CASSION · COURSE

Regression for Programme Data

Linear, logistic and multilevel models fitted to survey and routine data, with the interpretation written the way a non-statistician programme manager needs to read it.

Instructor	Pierre Robentz Cassion
Level	Advanced
Learner hours	20 h
Taught in	Python · R
Updated	July 29, 2026
Sectors	Education · Protection · Nutrition · Food Security
Standards and methodologies	SMART survey · Demographic and Health Survey (DHS) · UNICEF indicator definitions · OECD DAC evaluation criteria

Read and run the course online at data-analysis.cassion.dev/courses/regression-for-programmes

What you will be able to do

- Read a regression coefficient as the comparison of means it is, and say which comparison
- Distinguish adjustment, which moves the estimate, from clustering, which moves the standard error
- Fit a logistic model and report it as a risk difference rather than as an odds ratio a reader will misread
- Decide which covariates belong in a model and which ones destroy the answer by being there
- Fit a random intercept where the design assigned above the level of the row
- Weight a model to its sampling design, and report a model that does not fit rather than dropping it

Prerequisites

- Applied Statistics for Programmes

Contents

About this course

A regression is not a more sophisticated instrument than the comparisons in *Applied Statistics for Programmes*. It is the same comparison with more than one thing held fixed, and the first lesson proves it: fit a model with a single dummy variable and the coefficient **is** the difference between two group means, to every decimal place.

That is the spine of the course. A coefficient is always a comparison, and the work is saying which one.

Three results carry it, all fitted on files already committed.

Adjustment and clustering are different problems with different fixes. The school feeding effect is 4.9 points with a standard error of 0.9 fitted naively. Adding every child-level covariate in the register moves the estimate to 4.4 and leaves the standard error alone. Adding a term for the school leaves the estimate alone and moves the standard error to 1.5. Confusing the two is the commonest error in a programme regression, and it is why they are taught in the same course.

An odds ratio of 0.43 describes a risk ratio of 0.58. Cases reporting a disability complete a referral at 26.4% against 45.5%. The odds ratio is what a logistic model prints and the risk difference is what a reader thinks they are being told, and the gap between them widens exactly when the outcome is common — which, in programme data, it usually is.

Adjusting for the wrong covariate removes 42% of the effect. Referral completion runs through a gate — whether a referral was made at all — and cases reporting a disability clear that gate at 53.8% against 71.5%. Put the gate in the model and the disability gap falls from 19.4 points to 11.2, not because the estimate got better but because the model started answering a different question.

Both Python and R throughout. You need *Applied Statistics for Programmes* first; this course assumes you can already put an interval on a number and say what a p-value does not tell you.

Part I

A model is a comparison

CHAPTER 1

The coefficient is the difference

Lesson 1 of 8 · 150 min

Fit attendance against a single feeding dummy. The intercept is 85.21% — the mean of schools with no programme. The coefficient is 4.94 points — the difference between the two means, to four decimal places. A regression with one dummy is a two-group comparison and nothing more.

1.1 The same comparison, written twice

The statistics course compared attendance in schools with a feeding programme against schools without one, and got 90.15% against 85.21%. Fit that as a regression and look at what comes back.

```python
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf

attendance = pd.read_csv("school-attendance-2024.v1.csv")
roster = pd.read_csv("school-roster-2024.v1.csv").drop_duplicates(
    "student_id", keep="last")          # two never-de-registered transfers

MARKS = {"true": True, "Y": True, "false": False, "N": False}
marked = attendance[attendance["present"].isin(MARKS)].copy()
marked["attended"] = marked["present"].map(MARKS)

per_student = (marked.groupby("student_id")["attended"].mean()
               .rename("rate").reset_index()
               .merge(roster, on="student_id"))

model = smf.ols("rate ~ feeding_programme", data=per_student).fit()
print(model.summary().tables[1])
```

```r
library(dplyr)

per_student |> lm(rate ~ feeding_programme, data = _) |> summary()
```

Term	Coefficient	SE	t
Intercept	**0.8521**	0.0070	121.30
feeding_programme[True]	**0.0494**	0.0088	5.63

```python
means = per_student.groupby("feeding_programme")["rate"].mean()
print(means)
print(f"difference: {means[True] - means[False]:.4f}")
```

```r
per_student |> summarise(rate = mean(rate), .by = feeding_programme)
```

Mean without a programme: 0.8521. Mean with: 0.9015. Difference: 0.0494. The intercept is the first number and the coefficient is the third, exactly. **This is not an approximation or a coincidence** — with one binary predictor and nothing else, least squares has no freedom to do anything but reproduce the two group means.

