Stephanie Zimmer, Rebecca Powell, and Isabella Velásquez

Tidy Survey Book

To my son,

without whom I should have finished this book two years earlier

Contents

List of Figures

List of Tables

Preface

Hi there, this is my great book.

Why read this book

It is very important...

Structure of the book

Chapters 1 introduces a new topic, and ...

Software information and conventions

I used the **knitr** package (Xie, 2015) and the **bookdown** package (Xie, 2022) to compile my book. My R session information is shown below:

```
xfun::session_info()
```

```
## R version 4.2.2 (2022-10-31)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur ... 10.16
##
## Locale: en_US.UTF-8 / en_US.UTF-8 / en_US.UTF-8 / C / en_US.UTF-
8 / en_US.UTF-8
```

```
##
## Package version:
##   base64enc_0.1.3 bookdown_0.30   bslib_0.4.1
##   cachem_1.0.6    cli_3.4.1       compiler_4.2.2
##   digest_0.6.30   evaluate_0.18   fastmap_1.1.0
##   fs_1.5.2        glue_1.6.2      graphics_4.2.2
##   grDevices_4.2.2 highr_0.9       htmltools_0.5.3
##   jquerylib_0.1.4 jsonlite_1.8.3  knitr_1.41
##   lifecycle_1.0.3 magrittr_2.0.3  memoise_2.0.1
##   methods_4.2.2   R6_2.5.1        rappdirs_0.3.3
##   renv_0.16.0     rlang_1.0.6     rmarkdown_2.18
##   rstudioapi_0.14 sass_0.4.4      stats_4.2.2
##   stringi_1.7.8   stringr_1.5.0   tinytex_0.42
##   tools_4.2.2     utils_4.2.2     vctrs_0.5.1
##   xfun_0.35       yaml_2.3.6
```

Package names are in bold text (e.g., **rmarkdown**), and inline code and filenames are formatted in a typewriter font (e.g., `knitr::knit('foo.Rmd')`). Function names are followed by parentheses (e.g., `bookdown::render_book()`).

Acknowledgments

A lot of people helped me when I was writing the book.

Frida Gomam
on the Mars

1

Introduction

2

Introducing survey data

3

Understanding survey data files

4

Introducing the srvyr package

5

Specifying sample designs in srvyr

6

Descriptive analyses in srvyr

6.1 Goals

6.2 Introduction

Descriptive analysis allows you to investigate your dataset and gain insight into the information it contains. Common descriptive analyses include calculating mean, median of numeric data or proportions in categorical data.

6.3 Overview of descriptive analysis using `srvyr` package

1. Create a `tbl_svy` object using `srvyr::as_survey_design()` or `srvyr::as_survey_rep()`
2. Subset the data for subpopulations using `dplyr::filter()`, if needed
3. Specify domains of analysis using `dplyr::group_by()`, if needed
4. Within `srvyr::summarize()`, specify variables to calculate ,means, totals, proportions, quantiles, and more

6.3.1 A brief refresher on the `dplyr::summarize()` function

The `dplyr::summarize()` function collapses many values down to a single summary:

These verbs can be used in conjunction with `group_by()`, applying the functions on a group-by-group basis to create grouped summaries.

6.4 Setup

With the ANES data, we create a `tbl_svy` object using `srvyr::as_survey_design()`, adjusting the weight to add up to the citizen population (as described in Chapter 05):

```
library(survey) # for survey analysis
library(srvyr) # for tidy survey analysis
library(readr)
library(here)

anes <-
  read_rds(here::here(
    "/Users/ivelasq/R/tidy-survey-short-course/Data/anes_2020.rds"
  )) %>%
  mutate(Weight = Weight / sum(Weight) * 231592693)

anes_des <- anes %>%
  as_survey_design(
    weights = Weight,
    strata = Stratum,
    ids = VarUnit,
    nest = TRUE
  )
```

6.5 Categorical data

Categorical data, or the [definition],

Analyzing categorical data lets us…

Common analysis for categorical data include:

- Weighted proportions
- Weighted counts
- Unweighted proportions
- Unweighted counts

6.5.1 Count observations using survey methods with `survey_count()`

With `srvyr::survey_count()`, you can produce weighted counts and variance of your choice. The syntax is very similar to the `dplyr::count()` syntax; however, it can only be called on `tbl_srvy()` objects. Let's explore the syntax:

```
survey_count(
  x,
  ...,
  wt = NULL,
  sort = FALSE,
  name = "n",
  .drop = dplyr::group_by_drop_default(x),
  vartype = c("se", "ci", "var", "cv")
  )
```

The arguments are:

