3

Stratified Sampling

In a stratified random sample, the population is divided into subgroups called strata. An SRS is selected independently from each stratum. In this chapter, we look at methods for allocating and selecting stratified random samples using functions in base R and the `sampling` package (Tillé and Matei, 2021). We then discuss the usage of functions *svydesign*, *svymean*, *svytotal*, and *svyby* from the `survey` package (Lumley, 2020) in a stratified random sample.

All code in this chapter can be found in file `ch03.R` on the book website. As always, load the packages `survey`, `sampling`, and `SDAResources` before starting your work.

3.1 Allocation Methods

Data *agpop* contains a stratum variable *region* that describes the census region for each county in the population and takes on the values North Central (NC), Northwest (NE), South (S), and West (W). The following code calculates the population counts (N_h) for the variable *region* with the *table* function.

```
data(agpop)  # load the data set
names(agpop) # list the variable names
##  [1] "county"   "state"    "acres92"  "acres87"  "acres82"  "farms92"
##  [7] "farms87"  "farms82"  "largef92" "largef87" "largef82" "smallf92"
## [13] "smallf87" "smallf82" "region"
head(agpop)  # take a look at the first 6 obsns
##                 county state acres92 acres87 acres82 farms92 farms87 farms82
## 1 ALEUTIAN ISLANDS AREA    AK  683533  726596  764514      26      27      28
## 2        ANCHORAGE AREA    AK   47146   59297  256709     217     245     223
## 3        FAIRBANKS AREA    AK  141338  154913  204568     168     175     170
## 4           JUNEAU AREA    AK     210     214     127       8       8      12
## 5  KENAI PENINSULA AREA    AK   50810   85712   98035      93     119     137
## 6        AUTAUGA COUNTY    AL  107259  116050  145044     322     388     453
##   largef92 largef87 largef82 smallf92 smallf87 smallf82 region
## 1       14       16       20        6        4        1      W
## 2        9       10       11       41       52       38      W
## 3       25       28       21       12       18       25      W
## 4        0        0        0        5        4        8      W
## 5        9       18       17       12       18       19      W
## 6       25       32       32        8       19       17      S
nrow(agpop)  #number of rows, 3078
## [1] 3078
unique(agpop$region) # take a look at the four regions, NC, NE, S, W
## [1] "W"  "S"  "NE" "NC"
table(agpop$region)  # number of counties in each stratum
```

DOI: 10.1201/9781003228196-3

```
##
##   NC   NE    S    W
## 1054  220 1382  422
```

We can use the information about *region* to allocate a stratified sample.

Proportional allocation. With proportional allocation, the stratum sample sizes are proportional to the population stratum sizes N_h. A proportional allocation is easy to calculate in R; simply multiply N_h/N by the desired sample size. For example, region NC has 1054 counties and the population has 3078 counties. For a sample with $n = 300$, proportional allocation will select $300 * 1054/3078 = 103$ counties from region NC. The values in *propalloc* are fractions, so we round these to the nearest integers to obtain the sample size.

```
popsize <- table(agpop$region)
propalloc <- 300*popsize/sum(popsize)
propalloc
##
##       NC       NE        S        W
## 102.7290  21.4425 134.6979  41.1306
# Round to nearest integer
propalloc_int <- round(propalloc)
propalloc_int
##
##  NC  NE   S   W
## 103  21 135  41
sum(propalloc_int) # check that stratum sample sizes sum to 300
## [1] 300
```

Neyman allocation. For Neyman allocation, you need to provide additional information about the stratum variances. Sometimes you have information about a variable that is related to key survey responses from the sampling frame, or sometimes you have information on variances from a pilot study or from similar surveys that have been done. In other cases, you may need to make a conjecture about the stratum variances.

In the following example, we assume that the survey planner does not have the true population variances available, and we enter conjectures for the relative variances of the strata. For example, the variance in the West is set at twice the variance for the South. Using the *popsize* vector that was calculated in the previous code, we have:

```
stratvar <- c(1.1,0.8,1.0,2.0)
# Make sure the stratum variances in stratvar are in same
#  order as the table in popsize
neymanalloc <- 300*(popsize*sqrt(stratvar))/sum(popsize*sqrt(stratvar))
neymanalloc
##
##        NC        NE         S         W
## 101.07640  17.99204 126.36327  54.56828
neymanalloc_int <- round(neymanalloc)
neymanalloc_int
##
##  NC  NE   S   W
## 101  18 126  55
sum(neymanalloc_int)
## [1] 300
```

