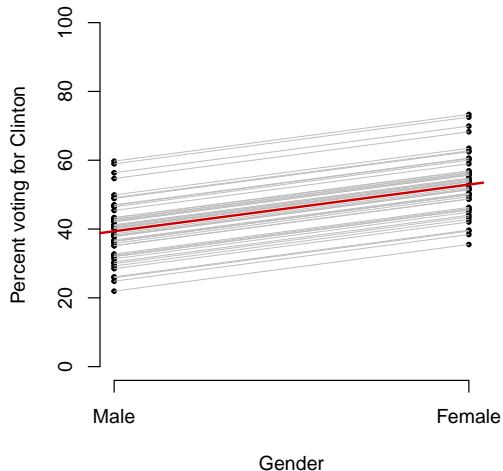
```
head(coef.fit6)
```

```
##      (Intercept)    female.c
## AK    0.3187581  0.10577811
## AL    0.2950838  0.06538934
## AR    0.4209361  0.16432114
## AZ    0.2903429  0.04284077
## CA    0.5890894  0.13720538
## CO    0.3867738  0.10209558
```

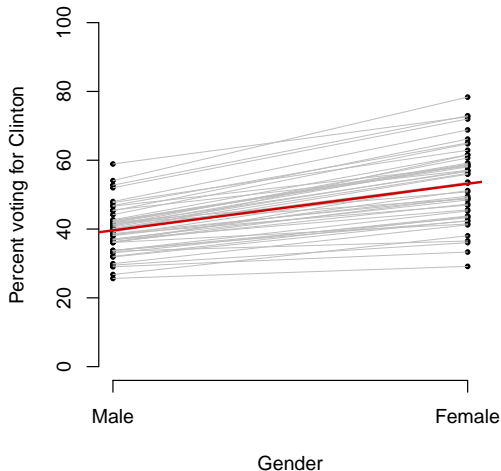
Comparing the random effects estimates

Gender gaps are correlated with state effects ($r = 0.56$).

Varying intercepts, common slope



Varying intercepts and slopes



Adding group-level covariates to the model

Add a **group-level covariate** (2012 vote) to the model. This is impossible in the no-pooling model, since group level variables are **collinear** with state indicators.

As before, i indexes respondents and j indexes states. The variables are:

y_i = respondent 2016 vote (1 = Clinton, 0 = other)

x_i = respondent gender (1 = female, 0 = male)

z_j = proportion in state voting for Obama in 2012

At the respondent level, both intercepts and slopes vary by group (state):

$$\textbf{Respondent model} \quad y_i = \alpha_{j_i} + \beta_{j_i} x_i + \epsilon_i \quad (i = 1, \dots, n)$$

The state intercepts and slopes are modelled as:

$$\textbf{Group model} \quad \alpha_j = \mu + \delta z_j + \eta_j \quad \beta_j = \gamma + \xi_j$$

The errors are all assumed to be iid normal with zero means:

$$\epsilon_i \stackrel{\text{iid}}{\sim} \text{Normal}(0, \sigma_\epsilon) \quad \eta_j \stackrel{\text{iid}}{\sim} \text{Normal}(0, \sigma_\eta) \quad \xi_j \stackrel{\text{iid}}{\sim} \text{Normal}(0, \sigma_\xi)$$

An equivalent version of the model

We can move covariates from the **upper** (group) level to the **lower** (respondent) level without changing anything:

$$y_i = \mu + \gamma x_i + \delta z_{j_i} + (\eta_{j_i} + \xi_{j_i} x_i + \epsilon_i)$$

The first three terms are **fixed effects**, while the last terms (in parentheses) are **random effects**.

The random effects in this formulation all have **mean zero**, which is required by `lmer`. This assumption is not restrictive – anything on the RHS of the group model with non-zero mean can be moved to the respondent level fixed effects.

Estimating models with covariates at both levels

Join group level covariates to respondent data:

```
obama12 <- votes12 %>%  
  mutate(obama12 = obama / turnout) %>%  
  select(state, obama12)  
pew <- left_join(pew, obama12, by = "state")  
fit7 <- lmer(demvote ~ 1 + female.c + obama12 + (1 + female.c | state), data = pew)
```

```
## lmer(formula = demvote ~ 1 + female.c + obama12 + (1 + female.c |  
##      state), data = pew)  
##           coef.est coef.se  
## (Intercept) -0.06      0.08  
## female.c      0.14      0.03  
## obama12       1.07      0.16  
##  
## Error terms:  
## Groups   Name      Std.Dev. Corr  
## state    (Intercept) 0.06  
##          female.c    0.09    0.35  
## Residual              0.48  
## ---  
## number of obs: 1698, groups: state, 49  
## AIC = 2370.6, DIC = 2329.2  
## deviance = 2342.9
```

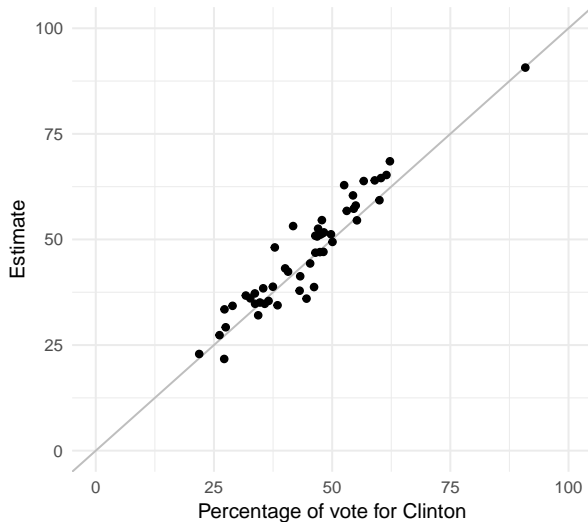
First attempt at MRP

We can use the R `predict` function to **impute** demvote onto cps:

```
cps <- cps %>%
  mutate(female = ifelse(gender == "female", 1, 0),
         female.c = female - 0.5) %>%
  left_join(obama12, by = "state")
prob <- predict(fit7, newdata = cps, allow.new.levels = TRUE)
mrp1 <- cps %>%
  mutate(prob = prob) %>%
  group_by(state) %>%
  summarize(mrp1 = 100 * weighted.mean(prob, weight))
estimates <- left_join(estimates, mrp1, by = "state")
head(estimates)
```

```
## # A tibble: 6 x 8
##   state name      actual post.stratified no.pooling partial.pooling      n mrp1
##   <fct> <chr>      <dbl>          <dbl>      <dbl>          <dbl> <int> <dbl>
## 1 AK     Alaska      36.6            0            0            35.1     5  35.4
## 2 AL     Alabama      34.4           28.4          25.           33.5    24  32.0
## 3 AR     Arkansas      33.7           57.6          50.0          47.6    14  37.2
## 4 AZ     Arizona       44.6           23.0          23.1          31.9    26  36.0
## 5 CA     California    61.5           72.0          69.4          66.9   144  65.3
## 6 CO     Colorado      48.2           39.4          42.9          43.9    28  47.1
```

Plotting first MRP estimates



The MRP estimates are now pretty good ($\text{RMSE} = 4.58$)!

