Two-phase designs and calibration

Thomas Lumley

UW Biostatistics

Copenhagen — 2008–4–3

Two-phase sampling

- Phase 1: sample people according to some probability design $\pi_{1,i}$, measure variables

- Phase 2: subsample people from the phase 1 sample using the variables measured at phase 1, $\pi_{2|1,i}$ and measure more variables

Sampling probability $\pi_i = \pi_{1,i} \times \pi_{2|1,i}$ and use $1/\pi_i$ as sampling weights.

These are **not** (in general) the marginal probability that i is in the sample, because $\pi_{2|1,i}$ may depend on which other observations are in the phase-one sample.

Two-phase sampling

For all the designs today $\pi_{1,i}$ is constant, so its value doesn't affect anything except estimated population totals.

We don't care about population totals in this setting, so we don't need to know the value of $\pi_{1,i}$ (eg can set to 1).

An alternative view is that we are using model-based inference at phase one, rather than sampling-based inference.

Two-phase sampling

Two sources of uncertainty:

- Phase-1 sample is only part of population
- Phase 2 observes full set of variables on only a subset of people.

Uncertainties at the two phases add.

Minimal phase 1

The classic survey example is 'two-phase sampling for stratification'. This is useful when a good stratifying variable is not available for the population but is easy to measure.

- Take a large simple random sample or cluster sample and measure stratum variables
- Take a stratified random sample from phase 1 for the survey

If the gain from stratification is larger than the cost of phase 1 we have won.

The case–control design is probably the most important example, although it isn't analyzed using sampling weights.

Minimal phase 2

Many large cohorts exist in epidemiology. These can be modelled as simple random samples. They have a lot of variables measured.

It is common to want to measure a new variable

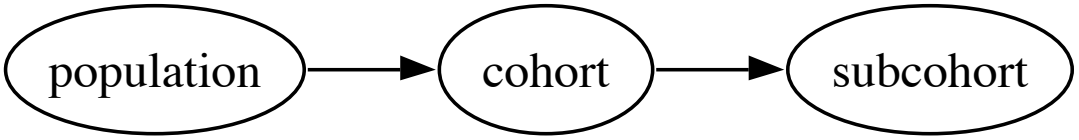
- New assay on stored blood
- Coding of open-text questionnaire
- Re-interview

The classic designs are a simple random sample and a case–control sample.

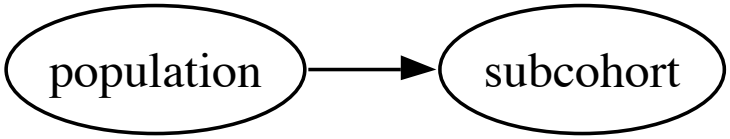
It is often more useful to sample based on multiple phase-1 variables: outcome, confounders, surrogates for phase-2 variable.

Analysis

The true sampling is two-phase



We can ignore the first phase and pretend that we had an unstratified population sample



This is conservative, but sometimes not very. It was used before software for eg Cox models in two-phase samples were developed.

Two-phase case–control

The case–control design stratifies on Y. We can stratify on X as well

	X=0	X=1	
Y=0	a	b	m_0
Y=1	c	d	m_1
	n_0	n_1	

The estimated variance of β is

$$\text{var}[\widehat{\beta}] = \frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d}$$

Ideally want all cells about the same size in this trivial case.

Example: Wilm's Tumor

- Wilm's Tumor is a rare childhood cancer of the kidney. Prognosis is good in early stage or favourable histology disease.
- Histology is difficult to determine. NWTSG central pathologist is much better than anyone else.
- To reduce cost of followup, consider central histology only for a subset of cases.
- Sample all relapses, all patients with unfavorable histology by local pathologist, 10% of remainder

[Breslow & Chatterjee (1999) Applied Statistics 457–68]

Analyses: sampling weights

Phase 1

	relapse	
instit	0	1
good	3207	415
bad	250	156

Phase 2

	relapse	
instit	0	1
good	314	415
bad	250	156

Weight is 1/sampling fraction in relapse $\times$ local histology strata: 1.0 for cases, controls with unfavorable local histology, $3207/314 = 10.2$ for controls with favorable local histology.

