Bayesian SAE using Complex Survey Data

Lecture5B: Survey sampling in R

Richard Li

Department of Statistics
University of Washington

Outline

Survey sampling with the `survey` package

The `survey` package

- Written by Thomas Lumley
- Site for the package `http://r-survey.r-forge.r-project.org/survey/`
- Book "Complex Surveys: a guide to analysis using R": `http://r-survey.r-forge.r-project.org/svybook/index.html`

Academic Performance Index (API) data

- This is a useful pedagogic dataset because the complete population data are available, along with various probability samples.
- From the help file (Type “?api” at the console): The Academic Performance Index is computed for all California schools based on standardised testing of students.
- The data sets contain information for all schools with at least 100 students.
- The API outcome is constructed from standardized tests.
- Full data: 6194 observations on the 37 variables including ‘cds‘ (unique identifier), ‘stype‘ (Elementary/Middle/High), ‘api00‘ (API in 2000),...

Academic Performance Index (API) data

- From “?api”:
  - ‘apipop’ is the entire population
  - ‘apisrs’ is a simple random sample
  - ‘apiclus1’ is a cluster sample of school districts
  - ‘apistrat’ is a sample stratified by “stype” (school type)
  - ‘apiclus2’ is a two-stage cluster sample of schools within districts
- Each row of the data contains data on one school, i.e. contains information on the children within that school

Academic Performance Index (API) data

We first look at the SRS dataset, since we can use regular R methods (i.e., no weighting)

```
library(survey)
data(api)
names(apisrs)

##  [1] "cds"      "stype"    "name"     "sname"    "snum"     "dname"
##  [8] "cname"    "cnum"     "flag"     "pcttest"  "api00"    "api99"
## [15] "growth"   "sch.wide" "comp.imp" "both"     "awards"   "meals"
## [22] "yr.rnd"   "mobility" "acs.k3"   "acs.46"   "acs.core" "pct.res
## [29] "hsg"      "some.col" "col.grad" "grad.sch" "avg.ed"   "full"
## [36] "enroll"   "api.stu"  "pw"       "fpc"

dim(apisrs)

## [1] 200  39
```

200 schools are randomly sampled

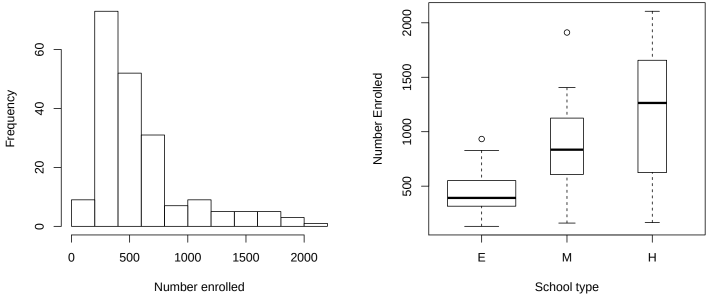
API SRS data: exploratory analysis

We find out the size of the data frame ‘apisrs‘, and then evaluate the mean and variance of the ‘enroll‘ variable, which is the number of children enrolled in the school.

```
dim(apisrs)

## [1] 200  39

ybar <- mean(apisrs$enroll)
sd <- sqrt(var(apisrs$enroll))
ybar

## [1] 584.61

sd

## [1] 393.4514
```

API SRS data: exploratory analysis

```
par(mfrow = c(1, 2))
hist(apisrs$enroll, xlab = "Number enrolled", main = "")
boxplot(apisrs$enroll ~ factor(apisrs$stype, levels = c("E",
    "M", "H")), xlab = "School type", ylab = "Number Enrolled",
    main = "", varwidth = TRUE)
```



API SRS data: exploratory analysis

Note the standard error is for the infinite population mean parameter, not the finite sample mean, since no finite population correction factor

```
se <- sd/sqrt(length(apisrs$enroll))

# 95% confidence interval
ybar - qnorm(0.975) * se

## [1] 530.0814

ybar + qnorm(0.975) * se

## [1] 639.1386

# We know the population mean here
mean(apipop$enroll, na.rm = T)

## [1] 619.0469
```

