Regularized Raking

for better survey estimates

Andy Timm

December 4, 2023

Introduction

Raking is among the most commonly used algorithms for building survey weights such that a unrepresentative sample can be used to make inferences about the general population.

Regularized Raking extends this framework, allowing for more explicit and granular tradeoffs to be made on properties of the weights set.

Goals:

1. Review vanilla raking, and clarify in what sense the weights it produces are 'optimal'.
2. Argue that vanilla raking's implicit objective isn't a great fit for modern survey inference, and show why Regularized Raking is a good framework for improvements.

Outline

- Quick about me (& my weights!)
- Raking: easy mode (introduce/review vanilla raking)
- Raking: hard mode (when raking gets confusing)
- Introduce regularized raking
- Examples where RR makes a difference
- RR in the survey weights cinematic universe (comparisons to MRP, multilevel calibration, broader challenges of modern survey inference, etc)

About me (& my weights)

- Data scientist working in politics and advertising; both have given me a lot of cause to think hard about weights.

Some recent types of weights I've worked on:

- TV viewership weights (weight ~41M TV viewership panel to US TV viewing gen pop)
- Political 'Microtargeting' Surveys (sample 5-10k people, weight to voter file, then model)
- COVID Vaccine Message Testing (multi-arm RCT to convince people to get vaccine, weight to customers of our pharmacy partner)
- + More (Market Research, RCTs in politics, RCTs in advertising with survey outcomes…)

Raking (Easy Mode)

Let’s introduce/review raking!

We have a **non-representative** sample from a larger universe, which we want to use to make inferences about that universe. How do we best assign weights to each observation in the sample to make (weighted) estimates representative?

In Easy Mode:

Raking

Raking is an iterative algorithm for building sample weights such that (weighted) sample marginal distributions match population marginal distributions.

We'll generate a synthetic universe, sample in biased fashion from it, then rake it by hand and with `autumn` in R.

Generating Data

```r
n_universe <- 100000
n_sample <- 1000

# Universe Data Generating Process
universe <- tibble(
  age = sample(c("young", "old"),
               n_universe, replace = TRUE,
               prob = c(0.5, 0.5)),
  race = sample(c("white", "black", "asian", "hispanic", "other"),
                n_universe, replace = TRUE,
                prob = c(0.6, 0.1, 0.1, 0.1, 0.1)))

# Defining the outcome (probability of enjoying pineapple on pizza)
universe <- universe %>%
  mutate(enjoys_pineapple = plogis(rnorm(n_universe,-1,.5) +  #bias term
                                1*ifelse(race == "white", 1, 0) + # More likely if white
                                .5*ifelse(age == "old", 1, 0))) # More likely if older
```

Check distribution

```r
universe %>% summarize(mean_p = mean(enjoys_pineapple))
```

```
# A tibble: 1 × 1
  mean_p
   <dbl>
1  0.467
```

```r
universe %>%
  group_by(age,race) %>%
  summarize(mean_p = mean(enjoys_pineapple)) %>%
  arrange(mean_p)
```

```
# A tibble: 10 × 3
# Groups:   age [2]
   age   race     mean_p
   <chr> <chr>     <dbl>
 1 young asian     0.277
 2 young black     0.279
 3 young other     0.281
 4 young hispanic  0.282
 5 old   other     0.382
 6 old   asian     0.383
 7 old   hispanic  0.385
 8 old   black     0.386
 9 young white     0.499
10 old   white     0.616
```

Sample with Bias

```r
# Parameters for the sample distribution
age_weights <- c(young = 1, old = 2)  # Old twice as likely
race_weights <- c(white = 5, black = 2,
                  asian = .5, hispanic = 1, other = 3)  # Also Wrong

universe <- universe %>%
  mutate(
    age_weight = age_weights[age],
    race_weight = race_weights[race],
    sample_weight = age_weight * race_weight
  )

# Draw the sample
sample_data <- universe %>%
  sample_n(size = n_sample, weight = sample_weight)

# Not our mean!
sample_data %>% summarize(mean_p = mean(enjoys_pineapple))
```

```
# A tibble: 1 × 1
  mean_p
   <dbl>
1  0.537
```

