Online Appendix:

Improved Multilevel Regression with Post-Stratification Through Machine Learning (autoMrP)*

Philipp Broniecki[†] Lucas Leemann[‡] Reto Wüest[§]

Contents

[†]ESRC Business and Local Government Data Research Centre, University of Essex, United Kingdom. Email: philipp.broniecki@essex.ac.uk, URL:]

[‡]Department of Political Science, University of Zurich, Switzerland. Email: leemann@ipz.uzh.ch, URL: http://www.lucasleemann.ch

[§]Department of Comparative Politics, University of Bergen, Norway. Email: reto.wueest@uib.no, URL: http://www.retowuest.net/

1 The Standard MrP model

The goal of MrP is to produce preference estimates for subnational units. The strategy is to estimate the average support among specific groups in society and then to weigh these estimates according to the prevalence of the groups in a given subnational unit. Specific groups in society are defined by a set of individual-level variables. In our real-world application, these individual-level variables are age category, education level, and gender-race combination. Based on these variables we can define ideal types. An example of such an ideal type is a young black woman with a university degree. As there are four age groups, four education groups, and six gender-race categories, we have 96 different ideal types, one for each combination of age group, education group, and gender-race category.

The model allows taking into account geographic variation that is not solely due to a different socio-economic make-up of the subnational populations by including a random effect for the subnational units. By relying on a hierarchical model where individuals are nested in subnational units, context-level variables can be included in the estimation. Hence, the estimates are based on individual ideal types on the one hand, and variation between subnational units that is not due to differences in the make-up of their populations on the other hand.

Technically, this is achieved by estimating a binary hierarchical model where the outcome variable y_i is a function of individual-level random effects for age, education, and gender-race (α_a^{age}, $\alpha_m^{education}$, $\alpha_j^{gender\text{-}race}$). In addition, there is a random effect that varies over the subnational units ($\alpha_c^{subnational\ unit}$). At the context level, the model may include subnational unit-specific predictors $\mathbf{X}_c$, such as Republican presidential vote share, shares of religious groups, and other measures that vary across subnational units.

$$
\begin{aligned}
Pr(y_i = 1) &= \Phi\left(\mathbf{X}_c'\boldsymbol{\beta} + \alpha_{a[i]}^{age} + \alpha_{m[i]}^{education} + \alpha_{j[i]}^{gender\text{-}race} + \alpha_{c[i]}^{subnational\ unit}\right), \qquad (1)\\
\alpha_a^{age} &\sim N(0, \sigma_{age}^2), \text{ for } a = 1, ..., A,\\
\alpha_m^{education} &\sim N(0, \sigma_{education}^2), \text{ for } m = 1, ..., M,\\
\alpha_j^{gender\text{-}race} &\sim N(0, \sigma_{gender\text{-}race}^2), \text{ for } j = 1, ..., J,\\
\alpha_c^{subnational\ unit} &\sim N(0, \sigma_{subnational\ unit}^2), \text{ for } c = 1, ..., C.
\end{aligned}
$$

In a second step, the average support of each ideal type in a given subnational unit (π_{amjc}) can be estimated based on Equation 1. For example, we might estimate the average support for a proposal among young black women with a university degree in subnational unit c. The prediction of the overall support in subnational unit c is then obtained by post-stratifying these ideal type-based predictions. That is, the support of each of the 96 ideal types is weighted by their relative share in the true population in c as shown in Equation 2.

$$
\hat{\pi}_c = \frac{\sum_a \sum_m \sum_j \hat{\pi}_{amjc} N_{amjc}}{N_c} = \frac{\sum_a \sum_m \sum_j \Phi\left(\mathbf{X}_c'\hat{\boldsymbol{\beta}} + \hat{\alpha}_a + \hat{\alpha}_m + \hat{\alpha}_j + \hat{\alpha}_c\right) N_{amjc}}{N_c}. \qquad (2)
$$

The model in Equation 1 takes into account that individual preferences may vary due to socio-economic characteristics at the individual level and it also incorporates the possibility that areas may differ from one another—so that some areas, for example, may be more progressive than others—irrespective of the socio-economic structure of their population. Based on census data of the true population in the subnational units, it is then possible to post-stratify the predicted support per ideal type.

2 Individual Classifiers

We rely on five classifiers that we describe in more detail below. The flexibility of our approach lies in the fact that anybody can add new classifiers or drop one of these in a specific application.

(i) Best Subset. In multilevel regression with best subset selection, our goal is to choose the subset of context-level variables that minimizes the out-of-sample prediction error (Hastie et al., 2009, 57f.). Let S be the set of all candidate variables for the context level. Given S, we fit a separate model for each combination of candidate variables, resulting in $N = 2^p$ fitted models, where $p = |S|$. Among the N fitted models, we choose the one with the smallest out-of-sample MSE. We rely on cross-validation to estimate the expected out-of-sample MSE. Instead of only taking into account the combinations of candidate variables, we could also consider polynomials and interactions between the variables. This, however, would rapidly increase the computation time necessary to select the optimal model.

(ii) PCA. Principal components analysis (PCA) is a procedure that converts a set of possibly correlated variables into a set of uncorrelated linear combinations of the original variables, called principal components (PCs, Hastie et al., 2009, 79f.). Using PCs instead of the original variables as context-level predictors in the MrP model allows us to reduce the number of variables while retaining most of the information in the data. PCA also allows us to overcome inherent problems with highly correlated predictors. Multicollinearity can lead to large variances of the estimated coefficients and unreliable coefficient estimates. As PCs are orthogonal to one another, there are no multicollinearities between them (**?**, 167ff.). PCA hence serves two purposes: it may reduce the number of context-level predictors and it avoids context-level multicollinearity. We proceed as follows. First, we use PCA to find the PCs of the p original context-level variables. Second, we rely on cross-validation to choose the subset of PCs that minimize the estimated prediction error.

(iii) Lasso. The previous procedures attempt to mitigate overfitting by selecting a subset of context-level predictors. Another approach to reduce the risk of overfitting is to rely on

$L1$ regularization (Lasso). The Lasso model includes a penalty that shrinks the coefficient estimates of context-level predictors towards zero and, when the tuning parameter λ is set to a sufficiently large value, forces (some of) them to be exactly equal to zero (Hastie et al., 2009, 68ff.). Lasso thus provides protection against overfitting through shrinkage and possibly also through variable selection. We use cross-validation to choose the optimal λ that minimizes the estimated prediction error.

(iv) GB. Our fourth classifier replaces the multilevel model in MrP with gradient tree boosting (see also **?**). At the core of gradient boosting are regression or classification trees, which, in our case, are simple classification rules involving the predictors at individual and context level (save the indicators for subnational units). The idea in gradient boosting is that a large number of trees are grown sequentially, with each tree being fit to the pseudo residuals from the previous model. Following Ridgeway (2007, 6) and Hastie et al. (2009, 361), our tuning parameters are the number of trees we grow, T, the maximum depth of each tree t, D_t, and the learning rate, λ. More details are provided in the online appendix (Section 7). We choose the set of tuning parameters that minimize the estimated prediction error using cross-validation.

(v) SVM. Our fifth classifier replaces the multilevel model in MrP with support vector machine (SVM). SVMs construct a non-linear decision boundary (a kernel) in the feature space that separates the two classes of the outcome (Hastie et al., 2009, 423). We use a computationally efficient radial kernel and cross-validate to choose the optimal values of two tuning parameters, c and γ: parameter c regulates the bias-variance trade-off and γ the basis of the radial kernel (Hastie et al., 2009, 430-432).

3 Loss Function: Individual vs. State Level

We use 5-fold cross-validation to tune the parameters of our five classifiers. Our quantity of interest is state-level public opinion. Since we lack information on true state-level opinion,

we evaluate model performance by creating a benchmark based on the mega-sample. We explored two alternative loss functions. In the letter, our loss function is the mean squared error at the individual level (Brier score, Brier, 1950):

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

where y_i is the actual vote choice of individual i and $\hat{y}_i$ is the predicted probability of i's vote choice. With this loss function, we optimize the model at the individual level. We also experimented with an alternative loss function that evaluates model performance at the state level:

$$\frac{1}{|S|}\sum_{S}\frac{1}{n_s}\sum_{i\in s}(y_{is} - \hat{y}_{is})^2,$$

where $s \in S$ indicates a state, $|S|$ denotes the number of states over which we evaluate, and $i \in s$ denotes individual i living in state s. We first compute the average prediction error per state and then average over all states. This approach avoids the problem that the MSE is dominated by the error in large states, i.e., states from which there are many respondents in the sample. The state-level loss function corresponds more closely to the quantity of interest: subnational support for each of the 89 political issues in the BH data set that we evaluate in our application. However, it turned out that the performance of the models optimized at the individual level is slightly better than the performance of the models optimized at the state level.

Table 1: Performance of Individual vs. State-Level Loss Function

	Individual-Level MSE	State-Level MSE
EBMA	0.00191	0.00196
% Reduction in error over BH baseline	12.2	10.0

Notes: The table compares the performance of our approach when we optimize the five classifiers at the individual level with performance when we optimize them at the state level. While the difference is moderate, optimizing at the individual level outperforms optimizing at the state level. Using individual-level optimization, we reduce the MSE compared to the BH baseline by 12.2%. Using optimization at the state level, we reduce the MSE by 10%.

