Sample surveys

Thomas Lumley

tlumley@u

2007–11–2

Probability sample

A sample taken from a well-defined population according to a well-defined random sampling scheme gives rise to an exactly known probability model: only the sampling is random and we control it.

Imperfections in the population definition or the sampling lead to 'non-sampling error'. This is hard to estimate and does not decrese with increasing sample size.

In the absence of a well-defined design or in the presence of substantial non-response the data are not much use except for qualitative summaries. It is much better to have a probability sample of 500 people than a convenience sample of 1000.

Analogy

Non-statisticians don't understand the power of sampling and prefer enumeration.

An apparently successful analogy is a blood sample, where a few cc is enough to represent the entire circulatory system.

Non-sampling error

- Plan for follow-ups (on a subsample if necessary)
- Pilot the questionnaire
- Don't ask unnecessary questions
- Use standard questions where possible
- Explain the purpose of the survey, if possible
- Bribe people in advance.

These all go against the initial instincts of researchers. They mostly result in smaller sample sizes and worse-looking power calculations.

Complex samples

A simple random sample can be analysed using all the standard statistical tools.

Simple random samples are not used very much

- Strata: if the population list is divided into groups, you can fix the number taken from each group to reduce the variance
- Clusters: it is often much cheaper to sample groups of nearby individuals
- Unequal probabilities: oversampling a interesting subgroup gives more precision at the same cost.

Analysis

Basic idea: each person represents a group in the population. Use the size of that group as the weight.

Sampling weights are just 1/sampling probability.

Standard errors come from sandwich estimator ('linearization') or from jackknife/bootstrap techniques ('replicate weights').

Courses

BIOST/CSSS/STAT 529 (Winter 2008) covers survey analysis in detail.

CSSS/HSERV 527 (Spring) is on practical design and implementation of a questionnaire.

Software

In the old days, survey analysis used special software (SUDAAN, VPLX, WesVar)

Most statistical software now has at least basic survey analysis. Stata and R are very comprehensive.

Important to check on meaning of **weights**

- Precision weights are proportional to 1/variance (eg classical WLS)
- Frequency weights are used to compress discrete data (number of observations like this one)
- Sampling weights are proportional to 1/variance

Confusing sampling weights with the other kinds will give reasonable point estimates but **seriously** bogus standard errors and p-values.

R survey package

Implements most of the useful survey analysis features, with reasonably familiar interface.

On modern PCs, can handle national-scale surveys of people (tens of thousands of observations)

64-bit computers needed for very large surveys (businesses, medical visits, internet traffic).

Specifying survey information

```
des <- svydesign(id=~PSU, strata=~strata,
    fpc=~Pop.size, weight=~weights,
    data=dataset)
```

Creates an object containing the survey design information. Features such as strata or finite population correction are optional.

Sampling weights can be specified as weights or probabilities, and for multi-stage sampling can also be specified as probabilities at each stage

Finite population correction can be specified as proportion or population size, per observation or per stratum (currently restricted to first stage of sampling).

Specifying survey information

```
rdes <- svrepdesign(dataset, weights=~weights,
    repweights=repweights, type="BRR")

adss<-svrepdesign(data = adssdata,
    repweights = adssdata[, 782:981],
    scale = 1, rscales = adssjack, type = "other",
    weights = ~PH1FW0, combined.weights = TRUE,
    fpc=adssfpc, fpctype="correction")

rdes<-as.svrepdesign(des)
```

Specifying survey information

The survey design object contains the observed data and the meta-data describing the sampling. This object, rather than a data frame, is supplied to survey analysis functions.