Shake and Rake

Raking proceeds variable by variable, adding weights to fix bias on 1 marginal at a time, until convergence.

```r
# 1. Determine the Target Distribution for Age in the Universe
age_distribution_universe <- universe %>%
  count(age) %>%
  mutate(proportion = n / sum(n))

# 2. Calculate Initial Sample Weights (assuming equal probability of selection initially)
sample_data <- sample_data %>%
  mutate(weight = 1)
```

Shake and Rake

```r
# 3. Raking Step: Adjust Weights for Age
# a. Calculate the distribution of age in the sample
age_distribution_sample <- sample_data %>%
  count(age) %>%
  mutate(proportion = n / sum(n))

# b. Join with the target distribution to get the adjustment factor
adjustment_factors <- age_distribution_sample %>%
  left_join(age_distribution_universe, by = "age", suffix = c("_sample", "_universe")) %>%
  mutate(adjustment_factor = proportion_universe / proportion_sample)

# c. Apply the adjustment factor to the sample weights
sample_data <- sample_data %>%
  left_join(adjustment_factors, by = "age") %>%
  mutate(raked_weight = weight * adjustment_factor)

# 2x weight for younger folks, as we'd expect
sample_data %>% group_by(age) %>% summarize(mean_weight = mean(raked_weight))
```

```
# A tibble: 2 × 2
  age   mean_weight
  <chr>       <dbl>
1 old         0.756
2 young       1.47
```

Shake and Rake

Now race

```
1 # Not showing calculation, since they're identical.
2 sample_data %>% group_by(race) %>% summarize(mean_weight = mean(raked_weight))
```

```
# A tibble: 5 × 2
  race     mean_weight
  <chr>          <dbl>
1 asian          6.68
2 black          1.87
3 hispanic       4.22
4 other          1.14
5 white          0.731
```

```
1 # Notice: slightly further from universe than after first iter
2 sample_data %>% group_by(age) %>% summarize(mean_weight = mean(raked_weight))
```

```
# A tibble: 2 × 2
  age   mean_weight
  <chr>       <dbl>
1 old         0.741
2 young       1.50
```

. . . and repeat until sample means on all marginals match population ones.

Less Manual: with autumn

autumn is a wonderful package for raking.

```
library(autumn)

target <- list(
  age = c(young = .5, old = .5),
  race = c(white = .6,
           hispanic = .1,
           black = .1,
           asian = .1,
           other = .1)
)
# See ?harvest for summary of other functionality
sample_data <- sample_data %>% harvest(target,max_weight = 10)

# With raked weights, we recover the population mean as expected
sample_data %>% summarize(mean_p = weighted.mean(enjoys_pineapple,weights))
```

```
# A tibble: 1 × 1
  mean_p
   <dbl>
1  0.467
```

Raking (hard Mode)

Back to reality

Sadly, reality makes weighting much harder.

- Phone response rates are now frequently less than 1%
- Many survey researchers now use online panels and/or non-probability sample methods
- Outcomes are frequently related to propensity to respond in complicated ways
- So respondents are getting increasingly weird, in ways we can't just weight out along a small set of census demographics

In other words, we live in a world of **Non-Ignorable Nonresponse**. What effects does this have on weighting?

A huge amount of pressure on weighting

Weighting — registered voters

The survey was weighted by The Times using the R survey package in multiple steps.

First, the samples were adjusted for unequal probability of selection by stratum.

Second, the six state samples were weighted separately to match voter file-based parameters for the characteristics of registered voters by state.