Table 1 shows the results for both loss functions. Optimizing at the individual level outperforms optimizing at the state level. With the former approach we reduce the MSE by 12.2% compared to the BH baseline. With the latter approach, we reduce the MSE by 10% compared to the BH baseline.

While the state-level MSE is closer to the quantity of interest—i.e., state-level public opinion—it suffers from shortcomings that are similar to those of the disaggregation approach. Disaggregation uses the average of all respondent preferences in a specific subnational unit to create an estimate for that unit (Miller and Stokes, 1963). The problem with this approach is, however, that for small subnational units many surveys contain only a handful of respondents. The (weak) law of large numbers states that as the size of a random sample increases indefinitely, the sample average converges in probability to the mean of the distribution from which the sample was taken (Lax and Phillips, 2009). Therefore, if the sample size is small for a subnational unit, disaggregation is unlikely to produce an estimate that is close to the population mean.

4 Comparison with BARP

In an approach similar to ours, Bisbee (2019) proposes combining Bayesian additive regression trees with post-stratification (BARP) to improve upon the conventional MrP model. While a thorough comparison of our approach with alternative approaches including conditions under which one might outperform the other is beyond the scope of this letter, we provide a quick comparison to the BARP approach here. In what follows, we rely on the publicly available R package `BARP` to implement the model proposed by Bisbee (2019). We rely on the package's default settings to estimate our models. The main differences between our approach and BARP are:

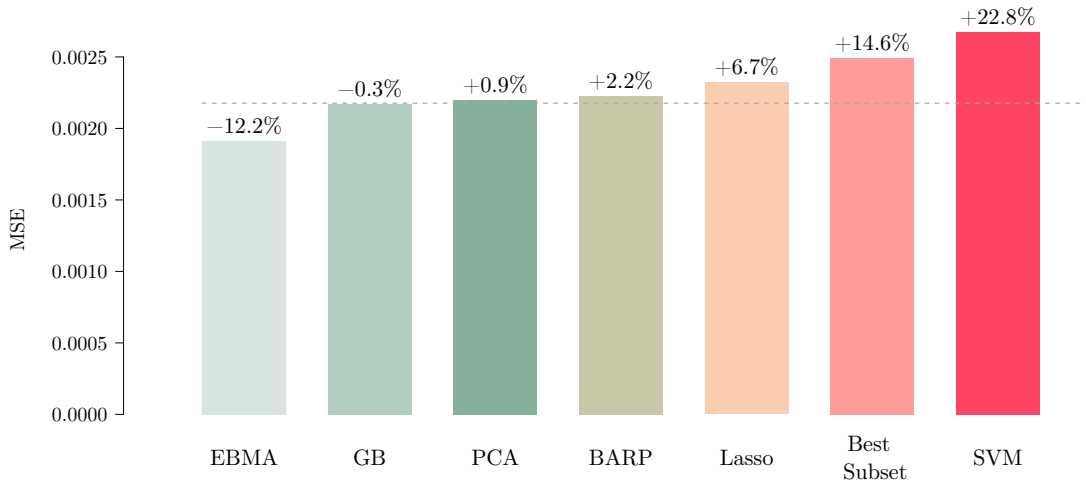
1. Our approach includes selection from context-level variables, whereas Bisbee (2019) focuses on individual-level variables.

2. We combine a larger set of classification methods through model averaging while BARP proposes one approach that yields strong overall results.

3. We tune all classifiers in our application, whereas in Bisbee (2019) BARP runs on out-of-the-box parameter settings: 250 trees, 250 burn-in, 1000 iterations after burn-in. Note that parameter tuning is possible with BARP but computationally expensive.

For the purpose of comparison, we have set up BARP as in Bisbee (2019), i.e., 250 trees, 250 burn-in, 1000 iterations after burn-in, and let it rely on the same two context-level variables as in Bisbee (2019). In our exercises BARP performs slightly weaker than our approach and has a mean squared error that is 14% larger than ours across the 89 items. Our model has an MSE of 0.0019, while the BARP model has an MSE of 0.0022 (see Figure 1).

Figure 1: Comparison with Replication of BARP

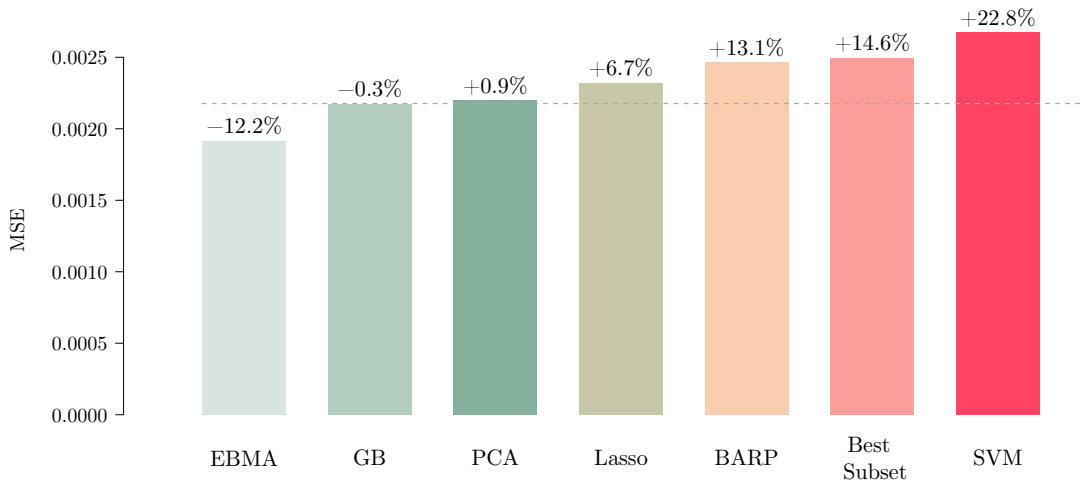


Note: Average MSE of state-level predictions over 89 survey items. Dashed line indicates MSE of the BH model. Percentage numbers show change in MSE relative to the BH model. Example: EBMA reduces prediction error by 12.2% compared to the baseline model.

We also compare our approach to BARP in a setup where we again use 250 trees, 250 burn-in and 1000 iterations after burn-in, but provide BARP with the same six context-level variables that we rely on in our approach (see Figure 2). In this case, BARP's performance decreases relative to our approach: the MSE of the BARP model is now 22% larger than

the MSE of our EBMA model. We suspect that the performance decrease of BARP, when using all six context-level variables, is indication of the risk of over-fitting that is especially pronounced for context-level variables. Furthermore, the performance drop speaks to the need for parameter tuning that is possible with BARP but not currently implemented in the BARP package.

Figure 2: Comparison with BARP where BARP Employs all 6 Context-level Variables



Note: Average MSE of state-level predictions over 89 survey items. Dashed line indicates MSE of the BH model. Percentage numbers show change in MSE relative to BH model. Example: EBMA reduces prediction error by 12.2% compared to the baseline model.

We are not entirely sure what accounts for the performance difference between our approach and BARP and leave this question for future research. It is possible that our approach outperforms BARP here because the latter relies on Bayesian additive trees whereas we use a broader set of classification methods, some of which have a more linear flavor. But if this is our advantage here there might be other applications where BARP outperforms our approach since we rely on a weighted average, through EBMA, of various classifiers. Some of those are more linear and some more flexible. What is more, Bisbee (2019) does not tune the parameters (trees, burn-in, and iterations after burn-in) whereas we tune the parameters of our classifiers. Finally, while tuning BARP turned out to be too computationally expensive, in the future, Bayesian additive regression trees could be included in our approach as

a candidate classifier.

5 Alternative Benchmark - How to Work with the BH Data

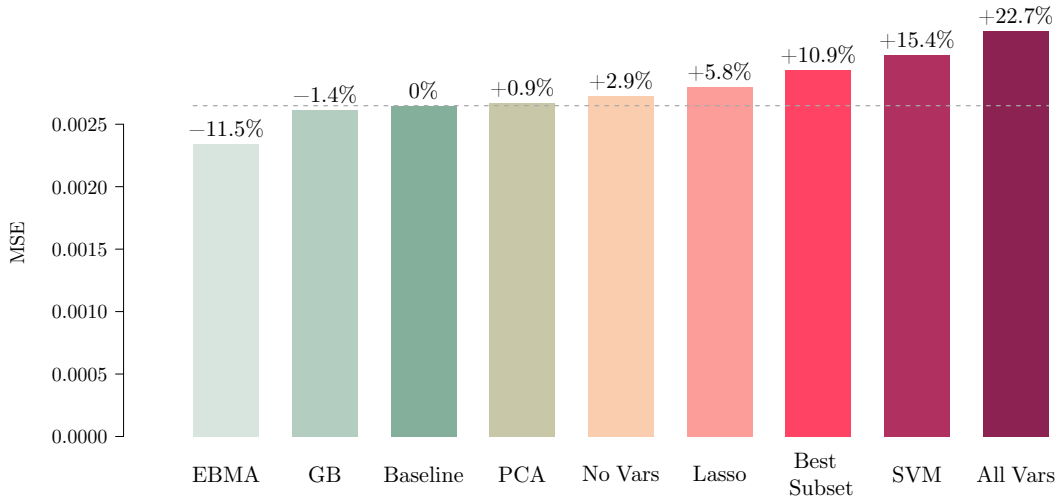
The results presented in the manuscript are all based on a comparison with what is labeled state-level true support. This support is calculated by looking at the state-level averages in the total mega-sample, i.e., all survey responses (disaggregation). This follows the setup on which BH relied. But the question is raised as to whether this is the best guess of what the true state-level preference is. Another approach is to take the data in the mega-sample and then apply some form of calibration, such as raking, to calculate the state-level truths.