Cluster sample

- There are 757 districts in total, and 15 are sampled.
- All schools within the district were then sampled, to give 183 schools in total
- The weight is calculated from $w_k = \pi_k^{-1}$
- So weight should be $757/15 = 50.47$ but reported as (‘pw‘) 33.847
- The sampling weights in ‘apiclus1’ are incorrect but are as obtained from UCLA

Cluster sample

```
length(unique(apipop$dname))

## [1] 757

length(unique(apiclus1$dname))

## [1] 15

dim(apiclus1)

## [1] 183  39

table(apiclus1$pw)

##
## 33.846996307373
##             183

table(apiclus1$dnum)

##
##  61 135 178 197 255 406 413 437 448 510 568 637 716 778 815
```

Stratified random sample

- The sample 'apistrat' is stratified on school types
- 100 elementary, 50 middle and 55 high schools

```
dim(apistrat)

## [1] 200  39

table(apistrat$pw)

##
## 15.1000003814697 20.3600006103516 44.2099990844727
##               50               50              100

table(apistrat$stype)

##
##   E   H   M
## 100  50  50
```

Example: Define survey object in R

Define a survey object

First step in analyzing survey data within R is to define a ‘survey‘ object using the ‘svydesign’ function. From the help file (edited):
‘svydesign(ids, probs=NULL, strata = NULL, variables = NULL, fpc=NULL, data = NULL,...)‘

- ‘ids’: Formula or data frame specifying cluster ids from largest level to smallest level, $\sim$0 or $\sim$1 is a formula for no clusters.
- ‘probs’: Formula or data frame specifying cluster sampling probabilities
- ‘strata’: Formula or vector specifying strata, use NULL for no strata
- ‘variables’: Formula or data frame specifying the variables measured in the survey. If NULL, the data argument is use
- ‘fpc’: Finite population correction (may be size or fraction in each stratum)
- ‘weights’: Formula or vector specifying sampling weights as an alternative to prob
- ‘data’: Data frame to look up vars in the formula arguments

Define a survey object: Simple random sample

```
srs_design <- svydesign(id = ~1, fpc = ~fpc, data = apisrs)
table(apisrs$fpc)

##
## 6194
##  200
```

- The id=∼1 says that individual schools were sampled (there is one row for each school in the data set)
- The data=apisrs argument specifies where the data are found
- The fpc=∼ fpc says that the variable called ‘fpc‘ in the dataset contains the population size for each stratum (in SRS, just population size)

Define a survey object: Cluster sample

```
clus_design <- svydesign(id = ~dnum, weights = ~pw,
    data = apiclus1, fpc = ~fpc)
table(apiclus1$fpc)

##
## 757
## 183
```

- The id=∼dnum says that District is the cluster identifier
- The weights gives the sampling weights (but remember it is wrongly calculated in the data!)
- The fpc=∼ fpc says that the variable called ‘fpc‘ in the dataset contains the **total number of clusters**

Define a survey object: Stratified random sample

```r
strat_design <- svydesign(id = ~1, strata = ~stype,
    weights = ~pw, data = apistrat, fpc = ~fpc)
```

- The strata=∼stype gives the stratum variable, which is school type
- The weights do not need to specify as the population size and sample size are available
- The fpc=∼ fpc says that the variable called ‘fpc‘ in the dataset contains the population size for each stratum

Define a survey object: Stratified random sample

- Notice we could leave off one of "weights" or "fpc"
- If both are left off then *equal sampling probabilities* are assumed (which would be incorrect here)

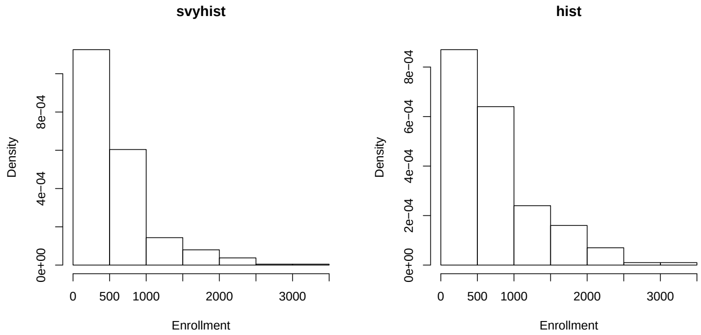
```
table(apistrat$pw)

##
## 15.1000003814697 20.3600006103516 44.2099990844727
##               50               50              100

table(apistrat$fpc)

##
##  755 1018 4421
##   50   50  100
```