1.2 What every term means

Three sentences cover the whole of reading a coefficient table, and they hold for every model in this course.

The intercept is the fitted value when every predictor is zero. Here that is `feeding_programme = False`, so the intercept is the mean of schools with no programme. It is not "the average attendance" and it is not a baseline in any programme sense — it is a specific group's mean, and which group depends entirely on how the predictors were coded.

A coefficient is the change in the outcome for a one-unit change in that predictor, with the others held fixed. For a dummy, "one unit" is False → True, so the coefficient is a difference between two groups.

A standard error is the same standard error the last course computed. The `t` column is coefficient over standard error, and the interval is the coefficient plus or minus about two standard errors. Nothing new is being introduced.

```python
coef = model.params["feeding_programme[T.True]"]
se = model.bse["feeding_programme[T.True]"]
print(f"{coef:+.4f}  95% CI [{coef - 1.96*se:+.4f}, {coef + 1.96*se:+.4f}]")
```

```r
confint(model)
```

+0.0494, 95% CI +0.0322 to +0.0665 — the same interval, arrived at from the same data by a different route.

1.3 Where the two routes differ, and by how much

The regression `t` is 5.63 and the Welch t-test in the statistics course gave 5.41. That difference is not an error in either.

Ordinary least squares assumes one residual variance for both groups. It is the pooled-variance t-test written in matrix form. Welch's test does not assume it, which is why the last course recommended Welch by default.

```python
from scipy import stats

fed = per_student[per_student["feeding_programme"]]["rate"]
unfed = per_student[~per_student["feeding_programme"]]["rate"]
print(f"variances: {fed.var():.5f} and {unfed.var():.5f}")
print("pooled t:", stats.ttest_ind(fed, unfed).statistic.round(2))
print("Welch  t:", stats.ttest_ind(fed, unfed, equal_var=False).statistic.round(2))
```

```r
t.test(rate ~ feeding_programme, data = per_student, var.equal = TRUE)
t.test(rate ~ feeding_programme, data = per_student)
```

Regression reproduces the pooled t exactly. When the group variances are close the two barely differ; here they are 0.019 and 0.025, and the gap is the reason Welch exists.

A regression cannot be told "use Welch" — the fix is a robust standard error, which lesson 6 introduces for a related reason.

1.4 Three predictors, three readings

A regression's usefulness starts when the predictor is not binary. Each type reads differently and each is a comparison.

Predictor	Coefficient reads as
Binary (`feeding_programme`)	The difference between the two groups
Continuous (`age_years`)	The change per one year
Categorical with k levels	k−1 coefficients, each a difference against the omitted level

```python
enrolment = pd.read_csv("school-enrolment-2024.v1.csv")
current = enrolment[enrolment["school_year"] == 2024]
joined = per_student.merge(
    current[["student_id", "sex", "age_years", "admin2"]], on="student_id")

print(smf.ols("rate ~ admin2", data=joined).fit().params.round(4))
print(joined.groupby("admin2")["rate"].mean().round(4))
```

```r
lm(rate ~ admin2, data = joined) |> coef()
joined |> summarise(rate = mean(rate), .by = admin2)
```

A categorical predictor produces one coefficient per level except one, and every coefficient is a comparison against that omitted level. Change which level is omitted and every number in the column changes while the model is identical — which is the first sign that a coefficient means nothing without knowing what it is being compared against.

Say the reference level in the table caption. "Against Artibonite" is four words and it is the difference between a readable table and one a reader has to guess at.