We also did this and used the individual-level variables from the BH analysis for which we had census information. That is, we raked the survey data relying on age, education, and race times gender. This produces a different truth than just using the raw data. We replicated our analysis with this alternative truth measure.

Figure 5 shows the relative performance of the various approaches and is a replication of Figure 1 in the manuscript (but based on the alternative truth measure). The results are almost identical to what we find with the original truth measure, which does not rely on raking to derive the true state preference. All nine approaches are in the same order and the only change we see is that our proposed approach is on average 11.5% better than the benchmark with the alternative measure; it is 12.2% better than the benchmark with the original truth measure. This additional analysis underscores that EBMA outperforms the benchmark clearly and this also holds when we use an alternative truth measure to evaluate the estimates.

We thank a reviewer for raising this issue and motivating this additional analysis

Figure 3: Comparison with Alternative Benchmark



Note: Average MSE of state-level predictions over 89 survey items. *Baseline* model is from BH, *No Vars* is empty at the context level, and *All Vars* includes all six context-level variables. Dashed line indicates MSE of the BH model. Percentage numbers show change in MSE relative to BH model. Example: EBMA reduces prediction error by 11.5% compared to the baseline model.

6 Tuning Ensemble Bayesian Model Averaging

We use ensemble Bayesian model averaging (EBMA) to combine our five classifiers into one overall prediction. EBMA is a method for pooling across multiple models in order to generate a combined forecast (Montgomery et al., 2012). The combined forecast is generated as a weighted average of the candidate models. The weights are determined based on the prediction accuracy and uniqueness of the candidate models' predictions (Montgomery et al., 2012).

We evaluate the performance and uniqueness of the candidate models using a holdout fold. The size of the holdout fold is one third of the data (500 observations) and has not been used in classifier training, i.e, all models predict outcomes on previously unseen data. Our holdout fold contains at least one observation from each state. The tuning parameter in the EBMA model is the tolerance, which is the minimum improvement of the log-likelihood before the expectation maximization algorithm will stop optimizing.

We choose the optimal tolerance value among the following seven candidate values: $1 \times$

10^{-2}, 5×10^{-3}, 1×10^{-3}, 5×10^{-4}, 1×10^{-4}, 5×10^{-5}, and 1×10^{-5}. We draw 100 bootstrapped samples with an equal number of observations from each state. We estimate the mean squared prediction error on each sample. We average the prediction error of the 100 draws to arrive at seven prediction errors, one for each tolerance value. Finally, we pick the tolerance value with the lowest overall prediction error to generate the EBMA model weights. We experimented with fixing the tolerance at 1×10^{-4} to increase computing speed. Fixing the tolerance parameter led to a performance drop as illustrated in Table 2.

Table 2: Tolerance Optimization vs. Fixing Tolerance

	Tolerance Tuning	Fixed Tolerance
EBMA	0.00191	0.00216
% Reduction in Error over BH Baseline	12.2	7.3

Notes: The table compares the performance of EBMA for a variant where we tune the tolerance parameter with one where we fix it. We tune tolerance using seven candidate values ranging from 1×10^{-2} to 1×10^{-5}. In the fixed tolerance version, we fix the tolerance at 1×10^{-4}. This led to a performance decrease but is computationally more efficient.

7 Item-by-Item Performance

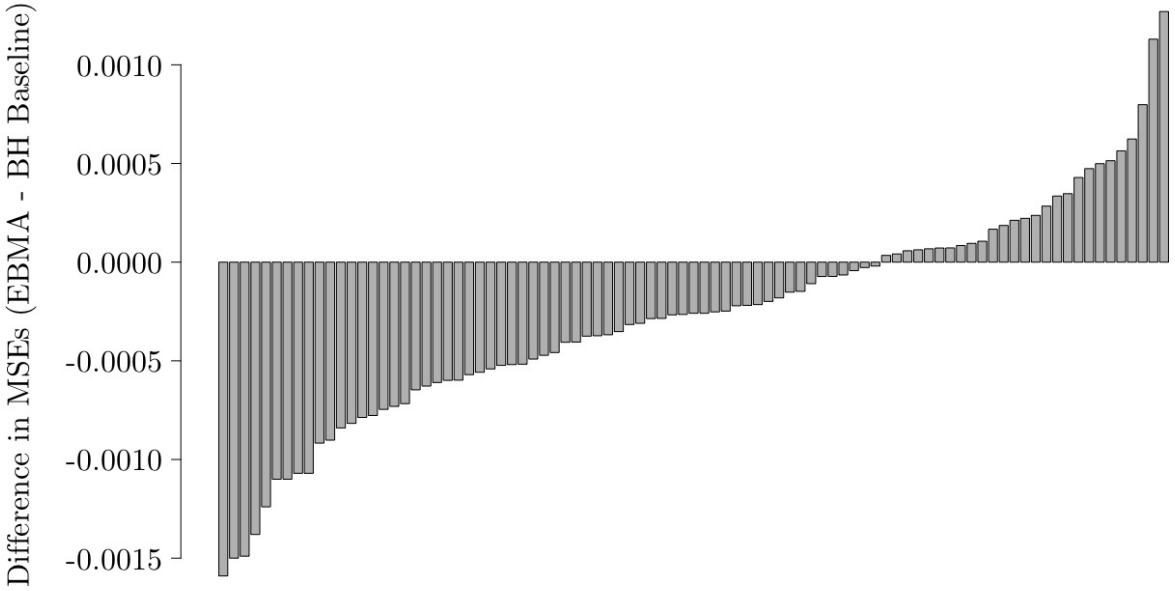
We demonstrated that our EBMA approach improves MrP prediction accuracy on average. EBMA reduces the mean squared prediction error by 12.2% compared to the BH baseline. We analyzed 89 public opinion items. Broken down, item by item, we improve prediction accuracy on 62 items while on 27 items the BH baseline outperforms EBMA, as illustrated in Figure 4.

Our data-driven approach may be outperformed by a theory-informed model on a single item. However, the same is true for the theory-informed model when compared to an MrP model without context-level variables. Overall, potential losses are outweighed by potential gains as illustrated in Figure 4. Furthermore, as we argue in the letter, we consider the comparison to the BH baseline a hard test. Unlike other applications of MrP our 89 survey items are all political issues. Moreover, the data are from the US, for which there is a vast

literature and tradition of public opinion research. US politics may also be more strongly characterized by a single dimension of conflict than politics in other countries. This leads us to expect that models specified by researchers based on their substantive knowledge perform very well.

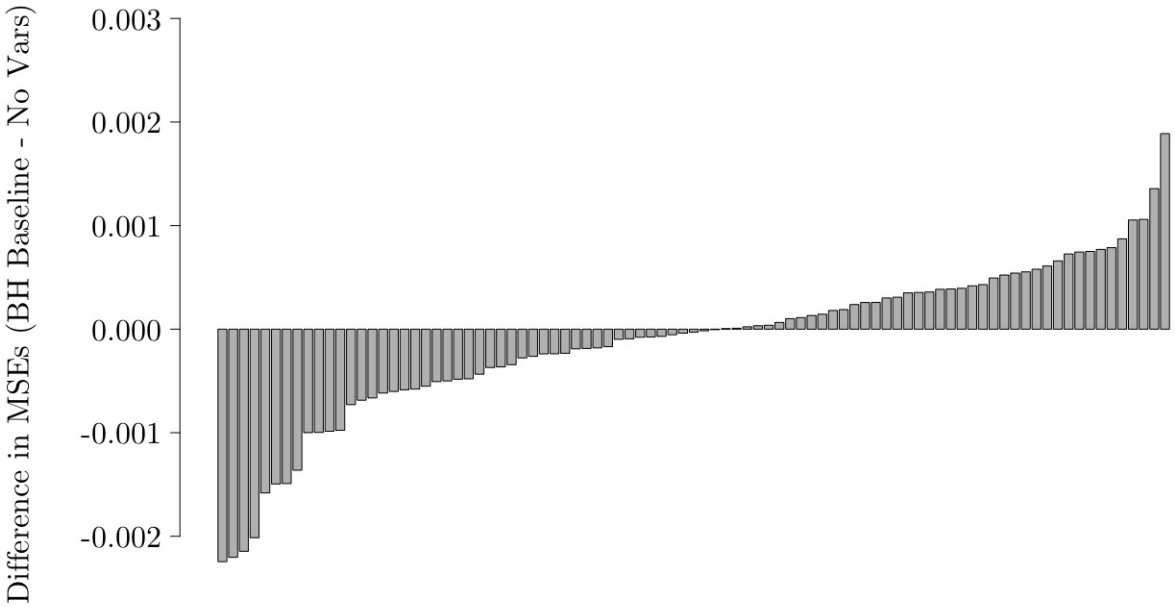
Unfortunately, we cannot provide guidance on the conditions under which our approach is more likely to outperform the theory-informed MrP model. Table 3 lists the survey items ranked by the performance of EBMA compared to the BH baseline.