Optimal allocation. Optimal allocation can be done similarly by defining costs or relative costs for sampling in each stratum.

```
relcost <- c(1.4,1.0,1.0,1.8)
# Make sure the relative costs in relcost are in same
# order as the table in popsize
optalloc <- 300*(popsize*sqrt(stratvar/relcost))/sum(popsize*sqrt(stratvar/relcost))
optalloc
##
##        NC        NE         S         W
##  94.75776  19.95766 140.16833  45.11626
optalloc_int <- round(optalloc)
optalloc_int
##
##  NC  NE   S   W
##  95  20 140  45
sum(optalloc_int)
## [1] 300
```

Table 3.1 summarizes the results of these three allocation methods for the *agpop* population. Of course, the Neyman and optimal allocations are only optimal under the assumed variances and costs used in the calculations. If those variances or costs are wrong, then these allocations will not be optimal for the variable of interest. And an allocation that is optimal for one response variable might not be optimal for another.

TABLE 3.1
Proportional, Neyman, and optimal allocation for the four regions.

Number of Counties in Stratum	NC	NE	S	W	Total
Population	1054	220	1382	422	3078
Sample with proportional allocation	103	21	135	41	300
Sample with Neyman allocation	101	18	126	55	300
Sample with optimal allocation	95	20	140	45	300

Other allocation methods. The sample sizes specified by the proportional, Neyman, and optimal methods are just guidelines. You can set the stratum sample sizes to any values that meet your research needs. For example, if you want to have high precision for comparing stratum means, you may want to select the same number of observations from each stratum.

There are other functions in R that you can use for allocation with stratified data such as functions *strAlloc* from the `PracTools` package (Valliant et al., 2020) and *optiallo* from the `optimStrat` package (Bueno, 2020). The package `SamplingStrata` (Barcaroli, 2014; Barcaroli et al., 2020) provides R functions for determining the optimal stratification and allocation that will achieve predetermined precisions for multiple y variables. For example, you can use the package to design a stratification that will ensure that the coefficients of variation for five key variables do not exceed 0.05. The package `stratification` (Baillargeon and Rivest, 2011; Rivest and Baillargeon, 2017) contains functions for determining stratum boundaries when the stratifying variable is continuous.

3.2 Selecting a Stratified Random Sample

The sample in Example 3.2 of SDA was selected using a spreadsheet but let's look at how to select a similar sample using R, with the *sample* and *strata* functions (this will, of course, give a different sample than obtained in Example 3.2 of SDA).

Using the *sample* function in base R. As we have discussed in Chapter 2, the *sample* function can be used to select an SRS. To select a stratified random sample, we select an SRS independently from each stratum.

Data `agpop.csv` contains a stratum variable *region* that describes the census region for each county in the population. In the following example, we use the proportional allocation from Table 3.1 to divide the $n = 300$ units among the four strata, that is, selecting 103, 21, 135, and 41 counties from regions NC, NE, S, and W, respectively.

```
# Select an SRS without replacement from each region with proportional allocation
# with total size n=300
regionname <- c("NC","NE","S","W")
# Make sure sampsize has same ordering as regionname
sampsize <- c(103,21,135,41)
# Set the seed for random number generation
set.seed(108742)
index <- NULL
for (i in 1:length(sampsize)) {
  index <- c(index,sample((1:N)[agpop$region==regionname[i]],
                          size=sampsize[i],replace=F))
}
strsample<-agpop[index,]
# Check that we have the correct stratum sample sizes
table(strsample$region)
##
##  NC  NE   S   W
## 103  21 135  41
# Print the first six rows of the sample to see
strsample[1:6,]
##                         county state acres92 acres87 acres82 farms92 farms87
## 1316             ISANTI COUNTY    MN  131563  142998  153003     680     817
## 2034           DEFIANCE COUNTY    OH  196759  206905  210781     830     987
## 864            ATCHISON COUNTY    KS  245099  233619  234730     686     694
## 553          DES MOINES COUNTY    IA  192467  210843  224770     681     753
## 1738               DUNN COUNTY    ND 1352738 1358843 1397141     650     733
## 1325 LAKE OF THE WOODS COUNTY    MN  103665  118959  119296     176     222
##      farms82 largef92 largef87 largef82 smallf92 smallf87 smallf82 region
## 1316     947       18       14        8       14       26       34     NC
## 2034    1033       25       20       18       40       50       50     NC
## 864      768       55       42       41       48       48       65     NC
## 553      815       33       30       24       56       56       72     NC
## 1738     697      358      368      361       19       13       34     NC
## 1325     230       30       35       26        4        4        1     NC
```

This simple code used a *for* loop to select an SRS from each stratum (defined by the subset having *region* equal to the stratum name) in turn; alternatively, one could use the *tapply* function, or write a custom R function, to do this without looping. The vector containing the sample sizes must be in the same order as the vector giving the stratum names.

