Understanding Survey Weights

Daniel Chen

2023-07-12

Motivation

A common practice in survey research is to weight the data so that the demographic composition of our sample matches our population of interest. This is critical because audiences such as Black urban voters hold different views than White rural voters. We need to ensure that the proportion of Black urban voters in our sample matches those proportion of Black urban voters in the United States otherwise we risk either over-emphasizing (when the sample proportion is greater than the population proportion) or under-emphasizing (when the sample proportion is less than the population proportion) the voices of Black urban voters.

We need to also consider practical reasons for weighting. When fielding a survey, there are hard to reach audiences from non-probability samples. Young men are the least likely cohort to self-select into an online survey panel. Given unlimited time in the field and resources (cost per interview), we could theoretical collect a desired N-size that proportionally reflects the true population N.

What are Weights?

Now that we have the theoretical and logistic motivations for weights, we can ask, what are they actually?

Simply put, they're a numeric value. When a respondent takes a survey, they count for "1 person". But say we under-collected Asian respondents (that is our sample proportion is smaller than the population proportion) - we'll need the Asian respondents we collected in the sample to represent "more people". In the raw data, then, they'll no longer have a "weight" of 1 (which really means the sample is unweighted unless our sample proportions match the population proportions). They can have a weight of 2.4 or 3.789, so that they represent more than one person.

We should, however, caution, when the weights are too big. Intuitively, we probably don't want the opinions of 1 Asian person to count for 25 Asian people. Although that person is Asian, it's unlikely that a singular person can represent an entire group.

Libraries

For this tutorial, we'll double check that we have installed the `tidyverse` (for general data manipulation purposes) suite of packages along with `tidycensus` (for retrieving Census data programatically) and `anesrake` (to apply the actual individual level weights)

```
packages <- c('tidyverse', 'tidycensus', 'anesrake', 'usethis')
purrr::walk(
  .x = packages,
  .f = function(pkg) {
```

```r
    tryCatch(
      expr = {
        pkg_path <- find.package(package = pkg)
        if (length(pkg_path > 0)) {
          message(glue::glue('{pkg} package already installed!'))
        }
      },
      error = function(cond) {
        message(
          glue::glue('{pkg} package does not exist. Attempting to download.')
        )
        install.packages(pkg)
      }
    )
  }
)
```

```
## tidyverse package already installed!

## tidycensus package already installed!

## anesrake package already installed!

## usethis package already installed!
```

Collecting Target Proportions

What does the true population actually look like? We need a source to weight our sample to such that the percentage of males and females in our sample lines up with the population of interest.

For this example, we want to weight to the general population of the United States, so we'll turn to Census data and retrieve counts of males versus females to derive proportions. We'll also look at race, hispanicity, and education as well because those are commonly used identifiers to weight on.

Of course, we can also weight to arbitrary targets as well. Say we don't have a source of truth to point to, and we decided to weight our sample to 30% male and 70% female. That's also a possibility, but using those targets in this example would not be recommended since we have Census data to point to. In reality, it's unlikely that 3 out of every 10 people are male.

