Illustration of the benefits of stratification

Introduction

Of the examples included in this package, this is the simplest one. It illustrates the benefit of stratification by comparing a simple random sample (SRS) of, say, 1000 units from the whole population to a sample allocated to each stratum by some method (see below), with the same total sample size. Many (B) samples are selected, and estimates of mean income (inc) are collected and used to get a Monte Carlo approximation of the true variance. In these simple cases, the true variances can be obtained exactly; these are included as a check. The outputs of the following code are:

- varsrs: MC variance of SRS
- varsrstrue: exact variance of SRS
- varstr: MC variance of stratified SRS
- varstropttrue: exact variance of stratified SRS

The program includes an option to see the impact of an excellent (but very unrealistic) stratification. This is done by uncommenting the following line. Note that uncommenting the line overwrites inc. If you want to restore it, run "inc <- inctemp" at the end

```
#inctemp <- inc; inc <- sort(inc00)
```

Notes:

(1) In the sort() statement in the previous line, do not use the inc variable because it has many zeros. Instead, use inc00, which is the original version of inc before some values (for some of the people who are not employed) were set to zero.

(2) Why is the stratificiation based on sort(inc00) so unrealistic? And why is it so good? It's unrealistic because in "real life" we cannot stratify the population by the variable of interest. It's so good because the stratification we get in this case is excellent: the lowest income people are all in the same stratum, the highest income people are also all in the same stratum, and the strata in between are also homogeneous.

(3) For students of sampling theory, it is informative to look at the sample size allocated to each stratum (nstr). Why are they so different?

(4) There are two key parameters. First, there is the desired overall sample size (nexpected). Second, there is the number of samples (B) we would like to obtain to get a Monte Carlo estimate of the variance. Increasing B will tend to bring the MC variance closer to the true variance.

Before doing anything else, we need to make sure a population file is available. If the file was saved as `onemonth.RData`, we load it as follows:

```
load("../onemonth.RData")
```

Next we set two parameters.

```
nexpected <- 1000  # desired overall sample size
B <- 10000  # number of samples to select; increase to get simul. closer to true
```

Next, let's evaluate the quantity we would like to estimate, namely, the mean income of the population.

```r
meaninc <- mean(inc)  # the value we are trying to estimate
```

In both the SRS and stratified SRS cases, we would like to know how good an estimate the procedure is likely to produce. In both cases, our estimator happens to be unbiased, so a good way of assessing the quality of the estimator is to look at its variance. In these simple cases, sampling theory gives us an exact formula for the variance (which we will compute below). But as an interesting alternative (interesting because the same approach can be used in much more complex situations), we can use simulation to get a good approximation to the true variance. We do this by selecting many (B) samples, thereby producing B estimates, which can then be used to approximate the true variance. To get a better approximation, we simply increase the value of B. Note that a value of B that yields a good approximation for SRS may not be big enough for stratified SRS.

Recall that our main goal is to see how much stratification improves the quality of estimates. So, before selecting *stratified* samples, we will select unstratified (SRS) ones to get a baseline for comparison. This is easy to do in R thanks to the sample() function. For each of the B samples, we simply compute mean income and accumulate its squared difference from the population mean income.

```r
varsrs <- 0  # will contain the (Monte Carlo) SRS variance

for (b in 1:B) {
  varsrs <- varsrs + (mean(inc[sample(1:HCM, nexpected)]) - meaninc)^2
}
varsrs <- varsrs/B
```

Next we use the formula from sampling theory to compute the true variance, and then compare the two variances by looking at their relative difference. It should be close to 0.

```r
# Compare to the theoretical value to make sure this is correct:
varsrstrue <- (1-nexpected/HCM) * var(inc) / nexpected
cat("Relative difference of true and simulation-based SRS variance:",
    1-varsrs/varsrstrue, "\n")
```

```
## Relative difference of true and simulation-based SRS variance: -0.009452689
```

We are now ready to make the comparison to *stratified* simple random sampling. We will use the existing stratification variable (stratum) that already exists in the population file. To determine the sample size in each stratum, we consider three options. For proportional allocation, as the name suggests, the sample allocated to a stratum is proportional to that stratum's size (HM[h]) relative to the whole population (HCM), so we would use

```
nstr <- round(HM/HCM*nexpected)
```

For equal allocation, each stratum gets the same sample size, so we would use

```
nstr <- rep(round(nexpected/H), H)
```

We will actually use the third option, Neyman allocation:

```r
NhSh <- as.numeric(by(inc, stratum, sd))*HM
nstr <- round(NhSh/sum(NhSh) * nexpected)

nall <- sum(nstr)  # overall sample size (may be off due to rounding)
```

Before proceeding, we need to know the start and end of each stratum, so we load a function and use it for that purpose.

```r
source("../R/functions.R")
stratrange <- cumrange(HM)
```

In stratified random sampling, we select an independent sample in each stratum, and then we use the income values of the sampled units to estimate the overall (population) mean income. To work out the results for the case of stratified simple random sampling is a bit more complicated than in the SRS case because we have to deal with each stratum separately and then combine the results.

```
samp_h <- numeric(H)  # will contain estimates of stratum means
varstr <- 0  # will contain the (Monte Carlo) stratified SRS variance

for (b in 1:B) {
   # estimated means by stratum
   for (h in 1:H) {
     samp_h[h] <-
       mean(inc[stratrange[h, 1]:stratrange[h, 2]] |> sample(size=nstr[h]))
   }

   inchat_str <- sum(HM * samp_h)/HCM # stratified SRS estimate of mean income
   varstr <- varstr + (inchat_str-meaninc)^2
}
varstr <- varstr/B
```

As in the SRS case, the formula for the true variance is available, so we use it and then compare the two variances using their relative difference.

```
NhShsq <- as.numeric(by(inc, stratum, var))*HM
varstropttrue <- ((sum(NhSh))^2/nall - sum(NhShsq)) / HCM^2
cat("Relative difference of true and simulation-based StratSRS variance:",
    1-varstr/varstropttrue, "\n")  # should be close to 0
```

```
## Relative difference of true and simulation-based StratSRS variance: -0.01456211
```

We are now ready to show how much improvement we get by stratifying our population.

```
##  ----------------------------------------------------------
##  Ratio of StratSRS variance to SRS variance (true):  0.9595503
##  -----------------------------------------------------------
```

```
##  ----------------------------------------------------------------
##  Ratio of StratSRS variance to SRS variance (simulation):  0.9644072
##  ----------------------------------------------------------------
```

With the defaults, we get about 0.96, i.e., there is a four percent gain (the simulation-based value is likely to be somewhat different). However, in the extreme (but unrealistic) case based on sort(inc00) we get about 0.0315, a *huge* gain in efficiency.