Multilevel Regression And Poststratification (A Primer)

Joseph T. Ornstein*

October 26, 2021

A few takeaways from this chapter:

- Underfitting and overfitting are both bad. The best first-stage models are regularized (as Gelman (2018) would advocate, regularized prediction and poststratification).
- Regularized ensemble models (Ornstein 2020) with unit-level predictors consistently produce the best estimates.

1 Running Example

To demonstrate how MRP works, we'll consider an example where we know the "real" answer, and can explore how various refinements to the model improve predictive accuracy. The approach we'll use mirrors that in Buttice and Highton (2013), taking responses from a large scale US survey of voters (schaffner ref).[1]

```
library(tidyverse)
library(ggrepel)

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

The data is available here, and we'll be using a tidied up version of the dataset created by `R/cleanup-ces-2020.R`. This tidied version of the data only includes the 33 states with at least 500 respondents.

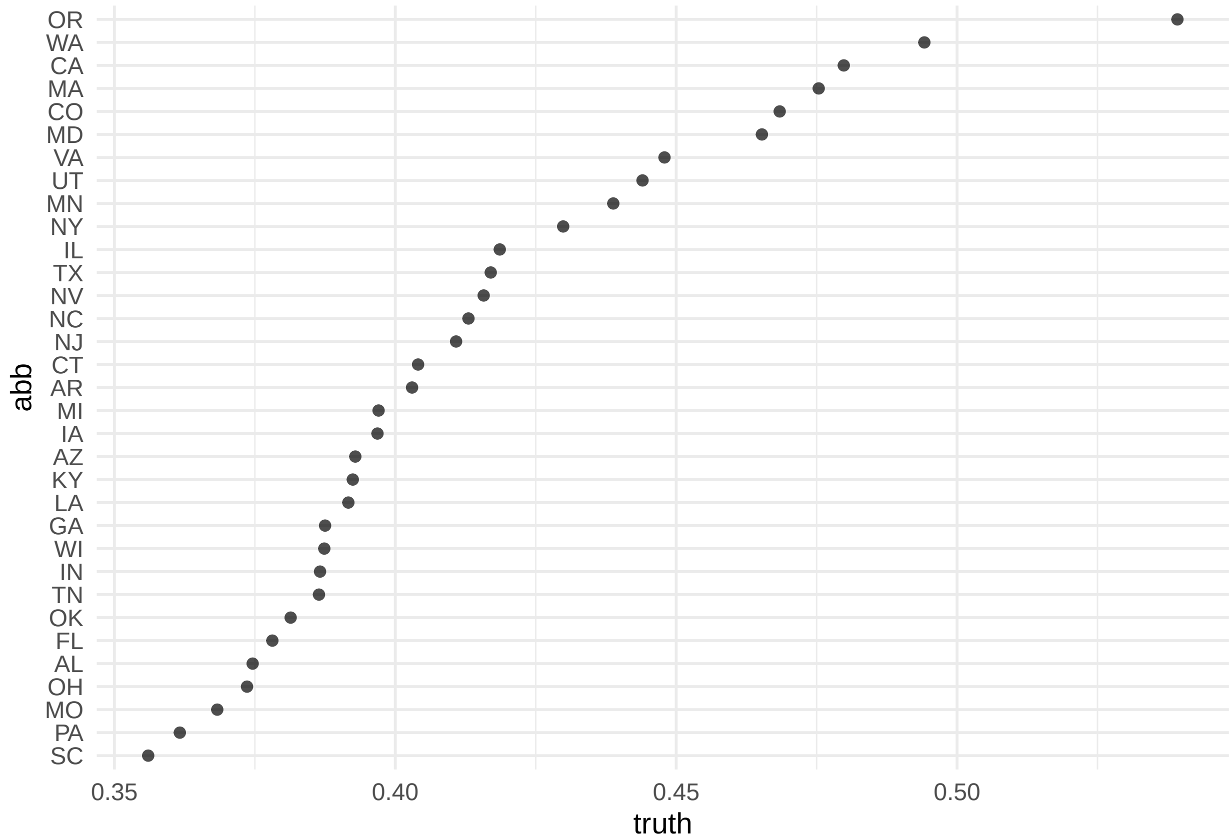
1.1 The Truth

```
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) +
  theme_minimal()
```

*Department of Political Science, University of Georgia

[1]Throughout, I will use R functions from the "tidyverse" to make the code more human-readable.



Note what I mean by the "truth" here is the true percentage of CES respondents who supported the assault rifle ban. That's our target. This overstates the percent of the total population that support such a ban, since the CES sample is not a simple random sample.

1.2 Draw a Sample

Step 1: draw a sample.

