Tidy Survey Analysis in R using the srvyr Package

Workshop Day 3 - Design Objects, Variables, and Process

Stephanie Zimmer, Abt Associates

Rebecca Powell, RTI International

Isabella Velásquez, RStudio

April 29, 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!

Greg Freedman-Ellis and 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 (day 1)
- Introduce the survey data we'll be using in the workshop (day 1)
- Analysis of categorical data with time for practice (day 1)
- Analysis of continuous data with time for practice (day 2)
- Survey design objects, constructing replicate weights, and creating derived variables (today)

Logistics

- We will be using RStudio Cloud today to ensure everyone has access
- Sign-up for a free RStudio Cloud account
    - 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

Specifying sample design objects

Overview of Survey Analysis using `srvyr` Package

Discussing step 1! Steps 2-4 discussed in prior workshops

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

Review of sampling designs

These features can be combined to form one design

- Simple random sampling: every unit has the same chance of being selected
    - Without replacement: units can only be selected once
    - With replacement: units can be selected more than once
- Systematic sampling: sample n individuals from a ordered list and sampling individuals at an interval with a random starting point
- Probability proportional to size: probability of selection is proportional to "size"
- Stratified sampling: divide population into mutually exclusive subgroups (strata). Randomly sample within each stratum
- Clustered sampling: divide population into mutually exclusive subgroups (clusters). Randomly sample clusters and then individuals within clusters

Determining the design

- Look at documentation associated with the analysis file
- Keywords to look for: methodology, design, analysis guide, technical documentation
- Documentation will indicate the variables needed to specify the design. Look for:
    - weight (almost always)
    - strata and/or clusters/PSUs. Sometimes pseudo-strata and pseudo-cluster OR
    - replicate weights (this is used instead of strata/clusters for analysis)
    - might also see finite population correction or population sizes
- Documentation may include syntax for SAS, SUDAAN, Stata and/or R!

Example: 2020 ANES

- https://electionstudies.org/data-center/2020-time-series-study/
- Opened the file "User Guide and Codebook"
- Section "Data Analysis, Weights, and Variance Estimation": Page 8-12 includes information on weights and strata/cluster variables

> For analysis of the complete set of cases using pre-election data only, including all cases and representative of the 2020 electorate, use the full sample pre-election weight, V200010a. For analysis including post-election data for the complete set of participants (i.e., analysis of post-election data only or a combination of pre- and post-election data), use the full sample post-election weight, V200010b. Additional weights are provided for analysis of subsets of the data...

For weight	Use variance unit/PSU/cluster	and use variance stratum
V200010a	V200010c	V200010d
V200010b	V200010c	V200010d

Example: RECS 2015

- https://www.eia.gov/consumption/residential/data/2015/index.php?view=microdata
- Opened the file "Using the 2015 microdata file to compute estimates and standard errors (RSEs)"
- Page 4:

> The following instructions are examples for calculating any RECS estimate using the final weights (NWEIGHT) and the associated RSE using the replicate weights (BRRWT1 – BRRWT96).
>
> Let ϵ be the Fay coefficent ... and $\epsilon = 0.5$

- Page 9: Syntax given for survey package which is similar to srvyr (as we will see)

```
library(survey)
RECS15 <- read.csv(file='< location where file is stored >', header=TRUE, sep=",")
sampweights <- RECS15$NWEIGHT
brrwts <- RECS15[, grepl("^BRRWT", names(RECS15))]
des <- svrepdesign(weights=sampweights, repweights=brrwts, type="Fay",
                   rho=0.5, mse=TRUE, data=RECS15)
```

Specify the sampling design: no replicate weights provided

- Specifying the sampling design when you don't have replicate weights
- This creates a `tbl_svy` object that then correctly calculates weighted estimates and SEs using methods from Workshop 1 and 2

```
as_survey_design(
   .data,
   ids = NULL,#cluster IDs/PSUs
   strata = NULL,#strata variables
   variables = NULL,#defaults to all in .data
   fpc = NULL,#variables defining the fpc
   nest = FALSE,#TRUE/FALSE - relabel clusters to nest within strata
   check_strata = !nest, #check that clusters are nested in strata
   weights = NULL,# weight variable
   ...
)
```

