Post-stratification and calibration

Thomas Lumley

UW Biostatistics

WNAR—2008–6–22

What are they?

Post-stratification and calibration are ways to use auxiliary information on the population (or the phase-one sample) to improve precision.

They are closely related to the Augmented Inverse-Probability Weighted estimators of Jamie Robins and coworkers, but are easier to understand.

Estimating a total

Population size N, sample size n, sampling probabilities π_i, sampling indicators R_i.

Goal: estimate

$$T = \sum_{i=1}^{N} y_i$$

Horvitz–Thompson estimator:

$$\hat{T} = \sum_{R_i=1} \frac{1}{\pi_i} y_i$$

To estimate parameters θ replace y_i by loglikelihood $\ell_i(\theta)$ or estimating functions $U_i(\theta)$.

Auxiliary information

HT estimator is inefficient when some additional population data are available.

Suppose x_i is known for all i

Fit $y \sim x\beta$ by (probability-weighted) least squares to get $\widehat{\beta}$. Let r^2 be proportion of variation explained.

$$\widehat{T}_{reg} = \sum_{R_i=1} \frac{1}{\pi_i}(y_i - x_i\widehat{\beta}) + \sum_{i=1}^{N} x_i\widehat{\beta}$$

ie, HT estimator for sum of residuals, plus population sum of fitted values

Auxiliary information

Let β^* be true value of β (ie, least-squares fit to whole population).

Regression estimator

$$\hat{T}_{reg} = \sum_{R_i=1} \frac{1}{\pi_i}(y_i - x_i\beta^*) + \left(\sum_{i=1}^{N} x_i\right)\beta^* + \sum_{i=1}^{N}\left(1 - \frac{R_i}{\pi_i}\right)x_i(\hat{\beta} - \beta^*)$$

compare to HT estimator

$$\hat{T} = \sum_{R_i=1} \frac{1}{\pi_i}(y_i - x_i\beta^*) + \left(\sum_{R_i=1} \frac{1}{\pi_i}x_i\right)\beta^*$$

Second term uses known vs observed total of x, third term is estimation error for β, of smaller order.

Auxiliary information

For large n, N and under conditions on moments and sampling schemes

$$\text{var}\left[\hat{T}_{reg}\right] = (1-r^2)\,\text{var}\left[\hat{T}\right] + O(N/\sqrt{n}) = \left(1 - r^2 + O(n^{-1/2})\right)\text{var}\left[\hat{T}\right]$$

and the relative bias is $O(1/n)$

The lack of bias does not require any assumptions about $[Y|X]$

$\hat{\beta}$ is consistent for the population least squares slope β, for which the mean residual is zero by construction.

Reweighting

Since $\hat{\beta}$ is linear in y, we can write $x\hat{\beta}$ as a linear function of y and so $\hat{T}_{reg}$ is also a linear function of Y

$$\hat{T}_{reg} = \sum_{R_i=1} w_i y_i = \sum_{R_i=1} \frac{g_i}{\pi_i} y_i$$

for some (ugly) w_i or g_i that depend only on the xs

For these weights

$$\sum_{i=1}^{N} x_i = \sum_{R_i=1} \frac{g_i}{\pi_i} x_i$$

$\hat{T}_{reg}$ is an IPW estimator using weights that are 'calibrated' or 'tuned' (French: *calage*) so that the known population totals are estimated correctly.

Calibration

The general calibration problem: given a distance function $d(\cdot, \cdot)$, find calibration weights g_i minimizing

$$\sum_{R_i=1} d(g_i, 1)$$

subject to the calibration constraints

$$\sum_{i=1}^{N} x_i = \sum_{R_i=1} \frac{g_i}{\pi_i} x_i$$

Lagrange multiplier argument shows that $g_i = \eta(x_i\beta)$ for some $\eta()$, β; and γ can be computed by iteratively reweighted least squares.

For example, can choose $d(,)$ so that g_i are bounded below (and above).

[Deville *et al* JASA 1993; JNK Rao *et al*, Sankhya 2002]

Calibration

When the calibration model in x is saturated, the choice of $d(,)$ does not matter: calibration equates estimated and known category counts.

In this case calibration is also the same as estimating sampling probabilities with logistic regression, which also equates estimated and known counts.

Calibration to a saturated model gives the same analysis as pretending the sampling was stratified on these categories: post-stratification

Post-stratification is a much older method, and is computationally simpler, but calibration can make more use of auxiliary data.