In the snippet below, we use utilize the `tidycensus` package to return our target proportions for weighting.

```r
# Collect vector containing all of the educational attainment variables
edu_vars <- tidycensus::load_variables(year = 2021, dataset = 'acs1') |>
  dplyr::filter(
    grepl(
      pattern = 'SEX BY AGE BY EDUCATIONAL ATTAINMENT FOR THE', x = concept
    )
  ) |>
  dplyr::pull(name)

# Store all the variables we want to pull into a list
variable_dict <- list(
```

```r
  gender = c('B01001_002', 'B01001_026'),
  race = glue::glue('B01001{LETTERS[1:5]}_001'),
  hispanicity = c('B03001_002', 'B03001_003'),
  education = edu_vars
)

# Store the survey metadata into its own object
census_dict <- tidycensus::load_variables(year = 2021, dataset = 'acs1')

# Iterate over each set of demographic variables, so we can determine proportions
proportions_dfs <- purrr::map2(
  .x = variable_dict,
  .y = names(variable_dict),
  .f = function(variables, variable_name) {

    cli::cli_h1('Retrieving summarized data for {variable_name}')

    # Load in Census data
    data <- tidycensus::get_acs(
      geography = 'us',
      year = 2021,
      variables = variables,
      output = 'tidy',
      variables_filter = list(AGE = 18:99),
      geometry = FALSE,
      key = Sys.getenv('CENSUS_API_KEY'),
      survey = 'acs1',
      show_call = FALSE
    )

    # Join on the census dict so it's clear what variables we're working with
    data <- data |>
      dplyr::left_join(census_dict, by = c('variable' = 'name'))

    # Education and race data pulled need to be cleaned up a bit more than
    # gender and hispanicity
    if (variable_name %in% 'education') {
      proportions <- data |>
        dplyr::filter(
          grepl(
            pattern = 'Less|9th|High|Some|Associate|Bachelor|Graduate',
            x = label
          )
        ) |>
        dplyr::mutate(
          label = gsub(pattern = '.*\\!\\!', replacement = '', x = label),
          label = dplyr::case_when(
            label %in% "Bachelor's degree" ~ 'College',
            label %in% 'Graduate or professional degree' ~ 'Graduate plus',
            TRUE ~ 'Less than college'
          )
        ) |>
        dplyr::group_by(label) |>
```

```r
        dplyr::summarise(estimate = sum(estimate)) |>
        dplyr::mutate(proportions = prop.table(estimate))
    } else if (variable_name %in% 'race') {
      proportions <- data |>
        dplyr::mutate(
          label = gsub(pattern = 'SEX BY AGE \\(', replacement = '', x = concept),
          label = gsub(pattern = '\\)', replacement = '', x = label),
          label = gsub(pattern = ' ALONE', replacement = '', label),
          label = dplyr::case_when(
            label %in% 'WHITE' ~ 'White',
            label %in% 'BLACK OR AFRICAN AMERICAN' ~ 'Black',
            TRUE ~ 'Other'
          )
        ) |>
        dplyr::group_by(label) |>
        dplyr::summarise(estimate = sum(estimate)) |>
        dplyr::mutate(proportions = prop.table(estimate))
    } else {
      proportions <- data |>
        dplyr::mutate(proportions = prop.table(estimate)) |>
        dplyr::select(label, estimate, proportions)
    }

    # If any demographic labels aren't clear, clean them up
    if (sum(grepl(pattern = 'Estimate', x = proportions$label)) > 0) {
      proportions <- proportions |>
        dplyr::mutate(
          label = gsub(pattern = '.*\\!\\!', replacement = '', x = label),
          label = gsub(pattern = '\\:', replacement = '', x = label)
        )
    }

    return(proportions)

  }
)
```

```
##

## -- Retrieving summarized data for gender ---------------------------------------

## Getting data from the 2021 1-year ACS

## The 1-year ACS provides data for geographies with populations of 65,000 and greater.

##

## -- Retrieving summarized data for race -----------------------------------------

## Getting data from the 2021 1-year ACS

## The 1-year ACS provides data for geographies with populations of 65,000 and greater.
```

```
##
## -- Retrieving summarized data for hispanicity ----------------------------------
## Getting data from the 2021 1-year ACS
## The 1-year ACS provides data for geographies with populations of 65,000 and greater.
##
## -- Retrieving summarized data for education ------------------------------------
## Getting data from the 2021 1-year ACS
## The 1-year ACS provides data for geographies with populations of 65,000 and greater.
```

Detour on Census API Key

One thing you'll see in the `tidycensus::get_acs()` function is the argument `key = Sys.getenv('CENSUS_API_KEY')`. If you'd like to reproduce this example, you'll need to generate your own Census API key which can be retrieved here: http://api.census.gov/data/key_signup.html.

As best practice, I do not recommend sharing your API keys publicly, so I hid mine in my `.Renviron` file. You can add your key to your `.Renviron` file by running:

```r
usethis::edit_r_environ()
```

```
## * Edit '/Users/danielchen/.Renviron'
## * Restart R for changes to take effect
```

And then typing in the following into the script that appears in your RStudio session.

```r
CENSUS_API_KEY='insert key here and keep the quotation marks'
```

Save over this file, and restart your RStudio session.