Analyses: sampling weights

Effect of unfavorable histology

	log OR	SE
Full data	1.81	0.11
Subcohort		
(ignoring sampling)	-0.02	0.12
two-phase	1.78	0.155
single-phase	1.78	0.159

Can use any analysis, eg RR rather than OR

	log RR	SE
Full data	1.39	0.07
Subcohort		
(ignoring sampling)	-0.01	0.06
two-phase	1.36	0.101
single-phase	1.36	0.107

Computation in R

Declaring a two-phase design

```
dccs2 <- twophase(id = list(~id, ~id), subset = ~in.ccs,
                  strata = list(NULL, ~interaction(instit, rel)),
                  data = nwt.exp)
```

- Data set has records for all phase-one people, `subset` variable indicates membership in second phase
- Two `id`, two `strata`.
- Second-phase `weights` and `fpc` are worked out by R

Computation in R

We can compare to the conservative approximation: single-phase sampling with replacement

```
dcons<-svydesign(id=~seqno, weights=weights(dccs2),
       data=subset(nwtco, incc2))
```

Almost no advantage of two-phase analysis in this example; but wait until this afternoon and calibration of weights.

Computation in R

```
>  summary(svyglm(rel~factor(stage)*factor(histol),design=dccs2,
                          family=quasibinomial()))
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                    -2.6946     0.1246 -21.623  < 2e-16 ***
factor(stage)2                  0.7733     0.1982   3.901 0.000102 ***
factor(stage)3                  0.7400     0.2048   3.614 0.000315 ***
factor(stage)4                  1.1322     0.2572   4.401 1.18e-05 ***
factor(histol)2                 1.1651     0.3196   3.646 0.000279 ***
factor(stage)2:factor(histol)2  0.3642     0.4462   0.816 0.414511
factor(stage)3:factor(histol)2  1.0230     0.3968   2.578 0.010056 *
factor(stage)4:factor(histol)2  1.7444     0.4973   3.508 0.000470 ***

>  summary(svyglm(rel~factor(stage)*factor(histol),design=dcons,
                          family=quasibinomial()))
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                    -2.6946     0.1355 -19.886  < 2e-16 ***
factor(stage)2                  0.7733     0.1982   3.902 0.000101 ***
factor(stage)3                  0.7400     0.2047   3.615 0.000314 ***
factor(stage)4                  1.1322     0.2571   4.403 1.17e-05 ***
factor(histol)2                 1.1651     0.3208   3.632 0.000294 ***
factor(stage)2:factor(histol)2  0.3642     0.4461   0.816 0.414408
factor(stage)3:factor(histol)2  1.0230     0.3974   2.574 0.010169 *
factor(stage)4:factor(histol)2  1.7444     0.4977   3.505 0.000474 ***
```

Computation in R

We are not restricted to logistic regression: eg, log link gives log relative risks rather than log odds ratios. SInce relapse is not rare for unfavorable histology these are noticeably different.

```
>  summary(svyglm(rel~factor(stage)*factor(histol),design=dccs2,
                  family=quasibinomial(log)))
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                    -2.7600     0.1167 -23.644  < 2e-16 ***
factor(stage)2                  0.7020     0.1790   3.922 9.30e-05 ***
factor(stage)3                  0.6730     0.1849   3.640 0.000285 ***
factor(stage)4                  1.0072     0.2207   4.564 5.58e-06 ***
factor(histol)2                 1.0344     0.2690   3.845 0.000127 ***
factor(stage)2:factor(histol)2  0.1153     0.3405   0.339 0.734920
factor(stage)3:factor(histol)2  0.4695     0.3118   1.505 0.132495
factor(stage)4:factor(histol)2  0.4872     0.3316   1.469 0.142020
```