1.5 What the model does not say

Two things a coefficient never carries, both of which readers supply for themselves.

It is not an effect. The 4.9-point coefficient is the difference between two sets of schools that were not randomised into having a feeding programme. Every regression in this course is a comparison of groups that already differed; lesson 5 is about which of those differences you can and cannot remove.

It is not a prediction worth making. Regression is taught elsewhere as a prediction machine, and for programme data that framing does damage: it invites model comparison by fit rather than by whether the comparison is the one the report needs. **Fit the model your question implies, then read the coefficient it was fitted for**, and treat R^2 as a diagnostic rather than a score — lesson 8 is about a model with an R^2 of 0.03 that is the most useful result in its report.

1.6 Report it whole

```
Attendance and school feeding, February to April 2024
```

```
Linear model, one predictor, 1,200 students.

Intercept (no programme)   85.21%
Feeding programme          +4.94 points   95% CI +3.22 to +6.65

The coefficient is the difference between the two group means and is
reported as such. Schools were not randomised into the programme; the
comparison is observational.

The standard error assumes independent students. It is not, and lesson 6
gives the corrected version.
```

The last line is a promissory note the course pays off, and writing it is better than not knowing it is owed. A model reported with a caveat you can name is in better shape than one reported without.

1.7 What comes next

One predictor reproduces a comparison you could have made without a model. The next lesson adds the second predictor, which is where a regression starts earning its keep — and where "adjusting for" gets a precise meaning that is narrower than most reports assume.

CHAPTER 2

Adjustment moves the estimate; clustering moves the standard error

Lesson 2 of 8 · 150 min

Adding every child-level covariate in the register moves the feeding coefficient from 4.47 to 4.37 and leaves the standard error at 0.9. Adding a term for the school leaves the coefficient alone and moves the standard error to 1.5. These are two different problems and they are routinely confused.

2.1 The one table this course is built around

```python
import pandas as pd
import statsmodels.formula.api as smf

attendance = pd.read_csv("school-attendance-2024.v1.csv")
roster = pd.read_csv("school-roster-2024.v1.csv").drop_duplicates(
    "student_id", keep="last")
MARKS = {"true": True, "Y": True, "false": False, "N": False}
marked = attendance[attendance["present"].isin(MARKS)].copy()
marked["attended"] = marked["present"].map(MARKS)
per_student = (marked.groupby("student_id")["attended"].mean()
               .rename("rate").reset_index().merge(roster, on="student_id"))

enrolment = pd.read_csv("school-enrolment-2024.v1.csv")
current = enrolment[enrolment["school_year"] == 2024]

d = per_student.merge(
    current[["student_id", "sex", "age_years", "disability_reported",
             "displacement_status", "admin2"]], on="student_id").dropna()
print(f"complete cases: {len(d)}")

m1 = smf.ols("rate ~ feeding_programme", data=d).fit()
m2 = smf.ols("rate ~ feeding_programme + sex + age_years"
             " + disability_reported + displacement_status", data=d).fit()
m3 = smf.ols("rate ~ feeding_programme + sex + age_years"
             " + disability_reported + displacement_status + admin2", data=d).fit()
m4 = m1.get_robustcov_results(cov_type="cluster", groups=d["school_id"])

for name, m in [("child covariates", m2), ("+ district", m3)]:
    print(f"{name:18} {m.params['feeding_programme[T.True]']:+.4f}")
```

```r
library(dplyr)

m1 <- lm(rate ~ feeding_programme, data = d)
m2 <- lm(rate ~ feeding_programme + sex + age_years +
           disability_reported + displacement_status, data = d)
m3 <- update(m2, . ~ . + admin2)
```

Model	Feeding coefficient	SE	t
M1 one predictor	**+0.0447**	0.0089	5.02
M2 + sex, age, disability, displacement	**+0.0437**	0.0090	4.85
M3 + district	**+0.0360**	0.0100	3.60
M4 = M1 with school-clustered SEs	**+0.0447**	**0.0154**	**2.90**

Read the columns rather than the rows, because they say two different things.