Figure 4: Item-by-Item Comparison of EBMA vs. BH Baseline



Notes: The barplot illustrates the performance of EBMA compared to the BH baseline for the 89 survey items. Negative differences indicate that EBMA outperforms the BH baseline. Positive values mean that the BH baseline is more accurate than EBMA. EBMA outperforms the BH baseline for 62 items (70%).

Figure 5: Item-by-Item Comparison of BH Baseline vs. No Level-2 Variables



Notes: The barplot illustrates item-by-item performance of the BH baseline compared to a model without context-level variables. The comparison is similar to Figure 4. The BH model is more accurate than a model without context-level variables for 47 items (53%).

Table 3: Survey Items Ranked by EBMA vs. BH Baseline Performance

Rank	Item	Survey	Topic	MSE Difference
1	item 47	cces2008	taxes v. spending (cc420)	-1.84e-03
2	item 66	cces2008	voter eligibility (cc419_3)	-1.57e-03
3	item 11	ann2004	income inequality (ccc41)	-1.50e-03
4	item 4	ann2008	border fence with Mexico (cdd04)	-1.38e-03
5	item 35	ann2000	gays in military cbl01)	-1.10e-03
6	item 48	cces2008	abortion (cc310)	-1.09e-03
7	item 65	cces2008	election day registration (cc419_2)	-1.06e-03
8	item 77	cces2006	late term abortion (v3060)	-1.03e-03
9	item 19	ann2004	homeland security spending (ccd57)	-1.02e-03
10	item 54	cces2008	free trade – NAFTA (cc316h)	-9.18e-04
11	item 76	cces2006	abortion (v3019)	-9.07e-04

12	item 20	ann2004	Patriot Act (ccd67)	-8.32e-04
13	item 69	cces2008	photo id to vote (cc419_6)	-8.26e-04
14	item 1	ann2008	tax rates-a (cbb01)	-8.17e-04
15	item 73	cces2006	capital gains tax rates (v3075)	-8.14e-04
16	item 58	cces2008	military use – oil supply (cc418_1)	-7.39e-04
17	item 59	cces2008	military use – terrorist camps (cc418_2)	-7.28e-04
18	item 55	cces2008	bank bailout (cc316i)	-6.65e-04
19	item 85	cces2006	military use – genocide (v3031)	-6.24e-04
20	item 50	cces2008	gay marriage (cc316f)	-6.14e-04
21	item 14	ann2004	abortion ban (cce01)	-6.11e-04
22	item 36	ann2000	job discrimination (cbl05)	-6.04e-04
23	item 25	ann2000	universal health care for children (cbe08)	-5.95e-04
24	item 53	cces2008	eavesdropping without court order (cc316d)	-5.92e-04
25	item 89	cces2006	Iraq troop withdrawal (v3066)	-5.88e-04
26	item 71	cces2006	social security private accounts (v3024)	-5.61e-04
27	item 57	cces2008	Iraq troop withdrawal (cc316a)	-5.50e-04
28	item 61	cces2008	military use – spread democracy (cc418_4)	-5.13e-04
29	item 84	cces2006	military use – terrorist camps (v3030)	-5.04e-04
30	item 88	cces2006	military use – help UN (v3034)	-4.94e-04
31	item 68	cces2008	automatic registration (cc419_5)	-4.77e-04
32	item 83	cces2006	military use – oil supply (v3029)	-4.64e-04
33	item 41	ann2000	job discrimination (cbm01)	-4.19e-04
34	item 29	ann2000	military spending (cbj07)	-4.09e-04
35	item 49	cces2008	stem cell research (cc316c)	-4.00e-04
36	item 2	ann2008	tax rates-b (cbb01)	-3.75e-04
37	item 18	ann2004	free trade agreements (ccb82)	-3.61e-04
38	item 79	cces2006	illegal immigrant citizenship (v3069)	-3.46e-04
39	item 28	ann2000	invest social security in stock market (cbc05)	-3.15e-04
40	item 8	ann2008	American troops in Iraq (cdb01)	-3.10e-04

41	item 70	cces2006	minimum wage (v2072)	-3.06e-04
42	item 75	cces2006	taxes v. spending v. borrowing (v4044)	-2.93e-04
43	item 12	ann2004	military spending (ccd03)	-2.88e-04
44	item 3	ann2008	immigrant path to citizenship (cdd01)	-2.65e-04
45	item 80	cces2006	environment (v3022)	-2.63e-04
46	item 30	ann2000	tax rates a problem (cbb01)	-2.63e-04
47	item 82	cces2006	free trade – CAFTA (v3078)	-2.47e-04
48	item 38	ann2000	handgun licenses (cbg05)	-2.45e-04
49	item 26	ann2000	poverty a problem (cbp01)	-2.36e-04
50	item 78	cces2006	stem cell funding (v3063)	-2.04e-04
51	item 16	ann2004	school vouchers (ccc39)	-1.93e-04
52	item 56	cces2008	carbon tax (cc422)	-1.92e-04
53	item 64	cces2008	internet absentee voting (cc419_1)	-1.76e-04
54	item 60	cces2008	military use – genocide (cc418_3)	-1.53e-04
55	item 63	cces2008	military use – help UN (cc418_6)	-1.22e-04
56	item 22	ann2004	American troops in Iraq (ccd35)	-8.46e-05
57	item 21	ann2004	rebuilding Iraq spending (ccd34)	-8.23e-05
58	item 40	ann2000	underpunished criminal problem (cbg12)	-7.63e-05
59	item 46	cces2008	assistance for housing crisis (cc316g)	-7.58e-05
60	item 62	cces2008	military use – protect allies (cc418_5)	-7.55e-05
61	item 86	cces2006	military use – spread democracy (v3032)	-1.91e-05
62	item 51	cces2008	jobs v. environment (cc311)	-1.05e-05
63	item 72	cces2006	minimum wage (v3072)	3.05e-05
64	item 9	ann2004	reduce taxes (ccb13)	6.76e-05
65	item 31	ann2000	prescription coverage for seniors (cbe05)	6.97e-05
66	item 44	cces2008	minimum wage (cc316b)	7.32e-05
67	item 52	cces2008	affirmative action (cc313)	8.70e-05
68	item 37	ann2000	school vouchers (cbd02)	9.20e-05
69	item 13	ann2004	invest social security in stock market (ccc32)	9.46e-05

70	item 27	ann2000	social security spending (cbc01)	1.59e-04
71	item 87	cces2006	military use – protect allies (v3033)	1.77e-04
72	item 74	cces2006	taxes v. spending (v4040)	2.21e-04
73	item 45	cces2008	health insurance for children (cc316e)	2.31e-04
74	item 32	ann2000	right to sue HMOs (cbe14)	2.45e-04
75	item 24	ann2000	health care spending for uninsured (cbe02)	2.53e-04
76	item 10	ann2004	aid to schools (ccc40)	2.76e-04
77	item 34	ann2000	death penalty (cbg01)	2.83e-04
78	item 15	ann2004	marriage amendment (cce21)	2.89e-04
79	item 42	cces2008	balanced budget (cc309)	3.18e-04
80	item 67	cces2008	vote by mail (cc419_4)	3.28e-04
81	item 7	ann2008	environment v. economy (cfb01)	3.35e-04
82	item 6	ann2008	same-sex marriage (cec01)	4.99e-04
83	item 81	cces2006	affirmative action (v3027)	5.37e-04
84	item 5	ann2008	abortion availability (cea01)	6.24e-04
85	item 39	ann2000	restrict gun purchases (cbg06)	6.41e-04
86	item 43	cces2008	privatizing social security (cc312)	7.78e-04
87	item 33	ann2000	abortion restrictions (cbf02)	1.01e-03
88	item 23	ann2000	cutting taxes v. strengthening social security (cbb05)	1.02e-03
89	item 17	ann2004	gun control (cce31)	1.14e-03

Notes: ann abbreviates the National Annenberg Election Studies and cces the Cooperative Congressional Election Studies.

8 Algorithm for Gradient Boosting

Our algorithm follows closely Ridgeway (2007, 6) and Hastie et al. (2009, 361):

1. Initialize $f_0(x)$ to the optimal constant model (which is a single terminal node tree), $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.

2. For $t = 1, \ldots, T$:

   (a) For $i = 1, \ldots, N$ compute

$$r_{it} = - \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f=f_{t-1}}.$$

   (b) Fit a tree with a maximum number of D terminal nodes to the targets r_{it} giving terminal regions R_{dt}, $d = 1, \ldots, D_t$.

