autoMrP: Multilevel Models and Post-Stratification (MrP) Combined with Machine Learning in R

Philipp Broniecki
University of Oslo

Lucas Leemann
University of Zürich

Reto Wüest
University of Bergen

Abstract

This introduction to the R package **autoMrP** is a version of Broniecki, Leemann, and Wüest (2021), submitted to the Journal of Statistical Software. A paper on using Machine Learning to improve Multilevel Regression with Post-Stratification is available in Broniecki, Leemann, and Wüest (forthcoming).

In the past twenty years we have witnessed a surge in methodological innovations for small-area estimation. In short, scholars often have nationally representative survey data and would like to create sub-national, e.g., state-level, estimates based on these data. Multilevel regression and post-stratification (MrP) has emerged as the gold standard to achieve this goal (Selb and Munzert 2011). Different improvements to the original MrP model have been proposed and the latest developments combine insights from statistical learning and MrP to provide better estimates. This article introduces the R package **autoMrP**, which allows users to fit traditional MrP models as well as leverage a number of prediction algorithms. This allows building optimized models for generating sub-national estimates from national survey data that outperform those generated by simple MrP models.

Keywords: Multilevel modeling, machine learning, mixed effects, MrP, MrsP, R, survey research.

1. Introduction: Multilevel Regression and Post-Stratification

A frequent problem that arises in various disciplines is that researchers have population-representative data and want to draw inferences for sub-populations from these data. This problem may be encountered by political scientists who wish to estimate sub-national, e.g., state-level, support for public policies based on national survey data (Lax and Phillips 2009a) or epidemiologists estimating state-level prevalence of health outcomes (Downes, Gurrin, English, Pirkis, Currier, Spittal, and Carlin 2018). Common to all of these applications is that limited national data are being used to estimate outcomes at a lower level, such as the state level.

One 'solution' that has been proposed to deal with this problem is disaggregation, which means taking the average value of the outcome for each lower-level unit as the estimate (see, e.g., Erikson, Wright, and McIver 1993). This approach performs badly for small units with few data and is generally not efficient when data are sparse. We can achieve better estimates by modeling the individual-level outcome as a function of individual-level variables and variables at higher levels in a multilevel model and then post-stratify these predictions

(Gelman and Little 1997; Warshaw and Rodden 2012; Lax and Phillips 2009b).

Gelman and Little (1997) propose a procedure, multilevel regression and post-stratification (MrP), that allows researchers to model an individual outcome as a function of individual-level variables as well as variables at higher levels, e.g., the state or regional level. Based on the multilevel model researchers can create predictions for all demographic-geographic ideal types (defined as the combinations of the values of individual-level variables and sub-national units) and then weigh these predictions by the frequency of the ideal types within the sub-national units. Since the publication of this article many studies have demonstrated the superiority of MrP over disaggregation (see, e.g., Warshaw and Rodden 2012; Lax and Phillips 2009b; Leemann and Wasserfallen 2017; Hanretty, Lauderdale, and Vivyan 2018).

Recently, a number of contributions have shown how common models from the statistical learning literature can be fruitfully employed to improve MrP (Bisbee 2019; Ornstein 2020b; Broniecki *et al.* forthcoming). In this article we present our approach proposed in Broniecki *et al.* (forthcoming) and the **autoMrP** package implementing this approach. The **autoMrP** package allows users to harvest the fruits of combining the standard MrP model with statistical learning algorithms to create improved prediction models.

1.1. Standard Multilevel Regression and Post-Stratification (MrP)

MrP relies on national survey data to estimate outcomes, e.g., public opinion about government policy, in sub-national units. MrP is carried out in two steps. First, we fit a multilevel model to the survey data. If we are interested in estimating public support for a specific policy, Y, at the sub-national level, we can fit a multilevel probit model as follows:

$$
\begin{aligned}
Pr(y_i = 1) &= \Phi\left(\beta_0 + \mathbf{x}_{n[i]}^T \boldsymbol{\beta} + \alpha_{j[i]}^{education} + \alpha_{k[i]}^{gender} + \alpha_{m[i]}^{age} + \alpha_{n[i]}^{unit}\right), \qquad (1)\\
\alpha_j^{education} &\sim N(0, \sigma^2_{education}), \text{ for } j = 1, ..., J,\\
\alpha_k^{gender} &\sim N(0, \sigma^2_{gender}), \text{ for } k = 1, ..., K,\\
\alpha_m^{age} &\sim N(0, \sigma^2_{age}), \text{ for } m = 1, ..., M,\\
\alpha_n^{unit} &\sim N(0, \sigma^2_{unit}), \text{ for } n = 1, ..., N.
\end{aligned}
$$

The model includes a set of random effects, here shown for J education groups, K gender groups, M age categories, and N sub-national units. In addition, there is a matrix of predictor variables $\mathbf{X}$ that vary across the sub-national units. Each combination of education, gender, age category, and sub-national unit provides a unique demographic-geographic ideal type. We can now think of society as consisting of these different ideal types.

In a second step, we can calculate the predicted support of each ideal type for the policy based on the estimates in Equation 1. An ideal type's estimated support is denoted by $\hat{\pi}_{jkmn}$. To produce an estimate of the policy support in state n, we calculate the weighted average of $\hat{\pi}_{jkmn}$, where the weights are determined by how prevalent the ideal types are in the population of state n. Since the predictions are not linear in the random effects, we need to know the joint distribution of education, gender, and age in each unit for which we want to obtain an estimate of the public's support for the policy. We calculate the weighted average policy support according to the prevalence of the ideal types in state n as follows

(demographic ideal types in state n are here indexed by g):

$$\hat{\pi}_n = \frac{\sum_{g=1}^{G} \hat{\pi}_{gn} N_{gn}}{\sum_{g=1}^{G} N_{gn}}. \tag{2}$$

Using MrP to generate an estimate for π_n has been shown to outperform other alternatives (e.g., Lax and Phillips 2009b; Warshaw and Rodden 2012; Leemann and Wasserfallen 2020). MrP has been used successfully in the US (e.g., Lax and Phillips 2012), Germany (e.g., Selb and Munzert 2011), the UK (e.g., Claassen and Traunmüller 2018; Hanretty *et al.* 2018), and Switzerland (e.g., Leemann and Wasserfallen 2016) among other countries.

1.2. Improvements to the Standard MrP Approach

Since the introduction of the standard MrP approach authors have suggested various ways of improving its prediction performance. Ghitza and Gelman (2013) propose to include interactions of random effects to provide the model with more flexibility, which in turn provides more precise estimates for different ideal types. Selb and Munzert (2011) show that if there are many sub-national units, such as in the case of legislative elections in Germany, spatially correlated random effects can be included to improve the estimates. Many models are underspecified at the individual level of the response model (see Equation 1) because there is no joint distribution of demographic variables available, which is needed in the post-stratification step. Leemann and Wasserfallen (2017) show how variables for which the joint distribution is unknown can nevertheless be included by creating a synthetic joint distribution.[1]

Finally, there has been some discussion in recent years about leveraging insights from the statistical learning literature for small-area estimation.[2] Several proposals have been made, some of them are published while others are currently working papers (Goplerud, Kuriwaki, Ratkovic, and Tingley 2018; Bisbee 2019; Ornstein 2020b; Broniecki *et al.* forthcoming). At a very basic level, all of these papers are similar in that they recognize that MrP is a prediction task and there is value to a principled selection of features and a more flexible selection of the functional form.