Subsetting preserves the meta-data

```
first100.schools <- cal.schools[1:100,]
low.ses <- subset(cal.schools, ell>25 | meals >50)
```

and an `update` function allows adding variables

```
cal.schools <- update(cal.schools,
                      low.ses= ell>25 | meals >50,
                      APIchange = api00-api99)
```

Specifying survey information

Printing the object gives some summary information; more is available with the `summary` function:

```
> dstrat
Stratified Independent Sampling design
svydesign(id = ~1, strata = ~stype, weights = ~pw, data = apistrat,
    fpc = ~fpc)

> dclus2
2 - level Cluster Sampling design
With ( 40, 126 ) clusters.
svydesign(id = ~dnum + snum, weights = ~pw, data = apiclus2)

> dclus1
1 - level Cluster Sampling design
With ( 15 ) clusters.
svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc)
```

Specifying survey information

```
> summary(subset(dstrat, stype != "H"))
Stratified Independent Sampling design
subset.survey.design(dstrat, stype != "H")
Probabilities:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.02262 0.02262 0.02262 0.03145 0.04912 0.04912
Stratum sizes:
             E  H  M
obs        100  0 50
design.PSU 100 50 50
actual.PSU 100  0 50
Population stratum sizes (PSUs):
  strata    N
1      E 4421
2      H  755
3      M 1018
Data variables:
 [1] "cds"      "stype"    "name"     "sname"    "snum"     "dname"    "dnum"
 [8] "cname"    "cnum"     "flag"     "pcttest"  "api00"    "api99"    "target"
...
```

Specifying survey information

Post-stratification, raking, and calibration adjust the weights so that estimated totals for some variables match known population totals.

```
cal.ps<-postStratify(cal.schools,
   strata=~stype, population=pop.table)
```

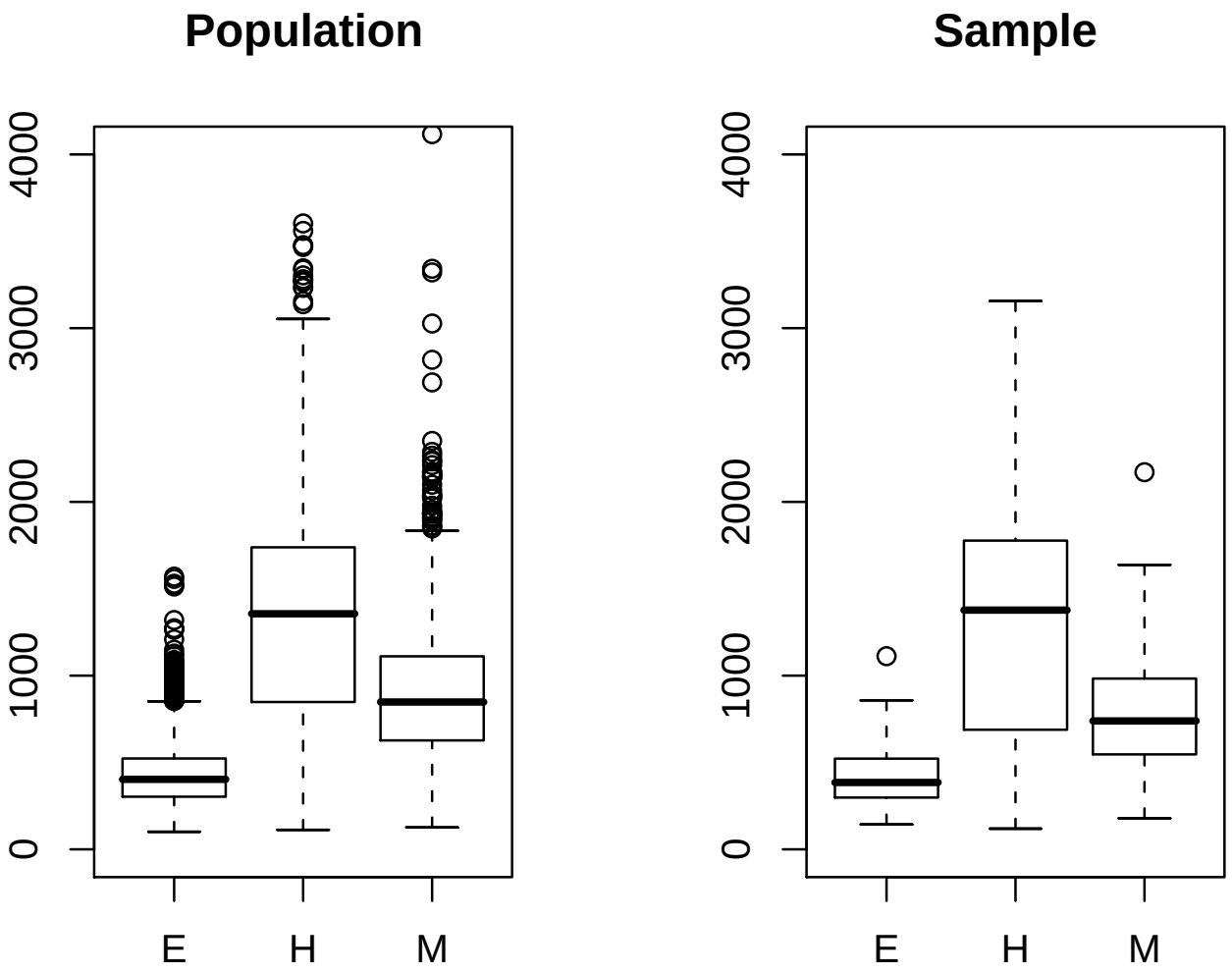
The `rake()` function is similar but takes a list of formulas and a list of population tables, `calibrate()` takes a model formula and the column sums for the corresponding population design matrix.