   (c) For $d = 1, \ldots, D_t$ compute

$$\gamma_{dt} = \arg\min_{\gamma} \sum_{x_i \in R_{dt}} L(y_i, f_{t-1}(x_i) + \gamma).$$

   (d) Update

$$f_t(x) = f_{t-1}(x) + \lambda \sum_{d=1}^{D_t} \gamma_{dt} \mathbb{1}(x \in R_{dt}).$$

3. Output $\hat{f}(x) = f_T(x)$.

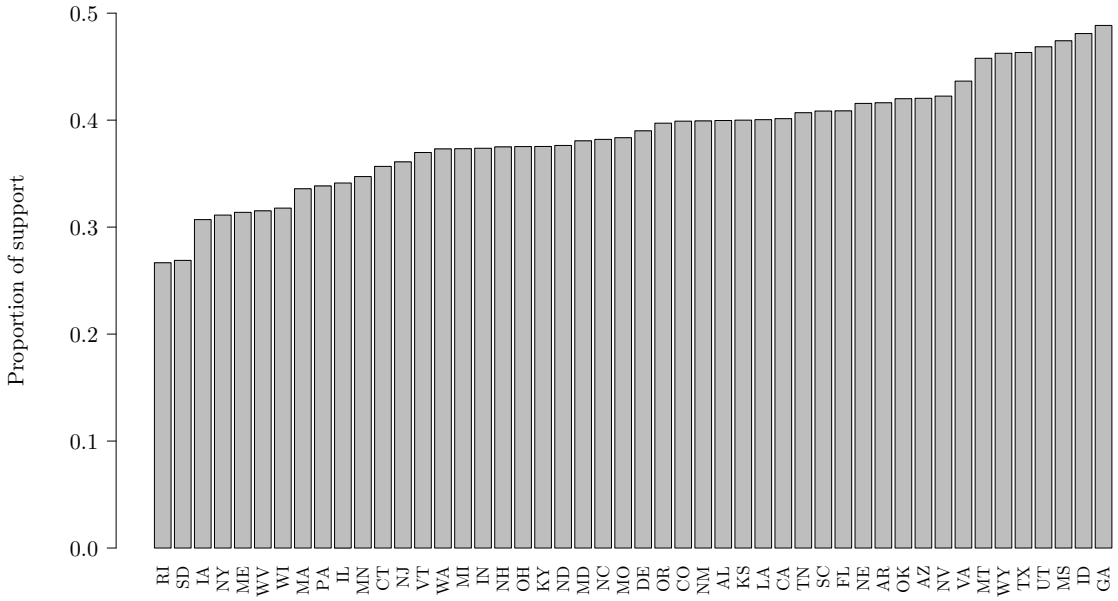
9 Illustrative Example

We estimate state-level opinion based on five classifiers. Subsequently, we combine these five predictions into one overall prediction using ensemble Bayesian model averaging (EBMA). In the following, we demonstrate our approach using survey item 11 on the use of troops to secure the supply of oil as an example. The 2008 Cooperative Congressional Election Studies Survey asked: "Would you approve of the use of U.S. military troops in order to ensure the supply of oil?"

The super survey, which we treat as the population, contains 36,832 individual responses. We aggregate individual responses to the state level and treat these 48 state means as the

true state-level support for the use of the military to secure the supply of oil—we label these estimates "true state support." Figure 6 displays the estimates. We compare our state-level predictions to the "true state support" estimates.

Figure 6: "True State-Level Support" for Use of Military to Secure Oil Supplies



Notes: Estimates of state-level truth are based on "disaggregation." We average the responses of 36,832 individuals by the respective state they are from. The survey item is from the 2008 Cooperate Congressional Election Studies (item id cc418_1).

We draw 1,500 observations from the 36,832 total observations to arrive at a sample size of a typical survey. Our sample contains at least five respondents from each state but is otherwise a random sample. We add six context-level variables to the data: (1) the *share of votes for the Republican candidate in the previous presidential election*; (2) the *percentage of Evangelical Protestant or Mormon respondents*; (3) the *percentage of the population living in urban areas*; (4) the *unemployment rate*; (5) the *share of Hispanics*; (6) the *share of whites*. We normalize all context-level variables and add the six principal components of the context-level variables to the data. Next, the sample is split into two subsets. The first subset contains 1,000 observations (2/3 of the data) and is used in classifier training.

The second subset contains 500 observations (1/3 of the data) and is used to tune ensemble Bayesian model averaging (EBMA).

We perform five-fold cross-validation to tune all five classifiers: The multilevel model with best subset selection, the multilevel model with principal components as context-level variables, the multilevel model with L1 regularization, gradient tree boosting, and support vector machine. We assign states at random to folds. All folds contain roughly the same amount of states. For example, with 48 states, 4 folds contain 10 states and 1 fold contains 8 states. Respondents from the same state are in the same fold. The folds contain roughly the same number of states but not necessarily the same number of respondents. For instance, for item 11, the folds contain 211, 240, 149, 198, and 202 respondents, respectively.

In best subset selection, we fit a multilevel model for each combination of the candidate context-level variables. With six candidate variables, we have $2^6 = 64$ possible variable combinations and with five-fold cross-validation, we need to estimate a total of $64 \times 5 = 320$ models. In lme4 formula notation (Bates et al., 2015), we fit the following models:

```
YES ~ (1 | L1x1) + (1 | L1x2) + (1 | L1x3) + (1 | region/L2.unit) + X,
```

where X is one of the 64 combinations of context-level variables. For each model, we estimate the mean squared error (MSE) on the fold that was not used to fit the model. We average the MSE over the five folds for all 64 models and choose the model with the lowest MSE as our candidate model for the multilevel model with best subset selection.

For the multilevel model with principal components as context-level variables, we fit seven candidate models. The first model does not include context-level variables. We then successively add the principal components to our model. We use cross-validation to determine the best model out of the seven candidates in the same fashion as in the best subset classifier. As in best subset, we use the glmer() function for R to fit the model (Bates et al., 2019).

```
# run pca model
model <- glmer(glmer.models[[m]], data = data.train,
               family = binomial(link = "probit"),
```

```
glmerControl(optimizer = "bobyqa",
                    optCtrl = list(maxfun = 1000000)))
```

In the multilevel model with L1 regularization, we tune the shrinkage parameter λ and use cross-validation to determine the optimal value for λ. We use an exhaustive grid search where we stop increasing λ only if the overall cross-validation MSE has not been decreased in 60 iterations. We successively increase the step size by which we increase λ depending on the current value of λ. Table 4 illustrates the rules of the grid search.

Table 4: Lasso Grid Search

Condition	Step increase in λ
$\lambda < 1$	$\lambda = \lambda + 0.1$
$1 < \lambda < 10$	$\lambda = \lambda + 0.3$
$10 < \lambda < 10,000$	$\lambda = \lambda + 1$
$100 < \lambda < 10,000$	$\lambda = \lambda + 10$

Notes: In the grid search for the optimal value of λ, the stopping rule is 60 iterations without improvement of the cross-validation error.

We use the glmmLasso package for R to fit the models (Groll, 2017). In the code snippet below, note that we already normalized our predictors and therefore do not need to do so again.

```
glmmLasso(fix,
          rnd = list(L1x1 = ~ 1, L1x2 = ~ 1, L1x3 = ~ 1, region = ~ 1,
                     L2.unit = ~ 1),
          data = data.train,
          lambda = lambda,
          family = binomial(link = "probit"),
          switch.NR = FALSE,
          final.re = TRUE,
          control = list(standardize=FALSE))
```

In gradient tree boosting, we tune (1) the learning parameter, (2) the maximum tree depth, and (3) the number of trees to be grown. The learning rate takes the values 0.04, 0.01, 0.008, 0.005, and 0.001. We vary tree depth from 1 to 3. We add trees in increments of 50 to our model

until the cross-validation MSE has not improved for 70 iterations. We have experimented with various grid sizes and have chosen the above as a compromise between computational efficiency and exhaustiveness. The gbm package for R is used for gradient boosting (Ridgeway, 2007). The first tree is grown using the following code:

```
gbm(YES ~ . -L2.unit -state,
    distribution = "bernoulli",
    data = data.train,
    n.trees = 1,
    interaction.depth = depth,
    n.minobsinnode = 5,
    shrinkage = eta[l.rate],
    train.fraction = 1,
    n.cores = 1,
    keep.data = TRUE)
```

We grow additional trees in the following way:

```
gbm.more(gbmodels[[kf]],
         n.new.trees = 50,
         data = NULL,
         weights = NULL,
         offset = NULL,
         verbose = NULL)
```

In the support vector machine classifier, we use the radial kernel and tune γ and the cost parameter c. For γ we search across the following vector: $0.3, 0.5, 0.55, 0.6, 0.65, 0.7, 0.8, 0.9, 1, 2, 3, 4$. The cost parameter takes on the value 1 or 10. As with the previous classifiers we experimented extensively with the grid. Searching a much wider grid did not yield improvements in our example but led to a substantial decrease in computational efficiency. We use the e1071 package to tune the support vector machine (Meyer et al., 2019).

```
svm(
  formula = svm.formula,
  data = data,
  type = "C-classification",
  kernel = "radial",
  scale = FALSE,
  probability = TRUE,
  cost = cost,
  gamma = gamma)
```

The final step is to combine our five classifiers into one overall prediction. We use ensemble Bayesian model averaging (EBMA) implemented in the EBMAforecast package for R to do this (Montgomery et al., 2016). The combined prediction is generated as a weighted average of the candidate models. The weights are determined based on prediction accuracy and the uniqueness of the candidate models' predictions (Montgomery et al., 2012). We tune the tolerance which is the minimum improvement of the log-likelihood before the expectation maximization algorithm will stop optimizing. We use the following values for the tolerance: 1e-2, 5e-3, 1e-3, 5e-4, 1e-4, 5e-5, 1e-5. For each tolerance value, we draw 100 samples from the thus far unused 1/3 of the data (500 observations) that we held out for EBMA. Each of the 100 samples is of roughly the same size as the full EBMA sample (480 observations). We draw the same number of respondents from each state at random with replacement. We determine the number of respondents from each state to include in the sample as $\lfloor \frac{\text{observations in the EBMA data}}{\text{number of states}} \rfloor$, i.e., $\lfloor \frac{500}{48} \rfloor$.