Adding covariates changed the coefficient and left the standard error alone. M1 to M3 moves the estimate from 4.5 points to 3.6 while the standard error goes from 0.9 to 1.0.

Clustering changed the standard error and left the coefficient untouched. M4 is the *same fitted line* as M1 — identical coefficient to every decimal — with a standard error 1.7 times larger.

These are two different problems. One is about whether the comparison is between comparable groups; the other is about how much information the sample actually contains. Fixing one does nothing for the other, and a model that fixes neither is wrong in two independent ways.

2.2 What "adjusting for" actually does

The phrase appears in every programme report and means something narrower than readers assume.

A coefficient with covariates in the model is the comparison within levels of those covariates, pooled across them. "Adjusted for district" means: compare fed and unfed students *inside* each district, then average those comparisons.

```
within = (d.groupby("admin2")
          .apply(lambda g: g[g["feeding_programme"]]["rate"].mean()
                 - g[~g["feeding_programme"]]["rate"].mean()))
print(within.round(4))
print(f"unadjusted: {m1.params['feeding_programme[T.True]']:.4f}")
print(f"adjusted:   {m3.params['feeding_programme[T.True]']:.4f}")
```

```
d |> summarise(gap = mean(rate[feeding_programme]) -
                     mean(rate[!feeding_programme]), .by = admin2)
```

District	Gap	Fed	Unfed
Artibonite	+3.8 pts	146	141
Centre	**−4.1 pts**	244	**42**
Nord-Ouest	+4.9 pts	251	45
Sud	+7.0 pts	104	178

The adjusted coefficient is a weighted average of those four gaps, and seeing them separately is worth the two lines it costs — because one of them runs the other way. Centre's fed schools attend 4.1 points *worse* than its unfed schools, against a pooled estimate of +3.6.

A single coefficient averaged over a disagreement is still a number, and it is no longer a summary. Before reporting the adjusted 3.6, say that Centre disagrees and that its comparison rests on 42 unfed students — which is thin enough that the reversal may be noise, and thin enough that it should be checked rather than averaged away.

2.3 Why the covariates barely moved anything

M2 adds sex, age, disability and displacement status and moves the coefficient by one tenth of a point. That is not a failure of the covariates — it is a fact about the design.

The feeding programme was assigned by school. Nothing about a *child* decided whether they got school meals, so child characteristics cannot be confounders of the feeding comparison, and adjusting for them cannot move the estimate much.

District did move it, from 4.5 to 3.6, and for the same reason in reverse: schools with a feeding programme are not evenly spread across districts, so district is a genuine confounder of a school-level exposure.

The rule the design implies: confounders live at the level of assignment. For a school-level programme, look for school-level and district-level differences; adding child-level covariates is neither harmful nor useful here, and mostly signals that the analyst reached for whatever the file happened to carry.

2.4 Reading a coefficient table without being misled

Four habits, each of which prevents a specific misreading.

Report the sample size of the model, not of the file. Complete-case fitting dropped 49 students here — 1,200 rows became 1,151 — and the drop is silent. Every model in the table above must be fitted on the *same* rows or the comparison across models is contaminated by which rows each one kept.

```python
print(f"file {len(per_student)}, model {int(m3.nobs)}")
```

```r
c(file = nrow(per_student), model = nobs(m3))
```

Do not read the covariate coefficients as findings. They are adjusted for a set of variables chosen to answer a question about feeding, not about sex or displacement. A coefficient is interpretable only for the exposure the model was built around — this is the "Table 2 fallacy", and it is how a displacement coefficient fitted as a control ends up quoted as a displacement finding.

Check whether an added covariate changed the sample. A covariate with missing values silently drops rows, so a coefficient that moves when you add it may have moved because the sample changed rather than because the adjustment did anything.

Report the unadjusted estimate too. The pair is what shows the reader how much work the adjustment did, and a report giving only the adjusted number has hidden the one comparison that needs no assumptions.

