Multilevel Regression And Poststratification (A Primer)

Joseph T. Ornstein*

Often we'd like to estimate public opinion on some policy issue, but our surveys are unrepresentative in an important way. Perhaps their respondents come from a convenience sample (Wang et al. 2015), or non-response bias skews an otherwise random sample. Or perhaps the data is representative of some larger population (i.e. a country-level random sample), but contains too few observations to make inferences about a subgroup of interest. Even the largest US public opinion surveys do not have enough respondents to make reliable inferences about lower-level political entities like states or municipalities. Conclusions drawn from low frequency observations – even in a large sample survey – can be wildly misleading (Ansolabehere, Luks, and Schaffner 2015).

This presents a challenge for public opinion research: how to take unrepresentative survey data and adjust it so that it is useful for our particular research question. In this chapter, I will demonstrate a method called **multilevel regression and poststratification** (MRP). Using this approach, the researcher first constructs a model of public opinion (multilevel regression) and then reweights the model's predictions based on the observed characteristics of the population of interest (poststratification). In the sections to follow, I will describe this approach in detail, and will accompany this explanation with code in the `R` statistical language.

MRP was first introduced by Gelman and Little (1997), and the subsequent decades it has helped address a diverse set of research questions in political science. These range from generating election forecasts using unrepresentative survey data (Wang et al. 2015) to assessing the responsiveness of state (Lax and Phillips 2012) and local policymakers (Tausanovitch and Warshaw 2014) to their constituents' policy preferences.

In the following sections, I will illustrate how MRP can improve estimates of small area public opinion. Our running example will be drawn from the Cooperative Election Study (Schaffner, Ansolabehere, and Luks 2021), which includes a question asking respondents whether they support a policy that would "decrease the number of police on the street by 10 percent, and increase funding for other public services." Since police reform is a policy issue on which US local governments have a significant amount of autonomy, it would be interesting to know how opinions on this issue vary from place to place without having to conduct a large set of costly local-level surveys.

As we will see, the accuracy of our poststratified estimates depends critically on whether the first-stage model makes good predictions. The best first-stage models are *regularized* (Gelman 2018) to avoid both over- and under-fitting to the survey data. Regularized ensemble models (Ornstein 2020) with group-level predictors tend to produce the best estimates, especially when trained on large datasets.

1 Running Example

To demonstrate how MRP works, we'll consider an example where we know the true population-level estimands, and can explore how various refinements to the method can improve predictive accuracy. This approach mirrors Buttice and Highton (2013), who use disaggregated ressponses from large-scale US survey of voters (Schaffner, Ansolabehere, and Luks 2021) as the target.[1] The Cooperative Election Study data is available here, and we'll be using a tidied version of the dataset created by the `R/cleanup-ces-2020.R` script.

*Department of Political Science, University of Georgia

[1]Throughout, I will use R functions from the "tidyverse" to make the code more human-readable. All code will be available for download on a public repository.

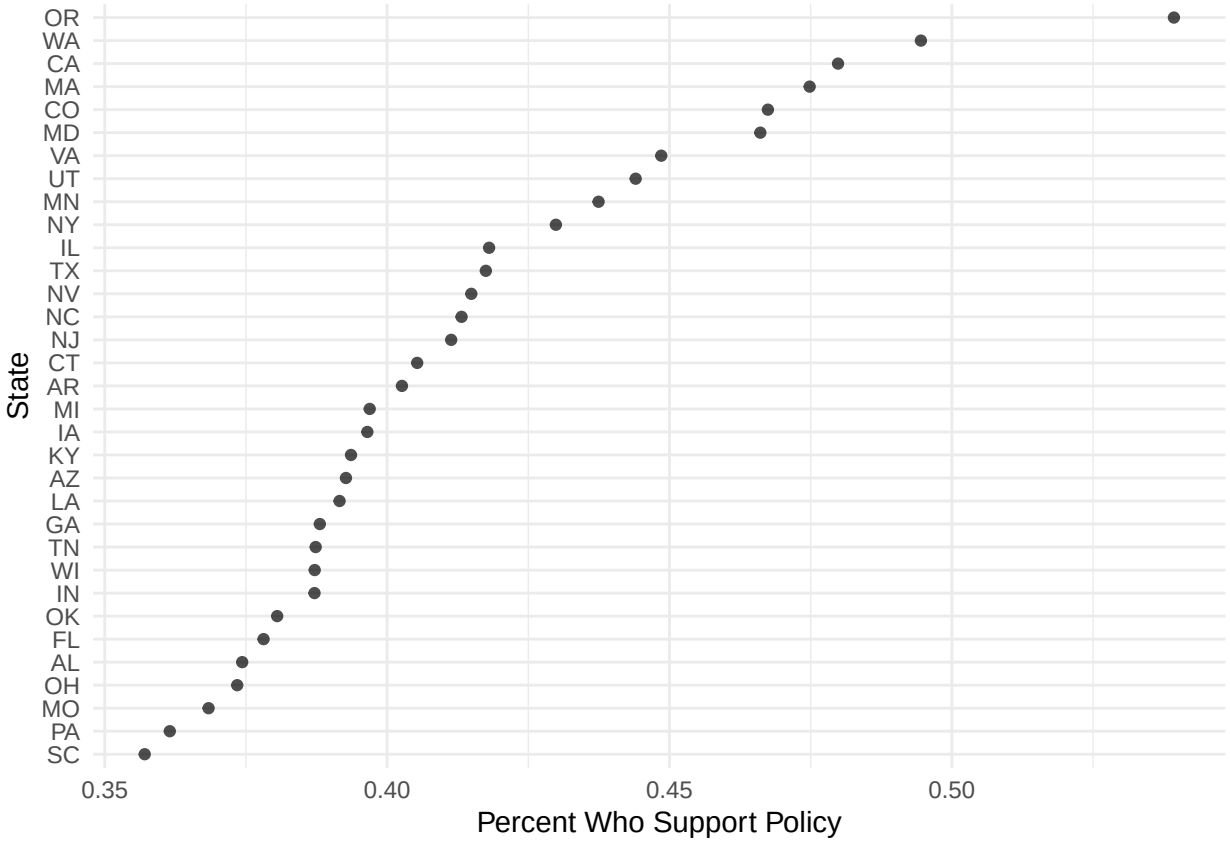
```
library(tidyverse)
library(ggrepel)

load('data/CES-2020.RData')
```

This tidied version of the data only includes the 33 states with at least 500 respondents.

