Coding Supplement: Weighted Survey Analysis

Please remember: the commands given below are guides only. For example, you will need to insert appropriate variable names where it says VARIABLE. Also, think about what kind of variable each one is (continuous vs categorical) and how you might treat it differently in your analysis. Finally, cutting and pasting code from Word into Stata or R sometimes results in errors. We suggest typing in the code yourself.

Section A: Stata

Setting Survey Specifications

You need to use the **svyset** command to tell Stata that you are analyzing weighted survey data. The command is used to specify design characteristics. These will continue to be used until they are cleared or a new dataset is loaded. There are many characteristics you can specify – these are beyond what is needed for this class, but you can explore the Stata manuals available online for more details.

Homework Hint
For the NHIS 2022 dataset: svyset [pweight = wtfa_a], strata(pstrat) psu(ppsu)

Calculations

All weighted survey data analysis commands in Stata are done using the "**svy:**" prefix.

- As a default, Stata incorporates the design effect (DEFF) into calculations

A. Proportion

svy: proportion VARIABLE: Automatically gives you standard errors/confidence intervals.

- Specifying more than 1 variable gives proportions for each separately, not a "two-way" table like "tab".
- To get a two-way table, you would specify "**svy: proportion VARIABLE, over(VARIABLE)**"
- To get the design effect, follow the svy: proportion command with "**estat effects**"

svy: tab VARIABLE:

- Need to specify "se" and "ci" if you want those (example: "**svy: tab VARIABLE, se ci**").
- Specify "row" if you want row proportions to add up to 1, or 100% (default is for columns add up to 1).
- To get the design effect, just add "deff" (example: "**svy: tab VARIABLE, se ci deff**")

Homework Hint: For unweighted data, using **tab** gives you "percents" to two decimal places, which can be converted (by dividing by 100) to "proportions". In contrast, **svy: tab** automatically gives you "proportions".

B. Mean

svy: mean VARIABLE: Automatically gives you standard errors/confidence intervals.

- Can further specify strata using "over"

C. Median (*not needed for homework assignment*): there is no "median" command

- Sort file from low to high on your variable of interest (y).
- Cumulate weights until 50% is reached.
- The value of y for the unit that is the first one to have a cumulative of 50% or more in weight is the estimated median.
- You could do the same thing for the 25th or 75th percentile.

D. Subpopulations: (*i.e.*, only look at people over 6 feet tall in your data, in a variable "tall")

- Note: Need to use "**subpop**". Do not use "**if**" as a stand-alone restriction (*i.e.,* svy: proportion VARIABLE if tall==1) as this *deletes* the non-relevant entries, but you need those to properly calculate standard errors in weighted data.
- "Subpop" will restrict on that variable where the value does not equal 0 or missing
  - Example: **svy, subpop(tall): proportion VARIABLE**
  - This will restrict to individuals who are coded in your dataset as "tall==1"
  - If your variables are not coded as 0 and 1, you can use an "if" statement within subpop **(svy, subpop(if tall==1 & sex==2): proportion VARIABLE)**
- For multiple restrictions/stratifications/subpopulations
  - Given a dataset with variables: tall (binary), age (continuous), and sex (binary)
  - Example: **svy, subpop(tall): mean age, over(sex)**
    - This will give you mean age for 2 strata (sex==1 & sex==0) only for those for whom tall==1.
  - Example: **svy: mean age, over (sex tall)**
    - This will give you mean age for 4 strata (tall==1 & sex==1; tall==1 & sex==0; tall==0 & sex==1, tall==0 & sex==0)

E. Graphs

- Histograms: You specify weights using "**fw**"
  - Your weights must be integers.
  - Example: gen NEW_WEIGHT_VARIABLE=int(WEIGHT_VARIABLE)
    histogram VARIABLE [fw=NEW_WEIGHT_VARIABLE], bin(20)
- Boxplots: Your weight is specified using "**pw**"
  - These weights don't need to be integers
  - Example: graph box VARIABLE1 [pw=WEIGHT_VARIABLE], by(VARIABLE2)
- Bar Graphs: Your weight is specified using "**pw**"
  - To get proportions by categories of a variable, you need to create a variable for each subcategory.
    - If you want to create a bar graph by sex (as a binary variable of males & females), you need to create a variable where sex==1 and another where sex==2.
    - Example: **tab sex, gen(class)** will generate two new variables, class1 where sex==1 & class2 where sex==2

- You can then create a bar graph: **graph bar class1 class2 [pw=wtfa]**
- You can create further strata using "over"
    - Imposing a gap between the two strata makes the graph easier to read
    - Example: **graph bar class1 class2 [pw=wtfa], over(tall, gap(*2))**

F. Regression Models:

- Can use "**glm**" with the **svy**: prefix
    - Specify family is binomial for logistic regression, logit link
    - "**logit**" and "**logistic**" also work
    - Example: **svy: glm OUTCOME EXPOSURE COVARIATES, family(binomial) link(logit)**
    - Example: **svy: logit OUTCOME EXPOSURE COVARIATES**

Section B: R

Setting Survey Specifications
You will need the "survey" and "sandwich" packages to do this.

