Illustration of how a good stratification deteriorates over time

Introduction

In this example, we use the longitudinal synthetic population to illustrate how a good stratification of the population becomes less efficient over time, as the characteristics of the people in the population gradually change. We will not use the stratum variable included in the data file. Rather we create a new stratification based on the income variable at time (month) 1: the first stratum wiil have all the lowest-income people, the last stratum will have all the highest income people, with the obvious gradation in between. A person's income can change from month to month, so some people in the first stratum will eventually have high incomes, and so on for other strata. The illustration is done in two ways: (1) empirically (or, more precisely, using Monte Carlo simulation) and (2) using results from the theory of survey sampling.

For (1), after the stratification is done, many stratified samples are selected, with each sample producing an estimate of mean income for each month (we use the first month of each year as our sampling month). Because we have the whole population, we know the true monthly mean incomes. Therefore we simply accumulate $(estimate - true)^2$ over all samples. Dividing this by the number of samples, we get estimates of the mean squared errors for each month in the survey (they are actually variances since the estimators are unbiased).

For (2), we use the formula for the variance of a stratified SRS based estimator (available in any good book on the theory of survey sampling). Because we know the true population, we can obtain *exact* variances. As a bonus, we can compare the results of (1) and (2) to see how well the approach used in (1) approximates the true variances.

First, some preliminary variable definitions are needed. We will look at income every twelfth month, i.e., in months 1, 13, 25 and so on. We will also drop people who had zero income in month 1 (they make the illustration less interesting).

```
month <- c(1, 13, 25, 49, 61, 73, 85, 97, 109)  # every 12th month
nmonths <- length(month)
nexpected <- 1000  # the desired overall sample size
B <- 1000  # the number of samples that will be selected

# pick out every twelfth column of the income matrix
incjan <- data.frame(incmat[, month]*12)  # income in each of those Januaries
# we multiplied by 12 to convert monthly income into annual ones

keepin <- incjan[, 1] > 0  # to remove people with zero income in month 1
incjan <- incjan[keepin, ] # remove them

HCMj <- nrow(incjan)  # the new population size after dropping zeroes
if (HCMj %% H > 0) {
  HCMj <- HCMj - HCMj %% H  # make it a multiple of H
  incjan <- incjan[1:HCMj, ]  # drop rows accordingly
}
```

We will now stratify the population by sorting it by income in month 1 (this is the part that would not happen in real life because there would be no need to conduct a survey if we already knew everyone's income), and then making the first 10 percent of the population be stratum 1, the next 10 percent be stratum 2, and

so on (we are using the default value $H = 10$). Therefore stratum 1 is the low-income stratum and stratum H is the high income stratum; the new stratum variable is called `stratumj`. Note that making each stratum the same size is not optimal but is good enough for our purpose.

```
incjan <- incjan[order(incjan$X1), ]  # sort by the first month income

# create the stratum variable
strsize <- round(HCMj/H)
HMj <- rep(strsize, H)  # redefine HM; all strata are the same size
stratumj <- rep(1:H, each=strsize)  # assumes HCMj is a multiple of H

incjan$stratumj <- stratumj  # add stratum to the dataframe
ncolX <- ncol(incjan)
```

We are now ready to allocate the total sample size (1000 people by default) to the H strata. We will use Neyman allocation (based on data from month 1).

```
Sh <- as.numeric(by(incjan$X1, incjan$stratumj, sd))  # stratum std deviations
NhSh <- Sh*HMj
nstr <- round(NhSh/sum(NhSh) * nexpected)  # sample size for each stratum
nall <- sum(nstr)  # overall sample size (may be off due to rounding)
```

It is interesting to note the variation in stratum allocation:

```
print(nstr)
```

```
##  [1]  53  34  33  34  36  42  52  68 112 537
```

The high income stratum is allocated much more sample than the other strata. To approximate variances later, we will need the true mean income for each of the months under consideration.

```
meanincs <- apply(incjan[, -ncolX], 2, mean)  # the series we want to estimate
```

And the final step in the preliminaries is to set some variables.

```
stratrangej <- cumrange(rep(strsize, H))  # H x 2 matrix, 1st and last units
vars <- rep(0, nmonths)  # initialize the vector of variances
wstr <- rep(strsize/HCMj, H)  # relative weight (size) of each stratum
```

We are now ready to use method (1), the Monte Carlo approach.