```
sample_data <- ces %>%
  slice_sample(n = 500)
```

```
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       0.429     7
##  2 AR       0.429     7
##  3 AZ       0.5      18
##  4 CA       0.535    43
##  5 CO       0.429     7
##  6 CT       0.25      4
```

```
## 7 FL       0.395    38
## 8 GA       0.467    30
## 9 IA       0         2
## 10 IL      0.4      20
## # ... with 23 more rows
```

One approach: disaggregation.

For readers who are less familiar with American politics, rest assured that this is an unrepresentative draw from the state of Iowa. So simply disaggregating and taking sample means will not yield good estimates, as you can see by comparing the percent of respondents from the sample who supported the ban against the percent of CES respondents.

```
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))) +
    scale_x_continuous(limits = c(0,1)) +
    scale_y_continuous(limits = c(0,1))
}

compare_to_truth(sample_summary, truth)
```

1.3 Multilevel Regression

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

1.4 Poststratification

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

head(psframe)
```

```
## # A tibble: 6 x 6
##   abb   gender educ  race    age     n
##   <chr> <chr>  <fct> <chr> <dbl> <int>
## 1 AL    Female No HS Black    28     1
## 2 AL    Female No HS Black    29     1
## 3 AL    Female No HS Black    34     1
```

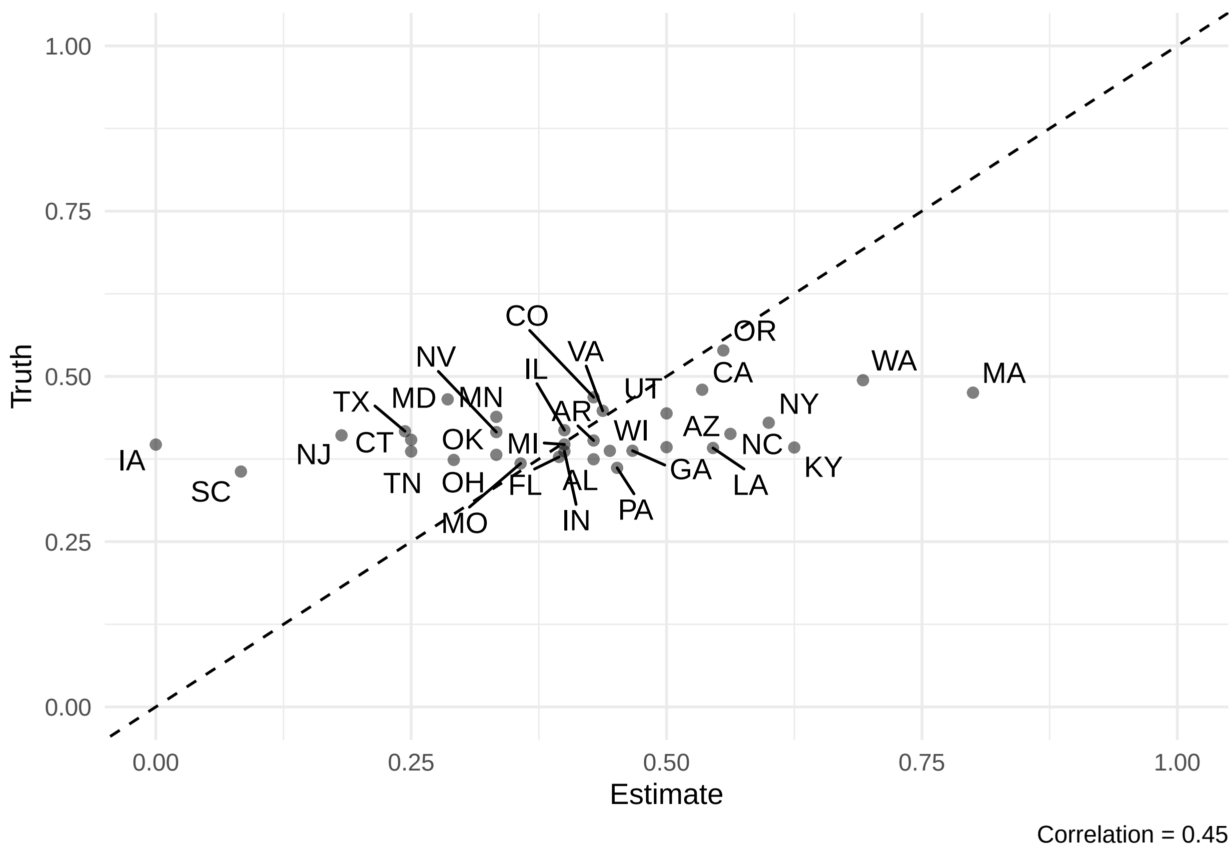


Figure 1: Esimates from disaggregated sample data

```
## 4 AL    Female No HS Black    54     1
## 5 AL    Female No HS Black    64     1
## 6 AL    Female No HS Other    36     1
```

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)
```

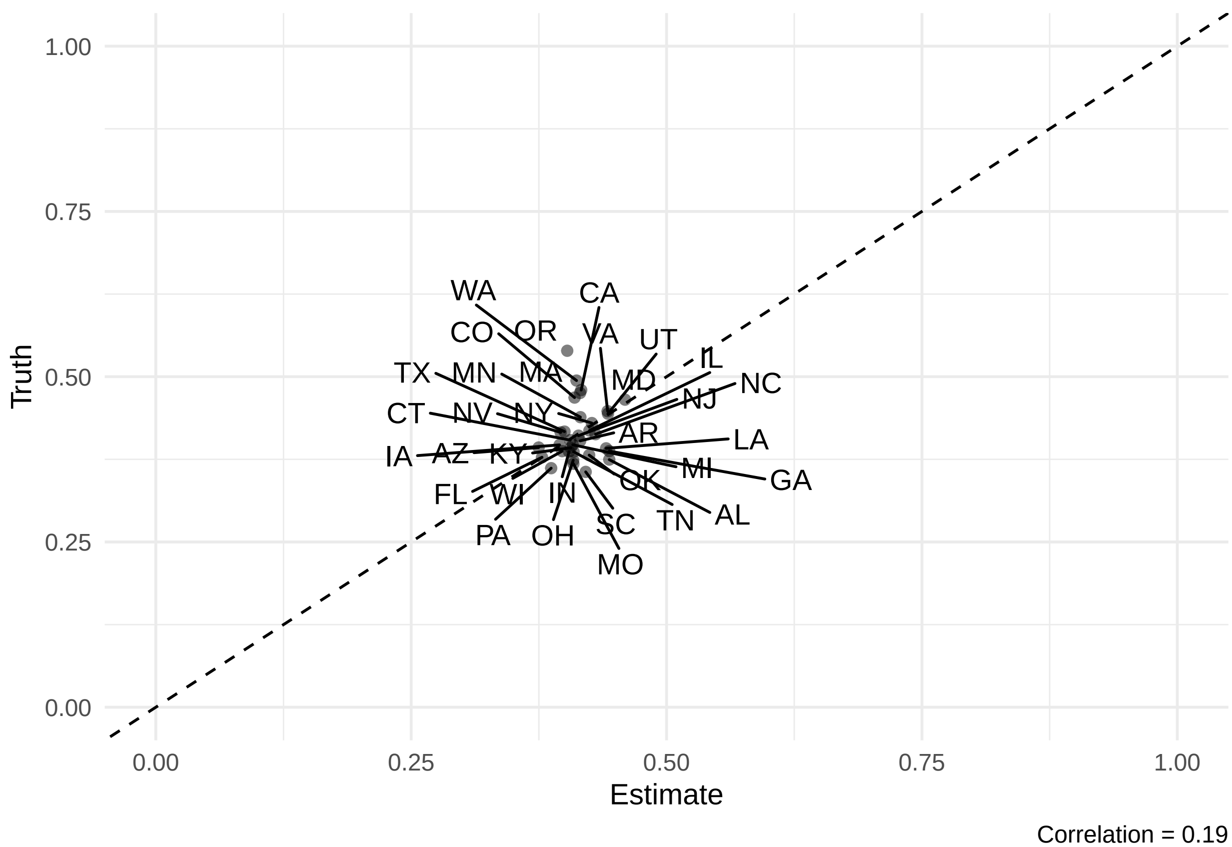


Figure 2: Underfit MRP estimates from complete pooling model

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 collapse towards the global mean [CITATION NEEDED + BETTER TERMINOLOGY].

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


compare_to_truth(poststratified_estimates, truth)
```