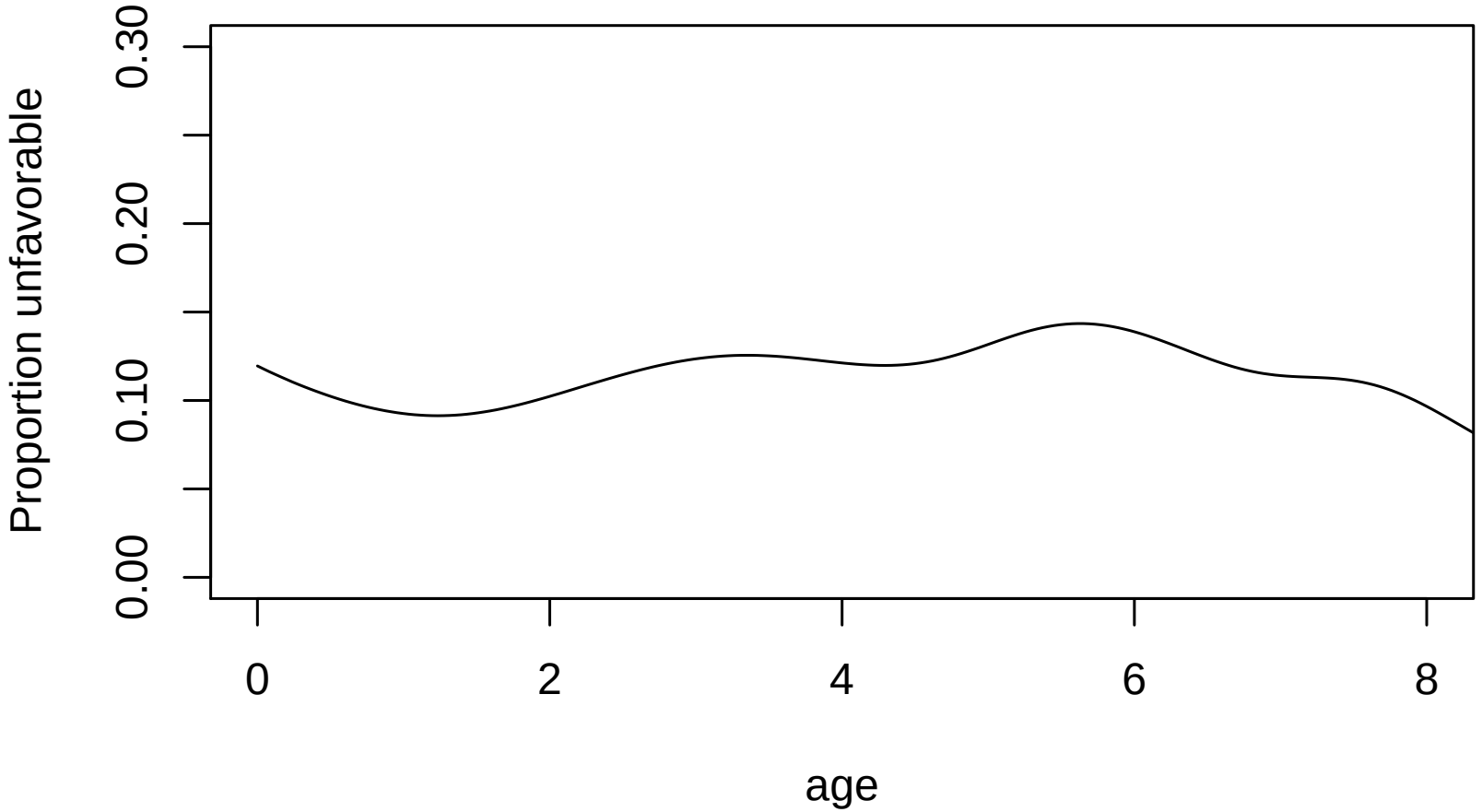
Computation in R

Other exploratory and descriptive analyses are also possible: how does histology vary with age at diagnosis?

