Variance estimation

Chapter 9 topics
Math 255

1

Some general methods of variance estimation

- Linearization
  - Approximate the statistic using a 1st order Taylor's series expansion (linear function)
  - Get a formula for SE
- Replication methods
  - Replication weights created to generate replicated estimates that mimic the actual variance of the estimate
  - Often created with replacement so they overestimate SEs.
  - Computationally intensive

2

Random Group Method

- Survey design is replicated (repeated) independently R times.
- Parameter of interest, θ, is estimated R times:

$$\hat{\theta}_1, \ldots, \hat{\theta}_R$$

- An estimator of θ is

$$\tilde{\theta} = \frac{1}{R}\sum_{r=1}^{R}\hat{\theta}_r$$

- The estimated variance is

$$\hat{V}(\tilde{\theta}) = \frac{1}{R}\frac{1}{R-1}\sum_{r=1}^{R}\left(\hat{\theta}_r - \tilde{\theta}\right)^2 = \frac{S^2_{\hat{\theta}}}{R}$$

3

Jackknife

- Extends random group method
- Delete-1 jackknife (done in each stratum)
  - Estimate the parameter of interest, θ, the usual way, denoted as $\hat{\theta}$.
  - Replication: Delete 1 PSU at a time, creating R=n replicates.
  - Jackknife variance can give same SE as the "usual" SE for known designs
- R can compute jackknife variance but design needs to be specified.

4

Jackknife

- Some of the mechanics:
  - When deleting PSU j, recompute weights

$$w^r_{i(j)} = \begin{cases} \frac{n}{n-1}w_i & \text{element } i \text{ is not in PSU } j \\ 0 & \text{element } i \text{ is in PSU } j \end{cases}$$

  - When deleting PSU j, use replicate weights to estimate the parameter of interest, θ

$$\hat{\theta}_{(1)}, \ldots, \hat{\theta}_{(n)}$$

  - The jackknife variance is estimated as

$$\hat{V}_{JK}(\hat{\theta}) = \frac{n-1}{n}\sum_{j=1}^{n}\left(\hat{\theta}_{(j)} - \hat{\theta}\right)^2$$

5

Jackknife in R

- Agstat data (n=300, H=4 region strata)
- First fit sampling design, then update to a replicated design.

```
> agstrat$N <- recode(agstrat$region, NC = 1054, NE = 220
S = 1382, W = 422)
> design.strat<- svydesign(id= ~1, fpc= ~N, weights= ~wei
t, strata= ~region, data=agstrat)
> agstrat.JK<- as.svrepdesign(design.strat,type="JKn")
> agstrat.JK
Call: as.svrepdesign(design.strat, type = "JKn")
Stratified cluster jackknife (JKn) with 300 replicates.
```

6

Jackknife in R

- Use replicate design to estimate parameters.
- Estimating average number of acres in 1992.

```
> svymean(~acres92, design.strat)
          mean    SE
acres92 295561 16380
> svymean(~acres92, agstrat.JK)
          mean    SE
acres92 295561 16380
```

- Standard SE (formula from ch. 3) and jackknife are equivalent for estimating a mean.

7

Jackknife in R

- Estimating ratio of 1992 total acreage to 1987 total acreage.

```
> svyratio(~acres92,~acres87,design.strat)
Ratio estimator: svyratio.survey.design2(~a
Ratios=
          acres87
acres92 0.9899971
SEs=
            acres87
acres92 0.006187757
> svyratio(~acres92,~acres87, agstrat.JK)
Ratio estimator: svyratio.svyrep.design(~ac
Ratios=
          acres87
acres92 0.9899971
SEs=
            [,1]
[1,] 0.006229751
```

- Standard SE (formula from ch. 3) and jackknife are not equivalent for estimating nonlinear functions like a ratio.

8

Why use jackknife SE?

