ch09output

2022-05-03

```r
# Code from Chapter 9 of R Companion for Sampling: Design and Analysis by
# Yan Lu and Sharon L. Lohr
# All code is presented for educational purposes only and without warranty.

##### Install the R packages needed for the chapter

library(survey)
```

```
## Loading required package: grid

## Loading required package: Matrix

## Loading required package: survival

##
## Attaching package: 'survey'

## The following object is masked from 'package:graphics':
##
##     dotchart
```

```r
library(sampling)
```

```
##
## Attaching package: 'sampling'

## The following objects are masked from 'package:survival':
##
##     cluster, strata
```

```r
library(SDAResources)

########## Replicate Samples and Random Groups ##########

##### Example 9.3

# Replicate samples
data(college)
# define population with public colleges and universities
public_college<-college[college$control==1,]
N<-nrow(public_college) #500
# select five SRSs and calculate means
xbar<-rep(NA,5)
ybar<-rep(NA,5)
set.seed(8126834)
for(i in 1:5){
  index <- srswor(10,N)
  replicate <- public_college[(1:N)[index==1],]
  # save replicate in a data frame if you want to keep it for later analyses
```

```
  # define design object (since SRS, weights are computed from fpc)
  dcollege<-svydesign(id = ~1, fpc = ~rep(500,10), data = replicate)
  # calculate mean of in-state and out-of-state tuition fees
  xbar[i]<-coef(svymean(~tuitionfee_in, dcollege))
  ybar[i]<-coef(svymean(~tuitionfee_out,dcollege))
}
# print the 5th replicate sample
replicate[,c(2,24:25)]
```

```
##                                            instnm tuitionfee_in tuitionfee_out
## 459                      Coppin State University          8873          15144
## 474                            Towson University          9940          23208
## 556                University of Michigan-Flint         11304          22065
## 674                   University of Nevada-Reno          7599          22236
## 735                       CUNY Brooklyn College          7240          14910
## 853  University of North Carolina at Greensboro          7331          22490
## 1024    Millersville University of Pennsylvania         12226          22196
## 1030  Pennsylvania State University-Main Campus         18454          34858
## 1359            Texas A&M University-San Antonio          8656          21159
## 1368         University of North Texas at Dallas          9139          21589
```

```
# calculate and print the five ratio estimates
thetahat<-ybar/xbar
thetahat
```

```
## [1] 2.172545 2.055528 2.107828 2.213799 2.181924
```

```
# calculate mean of the five ratio estimates, and SE
thetatilde<-mean(thetahat)
thetatilde
```

```
## [1] 2.146325
```

```
setheta<-sqrt(var(thetahat)/5)
# calculate confidence interval by direct formula using t distribution
c( thetatilde- qt(.975, 4)*setheta, thetatilde+ qt(.975, 4)*setheta)
```

```
## [1] 2.067224 2.225426
```

```
# easier: use t.test function to calculate mean and confidence interval
t.test(thetahat)
```

```
##
##  One Sample t-test
##
## data:  thetahat
## t = 75.336, df = 4, p-value = 1.861e-07
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  2.067224 2.225426
## sample estimates:
## mean of x
##  2.146325
```

```
##### Example 9.4

# Random groups
data(syc)
```

```
dsyc<-svydesign(id = ~1, weights = ~finalwt, data = syc)