Using the *strata* function from the sampling package. An alternative is to use function *strata* to select a stratified random sample. This function is in the `sampling` package (Tillé and Matei, 2021), which you installed in Chapter 1.

First, sort the data by the stratification variable *region* before selecting the sample. Next, call the *strata* function with sorted data *agpop2* and the stratification variable *region* with first argument `agpop2` and second argument `stratanames="region"`. You can also use a vector of variables to define the strata, such as `stratanames=c("A","B")`, if the strata are formed from multiple variables. Add the information on number of counties to be selected within each stratum by `size=c(103,21,135,41)` in the *strata* function. Finally, choose the method to select the sample within each stratum; for this chapter we use either SRS without replacement (`method="srswor"`) or SRS with replacement (`method="srswr"`).

```
# Sort the population by stratum
agpop2<-agpop[order(agpop$region),]
# Use the strata function to select the units for the sample
# Make sure size argument has same ordering as the stratification variable
index2<-strata(agpop2,stratanames=c("region"),size=c(103,21,135,41),
               method="srswor")
table(index2$region)  # look at number of counties selected within each region
##
##  NC  NE   S   W
## 103  21 135  41
head(index2)
##    region ID_unit       Prob Stratum
## 2      NC       2 0.09772296       1
## 9      NC       9 0.09772296       1
## 27     NC      27 0.09772296       1
## 36     NC      36 0.09772296       1
## 42     NC      42 0.09772296       1
## 43     NC      43 0.09772296       1
strsample2<-getdata(agpop2,index2) # extract the sample
head(strsample2)
##              county state acres92 acres87 acres82 farms92 farms87 farms82
## 526    ADAMS COUNTY    IA  239800  243607  254071     643     688     737
## 533   BREMER COUNTY    IA  236668  235086  250402    1058    1140    1287
## 551  DECATUR COUNTY    IA  261494  278714  300684     648     715     769
## 560  FREMONT COUNTY    IA  302352  308796  306786     596     719     771
## 566   HARDIN COUNTY    IA  332358  337990  355823     986    1065    1208
## 567 HARRISON COUNTY    IA  399155  387190  408601     919    1024    1192
##     largef92 largef87 largef82 smallf92 smallf87 smallf82 region ID_unit
## 526       38       32       21       40       50       33     NC       2
## 533       25       18       11       96      116      109     NC       9
## 551       52       54       56       20       34       37     NC      27
## 560       91       72       51       37       59       50     NC      36
## 566       56       36       42       90      115      132     NC      42
## 567       88       62       51       60       60       66     NC      43
##           Prob Stratum
## 526 0.09772296       1
## 533 0.09772296       1
## 551 0.09772296       1
## 560 0.09772296       1
## 566 0.09772296       1
## 567 0.09772296       1
```

The data frame *index2* contains the stratum variables, the identifiers of the units selected to be in the sample, and the inclusion probability for each unit in the sample. The function *getdata* then extracts the sampled units from the population data.

The *strata* function gives the inclusion probabilities for the sample units but not the weights. You can calculate the sampling weights by taking the reciprocal of the inclusion probabilities. When calculating weights for a stratified random sample, always check that the weights sum to the stratum population sizes. If they do not sum to the stratum population sizes, you have made a mistake somewhere in the weight calculations.

```
# Calculate the sampling weights
# First check that no probabilities are 0
sum(strsample2$Prob<=0)
## [1] 0
strsample2$sampwt<-1/strsample2$Prob
# Check that the sampling weights sum to the population sizes for each stratum
tapply(strsample2$sampwt,strsample2$region,sum)
##   NC   NE    S    W
## 1054  220 1382  422
```