Graphics

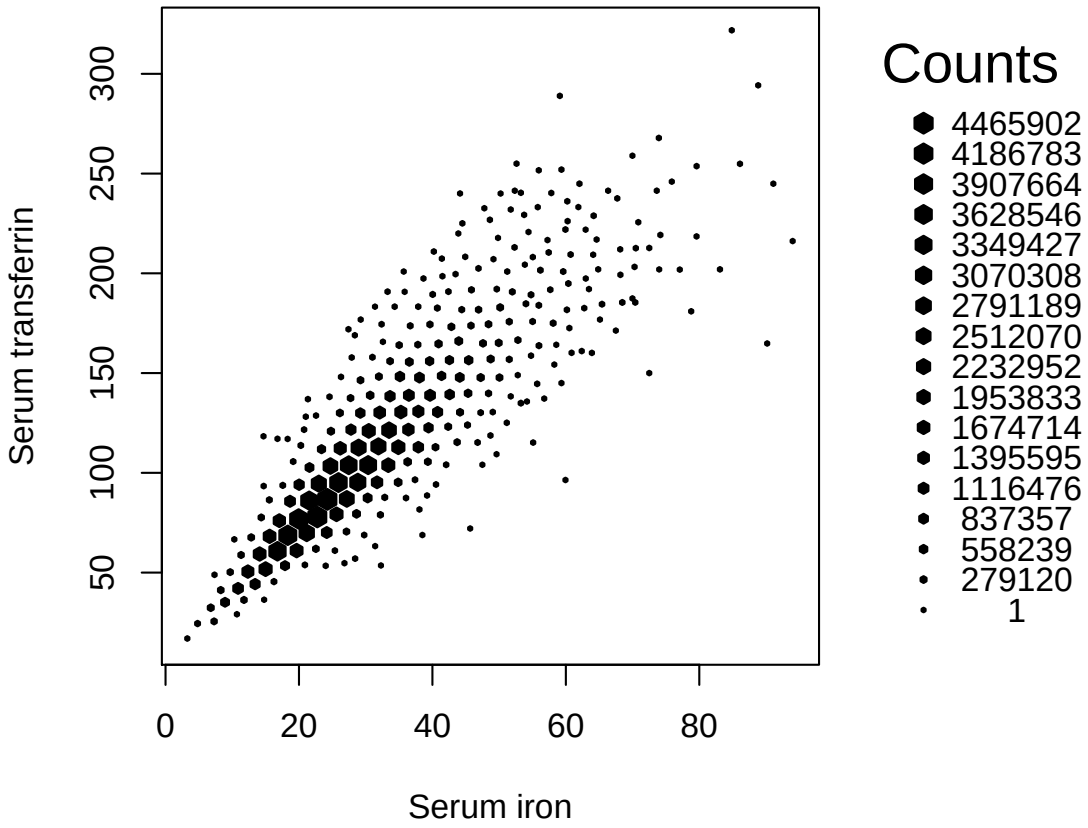
Data sets are large and weights are uneven, making graphics difficult