1.3. Statistical Learning and Small-Area Estimation

In a forthcoming paper, we propose an ensemble modeling approach that helps to provide better small-area estimates (Broniecki *et al.* forthcoming). We start by recognizing that information enters the standard MrP model via context-level variables, $\mathbf{X}$, and individual-level variables. The latter are set up as random effects, α, which implies that for the coefficients of the individual-level variables there is a form of regularization, albeit a crude one, built into the model. The contextual information $\mathbf{X}$, however, is just modeled as fixed effects and hence all the known problems of feature selection, over-fitting, and choice of functional form apply to context-level variables. This challenge is compounded by the fact that context-level

[1] An important contribution, albeit not aiming at improving MrP *per se* but rather using it in combination with item-response theory models to scale ideal points, has been put forward by Caughey and Warshaw (2015). This allows Caughey and Warshaw (2016) to scale US states from 1936 to 2014 and provide new estimates of states' liberalism.

[2] See Montgomery and Olivella (2018) for, to our knowledge, the first application of machine learning methods to small-area estimation and also other applications of statistical learning in the political science survey literature (Caughey and Hartman 2017).

variables have been shown by previous research to do the heavy lifting in MrP (Warshaw and Rodden 2012). We propose **autoMrP** as a remedy.

Our approach combines five candidate classifiers—multilevel regression with best subset selection of context-level predictors, multilevel regression with principle components of context-level predictors (PCA), multilevel regression with L1 regularization (Lasso), gradient tree boosting, and support vector machine—via ensemble Bayesian model averaging (EBMA, Montgomery, Hollenbach, and Ward 2012) into one final mega-classifier. In section 3 we provide more details on this approach.

2. Multilevel Regression and Post-Stratification in R

Thus far, analysts have written their own code to perform MrP since there is no widely used R package available. Some years ago, an R package implementing MrP was created but never fully developed and the package was removed from CRAN in 2012 (see `https://cran.r-project.org/src/contrib/Archive/mrp/`).[3] Primers and replication files have been the main source of how insights on implementation were shared. An early example is the (unpublished) primer by Kastellec, Lax, and Phillips (2019), which was updated in 2019 and gathered 50 citations on Google Scholar (verified on July 29th, 2020). Recently, Leemann and Wasserfallen (2020) published a handbook chapter containing a step-by-step account as well as a practical example that readers can replicate (`https://github.com/lleemann/MrP_chapter`). Finally, users interested in a Bayesian implementation can also access the primer by Kennedy and Gabry (2020) on MrP in **rstanarm** (`https://cran.r-project.org/web/packages/rstanarm/vignettes/mrp.html#mrp-with-rstanarm`).

While these primers offer R code chunks there is as yet no package allowing researchers to freely estimate MrP models.[4] The main purpose of **autoMrP** is to allow users to easily apply the *autoMrP* model (Broniecki *et al.* forthcoming), but it also enables the estimation of standard MrP models. This distinguishes it from another recent package, **BARP** (Bisbee 2020), which focuses on a specific model based on MrP and Bayesian additive regression trees.[5]

3. Illustration of autoMrP

In the following, we show how to apply **autoMrP** to a typical survey item and use for our illustration item "CBb01" from the 2008 National Annenberg Election Studies (NAES). We first install the most recent version of the package (currently 0.97) from GitHub.

```
devtools::install_github("retowuest/autoMrP")
```

The 2008 NAES survey item CBb01 states "I'm going to read you some options about federal income taxes. Please tell me which one comes closest to your view on what we should be

[3] Another exception is the R package **swissMrP** (Leemann 2018), but its usability is very limited as it was mostly created for teaching purposes and is only applicable to Swiss data.

[4] See **SRP** for a new package that allows users to estimate MrP models.

[5] **BARP** can also post-stratify other models that are implemented in the **SuperLearner** package (Polley, LeDell, Kennedy, Lendle, and van der Laan 2019).

doing about federal income taxes: (1) Cut taxes; (2) Keep taxes as they are; (3) Raise taxes if necessary; (4) None of these; (998) Don't know; (999) No answer. Category (3) was turned into a 'raise taxes' response, and categories (1) and (2) were combined into a 'do not raise taxes' response. The original survey data contain 50,483 responses favoring or opposing a tax hike. From these data, we include a sample of 1,500 respondents in **autoMrP** to represent the size of a typical national survey. Our sample is drawn at random with the condition that it includes at least five respondents from each state. The name referring to the object containing the survey data is `taxes_survey` and the codebook can be obtained via the help files:

```
library(autoMrP)
?taxes_survey
```

The dependent variable `YES` takes the value 1 if an individual supports raising taxes and 0 otherwise. The individual-level variables `L1x1`, `L1x2`, and `L1x3` represent age, education, and gender-race combinations, respectively, and they are stored as factors. The factor variables `state` and `L2.unit` identify the geographical units, i.e., the US states in our survey. Furthermore, the factor `region` divides US states into the Northeast, Midwest, South, and West.

In addition to the survey data, we require census data to carry out post-stratification. The object name of the census data that accompany the taxation survey data is `taxes_census` and the codebook can be obtained via the help files. The census data are structured such that one row represents a combination of the individual-level variables in a given state. For instance, the first row in `taxes_census` represents white males aged 18–29 without a high school diploma in Alabama. The variable `proportion` identifies the proportions of the demographic ideal types in the state populations—hence, in the first row, the proportion of white males aged 18–29 without a high school degree in the population of Alabama—and it is required to post-stratify estimates.

3.1. Multilevel Regression and Post-Stratification in autoMrP

The standard multilevel regression and post-stratification model can be conveniently estimated with **autoMrP**. Here, we illustrate how to do so using the item on raising taxes.

In our MrP model, we make use of all six context-level variables that are included in the survey data (`L2.x1`, `L2.x2`, `L2.x3`, `L2.x4`, `L2.x5`, and `L2.x6`). These are: (i) *share of votes for the Republican candidate in the previous presidential election*, (ii) *percentage of Evangelical Protestant and Mormon respondents*, (iii) *state percentage of the population living in urban areas*, (iv) *state unemployment rate*, (v) *state share of Hispanics*, and (vi) *state share of whites*. Note that this model over-fits the data and we demonstrate in subsection 3.2 that we can outperform it using the machine learning capabilities of **autoMrP**.

```
mrp_out <- auto_MrP(
  y = "YES",
  L1.x = c("L1x1", "L1x2", "L1x3"),
  L2.x = c("L2.x1", "L2.x2", "L2.x3", "L2.x4", "L2.x5", "L2.x6"),
  L2.unit = "state",
```

```
  L2.reg = "region",
  bin.proportion = "proportion",
  survey = taxes_survey,
  census = taxes_census,
  ebma.size = 0,
  cores = max_cores,
  best.subset = FALSE,
  lasso = FALSE,
  pca = FALSE,
  gb = FALSE,
  svm = FALSE,
  mrp = TRUE
)
```

In **lme4** notation, this function call estimates the following model:

```
YES ~ (1 | L1x1) + (1 | L1x2) + (1 | L1x3) + (1 | region/state) + L2.x1
    + L2.x2 + L2.x3 + L2.x4 + L2.x5 + L2.x6
```

Based on this model, **autoMrP** computes estimates for each ideal type in each state. These estimates are then post-stratified and create the final state-level estimates. We inspect the state-level estimates via the `summary()` function. Note that `summary()` returns the first 10 rows by default. To view all state estimates, we would call `summary(mrp_out, n = 48)`.

```
summary(mrp_out)
```

```
 # estimates of:  mrp

state    median
------  -------
AL       0.1136
AR       0.1126
AZ       0.2167
CA       0.3106
CO       0.2086
CT       0.2369
DE       0.2259
FL       0.1885
GA       0.1324
IA       0.1905
... with 38  more rows
```

In order to estimate only the standard MrP model, we deactivated the machine learning classifiers by setting the classifier arguments `best.subset`, `lasso`, `pca`, `gb`, and `svm` to FALSE. The argument `mrp` controls whether the standard MrP model should be estimated and we hence set it to TRUE. Furthermore, the argument `ebma.size` is the proportion of the sample

that will be used for tuning EBMA. Whenever we use only one classifier, `ebma.size` should be set to 0 to use all available information to fit the classifier.