- Pro: To simplify SE calculations
  - You can include jackknife weights, along with sampling weights, in your dataset
  - Users of your data only need to know basic weighted estimation formulas (HT) and the jackknife SE formula.
  - E.g. these are values can be computed in spreadsheet software like Excel.
- Con: jackknife replication size is large (n)

9

Jackknife weights

- For the jth PSU removal: element-level JK weights:

$$w_{i(j)}^{r} = \begin{cases} \dfrac{n}{n-1} w_i & \text{element } i \text{ is not in PSU } j \\ 0 & \text{element } i \text{ is in PSU } j \end{cases}$$

- In a stratified sample, this is done separately for each stratum
  - In NE, we have n=103 counties sampled
  - The JK weight adjustment to the sampling weights in NE is about 103/102=1.0098

10

Jackknife in R

- The first weight adjustments are for the NE stratum:

```
> 103/102
[1] 1.009804
> dim(agstrat.JK$repweights$weights)
[1] 300 300
> agstrat.JK$repweights$weights[,1:5]
         [,1]     [,2]     [,3]     [,4]     [,5]
  [1,] 0.000000 1.009804 1.009804 1.009804 1.009804
  [2,] 1.009804 0.000000 1.009804 1.009804 1.009804
  [3,] 1.009804 1.009804 0.000000 1.009804 1.009804
  [4,] 1.009804 1.009804 1.009804 0.000000 1.009804
  [5,] 1.009804 1.009804 1.009804 1.009804 0.000000
  [6,] 1.009804 1.009804 1.009804 1.009804 1.009804
```

11

Jackknife in R

- Replicate weight construction: Multiply sampling weights by the jackknife weight adjustments:

```
> # first replicate weights:
> wt.r1<- agstrat$weight*agstrat.JK$repweights$weights[,1]
> wt.r1
  [1]  0.00000 10.33333 10.33333 10.33333 10.33333 10.33333 10.33333
  [8] 10.33333 10.33333 10.33333 10.33333 10.33333 10.33333 10.33333
 [15] 10.33333 10.33333 10.33333 10.33333 10.33333 10.33333 10.33333
```

- Then use replicate weights to construct an estimate:

$$\hat{\theta}_{(1)} = 0.9903076$$

```
> # first replicated estimate of the ratio
> sum(wt.r1*agstrat$acres92)/sum(wt.r1*agstrat$acres87)
[1] 0.9903076
```

- This is done a total of n=300 times for this data set.

12

Bootstrap

- Treats sample as the population
- Basic idea
  - Estimate the parameter of interest, θ, the usual way.
  - Resampling: Randomly sample PSUs with replacement.
  - Compute the parameter estimate for each resample.
  - Repeat this R times (R needs to be big!)
  - Histogram of resampled estimates mimics shape and variability of the sampling distribution of the estimate.

13

Bootstrap

- Mechanics for the **rescaling bootstrap**:
  - Take a SRS of n-1 PSUs with replacement
  - Recompute weights for each resample r

$$w_i^r = w_i \frac{n}{n-1} \times (\#\text{ times PSU } i \text{ is seen in resample } r)$$

  - For each resample, use new weights to estimate the parameter of interest, θ

$$\hat{\theta}_1^*, \ldots, \hat{\theta}_r^*$$

  - The bootstrap variance is estimated as

$$\hat{V}_B(\hat{\theta}) = \frac{1}{R-1} \sum_{r=1}^{R} \left(\hat{\theta}_r^* - \hat{\theta}\right)^2$$

14

Bootstrap in R

- First fit sampling design, then update to a replicated design.
- Here R=1000 resamples (replicates) are taken

```
> agstrat.boot<-
as.svrepdesign(design.strat,type="subbootstrap",rep=1000)
> agstrat.boot
Call: as.svrepdesign(design.strat, type = "subbootstrap", rep = 1000)
(n-1) bootstrap with 1000 replicates.
```

15

Bootstrap in R

- Unlike the jackknife, bootstrap SEs will not be mathematically equivalent to the formula SE for mean:

```
> svymean(~acres92, design.strat)
          mean     SE
acres92 295561 16380
> svymean(~acres92, agstrat.boot)
          mean     SE
acres92 295561 17232
```