The following targets were used:

- Party (party registration if available, else classification based on a model of vote choice in prior Times/Siena polls)
- Age (Self-reported age, or voter file age if the respondent refuses)
- Gender (L2 data)
- Race or ethnicity (L2 model)
- Education (four categories of self-reported education, weighted to match NYT-based targets derived from Times/Siena polls, census data and the L2 voter file)
- Marital status (L2 model)
- Home ownership (L2 model)
- State regions (NYT classifications by county or city)
- Turnout history (NYT classifications based on L2 data)
- Vote method in the 2020 elections (NYT classifications based on L2 data)
- Census block group density (A.C.S. 5-Year Census Block Group data)
- City type (Nevada only, added based on a post-hoc analysis of the difference between the weighted sample and voter file parameters. The weight had no meaningful effect on the topline result.)
- Census tract educational attainment (Georgia only, added based on a post-hoc analysis of the difference between the weighted sample and voter file parameters. The weight had no meaningful effect on the topline result.)

- Trying to weight to cover all the issues sampling can't
- On the left, everything the NYT weighted their recent 2024 battleground state poll on
- Of course, we probably believe interactions of variables matter too…
- Many of these variables are included because they seem correlated with constructs we'd struggle to quantify directly

Raking on everything isn't free

- In many cases, raking on everything we want to won't even converge.
- So we end up picking and choosing which variables and interactions theoretically matter most.

Second class of issues: variance problems

- Raking on more dimensions can lower bias, but it often adds variance too
- This leads to heuristic ways to control variance
- Rules of thumb like desiring a $deff_{kish}$ < 1.5
- Trimming weights (cap/floor the tails of the weights distribution)

Restating problems with raking on hard mode

Let's restate the last few slides in terms of specific problems alternative weighting methods can solve:

1. Variables are either "in" or "out"
2. We have a fairly limited set of tools to manage variance of estimates
3. It's not clear in what sense the raking solution is optimal

Regularized Raking

Regularized Raking

(Barratt, Angeris, and Boyd 2021) show that the broader representative sample weights problem can be understood as an optimization problem:

$$\begin{aligned} \textbf{minimize} \quad & \ell\left(f, f^{\text{target}}\right) + \lambda r(w) \\ \textbf{subject to} \quad & w \geq 0, \quad \mathbf{1}^T w = 1 \end{aligned}$$

Let's step through this term by term.

Regularized Raking - Loss

$$\begin{aligned} \text{minimize} \quad & \ell\left(f, f^{\text{target}}\right) + \lambda r(w) \\ \text{subject to} \quad & w \geq 0, \quad \mathbf{1}^T w = 1 \end{aligned}$$

- We have f and f^{target}, which are functions of our sample and target population
- We'll also specify some loss function ℓ, which clarifies how we want f and f^{target} to be close

Regularized Raking

Loss examples - Ok, so what?

$$\ell\left(f, f^{\text{des}}\right) = \begin{cases} 0 & f = f^{target} \\ +\infty & \text{otherwise} \end{cases}$$

- In other words, f must match f^{target} exactly to be a valid solution
- If we take f to be a simple expected value, this is what vanilla raking requires on all raking dimensions!

Regularized Raking

Loss examples - Getting Interesting

$$\ell\left(f, f^{\text{des}}\right) = \begin{cases} 0 & f^{\min} \leq f \leq f^{\max} \\ +\infty & \text{otherwise} \end{cases}$$

- . . . But why do we always want exact matching on every raking variable?
- That would expand the realm of possible solutions!

Regularized Raking

Loss examples - Now we're getting somewhere

Regularized Raking

Loss examples - full force

- **We can have different losses for different f!**
- Example:
  - → require equality on single dimensions,
  - → least squares on all interactions, and
  - → different scaling such that different interactions are “worth” more according to theory or backtesting.

Regularized Raking

Regularization

$$\begin{aligned} \text{minimize} \quad & \ell\left(f, f^{\text{target}}\right) + \lambda r(w) \\ \text{subject to} \quad & w \geq 0, \quad \mathbf{1}^T w = 1 \end{aligned}$$

- We don't just care about adherence to population totals.
- What about variance of weights, shape of distribution, etc?
- We might want weights that are:
  - → As uniform as possible
  - → As close to some other target distribution with different variance properties
- $r(w)$ can be chosen from a variety of regularizers that encode such preferences
- Also have a hyperparameter λ, which allows us to make explicit tradeoffs between our loss term and our regularization term.