3.2. Improved Predictions with Machine Learning in autoMrP

In the following, we improve prediction accuracy by estimating state-level opinion using the five machine learning classifiers implemented in **autoMrP** and combining their predictions into an overall prediction via EBMA. We strongly recommend the utilization of parallel processing capabilities to speed up the estimation process. To do so, we first determine how many cores there are available in the system:

```
max_cores <- parallel::detectCores()
```

In the function call to **autoMrP**, we decide to accept the default settings for the tuning parameters.

```
ml_out <- auto_MrP(
  y = "YES",
  L1.x = c("L1x1", "L1x2", "L1x3"),
  L2.x = c("L2.x1", "L2.x2", "L2.x3", "L2.x4", "L2.x5", "L2.x6"),
  L2.unit = "state",
  L2.reg = "region",
  bin.proportion = "proportion",
  survey = taxes_survey,
  census = taxes_census,
  gb.L2.reg = TRUE,
  svm.L2.reg = TRUE,
  cores = max_cores)
```

An **autoMrP** object is a list with three elements that can be accessed via the `$` operator. First, `$ebma` returns the second-level estimates of the EBMA ensemble. Second, `$classifiers` returns the second-level estimates of the individual classifiers. Third, `$weights` returns the weighting of the individual classifiers in the EBMA ensemble. To obtain a summary of the EBMA state-level predictions, we use `summary()` on our **autoMrP** object.

```
summary(ml_out)
```

```
 # EBMA estimates:

state    Median
------  -------
AL       0.1530
AR       0.1387
AZ       0.2097
CA       0.2658
CO       0.1808
CT       0.2120
```

```
DE       0.2120
FL       0.1778
GA       0.1571
IA       0.1792
... with 38  more rows
```

Using `summary()` on an **autoMrP** object returns the estimates of the EBMA ensemble. In Broniecki *et al.* (forthcoming), we demonstrate based on US public opinion data compiled by Buttice and Highton (2013) for 89 survey items that the EBMA estimates outperform those of any individual classifier. Nonetheless, we may wish to inspect the predictions from individual classifiers. We can do so by calling `summary()` on the `$classifiers` element.

```
summary(ML_out$classifiers)
```

```
 # estimates of classifiers:  best_subset, lasso, pca, gb, svm

state    best_subset    lasso      pca       gb      svm
------  ------------  -------  -------  -------  -------
AL            0.1328   0.1173   0.1491   0.1753   0.2042
AR            0.1123   0.1043   0.1189   0.1750   0.2002
AZ            0.2016   0.2343   0.2472   0.1783   0.1746
CA            0.3006   0.3197   0.2956   0.2174   0.1696
CO            0.1768   0.1965   0.1693   0.1819   0.1782
CT            0.2260   0.2341   0.1969   0.2131   0.1843
DE            0.2455   0.2321   0.1764   0.2109   0.1912
FL            0.1790   0.1861   0.1653   0.1784   0.1806
GA            0.1508   0.1310   0.1407   0.1759   0.1985
IA            0.1769   0.1850   0.1651   0.1789   0.1917
... with 38  more rows
```

In addition, we can obtain information on the classifier weights in EBMA by calling `summary()` on the `$weights` element.

```
summary(ML_out$weights)
```

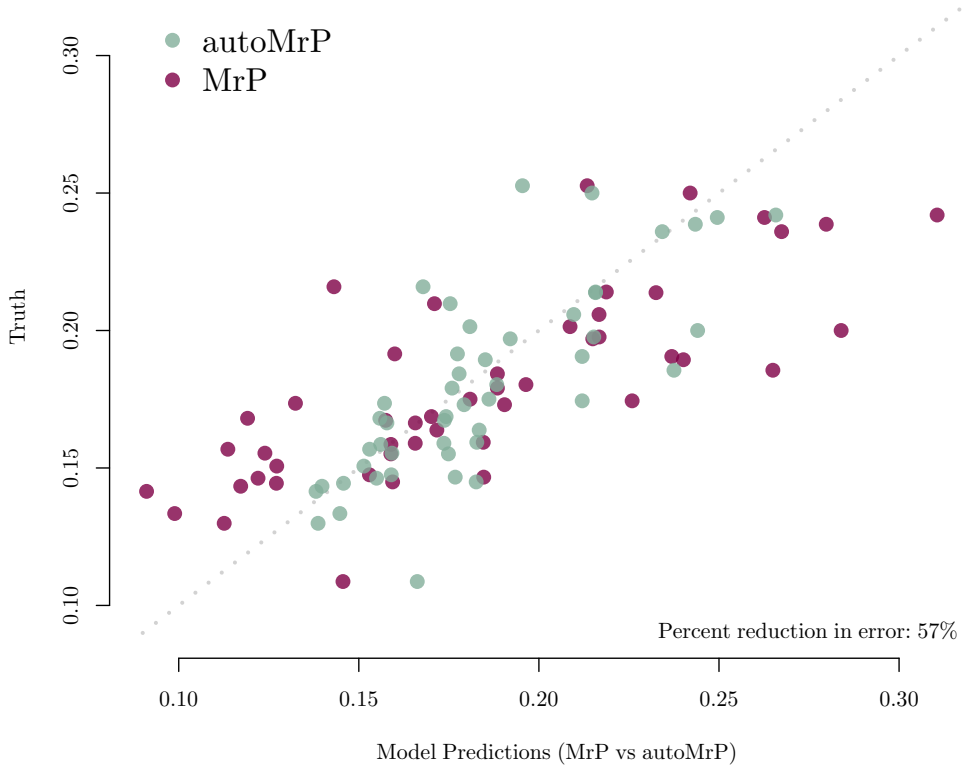
```
 # EBMA classifier weights:

Classifier     Weight
------------  -------
lasso          0.2258
pca            0.2116
best_subset    0.2058
gb             0.1809
svm            0.1759
```

For additional information on the **autoMrP** summary method, refer to the help files using `?summary.autoMrP`.

How does the **autoMrP** machine learning approach fare compared to the standard MrP model? In most real-world applications, researchers do not know the true state (or lower-level unit) estimates. In our example, however, the survey data is a sample from a super-survey with 50,483 respondents. Using disaggregation on the super-survey, we obtain state estimates that should be close to the population truth and, therefore, treat them as the state-level "truth" (see Buttice and Highton 2013; Bisbee 2019; Broniecki *et al.* forthcoming). We use these values as the ground truth in the following comparison, where we illustrate that by using machine learning we obtain more precise estimates than by relying on the standard MrP model.