3.3 Computing Statistics from a Stratified Random Sample

Examples 3.2 and 3.6 of SDA. As in Chapter 2, function *svydesign* from the `survey` package can be used to enter the stratified random sample information, and functions *svymean* and *svytotal* will calculate estimated means and totals from a stratified random sample. The data set *agstrat* is a stratified random sample taken from the population data *agpop* with proportional allocation. First, let's look at the data.

```
data(agstrat)
names(agstrat)  # list the variable names
##  [1] "county"   "state"    "acres92"  "acres87"  "acres82"  "farms92"
##  [7] "farms87"  "farms82"  "largef92" "largef87" "largef82" "smallf92"
## [13] "smallf87" "smallf82" "region"   "rn"       "strwt"
agstrat[1:6,1:8] # take a look at the first 6 obsns from columns 1 to 8
##     county state acres92 acres87 acres82 farms92 farms87 farms82
## 1 PIERCE C    NE  297326  332862  319619     725     857     865
## 2 JENNINGS    IN  124694  131481  139111     658     671     751
## 3 WAYNE CO    OH  246938  263457  268434    1582    1734    1866
## 4 VAN BURE    MI  206781  190251  197055    1164    1278    1464
## 5  OZAUKEE    WI   78772   85201   89331     448     483     527
## 6 CLEARWAT    MN  210897  229537  213105     583     699     693
nrow(agstrat)  # number of rows, 300
## [1] 300
unique(agstrat$region) # take a look at the four regions, NC, NE, S, W
## [1] "NC" "NE" "S"  "W"
table(agstrat$region)  # number of counties in each stratum
##
##  NC  NE   S   W
## 103  21 135  41
# check that the sum of the weights equals the population size
sum(agstrat$strwt)  #3078
## [1] 3078
```

Figure 3.1 gives a boxplot for variable *acres92* (scaled to millions of acres). We can see that the West region has the highest median and largest variability, while the Northeast region has the lowest median and smallest variability. Note that we can use the *boxplot* function in the following code because an SRS is taken within each stratum (and, because of proportional allocation, the sample is approximately self-weighting); for other designs, one should incorporate the weights into the plot as shown in Chapter 7.

```
boxplot(acres92/10^6 ~ region, xlab = "Region", ylab = "Millions of Acres",
        data = agstrat)
# notice the large variability in western region
```

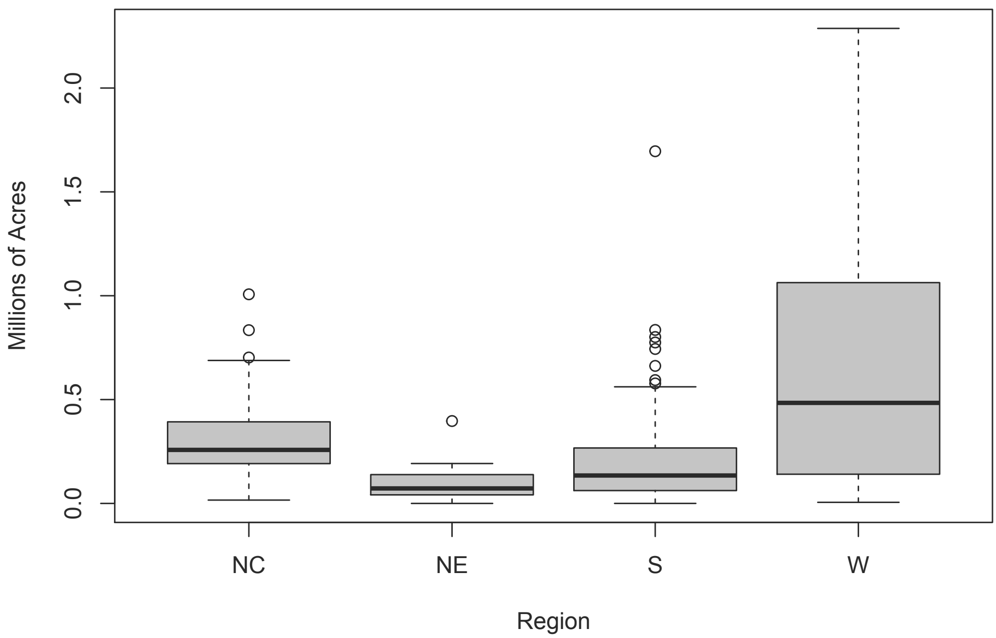

FIGURE 3.1: Boxplot of 1992 acreage by region (data *agstrat*).