Summary of multilevel models

- ▶ Multilevel models allow us to estimate models with large numbers of parameters and still get sensible results.
- ▶ Multilevel models **shrink** or **regularize** estimates.
 - ▶ Random effect estimates of group parameters are shrunk toward a “grand mean” effect.
- ▶ The amount of **shrinkage** in estimating a parameter depends upon:
 - ▶ How much data is available to estimate the parameter.
 - ▶ How much the parameter varies across groups.
- ▶ Substantial improvements can be made by adding **group-level predictors** (like past vote).
- ▶ Also possible to estimate multilevel **logistic regressions** using `glmer`, but we will focus on Bayesian estimation instead.

4. Bayesian inference

What is Bayesian statistics?



Thomas Bayes (1701-61)

Bayesian statistics is a method for combining **prior information** with **data** to draw inferences.

Probabilities are **subjective measures of uncertainty** and, in Bayesian inference, can be applied to *anything*, including events that have already happened.

Bayesian inference seems particularly well suited for situations where there is uncertainty about the sampling process and no **objective sampling distribution**.

Advantages and disadvantages of Bayesian inference



Cotton and Bayes
family vault
(Bunhill Fields, London)

Advantages

- ▶ Does not depend on having large samples
- ▶ Can incorporate prior (non-sample) information
- ▶ Better estimates of variance parameters
- ▶ Simpler conceptual framework, avoids *ad hockery*

Disadvantages

- ▶ Requires specification of prior probability distributions
- ▶ Inferences are not “objective”
- ▶ Better estimates of variance parameters
- ▶ Takes *much* more computational effort

Basic ideas

In Bayesian statistics, both data and parameters are considered to be random variables. The important distinction is between what is *observed* (the data, except possibly for missing values) and what isn't (parameters, such as regression coefficients).

As in classical statistics, we specify a *model* (probability distribution) for the data, which depends on some unknown parameters. In addition, we summarize any information that we might have about the unknown parameters using a *prior probability distribution*.

Bayesian inference uses Bayes theorem to combine the prior information with the data to obtain a *posterior probability distribution* for the parameters.

Notation for Bayesian analysis

Data $y = (y_1, \dots, y_n)$

Parameters $\theta = (\theta_1, \dots, \theta_k)$

Prior distribution $p(\theta) = p(\theta_1, \dots, \theta_k)$

Model $p(y|\theta)$ (*a.k.a.* the **likelihood function**)

Posterior distribution $p(\theta|y)$

Example: What is the average airfare to AAPOR in Denver?

Population parameters

- ▶ θ = population average fare
- ▶ σ = population s.d. = 150 (assumed to be known)

Data My airfare was $y_1 = 161$ and two friends spent $y_2 = 250$ and $y_3 = 489$.

$$\bar{y} = \frac{161 + 250 + 489}{3} = 300 \quad n = 3$$

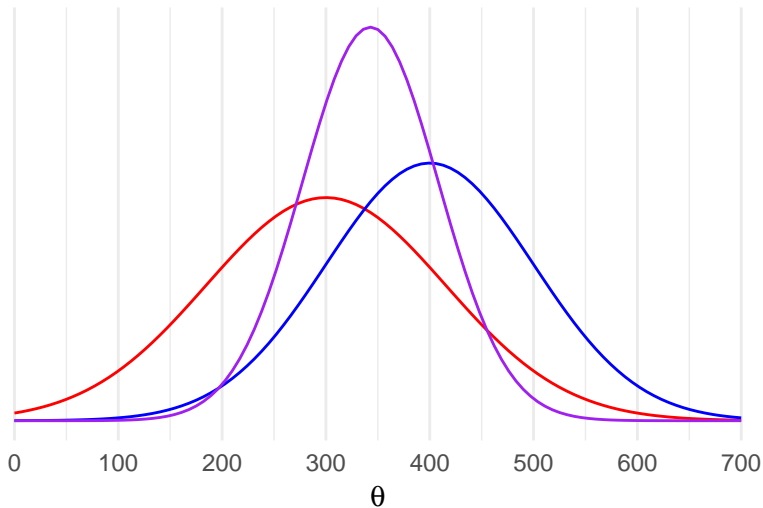
Model y_1, \dots, y_n are independent $\text{Normal}(\theta, 150)$ random variables

Prior information $\theta \sim \text{Normal}(400, 100)$ (so $P(200 < \theta < 600) \approx 95\%$)

Posterior distribution $\theta|y_1, y_2, y_3 \sim \text{Normal}(343, 65)$

The posterior distribution

From Bayes' Theorem, it can be shown that $\theta|y \sim \text{Normal}(343, 65)$.



Relationship between prior and posterior mean

In the normal model, the **posterior mean** is a weighted average of the **prior mean** and the **sample mean**.

$$\text{posterior mean} = \text{weight} \times \text{prior mean} + (1 - \text{weight}) \times \text{sample mean}$$

$$\text{weight} = \frac{\text{prior precision}}{\text{prior precision} + \text{sample precision}}$$

$$\text{prior precision} = \frac{1}{\text{prior variance}}$$

$$\text{sample precision} = \frac{n}{\text{data variance}}$$

When the sample size is large, the weight on the prior becomes small. However, when the sample size is small, the posterior **shrinks** the sample mean toward the prior.

Markov Chain Monte Carlo (MCMC)

For simple models (like the normal model with a normal prior), the posterior can be derived **analytically**. This is impossible for most models and priors, which limited the application of Bayesian methods before MCMC (about 1990).

MCMC is a technique for **simulating** draws from a probability distribution. With simulated draws, we can:

- ▶ Use a histogram or density plot to see what the **posterior distribution** looks like.
- ▶ Compute an estimate of the **posterior mean** (the average of the draws) or **posterior standard deviation** (the s.d. of the draws)
- ▶ Compute quantiles to obtain **credible intervals** (a.k.a. “confidence intervals”).

Programs like BUGS and Stan automate most of this process.

Instructions for installing Stan on MacOS

<https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started>

1. Install a C++ compiler (Xcode for MacOS or Rtools for Windows)
2. Create a Makevars file your .R folder (in your home directory):

```
CC=clang
CXX=clang++ -arch x86_64 -ftemplate-depth-256
CXXFLAGS=-O3 -mtune=native -march=native -Wno-unused-variable \
-Wno-unused-function -Wno-macro-redefined \
-Wno-unused-local-typedefs -Wno-c++11-inline-namespace \
-Wno-unneeded-internal-declaration -Wno-unknown-pragmas
```

3. Install rstan from CRAN

```
install.packages(c("rstan", "rstanarm"))
```

A first Stan program

Model $y_1, \dots, y_n \stackrel{\text{iid}}{\sim} \text{Normal}(\theta, \sigma)$ ($\sigma = 150$)