[Really needs more detailed classification to be useful analysis]

```
s<-svysmooth(I(histol==2)~I(age/12),design=dccs2,bandwidth=1)
plot(s,ylim=c(0,0.3),xlim=c(0,8),
       xlab="age",ylab="Proportion unfavorable")
```

Computation in R



Case–cohort design

Cox model estimation involves solving

$$\sum_{\text{events}} Z_{\text{event}} - E(\beta, t) = 0$$

where Z_{event} is the covariate for the person having an event and $E(\beta, t)$ is the expected covariate based on a weighted average over everyone at risk at that time.

Case–cohort design

- Prentice (1986) suggested estimating $E(\beta, t)$ from the current case plus a subcohort selected at random at baseline, saving money and computation time.
- Self & Prentice suggested dropping current case from $E(\beta, t)$ for mathematical simplicity.
- Barlow used the subcohort and all cases, with the conservative single-phase sampling standard errors
- Lin & Ying used the same estimates as Barlow, but the correct standard errors.
- Borgan et al (2000) allowed stratified sampling and time-dependent sampling weights.

Advantages

Case–cohort and nested case–control studies have similar power, but in a case–cohort design

- The same subcohort can be used for more than one outcome and for different lengths of followup
- Subcohort defined at baseline allows exposure measurement spread over whole study (for some measurements)
- Information from whole cohort can easily be incorporated when selecting subcohort (eg surrogate exposure) or with post-stratification at time of analysis

Design has been used in ARIC, CHS for measuring genotypes or biomarkers.

Historical artifact

Mathematical techniques in 1980s,1990s, required viewing the Cox model estimating equations

$$\sum_{\text{events}} Z_{\text{event}} - E(\beta) = 0$$

as a process evolving over time.

A member of the subcohort who was in the future going to be a case still had to be treated the same as someone who was not going to become a case, because that information was not predictable. Weight might be 10 until the event and then 1 at the event.

Survey techniques allow a simpler view *sub specie aeternitatis*; no hang-ups about predictable weight processes: weight of 1 for any future case.

Computation

- Prentice, Self & Prentice methods are not just weighted Cox regression, so need some trickery. Mayo Clinic Biostatistics Technical Report #62 describes the trickery and gives S-PLUS and SAS code fragments. `cch()` in R `survival` package is based on these methods.

- Barlow's method, using the conservative single-phase approximateion, is just weighted Cox regression and works in any software with survey or robust standard errors.

- Lin & Ying's method is two-phase weighted Cox regression, needs software for two-phase analysis.

Computation in R

Estimators based on survey-weighted Cox models (Lin & Ying)

```
dcchs<-twophase(id=list(~seqno,~seqno), strata=list(NULL,~rel),
                subset=~I(in.subcohort | rel), data=nwtco)
svycoxph(Surv(edrel,rel)~factor(stage)+factor(histol)+I(age/12),
                design=dcchs)
```

Traditional estimators based on the Therneau & Li trick.

```
fit.ccSP <- cch(Surv(edrel, rel) ~ stage + histol + age,
                 data =ccoh.data, subcoh = ~subcohort, id=~seqno,
                 cohort.size=4028,  method="SelfPren")

stratsizes<-table(nwtco$instit)
fit.BI<- cch(Surv(edrel, rel) ~ stage + histol + age,
              data =ccoh.data, subcoh = ~subcohort,
              id=~seqno, stratum=~instit, cohort.size=stratsizes,
              method="I.Borgan")
```