Regularized Raking

Regularization Example 1

$$r(w) = \sum_{i=1}^{n} w_i \operatorname{logw}_i$$

- This is the negative entropy, equivalent to the Kullback–Leibler divergence from the weights distribution w and the uniform distribution
- This would express a desire that our weights be as uniform as possible, subject to all our other constraints
- This is the other half of what vanilla raking does!

Regularized Raking

Regularization Example 2

$$D_{\mathrm{KL}}(w \parallel w^{des}) = \sum_{i=1}^{n} w \log\left(\frac{w}{w^{des}}\right)$$

- Alternatively, might want a weights distribution close to some target distribution w^{des}, where closeness is defined in terms of KL divergence
- Motivations: minimizing extreme tails, more demand for smoothness

Regularized Raking

Constraints

$$\begin{aligned} \textbf{minimize} \quad & \ell\left(f, f^{\text{des}}\right) + \lambda r(w) \\ \textbf{subject to} \quad & w \geq 0, \quad \mathbf{1}^T w = 1 \end{aligned}$$

- Non-negative weights are useful (not all estimators play well with negative weights)
- We want weights to sum to 1 (equivalently: rescale to sum to n)

Quick note that there's a third constraint required $f = Fw$ that I'm suppressing- it's necessary to state the

So what does vanilla raking do?

$$\ell\left(f, f^{\text{target}}\right) = \begin{cases} 0 & f = f^{\text{target}}, \\ +\infty & \text{otherwise}, \end{cases} \qquad r(w) = \sum_{i=1}^{n} w_i \log w_i$$

- Equality loss on all raking dimensions
- Negative entropy regularizer
- Hopefully, this gives some clarity about what raking optimizes for
- As we've seen, this is far from the only choice of ℓ and $r(w)$; are other choices better?

If you're interested in a precise statement of this, raking can be seen as coordinate ascent on the dual of the maximum entropy problem. See Appendix A of (Barratt, Angeris, and Boyd 2021) for a illustration of this, or

Examples where RR helps

Data

Let's get back into some code, and comparisons. I'll be using a poll conducted by Pew in 2016, with 2074 likely voter respondents. The outcome of interest is **vote margin**, or the number of voters who favored Clinton minus those who favored Trump.

I like this dataset for a few reasons:

- Easy access
- There is a correct answer, and we know it (Hilary won the national vote by 2.2%)
- If you do not weight by education, you will be wrong by like 10 points :)
- Decent variety of covariates

We'll treat estimates from the 2016 CCES as a target, because it's huge, representative, and also easily accessible.

Introducing `rsw`

Let's walk through a basic example in both `survey`, and `rsw`, the python package from (Barratt, Angeris, and Boyd 2021) to build familiarity. We'll weight just on age, gender, race, and region to show syntax.

R First

```r
formula_1 <- ~recode_age_bucket + recode_female + recode_race + recode_region

# Convenience function to make targets for a formula given a `survey` design obj
target_1 <- create_targets(cces_wt_design,formula_1)
rsw_design_mat_1 <- rsw_design_matrix(pew_design, formula_1)
rsw_target_1 <- rsw_target(cces_wt_design, formula_1)

pew_raked_1 <- calibrate(design = pew_design,
                         formula = formula_1,
                         population = target_1,
                         calfun = "raking")

svycontrast(svymean(~recode_vote_2016, pew_raked_1, na.rm = TRUE),
                          vote_contrast)
```

```
            nlcon     SE
contrast 0.08901 0.0259
```

Now with `rsw`

```
import rsw

design_mat_1 = pd.read_csv("rsw_inputs/rsw_design_mat_1.csv")
target_1 = pd.read_csv("rsw_inputs/rsw_target_1.csv")

losses = [rsw.EqualityLoss(np.array(target_1).flatten())]
regularizer = rsw.EntropyRegularizer()
w, out, sol = rsw.rsw(df = design_mat_1,
                      funs = None,
                      losses = losses,
                      regularizer = regularizer,
                      lam = 1)
```