Now let's calculate the estimates from the stratified random sample. We use function *svydesign* to input the design information and functions *svymean* and *svytotal* to calculate the survey statistics. The following gives the code to find estimates for data *agstrat* along with the output of the statistics calculated. These estimates were given in Examples 3.2 and 3.6 of SDA.

First, we set up the information for the survey design.

```
# create a variable containing population stratum sizes, for use in fpc (optional)
# popsize_recode gives popsize for each stratum
popsize_recode <- c('NC' = 1054, 'NE' = 220, 'S' = 1382, 'W' = 422)
# next statement substitutes 1054 for each 'NC', 220 for 'NE', etc.
agstrat$popsize <- popsize_recode[agstrat$region]
table(agstrat$popsize)  #check the new variable
##
```

```
##  220  422 1054 1382
##   21   41  103  135
# input design information for agstrat
dstr <- svydesign(id = ~1, strata = ~region, weights = ~strwt, fpc = ~popsize,
                  data = agstrat)
dstr
## Stratified Independent Sampling design
## svydesign(id = ~1, strata = ~region, weights = ~strwt, fpc = ~popsize,
##     data = agstrat)
```

The syntax is similar to that for an SRS. The only difference is in the arguments to the *svydesign* function. The arguments used for stratification are as follows:

id As for an SRS, we use `id = ~1` to indicate that there is no clustering.

strata The *strata* argument gives the variable name(s) containing the stratification information (here, the stratification variable is *region*).

weights For stratified sampling, we use the weights associated with the selection probabilities in each stratum. This example has a stratified sample with proportional allocation, where the weights are almost identical for all strata. In samples with disproportionate stratification, however, the weights will vary across strata, and estimates calculated without weights will be biased.

fpc The *fpc* argument specifies the variable that contains information for calculating the finite population correction (fpc) in each stratum. The easiest way to do that is to create a new variable in the data frame that contains the population stratum sizes. Our code defines a variable *popsize_recode* that associates each stratum name with its population size; alternatively, the variable *popsize* could be created with the *merge* or *match* function or by assigning values separately to each region in a *for* loop.

If you omit the *fpc* argument (and still include the *weights* argument), the estimates of means and totals are the same, but standard error estimates are without the finite population correction.

All of the work specifying the design information is done in the *svydesign* function; after you have defined the design there, the *svymean* and *svytotal* functions are used exactly as in Chapter 2 for an SRS.

```
# calculate mean, SE and confidence interval
smean<-svymean(~acres92, dstr)
smean
##           mean    SE
## acres92 295561 16380
confint(smean, level=.95, df=degf(dstr)) # note that df = n-H = 300-4
##          2.5 %   97.5 %
## acres92 263325 327796.5
# calculate total, SE and CI
stotal<-svytotal(~acres92, dstr)
stotal
##             total       SE
## acres92 909736035 50417248
degf(dstr) # Show the degrees of freedom for the design
## [1] 296
# calculate confidence intervals using the degrees of freedom
confint(stotal, level=.95,df= degf(dstr))
##             2.5 %     97.5 %
```

```
## acres92 810514350 1008957721
```

The output is pretty self-explanatory. Note that 296 degrees of freedom (df $= n - H$) are used for the confidence intervals. The df can also be found by applying function *degf* to the design object *dstr*, that is, `degf(dstr)`. If you want to calculate confidence intervals that are based on the normal distribution, simply omit the *df* argument in the *confint* function. If the sample has few observations, however, we need to specify the degrees of freedom and use the t distribution to calculate confidence intervals.

Weights and fpc arguments. We supplied both *weights* and *fpc* arguments to the *svydesign* function in this example, but for a stratified random sample with no nonresponse, the *svydesign* function will calculate weights from the *fpc* information and the sample sizes in the data set. The design object *dstrfpc* in the following code results in the same statistics as the design object *dstr* (with the *weights* and *fpc* arguments) that we used earlier. Including the *weights* argument but omitting the *fpc* argument results in standard errors that are calculated without the fpc. (Do not omit both *weights* and *fpc*; then the *svydesign* function will assume all weights are equal.)

```
# Alternative design specifications
# Get same result if omit weights argument since weight = popsize/n_h
dstrfpc <- svydesign(id = ~1, strata = ~region, fpc = ~popsize, data = agstrat)
svymean(~acres92, dstrfpc)
##            mean    SE
## acres92 295561 16380
# If you include weights but not fpc, get SE without fpc factor
dstrwt <- svydesign(id = ~1, strata = ~region, weights = ~strwt, data = agstrat)
svymean(~acres92, dstrwt)
##            mean    SE
## acres92 295561 17241
```