Syntax for common designs

```
# simple random sample (SRS)
apisrs %>% as_survey_design(fpc = fpc)

# stratified sample
apistrat %>% as_survey_design(strata = stype, weights = pw)

# one-stage cluster sample
apiclus1 %>% as_survey_design(ids = dnum, weights = pw, fpc = fpc)

# two-stage cluster sample, weights computed from pop size
apiclus2 %>% as_survey_design(ids = c(dnum, snum), fpc = c(fpc1, fpc2))

# stratified, cluster design
apistrat %>% as_survey_design(ids = dnum, strata = stype, weights =pw, nest = TRUE)
```

- examples from `srvyr` help documentation

ANES Example

For weight	Use variance unit/PSU/cluster	and use variance stratum
V200010b	V200010c	V200010d

```
options(width=130)
library(tidyverse) # for tidyverse
library(here) # for file paths
library(srvyr) # for tidy survey analysis
anes <- read_rds(here("Data", "anes_2020.rds")) %>%
   mutate(Weight=V200010b/sum(V200010b)*231592693)

anes_des <- anes %>%
   as_survey_design(weights = Weight,
                    strata = V200010d,
                    ids = V200010c,
                    nest = TRUE)
summary(anes_des)
```

ANES Example (cont'd)

```
## Stratified 1 - level Cluster Sampling design (with replacement)
## With (101) clusters.
## Called via srvyr
## Probabilities:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 4.839e-06 2.657e-05 4.689e-05 7.688e-05 8.331e-05 3.895e-03
## Stratum Sizes:
##              1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29
## obs        167 148 158 151 147 172 163 159 160 159 137 179 148 160 159 148 158 156 154 144 170 146 165 147 169 165 172 133 157
## design.PSU   3   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
## actual.PSU   3   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
##             30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50
## obs        167 154 143 143 124 138 130 136 145 140 125 158 146 130 126 126 135 133 140 133 130
## design.PSU   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
## actual.PSU   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
## Data variables:
##  [1] "V200010b"                "V200010d"                "V200010c"                "V200002"
##  [5] "V201006"                 "V201102"                 "V201101"                 "V201103"
##  [9] "V201025x"                "V201231x"                "V201233"                 "V201237"
## [13] "V201507x"                "V201510"                 "V201549x"                "V201600"
## [17] "V201617x"                "V202066"                 "V202109x"                "V202072"
## [21] "V202073"                 "V202110x"                "InterviewMode"           "Weight"
## [25] "Stratum"                 "VarUnit"                 "Age"                     "AgeGroup"
## [29] "Gender"                  "RaceEth"                 "PartyID"                 "Education"
## [33] "Income"                  "Income7"                 "CampaignInterest"        "TrustGovernment"
## [37] "TrustPeople"             "VotedPres2016"           "VotedPres2016_selection" "VotedPres2020"
## [41] "VotedPres2020_selection" "EarlyVote2020"
```

RECS Example

- Final weights: NWEIGHT Replicate weights: BRRWT1 – BRRWT96

```
options(width=130)
recs <- read_rds(here("Data", "recs.rds"))

recs_des <- recs %>%
   as_survey_rep(weights=NWEIGHT,
                 repweights=starts_with("BRRWT"),
                 type="Fay",
                 rho=0.5,
                 mse=TRUE)

summary(recs_des)
```