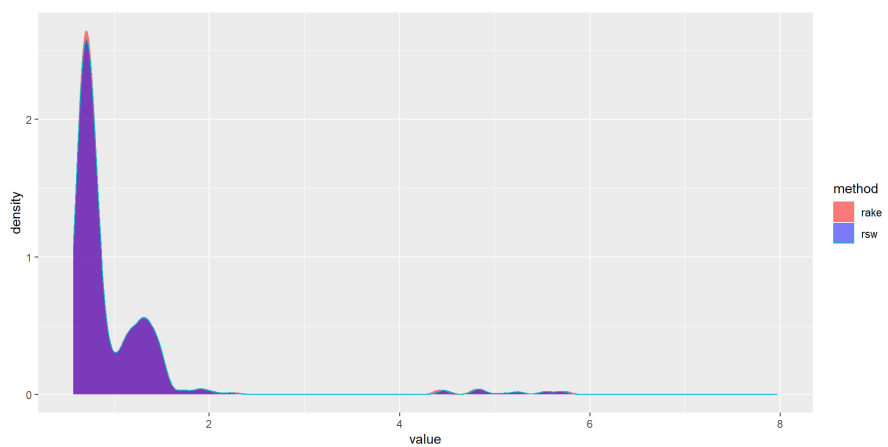
Steps:

Illustration: this is what raking is

```
rsw_design_1 <- svydesign(ids = ~1, data = pew, weights = rsw_weights_1$value,
                # Ensures we handle these not being sampling weights
                # See: https://cran.r-project.org/web/packages/survey/vignettes/precalibrated.pdf
                calibrate.formula = formula_1)

svycontrast(svymean(~recode_vote_2016, rsw_design_1, na.rm = TRUE),
                          vote_contrast)
```

```
             nlcon     SE
contrast 0.089042 0.0257
```

```
compare_weights_1 %>%
  ggplot(aes(x = value, color = method, fill = method)) +
  geom_density(alpha=.5)+
  scale_fill_manual(values=c("red","blue"))
```



About as far as raking will take us

```r
formula_2 <- ~ recode_age_bucket + recode_female +
                recode_race *recode_region * recode_educ_3way


# Convenience function to make targets for a formula given a `survey` design obj
target_2 <- create_targets(cces_wt_design,formula_2)

pew_raked_2 <- calibrate(design = pew_design,
                         formula = formula_2,
                         population = target_2,
                         calfun = "raking")

svycontrast(svymean(~recode_vote_2016, pew_raked_2, na.rm = TRUE),
                          vote_contrast)
```

```
            nlcon     SE
contrast 0.011029 0.0271
```