repmean<-svyby(~age, ~randgrp, dsyc, svymean)
repmean # we use only the means, not the SEs
```

```
##   randgrp      age        se
## 1       1 16.54947 0.1171541
## 2       2 16.66331 0.1133751
## 3       3 16.82544 0.1242695
## 4       4 16.05688 0.1240046
## 5       5 16.31776 0.1160307
## 6       6 17.02798 0.1181861
## 7       7 17.26605 0.1110258
```

```
# Estimate and SE 1 (could also use t.test function)
thetatilde<-mean(repmean$age)
SEthetatilde<- sqrt( (1/7)*var(repmean$age) )
# Estimate and SE 2
thetahat<-coef(svymean(~age,dsyc))
SEthetahat<- sqrt((1/7)*(1/6)*sum((repmean$age-thetahat)^2))

#calculate confidence interval by direct formula using t distribution
Mean_CI1 <- c(thetatilde, SEthetatilde, thetatilde- qt(.975, 7-1)*SEthetatilde,
              thetatilde+ qt(.975, 7-1)*SEthetatilde)
names(Mean_CI1) <- c("thetatilde","SE","lower CL", "upper CL")
Mean_CI1
```

```
## thetatilde         SE   lower CL   upper CL
## 16.6724103  0.1559995 16.2906932 17.0541274
```

```
Mean_CI2 <- c(thetahat,SEthetahat, thetahat- qt(.975, 7-1)*SEthetahat,
               thetahat+ qt(.975, 7-1)*SEthetahat)
names(Mean_CI2) <- c("thetahat","SE","lower CL", "upper CL")
Mean_CI2
```

```
##   thetahat         SE   lower CL   upper CL
## 16.6392931  0.1565843 16.2561452 17.0224411
```

```
########## Constructing Replicate Weights ##########

####### Balanced repeated replication (BRR)

##### Example 9.5

brrex<-data.frame(strat = c(1,1,2,2,3,3,4,4,5,5,6,6,7,7),
  strfrac =c(0.3,0.3,0.1,0.1,0.05,0.05,0.1,0.1,0.2,0.2,0.05,0.05,0.2,0.2),
  y =c(2000,1792,4525,4735,9550,14060,800,1250,9300,7264,13286,12840,2106,2070)
      )

brrex$wt <- 10000*brrex$strfrac/2
brrex
```

```
##    strat strfrac     y   wt
## 1      1    0.30  2000 1500
## 2      1    0.30  1792 1500
## 3      2    0.10  4525  500
## 4      2    0.10  4735  500
```

```
## 5       3    0.05  9550  250
## 6       3    0.05 14060  250
## 7       4    0.10   800  500
## 8       4    0.10  1250  500
## 9       5    0.20  9300 1000
## 10      5    0.20  7264 1000
## 11      6    0.05 13286  250
## 12      6    0.05 12840  250
## 13      7    0.20  2106 1000
## 14      7    0.20  2070 1000
```

```
dbrrex<-svydesign(id=~1, strata=~strat,weights=~wt,data=brrex)
dbrrex # stratified random sample
```

```
## Stratified Independent Sampling design (with replacement)
## svydesign(id = ~1, strata = ~strat, weights = ~wt, data = brrex)
```

```
# convert to BRR replicate weights
dbrrexbrr <- as.svrepdesign(dbrrex, type="BRR")
dbrrexbrr # identifies as BRR
```

```
## Call: as.svrepdesign(dbrrex, type = "BRR")
## Balanced Repeated Replicates with 8 replicates.
```

```
# now use the replicate weights to calculate the mean and confidence interval
svymean(~y,dbrrexbrr)
```

```
##     mean     SE
## y 4451.7 236.42
```

```
degf(dbrrexbrr)
```

```
## [1] 7
```

```
confint(svymean(~y,dbrrexbrr),df=7)
```

```
##      2.5 %   97.5 %
## y 3892.664 5010.736
```

```
## Fay's method for BRR
dbrrexfay <- as.svrepdesign(dbrrex, type="Fay",fay.rho=0.5)
svymean(~y,dbrrexfay)
```

```
##     mean     SE
## y 4451.7 236.42
```

```
confint(svymean(~y,dbrrexfay),df=7)
```

```
##      2.5 %   97.5 %
## y 3892.664 5010.736
```

```
# look at replicate weights for contrast with regular BRR
# note values for replicate weight multiplier are now 1.5 and 0.5
dbrrexfay$repweights$weights
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
