mrpkit: A reproducible, opinionated, workflow for multilevel regression with post-stratification in R*

Lauren Kennedy† Jonah Gabry‡ Dewi Amaliah§ Mitzi Morris¶
Rohan Alexander‖

13 December 2021

Abstract

The R package mrpkit implements a structured workflow for Multilevel Regression and Poststratification (MRP). The package helps appropriately set-up the survey and post-stratification datasets and then establish a relationship between different variables in these datasets. A substantial amount of the work to do this is specified by the methods in the package, saving time, enhancing reproducibility, and reducing the risk of coding errors. The package natively supports multilevel binomial and Bernoulli models fit with lme4 and Stan (via brms and rstanarm) and allows for the use of custom modeling functions. Summaries and visualizations of the resulting post-stratified estimates can be conducted using the methods in the package. This implementation provides a detailed workflow when conducting MRP that will be useful for researchers and in teaching. Using mrpkit proactively addresses many common issues and makes it possible to conduct MRP in an efficient way.

Keywords: multilevel regression with post-stratification; R; reproducibility; workflow; statistics.

1 Introduction

Multilevel regression with post-stratification (MRP) is a statistical approach in which surveys are related to each other using a statistical model. This is useful because known biases in one survey can be adjusted by another in a statistically reasonable way. This enables better use of non-representative surveys, consideration of additional information, and for the propagation of uncertainty. However, it can be difficult to use MRP due to this need to relate two different datasets. This seemingly straight-forward requirement imposes considerable hassle in practice and tends to substantially complicate reproducibility. The R (R Core Team 2020) package mrpkit allows users to conduct multilevel regression with post-stratification in an opinionated and specific way. The package provides a workflow whereby users specify two datasets (a sample dataset and a post-stratification dataset), establish a correspondence between them, and then conduct analysis.

The process of conducting MRP involves considering some survey outcome as being due to a combination of many different cells that are typically demographic or geographic in nature, estimating this relationship using a multilevel regression model and then aggregating cell-level estimates to match the population (Wang et

*[*]**Feedback:** Comments on the 13 December 2021 version of this paper are welcome at: lauren.kennedy729@gmail.com. **Funding statement:** We have no direct financial disclosures. **Conflicts statement:** We have no conflicts to disclose. **Code and data accessibility statement:** Code and data are available at: https://github.com/lauken13/mrpkit. We are unable to share direct access to the datasets used in the 2020 US Presidential example in this paper, however they are freely available and we provide detailed instructions for how to obtain them. **Author contributions:** Kennedy had the original idea. Gabry, Kennedy, and Morris made architectural and other high-level decisions. Gabry and Kennedy wrote most of the code and tests in the package. Amaliah wrote the vignette and contributed tests. Alexander wrote the first draft of this paper. All authors contributed to writing the paper, and approved the final version.*

[†] Monash University, lauren.kennedy729@gmail.com

[‡] Columbia University

[§] Monash University

[¶] Columbia University

[‖] University of Toronto

al. 2015). The survey of interest would typically be a regular survey, such as a political poll of, say, 1,000 respondents, but it could also be a larger survey, such as the American National Election Studies, or similar. The post-stratification dataset would typically be a larger survey, such as, in the case of the US, the American Community Surveys, or a census. The correspondences need to be established for each possible answer for each question, for instance, the age-groups used in both surveys need to be homogenized. The model would typically be a Bayesian hierarchical model, but in principle any variety could be used.

At its core, MRP is a mapping between a survey object and a population object. It is from this mapping that the power of MRP exists, but it can be difficult to establish this mapping in a reproducible way that fits into a typical applied statistics workflow. Making this implementation easier and more reproducible is important because interest in, and the use of, MRP has considerably increased. It can be difficult even for those experienced with MRP to ensure there are no mistakes in this mapping. The `mrpkit` package creates a well-defined workflow for MRP that represents an important contribution for enhancing the reproducibility of MRP analysis.

Our package complements existing packages such as `survey` (Lumley 2020, 2004, 2010), and `DeclareDesign` (Blair et al. 2019). The `survey` package is well-established and allows a user to specify a survey design, such as clustered, stratified, etc, generate summary statistics, and conduct sub-population analysis. The `srvyr` package (Freedman Ellis and Schneider 2021) is a variant of `survey` that adds a syntax similar to `dplyr` (Wickham et al. 2021). The `DeclareDesign` package (Blair et al. 2019) is a newer package that is based around a grammar of research design aspects including: data generation, creating treatment and control groups, and sampling. In general, these packages are focused on designing, implementing, and simulating from, surveys. Ours is focused on what is needed for MRP, and we draw on their packages where possible.

In terms of packages to specifically help with MRP, the most common alternative to this package at the moment is for users to do all aspects of the MRP workflow themselves. This can be difficult. In particular, preparing and matching different levels between surveys can be time consuming and potentially introduce undocumented errors. The package `ccesMRPprep` (Kuriwaki 2020) is focused on MRP using the Cooperative Congressional Election Study (CCES). More recently the package `tidymrp` (Kroese 2021) is being developed, which focuses on conducting analysis in a tidy way (Wickham et al. 2019).

The main alternative approach is for MRP to be conducted on a case-specific basis. This means that a researcher who is interested in MRP estimates writes the code needed to obtain, clean, prepare, analyze, and ultimately interpret the estimates. Again, while there is nothing inherently wrong with this approach, successfully conducting MRP requires dealing with many small issues. Each of these is small but getting them wrong can have large effects that are potentially unnoticed on the estimates. It is also easy to overlook steps, or to not appropriately document them. `mrpkit` is prescriptive about the steps that must be taken, and ensures that all aspects are documented.

The remainder of this paper is structured as follows: Section 2 discusses the core use case and workflow for MRP as implemented in `mrpkit`. Section 3 discusses some of the implementation issues and technical notes related to the decisions that were made. Section 4 provides an example of the package in use. Finally, Section 5 provides a summary discussion, some cautions, and weaknesses, as well as notes about next steps.

2 Use cases and workflow

The MRP workflow implemented by `mrpkit` is summarized in Figure 1. In this section, these steps are expanded on. At each of them there are a variety of decisions that must be made. The `mrpkit` package makes each of these explicit.

We see a variety of use cases for `mrpkit` including:

- A professor wants to introduce MRP to a class that might not be especially experienced with R, nor have time to develop the necessary skills, instead wanting to focus on the estimates and applications.
- A researcher with domain or survey expertise, but who does not have the resources to ensure that the code has been appropriately tested.
- A researcher who is experienced with MRP needs to run many different models on the same dataset.