To determine the model weights and optimal tolerance value, we predict outcomes for our sample of 480 bootstrapped observations from each winning individual classifier to generate model weights. We record the MSE of the weighted average prediction in each iteration. Next, we determine which tolerance value led to the lowest overall MSE averaged across the 100 samples. The overall model weights are then the average model weights determined for the 100 samples at the winning tolerance value. The final step is to apply the weights to the post-stratified state-level predictions of the five best models for each of our five classifiers.

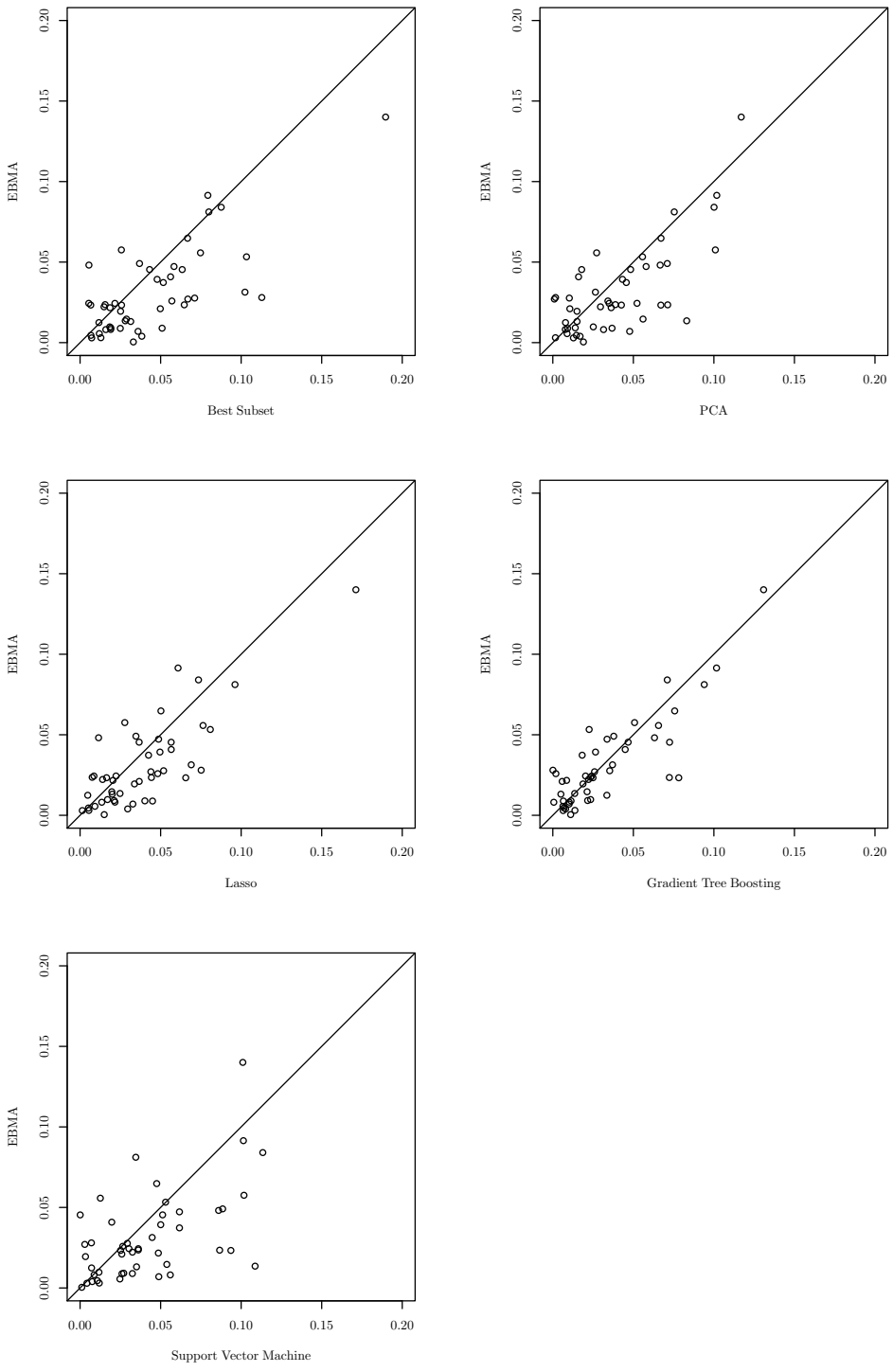
In item 11, respondents were asked: "Would you approve of the use of U.S. military troops in

order to ensure the supply of oil?" We determined the following model weights:

$$\begin{aligned} \text{EBMA} = 0.146312857 \times \text{Best Subset} + 0.169186367 \times \text{PCA} + 0.140194973 \times \text{Lasso} \\ +0.400420411 \times \text{Gradient Tree Boosting} + 0.143885391 \times \text{Support Vector Machine} \end{aligned} \tag{3}$$

As Figure 8 illustrates, by combining the predictions from all classifiers to a weighted average, we reduce the absolute error across all 48 states.

Figure 7: Absolute Error of EMBA compared to all five Classifiers

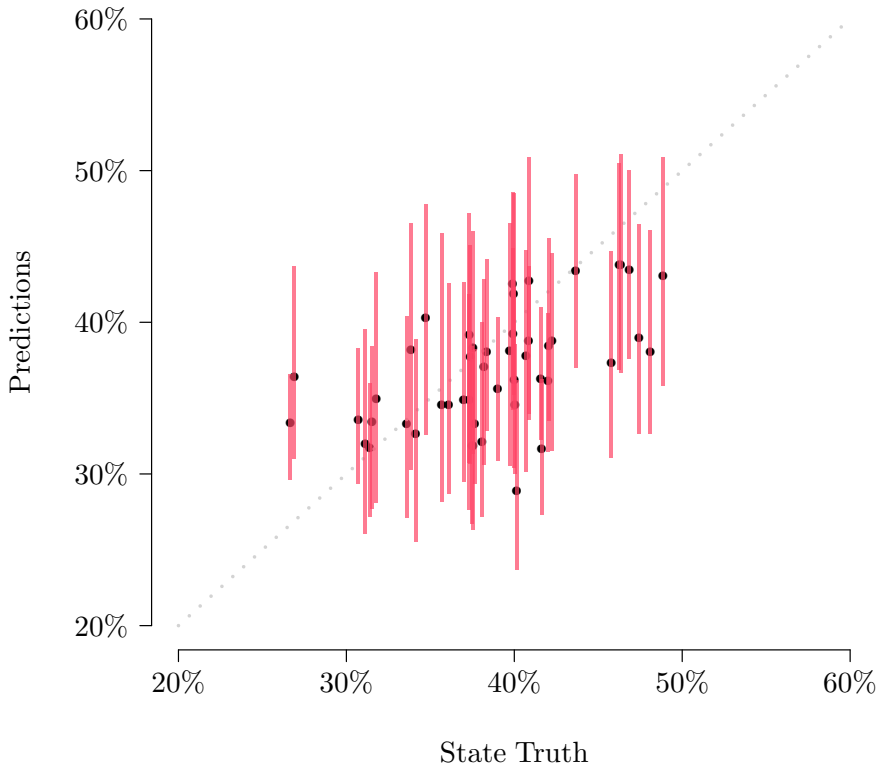


Notes: We compare the EBMA prediction for all 48 states to the predictions of the five classifiers.

10 Uncertainty of State-Level Estimates

In this section we illustrate how uncertainty measures can be derived. It is straightforward to generate uncertainty measures when using use ordinary MrP, such as via sampling from a multivariate normal distribution approximating the posterior coefficient vector (Herron, 1999). In principle, we could try and do something similar here but some of the classifiers pose problems. For example, take support vector machine (SVM): it is not clear how we can incorporate the uncertainty of SVM into a simulation approach. Hence, we opt for bootstrapping as it is flexible enough to generate uncertainty measures for all classifiers and the aggregation (Efron and Tibshirani, 1994).

Figure 8: State Level Predictions and True State Level Opinion



Notes: The segments represent the simulated 95 percent confidence intervals. The segments largely overlap with the diagonal, which means that the true state-level opinion falls within the prediction interval.

To illustrate the approach, we rely again on item 11 from the 2008 Cooperative Congressional

Election Studies Survey that asked "Would you approve of the use of U.S. military troops in order to ensure the supply of oil?" We take a sample of 1,500 observations and rely on resampling with replacement to generate 500 samples with each 1,500 observations. On each of the 500 samples we carry out our estimation approach and save the results. This leads to 500 estimates per state and we can now exploit the variation across these 500 estimates to describe our uncertainty.

As Figure 8 shows, the estimates uncover the true state opinion fairly well. The uncertainty estimates also show that 79% of states are correctly estimated.