Prior $\theta \sim \text{Normal}(\theta_0, \omega_0)$ ($\theta_0 = 350$, $\omega_0 = 100$)

```
model_code <- "data {  
  int n;  
  real y[n];  
  real theta_0;  
  real<lower=0> omega_0;  
  real sigma;  
}  
parameters {  
  real theta;  
}  
model {  
  theta ~ normal(theta_0, omega_0);  
  for (i in 1:n) {  
    y[i] ~ normal(theta, sigma);  
  }  
}"
```


Components of a Stan program

Data (what you know)

```
data {  
  int n; // number of data points  
  real y[n]; // data on fares  
  real theta_0; // prior mean  
  real<lower=0> omega_0; // prior sd  
  real sigma; // sd of fares  
}
```

Parameters (what you don't know)

```
parameters {  
  real theta; // mean fare  
}
```

Model (joint distribution of data and parameters)

```
model {  
  theta ~ normal(theta_0, omega_0); // prior  
  for (i in 1:n)  
    y[i] ~ normal(theta, sigma); // likelihood  
}
```

Running a Stan program from R

To run a Stan program from R, you will need:

- ▶ Load `rstan` package (using the `library` function)
- ▶ Your Stan `model_code` (as a string or from an external file)
- ▶ A named list `data` (containing all of the items listed in the Stan data block)
- ▶ Call the `stan` function with `model_code` and `data` as arguments.

Stan returns draws from the posterior distribution of the parameters, arranged into a three-dimensional array (draws \times chains \times parameters).

Example (continued)

```
suppressMessages(library(rstan))
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
y <- c(161, 250, 489)
data <- list(y = y, n = length(y), theta_0 = 350,
             omega_0 = 150, sigma = 100)
sims <- stan(model_code = model_code, data = data,
             chains = 4, iter = 500, seed = 1234)
```

This will simulate four **chains** of length `iter = 500`.

By default, half of each chain is for **warmup**, so the result will be a $500 \times 4 \times 1$ array (iterations x chains x parameters).

Stan output

```
print(sims)
```

```
## Inference for Stan model: ba63e9dab118f6a2e4a91d41e5dc12b1.
## 4 chains, each with iter=500; warmup=250; thin=1;
## post-warmup draws per chain=250, total post-warmup draws=1000.
##
##           mean se_mean      sd   2.5%   25%   50%   75%  97.5% n_eff Rhat
## theta 307.88     2.65 52.74 212.15 271.19 306.06 342.62 415.30   395    1
## lp__   -3.40     0.03  0.67  -5.08  -3.57  -3.15  -2.98  -2.93   470    1
##
## Samples were drawn using NUTS(diag_e) at Mon May 14 11:05:02 2018.
## 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).
```

How to read Stan output

- ▶ `mean` is the **posterior mean** – estimated to be 302.94 for θ .
- ▶ `se_mean` is the **monte carlo standard error**.
- ▶ The value 2.63 tells you that only the first two digits are reliable – the point estimate is probably between 297 and 309.
- ▶ To make this smaller, redo the simulations with a larger value of `iter`.
- ▶ `sd` is the **posterior standard deviation** of θ – beyond the error in the computation of the point estimate.
- ▶ The percentages are **posterior quantiles**.
- ▶ The 95% **credible interval** is 200.47 to 404.01.
- ▶ `Rhat` is a **convergence** diagnostic. You should look for values near one.

Weakly informative priors

Bayesian inference is often criticized for being *subjective* since different priors will yield different inferences.

Non-informative priors can be used to reproduce many classical frequentist (“objective”) results. Sometimes these priors are not *proper* (e.g., θ is uniformly distributed on \mathbb{R}).

Weakly informative priors purposely include less information than we actually have, but still provide some *regularization* (shrinkage).

For example, logistic regression coefficients are rarely greater than five or six (the difference between the top and bottom decile is about 4.4, between the .95 and .05 quantiles is about 5.9). In a linear model, a “six sigma” effect would be considered extremely large by any standard.

Priors on standard deviations

In real world applications, we do not know the standard deviation of the data, so there are two unknown parameters (μ and σ).

Since $\sigma > 0$, we need to use a prior distribution for σ that assigns zero probability to negative numbers. There are many possible choices.

Traditionally, variance priors were selected for computational convenience, but with modern MCMC methods this is no longer necessary.

Gelman, "Prior distributions for variance parameters in hierarchical models," *Bayesian Analysis* (2006) recommends the *Folded noncentral t* distribution (the distribution of the absolute value of a noncentral t random variable). A *half normal* prior may also be suitable.

Stan code for estimating both mean and SD

Move sigma from the data block to the parameters block.

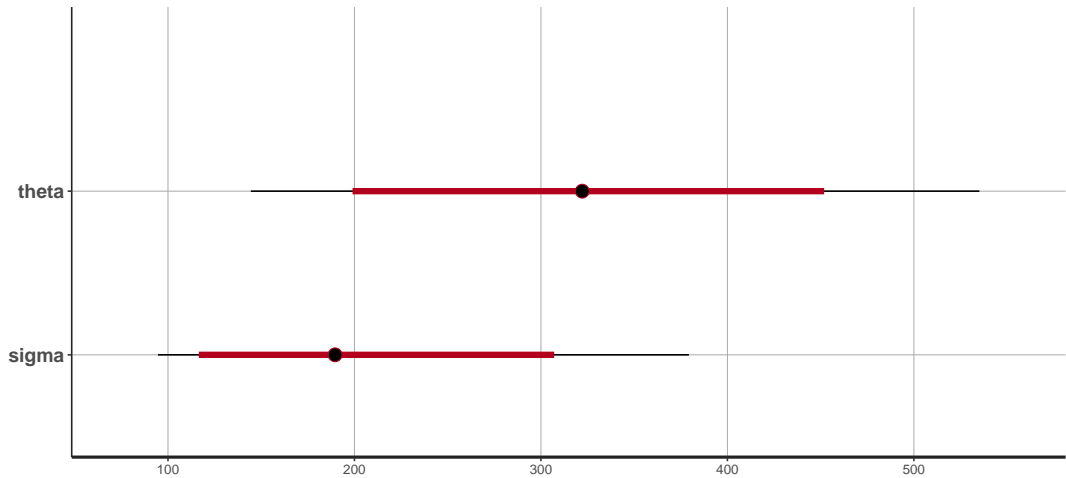
```
data <- list(y = y, n = length(y), theta_0 = 350, omega_0 = 150)
model_code <- "data {
  int n;
  real y[n];
  real theta_0;
  real omega_0;
}
parameters {
  real theta;
  real<lower=0> sigma; // moved to parameter block
}
model {
  theta ~ normal(theta_0, omega_0);
  sigma ~ normal(150, 150);
  y ~ normal(theta, sigma); // assumed iid
}"
sims <- stan(model_code = model_code, data = data,
  iter = 500, seed = 1234)
```