Taking advantage of RR's flexibility

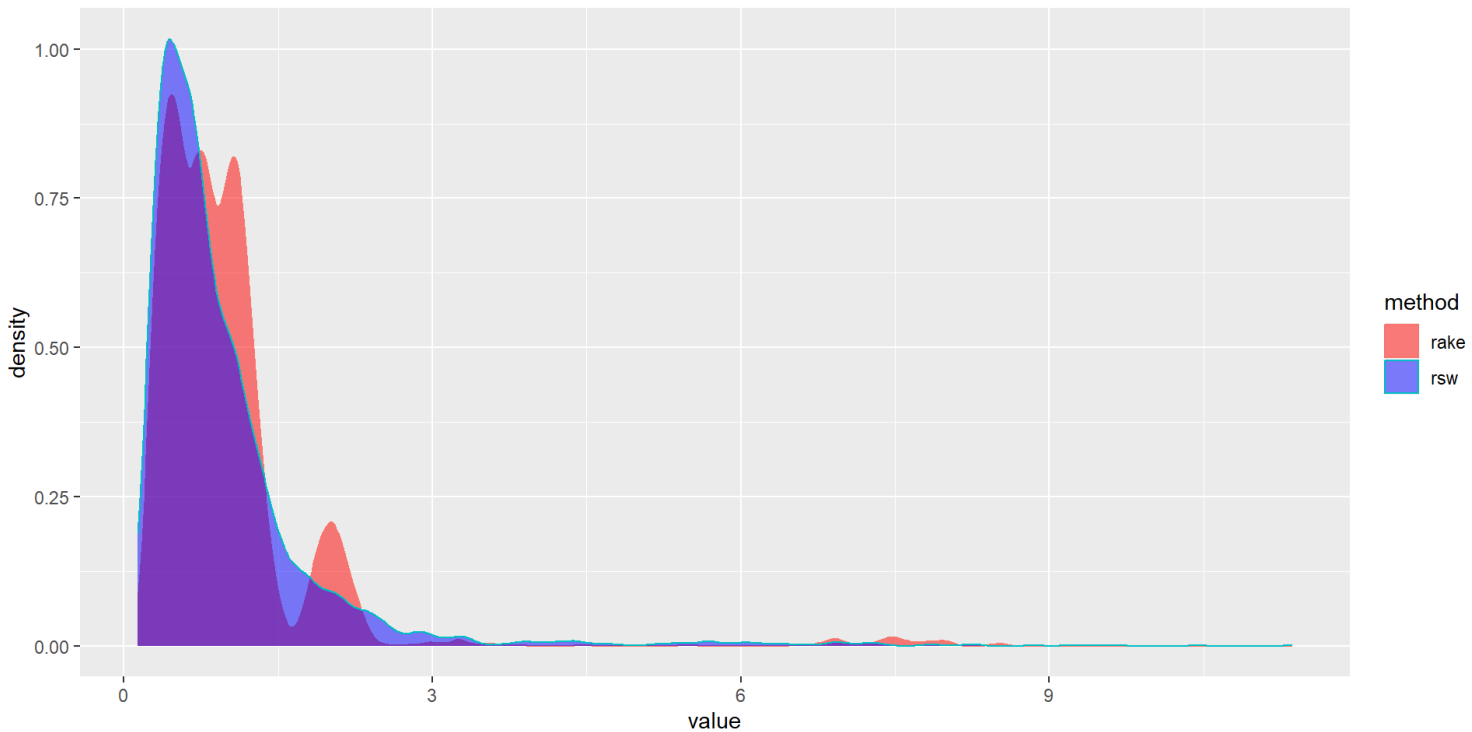
```
formula_3 <- ~recode_age_bucket + recode_female +
                recode_inputstate + recode_race * recode_region * recode_educ
```

```
design_mat_2 = pd.read_csv("rsw_inputs/rsw_design_mat_2.csv")
target_2 = pd.read_csv("rsw_inputs/rsw_target_2.csv")

# Take advantage of interactions being last in design matrix
main_fx = target_2.iloc[:71]
interactions = target_2.iloc[71:]

# Equality Loss on mains, LS on interactions for flexibility
losses = [rsw.EqualityLoss(np.array(main_fx).flatten()),
          rsw.LeastSquaresLoss(np.array(interactions).flatten())]

regularizer = rsw.EntropyRegularizer()
w_2, out_2, sol_2 = rsw.rsw(df = design_mat_2,
                        funs = None,
                        losses = losses,
                        regularizer = regularizer,
                        lam = 1)
```

How does this do?

Pretty happy with this

```
compare_weights_2 %>%
  ggplot(aes(x = value, color = method, fill = method)) +
  geom_density(alpha=.5)+
  scale_fill_manual(values=c("red","blue"))
```




```
            nlcon     SE
contrast 0.021118 0.0261
```