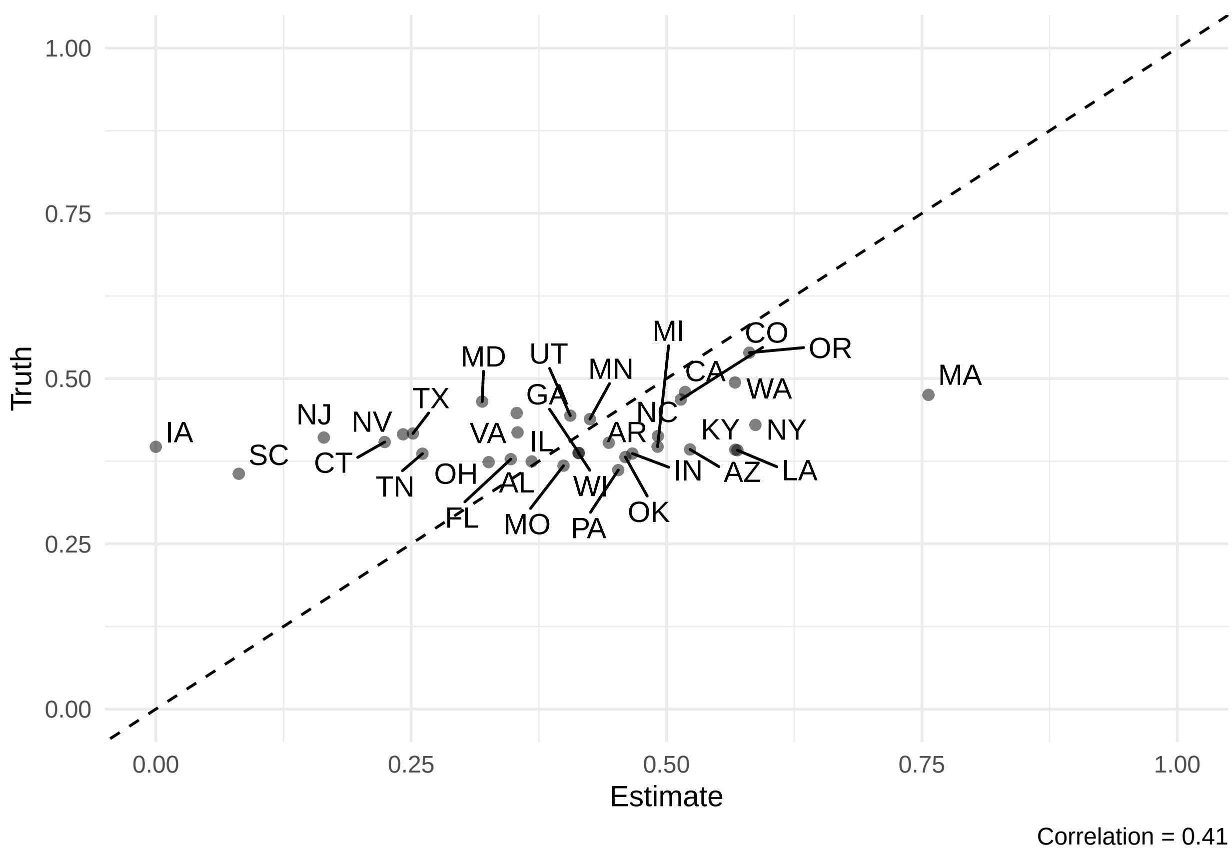


Figure 3: Overfit MRP estimates from fixed effects model

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'))