1.1 The Truth

First, let's plot the percent of CES respondents who supported "defunding" the police[2] by state.

```
truth <- ces %>%
  group_by(abb) %>%
  summarize(truth = mean(defund_police))

# plot
truth %>%
  # reorder abb so the chart is organized by percent who support
  mutate(abb = fct_reorder(abb, truth)) %>%
  ggplot(mapping = aes(x=truth, y=abb)) +
  geom_point(alpha = 0.7) +
  labs(x = 'Percent Who Support Policy', y = 'State') +
  theme_minimal()
```



Note that these values likely overstate the percent of the total population that support such a policy, as

[2]Obviously that phrase means different things to different people. In this case, we'll stick with the CES proposed policy of reducing police staffing by 10% and diverting those expenditures to other priorities.

self-identified Democrats are overrepresented in the CES sample. But they will nevertheless serve as the "truth" that we will try to estimate with MRP.

1.2 Draw a Sample

Now suppose that we did not have access to the entire CES dataset, but only to a random sample of 1,000 respondents. How good of a job can we do at estimating those state-level means?

```
sample_data <- ces %>%
  slice_sample(n = 1000)
```

```
sample_summary <- sample_data %>%
  group_by(abb) %>%
  summarize(estimate = mean(defund_police),
            num = n())

sample_summary
```

```
## # A tibble: 33 x 3
##    abb   estimate   num
##    <chr>    <dbl> <int>
##  1 AL       0.55     20
##  2 AR       0         4
##  3 AZ       0.438    16
##  4 CA       0.435    85
##  5 CO       0.478    23
##  6 CT       0.375     8
##  7 FL       0.402    87
##  8 GA       0.346    26
##  9 IA       0.308    13
## 10 IL       0.28     50
## # ... with 23 more rows
```

In a with only 1,000 respondents, there are several states with very few (or no) respondents. Notice, for example, that this sample includes only four respondents from Arkansas, of whom zero support reducing police budgets. So simply disaggregating and taking sample means will not yield good estimates, as you can see by comparing the sample means against the truth.

```
# a function to plot the state-level estimates against the truth
compare_to_truth <- function(estimates, truth){

  d <- left_join(estimates, truth, by = 'abb')

  ggplot(data = d,
         mapping = aes(x=estimate,
                       y=truth,
                       label=abb)) +
  geom_point(alpha = 0.5) +
  geom_text_repel() +
  theme_minimal() +
  geom_abline(intercept = 0, slope = 1, linetype = 'dashed') +
  labs(x = 'Estimate',
       y = 'Truth',
       caption = paste0('Correlation = ', round(cor(d$estimate, d$truth), 2)))
}
```

```
compare_to_truth(sample_summary, truth)
```

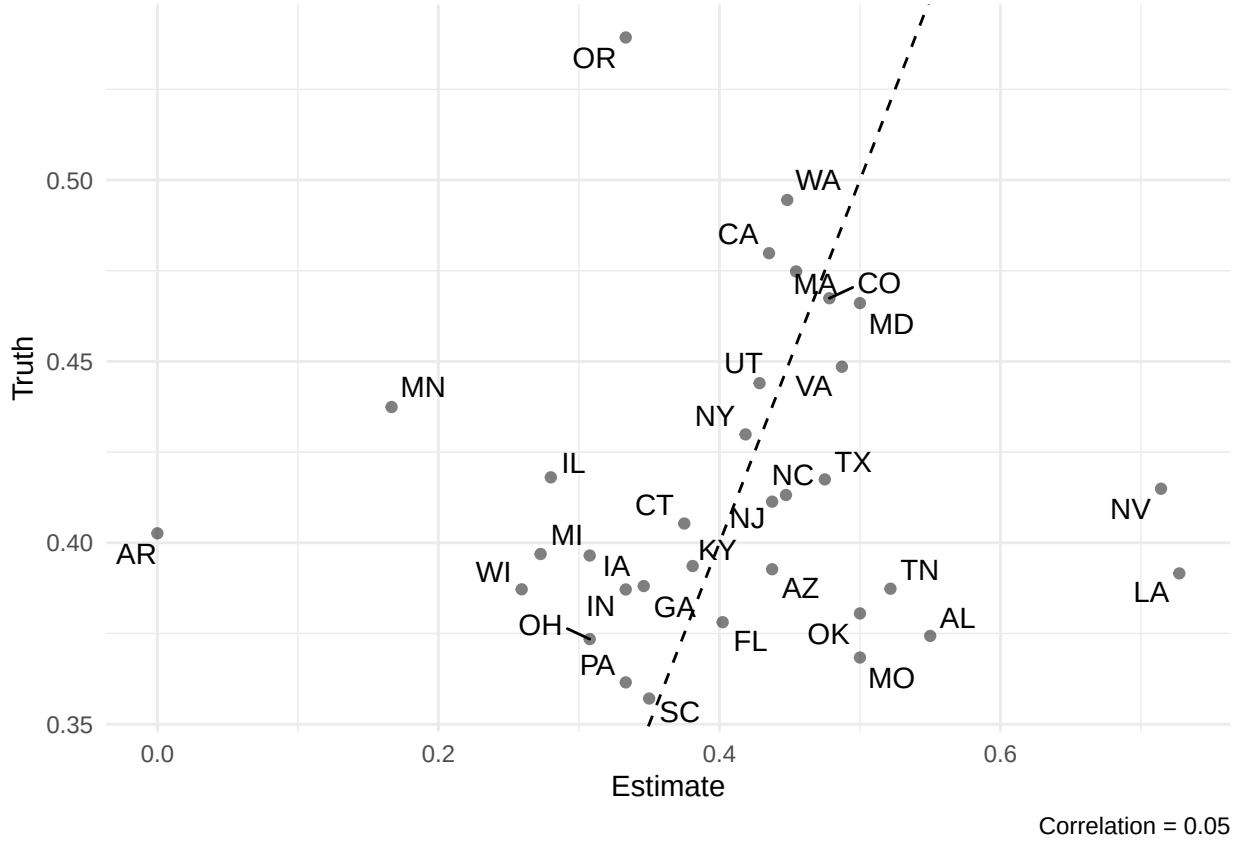


Figure 1: Esimates from disaggregated sample data

This is clearly a poor estimate of state-level public opinion. The four respondents from Arksansas simply do not give us enough information to adequately measure public opinion in Arkansas. But one of the key insights behind MRP is that the respondents from Arkansas are not the only respondents who can give us information about Arkansas! There are other respondents in, for example, Missouri, that are similar to Arkansas residents on their observed characteristics. If we can determine the characteristics that predict support for police reform using the entire survey, then we can use those predictions – combined with demographic information about Arkansans – to generate better estimates. The trick, in essence, is that our estimate for Arkansas will be borrowing information from similar respondents in other states.

MRP proceeds in two steps.