Standard errors

Standard errors come from the regression formulation

$$\hat{T}_{reg} = \sum_{R_i=1} \frac{1}{\pi_i}(y_i - x_i\hat{\beta}) + \sum_{i=1}^{N} x_i\hat{\beta}$$

The variance of the second term is of smaller order and is ignored.

The variance of the first term is the usual Horvitz–Thompson variance estimator, applied to residuals from projecting y on the calibration variables.

Computing

R provides `calibrate()` for calibration (and `postStratify()` for post-stratification)

Three basic types of calibration

- Linear (or regression) calibration: identical to regression estimator
- Raking: multiplicative model for weights, popular in US, guarantees $g_i > 0$
- Logit calibration: logit link for weights, popular in Europe, provides upper and lower bounds for g_i

Computing

Upper and lower bounds for g_i can also be specified for linear and raking calibration (these may not be achievable, but we try). The user can specify other calibration loss functions (eg Hellinger distance).

Computing

The `calibrate()` function takes three main arguments

- a survey design object
- a model formula describing the design matrix of auxiliary variables
- a vector giving the column sums of this design matrix in the population.

and additional arguments describing the type of calibration.

Computing

```
> data(api)
> dclus1<-svydesign(id=~dnum, weights=~pw, data=apiclus1, fpc=~fpc)
> pop.totals<-c(`(Intercept)`=6194, stypeH=755, stypeM=1018)

> (dclus1g<-calibrate(dclus1, ~stype, pop.totals))
1 - level Cluster Sampling design
With (15) clusters.
calibrate(dclus1, ~stype, pop.totals)
> svymean(~api00, dclus1g)
        mean     SE
api00 642.31 23.921
>  svymean(~api00,dclus1)
        mean     SE
api00 644.17 23.542
```

Computing

```
> svytotal(~enroll, dclus1g)
         total     SE
enroll 3680893 406293
> svytotal(~enroll,dclus1)
         total     SE
enroll 3404940 932235
> svytotal(~stype, dclus1g)
       total         SE
stypeE  4421 1.118e-12
stypeH   755 4.992e-13
stypeM  1018 1.193e-13
```

Computing

```
> (dclus1g3 <- calibrate(dclus1, ~stype+api99,
                         c(pop.totals, api99=3914069)))
1 - level Cluster Sampling design
With (15) clusters.
calibrate(dclus1, ~stype + api99, c(pop.totals, api99 = 3914069))
> svymean(~api00, dclus1g3)
        mean     SE
api00 665.31 3.4418
> svytotal(~enroll, dclus1g3)
         total     SE
enroll 3638487 385524
> svytotal(~stype, dclus1g3)
       total        SE
stypeE  4421 1.179e-12
stypeH   755 4.504e-13
stypeM  1018 9.998e-14
```

Computing

```
> range(weights(dclus1g3)/weights(dclus1))
[1] 0.4185925 1.8332949

> (dclus1g3b <- calibrate(dclus1, ~stype+api99,
          c(pop.totals, api99=3914069),bounds=c(0.6,1.6)))
1 - level Cluster Sampling design
With (15) clusters.
calibrate(dclus1, ~stype + api99, c(pop.totals, api99 = 3914069),
    bounds = c(0.6, 1.6))

> range(weights(dclus1g3b)/weights(dclus1))
[1] 0.6 1.6
```

Computing

```
> svymean(~api00, dclus1g3b)
        mean     SE
api00 665.48 3.4184
> svytotal(~enroll, dclus1g3b)
         total     SE
enroll 3662213 378691
> svytotal(~stype, dclus1g3b)
       total         SE
stypeE  4421 1.346e-12
stypeH   755 4.139e-13
stypeM  1018 8.238e-14
```

Computing

```
> (dclus1g3c <- calibrate(dclus1, ~stype+api99, c(pop.totals,
+     api99=3914069), calfun="raking"))
1 - level Cluster Sampling design
With (15) clusters.
calibrate(dclus1, ~stype + api99, c(pop.totals, api99 = 3914069),
    calfun = "raking")
> range(weights(dclus1g3c)/weights(dclus1))
[1] 0.5342314 1.9947612
> svymean(~api00, dclus1g3c)
        mean     SE
api00 665.39 3.4378
```