Figure 1: Comparison of **autoMrP** Predictions with and without Machine Learning



Note: State-level **autoMrP** predictions with machine learning (EBMA) and without (MrP). Both approaches use all context-level information. EBMA reduces the mean squared prediction error by 57%.

The **autoMrP** package can produce estimates based on any combination of the five implemented machine learning classifiers: (i) the multilevel model with best subset selection of context-level variables, (ii) the multilevel model with principal components of context-level variables, (iii) the multilevel model with L1 regularization (lasso), (iv) gradient tree boosting, and (v) support vector machine.

We now describe how **autoMrP** obtains the overall state estimates. In our example data set, context-level variables are not on the same scale. As a first step, **autoMrP** normalizes

context-level variables. Second, it adds the principal components of context-level variables to the survey and census data. Third, it splits the 1,500 observations in the sample into two parts. The first part is used for classifier training while the second is used for tuning EBMA.

All individual classifiers are tuned using cross-validation based on the first part of the observations. The survey respondents in our data are nested within states. The folds are constructed in a way so that all respondents from the same state are assigned to the same fold.[6]

In the next step **autoMrP** post-stratifies the state estimates of the winning model specifications of the individual classifiers using the census data. Finally, overall state-level predictions are generated by averaging the results of the individual classifiers to an ensemble, using the Bayesian model averaging implemented in the R package **EBMAforecast** (Montgomery, Hollenbach, and Ward 2016). In this last step, **autoMrP** tunes the EBMA model based on the second part of the observations.

3.3. Uncertainty Estimates in autoMrP

We implement uncertainty estimates via bootstrapping. Bootstrapping is computationally expensive. On a standard Windows machine with an i5-8400U processor with six cores and six threads, the following example took twelve hours to complete.

```
# Detect the number of cores
max_cores <- parallel::detectCores()

# Run autoMrP with ML & uncertainty
boot_out <- auto_MrP(
  y = "YES",
  L1.x = c("L1x1", "L1x2", "L1x3"),
  L2.x = c("L2.x1", "L2.x2", "L2.x3", "L2.x4", "L2.x5", "L2.x6"),
  L2.unit = "state",
  L2.reg = "region",
  bin.proportion = "proportion",
  survey = taxes_survey,
  census = taxes_census,
  gb.L2.reg = TRUE,
  svm.L2.reg = TRUE,
  cores = max_cores,
  uncertainty = TRUE)
```

In this example, we set the argument `uncertainty = TRUE` to carry out bootstrapping. In addition, the user may specify the argument `boot.iter` to set the number of bootstrap iterations. The argument defaults to 200 iterations. Running **autoMrP** with uncertainty estimates returns an object that contains estimates and weights for each bootstrap iteration. Calling `summary()` on this **autoMrP** object returns EBMA state-level median estimates, the lower and upper 95% confidence bounds as well as the minimum and maximum estimates

[6]The user may override this behavior so that observations are assigned to folds at random. However, We show in the appendix to Broniecki *et al.* (forthcoming) on pages 5–7 why our approach is superior and provide empirical evidence.

by default. The user may specify the confidence level with the `ci.lvl` argument (default is `ci.lvl = 0.95`). Specific classifiers can be inspected by setting the `classifiers` argument to one of the following: `"best_subset"`, `"lasso"`, `"pca"`, `"gb"`, `"svm"`, or `"mrp"` (please refer to the help files via `?summary.autoMrP` for additional information).

```
summary(boot_out)
```

```
 # EBMA estimates:

state      Min.   Lower bound   Median   Upper bound      Max
------  -------  ------------  -------  ------------  -------
AL       0.0593        0.0897   0.1441        0.2118   0.2673
AR       0.0408        0.0488   0.1118        0.1609   0.1965
AZ       0.1298        0.1678   0.2457        0.3618   0.4261
CA       0.1961        0.2312   0.2970        0.3848   0.4091
CO       0.0690        0.0989   0.1618        0.2231   0.2369
CT       0.1113        0.1376   0.2053        0.2769   0.3115
DE       0.1315        0.1419   0.2173        0.3368   0.4088
FL       0.1145        0.1344   0.1835        0.2356   0.2519
GA       0.0529        0.0877   0.1406        0.1877   0.2623
IA       0.0733        0.1076   0.1589        0.2189   0.2543
... with 38  more rows
```

autoMrP has a plot method. Calling the `plot()` function on an `autoMrP` object returns EBMA state-level estimates as well as uncertainty bars if they were estimated via bootstrapping. The confidence level can be set with the `ci.lvl` argument (default is `ci.lvl = 0.95`) and estimates for individual classifiers can be plotted by setting the `algorithm` argument to one of the follwing: `"best_subset"`, `"lasso"`, `"pca"`, `"gb"`, `"svm"`, or `"mrp"` (please refer to the help files via `?plot.autoMrP` for additional information).

In Figure 3 we plot our model predictions and the 95% confidence intervals on the x-axis against the true state-level opinion on the y-axis.[7] The confidence intervals for almost all state-level estimates overlap the diagonal, i.e., the true state-level opinion falls within the 95% confidence intervals for those estimates.

4. Implementation of autoMrP

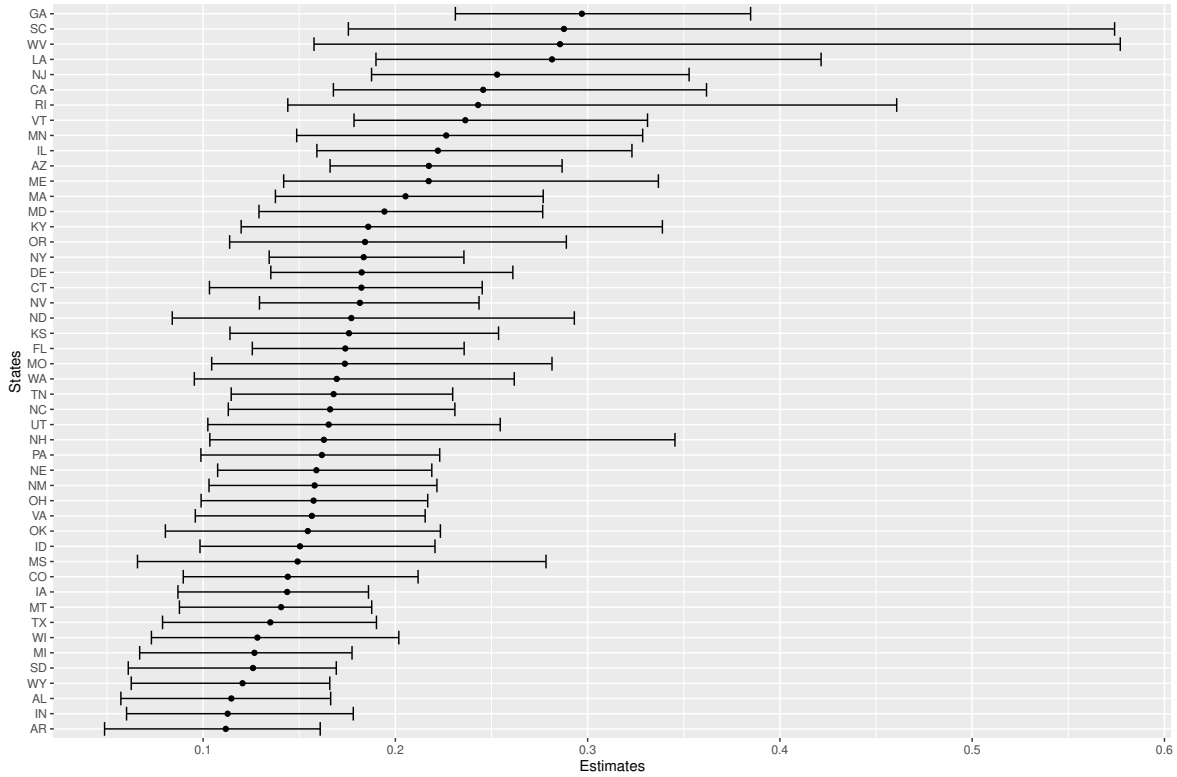
As illustrated by the above example, the `auto_MrP()` function provided by the **autoMrP** package relies on four steps to produce small-area estimates: data preparation, training and tuning of individual classifiers, post-stratification of individual classifiers' predictions, and aggregation of individual classifiers' predictions via EBMA. Each step requires the user to make a number of decisions and pass them to the function via its arguments.