RECS Example (cont'd)

```
## Call: Called via srvyr
## Fay's variance method (rho= 0.5 ) with 96 replicates and MSE variances.
## Sampling variables:
##  - repweights: `BRRWT1 + BRRWT2 + BRRWT3 + BRRWT4 + BRRWT5 + BRRWT6 + BRRWT7 + BRRWT8 + BRRWT9 + BRRWT10 + BRRWT11 + BRRWT12 + BRRWT13
##  - weights: NWEIGHT
## Data variables: DOEID (dbl), Region (fct), Division (fct), MSAStatus (fct), Urbanicity (fct), HousingUnitType (fct), YearMade
##   (ord), SpaceHeatingUsed (lgl), HeatingBehavior (fct), WinterTempDay (dbl), WinterTempAway (dbl), WinterTempNight (dbl), ACUsed
##   (lgl), ACBehavior (fct), SummerTempDay (dbl), SummerTempAway (dbl), SummerTempNight (dbl), TOTCSQFT (dbl), TOTHSQFT (dbl),
##   TOTSQFT_EN (dbl), TOTUCSQFT (dbl), TOTUSQFT (dbl), NWEIGHT (dbl), BRRWT1 (dbl), BRRWT2 (dbl), BRRWT3 (dbl), BRRWT4 (dbl),
##   BRRWT5 (dbl), BRRWT6 (dbl), BRRWT7 (dbl), BRRWT8 (dbl), BRRWT9 (dbl), BRRWT10 (dbl), BRRWT11 (dbl), BRRWT12 (dbl), BRRWT13
##   (dbl), BRRWT14 (dbl), BRRWT15 (dbl), BRRWT16 (dbl), BRRWT17 (dbl), BRRWT18 (dbl), BRRWT19 (dbl), BRRWT20 (dbl), BRRWT21 (dbl),
##   BRRWT22 (dbl), BRRWT23 (dbl), BRRWT24 (dbl), BRRWT25 (dbl), BRRWT26 (dbl), BRRWT27 (dbl), BRRWT28 (dbl), BRRWT29 (dbl), BRRWT30
##   (dbl), BRRWT31 (dbl), BRRWT32 (dbl), BRRWT33 (dbl), BRRWT34 (dbl), BRRWT35 (dbl), BRRWT36 (dbl), BRRWT37 (dbl), BRRWT38 (dbl),
##   BRRWT39 (dbl), BRRWT40 (dbl), BRRWT41 (dbl), BRRWT42 (dbl), BRRWT43 (dbl), BRRWT44 (dbl), BRRWT45 (dbl), BRRWT46 (dbl), BRRWT47
##   (dbl), BRRWT48 (dbl), BRRWT49 (dbl), BRRWT50 (dbl), BRRWT51 (dbl), BRRWT52 (dbl), BRRWT53 (dbl), BRRWT54 (dbl), BRRWT55 (dbl),
##   BRRWT56 (dbl), BRRWT57 (dbl), BRRWT58 (dbl), BRRWT59 (dbl), BRRWT60 (dbl), BRRWT61 (dbl), BRRWT62 (dbl), BRRWT63 (dbl), BRRWT64
##   (dbl), BRRWT65 (dbl), BRRWT66 (dbl), BRRWT67 (dbl), BRRWT68 (dbl), BRRWT69 (dbl), BRRWT70 (dbl), BRRWT71 (dbl), BRRWT72 (dbl),
##   BRRWT73 (dbl), BRRWT74 (dbl), BRRWT75 (dbl), BRRWT76 (dbl), BRRWT77 (dbl), BRRWT78 (dbl), BRRWT79 (dbl), BRRWT80 (dbl), BRRWT81
##   (dbl), BRRWT82 (dbl), BRRWT83 (dbl), BRRWT84 (dbl), BRRWT85 (dbl), BRRWT86 (dbl), BRRWT87 (dbl), BRRWT88 (dbl), BRRWT89 (dbl),
##   BRRWT90 (dbl), BRRWT91 (dbl), BRRWT92 (dbl), BRRWT93 (dbl), BRRWT94 (dbl), BRRWT95 (dbl), BRRWT96 (dbl), CDD30YR (dbl), CDD65
##   (dbl), CDD80 (dbl), ClimateRegion_BA (fct), ClimateRegion_IECC (fct), HDD30YR (dbl), HDD65 (dbl), HDD50 (dbl), GNDHDD65 (dbl),
##   BTUEL (dbl), DOLLAREL (dbl), BTUNG (dbl), DOLLARNG (dbl), BTULP (dbl), DOLLARLP (dbl), BTUFO (dbl), DOLLARFO (dbl), TOTALBTU
##   (dbl), TOTALDOL (dbl), BTUWOOD (dbl), BTUPELLET (dbl)
## Variables:
##   [1] "DOEID"             "Region"            "Division"          "MSAStatus"         "Urbanicity"
##   [6] "HousingUnitType"   "YearMade"          "SpaceHeatingUsed"  "HeatingBehavior"   "WinterTempDay"
##  [11] "WinterTempAway"    "WinterTempNight"   "ACUsed"            "ACBehavior"        "SummerTempDay"
##  [16] "SummerTempAway"    "SummerTempNight"   "TOTCSQFT"          "TOTHSQFT"          "TOTSQFT_EN"
##  [21] "TOTUCSQFT"         "TOTUSQFT"          "NWEIGHT"           "BRRWT1"            "BRRWT2"
```

Create Survey Design Object for ACS

Fill in the blanks

- Analysis weight: PWGTP
- replicate weights: PWGTP1-PWGTP180
- jackknife with scale adjustment of 4/80

```
acs_des <- acs_pums %>%
   as_survey_rep(
      weights=___________,
      repweights=___________,
      type=___________,
      scale=_________
   )
```