Computing

```
> (dclus1g3d <- calibrate(dclus1, ~stype+api99, c(pop.totals,
+     api99=3914069), calfun="logit",bounds=c(0.5,2.5)))
1 - level Cluster Sampling design
With (15) clusters.
calibrate(dclus1, ~stype + api99, c(pop.totals, api99 = 3914069),
    calfun = "logit", bounds = c(0.5, 2.5))
> range(weights(dclus1g3d)/weights(dclus1))
[1] 0.5943692 1.9358791
> svymean(~api00, dclus1g3d)
        mean     SE
api00 665.43 3.4325
```

Types of calibration

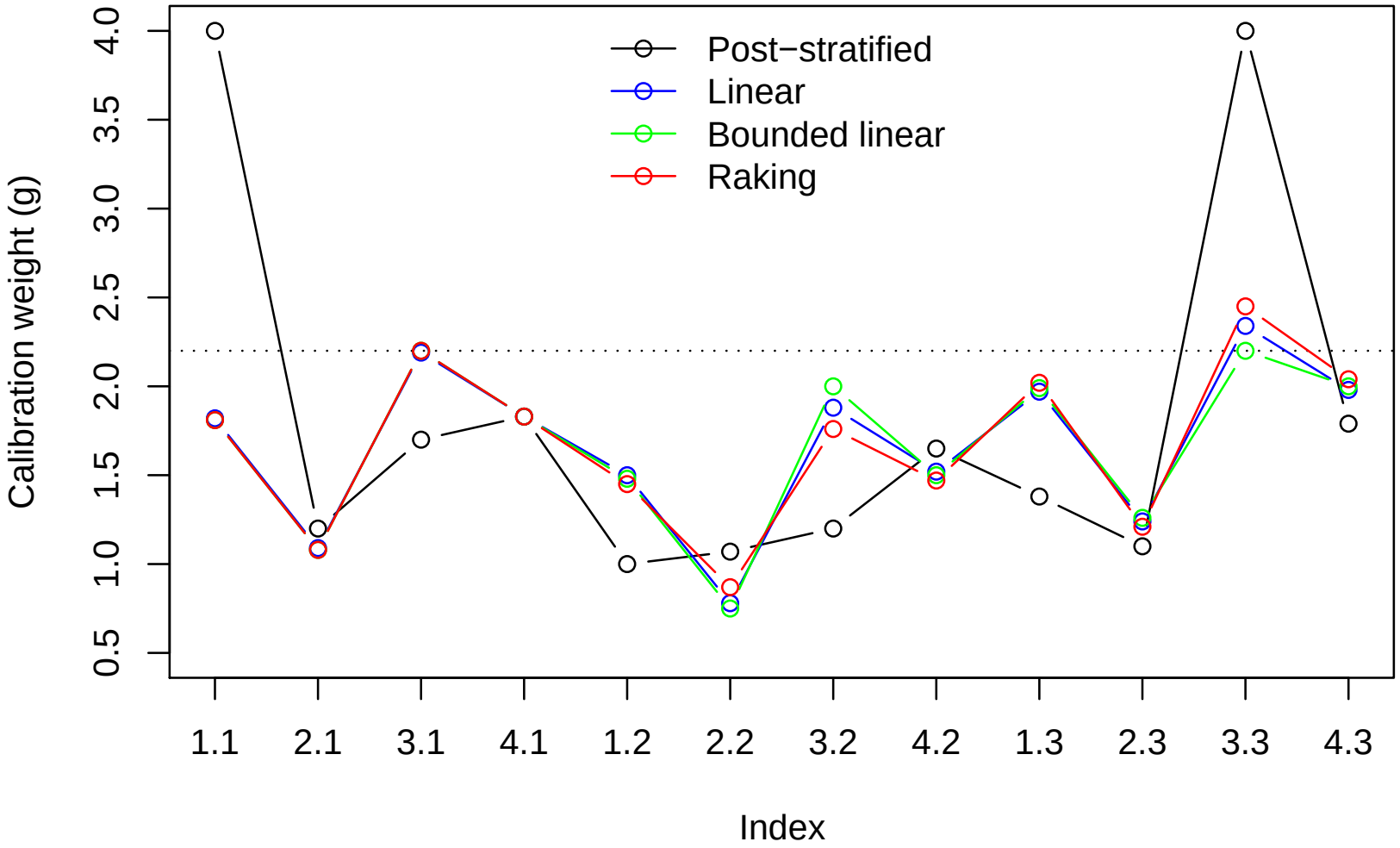
Post-stratification allows much more flexibility in weights, in small samples can result in very influential points, loss of efficiency.