4.1. Data Preparation

The survey and census data sets are passed to the `auto_MrP()` function as `data.frame`s via

[7] True state-level opinion is based on disaggregation of the super survey. Please refer to section 5 for more detail.

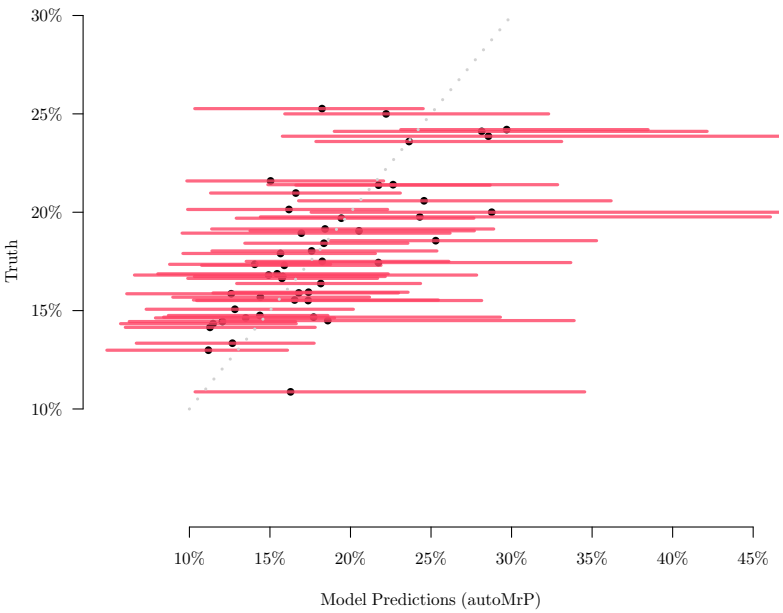
Figure 2: State-Level EBMA Estimates with 95% Confidence Intervals



the arguments `survey` and `census`. The survey `data.frame` must include the individual-level outcome variable specified by function argument `y`, the individual-level covariates specified by argument `L1.x`, the context-level covariates specified by argument `L2.x`, and the context-level unit specified by argument `L2.unit`, at which the outcome variable should be aggregated. Optionally, the survey `data.frame` can also include a variable, specified by `L2.reg`, that captures the hierarchical grouping of the context-level units (e.g., the geographic regions in which the subnational units are nested), a set of variables, specified by `pcs`, representing the principal components of the context-level covariates, and a variable, specified by `folds`, that determines the fold to which each observation in the survey data set is to be allocated (this can be either the EBMA fold or one of the K CV folds).

Setting `folds = NULL` implies that the user does not wish to provide a variable for the partitioning of the survey data into different folds. In that case, the user must specify the arguments `ebma.size`, `k.folds`, and `cv.sampling` in order for `auto_MrP()` to perform the partitioning. Argument `ebma.size` is a number in the (closed) unit interval indicating the proportion of survey respondents to be allocated to the EBMA fold. It defaults to 1/3. Argument `k.folds` is an integer indicating the number of folds, K, to be used in the cross-validation of individual classifiers. Its default value is $K = 5$. Argument `cv.sampling`, finally, specifies whether the K cross-validation folds should be created by sampling context-level units, in which case the user sets `cv.sampling = "L2 units"`, or by sampling respondents, in which case the argument is set to `cv.sampling = "individuals"`. The default setting is `cv.sampling = "L2`

Figure 3: **autoMrP** Estimates with Bootstraped Uncertainty



Note: State-level **autoMrP** point estimates and 95% confidence intervals.

`units"`.

Like the survey `data.frame`, the census `data.frame` must include the variables specified by `L1.x`, `L2.x`, and `L2.unit` and, optionally, can include the variables specified by `L2.reg` and `pcs`. In addition, the census `data.frame` must include either the `bin.proportion`, which is a variable containing the population proportion of individuals by ideal type and context-level unit, or the `bin.size`, which is a variable indicating the population bin size of ideal types by context-level unit.

Setting `pcs = NULL` means that there are no user-provided principal components (PCs) of the context-level variables in the survey and census data sets. In this case, `auto_MrP()` uses the `prcomp()` function from the **stats** package to obtain the PCs of the context-level variables in the survey data. The PCs are then added to the survey and census `data.frame`s. See `?stats::prcomp()` for more information on the calculation of principal components.

By default, `auto_MrP()` normalizes all context-level variables to have a mean of zero and a variance of one. Normalization is performed individually for the survey and census data set. Whether the context-level variables should be normalized is controlled by the logical argument `L2.x.scale`. If the user chooses to set `L2.x.scale = FALSE`, then the context-level covariates should be normalized prior to calling `auto_MrP()`.

4.2. Training and Tuning of Individual Classifiers

The **autoMrP** package allows the user to fit either a single classifier or set of classifiers to the survey data. The predictions of the fitted classifier or classifiers are then post-stratified based on the census data and, if there are multiple classifiers, combined via EBMA. The classifiers

currently supported by **autoMrP** are (i) multilevel regression with best subset selection of context-level covariates (Best Subset), (ii) multilevel regression with best subset selection of principal components of context-level covariates (PCA), (iii) multilevel regression with $L1$ regularization of context-level covariates (Lasso), (iv) gradient boosting (GB), (v) support vector machine (SVM), and (vi) standard multilevel regression (MRP). More classifiers may be added in future versions of the package. The user can choose to rely on any combination of these classifiers. For each individual classifier there is a logical argument that indicates, if set to `TRUE`, that the classifier should be used for prediction of the outcome or, if set to `FALSE`, that it should not be used in the prediction task. These arguments are `best.subset`, `pca`, `lasso`, `gb`, `svm`, and `mrp`. The arguments `best.subset`, `pca`, `lasso`, `gb`, and `svm` default to `TRUE`, while the argument `mrp` defaults to `FALSE`. The user can also control which context-level covariates should be considered by a classifier to predict the outcome. This can be done via the arguments `best.subset.L2.x`, `pca.L2.x`, `lasso.L2.x`, `gb.L2.x`, `svm.L2.x`, and `mrp.L2.x`. If these arguments are set to `NULL`, which is the default option, the respective classifier relies on all available context-level variables (i.e., all variables specified by `L2.x`). For GB and SVM, the user can additionally specify the logical arguments `gb.L2.unit`, `gb.L2.reg`, `svm.L2.unit`, and `svm.L2.reg`. These arguments control whether the classifier should include dummy variables for the context-level units `L2.unit` and the groupings of context-level units `L2.reg`, respectively.

autoMrP draws on a number of existing packages to implement the above classifiers. The multilevel models in Best Subset and PCA are fit using the `glmer()` function from the **lme4** package (Bates, Mächler, Bolker, Walker, Bojesen Christensen, Singmann, Dai, Scheipl, Grothendieck, Green, and Fox 2019). Lasso uses the `glmmLasso()` function from the **glmmLasso** package (Groll 2017). GB relies on the `gbm()` function from the **gbm** package (Ridgeway 2007). And SVM, finally, makes use of the `svm()` function from the **e1071** package (Meyer, Dimitriadou, Hornik, Wingessel, Leisch, Chang, and Lin 2019). Please refer to the respective package reference manual for more information on these functions.