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


compare_to_truth(poststratified_estimates, truth)
```

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 Stacking

```
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 s(educ, 2) :
##   unordered factors cannot be used as smoothing variables
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in s(educ, 2) :
##   unordered factors cannot be used as smoothing variables
```

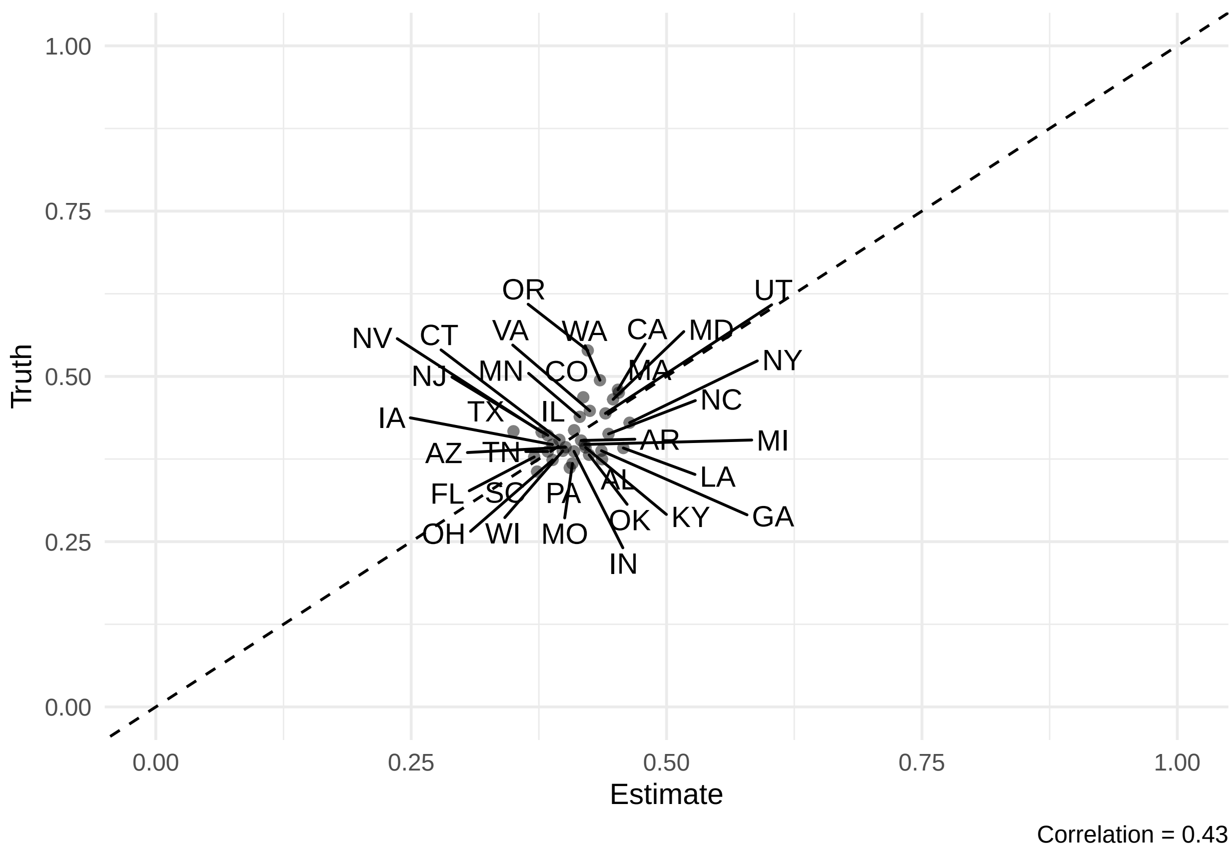


Figure 4: MRP estimates from model with partial pooling