Output from two-parameter model

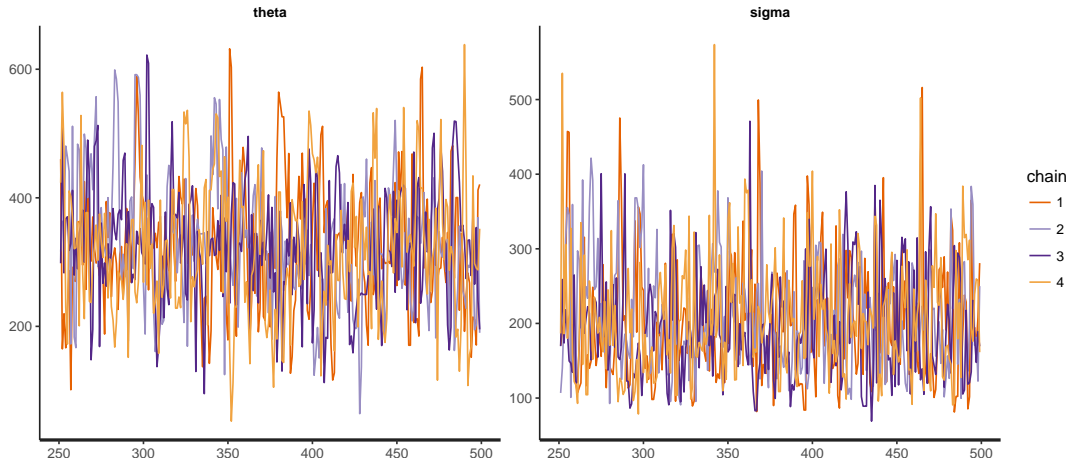
```
print(sims)
```

```
## Inference for Stan model: 35efec9e0aa9783871609df605a689bd.  
## 4 chains, each with iter=500; warmup=250; thin=1;  
## post-warmup draws per chain=250, total post-warmup draws=1000.  
##  
##           mean se_mean      sd   2.5%   25%   50%   75%  97.5% n_eff Rhat  
## theta 325.36     4.69 97.93 144.51 259.09 322.06 385.01 535.02   437 1.02  
## sigma 204.36     3.09 77.36  94.82 145.37 190.16 250.05 378.86   626 1.00  
## lp__ -12.34     0.05  0.97 -15.10 -12.79 -12.05 -11.62 -11.33   413 1.01  
##  
## Samples were drawn using NUTS(diag_e) at Mon May 14 11:05:27 2018.  
## 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).
```

Graphical display of parameters



Traceplot



Hierarchical priors

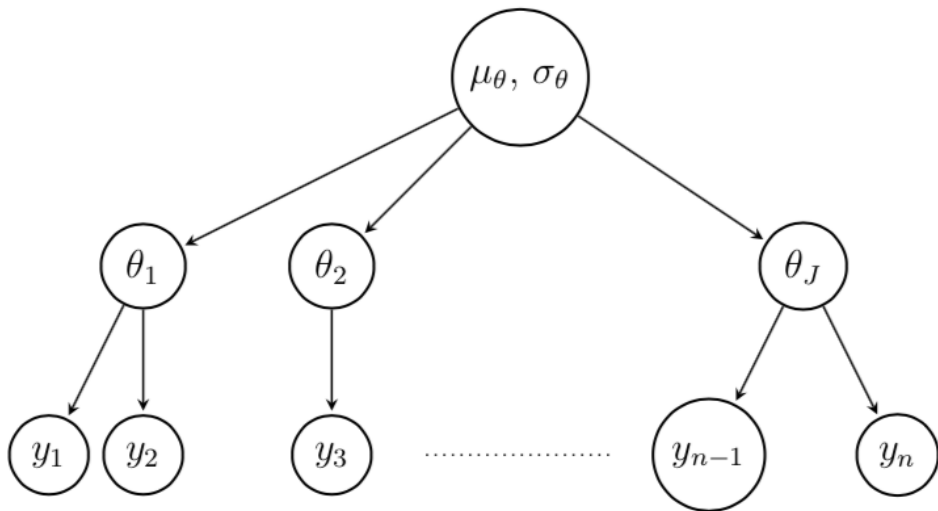
Multilevel models contain many parameters and it is difficult to specify **high dimensional priors**. In a model for respondents in U.S. states, we might have intercepts for each state, each of which would need to be assigned a prior distribution.

Instead, we use **hierarchical priors** where the priors for the model parameters depend upon **hyperparameters**. That is, we model relationships between the many model parameters using a second level model.

Hierarchical priors are **data dependent**. We infer relationships between model parameters using observed patterns in the data.

For example, we can model state intercepts in a voting model using past vote. To the extent that 2016 respondents in states that voted Republican in 2012 also say they will vote Republican in 2016, we can infer a correlation between past vote and state intercepts in the 2016 model.

Example: a simple two-level model



Notation for hierarchical normal model

Individuals indexed by $i = 1, \dots, n$.

Groups indexed by $j = 1, \dots, J$. Individual i belongs to group j_i

Data Model $y_i \stackrel{\text{ind}}{\sim} \text{Normal}(\theta_{j_i}, \sigma_y)$

Group Model $\theta_j \stackrel{\text{ind}}{\sim} (\mu_\theta, \sigma_\theta)$ (Hierarchical prior)

Hyper Prior $\mu_\theta \sim \text{Uniform}(-\infty, \infty)$ and $\sigma_\theta \sim \text{Normal}_+(0, 5)$

We can then use Bayes' Theorem to derive the posterior distribution of both the group parameters $(\theta_1, \dots, \theta_J)$ and the hyperparameters $(\mu_\theta, \sigma_\theta)$.

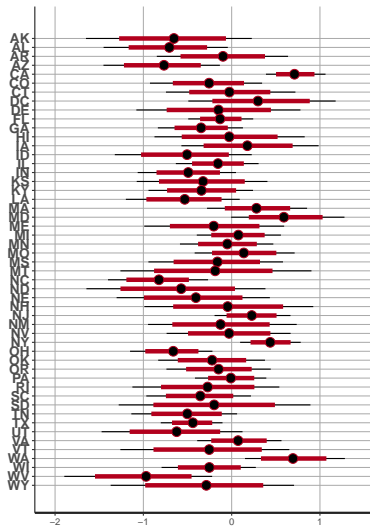
First hierarchical model in Stan

```
data <- with(pew, list(y = demvote, group = as.integer(state),
  n = nrow(pew), J = nlevels(state)))
code <- "data {
  int n; // number of respondents
  int J; // number of groups (states)
  int<lower=0, upper=1> y[n]; // demvote
  int<lower=1, upper=J> group[n]; // state index
}
parameters {
  real mu_theta; // hyper parameters
  real<lower=0> sigma_theta;
  vector[J] theta; // group parameters
}
model {
  sigma_theta ~ normal(0, 5);
  for (j in 1:J)
    theta[j] ~ normal(mu_theta, sigma_theta);
  for (i in 1:n)
    y[i] ~ bernoulli_logit(theta[ group[i] ]);
}"
sims <- stan(model_code = code, data = data)
```

