Déclarez le plan une seule fois

Un objet de plan, et toute estimation ultérieure hérite des pondérations, des strates et des grappes. Le paquet survey de R, l'arithmétique équivalente en Python, et les quatre erreurs qui produisent en silence une simple moyenne.

Pierre Robentz Cassion

Leçon 5 sur 8

Analyse d'enquêtes, échantillonnage et pondération — Estimer correctement

Ce que couvre cette leçon

- Cessez de porter le plan à la main
- L'objet de plan sous R
- La même chose en Python
- L'approximation par grappe ultime
- Les quatre façons d'obtenir une simple moyenne par accident
- Des estimations autres que des proportions
- Le plan du sous-échantillon
- Écrivez le plan à côté des estimations
- La suite

Cessez de porter le plan à la main

- Tout ce qui précède a été calculé explicitement, et c'était voulu — il faut savoir ce qu'est l'arithmétique.

L'objet de plan sous R — En 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
```

L'objet de plan sous R

- `ids` est l'unité primaire de sondage — la zone de dénombrement, non le ménage. Passer `~household_id` ici est...
- `strata` retire la variance inter-strates, qui est le gain mesuré à la leçon précédente.
- `weights` rend l'estimation ponctuelle sans biais.
- `nest = TRUE` indique au paquet que les identifiants de zone ne sont uniques qu'à l'intérieur d'une strate....

L'objet de plan sous R — En 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")
```

La même chose en Python — En Python (suite)

```python
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)
```

La même chose en Python — En Python (suite)

```
        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")
```

La même chose en Python — En Python (suite)

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

La même chose en Python — En R

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

L'approximation par grappe ultime

- Les deux implémentations emploient le même raccourci, et il vaut la peine de le nommer car il ressemble à une simplification sans en être une.

Les quatre façons d'obtenir une simple moyenne par accident

- Passer le ménage comme grappe — `ids = ~household_id` dit que chaque ménage est sa propre unité primaire, donc qu'il...
- Oublier les pondérations — `svydesign(..., weights = NULL)` donne à chaque ménage le même poids, ce qui est le 32,8 %...
- Filtrer le tableau au lieu de sous-ensembler le plan — Celle-ci est subtile et fréquente

Les quatre façons d'obtenir une simple moyenne par accident — En 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")
```

Les quatre façons d'obtenir une simple moyenne par accident — En 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")
```

Les quatre façons d'obtenir une simple moyenne par accident

- Calculer sur un tableau joint — Joignez l'enquête à quoi que ce soit en un-à-plusieurs et les pondérations deviennent…

Des estimations autres que des proportions — En 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)
```

Des estimations autres que des proportions — En Python

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

Le plan du sous-échantillon — En R

```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")
```

Le plan du sous-échantillon — En 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))
```

Écrivez le plan à côté des estimations — Exemple

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

La suite

- L'objet de plan suppose que tous les ménages tirés ont été enquêtés et que toutes les zones tirées ont été visitées.

Lire la leçon complète, avec le code exécutable

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