```
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in s(educ, 2) :
##   unordered factors cannot be used as smoothing variables
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in s(educ, 2) :
##   unordered factors cannot be used as smoothing variables
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in s(educ, 2) :
##   unordered factors cannot be used as smoothing variables
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in s(educ, 2) :
##   unordered factors cannot be used as smoothing variables
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in s(educ, 2) :
##   unordered factors cannot be used as smoothing variables
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in s(educ, 2) :
##   unordered factors cannot be used as smoothing variables
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in s(educ, 2) :
##   unordered factors cannot be used as smoothing variables
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in s(educ, 2) :
##   unordered factors cannot be used as smoothing variables
## Error in predict.xgb.Booster(model, newdata = newX) :
##   Feature names stored in `object` and `newdata` are different!
## Error in s(educ, 2) :
##   unordered factors cannot be used as smoothing variables
```

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

```
##  SL.ranger_All     SL.gam_All SL.xgboost_All     SL.glm_All
##      0.4672328      0.0000000      0.0000000      0.5327672
```

```r
# 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)
```

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

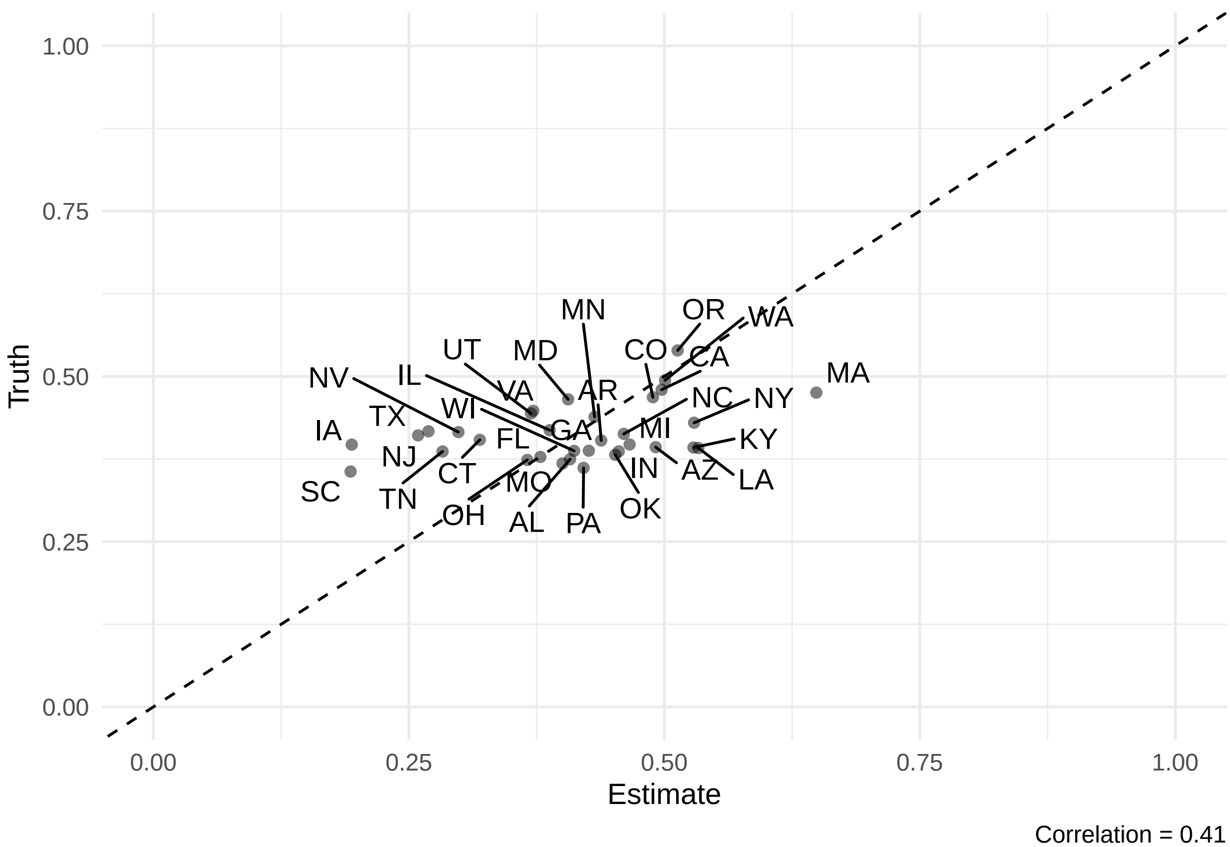


Figure 5: Estimates from an ensemble first-stage model

1.8 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![2] 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.