Stan output

```
## Inference for Stan model: 144ca4c42440f974d33a212a411e7043.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##               mean se_mean   sd    2.5%    25%    50%    75%   97.5% n_eff
## mu_theta      -0.19    0.00 0.10   -0.39   -0.25   -0.19   -0.12    0.00 3703
## sigma_theta    0.53    0.00 0.09    0.36    0.46    0.52    0.59    0.73 1202
## theta[1]      -0.67    0.01 0.48   -1.65   -0.97   -0.65   -0.35    0.23 4000
## theta[2]      -0.72    0.01 0.35   -1.45   -0.95   -0.71   -0.47   -0.04 4000
## theta[3]      -0.10    0.01 0.38   -0.85   -0.35   -0.10    0.15    0.64 4000
## theta[4]      -0.78    0.01 0.34   -1.45   -1.01   -0.77   -0.55   -0.13 4000
## theta[5]       0.72    0.00 0.17    0.39    0.60    0.71    0.83    1.06 4000
## theta[6]      -0.26    0.01 0.32   -0.92   -0.46   -0.26   -0.05    0.34 4000
## theta[7]      -0.02    0.01 0.36   -0.74   -0.25   -0.03    0.21    0.72 4000
## theta[8]       0.32    0.01 0.43   -0.49    0.03    0.30    0.60    1.18 4000
## theta[9]      -0.14    0.01 0.47   -1.08   -0.45   -0.15    0.16    0.78 4000
## theta[10]     -0.13    0.00 0.18   -0.49   -0.25   -0.13   -0.01    0.24 4000
## theta[11]     -0.35    0.00 0.24   -0.84   -0.51   -0.35   -0.19    0.13 4000
## theta[12]     -0.02    0.01 0.43   -0.87   -0.29   -0.03    0.25    0.83 4000
## theta[13]     -0.18    0.01 0.40   -0.57   -0.09    0.18    0.45    0.98 4000
## theta[14]     -0.52    0.01 0.39   -1.32   -0.77   -0.51   -0.26    0.23 4000
## theta[15]     -0.16    0.00 0.23   -0.63   -0.31   -0.16    0.00    0.30 4000
## theta[16]     -0.49    0.00 0.28   -1.06   -0.67   -0.49   -0.30    0.05 4000
## theta[17]     -0.33    0.01 0.38   -1.08   -0.58   -0.32   -0.07    0.40 4000
## theta[18]     -0.34    0.00 0.30   -0.94   -0.54   -0.34   -0.14    0.24 4000
## theta[19]     -0.54    0.01 0.33   -1.20   -0.76   -0.53   -0.31    0.09 4000
## theta[20]      0.29    0.00 0.29   -0.28    0.09    0.28    0.48    0.85 4000
## theta[21]      0.60    0.01 0.33   -0.01    0.38    0.59    0.81    1.28 4000
## theta[22]     -0.20    0.01 0.40   -0.99   -0.46   -0.20    0.06    0.60 4000
## theta[23]      0.08    0.00 0.24   -0.40   -0.08    0.08    0.23    0.56 4000
## theta[24]     -0.05    0.00 0.26   -0.59   -0.22   -0.05    0.13    0.47 4000
## theta[25]      0.14    0.00 0.28   -0.42   -0.05    0.14    0.32    0.71 4000
```


A plot of the state estimates



Add a covariate to the model

```
data <- with(pew, list(y = demvote, group = as.integer(state),
  x = female, n = nrow(pew), J = nlevels(state)))
code <- "data {
  int n; // number of respondents
  int J; // number of groups (states)
  int<lower=0, upper=1> y[n]; // demvote
  int<lower=1, upper=J> group[n]; // state index
  vector[n] x; // added covariate (gender)
}
parameters {
  real mu_alpha; // mean intercept
  real<lower=0> sigma_alpha; // sd intercept
  real beta; // coefficient of gender
  vector[J] alpha; // group intercepts
}
model {
  sigma_alpha ~ normal(0, 5);
  for (j in 1:J)
    alpha[j] ~ normal(mu_alpha, sigma_alpha);
  for (i in 1:n)
    y[i] ~ bernoulli_logit(alpha[group[i]] + beta * x[i]);
}"
sims <- stan(model_code = code, data = data)
```

Simplifying the computations

Make a `model.matrix` X in R for fixed effects.

```
X <- model.matrix(~ 1 + age4 + gender + race3 + educ4 +  
  region + qlogis(obama12), data = pew)
```

```
##      (Intercept) age430-44 age445-64 age465+ genderfemale race3black race3hispanic  
## 1             1         0         1         0             0             0             0  
## 2             1         0         0         1             1             0             0  
## 3             1         0         1         0             0             0             0  
## 4             1         0         0         1             0             0             0  
##      educ4some col educ4col grad educ4postgrad regionSouth regionNorth Central  
## 1             0             0             0             0             0             1  
## 2             0             0             1             1             0             0  
## 3             1             0             0             1             0             0  
## 4             1             0             0             1             0             0  
##      regionWest qlogis(obama12)  
## 1             0         0.1061918  
## 2             0         0.3477157  
## 3             0         0.3477157  
## 4             0        -0.3481398
```

Simplifying the computations (continued)

Add $X = X$ to the R data list and modify the Stan code to use matrix multiplication:

```
data {  
  ...  
  matrix[n, k] X; // covariate matrix  
}  
parameters {  
  vector[k] beta; // fixed effects  
  ...  
}  
model {  
  vector[n] Xb;  
  Xb = X * beta;  
  ...  
}
```

Simplifying the computations (continued)

- ▶ Move hierarchical means into fixed effects, so mean of random effects is zero.
- ▶ Random effects are usually easier to program directly in Stan.
- ▶ Sometimes helpful to standardize parameters and multiply by the SD.
- ▶ Stan's default is to assume elements of a vector are independent.

```
parameters {  
  real<lower=0> sigma_alpha; // sd intercept  
  vector[J] alpha; // group intercepts  
  ...  
}  
model {  
  sigma_alpha ~ normal(0.2, 1); // prior for sd  
  alpha ~ normal(0, 1); // standardized intercept  
  ...  
  for (i in 1:n)  
    Xb[i] += sigma_alpha * alpha[ group[i] ];  
  y ~ bernoulli_logit(Xb);  
}
```

Hierarchical model with multiple covariates

```
model_code <- "data {  
  int n; // number of respondents  
  int k; // number of covariates  
  matrix[n, k] X; // covariate matrix  
  int<lower=0, upper=1> y[n]; // outcome (demvote)  
  int J; // number of groups (states)  
  int<lower=1, upper=J> group[n]; // group index  
}  
parameters {  
  vector[k] beta; // fixed effects  
  real<lower=0> sigma_alpha; // sd intercept  
  vector[J] alpha; // group intercepts  
}  
model {  
  vector[n] Xb;  
  beta ~ normal(0, 4);  
  sigma_alpha ~ normal(0.2, 1); // prior for sd  
  alpha ~ normal(0, 1); // standardized intercepts  
  Xb = X * beta;  
  for (i in 1:n)  
    Xb[i] += sigma_alpha * alpha[ group[i] ];  
  y ~ bernoulli_logit(Xb);  
}"
```

Estimating the model in R

```
X <- model.matrix(~ 1 + age4 + gender + race3 + educ4 +  
  region + qlogis(obama12), data = pew)  
data <- list(n = nrow(X), k = ncol(X), X = X, y = pew$demvote,  
  J = nlevels(pew$state), group = as.integer(pew$state))  
sims <- stan(model_code = model_code, data = data,  
  seed = 1234)
```

