9

Variance Estimation in Complex Surveys

In this chapter, we discuss variance estimation in complex surveys. We have already seen the use of linearization methods to calculate variances in functions such as *svymean* and *svytotal* in the `survey` package (Lumley, 2020). This chapter will focus on the other methods for calculating variances—random groups, balanced repeated replication (BRR), jackknife, and bootstrap. The code is in file `ch09.R` on the book website.

9.1 Replicate Samples and Random Groups

The random group methods in Section 9.2 of SDA are seldom used in practice. But they are of interest because they motivate how replication methods work in general, and they can be useful for providing quick-and-easy variance estimates for a complex survey design. The methods require three steps:

1. Select the replicate samples from the population, or divide the probability sample among the random groups.
2. Calculate the statistic of interest from each replicate, using the survey weights.
3. Use the estimated statistics from the replicates to calculate the standard error.

Example 9.3 of SDA. In this example, we consider estimating the variance of the ratio of out-of-state tuition fee (*tuitionfee_ out*) and in-state tuition fee (*tuitionfee_ in*) for a population of public colleges using replicate samples. Data *public_ college*, created below, consists of 500 public colleges and universities from the *college* data. This data set serves as the population from which we draw replicate samples.

For Step 1, we use a loop and the *srswor* function (package `sampling`, Tillé and Matei, 2021) to select five independent replicate SRSs, each of size 10, from *public_ college* (one could also write a function to do this). We print the values of the selected variables for the colleges in the fifth replicate sample. Note that the weights for each replicate sample sum to the population size, 500. The replicate samples in this book differ from those in SDA, which were selected using SAS software.

For Step 2, compute the statistic of interest from each replicate sample. We calculate the ratio of average out-of-state tuition *ybar*[*i*] to average in-state tuition *xbar*[*i*] for each replicate sample i by calling the *svymean* function.

Variable *thetahat* contains the values of $\hat{\theta}_i$ for the five replicate samples. These values are the only part of the output used in Step 3; The standard errors given by the *svymean* function are ignored—only the point estimates of the means are used.

DOI: 10.1201/9781003228196-9

For Step 3, treat the 5 estimated ratios in *thetahat* as independent and identically distributed observations, and calculate their mean $\tilde{\theta}$. We also calculate a 95% confidence interval, which can be done either by formula or with the *t.test* function.

```
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
```

The output shows that the mean of the ratios estimated from the five replicate samples is 2.146 and the 95% confidence interval for the population ratio is $[2.067, 2.225]$. The 95% confidence interval is calculated as $\tilde{\theta} \pm t\,\mathrm{SE}(\tilde{\theta})$, where t is the critical value from a t distribution with 4 (number of replicates minus one) degrees of freedom (df). This critical value is 2.78, giving a wider interval than would be obtained from a variance estimate with more df.

In Example 9.3, the survey weights are the only design feature used for the calculations. Even if a replicate sample has stratification or clustering, that information is not needed to calculate the point estimate of the parameter of interest θ for the replicates. The effect of the stratification or clustering on the variance is incorporated in the variability among the $\hat{\theta}_i$'s. For example, if clustering decreases the precision for $\hat{\theta}_i$, then the estimates $\hat{\theta}_i$ will vary more from replicate to replicate, and the decreased precision will show up in a large value of the sample variance for the replicate values $\hat{\theta}_i$.

We illustrated the method for estimating a ratio, but the same method can be used for any statistic you would like to estimate. The method can also be applied to statistics that are not calculated by the survey analysis functions. All you need to do is to calculate the statistic of interest for each replicate using the survey weights, then apply t confidence interval methods to the statistics calculated from the replicates.

Random groups. After dividing the survey data into R random groups, the procedure for calculating the estimator and variance is exactly the same as Steps 1 to 3 for replicate sample methods. Note that if you are estimating population totals, you need to scale the weights for each random group so they sum to the population size.

Example 9.4 of SDA. This example illustrates the random group method with the Survey of Youth in Custody (*syc*) data. The whole sample is divided into seven random groups by variable *randgrp*. The *svyby* function computes the mean of variable *age* using *svymean* separately for each random group. Each is calculated using the survey weight *finalwt*.