- `x`: a `tbl_svy` object created by `as_survey`
- `...`: variables to group by, passed to `group_by`
- `wt`: a variable to weight on in addition to the survey weights, defaults to `NULL`
- `sort`: how to sort the variables, defaults to `FALSE`
- `name`: the name of the count variable, defaults to `n`
- `.drop`: whether to drop empty groups
- `vartype`: type(s) of variation estimate to calculate, defaults to `se` (standard error)

The steps to use `survey_count()` are:

- Specify the sample design,
- Filter subsets using `dplyr::filter()`, if needed
- Run `survey_count()`, specifying the required arguments within the function

Let's see the weighted count of responses in ANES:

```
anes_des %>% #  Specify the sample design
  survey_count() # Run `survey_count()`
```

```
## # A tibble: 1 x 2
##           n     n_se
##       <dbl>    <dbl>
## 1 231592693 3762243.
```

`srvyr::count()` can take one or many variables. To calculate a cross-tab of population in each age group and gender, we run the below:

```
anes_des %>%
  # Specify the required arguments within the function
  survey_count(AgeGroup, Gender, name = "N")
```

```
## # A tibble: 21 x 4
##    AgeGroup Gender         N     N_se
##    <fct>    <fct>      <dbl>    <dbl>
##  1 18-29    Male   21600792. 1418333.
##  2 18-29    Female 22193812. 1766188.
##  3 18-29    <NA>      65204.   56033.
##  4 30-39    Male   19848178. 1077514.
##  5 30-39    Female 19780778. 1158766.
##  6 30-39    <NA>     118195.   62999.
##  7 40-49    Male   17915676. 1123493.
##  8 40-49    Female 18932548.  946369.
##  9 40-49    <NA>      71911.   55174.
## 10 50-59    Male   19054298. 1029844.
## # ... with 11 more rows
```

6.5.2 Calculate totals using survey methods using `survey_total()`

With `srvyr::survey_total()`, we can calculate totals from complex survey data. Let's explore the syntax:

```
survey_total(
  x,
  na.rm = FALSE,
  vartype = c("se", "ci", "var", "cv"),
  level = 0.95,
  deff = FALSE,
  df = NULL,
  ...
)
```

- `x`: a variable, expression, or empty
- `na.rm`: an indicator of whether missing values should be dropped, defaults to `FALSE`
- `vartype`: type(s) of variation estimate to calculate, defaults to `se` (standard error)
- `level`: a number or a vector indicating the confidence level, defaults to 0.95
- `deff`: a logical value stating whether the design effect should be returned, defaults to FALSE
- `df`: for 'vartype = 'ci'), a numeric value indicating degrees of freedome for the t-distribution
  - For the {srvyr} package, this defaults to `NULL` whereas the {survey} package defaults to `Inf`

The steps to use `survey_total()` are:

- Specify the sample design,
- Specify the cross tab in `group_by()`,
- Within `summarize`, run `survey_total()`, specifying the required arguments within the function

To calculate a population count estimate with `survey_total()`, we can run the below:

```
anes_des %>%
  summarize(survey_total(), .groups = "drop")
```

```
## # A tibble: 1 x 2
##        coef    `_se`
##       <dbl>    <dbl>
## 1 231592693 3762243.
```

The `.groups =` argument controls the grouping structure of the output. When the output no longer have grouping variables because they are dropped, it becomes ungrouped.

Notice that `anes_des %>% summarize(survey_total(), .groups = "drop")` is equivalent to the `survey_count()` call:

```
anes_des %>%
  survey_count()
```

```
## # A tibble: 1 x 2
##           n     n_se
##       <dbl>    <dbl>
## 1 231592693 3762243.
```

The `survey_total()` function is called within `summarize`, where as `survey_count()`, like `dplyr::count()`, is not.

6.5.3 Calculate mean/proportion using survey methods with `survey_mean()` and `survey_prop()`

The `srvyr::survey_mean()` and `survey_prop()` functions calculate the means and proportions from complex survey data. Like `survey_total()`, they are called within `summarize()`. Let's explore the syntax:

```
survey_mean(
  x,
  na.rm = FALSE,
  vartype = c("se", "ci", "var", "cv"),
  level = 0.95,
  proportion = FALSE,
  prop_method = c("logit", "likelihood", "asin", "beta", "mean"),
  deff = FALSE,
  df = NULL,
  ...
)

survey_prop(
  vartype = c("se", "ci", "var", "cv"),
  level = 0.95,
  proportion = FALSE,
  prop_method = c("logit", "likelihood", "asin", "beta", "mean"),
  deff = FALSE,
  df = NULL,
  ...
)
```

The steps involved are:

- Specify the sample design,
- Specify the cross tab in `group_by()`,
- Run `survey_mean()` or `survey_prop()` within `summarize()`