Create Survey Design Object for ACS

Fill in the blanks

- Analysis weight: PWGTP
- replicate weights: PWGTP1-PWGTP180
- jackknife with scale adjustment of 4/80

```
acs_des <- acs_pums %>%
   as_survey_rep(
      weights=___________,
      repweights=___________,
      type=___________,
      scale=_________
   )
```

```
acs_des <- acs_pums %>%
   as_survey_rep(
      weights=PWGTP,
      repweights=stringr::str_c("PWGTP", 1:80),
      type="JK1",
      scale=4/80
   )
```

Create Survey Design Object for CPS 2011 Supplement

Fill in the blanks

- Analysis weight: wtsupp
- replicate weights: repwtp1 -repwtp160
- BRR

```
cps_des <- cps %>%
   as_survey_rep(
      weights=___________,
      repweights=___________,
      type=___________
   )
```

Create Survey Design Object for CPS 2011 Supplement

Fill in the blanks

- Analysis weight: wtsupp
- replicate weights: repwtp1 -repwtp160
- BRR

```
cps_des <- cps %>%
   as_survey_rep(
      weights=___________,
      repweights=___________,
      type=___________
   )
```

```
cps_des <- cps %>%
   as_survey_rep(
      weights=wtsupp,
      repweights=starts_with("repwtp"),
      type="BRR"
   )
```

Create Survey Design Object for NHANES

Fill in the blanks

- Analysis weight: WTINT2YR
- Variance Stratum: SDMVSTRA
- Variance Primary Sampling Unit: VPSU

```
nhanes_des <- nhanes %>%
   as_survey_design(
      weights=___________,
      ids=___________,
      strata=___________,
      fpc=___________
   )
```

Create Survey Design Object for NHANES

Fill in the blanks

- Analysis weight: WTINT2YR
- Variance Stratum: SDMVSTRA
- Variance Primary Sampling Unit: VPSU

```
nhanes_des <- nhanes %>%
   as_survey_design(
      weights=___________,
      ids=___________,
      strata=___________,
      fpc=___________
   )
```

```
nhanes_des <- nhanes %>%
   as_survey_design(
      weights=WTINT2YR,
      ids=VPSU,
      strata=SDMVSTRA,
      fpc=NULL
   )
```

Create Survey Design Object for LEMAS 2016

Fill in the blanks

- Analysis weight: ANALYSISWEIGHT
- Variance Stratum: STRATA
- FPC: FRAMESIZE

```
lemas_des <- lemas %>%
   as_survey_design(
      weights=___________,
      ids=___________,
      strata=___________,
      fpc=___________
   )
```

Create Survey Design Object for LEMAS 2016

Fill in the blanks

- Analysis weight: ANALYSISWEIGHT
- Variance Stratum: STRATA
- FPC: FRAMESIZE

```
lemas_des <- lemas %>%
   as_survey_design(
      weights=___________,
      ids=___________,
      strata=___________,
      fpc=___________
   )
```

```
lemas_des <- lemas %>%
   as_survey_design(
      weights=ANALYSISWEIGHT,
      ids=1,
      strata=STRATA,
      fpc=FRAMESIZE
   )
```

Breakout rooms: Practice time

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

Creating replicate weights

Creating replicate weights syntax

- Begin with a design object (e.g. `tsl_des`) and then create replicate weights
- "auto" uses JKn for stratified, JK1 for unstratified designs
- See help file for `survey::svrepdesign` for more information on replicate weight types

```
tsl_des %>%
   as_survey_rep(
      type = c("auto", "JK1", "JKn", "BRR", "bootstrap", "subbootstrap", "mrbbootstrap", "Fay"),
      ...
   )
```

Create Replicate Weights: example 1 (jackknife)

- Since this is not stratified, automatically used JK1

```
data(api)
dclus1 <- apiclus1 %>% as_survey_design(ids = dnum, weights = pw, fpc = fpc)
rclus1 <- as_survey_rep(dclus1)
summary(rclus1)
```

```
## Call: Called via srvyr
## Unstratified cluster jacknife (JK1) with 15 replicates.
## Data variables: cds (chr), stype (fct), name (chr), sname (chr), snum (dbl), dname (chr), dnum (int), cname (chr), cnum (int),
##   flag (int), pcttest (int), api00 (int), api99 (int), target (int), growth (int), sch.wide (fct), comp.imp (fct), both (fct),
##   awards (fct), meals (int), ell (int), yr.rnd (fct), mobility (int), acs.k3 (int), acs.46 (int), acs.core (int), pct.resp (int),
##   not.hsg (int), hsg (int), some.col (int), col.grad (int), grad.sch (int), avg.ed (dbl), full (int), emer (int), enroll (int),
##   api.stu (int), fpc (dbl), pw (dbl)
## Variables:
##  [1] "cds"      "stype"    "name"     "sname"    "snum"     "dname"    "dnum"     "cname"    "cnum"     "flag"     "pcttest"
## [12] "api00"    "api99"    "target"   "growth"   "sch.wide" "comp.imp" "both"     "awards"   "meals"    "ell"      "yr.rnd"
## [23] "mobility" "acs.k3"   "acs.46"   "acs.core" "pct.resp" "not.hsg"  "hsg"      "some.col" "col.grad" "grad.sch" "avg.ed"
## [34] "full"     "emer"     "enroll"   "api.stu"  "fpc"      "pw"
```