16

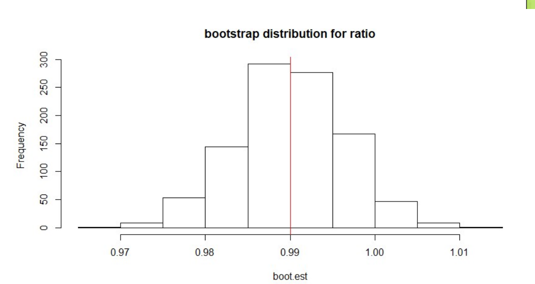
Bootstrap in R

- Ratio SE will also be different.

```
> svyratio(~acres92,~acres87,design.strat)
Ratio estimator: svyratio.survey.design2(~ac
design.strat)
Ratios=
          acres87
acres92 0.9899971
SEs=
            acres87
acres92 0.006187757
> svyratio(~acres92,~acres87, agstrat.JK)
Ratio estimator: svyratio.svyrep.design(~acr
agstrat.JK)
Ratios=
          acres87
acres92 0.9899971
SEs=
            [,1]
[1,] 0.006229751
> svyratio(~acres92,~acres87, agstrat.boot)
Ratio estimator: svyratio.svyrep.design(~acr
agstrat.boot)
Ratios=
          acres87
acres92 0.9899971
SEs=
            [,1]
[1,] 0.006578109
```

17

Bootstrap in R



18

Bootstrap in R

- R gives the bootstrap weight adjustments
- In the first (column) resample:
  - Unit 1: (103/102)x 1
  - Unit 2: (103/102)x 0
  - Unit 5: (103/102)x 2

```
> dim(agstrat.boot$repweights$weights)
[1]  300 1000
> agstrat.boot$repweights$weights[1:5,1:5]
         [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 1.009804 0.000000 1.009804 2.019608 0.000000
[2,] 0.000000 1.009804 2.019608 1.009804 0.000000
[3,] 0.000000 1.009804 0.000000 1.009804 2.019608
[4,] 0.000000 0.000000 1.009804 1.009804 1.009804
[5,] 2.019608 3.029412 1.009804 2.019608 3.029412
> 103/102; 2*103/102
[1] 1.009804
[1] 2.019608
```

19

Bootstrap

- Pros:
  - All-purpose method that is ok(ish) for quantiles
  - CIs can be produced from bootstrapped samples . (They mimic both the shape and variability of the sampling distribution.)
- Cons:
  - R (# of resamples) needs to be big to get an accurate estimate of SE
  - Doesn't produce the same SE value each time the algorithm is run

20

Summary

- Replication methods can give you SEs for estimators whose (true) SE are hard to derive analytically (formulaically)
- Need to know design features (stratification, clustering) to get these SEs in R.
- But, for large scale surveys, replicate weights are sometimes given as part of the data set.

21

American Community Survey (ACS)

- Ongoing survey covering entire U.S.
- Conducted by Census Bureau
- Gives communities and governments information about their populations.

https://www.census.gov/programs-surveys/acs

22

2009 PUMS

- PUMS = Public Use Microdata Sample
- Contains actual responses from the ACS
  - Some responses are adjusted for confidentiality
- The 2009 PUMS was designed to sample one percent of the housing units in the United States.

https://www.census.gov/programs-surveys/acs/data/pums.html

23

2009 PUMS

- PUMS is well documented:
  - Describes sampling weights
  - Describes replicate weights and SE calculation
- http://www.census.gov/content/dam/Census/library/publications/2009/acs/ACSPUMS.pdf

24

2009 PUMS: sampling weights

- Data is give at the household and person level.
  - http://math.carleton.edu/kstclair/data/ss09pmn.csv
- Using the person-level weights allows us to estimate the population size of MN in 2009:

```
> sum(acs$PWGTP)    # est. MN population size (2009)
[1] 5266215
> head(acs$PWGTP)
[1]  20   8 145 140 101  99
```