##  [1,]  1.5  0.5  1.5  0.5  1.5  0.5  1.5  0.5
##  [2,]  0.5  1.5  0.5  1.5  0.5  1.5  0.5  1.5
##  [3,]  1.5  1.5  0.5  0.5  1.5  1.5  0.5  0.5
##  [4,]  0.5  0.5  1.5  1.5  0.5  0.5  1.5  1.5
```

```
##  [5,]  1.5  0.5  0.5  1.5  1.5  0.5  0.5  1.5
##  [6,]  0.5  1.5  1.5  0.5  0.5  1.5  1.5  0.5
##  [7,]  1.5  1.5  1.5  1.5  0.5  0.5  0.5  0.5
##  [8,]  0.5  0.5  0.5  0.5  1.5  1.5  1.5  1.5
##  [9,]  1.5  0.5  1.5  0.5  0.5  1.5  0.5  1.5
## [10,]  0.5  1.5  0.5  1.5  1.5  0.5  1.5  0.5
## [11,]  1.5  1.5  0.5  0.5  0.5  0.5  1.5  1.5
## [12,]  0.5  0.5  1.5  1.5  1.5  1.5  0.5  0.5
## [13,]  1.5  0.5  0.5  1.5  0.5  1.5  1.5  0.5
## [14,]  0.5  1.5  1.5  0.5  1.5  0.5  0.5  1.5
```

```
### Example 9.6

data(nhanes)
nhanes$age20d<-rep(0,nrow(nhanes))
nhanes$age20d[nhanes$ridageyr >=20 & !is.na(nhanes$bmxbmi)]<-1
dnhanes<-svydesign(id=~sdmvpsu, strata=~sdmvstra,nest=TRUE,
                   weights=~wtmec2yr,data=nhanes)
dnhanesbrr <- as.svrepdesign(dnhanes, type="BRR")
# look at subset of adults age 20+
dnhanesbrrsub<-subset(dnhanesbrr, age20d =='1')
degf(dnhanes)
```

```
## [1] 15
```

```
degf(dnhanesbrrsub) # same df
```

```
## [1] 15
```

```
# calculate mean
bmimean<-svymean(~bmxbmi, dnhanesbrrsub)
bmimean
```

```
##          mean    SE
## bmxbmi 29.389 0.261
```

```
confint(bmimean,df=15)
```

```
##           2.5 %   97.5 %
## bmxbmi 28.83279 29.94541
```

```
# calculate quantiles
svyquantile(~bmxbmi, dnhanesbrrsub, quantiles=c(0.25,0.5,0.75,0.95),
            ties = "rounded")
```

```
## Statistic:
##         bmxbmi
## q0.25 24.35349
## q0.5  28.23490
## q0.75 33.06615
## q0.95 42.64092
## SE:
##          bmxbmi
## q0.25 0.2215986
## q0.5  0.3241246
## q0.75 0.3139102
## q0.95 0.3436826
```

```r
####### Jackknife

##### Example 9.7

data(collegerg)
collegerg1<-collegerg[collegerg$repgroup==1,]
collegerg1[,24:25]
```

```
##    tuitionfee_in tuitionfee_out
## 1           9912          23640
## 2           7140          14810
## 3           9808          26648
## 4           8987          35170
## 5           7930           8674
## 6           7200          17550
## 7           8929          21692
## 8          11976          22488
## 9           8935          27199
## 10          8316          18276
```

```r
collegerg1$sampwt<-rep(500/10,10)
# calculate SEs of means and ratio using linearization
dcollegerg1<-svydesign(id=~1, weights=~sampwt,data=collegerg1)
means.lin<-svymean(~tuitionfee_in+tuitionfee_out, dcollegerg1)
means.lin
```

```
##                   mean      SE
## tuitionfee_in   8913.3  454.46
## tuitionfee_out 21614.7 2325.15
```

```r
confint(means.lin,df=degf(dcollegerg1))
```

```
##                    2.5 %    97.5 %
## tuitionfee_in   7885.247  9941.353
## tuitionfee_out 16354.843 26874.557
```

```r
ratio.lin<-svyratio(~tuitionfee_out,~tuitionfee_in,dcollegerg1)
ratio.lin
```

```
## Ratio estimator: svyratio.survey.design2(~tuitionfee_out, ~tuitionfee_in, dcollegerg1)
## Ratios=
##                tuitionfee_in
## tuitionfee_out      2.424994
## SEs=
##                tuitionfee_in
## tuitionfee_out     0.2311776
```

```r
confint(ratio.lin,df=degf(dcollegerg1))
```

```
##                                 2.5 %   97.5 %
## tuitionfee_out/tuitionfee_in 1.902034 2.947954
```

```r
## define jackknife replicate weights design object
dcollegerg1jk <- as.svrepdesign(dcollegerg1, type="JK1")
dcollegerg1jk
```