Create Replicate Weights: example 2 (bootstrap)

- Specifying bootstrap weights

```
bclus1 <- as_survey_rep(dclus1, type="bootstrap", replicates=100)
summary(bclus1)
```

```
## Call: Called via srvyr
## Survey bootstrap with 100 replicates.
## Data variables: cds (chr), stype (fct), name (chr), sname (chr), snum (dbl), dname (chr), dnum (int), cname (chr), cnum (int),
##   flag (int), pcttest (int), api00 (int), api99 (int), target (int), growth (int), sch.wide (fct), comp.imp (fct), both (fct),
##   awards (fct), meals (int), ell (int), yr.rnd (fct), mobility (int), acs.k3 (int), acs.46 (int), acs.core (int), pct.resp (int),
##   not.hsg (int), hsg (int), some.col (int), col.grad (int), grad.sch (int), avg.ed (dbl), full (int), emer (int), enroll (int),
##   api.stu (int), fpc (dbl), pw (dbl)
## Variables:
##  [1] "cds"      "stype"    "name"     "sname"    "snum"     "dname"    "dnum"     "cname"    "cnum"     "flag"     "pcttest"
## [12] "api00"    "api99"    "target"   "growth"   "sch.wide" "comp.imp" "both"     "awards"   "meals"    "ell"      "yr.rnd"
## [23] "mobility" "acs.k3"   "acs.46"   "acs.core" "pct.resp" "not.hsg"  "hsg"      "some.col" "col.grad" "grad.sch" "avg.ed"
## [34] "full"     "emer"     "enroll"   "api.stu"  "fpc"      "pw"
```

Creating analysis variables

Best practices

Overview

- Terminology: Analysis variable, constructed variable, derived variable, recoded variables
- Variables created from other variables
- Examples:
    - Creating a categorical variable from a continuous variable: Creating a categorical income variable from a continuous variable
    - Collapsing levels of a categorical variable: Collapsing a 5-level party identification variable into 3 levels
    - Creating a construct one or more variables: Binge drinking is defined as men who consumer 5 or more drinks in one sitting OR women who consume 4 or more drinks in one sitting.
- Best practice to create code to check your variable was created as intend

Code example - creating categorical variable

V201507x is respondent age: -9=Refused

```
anes_age <- anes_in %>%
   mutate(
      Age = if_else(V201507x > 0, as.numeric(V201507x), NA_real_),
      AgeGroup = cut(Age, c(17, 29, 39, 49, 59, 69, 200),
                     labels = c("18-29", "30-39", "40-49", "50-59", "60-69", "70 or older")))

anes_age %>%
   group_by(AgeGroup) %>%
   summarise(
      minAge = min(Age),
      maxAge = max(Age),
      minV = min(V201507x),
      maxV = max(V201507x),
      NAV= sum(is.na(V201507x)),
      NAAge=sum(is.na(Age)),
      N=n()
   )
```

Code example - creating categorical variable: output

```
## # A tibble: 7 × 8
##   AgeGroup    minAge maxAge             minV                    maxV   NAV NAAge     N
##   <fct>        <dbl>  <dbl>         <dbl+lbl>                <dbl+lbl> <int> <int> <int>
## 1 18-29           18     29 18                29                            0     0   871
## 2 30-39           30     39 30                39                            0     0  1241
## 3 40-49           40     49 40                49                            0     0  1081
## 4 50-59           50     59 50                59                            0     0  1200
## 5 60-69           60     69 60                69                            0     0  1436
## 6 70 or older     70     80 70                80 [80. Age 80 or older]      0     0  1330
## 7 <NA>            NA     NA -9 [-9. Refused] -9 [-9. Refused]               0   294   294
```