Coefficients in the output are in the same order specified in the `model` block:

```
parameters {  
  vector[J] alpha; // group intercepts  
  real<lower=0> sigma_alpha; // sd intercept  
  vector[k] beta; // fixed effects  
}
```

```
## [1] "beta[1]"      "beta[2]"      "beta[3]"      "beta[4]"      "beta[5]"      "beta[6]"      "beta[7]"  
## [8] "beta[8]"      "beta[9]"      "beta[10]"     "beta[11]"     "beta[12]"     "beta[13]"     "beta[14]"  
## [15] "sigma_alpha"  "alpha[1]"     "alpha[2]"     "alpha[3]"     "alpha[4]"     "alpha[5]"     "alpha[6]"  
## [22] "alpha[7]"     "alpha[8]"     "alpha[9]"     "alpha[10]"    "alpha[11]"    "alpha[12]"    "alpha[13]"  
## [29] "alpha[14]"    "alpha[15]"    "alpha[16]"    "alpha[17]"    "alpha[18]"    "alpha[19]"    "alpha[20]"  
## [36] "alpha[21]"    "alpha[22]"    "alpha[23]"    "alpha[24]"    "alpha[25]"    "alpha[26]"    "alpha[27]"  
## [43] "alpha[28]"    "alpha[29]"    "alpha[30]"    "alpha[31]"    "alpha[32]"    "alpha[33]"    "alpha[34]"  
## [50] "alpha[35]"    "alpha[36]"    "alpha[37]"    "alpha[38]"    "alpha[39]"    "alpha[40]"    "alpha[41]"  
## [57] "alpha[42]"    "alpha[43]"    "alpha[44]"    "alpha[45]"    "alpha[46]"    "alpha[47]"    "alpha[48]"  
## [64] "alpha[49]"    "alpha[50]"    "alpha[51]"    "lp_--"
```

Rename the coefficients for easier reading

```
coef.names <- c(colnames(X), "sigma_alpha", levels(pew$state), "lp__")
names(sims) <- coef.names
```

```
## [1] "(Intercept)"      "age430-44"      "age445-64"      "age465+"
## [5] "genderfemale"     "race3black"     "race3hispanic"  "educ4some col"
## [9] "educ4col grad"    "educ4postgrad"  "regionSouth"     "regionNorth Central"
## [13] "regionWest"       "qlogis(obama12)" "sigma_alpha"     "AK"
## [17] "AL"               "AR"             "AZ"              "CA"
## [21] "CO"               "CT"             "DC"              "DE"
## [25] "FL"               "GA"             "HI"              "IA"
## [29] "ID"               "IL"             "IN"              "KS"
## [33] "KY"               "LA"             "MA"              "MD"
## [37] "ME"               "MI"             "MN"              "MO"
## [41] "MS"               "MT"             "NC"              "ND"
## [45] "NE"               "NH"             "NJ"              "NM"
## [49] "NV"               "NY"             "OH"              "OK"
## [53] "OR"               "PA"             "RI"              "SC"
## [57] "SD"               "TN"             "TX"              "UT"
## [61] "VA"               "VT"             "WA"              "WI"
## [65] "WV"               "WY"             "lp__"
```


Summary of fixed effect estimates

```
print(sims, par = "beta")
```

```
## Inference for Stan model: 023e7c53b6c68b914aff51f65bf5e090.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##               mean se_mean   sd  2.5%  25%   50%   75% 97.5% n_eff Rhat
## (Intercept)    -0.80     0.00  0.25 -1.29 -0.97 -0.80 -0.64 -0.32 2671   1
## age430-44       -0.14     0.00  0.20 -0.55 -0.28 -0.15 -0.01  0.28 3573   1
## age445-64       -0.35     0.00  0.18 -0.71 -0.47 -0.35 -0.23  0.00 3482   1
## age465+         -0.18     0.00  0.19 -0.55 -0.31 -0.18 -0.05  0.19 3610   1
## genderfemale    0.64     0.00  0.11  0.43  0.56  0.64  0.71  0.86 4000   1
## race3black      3.10     0.01  0.32  2.50  2.89  3.09  3.31  3.76 4000   1
## race3hispanic   1.12     0.00  0.21  0.72  0.98  1.12  1.27  1.53 4000   1
## educ4some col   0.09     0.00  0.16 -0.22 -0.01  0.09  0.20  0.41 4000   1
## educ4col grad   0.48     0.00  0.16  0.17  0.37  0.48  0.58  0.80 4000   1
## educ4postgrad   1.07     0.00  0.17  0.73  0.95  1.07  1.18  1.40 4000   1
## regionSouth     -0.27     0.00  0.23 -0.72 -0.42 -0.26 -0.12  0.17 2637   1
## regionNorth Central -0.08  0.00  0.22 -0.51 -0.22 -0.08  0.07  0.37 2825   1
## regionWest      0.11     0.00  0.23 -0.35 -0.03  0.12  0.26  0.55 2533   1
## qlogis(obama12)  0.95     0.00  0.22  0.51  0.80  0.94  1.09  1.38 4000   1
##
## Samples were drawn using NUTS(diag_e) at Mon May 14 11:07:06 2018.
## 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).
```

Predictive distributions: imputation of survey variables for the population

- ▶ The final step in MRP is to **impute** vote for the entire population.
 - ▶ The sample is a trivial proportion of the population.
 - ▶ We need to impute the survey variable to everyone **not** surveyed.
- ▶ The **posterior predictive distribution** $p(\tilde{y}|y)$ is the conditional distribution of a **new** draw \tilde{y} from the model, conditional upon the **observed** data y .
- ▶ This requires averaging $p(\tilde{y}|\theta)$ over the posterior distribution $p(\theta|y)$, *i.e.*, over the uncertainty in both \tilde{y} and θ .
- ▶ Contrast this with
 - ▶ **Regression imputation** the expected value of \tilde{y} is used
 - ▶ **Plug-in methods** a point estimate is substituted for the unknown parameter.

Imputation in Stan

Munge the population data in R

```
X0 <- model.matrix(~ 1 + age4 + gender + race3 + educ4 +  
  region + qlogis(obama12), data = cps)  
data <- list(n = nrow(X), k = ncol(X), X = X, y = pew$demvote,  
  J = nlevels(pew$state), group = as.integer(pew$state),  
  N = nrow(X0), X0 = X0, group0 = as.integer(cps$state))
```

and add to the Stan data block:

```
data {  
  ...  
  // add population data definitions  
  int N; // number of rows in population (cps)  
  matrix[N, k] X0; // covariates in population  
  int<lower=1, upper=J> group0[N]; // group index in population  
}
```

The generated quantities block in Stan

Tell Stan what you want to impute and how to create the imputations.

```
generated quantities {  
  int<lower=0, upper=1> yimp[N];  
  {  
    vector[N] Xb0;  
    Xb0 = X0 * beta;  
    for (i in 1:N)  
      yimp[i] = bernoulli_logit_rng(Xb0[i] + sigma_alpha * alpha[ group0[i] ] );  
  }  
}
```

Note the use of the `bernoulli_logit_rng` (random number generator) function to draw from the posterior predictive distribution. The `generated quantities` block cannot contain any distributions (indicated by `~`).