Looking at population by age group, we can calculate the weighted proportion for each group in the data:

```
anes_des %>%
  group_by(AgeGroup) %>%
  summarize(p1 = survey_mean(),
            .groups = "drop")
```

```
## # A tibble: 7 x 3
##   AgeGroup        p1   p1_se
##   <fct>        <dbl>   <dbl>
## 1 18-29       0.189  0.00838
## 2 30-39       0.172  0.00659
## 3 40-49       0.159  0.00609
## 4 50-59       0.169  0.00657
## 5 60-69       0.155  0.00488
## 6 70 or older 0.119  0.00474
## 7 <NA>        0.0369 0.00305
```

The `survey_prop()` function is equivalent to leaving out the `x` argument in `survey_mean()`.

```
anes_des %>%
  group_by(AgeGroup) %>%
  summarize(p1 = survey_prop(),
            .groups = "drop")
```

```
## # A tibble: 7 x 3
##   AgeGroup        p1   p1_se
##   <fct>        <dbl>   <dbl>
## 1 18-29       0.189  0.00838
## 2 30-39       0.172  0.00659
## 3 40-49       0.159  0.00609
## 4 50-59       0.169  0.00657
## 5 60-69       0.155  0.00488
## 6 70 or older 0.119  0.00474
## 7 <NA>        0.0369 0.00305
```

6.5.4 Calculate proportions with confidence intervals

We can also calculate confidence intervals within `summarize` using `survey_mean()` or `survey_mean()` by setting the `vartype` to `ci`. The confidence intervals provide us with an upper and lower column for each method.

```
anes_des %>%
  group_by(Income7, VotedPres2016, VotedPres2020) %>%
  summarize(pd = survey_prop(vartype = "ci") %>% round(4)) %>%
  select(Income7, VotedPres2016, VotedPres2020, contains("_"))
```

```
## # A tibble: 45 x 5
## # Groups:   Income7, VotedPres2016 [22]
##    Income7    VotedPres2016 VotedPres2~1  pd_low pd_upp
##    <fct>      <fct>         <fct>          <dbl>  <dbl>
##  1 Under $20k Yes           Yes           0.784  0.892
##  2 Under $20k Yes           No            0.108  0.216
##  3 Under $20k No            Yes           0.234  0.370
##  4 Under $20k No            No            0.630  0.766
##  5 Under $20k <NA>          Yes          -0.0705 1.03
##  6 Under $20k <NA>          No           -0.202  0.746
##  7 Under $20k <NA>          <NA>         -0.198  0.697
##  8 $20-40k    Yes           Yes           0.818  0.914
##  9 $20-40k    Yes           No            0.0845 0.178
## 10 $20-40k    Yes           <NA>         -0.0025 0.0073
## # ... with 35 more rows, and abbreviated variable name
## #   1: VotedPres2020
```

We can change the proportion method by adding `prop_method`. The available options are: `"logit"`, `"likelihood"`, `"asin"`, `"beta"`, and `"mean"`.

Using `proportion = TRUE` make confidence intervals more accurate near 0 and 1.

```
anes_des %>%
  group_by(Income7, VotedPres2016, VotedPres2020) %>%
  summarize(pl = survey_prop(
    proportion = TRUE,
    prop_method = "logit",
    vartype = "ci"
  ) %>% round(4)) %>%
  select(Income7, VotedPres2016, VotedPres2020, contains("_"))
```

```
## # A tibble: 45 x 5
## # Groups:   Income7, VotedPres2016 [22]
##    Income7    VotedPres2016 VotedPres2020 pl_low pl_upp
##    <fct>      <fct>         <fct>          <dbl>  <dbl>
##  1 Under $20k Yes           Yes           0.777   0.885
##  2 Under $20k Yes           No            0.115   0.223
##  3 Under $20k No            Yes           0.239   0.374
##  4 Under $20k No            No            0.626   0.761
##  5 Under $20k <NA>          Yes           0.0923  0.892
##  6 Under $20k <NA>          No            0.0332  0.804
##  7 Under $20k <NA>          <NA>          0.0297  0.784
##  8 $20-40k    Yes           Yes           0.811   0.907
##  9 $20-40k    Yes           No            0.0911  0.186
## 10 $20-40k    Yes           <NA>          0.0003  0.018
## # ... with 35 more rows
```

6.5.5 Other functions that use survey methods

The {srvyr} package includes other functions for summarizing datasets:

- Center: `survey_mean()`, `survey_prop()`, `survey_median()`
- Count: `survey_count()`, `survey_total()`
- Range: `survey_quantile()`
- Ratio: `survey_ratio()`
- Variance: `survey_var()`, `survey_sd()`