If included in the prediction task, classifiers Best Subset, PCA, Lasso, GB, and SVM are trained and tuned using K-fold cross-validation. This means that for each fold $k \in \{1, \ldots, K\}$, the classifiers are trained on all folds but the kth, which is used to evaluate the classifiers' prediction performance. To evaluate prediction performance, the user must specify the loss function and the unit for which prediction loss is calculated. The loss function is defined by the argument `loss.fun` and the user can choose between the mean squared error, by setting `loss.fun = "MSE"`, the mean absolute error, by setting `loss.fun = "MAE"`, binary cross-entropy loss, by setting `loss.fun = "cross-entropy"`, the mean squared false error (Wang, Liu, Wu, Cao, Meng, and Kennedy 2016), by setting `loss.fun = "msfe"`, the f1 score, by setting `loss.fun = "f1"`, or any combination of those loss functions. The default setting is to use multiple loss functions with the setting `loss.fun = c("MSE", "cross-entropy", "msfe", "f1")`. The unit for calculating prediction loss can be controlled via the argument `loss.unit`. Setting `loss.unit = "individuals"` means that performance loss is evaluated at the level of individual survey respondents and `unit = "L2 units"` means that it is evaluated at the level of context-level units. The default is to evaluate at both levels: `loss.unit = c("individuals", "L2 units")`. With evaluation based on multiple loss functions or both loss units, candidate tuning values are ranked according to their performance in each loss function or loss unit. The candidate tuning parameters with the lowest rank sum are chosen in classifier tuning. Ties are broken according to the order of the search grid.

Classifier tuning requires the user to specify grids of candidate values for the classifiers' tuning parameters. Best Subset and PCA do not have tuning parameters. For Best Subset, `auto_MrP()` simply fits as many models as there are combinations of context-level variables (i.e., 2^k). For PCA, `auto_MrP()` proceeds in a similar way but replaces the context-level variables with their principal components and then estimates $k + 1$ models: the first model empty while the following models successively add the principal components.

The tuning parameter of Lasso is the penalty parameter λ, which controls the shrinkage of the coefficients of context-level variables. Its grid of candidate values is specified by the argument `lasso.lambda`. The argument `lasso.lambda` is a numeric vector of non-negative values.

GB comes with five tuning parameters: the interaction depth, which defines the maximum depth of each tree grown, the learning rate or step-size reduction, the initial tree number fit by GB, the increase in trees fit, and the maximum fit number. The argument `gb.interaction.depth` is a vector of positive integers that are candidate values for the interaction depth. Argument `gb.shrinkage` is a numeric vector whose (positive) values are candidates for the learning rate. Note that a smaller learning rate typically requires a larger number of trees. Argument `gb.n.trees.init` is an integer-valued scalar specifying the initial number of trees to fit by GB. Argument `gb.n.trees.increase` is an integer-valued scalar that specifies the step increase in the number of trees to fit (until the maximum number of trees has been reached). Argument `gb.n.trees.max` is an integer-valued scalar defining the maximum number of trees to fit by GB.

SVM requires the user to choose a kernel. The choice of the kernel is controlled by the argument `svm.kernel`, which can be set to any of the following values: `"linear"`, `"polynomial"`, `"radial"`, or `"sigmoid"`. Depending on which kernel the user has chosen, SVM has one or two tuning parameters: the SVM kernel parameter γ (for all kernel types except the linear one) and a parameter controlling the cost of constraints violation in SVM. Argument `svm.gamma` is a numeric vector whose values are candidates for γ. Argument `svm.cost` is also a numeric vector and its values are the candidates for the SVM cost parameter.

We recommend that users of the package explicitly set all tuning parameters even when they accept **autoMrP** defaults because the default values may change in future package versions.

4.3. Post-Stratification of Individual Classifiers

After training and tuning the classifiers included in the prediction task, `auto_MrP()` selects for each classifier the model with the smallest expected out-of-sample prediction error (as estimated by the cross-validation error). The ideal type-specific predictions of these "winning" models are then post-stratified based on the census data to obtain predictions for subnational units for each classifier.

4.4. Aggregation of Individual Classifiers by EBMA

The final step in `auto_MrP()` is to generate overall predictions for subnational units by averaging the predictions of individual classifiers using EBMA. `auto_MrP()` performs EBMA relying on the `calibrateEnsemble()` function from the **EBMAforecast** package (Montgomery *et al.* 2016). The weights of classifiers in EBMA are determined based on the classifiers' prediction accuracy and the uniqueness of their predictions (Montgomery *et al.* 2012). EBMA can be tuned through the argument `ebma.tol`. Argument `ebma.tol` is a numeric vector that

contains the candidate values for tolerance in the improvement of the log-likelihood before the EM algorithm ends optimization. The default candidate values for the tolerance are 1e-2, 5e-3, 1e-3, 5e-4, 1e-4, 5e-5, 1e-5. Please refer to Montgomery *et al.* (2016) for advice on the specification of candidate values (**autoMrP** searches a wider grid of candidate values than recommended by Montgomery *et al.* 2016). By default, we draw 100 samples from the EBMA fold, which had not been previously used for classifier training. We use a bootstrapping approach where we draw the same number of respondents from each state and end up with 100 samples that are of about the same size as the original EBMA fold. The number of EBMA samples can be controlled via the `ebma.n.draws` argument. Note that EBMA tuning is time consuming and reducing the number of samples to be drawn or reducing the search grid will substantially speed up the prediction task.

5. Comparison of autoMrP with Alternative Approaches

For users interested in using more advanced MrP models that benefit from techniques of the statistical learning literature, there are currently three options: the **SRP** package by Ornstein (2020b), the **BARP** package by Bisbee (2019), and the **autoMrP** package by Broniecki *et al.* (forthcoming). All three packages are similar in that they rest on improved MrP models, but they differ in how exactly they improve upon classic MrP. In this section, we provide a comparison of the performance of the three packages. The comparison is carried out on a standard benchmark data set (Buttice and Highton 2013) and shows that **autoMrP** outperforms the other two alternatives (**SRP** and **BARP**). In the following, we describe the setup to evaluate prediction performance of the three approaches.

