# Tidy Survey Analysis in R Using the srvyr Package

## Workshop Day 1 - Categorical Data

Stephanie Zimmer, Abt Associates

Rebecca Powell, RTI International

Isabella Velásquez, RStudio

April 15, 2022

# Introduction

# Overview

- At the end of this workshop series, you should be able to
    - Calculate point estimates and their standard errors with survey data
        - Proportions, totals, and counts
        - Means, quantiles, and ratios
    - Perform t-tests and chi-squared tests
    - Fit regression models
    - Specify a survey design in R to create a survey object
- We will not be going over the following but provide some resources at the end
    - Weighting (calibration, post-stratification, raking, etc.)
    - Survival analysis
    - Nonlinear models

# About Us

**Stephanie Zimmer**
Abt Associates

**Rebecca Powell**
RTI International

**Isabella Velásquez**
RStudio

# About Us

**Stephanie Zimmer**
Abt Associates

**Rebecca Powell**
RTI International

**Isabella Velásquez**
RStudio

**Thank you to our volunteers!**

Raphael Nishimura will be assisting during our breakout rooms.

# About This Workshop

- Hosted by Midwest Association for Public Opinion Research (MAPOR), a regional chapter of the American Association for Public Opinion Research (AAPOR).
- Originally delivered at AAPOR Conference in May 2021

# Upcoming Work

- Book on analyzing survey data in R, published by CRC, Taylor & Francis Group
- We would love your help! After each course, we will send out a survey to gather your feedback on the material, organization, etc.
- Keep updated by following our project on GitHub: https://github.com/tidy-survey-r

# Workshop Overview

# Workshop Series Roadmap

- Get familiar with RStudio Cloud with a warm-up exercise using the tidyverse (today)
- Introduce the survey data we'll be using in the workshop (today)
- Analysis of categorical data with time for practice (today)
- Analysis of continuous data with time for practice (day 2)
- Survey design objects, constructing replicate weights, and creating derived variables (day 3)

# Logistics

- We will be using RStudio Cloud today to ensure everyone has access
- Sign-up for a free RStudio Cloud account (https://rstudio.cloud/)
- Access the project and files via link in email and Zoom chat
- Click "START" to open the project and get started
- RStudio Cloud has the same features and appearance as RStudio for ease of use
- All slides and code are available on GitHub: https://github.com/tidy-survey-r/tidy-survey-short-course

# Intro to RStudio Cloud: Penguins!!

- Using `palmerpenguins` data for warm-up exercises
- Data were collected and made available by Dr. Kristen Gorman and the Palmer Station, Antarctica LTER, a member of the Long Term Ecological Research Network.
- Access data through `palmerpenguins` package https://github.com/allisonhorst/palmerpenguins/

**If you are using your own RStudio environment:**

- Make sure you have `tidyverse`, `here`, and `palmerpenguins` installed

```
# Run package installation if you don't have these packages already
# As a reminder, installing takes package from internet to your computer
# and only needs to be done once, not each session

install.packages(c("tidyverse", "here", "palmerpenguins"))
```

# Intro to RStudio Cloud: Penguins!!

- Load `tidyverse`, `here`, and `palmerpenguins`
- Look at the penguins dataset using `glimpse`

```
library(tidyverse) # for tidyverse
library(here) # for file paths
library(palmerpenguins) # for warm-up data
glimpse(penguins)
```

```
## Rows: 344
## Columns: 8
## $ species           <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adel…
## $ island            <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgerse…
## $ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, …
## $ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, …
## $ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186…
## $ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, …
## $ sex               <fct> male, female, female, NA, female, male, female, male…
## $ year              <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007…
```

# Warm-up Exercises: WarmUpExercises.Rmd

- **Let's open RStudio cloud and do some warm-up examples**
- Take 10 minutes to set up RStudio Cloud and do these exercises in breakout rooms. We will then go over together
- Explore the penguins data
    - How many penguins of each species are there?
    - How many penguins of each species and sex are there?
    - What is the proportion of each species of penguins?
    - What is the proportion of each sex of penguins within species?