1.3 Step 1: Multilevel Regression

First, we fit a model of our outcome, using observed characteristics of the survey respondents as predictors. As a first pass, let's fit a simple logistic model of including only four demographic predictors: gender, education, race, and age.

```
model <- glm(defund_police ~
               gender + educ + race + age,
             data = sample_data,
             family = 'binomial')
```

1.4 Step 2: Poststratification

```
psframe <- ces %>%
  count(abb, gender, educ, race, age)

head(psframe)
```

```
## # A tibble: 6 x 6
##   abb   gender educ   race    age     n
##   <chr> <chr>  <chr>  <chr> <dbl> <int>
## 1 AL    Female 2-year Black    26     1
## 2 AL    Female 2-year Black    27     2
## 3 AL    Female 2-year Black    29     1
## 4 AL    Female 2-year Black    31     1
## 5 AL    Female 2-year Black    34     2
## 6 AL    Female 2-year Black    35     2
```

Append predicted probabilities to the poststratification frame.

```
psframe <- psframe %>%
  mutate(predicted_probability = predict(model, psframe, type = 'response'))
```

Poststratified estimates are the population-weighted predictions

```
poststratified_estimates <- psframe %>%
  group_by(abb) %>%
  summarize(estimate = weighted.mean(predicted_probability, n))
```

Merge and compare:

```
compare_to_truth(poststratified_estimates, truth)
```

This highlights one of the dangers of producing MRP estimates from an underspecified model. When the first-stage model is underfit, poststratified estimates tend to overshrink towards the global mean.

This compression means we should be wary of MRP studies that show policy outcomes "leapfrogging" estimated public opinion (Simonovits and Payson 2020). It could be that policymakers are more extreme than their constituents, or that MRP produces estimates of constituency preferences that are too moderate.

1.5 The Other Extreme: Overfitting

To illustrate the other extreme, let's estimate a model with a separate intercept term for each state – a "fixed effects" model. Because our sample contains several states with very few observations, these state-specific intercepts will likely overfit to sampling variability.

```
# fit the model
model2 <- glm(defund_police ~
              gender + educ + race + age +
                abb,
            data = sample_data,
            family = 'binomial')

# make predictions
psframe <- psframe %>%
  mutate(predicted_probability = predict(model2, psframe, type = 'response'))

# poststratify
poststratified_estimates <- psframe %>%
```

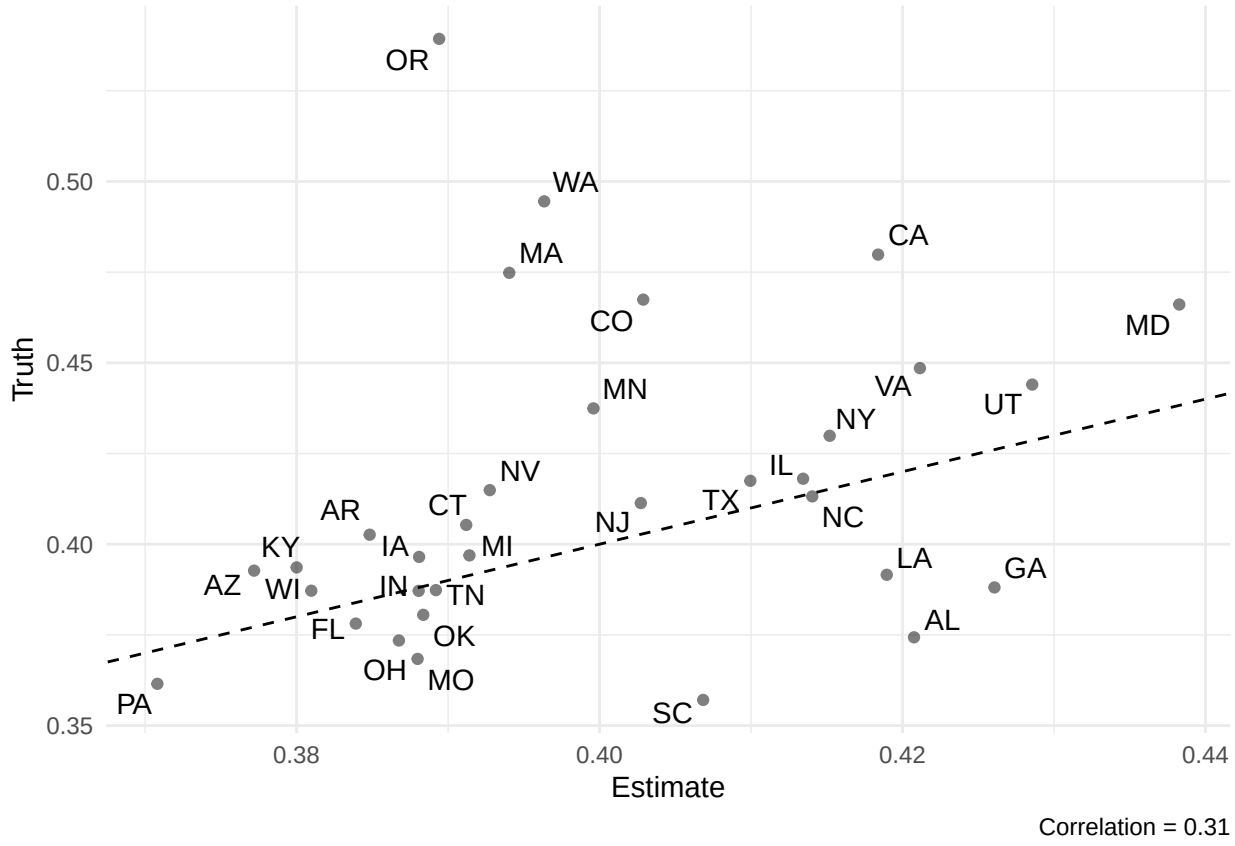


Figure 2: Underfit MRP estimates from complete pooling model

```
  group_by(abb) %>%
  summarize(estimate = weighted.mean(predicted_probability, n))


compare_to_truth(poststratified_estimates, truth)
```

