Declare the design once

One design object, and every estimate after it inherits the weights, the strata and the clusters. The R survey package, the equivalent arithmetic in Python, and the four mistakes that silently produce a plain average.

Pierre Robentz Cassion

Lesson 5 of 8

Survey Analysis, Sampling and Weighting — Estimating properly

What this lesson covers

- Stop carrying the design by hand
- The design object in R
- The same thing in Python
- The ultimate cluster approximation
- The four ways to get a plain average by accident
- Estimates other than proportions
- The sub-sample design
- Write the design down beside the estimates
- What comes next

Stop carrying the design by hand

- Everything so far has been computed explicitly, which was the point — you should know what the arithmetic is.

The design object in R — In R

```r
library(survey)

design <- svydesign(
  ids     = ~ea_id,       # the primary sampling unit
  strata  = ~stratum,     # the stratification
  weights = ~weight,      # the weight built in lesson 2
  data    = survey,
  nest    = TRUE          # cluster ids are nested within strata
)

design
```

The design object in R

- `ids` is the *primary* sampling unit — the enumeration area, not the household. Passing `~household_id` here is the...
- `strata` removes the between-stratum variance, which is the gain the previous lesson measured.
- `weights` makes the point estimate unbiased.
- `nest = TRUE` tells the package that area identifiers are unique only within a stratum. Omit it where they are...

The design object in R — In R

```r
svymean(~I(food_insecure == "true"), design, deff = TRUE)
svytotal(~I(food_insecure == "true"), design)
svyby(~I(food_insecure == "true"), ~stratum, design, svymean, vartype = c("se", "ci"))
svyciprop(~I(food_insecure == "true"), design, method = "logit")
```

The same thing in Python — In Python (cont.)

```
import numpy as np
import pandas as pd


class SurveyDesign:
    """Stratified two-stage design, ultimate-cluster variance."""

    def __init__(self, data, ids, strata, weights):
        self.data, self.ids, self.strata, self.weights = data, ids, strata, weights

    def mean(self, indicator):
        d = self.data
        y = indicator(d).astype(float)
        w = d[self.weights]
        p = np.average(y, weights=w)
```

The same thing in Python — In Python (cont.)

```
        variance, total_w = 0.0, w.sum()
        for _, stratum in d.groupby(self.strata):
            residual = stratum[self.weights] * (indicator(stratum).astype(float) - p)
            totals = residual.groupby(stratum[self.ids]).sum()
            n = len(totals)
            if n < 2:
                continue
            variance += n / (n - 1) * ((totals - totals.mean()) ** 2).sum()
        se = np.sqrt(variance) / total_w

        srs = np.sqrt(p * (1 - p) / len(d))
        return {"estimate": p, "se": se, "deff": (se / srs) ** 2,
                "ci_low": p - 1.96 * se, "ci_high": p + 1.96 * se}


design = SurveyDesign(survey, ids="ea_id", strata="stratum", weights="weight")
```

The same thing in Python — In Python (cont.)

```
print(design.mean(lambda d: d["food_insecure"] == "true"))
```

The same thing in Python — In R

```r
# The R equivalent of the whole class above:
svymean(~I(food_insecure == "true"), design, deff = TRUE)
```

The ultimate cluster approximation

- Both implementations use the same shortcut, and it is worth naming because it looks like a simplification and is not.

The four ways to get a plain average by accident

- Passing the household as the cluster — `ids = ~household_id` says each household is its own PSU, so there is no...
- Forgetting the weights — `svydesign(..., weights = NULL)` gives every household equal weight, which is the 32.8% from...
- Filtering the data frame instead of subsetting the design — This one is subtle and common:

The four ways to get a plain average by accident — In R

```r
# WRONG: the design object no longer knows about the dropped strata
rural <- svydesign(ids = ~ea_id, strata = ~stratum, weights = ~weight,
                   data = filter(survey, stratum != "urban"), nest = TRUE)

# RIGHT
rural <- subset(design, stratum != "urban")
```

The four ways to get a plain average by accident — In Python

```
# Same rule: subset the design, keeping the full strata structure in view
rural = SurveyDesign(survey[survey["stratum"] != "urban"],
                     ids="ea_id", strata="stratum", weights="weight")
```

The four ways to get a plain average by accident

- Computing on a merged frame — Join the survey to anything one-to-many and the weights are now wrong, because a...

Estimates other than proportions — In R

```r
svymean(~household_size, design, na.rm = TRUE)
svytotal(~I(food_insecure == "true"), design)
svyquantile(~minutes_to_water, design, quantiles = c(0.5, 0.9), na.rm = TRUE)
svyratio(~I(food_insecure == "true"), ~I(children_under5 > 0), design)
```

Estimates other than proportions — In Python

```
print(design.mean(lambda d: d["household_size"] > 7))
```

The sub-sample design — In R

```
children <- subset(design, !is.na(child_muac_mm))
children <- update(children, child_weight = weight * children_under5)

child_design <- svydesign(ids = ~ea_id, strata = ~stratum,
                          weights = ~child_weight,
                          data = subset(survey, !is.na(child_muac_mm)), nest = TRUE)

svyciprop(~I(child_muac_mm < 125), child_design, method = "logit")
```

The sub-sample design — In Python

```
measured = survey[survey["child_muac_mm"].notna()].copy()
measured["child_weight"] = measured["weight"] * measured["children_under5"]

child_design = SurveyDesign(measured, ids="ea_id", strata="stratum",
                            weights="child_weight")
print(child_design.mean(lambda d: d["child_muac_mm"] < 125))
```

Write the design down beside the estimates — Example

Design	Stratified two-stage cluster
Strata	urban, rural-accessible, rural-remote
PSU	enumeration area (75 sampled, 25 per stratum)
Stage 1	PPS, size measure = households at frame listing
Stage 2	SRS, 14 households per area
Weights	base = M_h / (25 * 14), non-response adjusted within area
Variance	ultimate cluster, Taylor linearisation
Software	R survey 4.x / equivalent direct computation in Python

What comes next

- The design object assumes everyone you selected was interviewed and every area you drew was visited.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/survey-analysis-sampling/survey-05-survey-aware-estimation