2.5 When adjustment cannot help

Three cases where adding a covariate is not the answer, all of which appear later in this course.

The confounder is not in the file. Adjusting for what you measured says nothing about what you did not, and "adjusted" is not a synonym for "unconfounded". The epidemiology course made this point with age standardisation; regression does not improve on it.

The covariate is on the causal path. Lesson 5 shows a covariate that removes 42% of the effect by sitting between exposure and outcome, and adding it makes the answer worse in a way no fit statistic reveals.

The problem is the standard error, not the estimate. No covariate fixes clustering. That is the M1-to-M4 row, and lesson 6 is the whole answer to it.

2.6 Report it whole

```
Attendance and school feeding, adjusted analysis

  Linear model, 1,151 students with complete covariates (49 dropped).

  Unadjusted                     +4.47 points   95% CI +2.73 to +6.22
  Adjusted for district          +3.60 points   95% CI +1.64 to +5.56
  Child-level covariates (sex, age, disability, displacement) change the
  estimate by less than 0.1 points and are retained only to show that.

  The district gaps are +3.8, -4.1, +4.9 and +7.0 points. Centre runs the
  other way on 42 unfed students and is not averaged away silently.

  Standard errors above assume independent students and are too small; the
  school-clustered version of the unadjusted model gives +4.47 (95% CI
  +1.45 to +7.49).

  Observational. Schools were not randomised into the programme, and
  district adjustment does not make the remaining comparison causal.
```

Both estimates, and the clustered interval, in six lines. A reader who sees only the adjusted number cannot tell whether adjustment mattered, and a reader who sees only the naive interval is being told the result is twice as precise as it is.

2.7 What comes next

Everything so far has had a continuous outcome. The next lesson takes an outcome that is yes or no — did this referral reach a service — and meets the number this audience misreads more reliably than any other.

Part II

Outcomes that are yes or no

CHAPTER 3

An odds ratio of 0.43 for a risk ratio of 0.58

Lesson 3 of 8 · 150 min

Cases reporting a disability complete a referral at 26.4% against 45.5%. That is 58% as likely, and the logistic model prints 0.43. The two numbers describe the same data, only one of them is what a reader thinks they are reading, and the gap widens exactly when the outcome is common.

3.1 Fit it, and read what it prints

```python
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf

referrals = pd.read_csv("protection-referrals-2024.v1.csv")
referrals["disability"] = referrals["disability_reported"].map(
    {"true": 1, "Yes": 1, "false": 0, "No": 0})

consenting = referrals[referrals["consent_to_refer"]].copy()
consenting["completed"] = (consenting["referral_accepted"]
                           & consenting["days_to_first_service"].notna()).astype(int)
d = consenting.dropna(subset=["disability", "case_category", "age_band",
                              "sex", "service_requested", "admin1"])

crude = smf.logit("completed ~ disability", data=d).fit(disp=0)
print(np.exp(crude.params).round(3))
print(np.exp(crude.conf_int()).round(3))
```

```r
library(dplyr)

crude <- glm(completed ~ disability, data = d, family = binomial())
exp(cbind(OR = coef(crude), confint(crude)))
```

Odds ratio 0.430, 95% CI 0.308 to 0.601.
Now compute what the data plainly say.

```python
rates = d.groupby("disability")["completed"].agg(["sum", "size", "mean"])
print(rates.round(4))
p1, p0 = rates.loc[1, "mean"], rates.loc[0, "mean"]
print(f"risk ratio        {p1 / p0:.3f}")
print(f"risk difference   {p1 - p0:+.4f}")
print(f"odds ratio        {(p1/(1-p1)) / (p0/(1-p0)):.3f}")
```

```r
d |> summarise(k = sum(completed), n = n(), rate = mean(completed), .by = disability)
```

Group	Completed	n	Rate
Disability reported	52	197	**26.4%**
Not reported	629	1,384	**45.5%**

Measure	Value	Reads as
Risk ratio	**0.581**	58% as likely to complete
Risk difference	**−19.1 points**	19 fewer completions per 100 cases
Odds ratio	**0.430**	—

The model printed 0.430 and the answer is 0.581. Both are correct; they are answers to different questions, and only one of them is the question anyone asked.

