What precision did the budget buy?

The sample size formula, the four numbers it needs, and why the answer for a cluster survey is roughly double the textbook one. Plus reading it backwards, which is what you usually have to do.

Pierre Robentz Cassion

Lesson 3 of 8

Survey Analysis, Sampling and Weighting — What the design buys

What this lesson covers

- The question is always asked too late
- The formula, and the four numbers in it
- Reading it backwards
- The number that is really being negotiated
- Clusters versus households per cluster
- Write the calculation down
- What comes next

The question is always asked too late

- Sample size is decided in a proposal, months before anyone analyses anything, and usually by whoever is writing the budget.

The formula, and the four numbers in it — Example

```
n = z^2 * p * (1 - p) / d^2 * deff
```

The formula, and the four numbers in it

- `z` — the confidence level. 1.96 for 95%. Almost never changed.
- `p` — the expected proportion. You do not know it, which is the point of the survey. Use the last round, a...
- `d` — the precision you need, as an absolute margin. This is the number that is actually negotiated, and the one...
- `deff` — the design effect. The next lesson is entirely about it; for sizing, use the previous round's value or 2.0...

The formula, and the four numbers in it — In Python

```python
import math


def sample_size(p, d, deff=1.0, z=1.96, response=1.0):
    n = z**2 * p * (1 - p) / d**2 * deff
    return math.ceil(n / response)


print(sample_size(0.30, 0.05))             # simple random
print(sample_size(0.30, 0.05, deff=2.4))   # cluster survey
print(sample_size(0.30, 0.03, deff=2.4))   # tighter precision
```

The formula, and the four numbers in it — In R

```r
sample_size <- function(p, d, deff = 1, z = 1.96, response = 1) {
  ceiling(z^2 * p * (1 - p) / d^2 * deff / response)
}

c(sample_size(0.30, 0.05),
  sample_size(0.30, 0.05, deff = 2.4),
  sample_size(0.30, 0.03, deff = 2.4))
```

The formula, and the four numbers in it

Expected p	Precision	deff	Households needed
30%	±5 points	1.0	323
30%	±5 points	2.4	775
30%	±3 points	2.4	2,152

The formula, and the four numbers in it

- Clustering more than doubles the requirement — 323 becomes 775 for the same precision
- Precision is brutally expensive — Going from ± 5 to ± 3 points nearly triples the sample
- Non-response is a divisor, not an afterthought — Expecting 90% response means dividing by 0.9 — and it must be applied...

Reading it backwards — In Python

```
def precision(n, p, deff=1.0, z=1.96):
    return z * math.sqrt(p * (1 - p) / n * deff)


print(f"+/- {precision(996, 0.291, deff=1.60):.1%}")
print(f"+/- {precision(343, 0.464, deff=1.60):.1%}")   # rural remote alone
```

Reading it backwards — In R

```r
precision <- function(n, p, deff = 1, z = 1.96) z * sqrt(p * (1 - p) / n * deff)

c(overall = precision(996, 0.291, deff = 1.60),
  remote  = precision(343, 0.464, deff = 1.60))
```

Reading it backwards

- Do this before you promise anything — It tells you which comparisons the survey can settle and which it cannot, and it...

The number that is really being negotiated

- **A threshold to cross.** If the question is whether GAM exceeds 15%, and the estimate is near 14%, you need an...
- **A change to detect.** Detecting a five-point improvement between rounds needs roughly *four times* the sample of...
- **A comparison between groups.** Same problem: two intervals, both of which have to be narrow.

The number that is really being negotiated — In Python

```python
# Detecting a difference between two groups of equal size
def sample_per_group(p1, p2, deff=1.0, power_z=0.84, z=1.96):
    pbar = (p1 + p2) / 2
    n = (z + power_z) ** 2 * 2 * pbar * (1 - pbar) / (p1 - p2) ** 2
    return math.ceil(n * deff)


print(sample_per_group(0.30, 0.25, deff=2.4))
```

The number that is really being negotiated — In R

```r
sample_per_group <- function(p1, p2, deff = 1) {
  pbar <- (p1 + p2) / 2
  ceiling((1.96 + 0.84)^2 * 2 * pbar * (1 - pbar) / (p1 - p2)^2 * deff)
}

sample_per_group(0.30, 0.25, deff = 2.4)
```

Clusters versus households per cluster — In Python

```
for m in [8, 14, 20, 30]:
    deff = 1 + (m - 1) * 0.11
    clusters = math.ceil(775 * (deff / 2.4) / m)
    print(f"{m:>3} households x {clusters:>3} clusters -> deff {deff:.2f}")
```

Clusters versus households per cluster — In R

```
for (m in c(8, 14, 20, 30)) {
  deff <- 1 + (m - 1) * 0.11
  cat(sprintf("%3d households, deff %.2f\n", m, deff))
}
```

Write the calculation down — Example

```
Indicator          Household food insecurity prevalence
Expected p         0.30 (2023 round, same instrument)
Precision          +/- 5 percentage points, 95% confidence
Design effect      2.0 (2023 round measured 1.9; rounded up)
Expected response  92%
Households         323 * 2.0 / 0.92 = 703, rounded to 25 clusters x 14 = 350 per
                   stratum, 1,050 in total across three strata
Rationale          Equal allocation, because each stratum must support its own
                   estimate. National figures require weighting.
```

What comes next

- Two of the four inputs were the design effect, and it has been a placeholder so far.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/survey-analysis-sampling/survey-03-sample-size