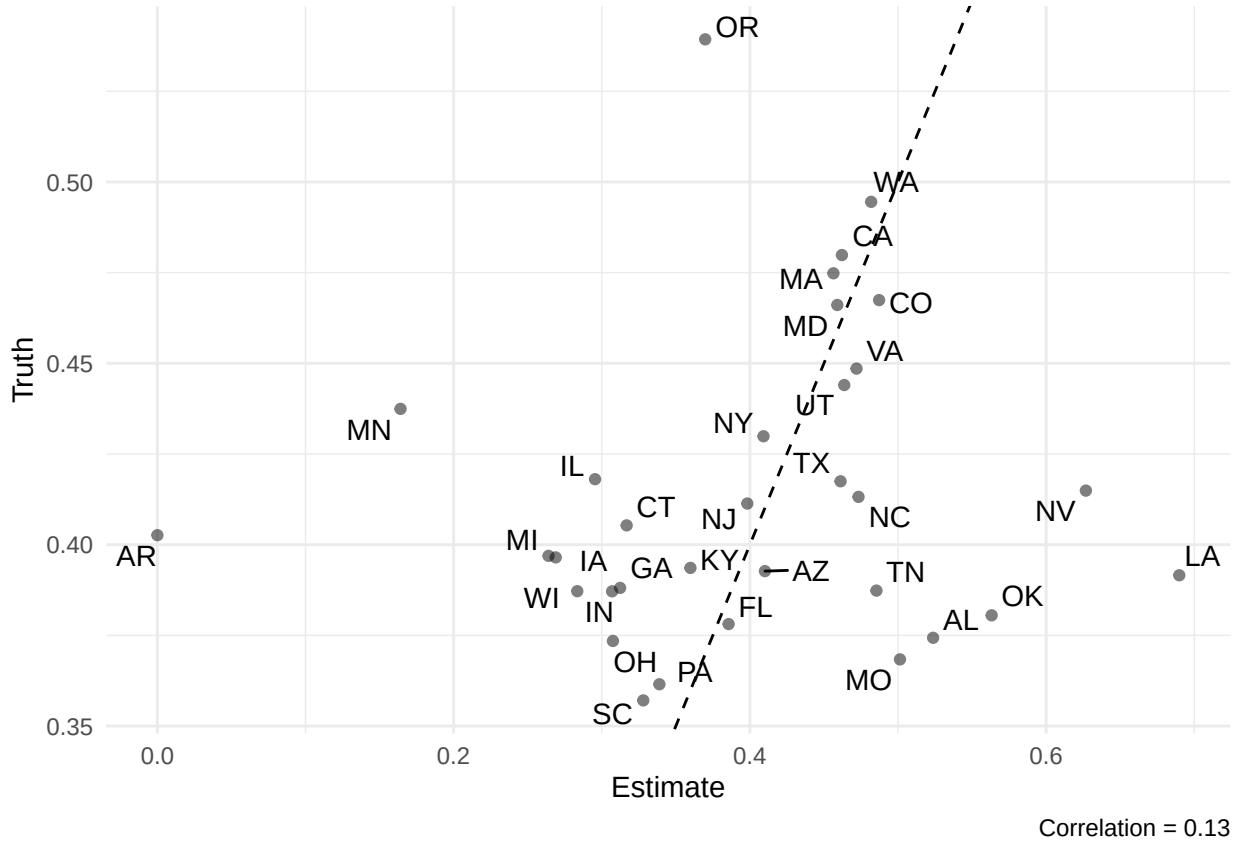


Figure 3: Overfit MRP estimates from fixed effects model

Compare this to Figure 1 – these estimates are similarly overfit. Iowa is predicted to have roughly 100% support due to an idiosyncratic sample, while Maryland has the opposite problem.

1.6 The Sweet Spot: Partial Pooling

Gelman and Little (1997) solution: multilevel models that partially pool across regions. (Explanation of partial pooling goes. . . here)

```
library(lme4)

model3 <- glmer(defund_police ~  gender + educ + race + age +
                (1|abb),
                 data = sample_data,
                 family = 'binomial')

# make predictions
psframe <- psframe %>%
  mutate(predicted_probability = predict(model3, psframe, type = 'response'))
```

```r
# poststratify
poststratified_estimates <- psframe %>%
  group_by(abb) %>%
  summarize(estimate = weighted.mean(predicted_probability, n))


compare_to_truth(poststratified_estimates, truth)
```

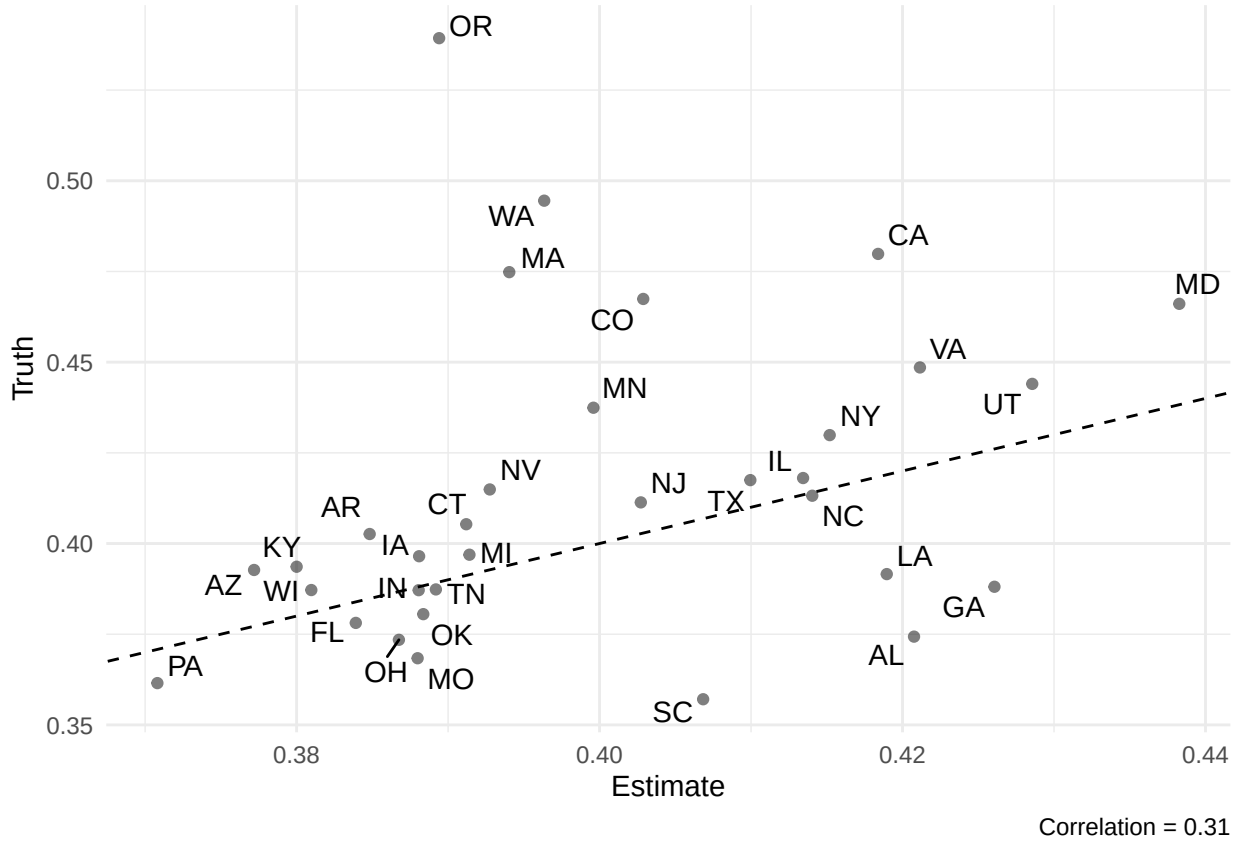


Figure 4: MRP estimates from model with partial pooling

TODO: Well that's not the approach we should take, then. Partial pooling isn't magic. It just undoes the damage that fixed effects does. The magic is in good geographic predictors.