3.2 Why they differ, and when it gets worse

An odds is `p / (1 − p)`. When `p` is small the denominator is near 1 and odds are close to risks, so the two ratios nearly agree. When `p` is large they do not.

```
for p0 in (0.02, 0.10, 0.25, 0.45, 0.70):
    p1 = 0.6 * p0                       # a true risk ratio of 0.6 throughout
    odds = (p1 / (1 - p1)) / (p0 / (1 - p0))
    print(f"baseline {p0:5.0%}   risk ratio 0.60   odds ratio {odds:.3f}")


# One true risk ratio, five baselines, five different odds ratios.
```

Baseline risk	True risk ratio	Odds ratio
2%	0.60	0.59
10%	0.60	0.57
25%	0.60	0.53
45%	**0.60**	**0.45**
70%	0.60	0.31

The odds ratio is not a fixed distortion of the risk ratio — it depends on the baseline. At a 2% outcome the two are interchangeable, which is why the odds ratio survives in epidemiology, where outcomes are rare.

Programme outcomes are not rare. Referral completion is 43%, attendance is 88%, enrolment is 97%, and at those baselines the odds ratio is a long way from what a reader will take it to mean. This is not a subtlety — it is the ordinary case in this work.

3.3 The trap that catches analysts, not just readers

Adding covariates changes an odds ratio *even when nothing is confounded*, and this surprises people who have only worked with linear models.

```
adjusted = smf.logit(
    "completed ~ disability + case_category + age_band + sex"
    " + service_requested + admin1", data=d).fit(disp=0)
print(f"crude OR    {np.exp(crude.params['disability']):.3f}")
print(f"adjusted OR {np.exp(adjusted.params['disability']):.3f}")
```

```
adjusted <- glm(completed ~ disability + case_category + age_band + sex +
                  service_requested + admin1, data = d, family = binomial())
exp(coef(adjusted))["disability"]
```

	Odds ratio	Risk difference
Crude	0.430	−19.05 points
Adjusted	**0.388**	**−19.41 points**

The odds ratio moved by 10% and the risk difference moved by a third of a point. The usual reading — "adjustment revealed a stronger effect" — is wrong here. Nothing was confounded away; the odds ratio simply is not *collapsible*.

A collapsible measure equals the average of the subgroup measures. Risk differences and risk ratios are; odds ratios are not. Add a covariate that predicts the outcome and the conditional odds ratio moves away from 1 even when the covariate is unrelated to the exposure.

So an adjusted odds ratio and a crude odds ratio cannot be compared to judge confounding. That comparison is the standard move in every regression tutorial and it does not work for logistic models. Compare risk differences instead, which the next lesson computes.

3.4 When the odds ratio is the right number

Three cases, and it is worth being precise because the answer is not "never".

A case-control study. Cases and controls are sampled separately, so risks are not estimable at all and the odds ratio is the only ratio the design supports. It is what the measure was invented for.

A rare outcome. Below about 10%, the odds ratio approximates the risk ratio closely enough that the distinction stops mattering. Say which one you computed anyway.

Comparing to published literature that reports odds ratios. Then report both, and lead with the risk difference.

```
def report(label, p1, p0):
    return (f"{label}: {p1:.1%} vs {p0:.1%} -- "
            f"risk difference {p1-p0:+.1%}, risk ratio {p1/p0:.2f}, "
            f"odds ratio {(p1/(1-p1))/(p0/(1-p0)):.2f}")

print(report("Referral completion, disability reported", p1, p0))
```

```
# Print all three. The reader picks; you do not pick for them by omission.
```

3.5 The sentence to write, and three not to

Not this: "Cases reporting a disability were 57% less likely to complete a referral (OR 0.43)." Two errors in one sentence — an odds ratio is not a likelihood, and 57% is not the reduction.

Nor this: "Disability halved the odds of completion." Correct arithmetic, and "halved" will be read as risk by everyone who is not a statistician.