```
# 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
## 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> <fct>                <int>  <dbl>
## 1 AL    No HS                   49 0.0519
## 2 AL    High school graduate   287 0.304
## 3 AL    Some college           189 0.2
## 4 AL    2-year                 122 0.129
## 5 AL    4-year                 179 0.189
## 6 AL    Post-grad              119 0.126
```

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

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

```
  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> <fct>                <int>  <dbl>
## 1 AL    Female Asian    24     1 0.0000549 No HS                   49 0.0519
## 2 AL    Female Asian    24     1 0.000321  High school graduate   287 0.304
## 3 AL    Female Asian    24     1 0.000212  Some college           189 0.2
## 4 AL    Female Asian    24     1 0.000137  2-year                 122 0.129
## 5 AL    Female Asian    24     1 0.000200  4-year                 179 0.189
## 6 AL    Female Asian    24     1 0.000133  Post-grad              119 0.126
```

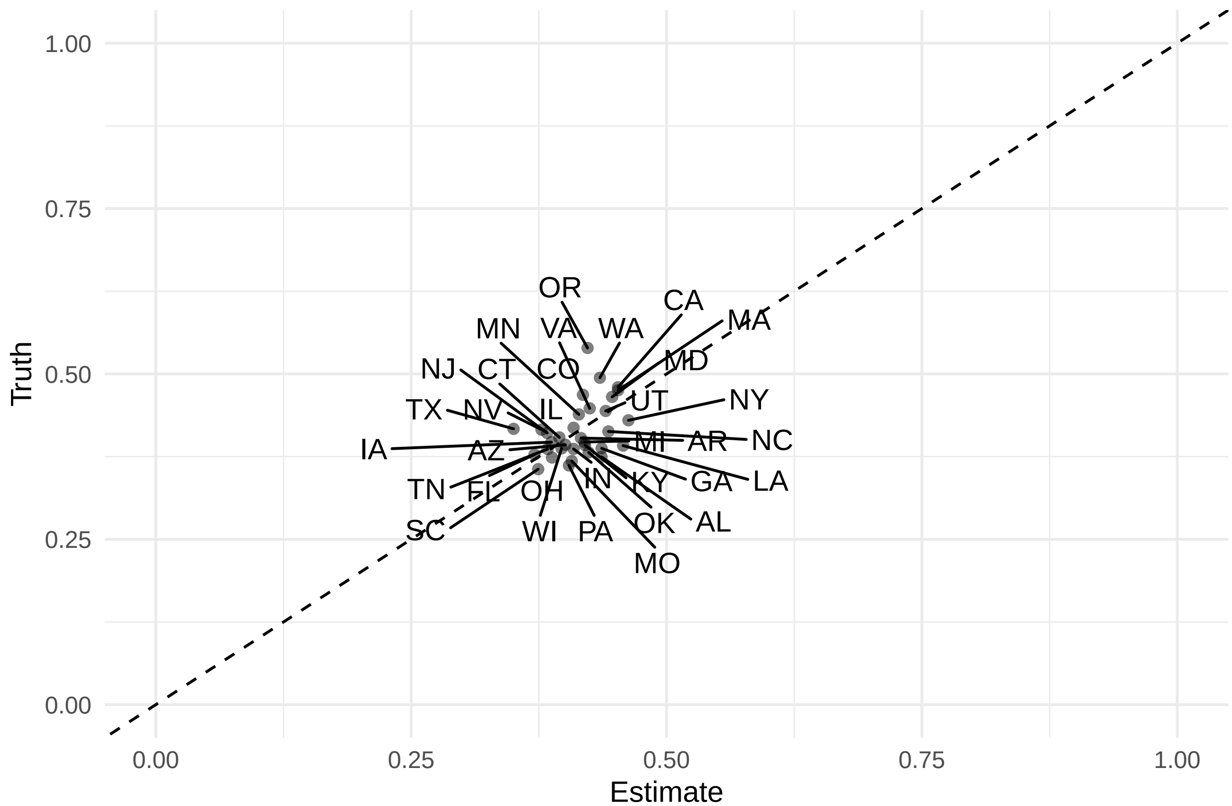
The SRP package contains a convenience function for this operation (see the vignette for more information).

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)
```

References

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.