Code example - collapsing levels

V202073 indicates who the person voted for

```
count(anes_in, V202073)
```

```
## # A tibble: 12 × 2
##                                       V202073     n
##                                      <dbl+lbl> <int>
##  1 -9 [-9. Refused]                                53
##  2 -6 [-6. No post-election interview]              4
##  3 -1 [-1. Inapplicable]                         1497
##  4  1 [1. Joe Biden]                             3267
##  5  2 [2. Donald Trump]                          2462
##  6  3 [3. Jo Jorgensen]                            69
##  7  4 [4. Howie Hawkins]                           23
##  8  5 [5. Other candidate {SPECIFY}]               56
##  9  7 [7. Specified as Republican candidate]        1
## 10  8 [8. Specified as Libertarian candidate]       3
## 11 11 [11. Specified as don't know]                 2
## 12 12 [12. Specified as refused]                   16
```

Code example - collapsing levels

Recode V202073 as Biden, Trump, Other, and missing for unknown/no one

```
anes_vote <- anes_in %>%
   mutate(VotedPres2020_selection = factor(
      case_when(
         V202073 == 1 ~ "Biden",
         V202073 == 2 ~ "Trump",
         V202073 >= 3~ "Other",
         TRUE ~ NA_character_
      ),
      levels = c("Biden", "Trump", "Other")))

anes_vote %>% count(VotedPres2020_selection, V202073)
```

Code example - collapsing levels: output

```
## # A tibble: 12 × 3
##    VotedPres2020_selection                                V202073     n
##    <fct>                                                  <dbl+lbl> <int>
##  1 Biden                    1 [1. Joe Biden]                          3267
##  2 Trump                    2 [2. Donald Trump]                       2462
##  3 Other                    3 [3. Jo Jorgensen]                         69
##  4 Other                    4 [4. Howie Hawkins]                        23
##  5 Other                    5 [5. Other candidate {SPECIFY}]            56
##  6 Other                    7 [7. Specified as Republican candidate]     1
##  7 Other                    8 [8. Specified as Libertarian candidate]    3
##  8 Other                   11 [11. Specified as don't know]              2
##  9 Other                   12 [12. Specified as refused]                16
## 10 <NA>                    -9 [-9. Refused]                             53
## 11 <NA>                    -6 [-6. No post-election interview]           4
## 12 <NA>                    -1 [-1. Inapplicable]                      1497
```

Code example - collapsing levels - fix

Recode V202073 as Biden, Trump, Other, and missing for unknown/no one

```
anes_vote <- anes_in %>%
   mutate(VotedPres2020_selection = factor(
      case_when(
         V202073 == 1 ~ "Biden",
         V202073 == 2 ~ "Trump",
         V202073 >= 3 & V202073 <= 8~ "Other",
         TRUE ~ NA_character_
      ),
      levels = c("Biden", "Trump", "Other")))

anes_vote %>% count(VotedPres2020_selection, V202073)
```

Code example - collapsing levels: output

```
## # A tibble: 12 × 3
##    VotedPres2020_selection                                V202073     n
##    <fct>                                                  <dbl+lbl> <int>
##  1 Biden                    1 [1. Joe Biden]                          3267
##  2 Trump                    2 [2. Donald Trump]                       2462
##  3 Other                    3 [3. Jo Jorgensen]                         69
##  4 Other                    4 [4. Howie Hawkins]                        23
##  5 Other                    5 [5. Other candidate {SPECIFY}]            56
##  6 Other                    7 [7. Specified as Republican candidate]     1
##  7 Other                    8 [8. Specified as Libertarian candidate]    3
##  8 <NA>                    -9 [-9. Refused]                             53
##  9 <NA>                    -6 [-6. No post-election interview]           4
## 10 <NA>                    -1 [-1. Inapplicable]                      1497
## 11 <NA>                    11 [11. Specified as don't know]             2
## 12 <NA>                    12 [12. Specified as refused]               16
```

Code example - creating construct

- Creating poverty level indicator from household size and income for Durham County, NC
- Data source: 2019 1-year ACS microdata
- In NC (and most states), poverty guideline is as follows:

Persons in Household	Poverty guideline
1	$12,490
2	$16,910
3	$21,330
4	$25,750
5	$30,170
6	$34,590
7	$39,010
8	$43,430
9+	Add $4,420 for each additional person