# Ex. 1: How many penguins of each species are there?

```
penguins %>%
   count(species)
```

```
## # A tibble: 3 × 2
##   species       n
##   <fct>     <int>
## 1 Adelie      152
## 2 Chinstrap    68
## 3 Gentoo      124
```

```
penguins %>%
   group_by(species) %>%
   summarise(
      n=n(), .groups="drop"
   )
```

```
## # A tibble: 3 × 2
##   species       n
##   <fct>     <int>
## 1 Adelie      152
## 2 Chinstrap    68
## 3 Gentoo      124
```

# Ex. 2: How many penguins of each species and sex are there?

```
penguins %>%
   count(species, sex)
```

```
## # A tibble: 8 × 3
##   species   sex        n
##   <fct>     <fct>  <int>
## 1 Adelie    female    73
## 2 Adelie    male      73
## 3 Adelie    <NA>       6
## 4 Chinstrap female    34
## 5 Chinstrap male      34
## 6 Gentoo    female    58
## 7 Gentoo    male      61
## 8 Gentoo    <NA>       5
```

# Ex. 3: What is the proportion of each species of penguins?

```
penguins %>%
   count(species) %>%
   mutate(
      p=n/sum(n)
   )
```

```
## # A tibble: 3 × 3
##   species       n     p
##   <fct>     <int> <dbl>
## 1 Adelie      152 0.442
## 2 Chinstrap    68 0.198
## 3 Gentoo      124 0.360
```

# What is the proportion of each sex of penguins within species?

```
penguins %>%
   count(species, sex) %>%
   group_by(species) %>%
   mutate(
      p=n/sum(n)
   )
```

```
## # A tibble: 8 × 4
## # Groups:   species [3]
##   species   sex        n      p
##   <fct>     <fct>  <int>  <dbl>
## 1 Adelie    female    73 0.480
## 2 Adelie    male      73 0.480
## 3 Adelie    <NA>       6 0.0395
## 4 Chinstrap female    34 0.5
## 5 Chinstrap male      34 0.5
## 6 Gentoo    female    58 0.468
## 7 Gentoo    male      61 0.492
## 8 Gentoo    <NA>       5 0.0403
```

# Survey Datasets

# American National Election Studies (ANES) 2020

- Pre and post election surveys
- Fielded almost every 2 years since 1948
- Topics include voter registration status, candidate preference, opinions on country and government, party and ideology affiliation, opinions on policy, news sources, and more
- Collaboration of Stanford, University of Michigan – funding by the National Science Foundation
- **Target Population**: US citizens, 18 and older living in US
- **Mode**: Web, videoconference, or telephone.
- **Sample Information**: Pseudo-strata and pseudo-cluster included for variance estimation

https://electionstudies.org/

# Categorical descriptive data analysis

# Overview of Survey Analysis using `srvyr` Package

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

**Note: We will be teaching this in the reverse order!!!**

# Weighted Analysis for Categorical Variable

- Functions to use within `summarize` after `group_by`
    - survey_mean/survey_prop
    - survey_total
- Functions to get counts
    - survey_count

# Set-up for Analysis

- `srvyr` package uses tidy-syntax but uses the `survey` package behind it to do calculations
- If using your own RStudio environment, install both packages:

```
# Install survey and srvyr packages

remotes::install_github("bschneidr/r-forge-survey-mirror")
install.packages("srvyr")
```

- First, we will set-up a design object and talk about what it means in Session 3

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

anes <- read_rds(here("Data", "anes_2020.rds")) %>%
   mutate(Weight=Weight/sum(Weight)*231592693)
# adjust weight to sum to citizen pop, 18+ in Nov 2020 per ANES methodology documentation
anes_des <- anes %>%
   as_survey_design(weights = Weight,
                    strata = Stratum,
                    ids = VarUnit,
                    nest = TRUE)
```

# `survey_count` Syntax

- `survey_count` functions similarly to `count` in that it is **NOT** called within `summarize`
- Produces weighted counts and variance of your choice of those counts

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

# `survey_count` Example

- Cross-tab of population in each age group and gender

```
anes_des %>%
   survey_count(AgeGroup, Gender, name="N")
```

```
## # A tibble: 21 × 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
```

# `survey_mean` and `survey_total` within `summarize`

- Specify the sample design,
- then specify the crosstab in `group_by`,
- then `survey_mean` or `survey_prop` used with no x (variable) calculates a proportion of groups within `summarize`, or
- `survey_total` used with no x (variable) calculates a population count estimate within `summarize`

# `survey_mean` and `survey_prop` 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,
  ...
)
```