1.7 The Importance of Group-Level Covariates

- The Democratic presidential two party vote share in 2020
- The percent of the state's residents that live in urban areas
- The state's 2020 homicide rate (homicides per 10,000 residents)
-

1.8 Stacking

```r
library(SuperLearner)

# fit Super Learner
SL.library <- c("SL.ranger", "SL.gam", "SL.xgboost", "SL.glm")

X <- sample_data %>%
  select(gender, educ, race, age, abb)

newX <- psframe %>%
  select(gender, educ, race, age, abb)

sl.out <- SuperLearner(Y = sample_data$defund_police,
                       X = X,
                       newX = newX,
                       family = binomial(),
                       SL.library = SL.library, verbose = FALSE)
```

```
## Error in signif(as.vector(x), 6) :
##   non-numeric argument to mathematical function
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in signif(as.vector(x), 6) :
##   non-numeric argument to mathematical function
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in signif(as.vector(x), 6) :
##   non-numeric argument to mathematical function
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in signif(as.vector(x), 6) :
##   non-numeric argument to mathematical function
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in signif(as.vector(x), 6) :
##   non-numeric argument to mathematical function
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in signif(as.vector(x), 6) :
##   non-numeric argument to mathematical function
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in signif(as.vector(x), 6) :
##   non-numeric argument to mathematical function
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in signif(as.vector(x), 6) :
##   non-numeric argument to mathematical function
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in signif(as.vector(x), 6) :
##   non-numeric argument to mathematical function
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
```

```
## Error in signif(as.vector(x), 6) :
##   non-numeric argument to mathematical function
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in signif(as.vector(x), 6) :
##   non-numeric argument to mathematical function
```

```
#sl.out$SL.predict
sl.out$coef
```

```
##  SL.ranger_All     SL.gam_All SL.xgboost_All     SL.glm_All
##      0.4749948      0.0000000      0.0000000      0.5250052
```

```
# make predictions
psframe <- psframe %>%
  mutate(predicted_probability = sl.out$SL.predict)

# poststratify
poststratified_estimates <- psframe %>%
  group_by(abb) %>%
  summarize(estimate = weighted.mean(predicted_probability, n))

compare_to_truth(poststratified_estimates, truth)
```

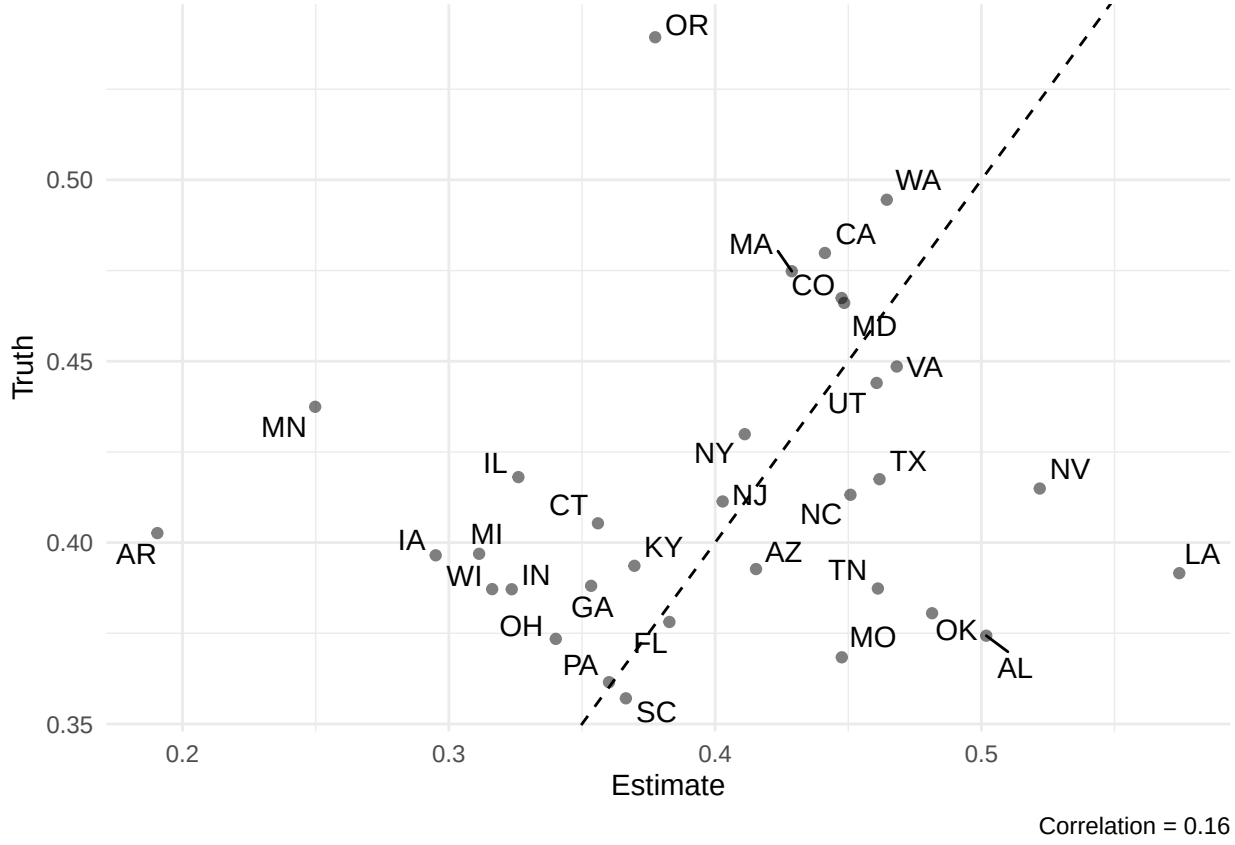


Figure 5: Estimates from an ensemble first-stage model

And that's just "out-of-the-box!" What if we were more careful about it?

This reflects the gains that come from modeling "deep interactions" in the predictors of public opinion (Ghitza and Gelman 2013). If, for example, income better predicts partisanship in some states but not in others (Gelman et al. 2007), then a model that captures that moderating effect will produce better poststratified estimates than one that does not. Machine learning techniques like random forest (Breiman 2001) are especially useful for detecting and modeling such deep interactions, and stacked regression and poststratification (SRP) tends to outperform MRP in simulations, particularly for training data with large sample size (Ornstein 2020).