Code example - creating construct

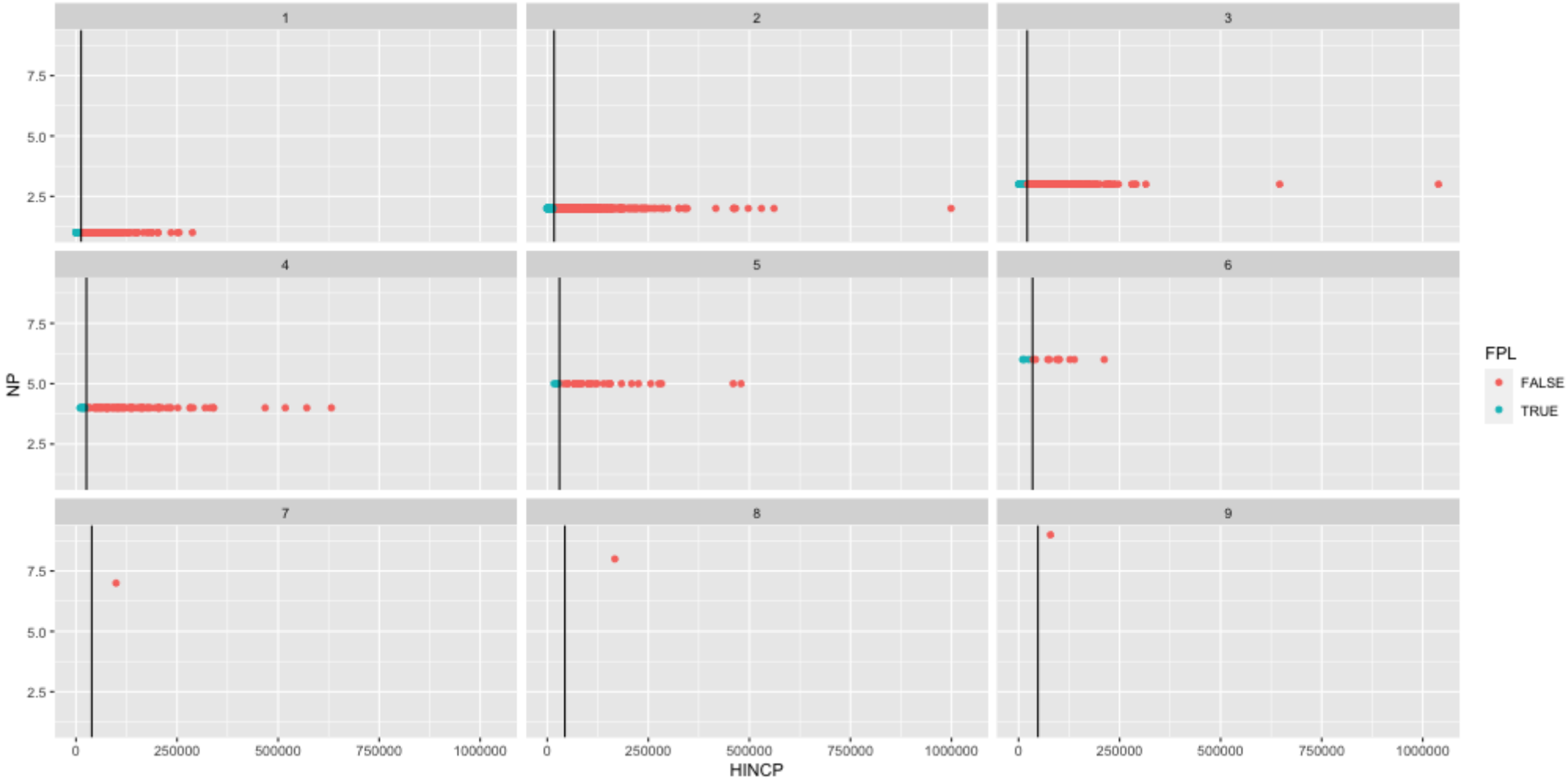
NP is the number of persons in a household, HINCP is the household income

```
dat19_pov <- dat19_in %>%
   mutate(PovGuide=case_when(
      NP==1~12490,
      NP==2~16910,
      NP==3~21330,
      NP==4~25750,
      NP==5~30170,
      NP==6~34590,
      NP==7~39010,
      NP==8~43430,
      NP>=9~43430+(NP-8)*4420
   ),
   FPL=HINCP<=PovGuide
   )

dat19_pov %>%
   count(NP, PovGuide)

p <- dat19_pov %>% ggplot(aes(x=HINCP, y=NP, colour=FPL)) +
   facet_wrap(~NP) + geom_point() + geom_vline(aes(xintercept=PovGuide))
```