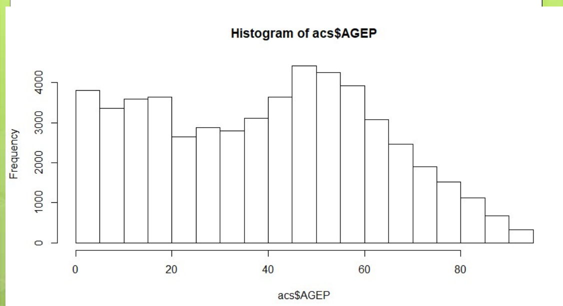
25

2009 PUMS: estimation

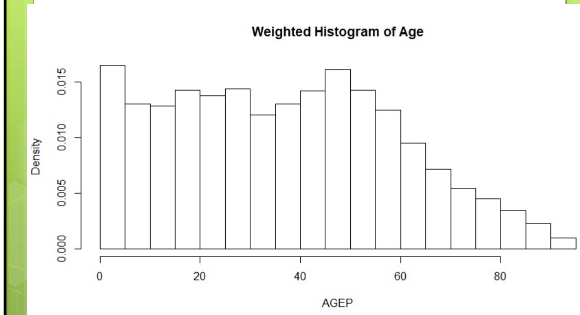
- Documentation:
  - Use HT type estimates for total, mean, or population sizes

$$\hat{t} = \sum_i w_i y_i \qquad \hat{\bar{y}} = \frac{\sum_i w_i y_i}{\sum_i w_i} \qquad \hat{N}_{Domain} = \sum_{i \text{ in } Domain} w_i$$

26



27



28

2009 PUMS: estimation

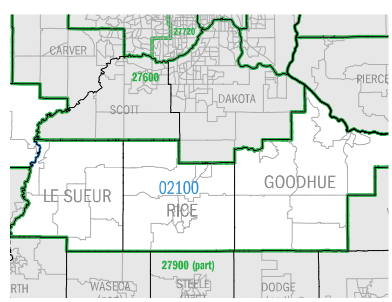
- Estimate average age of all people in MN is 37.24 years. (unweighted average is 39.98)

$$y_i = \text{ age of person } i, \qquad \hat{\bar{y}} = \frac{\sum_i w_i y_i}{\sum_i w_i}$$

```
> ## estimate average age in MN
> hist(acs$AGEP)
> w<- acs$PWGTP
> sum(w*acs$AGEP)/sum(w)
[1] 37.23876
```

29

Domain: PUM Area (PUMA) 2100



30

2009 PUMS: estimation

- Estimate average age in PUMA 2100:

$$x_i = 1 \text{ if person } i \text{ is in PUMA 2100},$$

$$\hat{\bar{y}}_{PUMA2100} = \frac{\sum_i w_i x_i y_i}{\sum_i w_i x_i} = \frac{\sum_{i \text{ in PUMA 2100}} w_i y_i}{\sum_{i \text{ in PUMA 2100}} w_i}$$

31

2009 PUMS: estimation

- Estimate average age in PUMA 2100 is 37.72 years.

```
> ## estimate average age in PUMA 2100
> x<- ifelse(acs$PUMA == 2100, 1, 0)
> sum(x)  # number of response
[1] 1740
> sum(x*w) # estimated number of residents
[1] 136712
> sum(w*x*acs$AGEP)/sum(w*x)
[1] 37.71997
```

32

2009 PUMS: replicate weights

- Two methods are provided for estimating the standard errors of PUMS estimates: replicate weights and design factors.
- The ACS employs the **Successive Differences Replication (SDR)** method (Wolter, 1984; Fay & Train, 1995; Judkins, 1990) to produce variance estimates.
- https://www2.census.gov/programs-surveys/acs/tech_docs/pums/accuracy/2009AccuracyPUMS.pdf?#

33

2009 PUMS: replicate weights

- Documentation:
  - Let θ be the population parameter
  - Let $\hat{\theta}$ be the estimate based on the sampling weights
  - Let $\hat{\theta}_r$ be the estimate based on the rth replicate weight
  - The SDR standard error of the estimate is

$$SE(\hat{\theta}) = \sqrt{\frac{4}{80}\sum_{r=1}^{80}\left(\hat{\theta}_r - \hat{\theta}\right)^2}$$