Only the point estimates for *age*, printed in *repmean*, are needed to calculate the mean and confidence interval. The standard errors are not used. Thus, although the survey has stratification and clustering, we do not need to include that information in the *svydesign* function when calculating variances with the random groups method—the estimated means of each group depend only on the weights in *finalwt*.

```
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
```

Let $\hat{\theta}_r$ be the estimate from the rth replicate sample, $r = 1, 2, \cdots, 7$, $\tilde{\theta}$ be the mean of the seven random group means, and $\hat{\theta}$ be the estimated mean age using the whole data set *syc*. We can calculate two variance estimates: *SEthetatilde* calculates the square root of

$$\hat{V}_1(\tilde{\theta}) = \frac{1}{R}\frac{1}{R-1}\sum_{r=1}^{R}\left(\hat{\theta}_r - \tilde{\theta}\right)^2 \tag{9.1}$$

and *SEthetahat* calculates the square root of

$$\hat{V}_2(\hat{\theta}) = \frac{1}{R}\frac{1}{R-1}\sum_{r=1}^{R}\left(\hat{\theta}_r - \hat{\theta}\right)^2. \tag{9.2}$$

```
# 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
```

The estimated mean of age using *thetatilde* is 16.67241 with a confidence interval of $[16.29069, 17.05413]$. The estimated mean of age using *thetahat* is 16.63929 with a confidence interval of $[16.25615, 17.02244]$. A t distribution with 6 (the number of random groups minus 1) df is used. Again, though, with few random groups, the confidence interval is wider than it would be if a variance estimation method having more df were used. We discuss such methods in the next section.

9.2 Constructing Replicate Weights

Replicate weights for balanced repeated replication (BRR), jackknife, and bootstrap methods can be created with the *as.svrepdesign* function in the `survey` package (Lumley, 2020). It creates a replicate-weights survey design object from a survey design object that contains stratification and clustering information.

The basic function call, shown here for BRR, is

```
as.svrepdesign(design.object, type = "BRR")
```

where *design.object* is a design object that has been created using *svydesign* (containing the stratification and clustering information) and *type* specifies the replication method to be used.

9.2.1 Balanced Repeated Replication

Example 9.5 of SDA. This example shows how to use BRR to calculate variances for the small data set in Table 9.2 of SDA. Here, we assume that $N = 10,000$. Variable *wt* contains the sampling weight, which is $N_h/2$ for this sample with $n_h = 2$ sampled observations per stratum and is obtained by $N(N_h/N)/n_h = (N)(strfrac)/2$. We request `type="BRR"` in the *as.svrepdesign* function.

The function creates 8 replicate weights because that is the smallest multiple of 4 that is larger than 7, the number of strata.

```
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
```

The estimated mean value of y is 4451.7 with a standard error of 236.42, and a confidence interval of [3892.664, 5010.736].

You can also look at the replicate weights multiplier if desired. Note that the two observations in each stratum have complementary patterns.

```
# can look at replicate weight multiplier if desired
# this is to illustrate weight structure; it is not needed for data analysis
dbrrexbrr$repweights$weights
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
##  [1,]    2    0    2    0    2    0    2    0
##  [2,]    0    2    0    2    0    2    0    2
##  [3,]    2    2    0    0    2    2    0    0
##  [4,]    0    0    2    2    0    0    2    2
##  [5,]    2    0    0    2    2    0    0    2
##  [6,]    0    2    2    0    0    2    2    0
##  [7,]    2    2    2    2    0    0    0    0
##  [8,]    0    0    0    0    2    2    2    2
##  [9,]    2    0    2    0    0    2    0    2
## [10,]    0    2    0    2    2    0    2    0
## [11,]    2    2    0    0    0    0    2    2
## [12,]    0    0    2    2    2    2    0    0
## [13,]    2    0    0    2    0    2    2    0
## [14,]    0    2    2    0    2    0    0    2
```