The complete Stan program

```
model_code <- "data {
  int n; // number of respondents
  int k; // number of covariates
  matrix[n, k] X; // covariate matrix
  int<lower=0, upper=1> y[n]; // outcome (demvote)
  int J; // number of groups (states)
  int<lower=1, upper=J> group[n]; // group index
  int N; // population size
  matrix[N, k] X0; // population covariates
  int group0[N]; // group index in population
}
parameters {
  vector[k] beta; // fixed effects
  real<lower=0> sigma_alpha; // sd intercept
  vector[J] alpha; // group intercepts
}
model {
  vector[n] Xb;
  beta ~ normal(0, 4);
  sigma_alpha ~ normal(0.2, 1);
  alpha ~ normal(0, 1);
  Xb = X * beta;
  for (i in 1:n)
    Xb[i] += sigma_alpha * alpha[ group[i] ];
  y ~ bernoulli_logit(Xb);
}
generated quantities {
  int<lower=0, upper=1> yimp[N];
  {
    vector[N] Xb0;
    Xb0 = X0 * beta;
    for (i in 1:N)
      yimp[i] = bernoulli_logit_rng(Xb0[i] + sigma_alpha * alpha[ group0[i] ]);
  }
}
```

Extracting the simulations

Stan has imputed 4000 values for each of the rows in `cps`. We sample 500 (much more than necessary, but it's still fast).

```
imputations <- extract(sims, pars = "yimp")$yimp[sample(
  nrow(sims), size = 500), ]
get_state_estimates <- function(imputations) {
  state_by_clinton <- function(imputed_values) 100 * prop.table(
    xtabs(weight ~ state + imputed_values, data = cps), 1)[,"1"]
  state_estimates <- apply(imputations, 1, state_by_clinton)
  apply(state_estimates, 1, mean)
}
estimates$mrp2 <- get_state_estimates(imputations)
RMSE["mrp2"] <- with(estimates, rmse(mrp2, actual))
```

Now we can perform any analyses we wish on the imputed `cps` data and average the results over the 10 imputed datasets to get point estimates.

The easy way with rstanarm

- ▶ Rstanarm is an R package that writes and executes Stan code for you.
- ▶ It uses the same notation as lme4 for specifying multilevel models.
- ▶ For example, to estimate the same model as mrp2, use the following code:

```
library(rstanarm)
fit <- stan_glmer(demvote ~ 1 + age4 + gender + race3 + educ4 +
  region + qlogis(obama12) + (1 | state), data = pew, family = binomial)
```

- ▶ The function posterior_predict in rstanarm substitutes for the usual predict function in R:

```
imputations <- posterior_predict(fit, draws = 500,
  newdata = select(cps, age4, gender, race3, educ4, region, obama12, state))
```

(This creates a matrix imputations of dimension draws x nrow(newdata).)

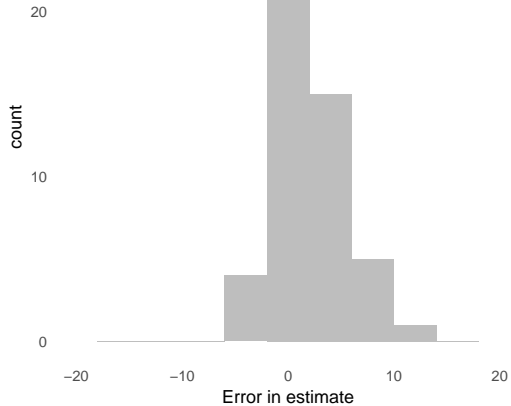
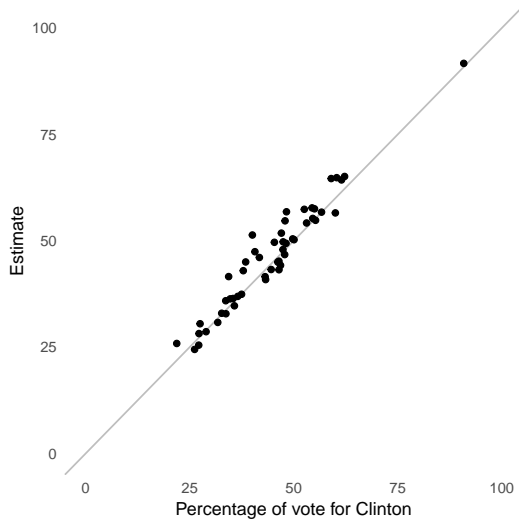
- ▶ Extract the estimates using get_state_estimates.

The complete program in rstanarm

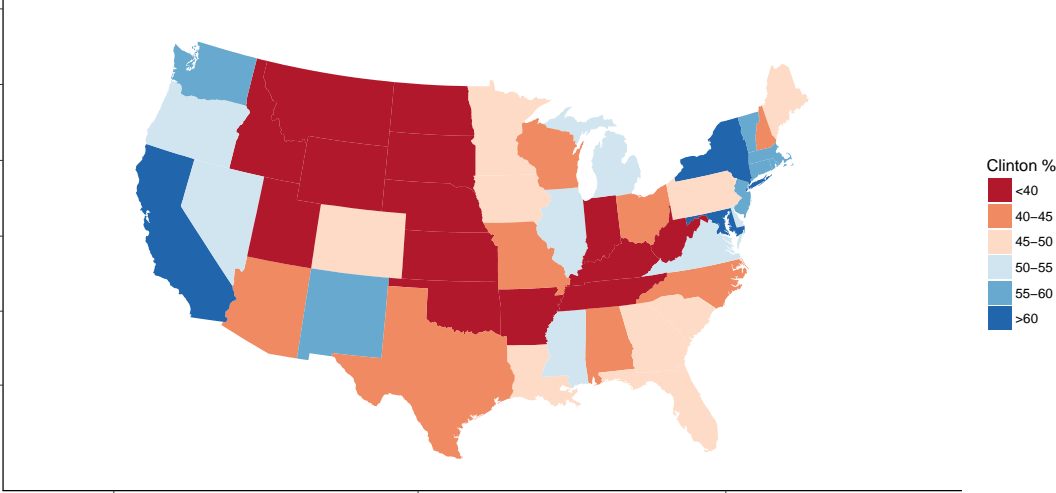
```
library(rstanarm)
fit <- stan_glmer(demvote ~ 1 + age4 + gender + race3 + educ4 +
  region + qlogis(obama12) + (1 | state), data = pew, family = binomial)
cpstmp <- cps %>%
  select(age4, gender, race3, educ4, region, obama12, state)
imputations <- posterior_predict(fit, draws = 500,
  newdata = select(cps, age4, gender, race3, educ4, region, obama12, state))
estimates$mrp3 <- get_state_estimates(imputations)
RMSE["mrp3"] <- with(estimates, rmse(mrp3, actual))
RMSE
```

| ## complete.pooling | no.pooling | partial.pooling | mrp1 | mrp2 |
|---------------------|------------|-----------------|----------|----------|
| ## 12.632470 | 12.799821 | 9.483346 | 4.580910 | 3.558746 |
| ## mrp3 | | | | |
| ## 3.647318 | | | | |

Accuracy of state level estimates



What the map now looks like



5. Advanced Topics

Where to from here?