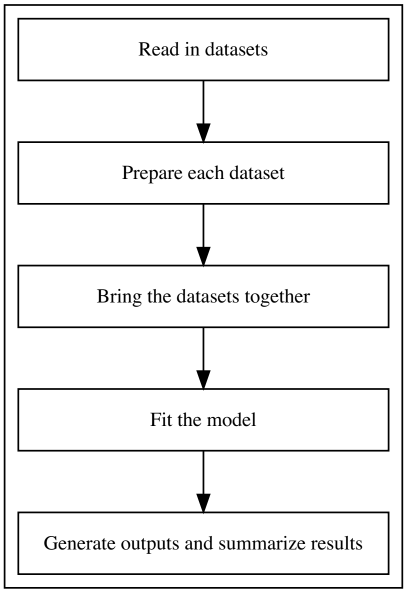

Read in datasets
Prepare each dataset
Bring the datasets together
Fit the model
Generate outputs and summarize results

Figure 1: Illustration of a mrpkit workflow

The main contribution of mrpkit across these situations is making steps explicit that are often implicit.

2.1 Set-up, load package, and read in the datasets

At the moment the mrpkit package has not been submitted to CRAN as it is being finalized. For now it can be installed from GitHub.

```
devtools::install_github("lauken13/mrpkit")
```

```
library(mrpkit)
```

Having installed and loaded the package, a user first needs to get their survey and post-stratification dataset into R. This means ensuring that it is in a rectangular format such as a CSV and then reading it in as a dataframe. The mrpkit package does not explicitly contain methods for this, instead a user could use readr::read_csv() (Wickham and Hester 2021) or haven::read_dta() (Wickham and Miller 2021).

A user will need two datasets. At this point, a user should be keenly aware of the class of the variables as this becomes important later in the workflow. This is because the mrpkit package will convert all variables to factor variables.

In this example we will use simulated datasets that are available as part of the package for the survey dataset, mrpkit::shape_survey, and the post-stratification dataset, mrpkit::approx_voters_popn.

```
survey <- shape_survey
poststrat <- approx_voters_popn

head(survey)
```

```
##     age gender     vote_for   highest_educ   state   y        wt
## 1 56-65   male           BP   some college State D  no  79.90290
## 2 46-55   male Circle Party 4-year college State B yes  89.12213
## 3 36-45   male Circle Party     associates State D  no  92.90569
## 4 18-25   male    Box Party    high school State B yes  80.78384
## 5 56-65 female           BP    high school State A yes  89.85756
## 6 56-65 female Circle Party 4-year college State B yes 117.04200
```

```
head(poststrat)
```

```
##   age_group gender vote_pref        wt       education state
## 1       66+      m        BP  6.007461 4-years college     A
## 2     56-65      f        CP  8.465851     high school     A
## 3     56-65      m        CP  4.098878 4-years college     C
## 4       66+      m        BP  3.883250 4-years college     C
## 5       66+      f        CP 16.179562    some college     B
## 6     18-35      m        BP  3.851837     high school     D
```

2.2 Preparing each dataset

Typically, MRP requires at least two datasets—usually a survey dataset and a post-stratification dataset. A user specifies these using the SurveyData class. SurveyData objects are used to represent both the survey and the post-stratification datasets, along with their metadata. The SurveyData$new method transforms a regular dataframe into a SurveyData object. A user needs to create SurveyData objects for both the survey and post-stratification datasets separately using the SurveyData$new method and assigning a name to each.

As part of creating these objects a user will typically specify the dataset (brought into R in the earlier step), the survey questions, the possible responses, and any weights and design formula that are applicable. A user does this separately for the survey and the post-stratification dataset, so there will be two objects created. Having created this object, a user can use the SurveyData$print method to display the created SurveyData object in the console.

Building on the datasets that were loaded earlier, we will show the steps to create SurveyData objects using the SurveyData$new method. This method takes the following arguments:

1. data: The name of the dataset, i.e., the survey or post-stratification data that will be transformed to a SurveyData object. By default, this method uses all of the factor, character, and binary variables in the dataset to create questions and responses, unless a user specifies the list of questions and responses that are of interest.
2. questions: The actual questions asked in the survey form or questionnaire for each of the columns in the dataset For instance, a column called 'age' may contain the results of a question 'Please identify your age-group.'
3. responses: All possible responses for each of the questions. For instance, possible responses in the 'age' column could be: '18-25,' '26-35,' '36-45,' '46-55,' '56-65,' '66-75,' and '76-90.'
4. (Optional) weights: If the weight is specified as a column in the dataset, then this argument should be specified as a string.
5. (Optional) design: If there is a design formula for the survey, then this can be specified using the notation of the survey package (Lumley 2020, 2004, 2010).

Here we create a SurveyData object for the survey dataset.

```
# Create a SurveyData object for the survey dataset
box_pref <- SurveyData$new(
  data = survey,
  questions = list(
    age = "Please identify your age group",
    gender = "Please select your gender",
    vote_for = "Which party did you vote for in the 2018 election?",
    highest_educ = "Please identify your completed highest education",
    state = "Which State do you live in?",
    y = "If today is election day, will you vote for the Box Party?"
  ),
  responses = list(
    age = levels(survey$age),
    gender = levels(survey$gender),
    vote_for = levels(survey$vote_for),
    highest_educ = levels(survey$highest_educ),
    state = levels(survey$state),
    y = c("no", "yes")
  ),
  weights = "wt",
  design = list(ids =  ~ 1)
)

# Display it
box_pref$print()
```

```
## Survey with 500 observations, 6 questions
## Independent Sampling design (with replacement)
##
## Column label: age
## Question: Please identify your age group
## Allowed answers: 18-25, 26-35, 36-45, 46-55, 56-65, 66-75, 76-90
##
## Column label: gender
## Question: Please select your gender
## Allowed answers: male, female, nonbinary
##
## Column label: highest_educ
## Question: Please identify your completed highest education
## Allowed answers: no high school, high school, some college, associates, 4-year college, post-graduate
##
## Column label: state
## Question: Which State do you live in?
## Allowed answers: State A, State B, State C, State D, State E
##
## Column label: vote_for
## Question: Which party did you vote for in the 2018 election?
## Allowed answers: Box Party, BP, Circle Party, CP
##
## Column label: y
## Question: If today is election day, will you vote for the Box Party?
## Allowed answers: no, yes
```

Now we create a SurveyData object for the post-stratification dataset. In this case, the weight is already specified in the dataset. Often in MRP, the post-stratification dataset is from a large survey, for example, the American Community Survey (ACS) or Demographic and Health Survey (DHS). It is possible to have an already summarized post-stratification dataset. If this is the case, then the weight would be the size of each cell. Alternatively (as in this example), if the entire individual-level population is given, then this argument should be omitted, and it would be automatically specified as 1.

```
poststrat_obj <- SurveyData$new(
  data = poststrat,
  questions = c(
    age_group = "Which age group are you?",
    gender = "Which gender are you identified?",
    vote_pref = "Which party do you prefer to vote?",
    education = "What is the highest grade or level of school you have completed",
    state = "Please identify the state where you live in"
  ),
  responses = list(
    age_group = levels(poststrat$age_group),
    gender = levels(poststrat$gender),
    vote_pref = levels(poststrat$vote_pref),
    education = levels(poststrat$education),
    state = levels(poststrat$state)
  ),
  weights = "wt",
  design = list(ids =  ~ 1)
)

# print it
poststrat_obj$print()
```

```
## Survey with 5000 observations, 5 questions
## Independent Sampling design (with replacement)
##
## Column label: age_group
## Question: Which age group are you?
## Allowed answers: 18-35, 36-55, 56-65, 66+
##
## Column label: education
## Question: What is the highest grade or level of school you have completed
## Allowed answers: no high school, high school, some college, 4-years college, post-grad
##
## Column label: gender
## Question: Which gender are you identified?
## Allowed answers: m, f, nb
##
## Column label: state
## Question: Please identify the state where you live in
## Allowed answers: A, B, C, D, E
##
## Column label: vote_pref
## Question: Which party do you prefer to vote?
## Allowed answers: BP, CP
```

A user can access the properties of the `SurveyData` object using various methods. For instance, the weights, `SurveyData$weights()`, the design, `SurveyData$design()`, the questions, `SurveyData$questions()`, the number of questions, `SurveyData$n_questions()`, and the responses, `SurveyData$responses()`.