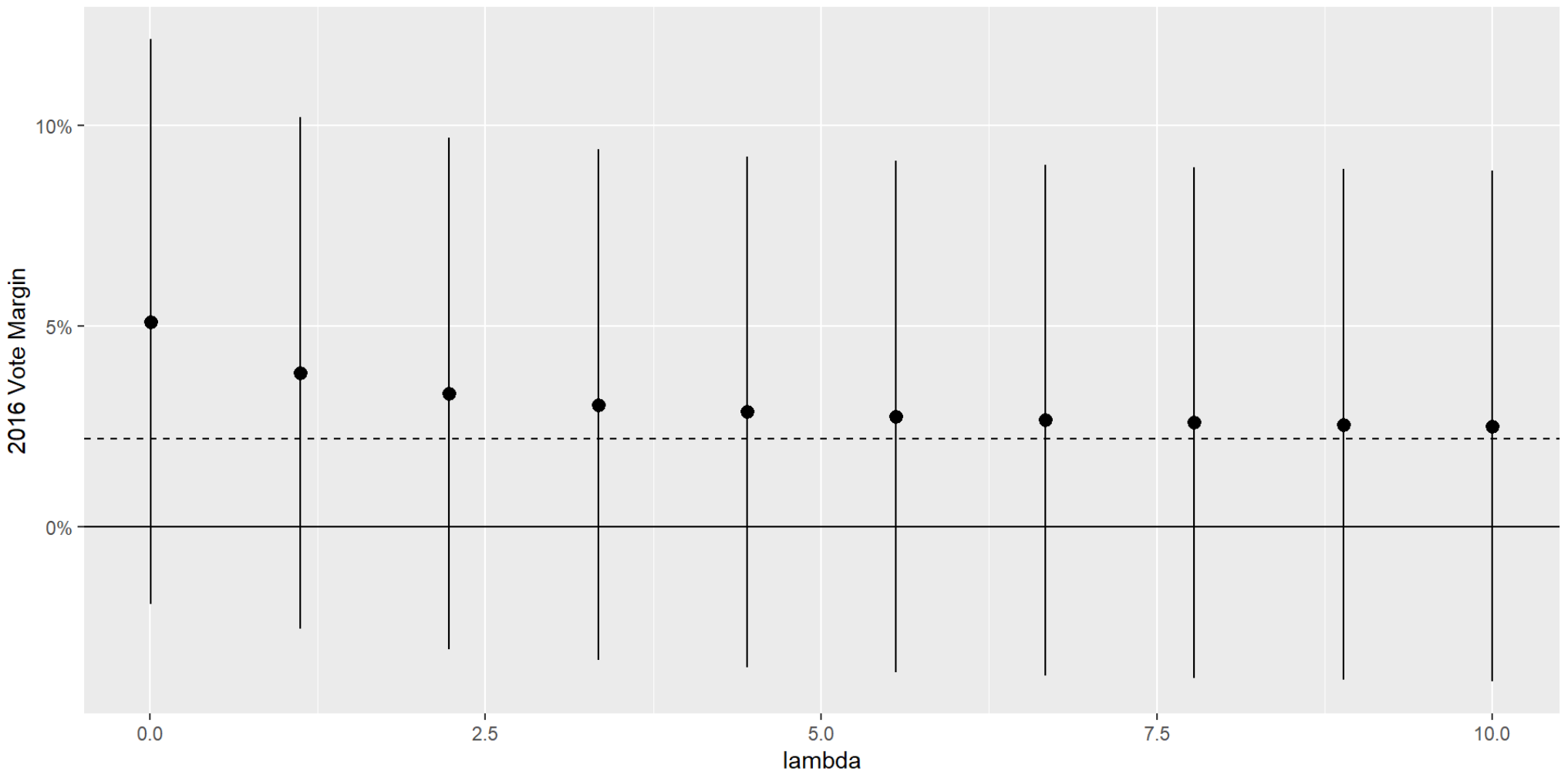
Pushing it further to see regularization

```
formula_4 <- ~recode_inputstate + recode_age_bucket * recode_female *
                  recode_race * recode_region * recode_educ * recode_pid_3way
```

```
design_mat_3 = pd.read_csv("rsw_inputs/rsw_design_mat_3.csv")
target_3 = pd.read_csv("rsw_inputs/rsw_target_3.csv")

# Take advantage of interactions being last in design matrix
main_fx = target_3.iloc[:71]
interactions = target_3.iloc[71:]

# Equality Loss on mains, LS on interactions for flexibility
losses = [rsw.EqualityLoss(np.array(main_fx).flatten()),
          rsw.LeastSquaresLoss(np.array(interactions).flatten())]

regularizer = rsw.EntropyRegularizer()
w_3, out_3, sol_3 = rsw.rsw(df = design_mat_3,
                        funs = None,
                        losses = losses,
                        regularizer = regularizer,
                        lam = 10)
```

Varying lambda can change estimates

I won't show a ton of search over λ, but just to illustrate:



Regularized Raking in the Survey Weights Cinematic Universe


AAPOR
GUARDIANS OF THE
POLLING

Versus other modern options

MRP (and friends!)