```
## Call: as.svrepdesign(dcollegerg1, type = "JK1")
## Unstratified cluster jacknife (JK1) with 10 replicates.
```

```r
# now look at jackknife SE for means
# these are same as linearization since SRS and statistic = mean
svymean(~tuitionfee_in + tuitionfee_out, dcollegerg1jk)
```

```
##                    mean      SE
## tuitionfee_in    8913.3  454.46
## tuitionfee_out 21614.7 2325.15
```

```r
# jackknife SE for ratio
svyratio(~tuitionfee_out, ~tuitionfee_in, design = dcollegerg1jk)
```

```
## Ratio estimator: svyratio.svyrep.design(~tuitionfee_out, ~tuitionfee_in, design = dcollegerg1jk)
## Ratios=
##                tuitionfee_in
## tuitionfee_out      2.424994
## SEs=
##           [,1]
## [1,] 0.2314828
```

```r
# can look at replicate weight multipliers if desired
# note that observation being omitted for replicate has weight 0
# weight multiplier for other observations is 10/9 = 1.11111
round(dcollegerg1jk$repweights$weights,digits=4)
```

```
##         [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]  [,10]
##  [1,] 0.0000 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111
##  [2,] 1.1111 0.0000 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111
##  [3,] 1.1111 1.1111 0.0000 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111
##  [4,] 1.1111 1.1111 1.1111 0.0000 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111
##  [5,] 1.1111 1.1111 1.1111 1.1111 0.0000 1.1111 1.1111 1.1111 1.1111 1.1111
##  [6,] 1.1111 1.1111 1.1111 1.1111 1.1111 0.0000 1.1111 1.1111 1.1111 1.1111
##  [7,] 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111 0.0000 1.1111 1.1111 1.1111
##  [8,] 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111 0.0000 1.1111 1.1111
##  [9,] 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111 0.0000 1.1111
## [10,] 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111 1.1111 0.0000
```

```r
##### Example 9.8

data(coots)
coots$relwt<-coots$csize/2
dcoots<-svydesign(id=~clutch,weights=~relwt,data=coots)
dcootsjk <- as.svrepdesign(dcoots, type="JK1")
dcootsjk
```

```
## Call: as.svrepdesign(dcoots, type = "JK1")
## Unstratified cluster jacknife (JK1) with 184 replicates.
```

```r
svymean(~volume,dcootsjk)
```

```
##          mean    SE
## volume 2.4908 0.061
```

```r
confint(svymean(~volume,dcootsjk),df=degf(dcootsjk))
```

```
##           2.5 %   97.5 %
## volume 2.370354 2.611203
```

```
####### Bootstrap

##### Example 9.9

data(htsrs)
nrow(htsrs)
```

```
## [1] 200
```

```
head(htsrs)
```

```
##     rn height gender
## 1  257    159      F
## 2 1016    174      M
## 3 1264    186      M
## 4  817    158      F
## 5  374    178      F
## 6 1063    177      M
```

```
wt<-rep(10,nrow(htsrs))
dhtsrs<-svydesign(id=~1, weights=~wt,data=htsrs)
dhtsrs
```

```
## Independent Sampling design (with replacement)
## svydesign(id = ~1, weights = ~wt, data = htsrs)
```

```
set.seed(9231)
dhtsrsboot <- as.svrepdesign(dhtsrs, type="subbootstrap",replicates=1000)
# linearization
svymean(~height,dhtsrs)
```

```
##          mean     SE
## height 168.94 0.7831
```

```
# bootstrap
svymean(~height,dhtsrsboot)
```

```
##          mean     SE
## height 168.94 0.7978
```

```
degf(dhtsrsboot) # 199 = n - 1
```

```
## [1] 199
```

```
confint(svymean(~height,dhtsrsboot),df=degf(dhtsrsboot))
```

```
##            2.5 %   97.5 %
## height 167.3667 170.5133
```

```
# bootstrap by direct coding
# number of iteration
R <- 10000
# init location for bootstrap theta
thetahat <- rep(NA, R)
# draw R bootstrap resamples
for (i in 1:R) {
  #
  resam <- sample(htsrs$height, 199, replace = TRUE)
thetahat[i] <- median(resam)
```

```
}
# variance and CI estimate by normal approximation
sebs<-sqrt(var(thetahat))
sebs
```

```
## [1] 0.971914
```

```
m<-median(htsrs$height)
m
```

```
## [1] 169
```

```
CI.bs1 <- c(m-1.96*sebs,m+1.96*sebs)
CI.bs1
```

```
## [1] 167.095 170.905
```

```
# sort the bootstrap estimates to obtain bootstrap CI
# 0.025th and 0.975th quantile gives equal-tail bootstrap CI
thetahat.sorted <- sort(thetahat)