1.9 Synthetic Poststratification

Suppose that we did not have access to the entire joint distribution of individual-level covariates. Leemann and Wasserfallen (2017) suggest an extension of Mr. P, which they (delightfully) dub Multilevel Regression and Synthetic Poststratification (Mrs. P). Lacking the full joint distribution of covariates for poststratification, one can instead create a *synthetic* poststratification frame by assuming that additional covariates are statistically independent of one another. So long as your first stage model is linear-additive, this approach yields the same predictions as if you knew the true joint distribution![3] And even if the first-stage model is not linear-additive, simulations suggest that the improved performance from additional predictors tends to overcome the error introduced by synthetic poststratification.

To create a synthetic poststratification frame, convert frequencies to probabilities and multiply. For example, suppose we only had the joint distribution for gender, race, and age, and wanted to create a synthetic poststratification including education.

Add:

- How important is religion to the respondent?
- Whether the respondent lives in an urban, rural, or suburban area
- Whether the respondent or a member of the respondent's family is a military veteran
- Whether the respondent owns or rents their home
- Is the respondent the parent or guardian of a child under the age of 18?

These variables may be useful predictors of opinion about police policy and the first-stage model could be improved by including them. But there is no dataset (that I know of) that would allow us to compute a state-level joint probability distribution over every one of them. Instead, we would typically only know the marginal distributions of each covariate (e.g. the percent of a state's residents that are military households, or the percent that live in urban areas). (TODO: Flip this up top to motivate synthetic poststratification)

```
# poststratification frame with 3 variables
psframe3 <- ces %>%
  count(abb, gender, race, age) %>%
  group_by(abb) %>%
  mutate(prob = n / sum(n))

head(psframe3)
```

```
## # A tibble: 6 x 6
## # Groups:   abb [1]
##   abb   gender race    age     n    prob
##   <chr> <chr>  <chr> <dbl> <int>   <dbl>
## 1 AL    Female Asian    24     1 0.00106
## 2 AL    Female Asian    27     1 0.00106
## 3 AL    Female Asian    29     1 0.00106
```

[3] See Ornstein (2020) appendix A for mathematical proof.

```
## 4 AL    Female Asian    30     1 0.00106
## 5 AL    Female Asian    34     2 0.00212
## 6 AL    Female Black    18     1 0.00106
```

```
# distribution of education variable by state
psframe_educ <- ces %>%
  count(abb, educ) %>%
  group_by(abb) %>%
  mutate(prob2 = n / sum(n))

head(psframe_educ)
```

```
## # A tibble: 6 x 4
## # Groups:   abb [1]
##   abb   educ                    n  prob2
##   <chr> <chr>               <int>  <dbl>
## 1 AL    2-year                122 0.129
## 2 AL    4-year                179 0.190
## 3 AL    High school graduate  287 0.304
## 4 AL    No HS                  49 0.0520
## 5 AL    Post-grad             119 0.126
## 6 AL    Some college          187 0.198
```

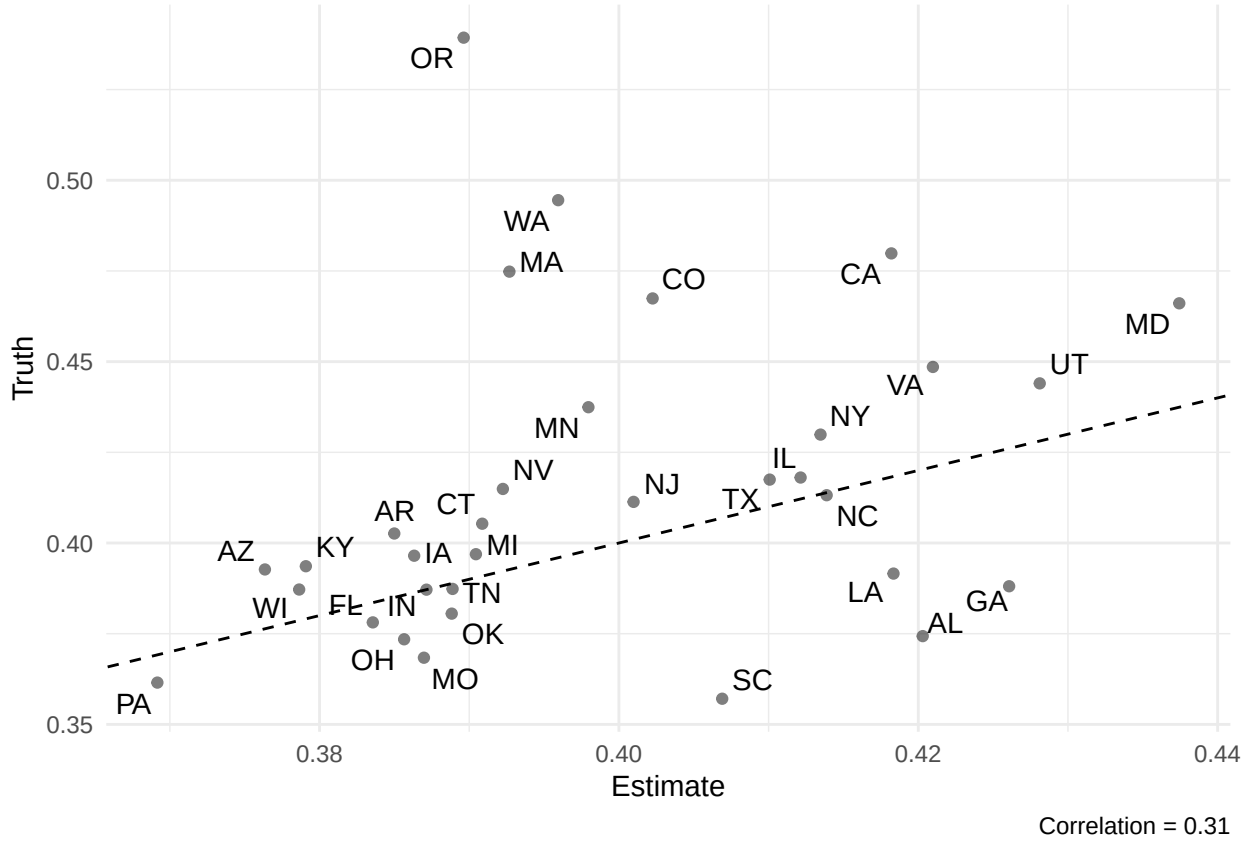
```
synthetic_psframe <- left_join(psframe3, psframe_educ,
                               by = 'abb') %>%
  mutate(prob = prob * prob2)

head(synthetic_psframe)
```

```
## # A tibble: 6 x 9
## # Groups:   abb [1]
##   abb   gender race    age   n.x      prob educ                  n.y  prob2
##   <chr> <chr>  <chr> <dbl> <int>     <dbl> <chr>               <int>  <dbl>
## 1 AL    Female Asian    24     1 0.000137  2-year                122 0.129
## 2 AL    Female Asian    24     1 0.000201  4-year                179 0.190
## 3 AL    Female Asian    24     1 0.000323  High school graduate  287 0.304
## 4 AL    Female Asian    24     1 0.0000551 No HS                  49 0.0520
## 5 AL    Female Asian    24     1 0.000134  Post-grad             119 0.126
## 6 AL    Female Asian    24     1 0.000210  Some college          187 0.198
```

The SRP package contains a convenience function for this operation (see the vignette for more information).