Fay's method for BRR. The *as.svrepdesign* function will also construct replicate weights using Fay's variation of BRR (Dippo et al., 1984; Judkins, 1990). Simply specify `type="Fay"` instead of `type="BRR"`. To set ε for Fay's method equal to 0.5, add the argument `fay.rho = 0.5`. This results in weight multipliers 1.5 and 0.5 instead of the multipliers 0 and 2 for the BRR example above.

```
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 of SDA. Now let's create BRR weights for a data set with stratification and clustering: the NHANES data. First, use *svydesign* to include primary sampling unit (psu) *sdmvpsu*, stratum *sdmvstra*, and weight *wtmec2yr* information, and then use the *as.svrepdesign* function to construct replicate weights using BRR by adding `type="BRR"`. By specifying the stratification and clustering information through *svydesign*, you ensure that observations in the same psu are kept together during the replicate weight construction.

This analysis finds summary statistics of body mass index (BMI, in variable *bmxbmi*) for adults age 20 and over, so the *subset* function is also used. Note that the df for the subset is the same as for the full sample—there are members from this domain in each psu, so there are (number of psus − number of strata) df.

```
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
```

The mean and quantiles are estimated using the full sample weights, and the replicate weights are used to calculate the standard errors. The estimated mean BMI for adults is 29.389 with a standard error of 0.261 and a 95% confidence interval of $[28.833, 29.945]$. You can also include optional argument `return.replicates=TRUE` in the *svymean* statement in case you want to look at the statistics calculated for each replicate.

Note that these replicate weights, constructed from the NHANES final weights available on the public-use data file, do not account for the effects of poststratification on the variances.

See Section 9.2.4 for how to calculate replicate weights so that the variance estimates include the effects of weighting adjustments.

Domain estimates with replicate weights. We carried out this analysis on the subset of observations with *age20d*=1. The *subset* function is used exactly the same way with a replicate-weight design object as with a strata/cluster design object. Make sure, though, that you define the replicate-weight design object on the full data set first, and then use the *subset* function to define the domain. This ensures that the full design information is used to calculate standard errors.

9.2.2 Jackknife

The `survey` package will create replicate weights for two types of jackknife: JK1 and JKn. These are described in detail by Brick et al. (2000) and Chapter 15 of Valliant et al. (2018). Briefly, JKn is the jackknife for stratified multistage sampling described in Chapter 9 of SDA. If there are n_h psus in stratum h and observation i has weight w_i, then the JKn jackknife weights are defined as follows:

$$w_{i(hj)} = \begin{cases} w_i & \text{if observation unit } i \text{ is not in stratum } h \\ 0 & \text{if observation unit } i \text{ is in psu } j \text{ of stratum } h \\ \dfrac{n_h}{n_h - 1} w_i & \text{if observation unit } i \text{ is in stratum } h \text{ but not in psu } j. \end{cases} \tag{9.3}$$

The weights $w_{i(hj)}$ are used to calculate $\hat{\theta}_{(hj)}$ for each replicate, and

$$\hat{V}_{\text{JKn}}(\hat{\theta}) = \sum_{h=1}^{H} \frac{n_h - 1}{n_h} \sum_{j=1}^{n_h} \left(\hat{\theta}_{(hj)} - \hat{\theta} \right)^2 . \tag{9.4}$$

Jackknife JK1 is a special case of JKn where there is one stratum.

Jackknife weights for an SRS. Let's start by looking at how the *as.svrepdesign* function works using jackknife variance estimation for an SRS, then move on to complex sample designs.

Example 9.7 of SDA. Data *collegerg* shows the values of in-state and out-of-state tuition for five replicate samples, each of size 10 (these samples were selected with SAS software, and differ from the replicate samples selected earlier in this chapter). We define data `collegerg1` to be replicate sample 1 (having *repgroup* = 1).

For comparison purposes, we first look at the linearization (Taylor series) variance estimate of the mean in-state tuition, mean out-of-state tuition, and ratio of mean out-of-state tuition to mean in-state tuition. Note that we did not include the *fpc* argument in *svydesign*, so the with-replacement variance is calculated.

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

```
confint(means.lin,df=degf(dcollegerg1))
##                     2.5 %    97.5 %
## tuitionfee_in   7885.247  9941.353
## tuitionfee_out 16354.843 26874.557
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
confint(ratio.lin,df=degf(dcollegerg1))
##                                  2.5 %   97.5 %
## tuitionfee_out/tuitionfee_in 1.902034 2.947954
```

Now let's calculate the jackknife variance by omitting observation j in replicate j. We use function *as.svrepdesign* with `type="JK1"` to create jackknife weights for this SRS of size 10. We then use *svymean* and *svyratio* to calculate the standard errors of the estimated means and ratio with the jackknife weights.