2.3 Aligning the datasets

Having established these two datasets separately, a user needs to begin to create a relationship between them. The first step is to relate each question, and their answers using `QuestionMap$new`. This needs to be done for every question that will be used and typically requires specifying a name that will be used for the underlying construct, the column names in each of the two `SurveyData` objects, and a mapping between the values in each. This is a deliberately specific task requiring the user to think about and manually link the values.

In the example, the survey (`box_pref`) and the post-stratification (`poststrat_obj`) objects have different column labels and response levels. Specifically, the column label for age in `box_pref` is 'age,' whereas, in `poststrat_obj`, the column label is 'age_group.' This variable also has different levels; `box_pref` has seven levels of age, whereas `poststrat_obj` has only four levels of age. We do this alignment using the `QuestionMap$new` method.

This method takes three arguments:

1. `name`: The name of the underlying construct. This will be used in the modeling stage.
2. `col_names`: A character vector of the column label in the survey and post-stratification objects. The column name of the survey should always be the first element, followed by the column name of the post-stratification object. This order is not interchangeable.
3. `values_map`: The list of the mapped values between the survey and post-stratification object. If there is a meaningful ordering over the values, they should be sorted over that order, either descending or ascending.

Here we define how age-groups, political preferences, gender, education, and states should come together. While this is tedious it is worth noting that in general much of this code is likely to be re-useable across different projects.

```r
# Create QuestionMap$object for the question related to age
q_age <- QuestionMap$new(
  name = "age",
  col_names = c("age", "age_group"),
  values_map = list(
    "18-25" = "18-35",
    "26-35" = "18-35",
    "36-45" = "36-55",
    "46-55" = "36-55",
    "56-65" = "56-65",
    "66-75" = "66+",
    "76-90" = "66+"
  )
)

# Create QuestionMap$object for the question related to party preference
q_party_pref <- QuestionMap$new(
  name = "party_pref",
  col_names = c("vote_for", "vote_pref"),
  values_map = list(
    "Box Party" = "BP",
    "BP" = "BP",
    "Circle Party" = "CP",
    "CP" = "CP"
  )
)

# Create QuestionMap$object for the question related to gender
q_gender <- QuestionMap$new(
  name = "gender",
  col_names = c("gender", "gender"),
  values_map = data.frame(
    "male" = "m",
    "female" = "f",
    "nonbinary" = "nb"
  )
)

# Create QuestionMap$object for the question related to education
q_educ <- QuestionMap$new(
  name = "highest_education",
  col_names = c("highest_educ", "education"),
  values_map = list(
    "no high school" = "no high school",
    "high school" = "high school",
    "some college" = "some college",
    "associates" = "some college",
    "4-year college" = "4-years college",
    "post-graduate" = "post-grad"
  )
)

# Create QuestionMap$object for the question related to state
q_state <- QuestionMap$new(
  name = "state",
  col_names = c("state", "state"),
  values_map = list(
    "State A" = "A",
    "State B" = "B",
    "State C" = "C",
    "State D" = "D",
    "State E" = "E"
  )
)
```

The maps between each question print nicely to allow checking.

```
q_age
```

```
## --------------
## age
## age = age_group
## --------------
## 18-25 = 18-35
## 26-35 = 18-35
## 36-45 = 36-55
## 46-55 = 36-55
## 56-65 = 56-65
## 66-75 = 66+
## 76-90 = 66+
```

2.4 Bringing the datasets together

The relationship between the two datasets is finalized by bringing all of this together to create a map between them using `SurveyMap$new` and assigning this to an object. This typically requires specifying:

1) the name of the `SurveyData` object related to the sample, in this case ‘box_pref’;
2) the name of the `SurveyData` object related to the post-stratification dataset, in this case ‘poststrat_obj’; and
3) the question mappings that should be included, which specifies the matched labels and values provided by `QuestionMap` objects.

To initialize a new `SurveyMap` object a user can use the `SurveyMap$new` method. This takes `SurveyData` and `QuestionMap` objects as argument.

```
# Create a new SurveyMap object
ex_map <- SurveyMap$new(sample = box_pref, population = poststrat_obj,
                        q_age, q_educ, q_gender, q_party_pref
                        )
```

```
## Warning: Variable(s) 'state' are available in the population but won't be used
## in the model
```

```
print(ex_map)
```

```
## ==============
## age = age_group
## --------------
## 18-25 = 18-35
## 26-35 = 18-35
## 36-45 = 36-55
## 46-55 = 36-55
## 56-65 = 56-65
## 66-75 = 66+
## 76-90 = 66+
## ==============
## highest_educ = education
## --------------
## no high school = no high school
## high school = high school
## some college = some college
## associates = some college
## 4-year college = 4-years college
## post-graduate = post-grad
## ==============
## gender = gender
## --------------
## male = m
## female = f
## nonbinary = nb
## ==============
## vote_for = vote_pref
## --------------
## Box Party = BP
## BP = BP
## Circle Party = CP
## CP = CP
```

Questions can also be added using `SurveyMap$add`. For instance, a user could add ‘state’ to the `QuestionMap` object.

```
# Add questions incrementally
ex_map$add(q_state)

print(ex_map)
```

```
## ==============
## age = age_group
## --------------
## 18-25 = 18-35
## 26-35 = 18-35
## 36-45 = 36-55
## 46-55 = 36-55
## 56-65 = 56-65
## 66-75 = 66+
## 76-90 = 66+
## ==============
## highest_educ = education
## --------------
## no high school = no high school
## high school = high school
## some college = some college
## associates = some college
## 4-year college = 4-years college
## post-graduate = post-grad
## ==============
## gender = gender
## --------------
## male = m
## female = f
## nonbinary = nb
## ==============
## vote_for = vote_pref
## --------------
## Box Party = BP
## BP = BP
## Circle Party = CP
## CP = CP
## ==============
## state = state
## --------------
## State A = A
## State B = B
## State C = C
## State D = D
## State E = E
```

At this point the user has mapped the questions of interest between the survey and the post-stratification datasets. Variables could be excluded using the `SurveyMap$delete` method, or replaced using the `SurveyMap$replace` method.