11 The autoMrP package

The autoMrP package makes it easy to apply our approach to predict state-level opinion. The package is currently in a beta version hosted on GitHub. In the following, we demonstrate using autoMrP with data from survey item 11 on the use of troops to secure the supply of oil as an example. The following steps illustrate how to install the package.

```
# install devtools package from CRAN
install.packages("devtools")

# install magrittr & import packages from CRAN
install.packages("magrittr")
install.packages("import")

# install auto_MrP package from Github
devtools::install_github("anonymized/autoMrP",
                         auth_token = "anonymized",
                         force = TRUE)

# import the pipe operator form magrittr
import::from(magrittr, "%>%")

# load autoMrP
library(autoMrP)
```

With the package installed and the library loaded, we now load data and run autoMrP. The following code illustrates our call to autoMrP for item 11.

```
auto_mrp_out <- auto_MrP(y = "y",
                         L1.x = c("age", "educ", "gXr"),
                         L2.x = c("pvote", "religcon", "urban", "unemp",
                                  "hispanics", "white"),
                         L2.unit = "stateid",
                         L2.reg = "region",
                         survey = survey_sample,
                         census = census_data,
```

```
                        proportion = "proportion",
                        bin.size = NULL,
                        uncertainty = FALSE,
                        ebma.size = NULL,
                        scale = TRUE,
                        forward.selection = FALSE,
                        k.folds = 5,
                        cv.sampling = "L2 units",
                        loss.unit = "individual",
                        loss.measure = "mse",
                        lasso.lambda.set = data.frame(step_size = c(0.1, 0.3, 1),
                                                      threshold = c(1, 10, 10000)),
                        lasso.iterations.max = 60,
                        gb.L2.unit.include = FALSE,
                        gb.L2.reg.include = FALSE,
                        gb.interaction.set = c(1, 2, 3),
                        gb.shrinkage.set = c(0.04, 0.01, 0.008, 0.005, 0.001),
                        gb.tree.start = 1,
                        gb.tree.increase.set = 50,
                        gb.trees.max.set = 1000,
                        gb.iterations.max = 70,
                        gb.n.minobsinnode = 5,
                        svm.kernel = "radial",
                        svm.error.fun = "MSE",
                        svm.gamma.set = c(0.3, 0.5, 0.55, 0.6, 0.65, 0.7,
                                          0.8, 0.9, 1, 2, 3, 4),
                        svm.cost.set = c(1, 10),
                        ebma.n.draws = 100,
                        ebma.tol.values = c(0.01, 0.005, 0.001,
                                            0.0005, 0.0001, 0.00005, 0.00001),
                        seed = 546213978,
                        verbose = TRUE,
                        best.subset = TRUE,
                        lasso = TRUE,
                        pca = TRUE,
                        gb = TRUE,
                        svm = TRUE,
                        mrp = TRUE,
                        forward.select = TRUE,
                        best.subset.L2.x = c("pvote", "religcon", "urban",
                                   "unemp", "hispanics", "white"),
                        lasso.L2.x = c("pvote", "religcon", "urban",
                                    "unemp", "hispanics", "white"),
                        pca.L2.x = c("pvote", "religcon", "urban",
                                    "unemp", "hispanics", "white"),
                        gb.L2.x  = c("pvote", "religcon", "urban",
                                    "unemp", "hispanics", "white"),
                        svm.L2.x  = c("pvote", "religcon", "urban",
                                    "unemp", "hispanics", "white"),
                        mrp.L2.x = c("pvote", "religcon", "urban",
                                    "unemp", "hispanics", "white"))
                      )
```

If one accepts all the default choices we make, the call reduces to the following lines:

```
auto_mrp_out <- auto_MrP(y = "y",
                         L1.x = c("age", "educ", "gXr"),
                         L2.x = c("pvote", "religcon", "urban", "unemp",
```

```
          "hispanics", "white"),
L2.unit = "stateid",
L2.reg = "region",
survey = survey_sample,
census = census_data,
proportion = "proportion")
```

A list of the arguments for the `auto_MrP()` function and their meaning follows:

- `y` — Outcome variable. A character scalar containing the column name of the outcome variable.
- `L1.x` — Individual-level covariates. A character vector of column names corresponding to the individual-level variables used to predict the outcome variable.
- `L2.x` — Context-level covariates. A character vector of column names corresponding to the context-level variables used to predict the outcome variable.
- `L2.unit` — Geographic unit. A character scalar indicating the column name of the geographic unit at which outcomes should be aggregated.
- `L2.reg` — Geographic region. A character scalar indicating the column name of the geographic region by which geographic units are grouped ('L2.unit' must be nested within 'L2.reg'). Default is NULL.
- `survey` — Survey data. A `data.frame` containing the y and x column names.
- `census` — Census data. A `data.frame` containing the x column names.
- `proportion` — Proportion of state individuals of each ideal type. A character vector containing the column name of the variable in census containing the propotion of individuals of a certain ideal type in a certain state. Default is NULL. Note: Not needed if bin.size is provided.
- `bin.size` — Bin size for ideal types. A character vector indicating the column name of the variable in census containing the bin size for ideal types in a geographic unit. Default is NULL. Note: Not needed if proportion is provided.
- `uncertainty` — Provide uncertainty estimates. A logical argument indicating whether uncertainty is computed or not. Default is FALSE.

- `ebma.size` — Size of EBMA hold-out fold. A rational number in the open unit interval indicating the share of respondents to be contained in the EBMA hold-out fold. Default is $\frac{1}{3}$ of number of observations in `survey` data set.

- `scale` — Whether to normalize context level variables (compute standard scores). A logical argument. Default is TRUE. Note that context level variables should be normalized prior to calling `auto_MrP()` if scale is FALSE.

- `forward.selection` — Apply forward selection for the multilevel model with post-stratification classifier instead of best subset selection. A logical argument indicating whether to apply forward selection instead of best subset selection or not. Default is FALSE. Note: With more than 8 context level variables, forward selection is recommended.

- `k.folds` — Number of folds. An integer-valued scalar indicating the number of folds to be used for cross-validation. Defaults to the value of 5.

- `cv.sampling` — Sampling method. A character-valued scalar indicating whether sampling in the creation of cross-validation folds should be done by respondents or geographic units. Default is by geographic units.

- `loss.unit` — Loss function unit. A character-valued scalar indicating whether the loss should be evaluated at the level of individual respondents or the level of geographic units. Default is at the individual level.

- `loss.measure` — Loss function measure. A character-valued scalar indicating whether the loss should be measured by the mean squared error or the mean absolute error. Default is `MSE`.

- `lasso.lambda.set` — Set of tuning parameters. Lambda is the penalty parameter that controls the shrinkage of fixed effects. Either a numeric vector of lambda values or a data.frame with two columns, the first containing the size by which lambda should increase and the second the upper threshold of the interval of lambdas to which the step size applies. Default is `data.frame(step_size = c(0.1, 0.3, 1), threshold = c(1, 10, 10000))`.

- `lasso.iterations.max` — Stopping rule. A numeric scalar specifying the maximum number of iterations without performance improvement the algorithm runs before stopping. Default is 60.

- `gb.L2.unit.include` — Include `L2.unit` in GB. A logical argument indicating whether `L2.unit` is included in the GB models. Default is FALSE.

- `gb.L2.reg.include` — Include `L2.reg` in GB. A logical argument indicating whether `L2.reg` is included in the GB models. Default is FALSE.

- `gb.interaction.set` — Set of interaction depth values. An integer-valued vector whose values define the maximum depth of each tree. Interaction depth is used to tune the model. Default is `c(1, 2, 3)`.

- `gb.shrinkage.set` — Learning rate. A numeric vector whose values define the learning rate or step-size reduction. Learning rate is used to tune the model. Values between 0.001 and 0.1 usually work, but a smaller learning rate typically requires more trees. Default is `c(0.04, 0.01, 0.008, 0.005, 0.001)`.

- `gb.tree.start` — Initial total number of trees. An integer-valued scalar specifying the initial number of total trees. Default is 1.

- `gb.tree.increase.set` — Increase in total number of trees. Either an integer-valued scalar specifying by how many trees the total number of trees is increased (until the maximum number of trees is reached) or an integer-valued vector of 'length(gb.shrinkage.set)' with each value being associated with a learning rate. Total number of trees is used to tune the model. Default is 50.

- `gb.trees.max.set` — Maximum number of trees. Either an integer-valued scalar specifying the maximum number of trees or an integer-valued vector of `length(gb.shrinkage.set)` with each value being associated with a learning rate and a number of tree increase. Default is 1000.

- `gb.iterations.max` — Stopping rule. A numeric scalar specifying the maximum number of iterations without performance improvement the GB classifier runs before stopping. Default is 70.

- `gb.n.minobsinnode` — Minimum number of observations in the terminal nodes. An integer-valued scalar specifying the minimum number of observations that each terminal node of the trees must contain. Default is 5.

