The thirty-seven households zero-filling would misclassify

Zero-filling an incomplete Household Hunger Scale moves the prevalence by three tenths of a point and can misclassify every one of the thirty-seven households it touches. Which of those matters depends on whether your output is a percentage or a list.

Pierre Robentz Cassion

Lesson 2 of 8

What this lesson covers

- Two more instruments, two more rules
- The Household Hunger Scale
- What zero-filling actually costs
- The reduced Coping Strategy Index
- Three instruments, three analysable samples
- What comes next

Two more instruments, two more rules

Instrument	What it asks	The rule that gets broken
HHS	Three questions on going without food, scored 0–2 each	Valid only when **all three** are answered
rCSI	Five behaviours, days in the last seven, weighted	The weights are not all 1, and the fifth is 3

The Household Hunger Scale — In Python

```
import pandas as pd

survey = pd.read_csv("food-security-survey-2024.v1.csv")
HHS = ["hhs_no_food_in_house", "hhs_sleep_hungry",
       "hhs_day_and_night_without_eating"]

complete = survey[HHS].notna().all(axis=1)
score = survey.loc[complete, HHS].sum(axis=1)

bands = pd.cut(score, [-1, 1, 3, 6],
               labels=["little or none", "moderate", "severe"])
print(f"complete: {complete.sum()} of {len(survey)}")
print((bands.value_counts(normalize=True) * 100).round(1))
```

The Household Hunger Scale — In R

```r
library(dplyr)

hhs <- c("hhs_no_food_in_house", "hhs_sleep_hungry",
         "hhs_day_and_night_without_eating")

survey |>
  filter(if_all(all_of(hhs), ~ !is.na(.x))) |>
  mutate(score = rowSums(across(all_of(hhs))),
         band = cut(score, c(-1, 1, 3, 6),
                    labels = c("little or none", "moderate", "severe"))) |>
  count(band) |> mutate(share = n / sum(n))
```

The Household Hunger Scale

Band	Households	Share
Little or no hunger (0–1)	1,186	57.2%
Moderate (2–3)	564	27.2%
Severe (4–6)	325	**15.7%**

The Household Hunger Scale

- 2,075 of 2,112 households answered all three — The thirty-seven that did not are the interesting ones

What zero-filling actually costs — In Python

```
zero_filled = survey[HHS].fillna(0).sum(axis=1)
naive = pd.cut(zero_filled, [-1, 1, 3, 6],
               labels=["little or none", "moderate", "severe"])
print((naive.value_counts(normalize=True) * 100).round(1))
```

What zero-filling actually costs — In R

```
survey |>
  mutate(score = rowSums(across(all_of(hhs)), na.rm = TRUE)) |>
  count(band = cut(score, c(-1, 1, 3, 6)))
```

What zero-filling actually costs

	Complete cases	**Zero-filled**
Little or none	57.2%	57.5%
Moderate	27.2%	26.9%
Severe	15.7%	15.5%

What zero-filling actually costs

- Three tenths of a percentage point — On a prevalence, zero-filling is practically harmless here, and if you stop at the...

What zero-filling actually costs — In Python

```
partial = survey.loc[~complete, HHS]
print(partial.sum(axis=1).value_counts().sort_index())
```

What zero-filling actually costs — In R

```
survey |> filter(if_any(all_of(hhs), is.na)) |>
  mutate(observed = rowSums(across(all_of(hhs)), na.rm = TRUE)) |> count(observed)
```

What zero-filling actually costs

- So the cost of zero-filling depends entirely on what the analysis produces
- If the output is a **prevalence**, the damage is three tenths of a point and you should say you zero-filled and move on.
- If the output is a **targeting list**, you have just told thirty-seven households they are food secure on the strength. . .
- Name the output before choosing the rule — This is the same decision as the CMAM cure-rate denominator and the water. . .

The reduced Coping Strategy Index — In Python

```
RCSI = {
    "rcsi_less_preferred_food": 1,
    "rcsi_borrowed_food": 2,
    "rcsi_limit_portion_size": 1,
    "rcsi_restrict_adult_consumption": 3,
    "rcsi_reduce_meal_numbers": 1,
}
rcsi = sum(survey[column] * weight for column, weight in RCSI.items())
print(f"median {rcsi.median():.0f}, mean {rcsi.mean():.1f}, max {rcsi.max():.0f}")
print(f"rCSI 19 or above: {(rcsi >= 19).mean():.1%}")
```

The reduced Coping Strategy Index — In R

```
rcsi_weights <- c(rcsi_less_preferred_food = 1, rcsi_borrowed_food = 2,
                  rcsi_limit_portion_size = 1,
                  rcsi_restrict_adult_consumption = 3,
                  rcsi_reduce_meal_numbers = 1)
```

The reduced Coping Strategy Index

- Median 19, mean 19.5, and 50.9% at or above 19 — The weight of 3 on *restricting adult consumption so children can eat*...
- It has no universal threshold — Unlike the FCS, the rCSI's cut-offs are context-specific and usually set against the...
- It goes up before consumption goes down — and it can also go *down* in a crisis, when a household has exhausted the...

Three instruments, three analysable samples — In Python

```
denominators = pd.DataFrame({
    "instrument": ["Food Consumption Score", "Household Hunger Scale",
                   "reduced Coping Strategy Index"],
    "analysable": [1989, 2075, 2112],
    "excluded": [123, 37, 0],
    "rule": ["all eight groups answered", "all three questions answered",
             "complete in this file"],
})
print(denominators)
```

Three instruments, three analysable samples — In R

```
# Print this before the results, every time.
```

Three instruments, three analysable samples

- Three numbers on three denominators, and none of them is 2,112 — The difference is small enough to be invisible in a...

What comes next

- Consumption and hunger say what a household is experiencing.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/food-security-ipc/fs-02-hunger-and-coping