Nor this: "OR 0.43 (95% CI 0.31–0.60, p < 0.001)." Complete, checkable, and it tells a programme manager nothing they can act on.

This: "Cases reporting a disability completed a referral at 26.4% against 45.5% for cases with no disability reported — 19.1 percentage points lower (95% CI −25.7 to −12.4), or 58% of the completion rate. Adjusted for case category, age, sex, service requested and department, the gap is 19.4 points. The adjusted odds ratio is 0.39; it is reported here for comparability with published studies and should not be read as a risk."

The last clause is the one that does the work. An odds ratio in a programme report with no translation beside it will be read as a risk by almost everyone who sees it, including the people who commissioned the analysis.

3.6 Report it whole

```
Referral completion by disability status, 1,581 consenting cases

  Disability reported     26.4%   52 of 197
  Not reported            45.5%   629 of 1,384

  Risk difference   -19.1 points   95% CI -25.7 to -12.4
  Risk ratio          0.58
  Odds ratio          0.43 (crude), 0.39 (adjusted)

  The odds ratio is reported for comparability only. Completion is a common
  outcome (43% overall), so the odds ratio overstates the relative
  difference: 0.43 in odds is 0.58 in risk.

  The adjusted odds ratio differs from the crude one partly because the odds
  ratio is not collapsible, not only because of confounding. The adjusted
  risk difference is -19.4 points, essentially unchanged from crude.

  Fitted on the 1,581 consenting cases with complete covariates. The
  statistics course reported this gap on all 1,638 consenting cases and got
  -19.4 points; the 57-case difference is the complete-case restriction.
```

The last paragraph is two lines and prevents the single most common overclaim — that adjustment "strengthened" a finding when the measure simply moved for arithmetic reasons.

3.7 What comes next

A logistic model prints coefficients on a scale nobody thinks in. The next lesson converts them back into probabilities — the number a programme manager can multiply by a caseload — and shows what the model says about the two gates the referral pathway actually has.

CHAPTER 4

Thirty-eight cases, not an odds ratio

Lesson 4 of 8 · 150 min

One odds ratio of 0.39 produces a 22.7-point gap where completion is common and a 15.2-point gap where it is rare. The predicted probability is what a logistic model actually claims, and multiplied by the caseload it becomes 38 cases that did not reach a service.

4.1 Turn the model back into probabilities

A logistic coefficient lives on the log-odds scale, which nobody thinks in. The fix is one line and it is the same line every time: predict, and average.

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf

model = smf.logit("completed ~ disability + case_category + age_band + sex"
                  " + service_requested + admin1", data=d).fit(disp=0)

everyone_with = d.assign(disability=1)
everyone_without = d.assign(disability=0)

p1 = model.predict(everyone_with).mean()
p0 = model.predict(everyone_without).mean()
print(f"{p1:.4f} vs {p0:.4f}   average marginal effect {p1 - p0:+.4f}")
```

```
library(marginaleffects)

avg_comparisons(model, variables = "disability")
```

26.11% against 45.52%, a difference of −19.41 points.

That is the **average marginal effect**: take every case in the data, ask the model what it predicts if that case reported a disability, ask again if it did not, and average the difference. It is the model's answer in the unit the question was asked in.

Compare it with the crude difference from the last lesson: −19.05 points. The adjusted odds ratio moved from 0.430 to 0.388 and the adjusted *risk difference* barely moved at all. The average marginal effect is the number that behaves the way a reader expects an adjusted estimate to behave.

4.2 Why one odds ratio is several risk differences

This is the property that makes the marginal effect necessary rather than merely convenient.