- weights and fpc: $755/50 = 15.1$, $1018/50 = 20.36$, $4421/100 = 44.2$

Histogram accounting for weights

Compare the ‘svyhist’ command and the regular ‘hist’ command (which does not account for non simple random sampling)

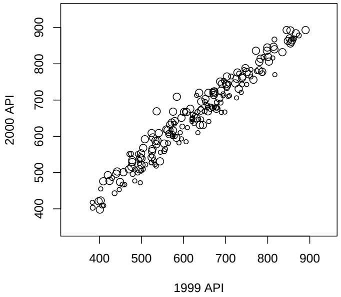
```r
par(mfrow = c(1, 2))
svyhist(~enroll, design = strat_design, xlab = "Enrollment",
    main = "svyhist")
hist(apistrat$enroll, xlab = "Enrollment", main = "hist",
    freq = FALSE)
```

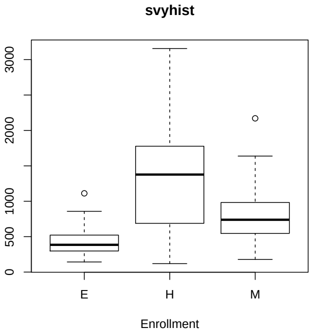


More visualization

'svyboxplot' and 'svyplot' (bubble area proportional to the sampling weight)

```
par(mfrow = c(1, 2))
svyplot(api00 ~ api99, design = strat_design, style = "bubble",
    xlab = "1999 API", ylab = "2000 API")
svyboxplot(enroll ~ stype, design = strat_design, xlab = "Enrollment",
    main = "svyhist")
```





Example: Estimate population mean

Simple random sample

by hand

```
n <- dim(apisrs)[1]
mean <- mean(apisrs$enroll)
fpc <- 1 - n/6194
se <- sqrt(var(apisrs$enroll)/n) * sqrt(fpc)
print(c(mean, se))

## [1] 584.61000  27.36837
```

use function

```
srsmean <- svymean(~enroll, srs_design)
srsmean

##           mean     SE
## enroll 584.61 27.368
```

Stratified random sample

Stratification reduces the standard error compared to SRS, by about 1/3 in this case.

```
stramean <- svymean(~enroll, strat_design)
sqrt(vcov(stramean)/vcov(srsmean))

##           enroll
## enroll 0.6762739
```

Similarly, to estimate the total enrollment

```
svytotal(~enroll, strat_design)

##          total     SE
## enroll 3687178 114642

svytotal(~stype, strat_design)  # no uncertainty since it is kno

##        total SE
## stypeE  4421  0
## stypeH   755  0
## stypeM  1018  0
```

Optimal allocation

- Suppose we wish to carry out stratified sampling of 200 schools with strata being school type
- We can find the optimal allocation by estimating the variances from the complete data
- Of course, this would be unknown in real design situation

```
sigma <- aggregate(enroll ~ stype, data = apipop, function(x) {
    sd(x)
})
sigma

##   stype   enroll
## 1     E 175.8747
## 2     H 688.3190
## 3     M 438.6117

count <- aggregate(enroll ~ stype, data = apipop, length)
n.h <- 200 * sigma[, 2] * count[, 2]/sum(sigma[, 2] *
    count[, 2])
n.h

## [1] 89.25641 59.66358 51.08001
```

Optimal allocation: create a stratified random sample

- So 89 elementary, 60 high school, and 51 middle schools is the optimal allocation for estimating the total enrollment

```r
mystrata <- stratsample(apipop$stype, counts = c(E = 89,
    M = 51, H = 60))
mystrat_data <- apipop[mystrata, ]
mystrat_data$Nstrat <- table(apipop$stype)[mystrat_data$stype]
strat_design2 <- svydesign(id = ~1, strata = ~stype,
    fpc = ~Nstrat, data = mystrat_data)
```

