

Making use of pre-calibrated weights

March 28, 2025

Public-use data sets often come with weights that have been adjusted by post-stratification, raking, or calibration. It is standard practice to ignore this fact and treat the weights as if they were sampling weights. An alternative approach is to calibrate the weights again in R. This might seem impossible: calibration needs the population totals. But we know the population totals, because the effect of calibration is precisely that the estimated population totals match the true population totals.

From version 4.2, the `svydesign` function has an option `calibrate.formula` to specify (as a model formula) the variables that the weights are already calibrated on. The weights will be recalibrated using this formula, to population totals estimated from the sample. The weights will not change — they were already calibrated — but information will be added to the survey design object to describe the calibration constraints and standard errors of estimates will change.

Consider, for example, the `apiclus1` example. This is a cluster sample of 15 school districts in California

```
> library(survey)
> data(api)
> dclus1 <- svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc)
```

The sampling weights `pw` are already calibrated to sum to the known number of schools in the population, 6194, but R does not know this:

```
> sum(weights(dclus1))
[1] 6194

> dim(apipop)
[1] 6194   37

> dclus1<-update(dclus1, one=rep(1,nrow(dclus1)))
> svytotal(~one,dclus1)

      total      SE
one  6194 1442.9
```

The standard error should be zero, because the calibration procedure ensures that the the estimated total is exactly 6194.

We could use `calibrate()` on the design object

```
> cal_dclus1<-calibrate(dclus1, formula=~1, population=sum(weights(dclus1)))
> svytotal(~one,cal_dclus1)
```

```
      total SE
one  6194  0
```

The standard errors have changed, but the weights haven't

```
> summary(weights(cal_dclus1)/weights(dclus1))
```

```
      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1         1         1         1         1         1
```

With the new option to `svydesign` we can do the same thing when the object is created

```
> precal_dclus1<-svydesign(id = ~dnum, weights = ~pw, data = apiclus1,
+                          fpc = ~fpc, calibrate.formula=~1)
> precal_dclus1<-update(precal_dclus1, one=rep(1,nrow(dclus1)))
> svytotal(~one,precal_dclus1)
```

```
      total SE
one  6194  0
```

Calibrating to the population size simplifies the standard error relationship between mean and total

```
> (enroll_t<-svytotal(~enroll, dclus1))
```

```
      total      SE
enroll 3404940 932235
```

```
> (enroll_m<-svymean(~enroll, dclus1))
```

```
      mean      SE
enroll 549.72 45.191
```

```
> SE(enroll_m)
```

```
      enroll
enroll 45.19137
```

```
> SE(enroll_t)/6194
```

```
      enroll
enroll 150.5061
```

```

> (cenroll_t<-svytotal(~enroll, precal_dclus1))

      total      SE
enroll 3404940 279915

> (cenroll_m<-svymean(~enroll, precal_dclus1))

      mean      SE
enroll 549.72 45.191

> SE(cenroll_m)

      enroll
enroll 45.19137

> SE(cenroll_t)/6194

      enroll
enroll 45.19137

```

Because calibration in this way changes the standard errors but not the point estimates, it's critical that you *only use it when the weights are in fact already calibrated*. If not, the standard errors will be wrong. In particular, it is not valid to take a subset of a data set with calibrated weights and then pretend the subset was also calibrated on the same variables.