1. More complicated models
2. More than two response categories in survey variable
3. More than two survey variables
4. Missing variables from the post-stratification
5. Computation of standard errors
6. The missing details

More complicated models

The models considered so far have been very simple. Instead, we might:

- ▶ Incorporate more **interactions** (such as race x region)
- ▶ Allow both **slopes** and **intercepts** to vary across states
- ▶ Add additional **levels** to the model (e.g., counties within states within regions)

This is not difficult with `rstanarm`, but as the next example demonstrates, there is no guarantee that more complicated models will perform better.

Allow race and gender effects to vary across states

```
fit <- stan_glmer(demvote ~ 1 + age4 + gender + race3 + educ4 +  
  region + qlogis(obama12) + (1 + gender + race3 | state),  
  data = pew, family = binomial)  
cpstmp <- cps %>%  
  select(age4, gender, race3, educ4, region, obama12, state)  
imputations <- posterior_predict(fit, draws = 500,  
  newdata = select(cps, age4, gender, race3, educ4, region, obama12, state))  
estimates$mrp4 <- get_state_estimates(imputations)  
RMSE["mrp4"] <- with(estimates, rmse(mrp4, actual))  
RMSE
```

| | | | | |
|---------------------|------------|-----------------|----------|----------|
| ## complete.pooling | no.pooling | partial.pooling | mrp1 | mrp2 |
| ## 12.632470 | 12.799821 | 9.483346 | 4.580910 | 3.558746 |
| ## mrp3 | mrp4 | | | |
| ## 3.647318 | 3.909148 | | | |

More than two response categories in the survey variable

In real applications, there are almost always more than two response categories, e.g. vote for Clinton, Trump, Johnson, Stein, do not vote, undecided, etc.

A simple solution is to estimate a set of binary response models (e.g., Clinton vs. not Clinton, Trump vs. not Trump, etc.), ignoring correlations between the parameters of these models.

A better solution is a **multinomial logit model**. When the response variable y_i takes values $q = 0, 1, \dots, Q$, a flexible model is

$$P(y_i = q) = \frac{e^{\beta_q^T x_i}}{\sum_{r=0}^Q e^{\beta_r^T x_i}}$$

See Section 9.6 of the Stan Reference Manual for code to implement this model. The R `brms` package can also estimate this model.

More than two survey variables

Most surveys contain more than a single question, so there are more than two variables to model. For example, pre-election polls contain questions about likelihood of voting, preferred candidate, and policy preferences.

Estimating separate MRP models for several variables can yield inconsistent results. Each estimate is equivalent to creating a separate weight for each variable.

In principle, we can specify bivariate or multivariate response models, yielding a single, consistent set of estimates for the marginal and joint distributions. In practice, this can be quite challenging and is an active research area.

Missing variables in the post-stratification

Population data often are **missing** for some predictors of survey outcomes. For example, voting models can benefit substantially from having individual level data on past vote. The assumption of **ignorable nonresponse** or selection is usually more plausible conditional upon a larger set of variables.

A reasonable approach is to **impute** missing variables onto PUMS. For instance, the 2012 exit poll can be used to impute candidate preference for voters in the 2012 CPS Registration and Voting Supplement. Usually single imputation, followed by raking of the marginals to known vote totals, performs well.

Computation of standard errors

When post-stratifying to **known population counts**, the only source of error is the estimate of the means of the survey variable in the post-strata. This is reflected in the posterior standard deviation of the draws and in the simulations of generated quantities.

When imputing onto a **public use microdata survey** (like CPS), there is an additional source of error (the sampling error in the PUMS survey).

There is a simple formula for calculating the combined variance:

- ▶ Create a set of imputed datasets.
- ▶ For each imputed dataset, compute the variance with the usual complete cases formula and average these.
- ▶ Also, compute the variance of the estimates across the imputed datasets.
- ▶ Add the two variances (adjusting for degrees of freedom)

With M imputations $\hat{\theta}_1, \dots, \hat{\theta}_M$:

$$\text{Var}(\bar{\theta}) = \frac{1}{M} \sum_i \widehat{\text{s.e.}}^2(\hat{\theta}_i) + \frac{M+1}{M} \frac{1}{M-1} \sum_{i=1}^M (\hat{\theta}_i - \bar{\theta})^2$$

References: Gelman papers

Gelman and Little, "Poststratification into many categories using hierarchical logistic regression," *Survey Methodology* (1998), pp. 127-135.

Park, Gelman and Bafumi, "Bayesian multilevel analysis with poststratification: state-level estimates from national polls," *Political Analysis* (2004), pp. 375-385.

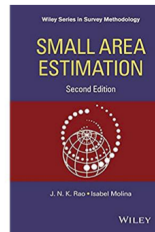
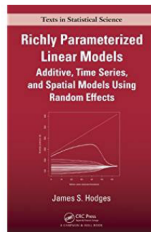
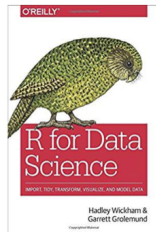
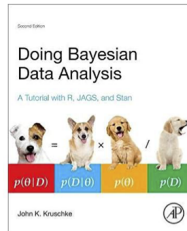
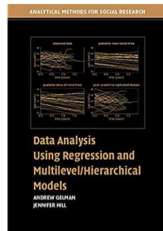
Gelman, "Analysis of variance: Why it is more important than ever" (with discussion), *Annals of Statistics* (2005), pp. 1-53.

Gelman, "Struggles with survey weighting" (with discussion), *Statistical Science* (2007), pp. 153-164.

Gelman and Ghitza, "Deep interactions with MRP: Election turnout and voting patterns among small electoral subgroups," *American Journal of Political Science* (2013), pp. 762-776.

Gelman, Goel, Rivers and Rothschild, "The Mythical Swing Voter," *Quarterly Journal of Political Science* (2016), pp. 103-130.

Further reading



What we have covered

1. Fit a multilevel regression model to survey data, using both individual and group-level covariates. The purpose of this model is **prediction**, not explanation.
2. Avoid **over-fitting** by using hierarchical priors or random effects.
3. The Stein paradox is real and important. When making estimates for many small areas, **shrinkage** or **partial pooling** can substantially improve over “direct” estimates (just using the data from each area separately).
4. The model is used to **impute** the survey variable for non-sampled units to either a public use microdata survey or census counts. This is the **post-stratification** component.
5. Model failure is a real risk, but many successful applications and validations suggest that it is often exaggerated. The estimates are fairly robust to failures of the upper level model if data is not too sparse.

Good luck!

Course evaluations

Please fillout the online course evaluation at

<https://www.surveymonkey.com/r/AAPOR2018ShortCourse>

Thanks!