The Monte Carlo approach

Monte Carlo simulation can be used to approximate the variance of the estimator of mean income. Many stratified simple random sample ($B = 1000$ by default) are selected from the population. Each random sample is used to produce a series of estimates (i.e., for months 1, 13, 25, etc.). The squared difference between each estimate and the known true value is accumulated and then used to approximate the true variance.

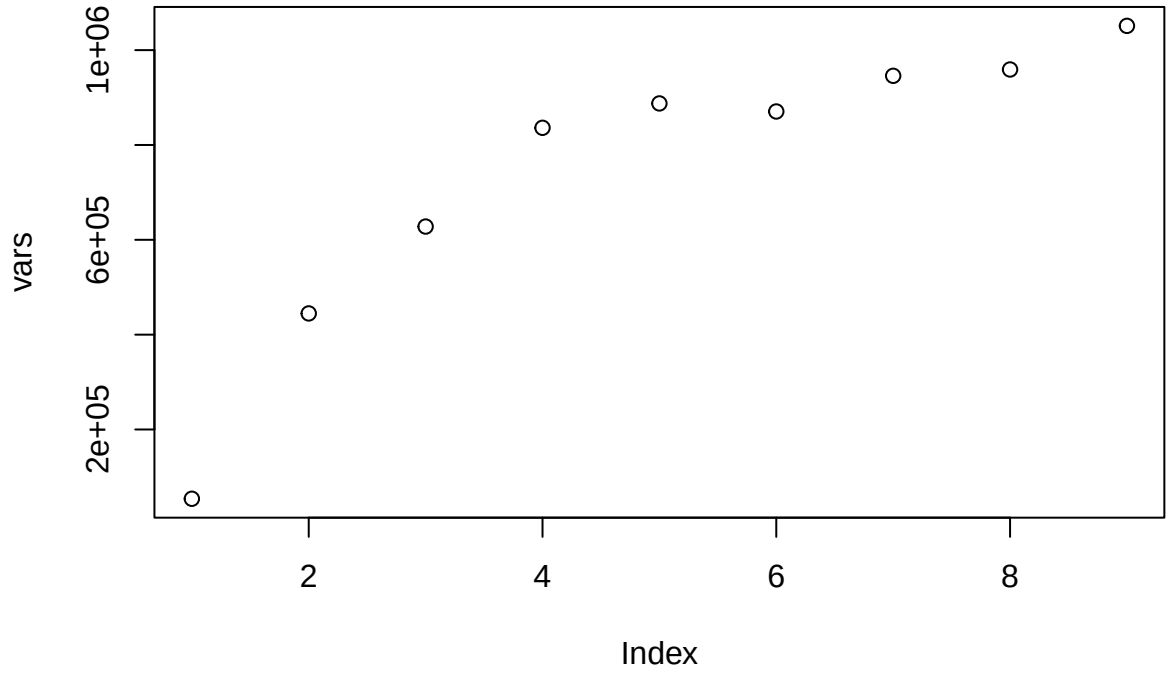
```
for (b in 1:B) {
  sample_h <- numeric()  # initialize vector of sampled units
  # Select a sample (build it one stratum at a time); to keep it simple,
  # the same sample will be used for all months
  for (h in 1:H) {
    sample_h <- c(sample_h,
                  sample(stratrangej[h, 1]:stratrangej[h, 2], nstr[h]))
  }
  Xsample <- incjan[sample_h, ]  # keep just the sampled units
```

```
  # stratum estimates
  stratestimates <-
    simplify2array( by(Xsample[, -ncolX], Xsample[, ncolX], colMeans) )

  # roll up the stratum estimates to the population level (weighted row sums)
  estimates <- apply(stratestimates %*% diag(wstr), 1, sum)

  vars <- vars + (estimates-meanincs)^2  # build up Monte Carlo variances
}

vars <- vars/B  # this is what we want -- the approximate variance
plot(vars)
```



In the graph, we see clearly that the variance increases quickly in the first few years.