CI.bs2 <- c(thetahat.sorted[round(0.025*R)], thetahat.sorted[round(0.975*R+1)])
CI.bs2
```

```
## [1] 167 171
```

```
##### Example 9.10

data(htstrat)
nrow(htstrat)
```

```
## [1] 200
```

```
head(htstrat)
```

```
##    rn height gender
## 1 201    166      F
## 2 965    163      F
## 3 490    166      F
## 4 249    155      F
## 5 260    154      F
## 6 324    160      F
```

```
dhtstrat <- svydesign(id = ~1, strata = ~gender, fpc = c(rep(1000,160),rep(1000,40)),
                      data = htstrat)
dhtstrat
```

```
## Stratified Independent Sampling design
## svydesign(id = ~1, strata = ~gender, fpc = c(rep(1000, 160),
##     rep(1000, 40)), data = htstrat)
```

```
set.seed(982537455)
dhtstratboot <- as.svrepdesign(dhtstrat, type="subbootstrap",replicates=1000)
svymean(~height,dhtstratboot)
```

```
##          mean     SE
## height 169.02 0.7296
```

```
degf(dhtstratboot)
```

```
## [1] 198
```

```r
confint(svymean(~height,dhtstratboot),df=degf(dhtstratboot))
```

```
##           2.5 %   97.5 %
## height 167.5769 170.4543
```

```r
####### Replicate Weights and Nonresponse Adjustments

##### Example 4.9

data(agsrs)
# define design object for sample
dsrs <- svydesign(id = ~1, weights=rep(3078/300,300), data = agsrs)
# define replicate weights design object
dsrsjk<-as.svrepdesign(dsrs,type="JK1")
# poststratify on region
pop.region <- data.frame(region=c("NC","NE","S","W"), Freq=c(1054,220,1382,422))
dsrspjk<-postStratify(dsrsjk, ~region, pop.region)
svymean(~acres92, dsrspjk)
```

```
##           mean    SE
## acres92 299778 18653
```

```r
confint(svymean(~acres92, dsrspjk),df=degf(dsrspjk))
```

```
##            2.5 % 97.5 %
## acres92 263069.2 336487
```

```r
svytotal(~acres92, dsrspjk)
```

```
##             total       SE
## acres92 922717031 57413300
```

```r
# Check: estimates of counts in poststrata = pop.region counts with SE = 0
svytotal(~factor(region),dsrspjk)
```

```
##                  total SE
## factor(region)NC  1054  0
## factor(region)NE   220  0
## factor(region)S   1382  0
## factor(region)W    422  0
```

```r
########## Using Replicate Weights from a Survey Data File ##########

##### Example 9.5

# Create data frame containing final and replicate weights, and y
repwts<- dbrrexbrr$repweights$weights * matrix(brrex$wt,nrow=14,ncol=8,byrow=FALSE)
brrdf<-data.frame(y=brrex$y,wt=brrex$wt,repwts)
colnames(brrdf)<-c("y","wt",paste("repwt",1:8,sep=""))
brrdf # contains weight, repwt1-repwt8, and y but no stratum info
```

```
##        y   wt repwt1 repwt2 repwt3 repwt4 repwt5 repwt6 repwt7 repwt8
## 1   2000 1500   3000      0   3000      0   3000      0   3000      0
## 2   1792 1500      0   3000      0   3000      0   3000      0   3000
## 3   4525  500   1000   1000      0      0   1000   1000      0      0
## 4   4735  500      0      0   1000   1000      0      0   1000   1000
## 5   9550  250    500      0      0    500    500      0      0    500
## 6  14060  250      0    500    500      0      0    500    500      0
```

```
## 7     800  500    1000    1000    1000    1000       0       0       0       0
## 8    1250  500       0       0       0       0    1000    1000    1000    1000
## 9    9300 1000    2000       0    2000       0       0    2000       0    2000
## 10   7264 1000       0    2000       0    2000    2000       0    2000       0
## 11 13286  250     500     500       0       0       0       0     500     500
## 12 12840  250       0       0     500     500     500     500       0       0
## 13  2106 1000    2000       0       0    2000       0    2000    2000       0
## 14  2070 1000       0    2000    2000       0    2000       0       0    2000
```

```
# create design object
dbrrdf<-svrepdesign(weights=~wt,repweights="repwt[1-9]",data=brrdf,type="BRR")
dbrrdf
```

```
## Call: svrepdesign.default(weights = ~wt, repweights = "repwt[1-9]",
##     data = brrdf, type = "BRR")
## Balanced Repeated Replicates with 8 replicates.
```

```
svymean(~y,dbrrdf) # same as before!
```

```
##     mean     SE
## y 4451.7 236.42
```