```
## 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.
# 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
# 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
```

The above output shows the statistics produced by the *svymean* and *svyratio* functions using linearization and jackkinfe methods. Typing `dcollegerg1jk` shows that this unstratified jackknife (JK1) has 10 replicates. The jackknife standard errors for the estimated means of *tuitionfee_in* and *tuitionfee_out* are 454.46 and 2325.15, respectively. These are the same as the linearization standard errors because $\hat{V}_{\text{JK}}(\bar{y}) = s_y^2/n$ for an SRS, as shown in Section 9.3 of SDA. The jackknife standard error for the nonlinear statistic of the ratio, 0.231483, differs slightly from the linearization standard error of 0.231178. These values are extremely close—after all, the linearization variance and the jackknife variance are both consistent estimators for $V(\hat{B})$—but are not exactly the same. A t distribution with $n - 1 = 9$ df is used for the confidence interval.

For an SRS such as this, the jackknife weights are calculated by setting the weight of observation j to zero and assigning weight $w_i \times n/(n-1) = (500/10) * (10/9) = 55.555$ to each of the other observations. You can print the replicate weight multipliers for each observation if you want (although you typically do not need to do this when carrying out a data analysis).

```
# 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
```

Jackknife weights for a complex survey. For a survey with stratification and clustering, you need to include the stratum and cluster information in the *svydesign* function. The replicate weights are then constructed by deleting one psu at a time from each stratum.

Example 9.8 of SDA. In this example, we consider a two-stage cluster sample. Because *clutch* is the primary sampling unit (psu), the procedure deletes one clutch at a time rather than one observation at a time.

The sampling weight *relwt* is defined as $M_i/m_i = \text{csize}/2$, the number of eggs in the clutch divided by the number of eggs selected from the clutch for measurement. The *as.svrepdesign* function sets the replicate weight variable equal to 0 for all observations from the psu being deleted in that replicate. This design has one stratum, so we use `type="JK1"`; if your survey has stratification, use `type="JKn"`.

```
data(coots)
coots$relwt<-coots$csize/2
dcoots<-svydesign(id=~clutch,weights=~relwt,data=coots)
dcootsjk <- as.svrepdesign(dcoots, type="JK1")
svymean(~volume,dcootsjk)
##          mean    SE
## volume 2.4908 0.061
confint(svymean(~volume,dcootsjk),df=degf(dcootsjk))
##           2.5 %   97.5 %
## volume 2.370354 2.611203
```

The estimated mean volume is 2.49 with a standard error of 0.061 by the jackknife method. The confidence interval for the mean volume is $[2.370354, 2.611203]$, calculated using 183 (number of psus minus 1) df.

9.2.3 Bootstrap

Example 9.9 of SDA. This example looks at creating bootstrap weights using the method in Rao et al. (1992) to estimate the population distribution of height from the SRS in data set *htsrs*. The number of replicates used is specified by `replicates = 1000` in the function

as.svrepdesign with `type = "subbootstrap"`. You will need to use *set.seed* to be able to reconstruct the same set of bootstrap weights later.

The weight variable *wt* equals 10 for each observation. In each replicate, if observation i is selected m times; the weight in the replicate for that observation is, using the formula in Section 9.3 of SDA, $10 \times (200/199) \times m$. The df with the bootstrap method should be set equal to the sample size minus one, which is $200 - 1 = 199$, not the number of bootstrap replicates, which can be set to any value.