- Show weights explicitly
- Estimate population quantities and graph them
- Estimate population distribution and take iid sample

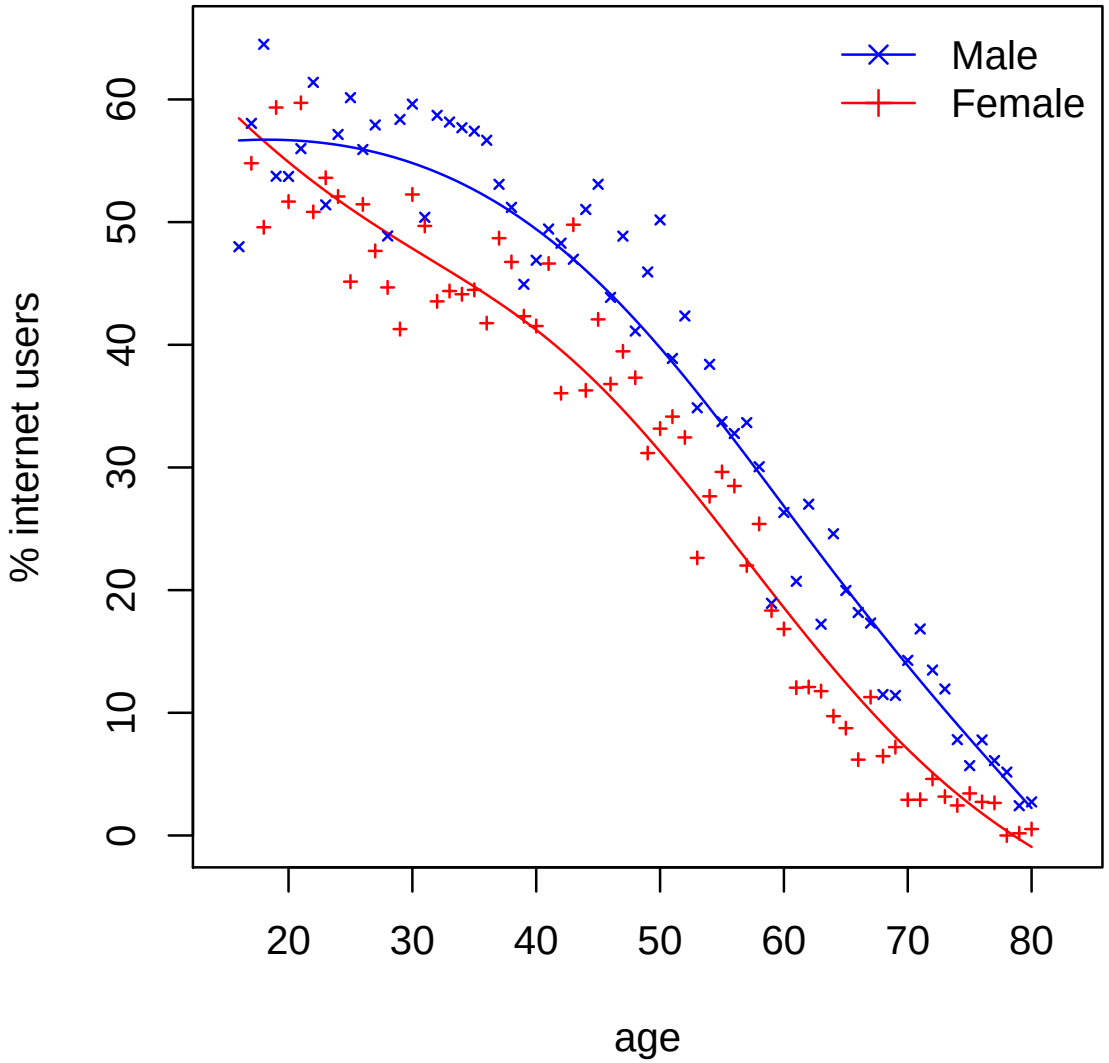
svyboxplot



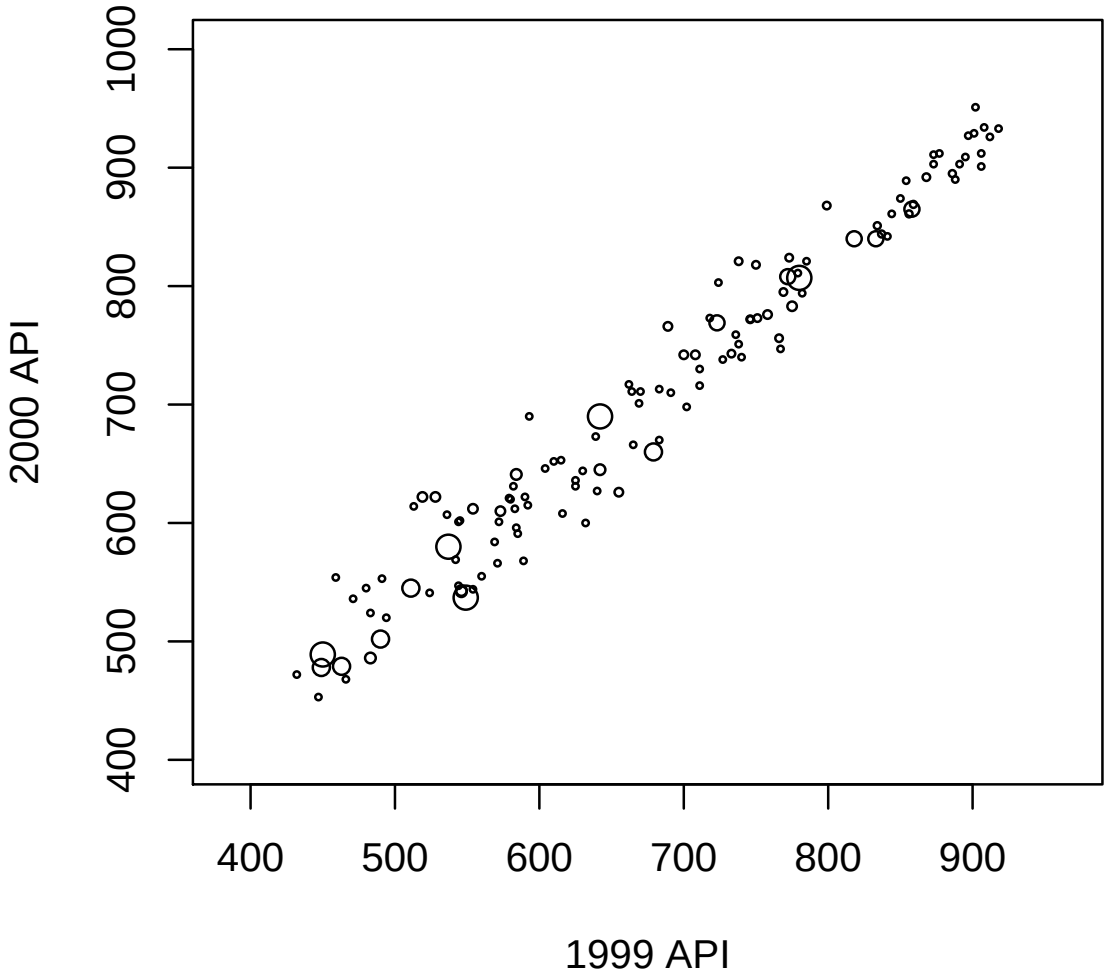
svyplot


Serum transferrin
50
100
150
200
250
300
0
20
40
60
80
Serum iron
Counts
4465902
4186783
3907664
3628546
3349427
3070308
2791189
2512070
2232952
1953833
1674714
1395595
1116476
837357
558239
279120
1

