Where to stop cutting

Three strata support an estimate. Stratum by displacement status gives nine cells with eighteen households in the smallest. A rule set before you look, and the table that shows a reader where the survey ran out.

Pierre Robentz Cassion

Lesson 8 of 8

Survey Analysis, Sampling and Weighting — What you may publish

What this lesson covers

- The request that always arrives
- What the cutting does
- Count clusters, not just households
- Set the rule before you look
- Suppress, but show the cell
- Domain estimation, and the thing that looks like a filter
- Plan the disaggregation before the survey
- The table to publish
- Where this course leaves you

The request that always arrives

- The survey report goes out with three stratum estimates.

What the cutting does — In Python

```
for keys in [["stratum"],
             ["stratum", "displacement_status"],
             ["stratum", "main_livelihood"]]:
    cells = survey.groupby(keys).size()
    print(f"{' x '.join(keys):40} {len(cells):>3} cells, "
          f"smallest {cells.min():>3}, {(cells < 50).sum()} below 50")
```

What the cutting does — In R

```
survey |> count(stratum) |> summarise(cells = n(), smallest = min(n))
survey |> count(stratum, displacement_status) |> summarise(cells = n(), smallest = min(n))
survey |> count(stratum, main_livelihood) |> summarise(cells = n(), smallest = min(n))
```

What the cutting does

Disaggregation	Cells	Smallest	Below 50
Stratum	3	316	0
Stratum $\times$ displacement	9	18	5
Stratum $\times$ livelihood	18	6	9

Count clusters, not just households — In Python

```
cells = survey.groupby(["stratum", "displacement_status"]).agg(
    households=("household_id", "size"),
    clusters=("ea_id", "nunique"),
)
print(cells.sort_values("clusters").head())
```

Count clusters, not just households — In R

```
survey |>
  summarise(households = n(), clusters = n_distinct(ea_id),
            .by = c(stratum, displacement_status)) |>
  arrange(clusters)
```

Count clusters, not just households

- A cell drawn from fewer than about ten clusters cannot support a variance estimate you would want to defend — however…

Set the rule before you look — In Python

```
MIN_HOUSEHOLDS = 50
MIN_CLUSTERS = 10
MAX_CI_WIDTH = 0.20      # 20 percentage points

def reportable(cell):
    return (cell["households"] >= MIN_HOUSEHOLDS
            and cell["clusters"] >= MIN_CLUSTERS
            and cell["ci_width"] <= MAX_CI_WIDTH)
```

Set the rule before you look — In R

```r
MIN_HOUSEHOLDS <- 50; MIN_CLUSTERS <- 10; MAX_CI_WIDTH <- 0.20

reportable <- function(d) {
  d$households >= MIN_HOUSEHOLDS & d$clusters >= MIN_CLUSTERS &
    d$ci_width <= MAX_CI_WIDTH
}
```

Suppress, but show the cell — In Python

```
by_cell = svyby_equivalent(survey, ["stratum", "displacement_status"], "food_insecure")
by_cell["reported"] = by_cell.apply(reportable, axis=1)
by_cell.loc[~by_cell["reported"], ["estimate", "ci_low", "ci_high"]] = None
by_cell["note"] = by_cell["reported"].map(
    {False: "too few clusters to estimate", True: ""}
)
print(by_cell)
```

Suppress, but show the cell — In R

```
by_cell <- svyby(~I(food_insecure == "true"), ~stratum + displacement_status,
                 design, svymean, vartype = "ci") |>
  mutate(reported = reportable(cur_data()),
         note = if_else(reported, "", "too few clusters to estimate"))
```

Suppress, but show the cell

- Never delete the row — A suppressed cell that still shows its household count and a reason tells the reader the group. . .

Domain estimation, and the thing that looks like a filter — In R

```r
# WRONG: rebuilding the design from a filtered frame
displaced <- svydesign(ids = ~ea_id, strata = ~stratum, weights = ~weight,
                       data = filter(survey, displacement_status == "displaced"),
                       nest = TRUE)

# RIGHT: a domain estimate keeps the full design in view
svyby(~I(food_insecure == "true"), ~displacement_status, design, svymean)
```

Domain estimation, and the thing that looks like a filter — In Python

```
# The same rule: the variance calculation must still see every cluster,
# including those with no displaced households in them.
```

Plan the disaggregation before the survey

- **Oversample the subgroup**, which is what stratification exists for. This survey oversampled rural remote precisely so...
- **Accept a wider interval** for that subgroup, stated in advance.
- **Drop the requirement**, and say in the protocol that the survey will not report on it.
- What you cannot do is decide afterwards — By the time the data exists the cells are whatever they are, and a table...

The table to publish — In Python

```
final = by_cell[["stratum", "displacement_status", "households", "clusters",
                 "estimate", "ci_low", "ci_high", "note"]]
final.to_csv("outputs/tables/food_insecurity_by_stratum_displacement.csv", index=False)
```

The table to publish — In R

```
readr::write_csv(by_cell,
  here::here("outputs", "tables", "food_insecurity_by_stratum_displacement.csv"))
```

The table to publish

Stratum	Status	Households	Clusters	Estimate	95% CI	Note
Urban	Resident	246	25	13.9%	10.2–18.7	
Urban	Displaced	49	18	—	—	too few households
Rural remote	Resident	279	25	46.1%	40.0–52.3	
Rural remote	Returnee	24	12	—	—	too few households

Where this course leaves you

- The last course in the module, *Routine Data and DHIS2*, goes back to the

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/survey-analysis-sampling/survey-08-how-far-to-disaggregate