# `survey_mean` and `survey_total` Examples

Looking at population by age group as done with `survey_count`.

```
anes_des %>%
   group_by(AgeGroup) %>%
   summarize(
      p1=survey_mean(),
      p2=survey_prop(),
      N=survey_total(),
      n=unweighted(n()), # this gets unweighted counts aka sample sizes
      .groups="drop" # summarize option to remove groups
   )
```

```
## # A tibble: 7 × 8
##   AgeGroup        p1   p1_se     p2   p2_se         N     N_se     n
##   <fct>        <dbl>   <dbl>  <dbl>   <dbl>     <dbl>    <dbl> <int>
## 1 18-29       0.189  0.00838 0.189  0.00838 43859809. 2340503.   871
## 2 30-39       0.172  0.00659 0.172  0.00659 39747151. 1556193.  1241
## 3 40-49       0.159  0.00609 0.159  0.00609 36920134. 1452300.  1081
## 4 50-59       0.169  0.00657 0.169  0.00657 39191266. 1602082.  1200
## 5 60-69       0.155  0.00488 0.155  0.00488 35833416. 1214320.  1436
## 6 70 or older 0.119  0.00474 0.119  0.00474 27503517. 1146535.  1330
## 7 <NA>        0.0369 0.00305 0.0369 0.00305  8537401.  710907.   294
```

# Conditional proportions with more than one group

- Specifying more than one group calculates conditional proportions
- Example: people voting in 2016 and 2020

```
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 × 7
##   VotedPres2016 VotedPres2020      p    p_se          N     N_se     n
##   <fct>         <fct>          <dbl>   <dbl>      <dbl>    <dbl> <int>
## 1 Yes           Yes           0.924  0.00566 144578247. 2617349.  5534
## 2 Yes           No            0.0762 0.00566  11917394.  955174.   274
## 3 No            Yes           0.455  0.0162   33923120. 1594478.   859
## 4 No            No            0.545  0.0162   40606907. 2036095.   761
```

# Joint proportions with more than one group

- Specify an interaction to get joint distribution - use `interact` within `group_by`
- Example: people voting in 2016 and 2020

```
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 × 6
##   VotedPres2016 VotedPres2020      p    p_se          N     N_se
##   <fct>         <fct>          <dbl>   <dbl>      <dbl>    <dbl>
## 1 Yes           Yes            0.626  0.00934 144578247. 2617349.
## 2 Yes           No             0.0516 0.00391  11917394.  955174.
## 3 No            Yes            0.147  0.00628  33923120. 1594478.
## 4 No            No             0.176  0.00770  40606907. 2036095.
```

# Proportions with Design Effects

```
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 × 7
##   VotedPres2016 VotedPres2020      p    p_se p_deff          N     N_se
##   <fct>         <fct>          <dbl>   <dbl>  <dbl>      <dbl>    <dbl>
## 1 Yes           Yes            0.626  0.00934   2.76 144578247. 2617349.
## 2 Yes           No             0.0516 0.00391   2.32  11917394.  955174.
## 3 No            Yes            0.147  0.00628   2.34  33923120. 1594478.
## 4 No            No             0.176  0.00770   3.04  40606907. 2036095.
```

# Proportions: confidence intervals

```
anes_des %>%
   group_by(interact(Income7, VotedPres2016, VotedPres2020)) %>%
   summarize(
      pd=survey_prop(vartype="ci") %>% round(4),
      pl=survey_prop(proportion = TRUE, prop_method="logit", vartype="ci") %>% round(4),
      px=survey_prop(proportion = TRUE, prop_method="likelihood", vartype="ci") %>% round(4)
   ) %>% select(Income7, VotedPres2016, VotedPres2020, contains("_")) %>%
   DT::datatable(fillContainer = FALSE, options = list(pageLength = 4))
```

