Building the weights from the frame

Selection probability at each stage, the base weight it implies, the non-response adjustment, and the check that proves the whole thing — weights that sum to 56,428, which is exactly what the frame says exists.

Pierre Robentz Cassion

Lesson 2 of 8

Survey Analysis, Sampling and Weighting — The sample is not the population

What this lesson covers

- A weight is one number with one meaning
- Stage one: probability proportional to size
- Stage two: a fixed take
- The two probabilities cancel
- The non-response adjustment
- The check that proves the whole construction
- Normalised weights, and when to use them
- Person-level and sub-sample weights
- Save the weights beside the data
- What comes next

A weight is one number with one meaning

- A sampling weight is how many population units this sampled unit stands for

Stage one: probability proportional to size — Example

```
P(area i selected) = n_h * M_i / M_h
```

Stage one: probability proportional to size — In Python

```
import pandas as pd

frame = pd.read_csv("household-survey-frame-2025.v1.csv")

stratum_households = frame.groupby("stratum")["households"].sum()
selected = frame[frame["selected"] == True].copy()

selected["p_stage1"] = (
    25 * selected["households"] / selected["stratum"].map(stratum_households)
)
print(selected[["ea_id", "stratum", "households", "p_stage1"]].head())
```

Stage one: probability proportional to size — In R

```r
library(dplyr)

stratum_households <- frame |> summarise(M_h = sum(households), .by = stratum)

selected <- frame |>
  filter(selected) |>
  left_join(stratum_households, by = "stratum") |>
  mutate(p_stage1 = 25 * households / M_h)
```

Stage two: a fixed take — Example

```
P(household selected | area selected) = m / M_i
```

Stage two: a fixed take — In Python

```
selected["p_stage2"] = 14 / selected["households"]
selected["p_overall"] = selected["p_stage1"] * selected["p_stage2"]
print(selected.groupby("stratum")["p_overall"].describe()[["min", "max"]])
```

Stage two: a fixed take — In R

```
selected <- selected |>
  mutate(p_stage2 = 14 / households,
         p_overall = p_stage1 * p_stage2)

selected |> summarise(min = min(p_overall), max = max(p_overall), .by = stratum)
```

The two probabilities cancel — Example

```
P(household) = (n_h * M_i / M_h) * (m / M_i) = n_h * m / M_h
```

The two probabilities cancel

- Two consequences worth holding — A weight that varies within a stratum in a PPS design means something is wrong —. . .

The two probabilities cancel — In Python

```
base_weight = stratum_households / (25 * 14)
print(base_weight.round(1))
```

The two probabilities cancel — In R

```
stratum_households |> mutate(base_weight = M_h / (25 * 14))
```

The two probabilities cancel

Stratum	Frame households	Base weight
Urban	21,270	60.8
Rural accessible	28,958	82.7
Rural remote	6,200	17.7

The non-response adjustment — In Python

```
survey = pd.read_csv("household-survey-2025.v1.csv")

interviewed = selected.set_index("ea_id")["households_interviewed"]

survey["base_weight"] = survey["stratum"].map(base_weight)
survey["nr_adjustment"] = 14 / survey["ea_id"].map(interviewed)
survey["weight"] = survey["base_weight"] * survey["nr_adjustment"]

print(survey["weight"].describe()[["min", "max"]].round(1))
```

The non-response adjustment — In R

```
survey <- survey |>
  left_join(select(selected, ea_id, households_interviewed), by = "ea_id") |>
  left_join(mutate(stratum_households, base_weight = M_h / (25 * 14)), by = "stratum") |>
  mutate(weight = base_weight * 14 / households_interviewed)

range(survey$weight)
```

The non-response adjustment

- Adjust within the area, not within the stratum — The households that did not answer in a given area resemble the...

The check that proves the whole construction — In Python

```
total = survey["weight"].sum()
frame_total = frame["households"].sum()
print(f"weights sum to {total:,.0f}; frame holds {frame_total:,}")
assert abs(total - frame_total) < 1
```

The check that proves the whole construction — In R

```
c(weights = sum(survey$weight), frame = sum(frame$households))
```

The check that proves the whole construction

- 56,428 and 56,428 — The weights sum to exactly the number of households the frame says exist, which is what "how many...

Normalised weights, and when to use them — In Python

```
survey["weight_norm"] = survey["weight"] / survey["weight"].mean()
```

Normalised weights, and when to use them — In R

```
survey <- survey |> mutate(weight_norm = weight / mean(weight))
```

Person-level and sub-sample weights

- A person-level weight — is the household weight times the household size, because a household of eight represents eight...

Person-level and sub-sample weights — In Python

```
people = survey[survey["household_size"].notna()].copy()
people["person_weight"] = people["weight"] * people["household_size"]
print(f"estimated population {people['person_weight'].sum():,.0f}")
```

Person-level and sub-sample weights — In R

```
people <- survey |>
  filter(!is.na(household_size)) |>
  mutate(person_weight = weight * household_size)

sum(people$person_weight)
```

Person-level and sub-sample weights

- A sub-sample weight — applies when one unit is chosen from several

Person-level and sub-sample weights — In Python

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

Person-level and sub-sample weights — In R

```
children <- survey |>
  filter(!is.na(child_muac_mm)) |>
  mutate(child_weight = weight * children_under5)
```

Save the weights beside the data — In Python

```
survey[["household_id", "ea_id", "stratum", "base_weight",
        "nr_adjustment", "weight"]].to_csv("outputs/weights.csv", index=False)
```

Save the weights beside the data — In R

```
survey |>
  select(household_id, ea_id, stratum, base_weight, households_interviewed, weight) |>
  readr::write_csv(here::here("outputs", "weights.csv"))
```

What comes next

- You can now produce a population estimate.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/survey-analysis-sampling/survey-02-reconstructing-weights