6.5.6 Calculate conditional proportions with more than one group

Specifying more than one group calculates conditional proportions. Say we wanted to know the proportion of people who voted in 2016 and 2020. After the `tbl_svy` object, we specify the two variables we want to calculate proportions for:

```
anes_des %>%
  filter(!is.na(VotedPres2016), !is.na(VotedPres2020)) %>%
  group_by(VotedPres2016, VotedPres2020) %>%
  summarize(
    p = survey_mean(),
    N = survey_total(),
    n = unweighted(n()),
```

```
    .groups = "drop"
  )
```

```
## # A tibble: 4 x 7
##   VotedPre~1 Voted~2      p    p_se      N   N_se     n
##   <fct>      <fct>    <dbl>   <dbl>  <dbl>  <dbl> <int>
## 1 Yes        Yes     0.924  0.00566 1.45e8 2.62e6  5534
## 2 Yes        No      0.0762 0.00566 1.19e7 9.55e5   274
## 3 No         Yes     0.455  0.0162  3.39e7 1.59e6   859
## 4 No         No      0.545  0.0162  4.06e7 2.04e6   761
## # ... with abbreviated variable names
## #   1: VotedPres2016, 2: VotedPres2020
```

Note that this is the proportion of **people voting in 2020 by whether people voted in 2016**. That is, it is the weighted number of people who voted in both 2016 and 2020 (144578247), divided by the weighted number of people who voted in 2016 (144578247 + 11917394). Running the above, we see that 92.4% of people who voted in 2016 voted in 2020.

6.5.7 Calculate joint proportions with more than one group

When we want to calculate multiple variables as if they were a single variable, we use {srvyr}'s `interact`. We use `interact` within `group_by()` to calculate the joint proportions of two or more variables.

```
anes_des %>%
  filter(!is.na(VotedPres2020), !is.na(VotedPres2016)) %>%
  group_by(interact(VotedPres2016, VotedPres2020)) %>% #<<
  summarize(p = survey_mean(),
            N = survey_total(),
            .groups = "drop")
```

```
## # A tibble: 4 x 6
##   VotedPres2016 VotedPre~1      p    p_se      N   N_se
##   <fct>         <fct>       <dbl>   <dbl>  <dbl>  <dbl>
## 1 Yes           Yes        0.626  0.00934 1.45e8 2.62e6
## 2 Yes           No         0.0516 0.00391 1.19e7 9.55e5
## 3 No            Yes        0.147  0.00628 3.39e7 1.59e6
## 4 No            No         0.176  0.00770 4.06e7 2.04e6
## # ... with abbreviated variable name 1: VotedPres2020
```

Since `interact` groups by multiple variables as if they were a single variable, the proportions sum to 100% across more than a single grouping variable.

```
anes_des %>%
  filter(!is.na(VotedPres2020),!is.na(VotedPres2016)) %>%
  group_by(interact(VotedPres2016, VotedPres2020)) %>% #<<
  summarize(p = survey_mean(),
            N = survey_total(),
            .groups = "drop") %>%
  summarize(p_sum = sum(p))
```

```
## # A tibble: 1 x 1
##   p_sum
##   <dbl>
## 1     1
```

6.5.8 Calculate proportions with design effects

Note above that functions `survey_total()`, `survey_mean()`, and `survey_prop()` have the argument `deff`. `deff` stands for Design Effect, the ratio of two variances. Use `deff = TRUE` argument to specify whether the design effect should be returned.

```
anes_des %>%
  filter(!is.na(VotedPres2016),!is.na(VotedPres2020)) %>%
  group_by(interact(VotedPres2016, VotedPres2020)) %>%
  summarize(p = survey_mean(deff = TRUE), #<<
            N = survey_total())
```

```
## # A tibble: 4 x 7
##   VotedPr~1 Voted~2      p    p_se p_deff      N   N_se
##   <fct>     <fct>    <dbl>   <dbl>  <dbl>  <dbl>  <dbl>
## 1 Yes       Yes     0.626  0.00934   2.76 1.45e8 2.62e6
## 2 Yes       No      0.0516 0.00391   2.32 1.19e7 9.55e5
## 3 No        Yes     0.147  0.00628   2.34 3.39e7 1.59e6
## 4 No        No      0.176  0.00770   3.04 4.06e7 2.04e6
## # ... with abbreviated variable names
## #   1: VotedPres2016, 2: VotedPres2020
```

7

Statistical testing

8

Modeling

9

Presenting results

A

More to Say

Yeah! I have finished my book, but I have more to say about some topics. Let me explain them in this appendix.

To know more about **bookdown**, see https://bookdown.org.

This is for testing GH Actions.

Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2022). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.30.

Index