Post-stratification and calibration

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

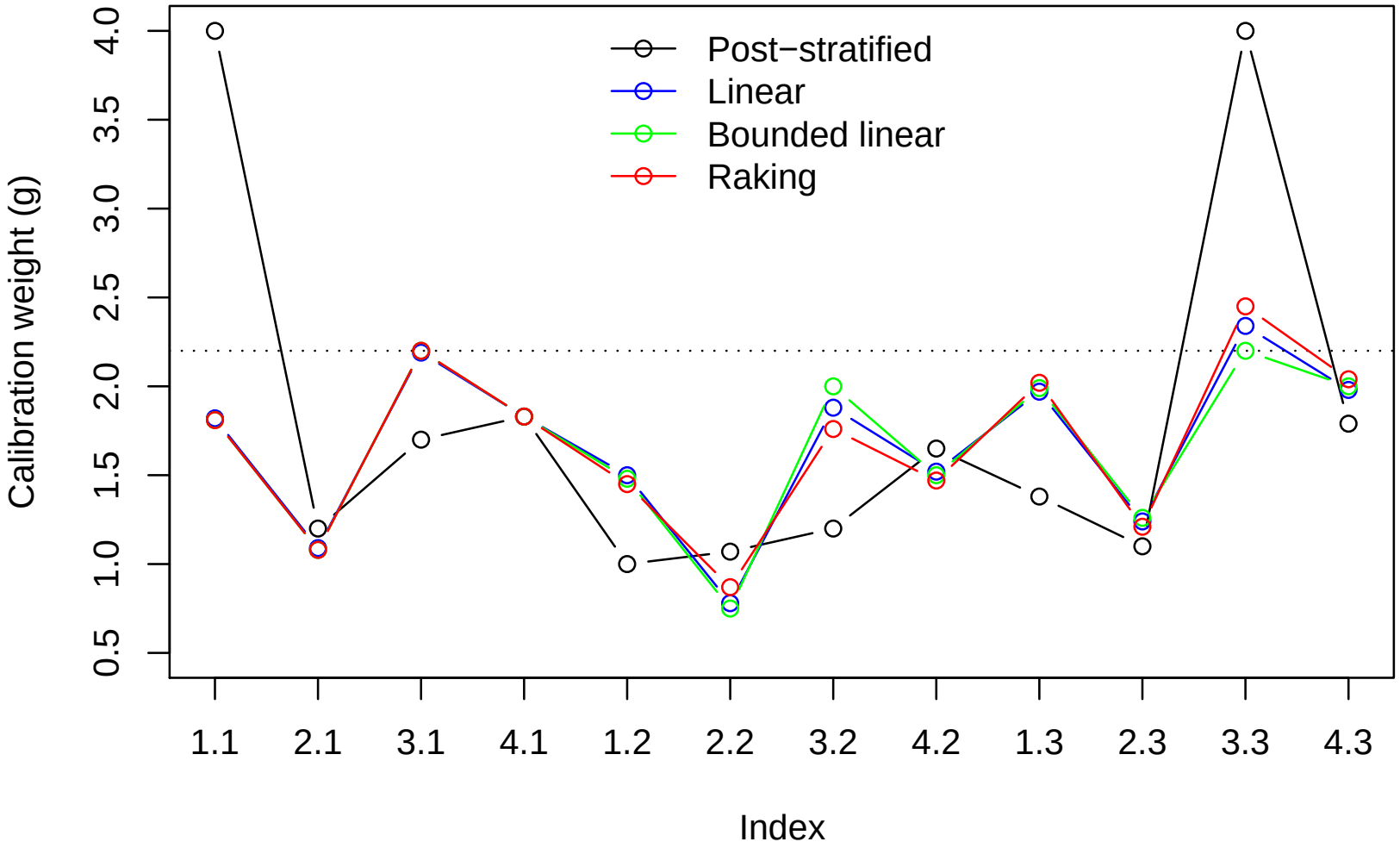
Post-stratification allows much more flexibility in weights, in small samples can result in very influential points, loss of efficiency.

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 the estimating function $U_i(\theta)$ 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 influence function.