```
data(htsrs)
nrow(htsrs)
## [1] 200
wt<-rep(10,nrow(htsrs))
dhtsrs<-svydesign(id=~1, weights=~wt,data=htsrs)
set.seed(9231)
dhtsrsboot <- as.svrepdesign(dhtsrs, type="subbootstrap",replicates=1000)
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
svyquantile(~height, dhtsrsboot, quantile=c(0.25,0.5,0.75), ties=c("rounded"))
## Statistic:
##       height
## q0.25 159.70
## q0.5  168.75
## q0.75 176.00
## SE:
##          height
## q0.25 0.8878743
## q0.5  0.9930391
## q0.75 1.1339674
```

The output above shows the summary statistics of the mean, selected quantiles, and associated standard errors calculated using the bootstrap method with seed 9231.

Example 9.10 of SDA. The code for creating bootstrap weights in a complex survey design is similar to that for an SRS. The survey design is specified by *svydesign*, and the *as.svrepdesign* function is used to create the replicate weights. Data set *htstrat* is a stratified random sample, so *svydesign* requires only the stratum information. When the *weights* argument is not supplied, the function calculates the weights from the information in the *fpc* argument.

```
data(htstrat)
nrow(htstrat)
## [1] 200
dhtstrat <- 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
confint(svymean(~height,dhtstratboot),df=degf(dhtstratboot))
##           2.5 %   97.5 %
## height 167.5769 170.4543
```

The estimated mean height is 169.02 with a standard error of 0.7296 by the bootstrap method. The 95% confidence interval for mean height is $[167.5769, 170.4543]$, calculated using a t distribution with 198 df (sample size minus number of strata, $200 - 2$). If you specify the bootstrap design object with a different random number seed, you will obtain a slightly different value for the standard error because a different set of bootstrap samples will be used for the variance calculations.

The *as.svrepdesign* function will also construct bootstrap replicate weights for other forms of bootstrap: `type="bootstrap"` uses the method of Canty and Davison (1999) and `type="mrbbootstrap"` uses the multistage rescaled bootstrap of Preston (2009).

9.2.4 Replicate Weights and Nonresponse Adjustments

The code given so far in this section constructs replicate sampling weights. When nonresponse adjustments are made to the final weights, as described in Chapter 8 of SDA, the steps of weighting class adjustments, poststratification, raking, and other adjustments that are used on the final weights need to be repeated for each replicate weight column.

For example, the combination of the *as.svrepdesign* and *postStratify* functions will create replicate weights that reflect poststratification weight adjustments. Let's look at that for the poststratified weights in Example 4.9 of SDA, which we discussed in Section 4.4.

Example 4.9 of SDA. Let's look at creating poststratified replicate weights for an SRS. Here are the steps:

1. Define a design object with the sampling weights, stratification, and clustering. Here, the object *dsrs* is an SRS with weights $3078/300$. Because most replicate weight methods calculate the with-replacement variance, only the first-stage strata and psu information need to be supplied.
2. Choose the variance estimation method (here, JK1), and create a replicate-weights design object.
3. Apply poststratification (function *postStratify*) to the replicate-weights design object. This will poststratify the sampling weights and each variable of replicate weights.

```
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
confint(svymean(~acres92, dsrspjk),df=degf(dsrspjk))
##            2.5 % 97.5 %
## acres92 263069.2 336487
```