Multilevel Regression and Postratification (Gelman et al. 2019) use a regularized model to predict the outcome, and poststratify.

Advantages

- with RR, you get typical weights, which can be applied to any outcome
- RR is OOMs faster, especially for big data and complex models
- Often, a little regularization goes a long way

Disadvantages

- Sometimes you actually want the whole posterior!
- For most small area estimates, MRP is going to be more efficient
- More generally, greater variety and flexibility of regularization

Versus other modern options

Multilevel Calibration

Multilevel Calibration (Ben-Michael, Feller, and Hartman 2023) also takes an optimization approach, requires exact matching on the marginals, more flexibility on interactions

Advantages

- RR comes with greater flexibility to define loss, regularization
- User retains more ability to make tradeoffs

Disadvantages

- MC will choose how deep the interactions should go for you- a lot less choice involved
- Gets most of the value of approximate balance on interactions without iteration

Regularized Raking in the Survey Weights Cinematic Universe

I've only been talking about only weighting today, so want to emphasize:

- Good sampling matters (and more than how you weight)
- Selection of weighting variables matters (and more than how you weight)

That said, how you weight is entangled with these, as more flexible weighting gives you more options with the above.

Regularized Raking gets a lot of the benefits of more heavily model based based approaches without huge workflow changes, and is a more realistic fit for the demands of modern weighting.

Thank you!

Hire me!

- Currently looking political/non-profit data science roles.
- Experience as a data science manager, data scientist, and campaign staffer.
- keywords: Causal Inference, Bayes, HTE estimation, Surveys

Materials/Suggestions

- QR code for repo with full talk materials to left
- (Barratt, Angeris, and Boyd 2021) for finer details of optimization behind talk today
- For perspective on challenges of modern survey weighting, highly recommend **A New Paradigm for Polling** (Bailey 2023)
- For how to think about picking targets and estimating them, I adore (Caughey et al. 2020)- **Target Estimation and Adjustment Weighting for Survey Nonresponse and Sampling Bias**

References

Bailey, Michael A. 2023. “A New Paradigm for Polling.” *Harvard Data Science Review* 5 (3). https://doi.org/10.1162/99608f92.9898eede.

Barratt, Shane, Guillermo Angeris, and Stephen Boyd. 2021. “Optimal Representative Sample Weighting.” *Statistics and Computing* 31 (2): 19. https://doi.org/10.1007/s11222-021-10001-1.

Ben-Michael, Eli, Avi Feller, and Erin Hartman. 2023. “Multilevel Calibration Weighting for Survey Data.” *Political Analysis*, March, 1–19. https://doi.org/10.1017/pan.2023.9.

Caughey, Devin, Adam J. Berinsky, Sara Chatfield, Erin Hartman, Eric Schickler, and Jasjeet S. Sekhon. 2020. “Target Estimation and Adjustment Weighting for Survey Nonresponse and Sampling Bias.” *Elements in Quantitative and Computational Methods for the Social Sciences*, September. https://doi.org/10.1017/9781108879217.

Deville, Jean-Claude, Carl-Erik Särndal, and Olivier Sautory. 1993. “Generalized Raking Procedures in Survey Sampling.” *Journal of the American Statistical Association* 88 (423): 1013–20. https://doi.org/10.1080/01621459.1993.10476369.

Gelman, Andrew, Jeffrey Lax, Justin Phillips, Jonah Gabry, and Robert Trangucci. 2019. “Using Multilevel Regression and Poststratification to Estimate Dynamic Public Opinion.” *Unpublished Manuscript*, August, 48.

Teh, Yee Whye, and Max Welling. 2003. “On Improving the Efficiency of the Iterative Proportional Fitting Procedure.” In *International Workshop on Artificial Intelligence and Statistics*, 262–69. PMLR. https://proceedings.mlr.press/r4/teh03a.html.