# Proportions: confidence intervals (results)

Show 4 entries Search:

| | Income7 | VotedPres2016 | VotedPres2020 | pd_low | pd_upp | pl_low | pl_upp | px_low | px_upp |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Under $20k | Yes | Yes | 0.0286 | 0.0377 | 0.0289 | 0.038 | 0.0288 | 0.0379 |
| 2 | Under $20k | Yes | No | 0.0042 | 0.0086 | 0.0045 | 0.0091 | 0.0044 | 0.0089 |
| 3 | Under $20k | No | Yes | 0.0109 | 0.0173 | 0.0113 | 0.0177 | 0.0112 | 0.0176 |
| 4 | Under $20k | No | No | 0.0263 | 0.039 | 0.0269 | 0.0397 | 0.0267 | 0.0394 |

Showing 1 to 4 of 45 entries Previous 1 2 3 4 5 ... 12 Next

# Breakout rooms: Practice time

- Open CategoricalExercises.Rmd and work through Part 1
- We will take 15 minutes. Use this time for the exercises and questions.

# Categorical data testing and modeling

# `svychisq` Syntax

- Testing and modeling is done with the `survey` package
- You can use the same design object

```
svychisq(formula,
         design,
         statistic = c("F",  "Chisq", "Wald", "adjWald", "lincom", "saddlepoint"),
         na.rm=TRUE,
         ...)
```

# `svychisq` Example 1: Function Defaults

- How often can you trust the federal gov't to do what is right?
- How often can you trust other people?

```
anes_des %>%
   svychisq(design=.,
            formula=~TrustPeople +TrustGovernment)
```

```
##
##     Pearson's X^2: Rao & Scott adjustment
##
## data:  NextMethod()
## F = 29.08, ndf = 11.443, ddf = 583.587, p-value < 2.2e-16
```

# `svychisq` Example 2: Wald Statistic

- How often can you trust the federal gov't to do what is right?
- Who did you vote for? Biden, Trump, or Other

```
anes_des %>%
   svychisq(design=.,
            formula=~TrustGovernment +VotedPres2020_selection,
            statistic="Wald")
```

```
##
##     Design-based Wald test of association
##
## data:  NextMethod()
## F = 6.136, ndf = 8, ddf = 51, p-value = 1.571e-05
```

# Refresher on formula notation

| Symbol | Example | Meaning |
|---|---|---|
| + | `+X` | include this variable |
| - | `-X` | delete this variable |
| : | `X:Z` | include the interaction between these variables |
| * | `X*Z` | include these variables and the interactions between them |
| ^n | `(X+Z+Y)^3` | include these variables and all interactions up to n way |
| I | `I(X-Z)` | as-as: include a new variable which is the difference of these variables |

# Formula notation - knowledge check

I want to model the following:

$$mpg_i = \beta_0 + \beta_1 cyl_i + \beta_2 disp_i + \beta_3 hp_i + \beta_4 cyl_i disp_i + \beta_5 cyl_i hp_i + \beta_6 disp_i hp_i + \epsilon_i$$

How can you write this formula? Select all that apply:

1. `mpg~cyl:disp:hp`
2. `mpg~(cyl+disp+hp)^2`
3. `mpg~cyl+disp+hp+cyl:disp+cyl:hp+disp:hp`
4. `mpg~cyl*disp*hp`
5. `mpg~cyl*disp+cyl*hp+disp*hp`

# Formula notation - knowledge check (solution)

I want to model the following:

$$mpg_i = \beta_0 + \beta_1 cyl_i + \beta_2 disp_i + \beta_3 hp_i + \beta_4 cyl_i disp_i + \beta_5 cyl_i hp_i + \beta_6 disp_i hp_i + \epsilon_i$$

How can you write this formula? Select all that apply:

1. `mpg~cyl:disp:hp` - no, this only has the interactions
2. `mpg~(cyl+disp+hp)^2` - yes
3. `mpg~cyl+disp+hp+cyl:disp+cyl:hp+disp:hp` - yes
4. `mpg~cyl*disp*hp` - no, this also has the 3-way interaction
5. `mpg~cyl*disp+cyl*hp+disp*hp` - yes

There may be other ways as well!!!