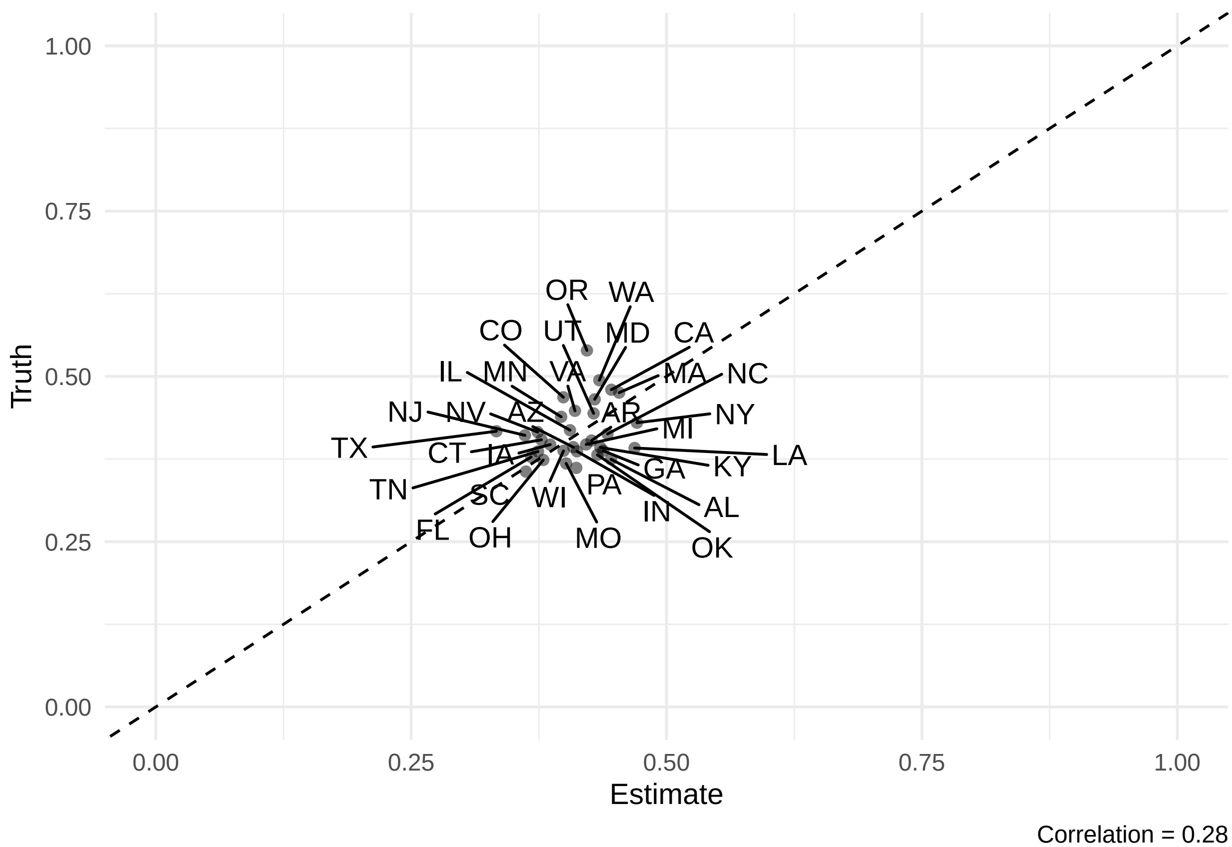


Figure 6: Poststratified estimates, omitting education

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.

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.

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