Calibration allows for less flexibility (cf stratification vs regression for confounding)

Different calibration methods make less difference

Example from Kalton & Flores-Cervantes (J. Off. Stat, 2003): a 3×4 table of values.

Types of calibration



Two-phase studies

Sample a cohort of N people from population and measure some variables then subsample n of them and measure more variables [genes, biomarkers, coding of open-text data, copies of original medical records]

Includes nested case–control, case–cohort designs.

Better use of auxiliary information by either stratifying the sampling or calibrating to full cohort data after sampling.

Calibration of second phase is just like calibration of a single-phase design.

RRZ estimators

Robins, Rotnitzky & Zhao defined augmented IPW estimators for two-phase designs

$$\sum_{i=1}^{N} \frac{R_i}{\pi_i} U_i(\theta) + \sum_{i=1}^{N} \left(1 - \frac{R_i}{\pi_i}\right) A_i(\theta) = 0$$

where $A_i()$ can be any function of phase-1 data. Equivalent to calibration estimator $\widehat{T}_{reg}$ using A_i as calibration variable.

$$\sum_{i=1}^{N} \frac{R_i}{\pi_i} \left(U_i(\theta) - A_i(\theta)\right) + \sum_{i=1}^{N} A_i(\theta) = 0$$

RRZ estimators

Includes the efficient estimator in the non-parametric phase-1 model (efficient design-based estimator) — the most efficient estimator that is consistent for the same limit as if we had complete data.

Typically not fully efficient if outcome-model assumptions are imposed at phase 1.

Example: Cox model assumes infinitely many constraints at phase 1, and efficient two-phase estimator is known (Nan 2004, Can J Stat) and is more efficient than calibration estimator.

Estimated weights

RRZ also note that estimating π from phase-1 data gives better precision than using true known π . Widely regarded as a paradox.

Estimated weights (eg logistic regression) solve

$$\sum_{i=1}^{N} x_i R_i = \sum_{i=1}^{N} x_i p_i$$

ie, equate observed and estimated population moments. For discrete x this is exactly calibration, for continuous x it is effectively equivalent.

Estimated weights

Gain of precision in calibration is not paradoxical: comes from replaceing variance of Y with variance of residuals for a reduction by $(1 - r^2)$– nothing to do with 'estimation'

Exactly same issue as gain of precision when adjusting randomized trial for baseline: can write randomized trial estimator as calibration with counterfactuals.

Estimation error in weights does increase uncertainty, but this is second order: for p predictors it is $O(1 + p/n)$

Calibration provides increased precision only when r^2 is large enough (compared to p/n).

[Judkins et al, Stat Med 26:1022-33]

Computing

`calibrate()` also works on two-phase design objects

Since the phase-one data are already stored in the object, there is no need to specify population totals when calibrating.

It is necessary to specify `phase=2`.

This morning we had a two-phase case–control design

```
dccs2<-twophase(id=list(~seqno,~seqno),
   strata=list(NULL,~interaction(rel,instit)),
   data=nwtco, subset=~incc2)
```

Calibrating it to 16 strata of relapse×stage×institutional histology:

```
gccs8<-calibrate(dccs2, phase=2,
       formula=~interaction(rel,stage,instit))
```

Logistic regression

As all the phase-one data are available we can also estimate sampling weights by logistic regression, as suggested by Robins,Rotnitzky & Zhao (JASA, 1994).

Either use `calibrate` with `calfun="rrz"` or `estWeights`.

`estWeights` takes a data frame with missing values as input and produces a corresponding two-phase design with weights estimated by logistic regression.

Choice of auxiliaries

The other heuristic gain from the calibration viewpoint is in choosing predictors for estimating π.

The regression formulation shows that the predictors should have strong linear relationships with $U_i(\theta)$.

If $U_i(\theta)$ is of a form such as

$$z_i w_i (y_i - \mu_i(\theta))$$

then z_i is approximately uncorrelated with U_i

So, don't use a variable correlated with a phase-2 predictor as a calibration variable, use a variable correlated with the phase-2 score function.

`estWeights()` can take a phase-one model as an argument and use the estimating functions from that model as calibration variables.

More detail from Norm.