Calculating stratum means and variances. Function *svyby* will calculate statistics and their standard errors for subgroups of the data. Here we use it to calculate the stratum means and totals. The first argument of *svyby* is the formula for the variable(s) for which statistics are desired, and the second argument (`by=`) is the variable that defines the groups. Then list the design object and the name of the function that calculates the statistics. Set `keep.var=TRUE` to display the standard errors for the statistics.

```
# calculate mean and se of acres92 by regions
svyby(~acres92, by=~region, dstr, svymean, keep.var = TRUE)
##    region   acres92       se
## NC     NC 300504.16 16107.59
## NE     NE  97629.81 18149.49
## S       S 211315.04 18925.35
## W       W 662295.51 93403.65
# calculate total and se of acres92 by regions
svyby(~acres92, ~region, dstr, svytotal, keep.var = TRUE)
##    region   acres92       se
## NC     NC 316731380 16977399
## NE     NE  21478558  3992889
## S       S 292037391 26154840
## W       W 279488706 39416342
```

If you want to check the calculations by formula, you can also calculate summary statistics directly for each stratum using the *tapply* function and then use the formulas from SDA to calculate the standard errors for each estimated stratum mean or total. The variances of the stratum means are calculated with the formula $(1 - n_h/N_h)s_h^2/n_h$, where n_h and N_h are the sample and population sizes, and s_h^2 is the sample variance within stratum h.

```
# formula calculations, using tapply
# variables sampsize and popsize were calculated earlier in the chapter
# calculate mean within each region
strmean<-tapply(agstrat$acres92,agstrat$region,mean)
strmean
##        NC        NE         S         W
## 300504.16  97629.81 211315.04 662295.51
# calculate variance within each region
strvar<-tapply(agstrat$acres92,agstrat$region,var)
strvar
##           NC           NE            S            W
##  29618183543   7647472708  53587487856 396185950266
# verify standard errors by direct formula
strse<- sqrt((1-sampsize/popsize)*strvar/sampsize)
# same standard errors as from svyby
strse
##
##       NC       NE        S        W
## 16107.59 18149.49 18925.35 93403.65
```

3.4 Estimating Proportions from a Stratified Random Sample

A proportion is a special case of a mean of a variable taking on values 1 and 0. As defined in Chapter 2, variable *lt200k* takes on value 1 if *acres92* $<$ 200,000 and takes on value 0 if *acres92* $\geq$ 200,000. The mean of variable *lt200k* estimates the proportion of farms that have fewer than 200,000 acres. The total of variable *lt200k* estimates the number of farms that have fewer than 200,000 acres.

```
# Create variable lt200k
agstrat$lt200k <- rep(0,nrow(agstrat))
agstrat$lt200k[agstrat$acres92 < 200000] <- 1
# Rerun svydesign because the data set now has a new variable
dstr <- svydesign(id = ~1, strata = ~region, fpc = ~popsize,
  weights = ~strwt, data = agstrat)
# calculate proportion, SE and confidence interval
smeanp<-svymean(~lt200k, dstr)
smeanp
##           mean     SE
## lt200k 0.51391 0.0248
confint(smeanp, level=.95,df=degf(dstr))
##            2.5 %    97.5 %
## lt200k 0.4651188 0.5627107
# calculate total, SE and CI
stotalp<-svytotal(~lt200k, dstr)
stotalp
##         total     SE
## lt200k 1581.8 76.318
confint(stotalp, level=.95,df=degf(dstr))
##           2.5 %   97.5 %
## lt200k 1431.636 1732.024
```

You can also calculate proportions and totals of categorical variables by defining them to be factors, either by declaring the variable to be a factor variable in the data set or in the function call of *svymean*. Here we define variable *lt200kf* to be a factor variable in the data set.

```
# Create a factor variable lt200kf
agstrat$lt200kf <- factor(agstrat$lt200k)
# Rerun svydesign because the data set now has a new variable
dstr <- svydesign(id = ~1, strata = ~region, fpc = ~popsize,
                  weights = ~strwt, data = agstrat)
# calculate proportion, SE and confidence interval
smeanp2<-svymean(~lt200kf, dstr)
smeanp2
##              mean     SE
## lt200kf0 0.48609 0.0248
## lt200kf1 0.51391 0.0248
confint(smeanp2, level=.95,df=degf(dstr))
##               2.5 %    97.5 %
## lt200kf0 0.4372893 0.5348812
## lt200kf1 0.4651188 0.5627107
# calculate total, SE and CI
stotalp2<-svytotal(~lt200kf, dstr)
stotalp2
##            total     SE
## lt200kf0 1496.2 76.318
## lt200kf1 1581.8 76.318
confint(stotalp2, level=.95,df=degf(dstr))
##              2.5 %   97.5 %
## lt200kf0 1345.976 1646.364
## lt200kf1 1431.636 1732.024
```