2.5 Finalizing the mapping and tabulation

The process of bringing the datasets together is finalized with the method `SurveyMap$mapping`. This creates new sample and post-stratification datasets with unified variable names, and levels. It prepares the mapped data for model fitting.

```
ex_map$mapping()
```

If a user wants to include all the variables in the post-stratification step, then a user can use the `SurveyMap$tabulate` method without any arguments. It is also possible to specify only certain variables in the post-stratification step, by including those variables as arguments.

```
ex_map$tabulate()
```

2.6 Modelling the survey data

The user can then fit a model to the survey dataset using `SurveyMap$fit` and naming it to create an object. Built-in options include: `rstanarm::stan_glmer()` and `rstanarm::stan_glm()` (Goodrich et al. 2020), `lme4::glmer()` (Bates et al. 2015), and `brms::brm()` (Bürkner 2017). A user could also specify their own function. For that purpose, there is a `data` argument that accepts a dataframe. In this example, we fit three models using `rstanarm::stan_glmer()`, `lme4::glmer()`, and `brms:brm()` with `age`, `gender`, and `education` as the predictors.

```r
# Example of using rstanarm::stan_glmer
fit1 <- ex_map$fit(
  fun = rstanarm::stan_glmer,
  formula = y ~ (1 | age) + (1 | gender) + (1 | state),
  family = "binomial",
  refresh = 100,
  cores = 2)

# Example of using lme4::glmer
fit2 <- ex_map$fit(
  fun = lme4::glmer,
  formula = y ~ (1 | age) + (1 | gender) + (1 | state),
  family = "binomial")
```

```r
# Example using brms::brm
fit3 <- ex_map$fit(
  fun = brms::brm,
  formula = y ~ (1 | age) + (1 | gender) + (1 | state),
  family = "bernoulli",
  refresh = 100,
  cores = 2)
```

The resulting object is not the fitted model object created by the modeling function, but rather a `SurveyFit` object that contains the fitted model and provides useful methods for working with it.

2.7 Using that model

At this stage, a user has fitted model objects, i.e., `fit1` and `fit2`. These objects have been automatically stored as `SurveyFit` objects. `mrpkit` has various built-in methods that enable a user to perform common tasks using the `SurveyFit` object. These include:

1) Generating the predicted probabilities for the post-stratification cells using `SurveyFit$population_predict`.
2) Considering these post-stratified estimates at a more-aggregated level, for instance state or province level, using `SurveyFit$aggregate`.
3) Make a table that summarizes the posterior mean and standard deviation, along with comparisons with raw estimates and weighted estimates, using the `summary` method.
4) Visualize the estimates using the `plot` method.

2.7.1 Get the probability of the outcome for each poststratification cell

After creating the `SurveyFit` object, the user can generate the predicted probability of the outcome of each post-stratification cell using the `SurveyFit$population_predict` method. This returns a matrix with rows that correspond to the columns of post-stratification data and columns that correspond to the posterior samples.

Furthermore, if the model fitting is done with one of the built-in functions (`rstanarm::stan_glmer`, `lme4::glmer`, `brms::brm`), then this method does not take any arguments. However, if you used a customized function in the model fitting, then this method takes multiple arguments, including a custom prediction function (see ?SurveyFit for details). In this example, we will show how to use this method for the fit1 object created using rstanarm::stan_glmer.

```
# predict the probability of voting for the Box Party using the fit1 model
poststrat_est_fit1 <- fit1$population_predict()
```

2.7.2 Aggregate the probability of the outcome to a higher level of estimate

The next step is to generate the population estimate or group estimate using the SurveyFit$aggregate method. This method takes two arguments, namely the post-stratification estimate and the variable whose level to which the estimated value would be aggregated to. In this example, we want to aggregate the estimated value by the level of age and state. If the variable is not specified, then this method would generate a population estimate.

```
# aggregate the predicted value by age
age_estimation <- fit1$aggregate(poststrat_est_fit1, by = "age")

# aggregate the predicted value by state
state_estimation <- fit1$aggregate(poststrat_est_fit1, by = "state")

# generate the population estimate
popn_estimation <- fit1$aggregate(poststrat_est_fit1)
```

This method then will return a dataframe with the variable levels and probability of the outcome, y. In this example, age_estimation represents the age group and the probability of voting for the Box Party for each age group.

```
head(age_estimation)
```

```
##             age     value
## 1 18-25 + 26-35 0.6551063
## 2 36-45 + 46-55 0.7395437
## 3         56-65 0.7820977
## 4 66-75 + 76-90 0.7252167
## 5 18-25 + 26-35 0.6213959
## 6 36-45 + 46-55 0.7973898
```

```
# get the mean and sd for each age group
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
age_estimation %>%
  group_by(age) %>%
  summarize(mean = mean(value), sd = sd(value))
```

```
## # A tibble: 4 x 3
##   age            mean     sd
##   <fct>         <dbl>  <dbl>
## 1 18-25 + 26-35 0.691 0.0617
## 2 36-45 + 46-55 0.758 0.0313
## 3 56-65         0.751 0.0282
## 4 66-75 + 76-90 0.721 0.0331
```