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

Example

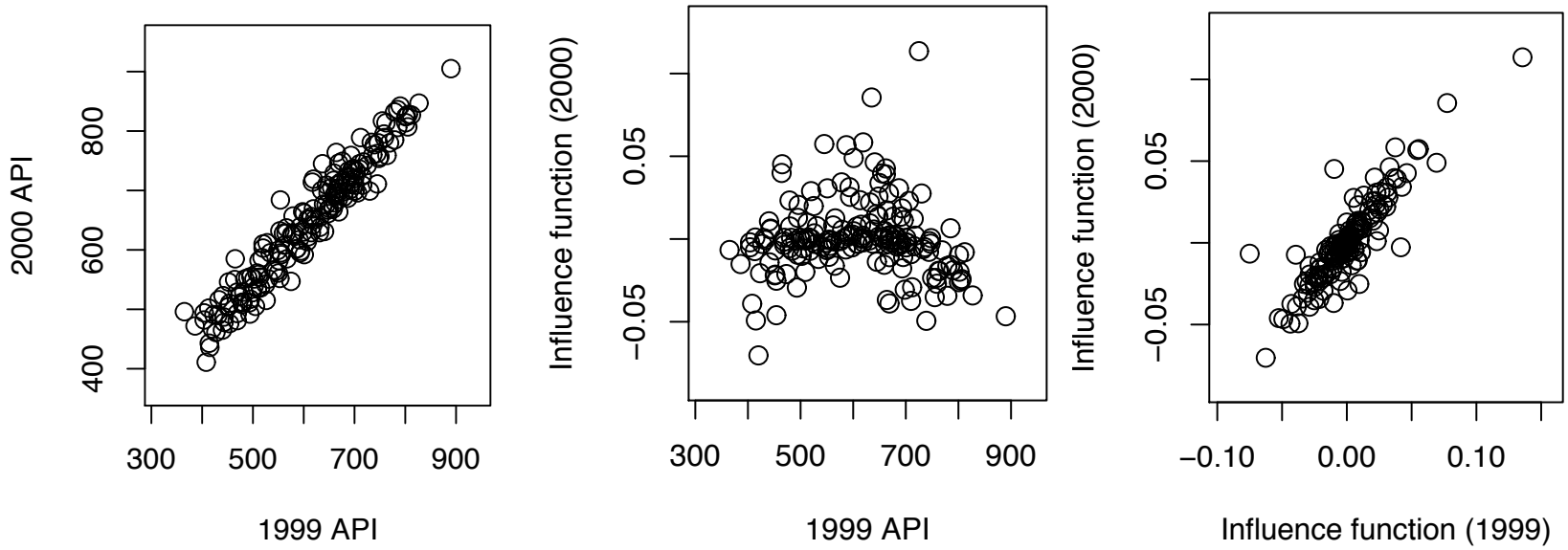
Single-phase sampling of California schools.

We want to fit a model with 2000 API as the outcome, with 2000 API measured only on a sample of schools.

We have population data on 1999 API and on the predictor variables. Calibrating using the raw variables has almost no effect.

Impute 2000 API by 1999 API and fit the model to complete imputed data. The parameter estimates in the imputed model are unrealiable, but the influence functions still are good calibration variables.

Example: correlations



Example: code

```
> m0 <- svyglm(api00~ell+mobility+emer, clus1)
> var_cal <- calibrate(clus1, formula=~api99+ell+mobility+emer,
      pop=c(6194,3914069, 141685, 106054, 70366),
      bounds=c(0.1,10))
> m1<-svyglm(api00~ell+mobility+emer, design=var_cal)
>
> popmodel <- glm(api99~ell+mobility+emer, data=apipop,
     na.action=na.exclude)
> inffun <- dfbeta(popmodel)
> index <- match(apiclus1$snum, apipop$snum)
> clus1if <- update(clus1, ifint = inffun[index,1],
     ifell=inffun[index,2], ifmobility=inffun[index,3],
     ifemer=inffun[index,4])
> if_cal <- calibrate(clus1if,
    formula=~ifint+ifell+ifmobility+ifemer,
    pop=c(6194,0,0,0,0))
```