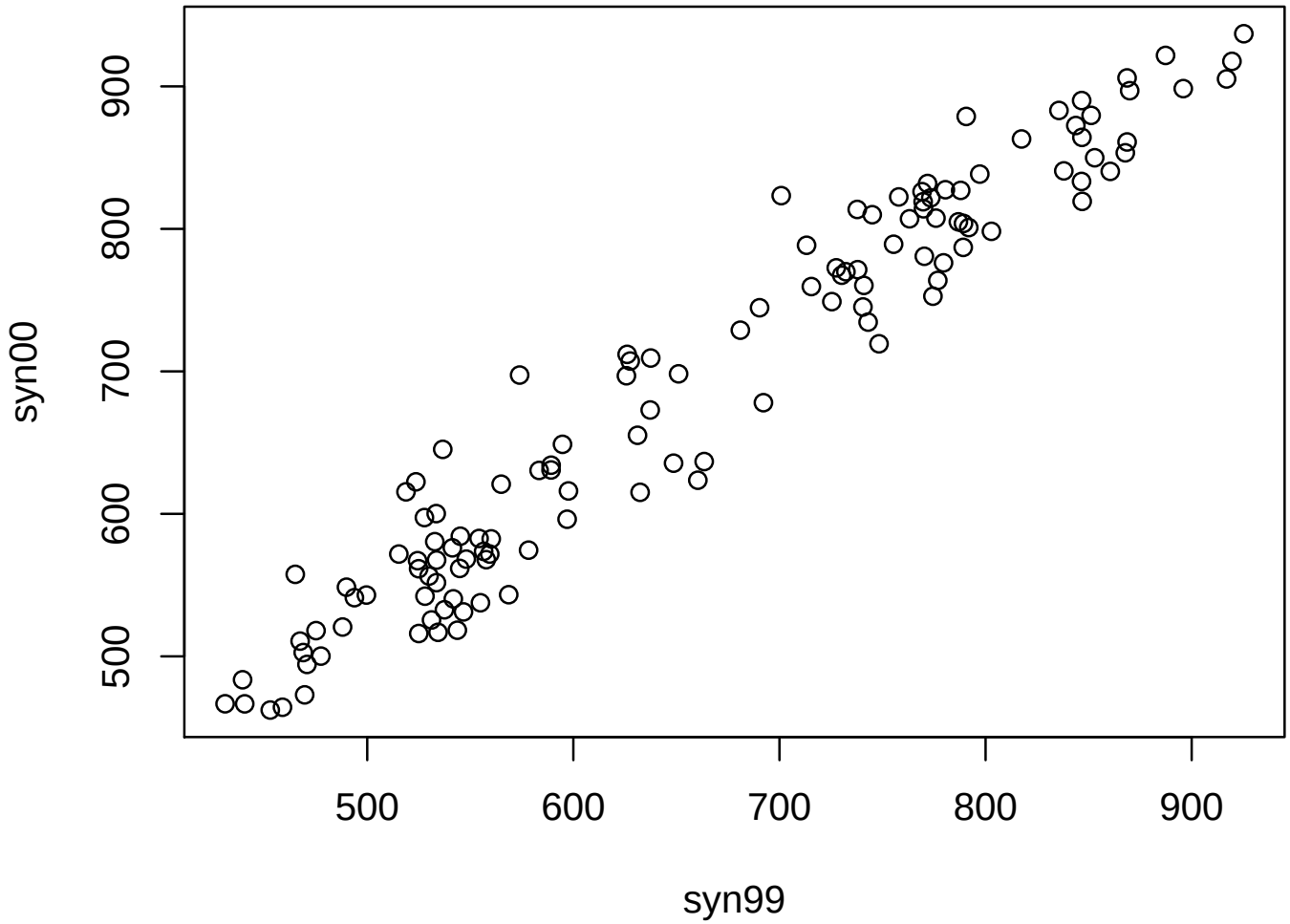
svysmooth


Male
Female
% internet users
0
10
20
30
40
50
60
20
30
40
50
60
70
80
age

svyboxplot


2000 API
400
500
600
700
800
900
1000
400
500
600
700
800
900
1999 API

svyplot



Design principles: interface

S has an established modelling interface where variables to be used in a computation are specified by a formula and a data frame.

The survey package uses this interface, but replaces the data frame with a survey object.

```
svymean(~api00, design=cal.schools)
svyratio(~api.stu, ~enroll, design=cal.schools)
svyglm(api00~meals+ell+mobility, design=cal.schools)
svytable(~sch.wide+stype, design=cal.schools)
svychisq(~sch.wide+stype, design=cal.schools)
svyby(~api00, by=~sch.wide+stype,
              design=cal.schools, svymean)
```

For regression models, the result of a model-fitting function is an object with methods to extract coefficients, variances, loglikelihood, AIC, residuals etc, as appropriate.

Design principles: interface

These functions return objects containing the results and other information that enables them to be printed attractively.

```
 > svyratio(~api.stu,~enroll,dclus1)
Ratio estimator: svyratio(~api.stu, ~enroll, dclus1)
Ratios=
           enroll
api.stu 0.8497087
SEs=
              enroll
api.stu 0.008386297
```

Here's the internal structure of the object

```
> str(svyratio(~api.stu,~enroll,dclus1))
List of 3
 $ ratio: num [1, 1] 0.85
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr "api.stu"
  .. ..$ : chr "enroll"
 $ var  : num [1, 1] 7.03e-05
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr "api.stu"
  .. ..$ : chr "enroll"