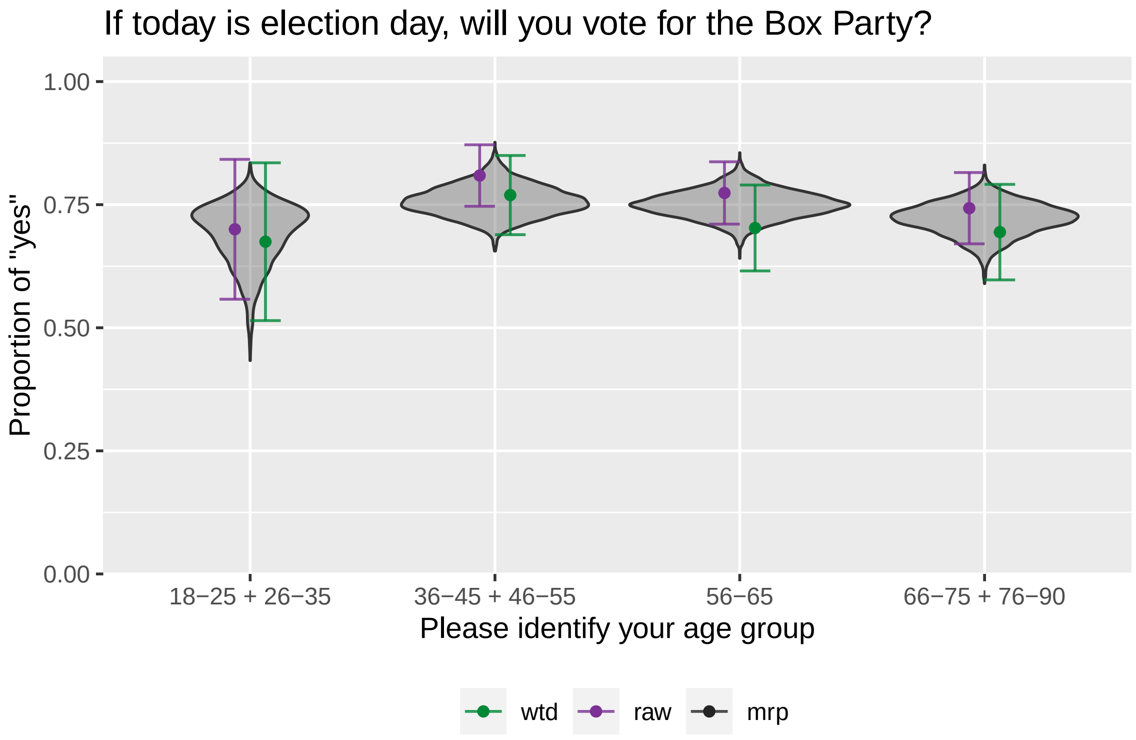
2.7.3 Summarize and compare

The output of aggregate is a posterior predictive distribution for the population (or sub-populations) of interest. This can be used with the plot function to visualize. However, we also might wish to make a table summary of the posterior mean and standard deviation, along with comparisons with raw estimates and weighted estimates. We can use the `summary` method do to this. This method takes the aggregated dataframe and produces a dataframe of mean and standard deviation estimates for each group (if in the aggregated frame) or overall in the population. The standard deviation for the weights is created by wrapping the survey package using the original design specified, and the raw standard deviation is $\sqrt{\frac{pq}{n}}$. This is called automatically in the plot method.

2.7.4 Visualize

Once we have the aggregated estimates, we can visualize them using the `plot` method. This method takes the aggregated dataframe and additional_stats option (the default is wtd and raw estimates) as its arguments. It generates a violin plot of the aggregated estimate by the level of a certain variable. When additional statistics are specified (options are raw, wtd and mrp; more than one can be specified as a vector), these are added as points with error bars representing the 95 per cent confidence interval.
In this example, we display the estimated plot for 'age' and 'state' and a density plot for the population estimates.

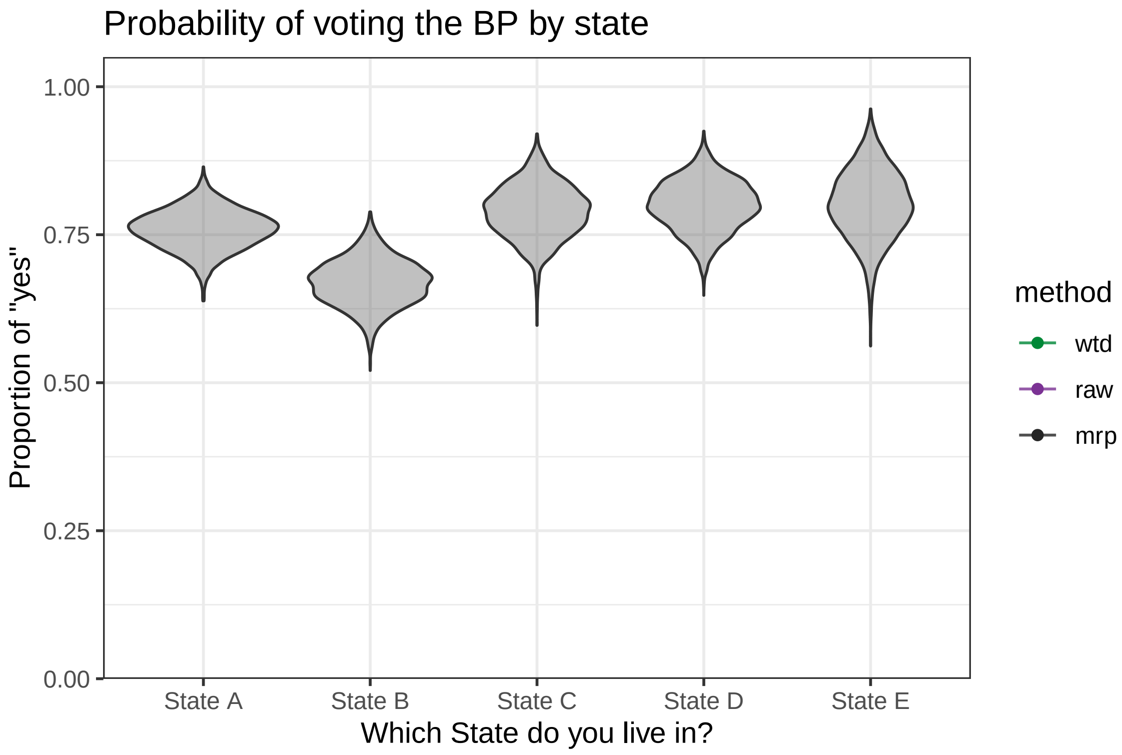
```
plot_age <- fit1$plot(age_estimation)
plot_age
```



The plot above shows the distribution of people who vote for the Box Party for each age group. The plot implies that people who are aged 36-55 are most likely to vote for the BP.

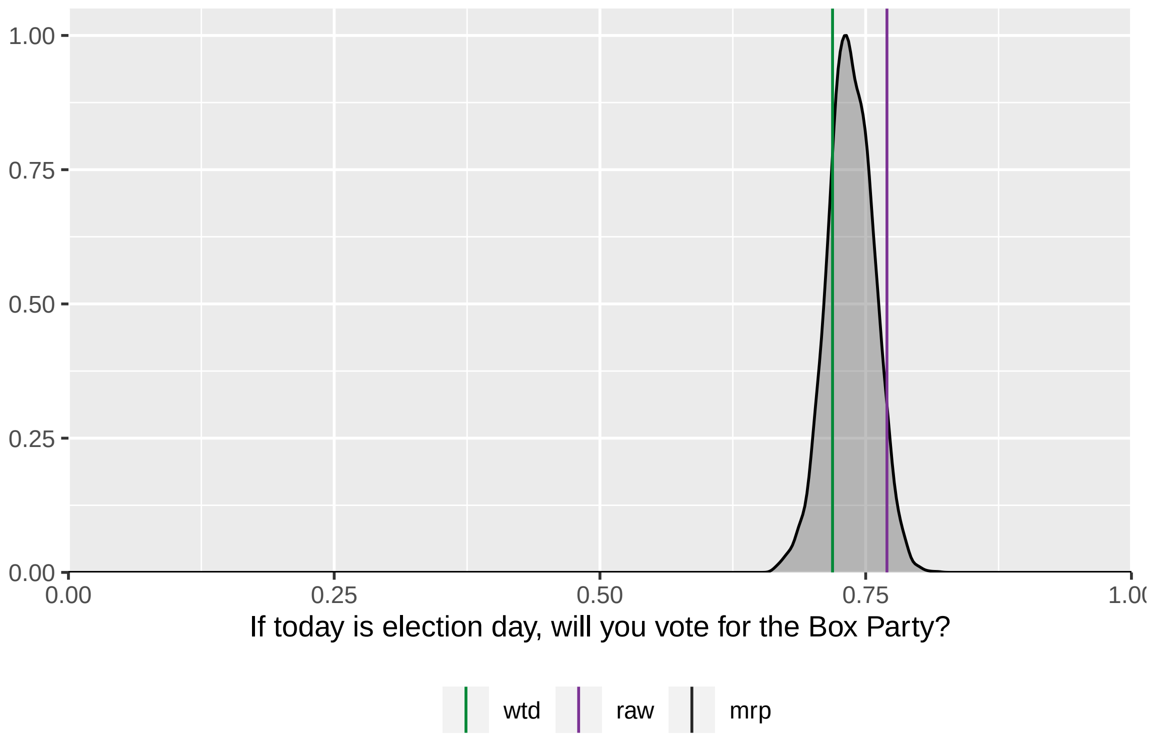
The next plot displays the probability of voting for the BP by the state without using additional statistics. We learn that the probability of the BP winning the election is high since most people in almost all the states (around 75%) vote for that party. State B is the state with the lowest probability of vote for the BP. The distribution still ranged more than 50 per cent. Note that this violin plot is generated using `ggplot2` (Wickham 2016). Hence, it could be modified the same way `ggplot2`'s plot is modified. For example, here, we can add the title and change the theme of the plot.

```
library(ggplot2)
plot_state <- fit1$plot(state_estimation, additional_stats = "none") +
  ggtitle("Probability of voting the BP by state") +
  theme_bw()
plot_state
```