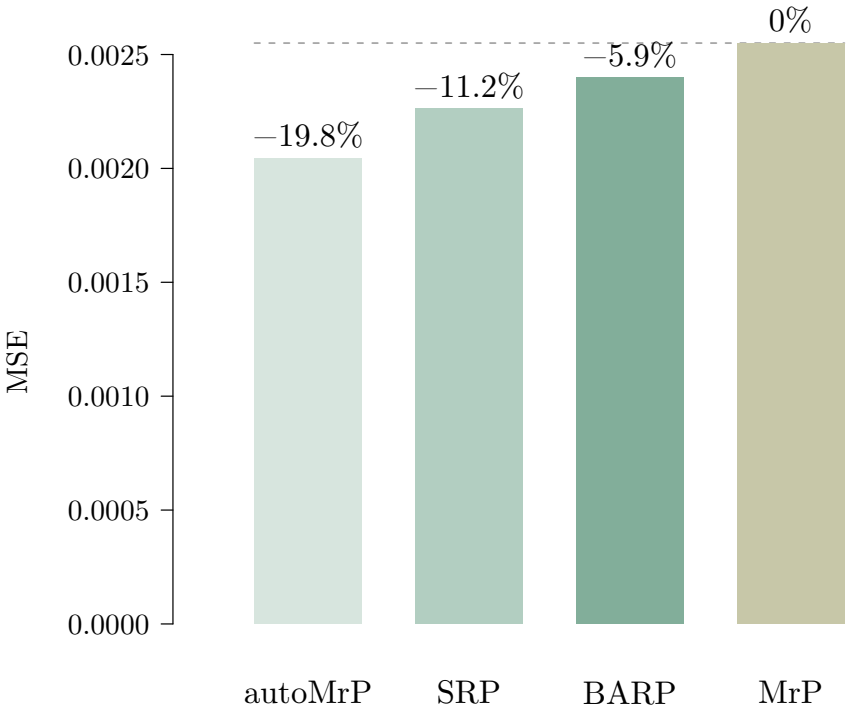
We evaluate the approaches based on real-world public opinion data from five waves of two large US surveys. The data set was compiled by Buttice and Highton (2013) who used it to evaluate the performance of MrP and we follow their assessment approach here (see also Bisbee 2019; Broniecki *et al.* forthcoming). The data set consists of 89 survey items from the National Annenberg Election Studies (2000, 2004, and 2008) and the Cooperative Congressional Election Studies (2006 and 2008).

Each survey item in the data set is binary; coded 1 if the respondent is in favor of the question asked and 0 otherwise. The three individual-level predictors are *age group* (four categories), *education level* (four categories), and *gender-race combination* (six categories). In addition, Buttice and Highton (2013) add two context-level predictors: (i) the state *share of votes for the Republican candidate in the previous presidential election* and (ii) the *percentage of Evangelical Protestant and Mormon respondents.* In addition, we added another four context-level variables: (i) the *state percentage of the population living in urban areas*, (ii) the *state unemployment rate*, (iii) the *state share of Hispanics*, *and* (iv) the *state share of whites* (Broniecki *et al.* forthcoming).

The survey items address issues such as internet absentee voting, gay marriage, taxes versus spending, and a fence at the border with Mexico. Each survey item has at least 25,000 individual responses. Following Buttice and Highton (2013), we treat the state average "yes" response as the population "truth." We then draw a sample of 1,500 respondents for each item to represent the size of a typical national survey. Our samples are random draws where we ensure that we draw at least five respondents from each state.

We use the samples to generate state-level predictions with **autoMrP**, **SRP**, and **BARP** for all

Figure 4: Comparison of the Prediction Performance



Note: Average MSE of state-level predictions over 89 survey items. *MrP* is the standard MrP model with all context-level variables. SRP is the Ornstein (2020a) package and BARP is the Bisbee (2020) package. Percentage numbers: Comparison to standard MrP model.

89 items. All packages make use of the same three individudal-level predictors, the six context-level variables as well as binary state and region variables.[8] In addition, we also compare with a standard MrP model that uses all six context-level variables. The prediction accuracy is evaluated as the mean squared prediction error, comparing the state-level predictions of each package and the MrP model to the state-level "truth."

The results show that **autoMrP** provides the largest gain over the baseline (the standard MrP model) among the three approaches implemented in these packages. There are some potential explanations for this behavior. Comparing the **autoMrP** package to the **BARP** package, we note that **BARP** does not tune parameters and only relies on one classifier (BART). **autoMrP** and **SRP** rely on a set of classifiers and then combine the predictions from these different classifiers with a superlearner.

Comparing **autoMrP** with **SRP**, we see that we rely on EBMA while **SRP** relies on stacking. **autoMrP** engages more intensely in parameter tuning and does not just use default values in most classifiers. In addition, **autoMrP** also relies on a different approach for assigning observations to folds (see p.6 in Broniecki *et al.* forthcoming). Finally, unlike **SRP**, we avoid *double-dipping* when tuning individual classifiers and aggregating the individual classifiers' predictions via the superlearner. Our comparison provides a first evaluation based

[8]The default behavior of **autoMrP** is not to use binary state variables in the gradient tree boosting and support vector machine classifiers because this tended to somewhat reduce prediction accuracy of those classifiers in our tests. Here, however, we override this behavior to ensure that all packages use all available information.

on a very common data set and shows that with these data and under these circumstances **autoMrP** outperforms alternative packages improving upon standard MrP. While some of the highlighted differences indicate superior performance of **autoMrP**, we emphasize that the empirical evaluation is limited to a data set on US public opinion commonly used for assessing MrP models.

6. Summary and Discussion

This article provided an introduction to **autoMrP**, which is a new R package allowing users to estimate classic MrP models as well as **autoMrP** models that rely on statistical learning methods. We first showed how using newer versions of MrP that rely on statistical learning outperform the classic model. We then moved on to a benchmark test between three packages that offer advanced MrP models and showed that **autoMrP** outperforms the two other approaches in terms of prediction accuracy on an often-used test data set.

Going forward, we want to make it possible to estimate some models in **autoMrP** via **rstanarm** rather than to rely on `glmer` models. In addition, we want to make it possible for users to include additional classifiers. The **autoMrP** procedure is in principle open to the inclusion of other classification methods and having this option will provide more flexibility and less dependency on package maintainers.

Computational details

The results in this paper were obtained using R 4.0.2 with the **dplyr** 1.0.2, **foreach** 1.5.0, **doParallel** 1.0.15, **doRNG** 1.8.2, **magittr** 1.5, **lme4** 1.1-23, **glmnet** 4.0-2, **ranger** 0.12.1, **kknn** 1.3.1, **xgboost** 1.2.0.1, **caret** 6.0-86, **SRP** 0.1.1, **BARP** 0.0.1.0001 and **autoMrP** 0.91 packages. R itself and all packages except **SRP**, **BARP**, and **autoMrP** used are available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/. The **SRP** package is available on GitHub at https://github.com/joeornstein/SRP, **BARP** is available on GitHub at https://github.com/jbisbee1/BARP, and **autoMrP** is availalbe on GitHub at https://github.com/retowuest/autoMrP.

Acknowledgments

The names of the authors are listed alphabetically. Philipp Broniecki would like to acknowledge the support of the Business and Local Government Data Research Centre (ES/S007156/1) funded by the Economic and Social Research Council (ESRC) for undertaking this work. Lucas Leemann acknowledges funding from the Swiss National Science Foundation (Grant no. 183120) and the University of Zürich (Einrichtingskredit). Reto Wüest has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (ERC StG 2018 CONSULTATIONEFFECTS, grant agreement No. [804288]).