 $ call : language svyratio(~api.stu, ~enroll, dclus1)
 - attr(*, "class")= chr "svyratio"
```

Design principles: performance

Good design in a high-level interpreted language emphasizes simplicity of code rather than efficiency. Simple code is easier to get right. The survey package contains only about 7000 lines of code (cf 250,000 for VPLX)

On my desktop the replicate-weight computations for a fairly large dataset are ten times faster than `NASSVAR` on a 1984 IBM mainframe.

Optimisation for speed and memory is a useful next step. The R profiler allows program bottlenecks to be identified in real use, so that optimisation can be directed sensibly. If necessary, parts of the program can be replaced by C or Fortran code.

Design principles: extensibility

The modelling functions (eg `svyglm`, `svrepglm`) call the ordinary modelling functions (eg `glm`) to obtain point estimates.

Standard error computations are localized in two functions (`svyCprod` and `svrVar`).

For Taylor linearization, `svyCprod` computes the variance of the estimating functions. The other component of the variance, the inverse of the expected derivative of the estimating functions, is already available as the model-based variance estimate.

Creating a new estimator involves working out the correct calls to the model-based version and to `svyCprod` or `svrVar`, or writing a wrapper for `withReplicates` or `svymle`.

References

- Levy & Lemeshow Sampling of Populations
- Korn & Graubard Analysis of Health Surveys
- Lohr Sampling: Design and Analysis
- Särndal, Swensson, & Wretman Model Assisted Survey Sampling
- (Not yet) Lumley (2009) Complex surveys: analysis using R