```
svytotal(~acres92, dsrspjk)
##             total       SE
## acres92 922717031 57413300
# 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
```

The estimated mean value of *acres92* is 299,778 with a standard error of 18,653, where the jackknife replicate weights are poststratified. Recall that in example 4.9, the poststratified standard error of the mean of *acres92* is 17,513—this value is smaller because it included a finite population correction (fpc), while the replicate weight methods calculate the with-replacement variance.

Note that the final poststratified weights, and each poststratified replicate weight variable, sum to 1054 for the NC region, to 220 for the NE region, to 1382 for the S region, and to 422 for the W region—exactly the poststratification totals defined in the data set *pop.region*. After poststratification, there is no sampling variability for the variables used in the poststratification. Variables associated with the poststratification variables are expected to have reduced variance as well.

Performing multiple steps of nonresponse adjustments. Many surveys have several steps of weighting class adjustments followed by calibration; sometimes intermediate or final weights are trimmed or smoothed so that the weight adjustments do not have "spikes" for some observations. Each step must be repeated for each replicate sampling weight.

1. Start with the sampling weight vector $\mathbf{w}$, which is calculated as the inverse of the probability of selection for each member of the selected sample, whether respondent or nonrespondent.
2. Create R replicate sampling weights $\mathbf{w}_1, \ldots, \mathbf{w}_R$ using the desired replication variance estimation method; if desired, this can be done using the *as.svyrepdesign* function.
3. Now carry out each step of the weight adjustments—weighting class adjustments, weight smoothing or trimming, propensity score adjustments, calibration, or other methods that may be used—on the sampling weight. The final weight variable results from this process.
4. Repeat the operations in Step 3 for each column of replicate weights. Each additional step in the weighting adjustments needs to be carried out separately on each replicate weight.

Some of the functions in the `survey` package, such as *postStratify*, will create replicate weights for the nonresponse adjustments they carry out. For a complicated weighting procedure, we prefer constructing the weights and replicate weights in a custom-written program so that we can see the adjustments at each step.

9.3 Using Replicate Weights from a Survey Data File

Section 9.2 gave examples of how to create your own replicate weights for a survey from the sampling weights, stratification, and clustering information.

Many survey organizations supply data files for which the replicate weights have already been created. These replicate weights have usually already accounted for the nonresponse weighting adjustments. You can analyze these files with the `survey` package, too.

Example 9.5 of SDA. Here, let's suppose that the BRR weights we created for the simple data set in Table 9.2 of SDA have been stored in an external data set and then imported into R. We create *brrdf* using the replicate weights from the design object *dbrrexbrr* that we formed in Section 9.2.1, but then use *brrdf* alone to illustrate how the function *svrepdesign* will create a design object from an imported data set that has replicate weights.

```
# 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
```

In the *svrepdesign* function, we specify the weight variable and tell the function that the replicate weights are in variables whose names are of the form "*repwt*" followed by a number. The argument `type="BRR"` indicates that the replicate weights were formed using the BRR method. The function requires only `weights=` and `repweights=` arguments to calculate point estimates and standard errors. The point estimates are calculated using the full sample weights (in the `weights=` argument), and the replicate weights are used to calculate standard errors. No stratification or clustering information is supplied to the function.

The *svrepdesign* function can also accommodate replicate weights with types JK1, JK2 (a version of jackknife for two-psu-per-stratum designs; see Brick et al., 2000), JKn, bootstrap, the Fay variant of BRR, and other structures. The producer of the survey you are analyzing will include information about the type of replicate weights that were produced and any special considerations that are needed for analysis.

9.4 Summary, Tips, and Warnings

Table 9.1 lists the major R functions used in this chapter.

TABLE 9.1
Functions used for Chapter 9.

Function	Package	Usage
subset	base	Work with a subset of a vector, matrix, or data frame
srswor	sampling	Select an SRS without replacement
svydesign	survey	Specify the survey design
svymean	survey	Calculate mean and standard error of mean
svyratio	survey	Calculate ratio and standard error of ratio
svyquantile	survey	Calculate quantiles and their standard errors
as.svrepdesign	survey	Creates a replicate-weights survey design object from a design object that includes weighting, stratification, and clustering information
svrepdesign	survey	Creates a replicate-weights survey design object from a data frame that contains columns for final and replicate weights

Tips and Warnings

- To create replicate weights for a complex survey for which stratification and clustering information is available, first create a design object with the *svydesign* function, then convert it to a replicate-weights survey design object with the *as.svrepdesign* function.
- If performing nonresponse adjustments, do the adjustments on the sampling weight variable and then on each replicated sampling weight. Then standard errors calculated using the replicate weights will account for the weighting adjustments.
- To analyze data for which replicate weights have already been supplied, create the survey design object with the *svrepdesign* function. You do not need stratification or clustering information to analyze survey data when you have the replicate weights.
- Use the *subset* function to analyze data for a domain. When using replication variance estimation, create the replicate-weights design object first, then apply the *subset* function.