# Logistic regression with **svyglm**

```
svyglm(formula, # response ~ terms
       design,
       na.action, #default is na.omit
       family = quasibinomial, # use this to avoid warning about non-integers
       ....)
```

# Example logistic regression

- Predicting trust in government by who someone voted in 2020

```
filter(anes_des, Weight>0) %>%
   svyglm(design=.,
          formula=TrustGovernment~ VotedPres2020_selection,
          family = quasibinomial) %>%
   summary()
```

```
##
## Call:
## svyglm(formula = TrustGovernment ~ VotedPres2020_selection, design = .,
##     family = quasibinomial)
##
## Survey design:
## Called via srvyr
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   4.6785     0.3266  14.323   <2e-16 ***
## VotedPres2020_selectionTrump -0.3530     0.4008  -0.881   0.3829
## VotedPres2020_selectionOther  2.5265     1.0868   2.325   0.0243 *
## ---
```

# Breakout rooms: Practice time

- Open CategoricalExercises.Rmd and work through Part 2
- We will take 15 minutes. Use this time for the exercises and questions.

# Closing

# Resources for more learning

- https://cran.r-project.org/web/packages/srvyr/vignettes/srvyr-vs-survey.html
- https://r-survey.r-forge.r-project.org/survey/
- Includes more advanced modeling

# Thank You!

## We hope you learned a lot in this session!

Please let us know if you have any feedback on this workshop. All feedback is welcome!

# Questions?

# Sources


- The American National Election Studies (https://electionstudies.org/). These materials are based on work supported by the National Science Foundation under grant numbers SES 1444721, 2014-2017, the University of Michigan, and Stanford University.
- Horst AM, Hill AP, Gorman KB (2020). palmerpenguins: Palmer Archipelago (Antarctica) penguin data. R package version 0.1.0. https://allisonhorst.github.io/palmerpenguins/
- T. Lumley (2020) "survey: analysis of complex survey samples". R package version 4.0. https://r-survey.r-forge.r-project.org/survey/
- Greg Freedman Ellis and Ben Schneider (2020). srvyr: 'dplyr'-Like Syntax for Summary Statistics of Survey Data. R package version 1.0.0. https://CRAN.R-project.org/package=srvyr
- Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2021). dplyr: A Grammar of Data Manipulation. R package version 1.0.5. https://CRAN.R-project.org/package=dplyr

# Session info - platform

```
##  setting  value
##  version  R version 4.1.2 (2021-11-01)
##  os       macOS Big Sur 10.16
##  system   x86_64, darwin17.0
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  ctype    en_US.UTF-8
##  tz       America/New_York
##  date     2022-04-12
##  pandoc   2.14.0.3 @ /Applications/RStudio.app/Contents/MacOS/pandoc/ (via rmarkdown)
```

# Session info - packages

```
##  package         * version date (UTC) lib source
##  dplyr           * 1.0.8   2022-02-08 [1] CRAN (R 4.1.2)
##  DT              * 0.20    2021-11-15 [1] CRAN (R 4.1.1)
##  forcats         * 0.5.1   2021-01-27 [1] CRAN (R 4.1.0)
##  ggplot2         * 3.3.5   2021-06-25 [1] CRAN (R 4.1.0)
##  here            * 1.0.1   2020-12-13 [1] CRAN (R 4.1.0)
##  knitr           * 1.38    2022-03-25 [1] CRAN (R 4.1.2)
##  Matrix          * 1.4-0   2021-12-08 [1] CRAN (R 4.1.0)
##  palmerpenguins * 0.1.0   2020-07-23 [1] CRAN (R 4.1.0)
##  purrr           * 0.3.4   2020-04-17 [1] CRAN (R 4.1.0)
##  readr           * 2.1.2   2022-01-30 [1] CRAN (R 4.1.2)
##  remotes         * 2.4.2   2021-11-30 [1] CRAN (R 4.1.0)
##  srvyr           * 1.1.1   2022-02-20 [1] CRAN (R 4.1.2)
##  stringr         * 1.4.0   2019-02-10 [1] CRAN (R 4.1.0)
##  survey          * 4.1-1   2021-07-19 [1] CRAN (R 4.1.0)
##  survival        * 3.2-13  2021-08-24 [1] CRAN (R 4.1.2)
##  tibble          * 3.1.6   2021-11-07 [1] CRAN (R 4.1.0)
##  tidyr           * 1.2.0   2022-02-01 [1] CRAN (R 4.1.2)
##  tidyverse       * 1.3.1   2021-04-15 [1] CRAN (R 4.1.0)
##  xaringan        * 0.22    2021-06-23 [1] CRAN (R 4.1.0)
##
##  [1] /Library/Frameworks/R.framework/Versions/4.1/Resources/library
```