Lastly, we show the plot of the population estimate, with lines representing the mean weighted and raw estimates. About 70 per cent of people in the Shape World will be likely to vote for the Box Party.

```
fit1$plot(popn_estimation, additional_stats = c("raw","wtd"))
```



3 Implementation and technical notes

There are a variety of technical notes related to this package and the architectural decisions that were made.

Section 3 discusses some of the implementation issues and technical notes related to the decisions that were made

4 Worked example

In this example we will forecast the 2020 US presidential election using survey data from the Democracy Fund Voter Study Group and use the Integrated Public Use Microdata Series (IPUMS), to access the 2018 American Community Survey (ACS) to use as post-stratification data. We are interested in considering the following variables: state, age-group, and gender.

The first step is to gather the datasets. For the survey data we are interested in the Nationscape dataset, which can be requested from the Voter Study Group. Access is usually granted within hours and is completely free. We are going to use the survey from 25 June 2020.

```
# This code based on that from Alen Mitrovski, Xiaoyan Yang, and Matthew Wankiewicz.
library(haven)
library(tidyverse)

raw_nationscape_data <-
  read_dta(here::here("paper/ns20200625.dta"))

# The Stata format separates labels so reunite those
raw_nationscape_data <-
  labelled::to_factor(raw_nationscape_data)
```

```
# Just keep relevant variables
nationscape_data <-
  raw_nationscape_data %>%
  select(vote_2020,
         gender,
         state,
         age)

# For simplicity, remove anyone undecided or planning to vote for someone other than Biden/Trump and ma
nationscape_data <-
  nationscape_data %>%
  filter(vote_2020 == "Joe Biden" | vote_2020 == "Donald Trump") %>%
  mutate(vote_biden = if_else(vote_2020 == "Joe Biden", 1, 0)) %>%
  select(-vote_2020)

# Create the dependent variables by grouping the existing variables
nationscape_data <-
  nationscape_data %>%
  mutate(
    age_group = case_when(
      age <= 29 ~ 'age_18-29',
      age <= 44 ~ 'age_30-44',
      age <= 59 ~ 'age_45-59',
      age >= 60 ~ 'age_60_or_more',
      TRUE ~ 'Trouble'
      )
    )

# Tidy the class
nationscape_data <-
  nationscape_data %>%
  mutate(across(c(gender, state, age_group, vote_biden), as_factor))

nationscape_data %>%
  head()
```

```
## # A tibble: 6 x 5
##   gender state   age vote_biden age_group
##   <fct>  <fct> <dbl> <fct>      <fct>
## 1 Female WI       49 0          age_45-59
## 2 Female VA       46 0          age_45-59
## 3 Female TX       75 0          age_60_or_more
## 4 Female WA       52 0          age_45-59
## 5 Female MA       21 1          age_18-29
## 6 Female TX       38 1          age_30-44
```

We also need a post-stratification dataset. In this case we can use the American Community Survey (ACS) accessed through IPUMS.

```
# Again, closely following code from Alen Mitrovski, Xiaoyan Yang, and Matthew Wankiewicz.
library(haven)
library(tidyverse)

raw_poststrat_data <-
  read_dta(here::here("paper/usa_00009.dta"))
```

```
# The Stata format separates labels so reunite those
raw_poststrat_data <-
  labelled::to_factor(raw_poststrat_data)
head(raw_poststrat_data)
```

```
## # A tibble: 6 x 13
##   year  sample  serial cbserial  hhwt cluster stateicp strata gq     pernum perwt
##   <fct> <fct>    <dbl>    <dbl> <dbl>   <dbl> <fct>     <dbl> <fct>   <dbl> <dbl>
## 1 2019  2019 a~      7  2.02e12  80.4 2.02e12 alabama  270201 grou~      1  80.4
## 2 2019  2019 a~     17  2.02e12 794.  2.02e12 alabama  100001 grou~      1 794.
## 3 2019  2019 a~     27  2.02e12  70.4 2.02e12 alabama  110001 othe~      1  70.4
## 4 2019  2019 a~     37  2.02e12 864.  2.02e12 alabama   30101 othe~      1 864.
## 5 2019  2019 a~     47  2.02e12 302.  2.02e12 alabama  220001 grou~      1 302.
## 6 2019  2019 a~     57  2.02e12 482.  2.02e12 alabama   60001 grou~      1 482.
## # ... with 2 more variables: sex <fct>, age <fct>
```

```
raw_poststrat_data$age <- as.numeric(raw_poststrat_data$age)
raw_poststrat_data$stateicp <-
  as.character(raw_poststrat_data$stateicp)

poststrat_data <-
  raw_poststrat_data %>%
  filter(age >= 18) %>%
  mutate(
    age_group = case_when(
      age <= 29 ~ 'age_18-29',
      age <= 44 ~ 'age_30-44',
      age <= 59 ~ 'age_45-59',
      age >= 60 ~ 'age_60_or_more',
      TRUE ~ 'Trouble'
    )
  ) %>%
  filter(
    !stateicp %in% c(
      "puerto rico",
      "state groupings (1980 urban/rural sample)",
      "military/mil. reservations",
      "state not identified"
    )
  )

# Just keep relevant variables
poststrat_data <-
  poststrat_data %>%
  select(sex,
         age_group,
         stateicp)

# Tidy the class
poststrat_data <-
  poststrat_data %>%
  mutate(across(c(sex, stateicp, age_group), as_factor))
```