Optimal allocation: create a stratified random sample

```
summary(strat_design2)

## Stratified Independent Sampling design
## svydesign(id = ~1, strata = ~stype, fpc = ~Nstrat, data = mystrat_da
## Probabilities:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02013 0.02013 0.05010 0.04557 0.07947 0.07947
## Stratum Sizes:
##             E  H  M
## obs        89 60 51
## design.PSU 89 60 51
## actual.PSU 89 60 51
## Population stratum sizes (PSUs):
##    E    H    M
## 4421  755 1018
## Data variables:
##  [1] "cds"       "stype"    "name"     "sname"    "snum"     "dname"
##  [8] "cname"     "cnum"     "flag"     "pcttest"  "api00"    "api99"
## [15] "growth"    "sch.wide" "comp.imp" "both"     "awards"   "meals"
## [22] "yr.rnd"    "mobility" "acs.k3"   "acs.46"   "acs.core" "pct.res
## [29] "hsg"       "some.col" "col.grad" "grad.sch" "avg.ed"   "full"
```

Optimal allocation: create a stratified random sample

- The SE may be larger than the "non-optimal" design previously, because of sampling variability (variances are estimated)

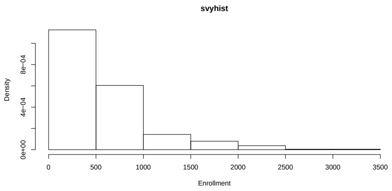
```
svytotal(~enroll, strat_design2, na.rm = T)

##           total     SE
## enroll 3895305 118629

# in comparison
svytotal(~enroll, strat_design, na.rm = T)

##           total     SE
## enroll 3687178 114642

svytotal(~enroll, srs_design, na.rm = T)

##           total     SE
## enroll 3621074 169520
```

Additional topic: Distribution of data and of estimator

Distribution of data and of estimator

- In all the examples so far, we assume the sample sizes are sufficiently large, so that we can apply central limit theorem and construct the confidence intervals.
- In survey sampling contexts the population distributions can be highly skewed (e.g., see below) and larger samples than we are used to in ‘regular’ situations may be needed.
- See Lohr (2010, Section 2.5) for more discussion.

```
svyhist(~enroll, design = strat_design, xlab = "Enrollment",
    main = "svyhist")
```



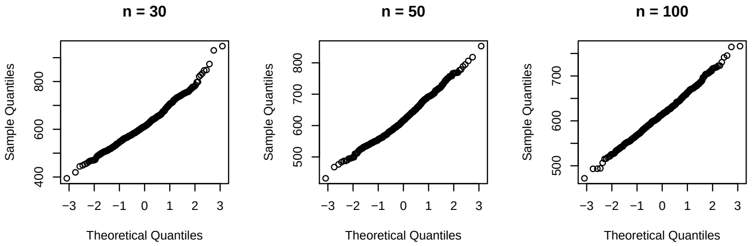
Distribution of data and of estimator

- we create random sample of size 30, 50, 100, or 200 of the number of students enrolled in the 6194 schools in the population
- We replicate the random sampling 500 times

```
set.seed(1)
one.mean <- function(n) {
    srs_rows <- sample(6194, n)
    mean(apipop$enroll[srs_rows], na.rm = TRUE)
}
one.mean(50)

## [1] 616.2041

means.n30 <- replicate(500, one.mean(30))
means.n50 <- replicate(500, one.mean(50))
means.n100 <- replicate(500, one.mean(100))
```

Distribution of data and of estimator

```r
par(mfrow = c(1, 3))
qqnorm(means.n30, main = "n = 30")
qqnorm(means.n50, main = "n = 50")
qqnorm(means.n100, main = "n = 100")
```


n = 30
Sample Quantiles
Theoretical Quantiles
n = 50
Sample Quantiles
Theoretical Quantiles
n = 100
Sample Quantiles
Theoretical Quantiles