34

2009 PUMS: SE

- Find the replicate weights at the end of the PUMS data set:

```
176] "FMIGSP"   "FMILPP"   "FMILSP"   "FOCCP"    "FOIP"     "FPAP"     "FPOBP"
183] "FPOWSP"   "FRACP"    "FRELP"    "FRETP"    "FSCHGP"   "FSCHLP"   "FSCHP"
190] "FSEMP"    "FSEXP"    "FSSIP"    "FSSP"     "FWAGP"    "FWKHP"    "FWKLP"
197] "FWKWP"    "FWRKP"    "FYOEP"    "pwgtp1"   "pwgtp2"   "pwgtp3"   "pwgtp4"
204] "pwgtp5"   "pwgtp6"   "pwgtp7"   "pwgtp8"   "pwgtp9"   "pwgtp10"  "pwgtp11"
211] "pwgtp12"  "pwgtp13"  "pwgtp14"  "pwgtp15"  "pwgtp16"  "pwgtp17"  "pwgtp18"
218] "pwgtp19"  "pwgtp20"  "pwgtp21"  "pwgtp22"  "pwgtp23"  "pwgtp24"  "pwgtp25"
225] "pwgtp26"  "pwgtp27"  "pwgtp28"  "pwgtp29"  "pwgtp30"  "pwgtp31"  "pwgtp32"
232] "pwgtp33"  "pwgtp34"  "pwgtp35"  "pwgtp36"  "pwgtp37"  "pwgtp38"  "pwgtp39"
239] "pwgtp40"  "pwgtp41"  "pwgtp42"  "pwgtp43"  "pwgtp44"  "pwgtp45"  "pwgtp46"
246] "pwgtp47"  "pwgtp48"  "pwgtp49"  "pwgtp50"  "pwgtp51"  "pwgtp52"  "pwgtp53"
253] "pwgtp54"  "pwgtp55"  "pwgtp56"  "pwgtp57"  "pwgtp58"  "pwgtp59"  "pwgtp60"
260] "pwgtp61"  "pwgtp62"  "pwgtp63"  "pwgtp64"  "pwgtp65"  "pwgtp66"  "pwgtp67"
267] "pwgtp68"  "pwgtp69"  "pwgtp70"  "pwgtp71"  "pwgtp72"  "pwgtp73"  "pwgtp74"
274] "pwgtp75"  "pwgtp76"  "pwgtp77"  "pwgtp78"  "pwgtp79"  "pwgtp80"
```

35

2009 PUMS: SE

- First replicated mean estimate is 37.27156

$$\hat{\theta}_1 = 37.27156$$

```
> # first 5 sampling weights
> w[1:5]
[1]  20   8 145 140 101
> # first 5 relicate weights for r=1
> repwts<- as.matrix(acs[,200:279])
> dim(repwts)
[1] 53140    80
> repwts[1:5,1]
[1]  19  11 138 137  98
> # replicate mean est for r=1
> sum(repwts[,1]*acs$AGEP)/sum(repwts[,1])
[1] 37.27156
```

36

2009 PUMS: SE

- The average age in MN is estimated to be 37.24 years (SE=0.02918).

```
> # SDR SE estimate for mean age in MN
> rep.ests <- apply(repwts, 2, function(u) sum(u*acs$AGEP)/sum(u))
> sqrt(4*sum((rep.ests - sum(w*acs$AGEP)/sum(w))^2)/80)
[1] 0.02917897
```

- The average age in PUMA 2100 is estimated to be 37.72 (SE=0.1921).

```
> # SDR SE estimate for mean age in PUMA 2100
> rep.ests <- apply(repwts, 2, function(u) sum(u*acs$AGEP*x)/sum(u*x))
> sqrt(4*sum((rep.ests - sum(w*acs$AGEP*x)/sum(w*x))^2)/80)
[1] 0.1920715
```

37