Then poststratify as normal.

```
# make predictions
synthetic_psframe$predicted_probability <- predict(model3, synthetic_psframe, type = 'response')

# poststratify
poststratified_estimates <- synthetic_psframe %>%
  group_by(abb) %>%
  summarize(estimate = weighted.mean(predicted_probability, prob))


compare_to_truth(poststratified_estimates, truth)
```



Note that the performance is slightly worse than when we knew the true joint distribution. But is it worse than omitting education entirely?

```
model4 <- glmer(defund_police ~  gender + race + age +
                (1|abb),
                 data = sample_data,
                 family = 'binomial')


# make predictions
psframe3$predicted_probability <- predict(model4, psframe3, type = 'response')

# poststratify
poststratified_estimates <- psframe3 %>%
  group_by(abb) %>%
  summarize(estimate = weighted.mean(predicted_probability, prob))


compare_to_truth(poststratified_estimates, truth)
```

1.10 Best Performing

Supposing we had access to the true joint distribution and fit an ensemble first-stage model...

```
psframe <- ces %>%
  count(abb, gender, race, age, educ,
        pew_religimp, homeowner, urban,
```

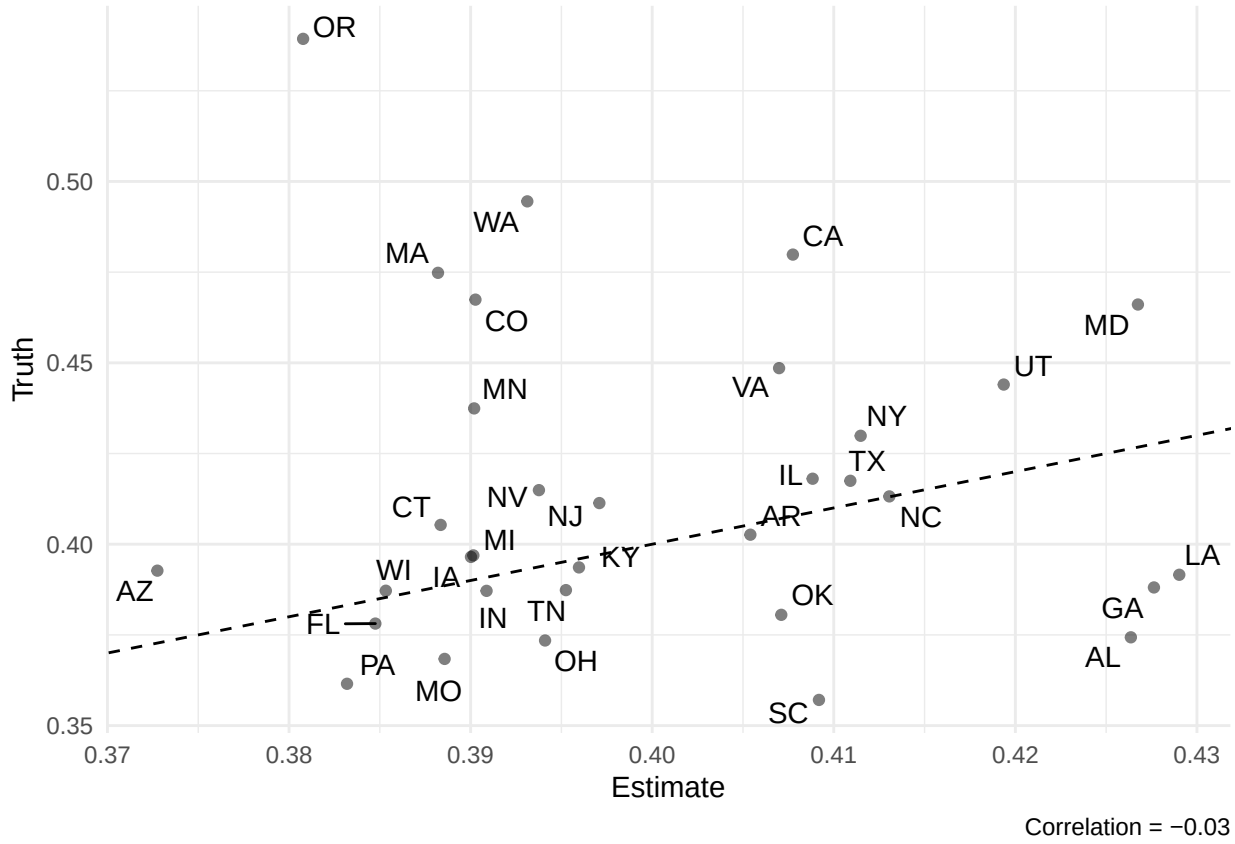
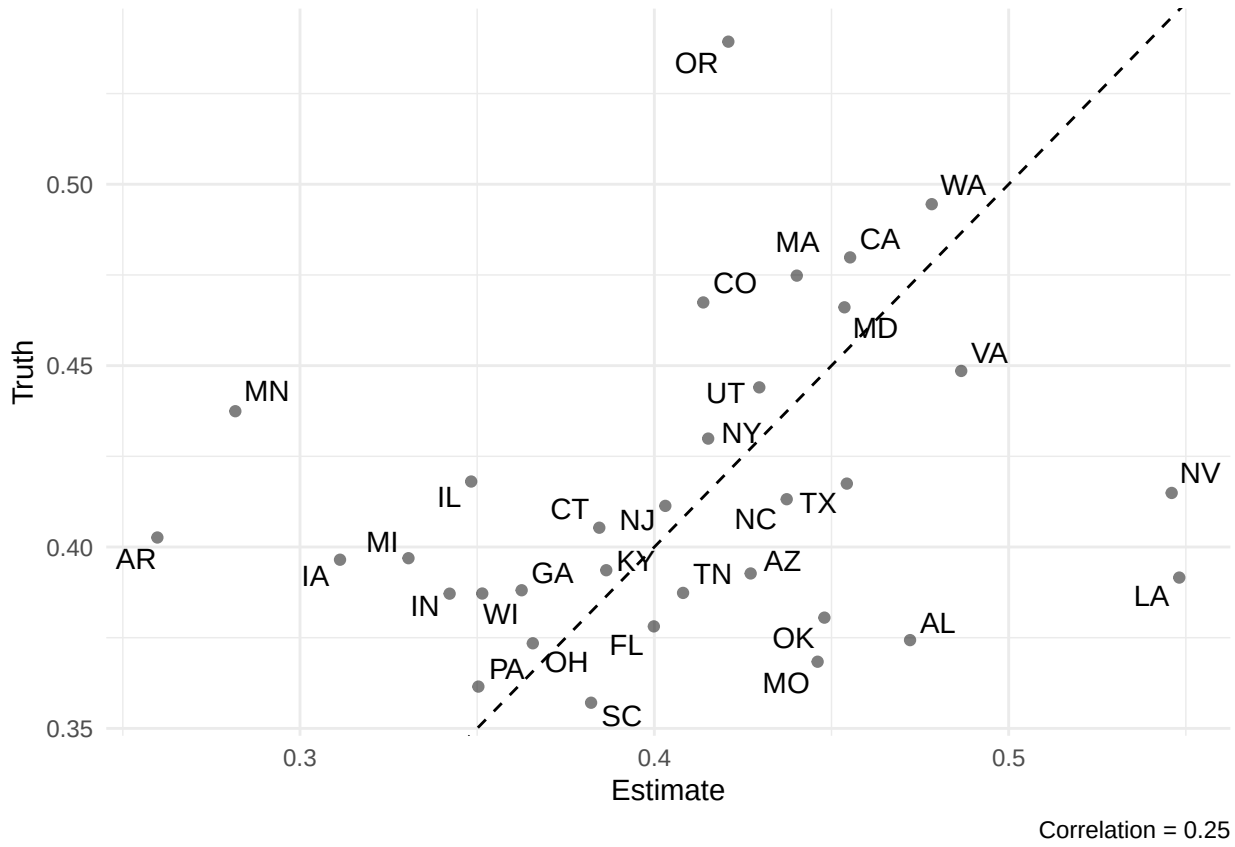