Example: code

```
> m2<-svyglm(api00~ell+mobility+emer, design=if_cal)
>
> coef(summary(m0))
              Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 780.459500 30.0210123 25.997108 3.156974e-11
ell          -3.297892  0.4689026 -7.033215 2.173478e-05
mobility     -1.445370  0.7342887 -1.968395 7.473627e-02
emer         -1.814215  0.4233504 -4.285374 1.287085e-03
> coef(summary(m1))
              Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 785.408240 13.7640081 57.062466 5.912274e-15
ell          -3.273108  0.6242978 -5.242864 2.756024e-04
mobility     -1.464732  0.6651257 -2.202188 4.989506e-02
emer         -1.676541  0.3742041 -4.480284 9.309647e-04
```

Example: code

```
> coef(summary(m2))
             Estimate Std. Error    t value     Pr(>|t|)
(Intercept) 790.631553  5.8409844 135.359298 4.480786e-19
ell          -3.260976  0.1300765 -25.069679 4.678967e-11
mobility     -1.405554  0.2247022  -6.255187 6.214930e-05
emer         -2.240431  0.2150534 -10.418024 4.902863e-07
```

Two-phase calibration

Strategy

- Build a predictive model for the phase-two variables to impute at phase one
- Fit a phase-one model for the parameter of interested, using the imputed data
- Use the influence functions from the phase-one model to calibrate
- Fit the model to the phase-two sample

Wilms' Tumor

The imputation model is based on Kulich & Lin (2004).

```
impmodel <- glm(histol~instit+I(age>10)+I(stage==4)*study,
  data=nwts, subset=in.subsample, family=binomial)
nwts$imphist <- predict(impmodel, newdata=nwts, type="response")
nwts$imphist[nwts$in.subsample] <- nwts$histol[nwts$in.subsample]

ifmodel <- coxph(Surv(trel,relaps)~imphist*age+I(stage>2)*tumdiam,
  data=nwts)
inffun <- resid(ifmodel, "dfbeta")
colnames(inffun) <- paste("if",1:6,sep="")

nwts_if <- cbind(nwts, inffun)
if_design <- twophase(id = list(~1, ~1), subset = ~in.subsample,
  strata = list(NULL, ~interaction(instit, relaps)),
  data = nwts_if)
```

Wilms' Tumor

```
if_cal <- calibrate(if_design, phase=2, calfun="raking"
  ~if1+if2+if3+if4+if5+if6+relaps*instit)

m1 <- svycoxph(Surv(trel, relaps)~histol*age+I(stage>2)*tumdiam,
  design=nwts_design)
m2 <- svycoxph(Surv(trel, relaps)~histol*age+I(stage>2)*tumdiam,
  design=if_cal)
m3 <- coxph(Surv(trel, relaps)~imphist*age+I(stage>2)*tumdiam,
  data=nwts)
m4 <- coxph(Surv(trel, relaps)~histol*age+I(stage>2)*tumdiam,
  data=nwts)
```

Wilms' Tumor

	Two-phase sample			full data
	sampling weights	raked	direct imputation	
Coefficient estimate				
histology	1.808	2.113	2.108	1.932
age	0.055	0.101	0.101	0.096
stage > 2	1.411	1.435	1.432	1.389
tumor diameter	0.043	0.061	0.061	0.058
histology:age	-0.116	-0.159	-0.159	-0.144
stage> 2:diameter	-0.074	-0.084	-0.083	-0.079
Standard error				
histology	0.221	0.171	0.174	0.157
age	0.023	0.014	0.016	0.016
stage > 2	0.361	0.276	0.249	0.250
tumor diameter	0.021	0.016	0.014	0.014
histology:age	0.054	0.039	0.040	0.035
stage > 2:diameter	0.030	0.022	0.020	0.020

Summary

Calibration reweights the sample to make it more representative of the population,

It is not paradoxical that this reweighting adds information.

Calibration relies on linear correlation, so calibration of parameter estimates should target the influence functions.

We don't know the optimal strategies yet.