Note that the *svytotal* function gives the estimated total for each category of variable *lt200kf*.

The `survey` package will also estimate asymmetric confidence intervals for survey data (Korn and Graubard, 1998), which may have more accurate coverage probabilities for proportions that are near 0 or 1 than the symmetric confidence intervals based on the normal approximation. This is done with the *svyciprop* function, choosing `method="beta"` to obtain a version of the Clopper-Pearson confidence interval (the function will also compute asymmetric confidence intervals using other methods). We illustrate with binary variable *lt200k*. Note that you need to list the formula as `~I(lt200k)` or `~I(lt200k==1)`.

```
# calculate proportion and confidence interval with svyciprop
svyciprop(~I(lt200k==1), dstr, method="beta")
##                        2.5% 97.5%
## I(lt200k == 1) 0.514 0.464  0.56
```

3.5 Additional Code for Exercises

Some of the exercises in Chapter 3 ask you to find an ANOVA table. Here's how to do that for *agstrat* using the *lm* function, which performs regression and analysis of variance. The first argument of *lm* is the formula for the regression model, of the form $y \sim x$. We specify *region* to be a factor so that the function will treat it as a categorical variable. (We'll see the function that conducts regression analyses for survey data in Chapter 4, and it will have a similar structure.)

```
myfit <- lm(acres92~factor(region), data=agstrat)
anova(myfit)
## Analysis of Variance Table
##
## Response: acres92
##                  Df     Sum Sq    Mean Sq F value    Pr(>F)
## factor(region)   3 7.2976e+12 2.4325e+12   27.48 1.048e-15 ***
## Residuals      296 2.6202e+13 8.8521e+10
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.6 Summary, Tips, and Warnings

Table 3.2 lists the major functions used in this chapter to select or analyze data from a stratified random sample.

Tips and Warnings

- When calculating optimal allocations, make sure that the variables containing the cost and variance information are in the same order as the variable(s) containing the stratum identifiers.
- Sort the population data set by the stratification variable(s) before calling the *strata* function to select a stratified sample.
- When calculating sampling weights for a stratified random sample, check that the sum of the sampling weights for each stratum equals the population size for that stratum.
- When analyzing data from a stratified random sample, first create the design object in the *svydesign* function, using the `strata=` argument. Then call the *svymean* and *svytotal* function with that design object.
- Functions *svymean*, *svytotal*, and *svyby* can be used to calculate statistics for two or more variables simultaneously. For example, `svymean(~acres92 + acres87, dstr)` will display statistics of both variables *acres92* and *acres87*.

TABLE 3.2
Functions used for Chapter 3.

Function	Package	Usage
order	base	Give indices for data sorted according to the specified variable
sample	base	Select a simple random sample with or without replacement
tapply	base	Apply a function to each group of values; groups are defined by the second argument
confint	stats	Calculate confidence intervals; add df for t confidence interval
lm	stats	Fit a linear model to a data set (not using survey methods)
anova	stats	Calculate an analysis of variance table from a model object
boxplot	graphics	Draw boxplot of data (used to display strata in a stratified random sample)
strata	sampling	Select a stratified random sample
getdata	sampling	Extract the sampled units from the population
svydesign	survey	Specify the survey design; add stratum information for stratified random sample
degf	survey	Find degrees of freedom based on design information
svymean	survey	Calculate mean and standard error of mean (if the variable is numeric), or proportion in each category (if variable is categorical)
svytotal	survey	Calculate total and standard error of total
svyby	survey	Calculate survey statistics on subsets of a survey defined by factors
svyciprop	survey	Compute confidence intervals for proportions using various methods (if estimated proportions are close to 0 or 1, sometimes an asymmetric confidence interval is preferred to the symmetric confidence interval produced by *svymean*)