Gut Check

Our `purrr::map2()` call will return the following:

```r
purrr::map(.x = proportions_dfs, .f = kableExtra::kable)
```

$gender

label	estimate	proportions
Male	164350703	0.4951907
Female	167543042	0.5048093

$race

label	estimate	proportions
Black	40194304	0.1510471
Other	22928430	0.0861632
White	202981791	0.7627897

$hispanicity

label	estimate	proportions
Not Hispanic or Latino	269364681	0.8115991
Hispanic or Latino	62529064	0.1884009

$education

label	estimate	proportions
College	52042103	0.2013869
Graduate plus	31734956	0.1228045
Less than college	174641408	0.6758085

At this point, it's worth checking that all the n-sizes for each variable match up, so that we know we're looking at the same sample for each of these variables.

```
purrr::map(.x = proportions_dfs, .f = function(df) { sum(df$estimate) })
```

```
## $gender
## [1] 331893745
##
## $race
## [1] 266104525
##
## $hispanicity
## [1] 331893745
##
## $education
## [1] 258418467
```

Unfortunately, the N-sizes do not line up, suggesting that we're looking at different subsets of the U.S. population. In reading the documentation, we pulled `education` for those 18 years or older, but because the N-size for `gender` is so much larger, we've likely included people under the age of 18 for the variable. `race` is also different because we've only included people who identify as a single race group.

Since the purpose of this exercise is an overview of weighting, we won't spend time correcting the differing n-sizes for each group, but if we're pulling data, it's usually best practice to quality check our samples.

Creating Fake Data

Now that we have our weighting targets, we'll need some data to actually weight. For this example, we'll create a fake data set containing our weighting variables `gender`, `race`, `hispanicity`, and `education`.

```
data <- purrr::map_df(
  .x = proportions_dfs,
  .f = function(df) {
    withr::with_seed(
      seed = 23,
      code = {
```

```r
        factor(sample(x = df$label, size = 1000, replace = TRUE))
      }
    )
  }
) |>
  (\(df) dplyr::mutate(df, case_id = as.character(1:nrow(df)), wts = 1))()

# View first ten rows of our fake data
kableExtra::kable(x = head(data, 10))
```

gender	race	hispanicity	education	case_id	wts
Male	Black	Not Hispanic or Latino	College	1	1
Female	White	Hispanic or Latino	Less than college	2	1
Female	Black	Hispanic or Latino	College	3	1
Male	White	Not Hispanic or Latino	Less than college	4	1
Female	White	Hispanic or Latino	Less than college	5	1
Male	Black	Not Hispanic or Latino	College	6	1
Male	Other	Not Hispanic or Latino	Graduate plus	7	1
Male	Black	Not Hispanic or Latino	College	8	1
Male	White	Not Hispanic or Latino	Less than college	9	1
Female	Black	Hispanic or Latino	College	10	1

Let's see how the proportions randomly shaped out in our sample data.

```r
sample_proportions <- purrr::map(
  .x = names(proportions_dfs),
  .f = function(demo) {
    tapply(X = data[['wts']], INDEX = data[[demo]], FUN = sum) |>
      prop.table() |>
      as.data.frame() |>
      tibble::rownames_to_column() |>
      stats::setNames(nm = c('demo', 'proportion_in_data')) |>
      dplyr::left_join(proportions_dfs[[demo]], by = c('demo' = 'label')) |>
      dplyr::rename(target_proportions = proportions) |>
      dplyr::select(-estimate)
  }
)

purrr::map(.x = sample_proportions, .f = kableExtra::kable)
```

[[1]]

demo	proportion_in_data	target_proportions
Female	0.473	0.5048093
Male	0.527	0.4951907

[[2]]

demo	proportion_in_data	target_proportions
Black	0.344	0.1510471
Other	0.299	0.0861632
White	0.357	0.7627897

[[3]]

demo	proportion_in_data	target_proportions
Hispanic or Latino	0.473	0.1884009
Not Hispanic or Latino	0.527	0.8115991

[[4]]

demo	proportion_in_data	target_proportions
College	0.344	0.2013869
Graduate plus	0.299	0.1228045
Less than college	0.357	0.6758085

We can see that our sample proportions are way off from our target proportions. In our data, Hispanics vs. Non-Hispanics are split almost evenly. However, in reality, 81% of the U.S. general population is Hispanic. If we were to analyze data from this sample, we would be way underrepresenting Non-Hispanic voices.