References

Bates D, Mächler M, Bolker BM, Walker SC, Bojesen Christensen RH, Singmann H, Dai B, Scheipl F, Grothendieck G, Green P, Fox J (2019). "Linear Mixed-Effects Models using 'Eigen' and S4." Https://www.rdocumentation.org/packages/lme4/versions/1.1-21/topics/glmer.

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

Bisbee J (2020). *BARP: Mister P with Bayesian Additive Regression Trees*. New York University. R package version 0.0.1.0001, URL https://github.com/jbisbee1/BARP/.

Broniecki P, Leemann L, Wüest R (2021). "autoMrP: Multilevel Models and Post-Stratification (MrP) Combined with Machine Learning in R."

Broniecki P, Leemann L, Wüest R (forthcoming). "Improving Multilevel Regression with Post-Stratification Through Machine Learning (autoMrP)." *The Journal of Politics*.

Buttice MK, Highton B (2013). "How does multilevel regression and poststratification perform with conventional national surveys?" *Political Analysis*, **21**(4), 449–467.

Caughey D, Hartman E (2017). "Target Selection as Variable Selection: Using the Lasso to Select Auxiliary Vectors for the Construction of Survey Weights." *Unpublished manuscript*, **2017**.

Caughey D, Warshaw C (2015). "Dynamic estimation of latent opinion using a hierarchical group-level IRT model." *Political Analysis*, pp. 197–211.

Caughey D, Warshaw C (2016). "The dynamics of state policy liberalism, 1936–2014." *American Journal of Political Science*, **60**(4), 899–913.

Claassen C, Traunmüller R (2018). "Improving and validating survey estimates of religious demography using Bayesian multilevel models and poststratification." *Sociological methods & research*, p. 0049124118769086.

Downes M, Gurrin LC, English DR, Pirkis J, Currier D, Spittal MJ, Carlin JB (2018). "Multilevel regression and poststratification: A modeling approach to estimating population quantities from highly selected survey samples." *American journal of epidemiology*, **187**(8), 1780–1790.

Erikson RS, Wright GC, McIver JP (1993). *Statehouse democracy: Public opinion and policy in the American states*. Cambridge University Press.

Gelman A, Little TC (1997). "Poststratification into many categories using hierarchical logistic regression." *Survey Research*, **23**, 127–135.

Ghitza Y, Gelman A (2013). "Deep interactions with MRP: Election turnout and voting patterns among small electoral subgroups." *American Journal of Political Science*, **57**(3), 762–776.

Goplerud M, Kuriwaki S, Ratkovic M, Tingley D (2018). "Sparse Multilevel Regression (and Poststratification (sMRP))." *Unpublished manuscript, Harvard University*.

Groll A (2017). "Variable Selection for Generalized Linear Mixed Models by L1-Penalized Estimation." Https://cran.r-project.org/web/packages/glmmLasso/glmmLasso.pdf.

Hanretty C, Lauderdale BE, Vivyan N (2018). "Comparing strategies for estimating constituency opinion from national survey samples." *Political Science Research and Methods*, **6**(3), 571–591.

Kastellec JP, Lax JR, Phillips J (2019). *Working Papers. "Estimating State Public Opinion with Multi-level Regression and Poststratification using R"*. Working Paper Princeton University. URL https://scholar.princeton.edu/jkastellec/publications/mrp_primer.

Kennedy L, Gabry J (2020). "MRP with rstanarm." Rstanarm Vignettes.

Lax JR, Phillips JH (2009a). "Gay Rights in the States: Public Opinion and Policy Responsiveness." *American Political Science Review*, **103**(3), 367–386.

Lax JR, Phillips JH (2009b). "How Should We Estimate Public Opinion in the States?" *American Journal of Political Science*, **53**(1), 107–121.

Lax JR, Phillips JH (2012). "The Democratic Deficit in States." *American Journal of Political Science*, **56**(1), 148–166.

Leemann L (2018). *swissMrP: Multilevel Regression with Post-Stratification (MrP) for Switzerland*. University of Zurich. R package version 0.62, URL https://cran.r-project.org/web/packages/swissMrP/index.html.

Leemann L, Wasserfallen F (2016). "The democratic effect of direct democracy." *American Political Science Review*, **110**(4), 750–762.

Leemann L, Wasserfallen F (2017). "Extending the Use and Prediction Precision of Subnational Public Opinion Estimation." *American Journal of Political Science*, **61**(4), 1003–1022.

Leemann L, Wasserfallen F (2020). "Measuring Attitudes–Multilevel Modeling with Post-Stratification (MrP)." In L Curini, R Franzese (eds.), *The SAGE Handbook of Research Methods in Political Science and International Relations*, pp. 371–384. SAGE Publications Limited.

Meyer D, Dimitriadou E, Hornik K, Wingessel A, Leisch F, Chang CC, Lin CC (2019). "Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien." Https://cran.r-project.org/web/packages/e1071/e1071.pdf.

Montgomery JM, Hollenbach FM, Ward MD (2012). "Improving predictions using ensemble Bayesian model averaging." *Political Analysis*, **20**(3), 271–291.

Montgomery JM, Hollenbach FM, Ward MD (2016). *EBMAforecast: Ensemble BMA Forecasting*. R package version 0.52, URL https://CRAN.R-project.org/package=EBMAforecast.

Montgomery JM, Olivella S (2018). "Tree-Based Models for Political Science Data." *American Journal of Political Science*, **62**(3), 729–744.

Ornstein JT (2020a). *SRP: R Package for Stacked Regression and Poststratification (SRP)*. University of Georgia. R package version 0.1.1, URL https://github.com/joeornstein/SRP.

Ornstein JT (2020b). "Stacked Regression and Poststratification." *Political Analysis*, **28**(2), 293–301.

Polley E, LeDell E, Kennedy C, Lendle S, van der Laan M (2019). *SuperLearner: Super Learner Prediction*. R package version 2.0-26, URL https://cran.r-project.org/package=SuperLearner.

Ridgeway G (2007). "Generalized Boosted Models: A guide to the gbm package." Http://www.saedsayad.com/docs/gbm2.pdf.

Selb P, Munzert S (2011). "Estimating constituency preferences from sparse survey data using auxiliary geographic information." *Political Analysis*, **19**(4), 455–470.

Wang S, Liu W, Wu J, Cao L, Meng Q, Kennedy PJ (2016). "Training deep neural networks on imbalanced data sets." In *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 4368–4374. doi:10.1109/IJCNN.2016.7727770.

Warshaw C, Rodden J (2012). "How Should We Measure District-Level Public Opinion on Individual Issues?" *Journal of Politics*, **74**(1), 203–219.

.

Affiliation:

Philipp Broniecki
Department of Political Science
University of Oslo
Eilert Sundts hus
Moltke Moesvei 31
0851 Oslo, Norway
E-mail: philipp.broniecki@stv.uio.no
URL: https://philippbroniecki.com/

Lucas Leemann
Department of Political Science
University of Zürich
Affolternstrasse 56
8050 Zürich, Switzerland
E-mail: leemann@ipz.uzh.ch
URL: https://lucasleemann.ch

Reto Wüest
Department of Comparative Politics
University of Bergen

Christies gate 15
5007 Bergen, Norway
E-mail: reto.wueest@uib.no
URL: https://retowuest.net/