Now that we have the data, we need to create a SurveyData object for the survey dataset.

```r
library(mrpkit)
# Create a SurveyData object for the survey dataset
us_presidential_election <-
  SurveyData$new(
    data = nationscape_data,
    questions = list(
      age_group = "Please identify your age",
      gender = "Please select your gender",
      vote_biden = "If today is election day, will you vote for Biden?",
      state = "Which state do you live in?"
      ),
    responses = list(
      age_group = levels(nationscape_data$age_group),
      gender = levels(nationscape_data$gender),
      vote_biden = levels(nationscape_data$vote_biden),
      state = levels(nationscape_data$state)
      )
    )
```

```
## Warning: 'Weights have not been provided, assume all data weighted with weight
## 1.
```

```r
# Display it
us_presidential_election$print()
```

```
## Survey with 5200 observations, 4 questions
## Independent Sampling design (with replacement)
##
## Column label: age_group
## Question: Please identify your age
## Allowed answers: age_45-59, age_60_or_more, age_18-29, age_30-44
##
## Column label: gender
## Question: Please select your gender
## Allowed answers: Female, Male
##
## Column label: state
## Question: Which state do you live in?
## Allowed answers: WI, VA, TX, WA, MA, CA, NC, MD, FL, WV, OH, NY, KY, IN, MI, IA, SC, MN, GA, PA, NJ,
##
## Column label: vote_biden
## Question: If today is election day, will you vote for Biden?
## Allowed answers: 0, 1
```

And we now need the same for the post-stratification dataset.

```r
# Create a SurveyData object for the survey dataset
us_post_strat <-
  SurveyData$new(
    data = poststrat_data,
    questions = list(
      age_group = "Please identify your age",
      sex = "Please select your sex",
      stateicp = "Which state do you live in?"
      ),
    responses = list(
      age_group = levels(poststrat_data$age_group),
      sex = levels(poststrat_data$sex),
      stateicp = levels(poststrat_data$stateicp)
      )
    )
```

```
## Warning: 'Weights have not been provided, assume all data weighted with weight
## 1.
```

```
# Display it
us_post_strat$print()
```

```
## Survey with 262814 observations, 3 questions
## Independent Sampling design (with replacement)
##
## Column label: age_group
## Question: Please identify your age
## Allowed answers: age_60_or_more, age_18-29, age_30-44, age_45-59
##
## Column label: sex
## Question: Please select your sex
## Allowed answers: male, female
##
## Column label: stateicp
## Question: Which state do you live in?
## Allowed answers: alabama, alaska, arizona, arkansas, california, colorado, connecticut, delaware, di
```

Now we need to start to bring the datasets together.

```
# Create QuestionMap$object for the question related to age
q_age <- QuestionMap$new(
  name = "age",
  col_names = c("age_group", "age_group"),
  values_map = list(
    "age_18-29" = "age_18-29",
    "age_30-44" = "age_30-44",
    "age_45-59" = "age_45-59",
    "age_60_or_more" = "age_60_or_more"
  )
)

# Create QuestionMap$object for the question related to gender
q_gender <- QuestionMap$new(
  name = "gender",
  col_names = c("gender", "sex"),
  values_map = data.frame(
    "Female" = "female",
    "Male" = "male"
  )
)

# Create QuestionMap$object for the question related to state
q_state <- QuestionMap$new(
  name = "state",
  col_names = c("state", "stateicp"),
  values_map = list(
  "AL" = "alabama",
  "AK" = "alaska",
  "AZ" = "arizona",
  "AR" = "arkansas",
  "CA" = "california",
  "CO" = "colorado",
  "CT" = "connecticut",
  "DE" = "delaware",
  "DC" = "district of columbia",
  "FL" = "florida",
  "GA" = "georgia",
  "HI" = "hawaii",
  "ID" = "idaho",
  "IL" = "illinois",
  "IN" = "indiana",
  "IA" = "iowa",
  "KS" = "kansas",
  "KY" = "kentucky",
  "LA" = "louisiana",
  "ME" = "maine",
  "MD" = "maryland",
  "MA" = "massachusetts",
  "MI" = "michigan",
  "MN" = "minnesota",
  "MS" = "mississippi",
  "MO" = "missouri",
  "MT" = "montana",
  "NE" = "nebraska",
  "NV" = "nevada",
  "NH" = "new hampshire",
  "NJ" = "new jersey",
  "NM" = "new mexico",
  "NY" = "new york",
  "NC" = "north carolina",
  "ND" = "north dakota",
  "OH" = "ohio",
  "OK" = "oklahoma",
  "OR" = "oregon",
  "PA" = "pennsylvania",
  "RI" = "rhode island",
  "SC" = "south carolina",
  "SD" = "south dakota",
  "TN" = "tennessee",
  "TX" = "texas",
  "UT" = "utah",
  "VT" = "vermont",
  "VA" = "virginia",
  "WA" = "washington",
  "WV" = "west virginia",
  "WI" = "wisconsin",
  "WY" = "wyoming"
  )
```

```
)
```

We can now bring them together.

```
# Create a new SurveyMap object
us_map <- SurveyMap$new(sample = us_presidential_election,
                        population = us_post_strat,
                        q_state, q_age, q_gender
                        )

print(us_map)
```

```
## ==============
## state = stateicp
## --------------
## AL = alabama
## AK = alaska
## AZ = arizona
## AR = arkansas
## CA = california
## CO = colorado
## CT = connecticut
## DE = delaware
## DC = district of columbia
## FL = florida
## GA = georgia
## HI = hawaii
## ID = idaho
## IL = illinois
## IN = indiana
## IA = iowa
## KS = kansas
## KY = kentucky
## LA = louisiana
## ME = maine
## MD = maryland
## MA = massachusetts
## MI = michigan
## MN = minnesota
## MS = mississippi
## MO = missouri
## MT = montana
## NE = nebraska
## NV = nevada
## NH = new hampshire
## NJ = new jersey
## NM = new mexico
## NY = new york
## NC = north carolina
## ND = north dakota
## OH = ohio
## OK = oklahoma
## OR = oregon
## PA = pennsylvania
## RI = rhode island
## SC = south carolina
## SD = south dakota
## TN = tennessee
## TX = texas
## UT = utah
## VT = vermont
## VA = virginia
## WA = washington
## WV = west virginia
## WI = wisconsin
## WY = wyoming
## ==============
## age_group = age_group
## --------------
## age_18-29 = age_18-29
## age_30-44 = age_30-44
## age_45-59 = age_45-59
## age_60_or_more = age_60_or_more
## ==============
## gender = sex
## --------------
## Female = female
## Male = male
```

We now finish the process of bringing the datasets together.

```
us_map$mapping()
```

If a user wants to include all the variables in the post-stratification step, then a user uses the method `SurveyMap$tabulate` without any arguments. This prepares the dataset for post-stratification. It is also possible to specify only certain variables in the post-stratification matrix, by including those variables as arguments.

```
us_map$tabulate()
```