Figure 6: Poststratified estimates, omitting education

```
        parent, military_household,
        biden_vote_share, homicide_rate)

library(SuperLearner)

# fit Super Learner
SL.library <- c("SL.ranger", "SL.glm")

X <- sample_data %>%
  select(abb, gender, race, age, educ,
        pew_religimp, homeowner, urban,
        parent, military_household,
        biden_vote_share, homicide_rate)

newX <- psframe %>%
  select(abb, gender, race, age, educ,
        pew_religimp, homeowner, urban,
        parent, military_household,
        biden_vote_share, homicide_rate)

sl.out <- SuperLearner(Y = sample_data$defund_police,
                       X = X,
                       newX = newX,
                       family = binomial(),
                       SL.library = SL.library,
                       verbose = FALSE)
#sl.out$SL.predict
sl.out$coef
```

```
## SL.ranger_All    SL.glm_All
##     0.6864439     0.3135561
```

```
# make predictions
psframe <- psframe %>%
  mutate(predicted_probability = sl.out$SL.predict)

# poststratify
poststratified_estimates <- psframe %>%
  group_by(abb) %>%
  summarize(estimate = weighted.mean(predicted_probability, n))

compare_to_truth(poststratified_estimates, truth)
```



2 Conclusion

In the code above I have emphasized "do-it-yourself" approaches to MRP – fitting a model, building a poststratification frame, and producing estimates separately. But there are a number of `R` packages available with useful functions to help ease the process. In particular, I would encourage curious readers to explore the `autoMrP` package (Broniecki, Leemann, and Wüest 2022), which implements the ensemble modeling approach described above, and performs quite well in simulations when compared to existing packages.

References

Ansolabehere, Stephen, Samantha Luks, and Brian F. Schaffner. 2015. "The Perils of Cherry Picking Low Frequency Events in Large Sample Surveys." *Electoral Studies* 40 (December): 409–10. https://doi.org/10.1016/j.electstud.2015.07.002.

Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32. https://doi.org/10.1023/A:1010933404324.

Broniecki, Philipp, Lucas Leemann, and Reto Wüest. 2022. "Improved Multilevel Regression with Poststratification Through Machine Learning (autoMrP)." *The Journal of Politics* 84 (1). https://doi.org/10.1086/714777.

Buttice, Matthew K., and Benjamin Highton. 2013. "How Does Multilevel Regression and Poststratification Perform with Conventional National Surveys?" *Political Analysis* 21 (4): 449–67. https://doi.org/10.1093/pan/mpt017.

Gelman, Andrew. 2018. "Regularized Prediction and Poststratification (the Generalization of Mister p)." *Statistical Modeling, Causal Inference, and Social Science (Blog)* May 19

(https://statmodeling.stat.columbia.edu/2018/05/19/).

Gelman, Andrew, and Thomas C Little. 1997. "Poststratification into Many Categories Using Hierachical Logistic Regression." *Survey Methodology* 23 (2): 127–35.

Gelman, Andrew, Boris Shor, Joseph Bafumi, and David Park. 2007. "Rich State, Poor State, Red State, Blue State: What's the Matter with Connecticut?" *Quarterly Journal of Political Science* 2 (June 2006): 345–67. https://doi.org/10.1561/100.00006026.

Ghitza, Yair, and Andrew Gelman. 2013. "Deep Interactions with MRP: Election Turnout and Voting Patterns Among Small Electoral Subgroups." *American Journal of Political Science* 57 (3): 762–76. https://doi.org/10.1111/ajps.12004.

Lax, Jeffrey R., and Justin H. Phillips. 2012. "The Democratic Deficit in the States." *American Journal of Political Science* 56 (1): 148–66. https://doi.org/10.1111/j.1540-5907.2011.

Leemann, Lucas, and Fabio Wasserfallen. 2017. "Extending the Use and Prediction Precision of Subnational Public Opinion Estimation." *American Journal of Political Science* 61 (4): 1003–22.

Ornstein, Joseph T. 2020. "Stacked Regression and Poststratification." *Political Analysis* 28 (2): 293–301. https://doi.org/10.1017/pan.2019.43.

Schaffner, Brian, Stephen Ansolabehere, and Sam Luks. 2021. "Cooperative Election Study Common Content, 2020." Edited by YouGov and Add your team name(s) here. https://doi.org/10.7910/DVN/E9N6PH.

Simonovits, Gabor, and Julia Payson. 2020. "Locally Controlled Minimum Wages Are No Closer to Public Preferences." *Working Paper*, 21.

Tausanovitch, Chris, and Christopher Warshaw. 2014. "Representation in Municipal Government." *The American Political Science Review* 108 (03): 605–41. https://doi.org/10.1017/S0003055414000318.

Wang, Wei, David Rothschild, Sharad Goel, and Andrew Gelman. 2015. "Forecasting Elections with Non-Representative Polls." *International Journal of Forecasting* 31 (3): 980–91. https://doi.org/10.1016/j.ijforecast.2014.06.001.