Code example - creating construct: output

```
## # A tibble: 9 × 3
##      NP PovGuide     n
##   <dbl>    <dbl> <int>
## 1     1    12490   363
## 2     2    16910   367
## 3     3    21330   152
## 4     4    25750    96
## 5     5    30170    35
## 6     6    34590    13
## 7     7    39010     1
## 8     8    43430     1
## 9     9    47850     1
```

Code example - creating construct: output



Breakout rooms: Practice time

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

Reproducible research

Reproducible research overview

Someone with the same data should be able to reproduce the same results

- Tools to help this
  - R projects
  - here package
  - R Markdown
- Processes to help this
  - Batching code
  - Be organized – create documentation and a clear folder structure
  - Version control

R projects and the here package

- R projects specify the root folder and other R options
- Stop doing this: `setwd("C:\Users\zimmers\Documents\tidy-survey-short-course")`
- here package makes relative paths easy: Relative from where .Rproj file is or current file (if no project)
- here package makes sure to create path correctly for OS (e.g. \ for Windows and / for Linux/Mac)
- Example

```
list.files(here())
```

```
##  [1] "Codebooks"                    "Data"                         "DataCleaningScripts"
##  [4] "Exercises"                    "FinalizeMaterials.R"          "LICENSE"
##  [7] "Presentation"                 "RawData"                      "README.md"
## [10] "tidy-survey-short-course.Rproj" "xaringan-themer.css"
```

```
list.files(here("RawData", "RECS_2015"))
```

```
## [1] "2020_RECS-457A.pdf"     "codebook_publicv4.xlsx" "microdata_v3.pdf"       "README.md"
```

R Markdown

- R Markdown combines R code with text
- Each time document is Knitted, a self-contained session is started.
    - Prevents problems with depending on something in your environment that aren't explicitly called out
- Knit to PDF, DOCX, HTML, PPTX, and more
- Don't copy/paste output to your manuscript/report. Make your manuscript/report with R Markdown
- Automatic table/figure numbering. If using Word, check out `officedown` and `officer`
- Can create parameterized reports. Example: run an analysis for each state and each state gets a report
- For beginners: https://rmarkdown.rstudio.com/lesson-1.html

Batching R code

OK, you want to stick with .R files. What can you do?

- Compiling R Scripts
  - In RStudio, use the Compile Report feature under File menu. Create output from your code and code runs in self-contained session
  - In code, use `rmarkdown::render(filename.R)`
  - Creates HTML, PDF, or Word document with your code, console output, and plots
- Batch from command line (Terminal)
  - Linux: `R CMD BATCH --no-save filename.R &`
  - Windows (something like): `"C:\Program Files\R\R-4.1.3\bin\R.exe CMD BATCH --no-save filename.R &`
  - Creates a .Rout file with your console output, timing information, and plots in PDFs (unless saved another way). .Rout file can be viewed in a text editor of your choice or Word

Documentation and organization

- Create a README file
- Set up folders in an easy to follow manner
- Example set-up

```
recs-analysis
└──Analysis
    |   01_ProcessData.Rmd
    |   01_ProcessData.html
    |   02_EDA.Rmd
    |   02_EDA.html
└──Data
|   └──Raw
|   |   codebook_publicv4.xlsx
|   |   microdata_v3.pdf
|   |   recs2015_public_v4.csv
|   └──Analysis
|       |   recs.rds
|   README.md
|   recs-analysis.Rproj
```

Version control

- Version control is a process to track and manage changes in code
- Common method (and has integration with RStudio) is GitHub
- Document why you change analysis over time
- Collaborate with others
- Resource to **Git** started: https://happygitwithr.com/

Useful packages for tables

- kableExtra: extends `kable` to allow piping for HTML and LaTeX
- gt: from tibble/data.frame to nice looking tables for HTML, LaTeX, and RTF
- gtsummary: tbl_svysummary creates tables of summary statistics from survey objects
- flextable: tables for HTML, PDF, Word, and Powerpoint
- huxtable: tables for LaTeX and HTML

Other useful packages

- ggsurvey: plotting data from surveys
- naniar: visualize missing data and see missing patterns
- likert: analyze and visualize Likert type items
- and more CRAN Task View: Official Statistics & Survey Statistics

Closing

General questions

- Open floor for questions and discussion

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!

- You will be receiving a follow-up survey to share feedback about course

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)
##  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)
##  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)
##  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)
##  tidycensus * 1.1.2   2022-03-09 [1] CRAN (R 4.1.2)
##  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
```