We can now model the dataset.

```
# Example of using rstanarm::stan_glmer
fit1 <- us_map$fit(
  fun = rstanarm::stan_glmer,
  formula = vote_biden ~ (1 | age) + (1 | gender) + (1 | state),
  family = "binomial",
  refresh = 100,
  cores = 2)
```

```
## Warning: There were 8 divergent transitions after warmup. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

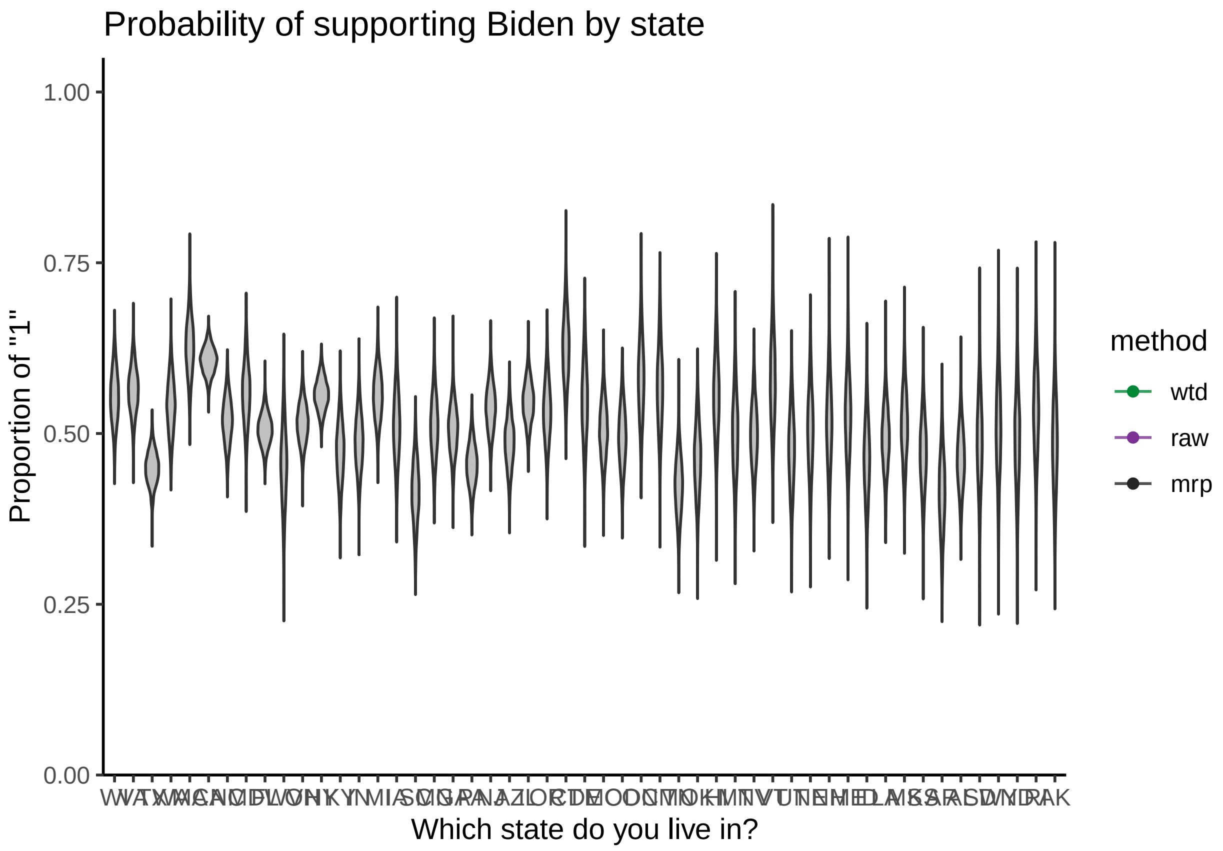
```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
# predict the probability of voting for Biden using the fit1 model
poststrat_est_fit1 <- fit1$population_predict()
```

```
# aggregate the predicted value by state
state_estimation <- fit1$aggregate(poststrat_est_fit1, by = "state")
```

```
# generate the population estimate
popn_estimation <- fit1$aggregate(poststrat_est_fit1)
```

```
library(ggplot2)
plot_state <- fit1$plot(state_estimation, additional_stats = "none") +
  ggtitle("Probability of supporting Biden by state") +
  theme_classic()
plot_state
```



5 Summary and discussion summary

5.1 On the importance of scaffolding for workflows

It is increasingly recognized that workflows are amongst the most important aspect of statistical practice. In the context of introducing a Bayesian workflow Gelman et al. (2020) says it is important because of factors including: computational challenges, model evolution, and model comparison. In the case of MRP we have similar concerns and the scaffolding provided by `mrpkit` is important.

Computational challenges mean that it is time-consuming to run models. This means that it is important to be and

The analysis often shifts focus which requires

The variety of skills

5.2 The use of R6 objects

One of the defining features of `mrpkit` is the use of R6 objects. This

5.3 Next steps and cautions

We expect that more papers will be

References

Bates, Douglas, Martin Mächler, Ben Bolker, and Steve Walker. 2015. "Fitting Linear Mixed-Effects Models Using lme4." *Journal of Statistical Software* 67 (1): 1–48. https://doi.org/10.18637/jss.v067.i01.

Blair, Graeme, Jasper Cooper, Alexander Coppock, and Macartan Humphreys. 2019. "Declaring and Diagnosing Research Designs." *American Political Science Review* 113: 838–59. https://declaredesign.org/paper.pdf.

Bürkner, Paul-Christian. 2017. "brms: An R Package for Bayesian Multilevel Models Using Stan." *Journal of Statistical Software* 80 (1): 1–28. https://doi.org/10.18637/jss.v080.i01.

Freedman Ellis, Greg, and Ben Schneider. 2021. *Srvyr: 'Dplyr'-Like Syntax for Summary Statistics of Survey Data*. https://CRAN.R-project.org/package=srvyr.

Gelman, Andrew, Aki Vehtari, Daniel Simpson, Charles C. Margossian, Bob Carpenter, Yuling Yao, Lauren Kennedy, Jonah Gabry, Paul-Christian Bürkner, and Martin Modrák. 2020. "Bayesian Workflow." http://arxiv.org/abs/2011.01808.

Goodrich, Ben, Jonah Gabry, Imad Ali, and Sam Brilleman. 2020. "Rstanarm: Bayesian Applied Regression Modeling via Stan." https://mc-stan.org/rstanarm.

Kroese, Joe. 2021. "Tidymrp: Tidy Multilevel Regression and Poststratification (MRP)."

Kuriwaki, Shiro. 2020. "ccesMRPprep: Functions and Data to Prepare CCES Data for MRP."

Lumley, Thomas. 2004. "Analysis of Complex Survey Samples." *Journal of Statistical Software* 9 (1): 1–19.

———. 2010. *Complex Surveys: A Guide to Analysis Using r: A Guide to Analysis Using r*. John Wiley; Sons.

———. 2020. "Survey: Analysis of Complex Survey Samples."

R Core Team. 2020. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

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

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2021. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.

Wickham, Hadley, and Jim Hester. 2021. *Readr: Read Rectangular Text Data*. https://CRAN.R-project.org/package=readr.

Wickham, Hadley, and Evan Miller. 2021. *Haven: Import and Export 'SPSS', 'Stata' and 'SAS' Files*. https://CRAN.R-project.org/package=haven.