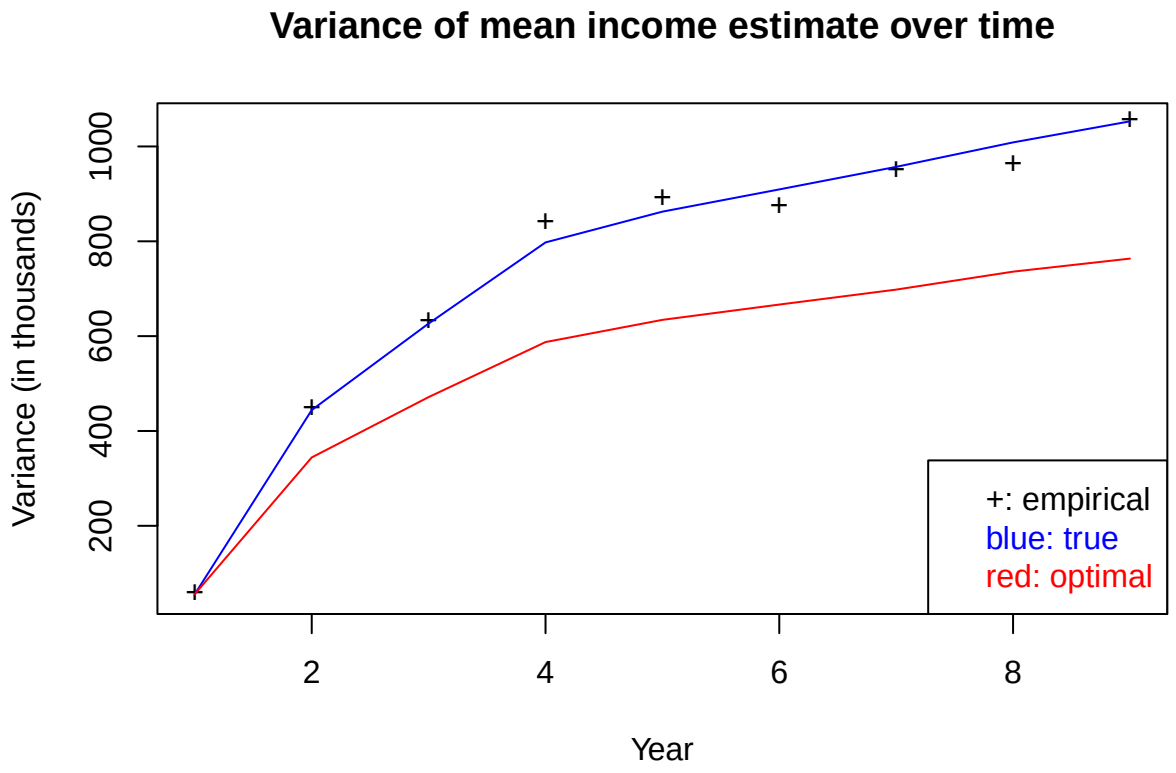
Results from the theory of survey sampling

In our simple scenario, because the whole population is available, we can get exact results by plugging in information from the population into the exact formulas found in any textbook on survey sampling. The following code uses the function `varstratsrs()` which is defined in the `function.R` file included in this package. As a bonus, we will also compute the variances we would get if we *re-stratified the population anew each year*. This can be viewed as a best-case scenario (which, of course, is even more unrealistic in the real world).

```
varstropttruem <- varstrtruem <- numeric(nmonths)  # initialize
```

```r
for (m in 1:nmonths) {
  incm <- incjan[, m]
  Shm <- as.numeric(by(incm, incjan$stratumj, sd))
  NhShm <- Shm*HMj
  nstrm <- round(NhShm/sum(NhShm) * nexpected)  # ideal sample allocation
  varstropttruem[m] <- varstratsrs(HMj, nstrm, Shm)  # ideal case
  varstrtruem[m] <- varstratsrs(HMj, nstr, Shm)      # actual sample allocation
}

plot(vars/1000, pch="+", xlab="Year", ylab="Variance (in thousands)")
title("Variance of mean income estimate over time")
legend("bottomright", legend=c("+: empirical", "blue: true", "red: optimal"),
       text.col=c("black", "blue", "red"))
lines(varstrtruem/1000, col='blue')  # true variances
lines(varstropttruem/1000, col="red")  # best variances using Neyman allocation
```



The red line is the best-case scenario. The blue line and the plus signs are the original scenario, with the blue line showing exact results and the plus signs showing the Monte Carlo approximations. We see that (i) the approximation is quite good (and can be made better by increasing B) and (ii) there would be significant improvement if we could re-stratify the population each time.