- `svm.kernel` — Kernel for SVM. A character string specifying the kernel to be used for SVM. The possible types are linear, polynomial, radial, and sigmoid. Default is `radial`.
- `svm.error.fun` — Error function for SVM. Default is `MSE`.
- `svm.gamma.set` — Gamma parameter for SVM. This parameter is needed for all kernels except linear. Default is `c(0.3, 0.5, 0.55, 0.6, 0.65, 0.7, 0.8, 0.9, 1, 2, 3, 4)`.
- `svm.cost.set` — Cost parameter for SVM. This parameter specifies the cost of constraints violation. Default is `c(1, 10)`.
- `ebma.n.draws` — The number of bootstrapped samples drawn from the EBMA fold and used for tuning EBMA. Integer value. Default is 100.
- `ebma.tol.values` — Tolerance for improvements in the log-likelihood before the EM algorithm will stop optimization. Numeric vector. Should range at least from 0.01 to 0.001. Default is `c(0.01, 0.005, 0.001, 0.0005, 0.0001, 0.00005, 0.00001)`.
- `seed` — Seed. An integer-valued scalar to control random number generation. If left unspecified (NULL), then seed is set to 546213978.
- `verbose` — Verbose output. A logical argument indicating whether or not verbose output should be printed. Default is TRUE.
- `best.subset` — A logical argument indicating whether best subset is used as one of the classifiers.
- `lasso` — A logical argument indicating whether Lasso is used as one of the classifiers.
- `pca` — A logical argument indicating whether PCA is used as one of the classifiers.
- `gb` — A logical argument indicating whether gradient boosting is used as one of the classifiers.
- `svm` — A logical argument indicating whether support vector machine is used as one of the classifiers.
- `mrp` — A logical argument indicating whether regular MrP is used as one of the classifiers.
- `forward.select` — A logical argument indicating whether forward selection is used for best subset.

- `best.subset.L2.x` – Vector with level 2 variable names to be used for the best subset classifier.
- `lasso.L2.x` – Vector with level 2 variable names to be used for the lasso classifier.
- `pca.L2.x` – Vector with level 2 variable names to be used for the PCA classifier.
- `gb.L2.x` – Vector with level 2 variable names to be used for the GB classifier.
- `svm.L2.x` – Vector with level 2 variable names to be used for the SVM classifier.
- `mrp.L2.x` – Vector with level 2 variable names to be used for the MrP classifier.

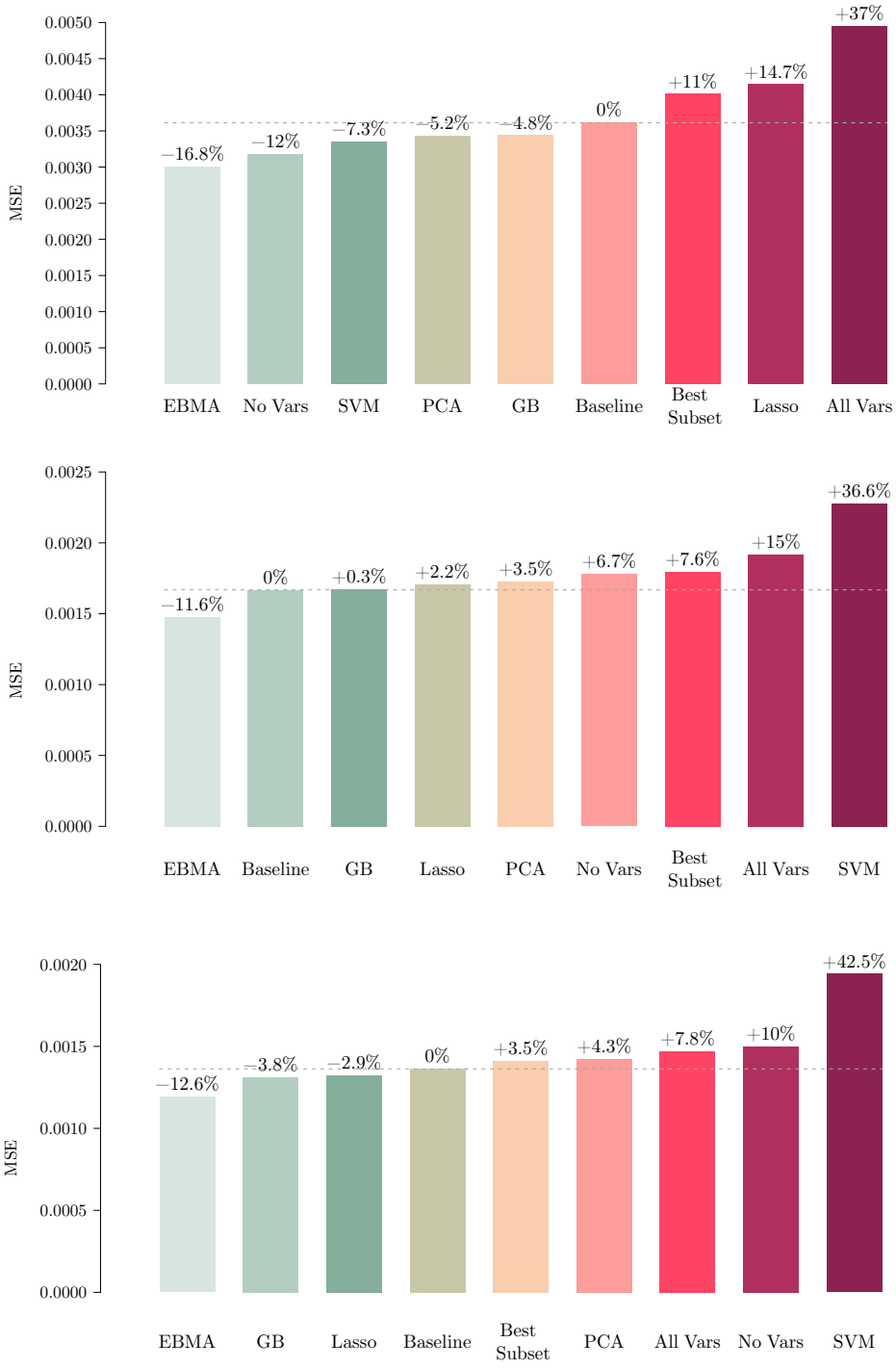
Finally, the package also allows users to select which classifiers should be used. For example, if a user faces a large number of context-level variables that disproportionally affect computation time for the best subset classifier, she can choose to suppress 'best subset'.

12 Sample Size

The results presented so far are all based on a sample size of 1,500. The motivation to do so is that MrP's prime use is to generate subnational estimates based on national surveys that often come with a sample size between 1,000 and 1,500 respondents. Buttice and Highton (2013) rely on a sample size of 1,500 and Lax and Phillips (2009) rely on 1,400 observations in their 5% samples. We have not yet fully explored how the performance of `autoMrP` depends on sample size and we will not be able to do so here. But we can show how the performance varies when we change the sample size to 500, 3,000, or 5,000.

Figure 9 shows the standard performance visualization we have relied on so far, but this time for the exercise based on sample sizes of 500, 3,000, and 5,000 observations. The MSE declines with larger sample size and also the relative performance varies over it. What remains unchanged is that our disciplined approach outperforms the alternatives.

Figure 9: Performance with $N = 500/3,000/5,000$



Note: Average MSE of state-level predictions over 89 survey items. Dashed line indicates MSE of the BH model. Percentage numbers show change in MSE relative to the BH model.

References

Bates, Douglas, Martin Mächler, Benjamin M Bolker, and Steven C Walker. 2015. "Fitting Linear Mixed-Effects Models Using lme4." *Journal of Statistical Software* .

Bates, Douglas, Martin Mächler, Benjamin M Bolker, Steven C Walker, Rune H Bojesen Christensen, Henrik Singmann, Bin Dai, Fabian Scheipl, Gabor Grothendieck, Peter Green, and John Fox. 2019. "Linear Mixed-Effects Models using 'Eigen' and S4.".

Bisbee, James. 2019. "BARP: Improving Mister P Using Bayesian Additive Regression Trees." *American Political Science Review* 113(4): 1060–1065.

Brier, Glenn W. 1950. "Verification of forecasts expressed in terms of probability." *Monthly Weather Review* 78(1): 1–3.

Buttice, Matthew K, and Benjamin Highton. 2013. "How does Multilevel Regression and Poststratification Perform with Conventional National Surveys?" *Political Analysis* 21(4): 449–467.

Efron, Bradley, and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Groll, Andreas. 2017. "Variable Selection for Generalized Linear Mixed Models by L1-Penalized Estimation.".

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning*. 2nd ed. Springer New York.

Herron, Michael C. 1999. "Postestimation Uncertainty in Limited Dependent Variable Models." *Political Analysis* 8(1): 83–98.

Jolliffe, I.T. 2002. *Principal Component Analysis*. 2nd ed. New York: Springer.

Lax, Jeffrey R., and Justin H. Phillips. 2009. "How Should We Estimate Public Opinion in the States?" *American Journal of Political Science* 53(1): 107–121.

Meyer, David, Evgenia Dimitriadou, Kurt Hornik, Andreas Wingessel, Friedrich Leisch, Chih-Chung Chang, and Chih-Chen Lin. 2019. "Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien.".

Miller, Warren E., and Donald W. Stokes. 1963. "Constituency Influence in Congress." *American Political Science Review* 57: 45–46.

Montgomery, Jacob M, and Santiago Olivella. 2018. "Tree-Based Models for Political Science Data." *American Journal of Political Science* 62(3): 729–744.

Montgomery, Jacob M, Florian M Hollenbach, and Michael D Ward. 2012. "Improving predictions using ensemble Bayesian model averaging." *Political Analysis* 20(3): 271–291.

Montgomery, Jacob M, Florian M Hollenbach, and Michael D Ward. 2016. "Ensemble BMA Forecasting.".

Ridgeway, Greg. 2007. "Generalized Boosted Models: A guide to the gbm package.".