You create a survey design object, which then gets specified in future commands.

- Example: [design name] <- svydesign(id=~ppsu, strata=~pstrat, weights=~wtfa_a, data=[insert name], nest=TRUE)
    - "nest" tells you if clusters are numbered uniquely within a stratum (i.e., start back at "1" in the next stratum)

A. Proportions/Means

- svymean gives you proportions/means with SE.
    - Example: svymean(~VARIABLE, design=DESIGN NAME, na.rm=TRUE, deff=TRUE)
    - In R, you need to specify the design effect (DEFF)
    - For binary variables coded 0/1, you need specify "factor(VARIABLE)"
- svyby is the general command for two-way tables. You can specify the "function" (svymean). You will specify the "design". ci=TRUE gives you the confidence intervals.
    - Example: svyby (~VARIABLE, ~VARIABLE, FUN=svymean, design=DESIGN NAME, ci=TRUE, deff=TRUE)
    - To get the Rao-Scott corrected chi-square statistic: svychisq (~VARIABLE + VARIABLE, design=DESIGN NAME, na.rm=TRUE, statistic="F")

B. Subpopulations: You need to create a "subset" data frame. You can do this within the svyby command or as a separate command.

C. Graphs:

Note: Using ggplot2 would be ideal. So far, the best method I am aware of for this is to create a data frame with the appropriately calculated weighted proportions, and then graph that information using ggplot.

Below are some simpler graphing commands where the weighted proportions are calculated directly.

- Histograms:
    - svyhist(~VARIABLE, design=DESIGN NAME, freq=FALSE)
        - freq=FALSE tells it to give densities instead of counts
- Boxplots: Needs to be a "two-sided" formula
    - Example: svyboxplot(VARIABLE1~VARIABLE2, design=DESIGN NAME)
    - For one variable, you can use: VARIABLE~1
- Bar Graphs:
    - Create an object with the proportion for each subcategory
    - Example: obj <- svyby(~VARIABLE1, by=~VARIABLE2, design=DESIGN NAME)
    - Then, create the graph: barplot(obj, legend.text=TRUE)

R has created the package "ggsurvey", which also directly incorporates survey weights. I have not fully mastered this package, but I encourage you to explore it if you are interested.

D. Regression Models:

- Can use "glm"
    - Specify family is binomial for logistic regression, logit link
    - Can avoid a warning message with family=quasibinomial
    - The command for weighted data is svyglm
    - Example: svyglm(OUTCOME~EXPOSURE+COVARIATES, design=DESIGN NAME, family=quasibinomial(link="logit"))

WEIGHTING

Note: You do not need this for the assignment. This is for your information only.

Method: Logistic Regression

Stata:
logit Survey educ race age
predict p, asif
(asif will calculate predictions for all observations, even if missing data precludes it from being used in the regression model)

R:
model <- glm (formula=Survey ~ educ+race+age, data=mydata, family=binomial(link="logit"))
predict (model, data=mydata, type="response")

Method: Raking

Stata:
Need to install survwgt
Survey dataset:

- If you don't have an existing design weight, define old_wt=1
- Generate subgroups for your variables of interest
  - Example: sex_surv=% female and % male in your survey population
  - Variables of interest include all of the characteristics on which you plan to rake

Create target population variables with percent breakdowns for the various categories

- Example: sex_perc = % female and % male in the full population

Generate a "total" variable

- This must add up to exactly the same number as in your survey population
- Example: tot_sex=round(sex_perc * [sample size])

Conduct the raking

- Example: survwgt rake old_wt, by(age_surv sex_surv ed_surv…) totvars(tot_age tot_sex…) gen(final_weight)
- final_wt is your new raked weight for each observation

R:
Packages: weights, anesrake
Weights: get relative percentages in survey data

- Example: wpct(survdata$age_surv)

Create "targets" of percent breakdowns for each variable in our target population

- Example: agetarg <- c(0.365, 0.260…)
  names(agetarg) <- c("20-29",…)

Create a list for your targets, with the SAME names as the corresponding variable in the survey dataset

- Example: targets <- list (agetarg…)

names(targets) <- c("age_surv"…)

Generate an ID variable

- Example: survdata$caseid=1:length(survdata$agex)

Raking: (there are many things you can specify)

- Example: outsave <- anesrake (targets, survdata, caseid=survdata$caseid, iterate=TRUE, force1=TRUE, pctlim=0.05)

Create a dataframe with your weights (weightvec is your weight variable)

- Example: caseweights <- data.frame(cases=outsave$caseid, weights=outsave$weightvec)