```
profile = dict(case_category="child-protection", age_band="0-11", sex="f",
               admin1="Artibonite")
rows = []
for service in ["health", "psychosocial", "legal", "livelihood-support"]:
    with_d = pd.DataFrame([{**profile, "service_requested": service, "disability": 1}])
    without = pd.DataFrame([{**profile, "service_requested": service, "disability": 0}])
```

```
    rows.append((service, model.predict(without)[0], model.predict(with_d)[0]))

print(pd.DataFrame(rows, columns=["service", "no disability", "disability"]).round(3))
```

```
predictions(model, newdata = datagrid(service_requested = unique(d$service_requested),
                                      disability = 0:1))
```

Service requested	No disability	Disability reported	Gap
Health	69.2%	46.5%	**−22.7 pts**
Psychosocial	64.5%	41.3%	−23.2 pts
Legal	41.0%	21.2%	−19.8 pts
Livelihood support	28.7%	13.5%	**−15.2 pts**

The odds ratio is 0.39 on every one of those rows. The model was fitted with a single disability coefficient, so by construction the odds ratio does not vary — and the risk difference varies from 15.2 points to 23.2 points anyway.

A constant odds ratio is not a constant effect. Where an outcome is already common, the same odds ratio moves more percentage points; where it is rare, fewer. So "the effect of disability" has no single answer in probability terms unless you say *at what baseline* — which is exactly what the average marginal effect does, by averaging over the baselines the caseload actually has.

4.3 An interval on the marginal effect

The marginal effect is a function of every coefficient, so its interval is not in the coefficient table. Two ways to get one, and the second is the one to reach for.

```
# statsmodels: delta method
margins = model.get_margeff(at="overall")
print(margins.summary())
```

```
avg_comparisons(model, variables = "disability")   # delta method, with CI
```

```
# bootstrap, when the delta method is awkward or the estimator is custom
rng = np.random.default_rng(20260729)
draws = []
for _ in range(400):
    sample = d.sample(len(d), replace=True, random_state=int(rng.integers(1e9)))
    m = smf.logit(model.model.formula, data=sample).fit(disp=0)
    draws.append(m.predict(sample.assign(disability=1)).mean()
                 - m.predict(sample.assign(disability=0)).mean())
print(np.percentile(draws, [2.5, 97.5]).round(4))
```

```
# 400 resamples is enough for a reportable interval; 2,000 for a published one.
```

−19.41 points, 95% CI −26.1 to −13.2 over 400 bootstrap resamples.

Set the seed and say how many resamples. A bootstrap interval that cannot be reproduced is not an interval, and this platform's rule about generated numbers applies to the ones a model produces as much as to the ones a generator does.

4.4 Multiply it by the caseload

This is the sentence a programme manager reads, and the last course established why: an effect has to arrive in the unit the decision is made in.

```
n_disability = int((d["disability"] == 1).sum())
comparator = d[d["disability"] == 0]["completed"].mean()
observed = d[d["disability"] == 1]["completed"].sum()
print(f"{n_disability} cases reporting a disability")
print(f"expected at the comparator rate: {comparator * n_disability:.0f}")
print(f"observed: {int(observed)}  shortfall: {comparator * n_disability - observed:.0f}")
```

```
# Three lines, and it is the only line of the analysis a manager will quote.
```

197 cases reporting a disability. 90 would have completed at the comparator rate; 52 did. Thirty-eight cases short.

Thirty-eight is a number a protection team can act on: it is roughly three cases a month, it names a caseload rather than a percentage, and it can be compared against what a fix would cost. "OR 0.39" cannot do any of those things.

State the arithmetic, not just the result. A reader who can see $197 \times 45.5\% - 52$ can check it; a reader given only "38 cases" has to trust it.

4.5 Four ways this goes wrong

Predicting at the mean instead of averaging the predictions. Setting every covariate to its mean and predicting once gives the effect for a case that does not exist — a case that is 0.6 female and 0.3 legal. Average the predictions over the real cases instead; that is what `at="overall"` and `avg_comparisons` do.

Reporting a marginal effect from a model with an interaction, without saying where. If the model lets the disability effect vary by service, the average is an average over a real difference and the table of profiles is the honest output.

Treating the shortfall as an effect. Thirty-eight cases is what the observed gap corresponds to, not what closing it would deliver. The comparison remains observational.

Extrapolating past the data. The model will happily predict a completion probability for an 80-year-old legal case in a department where none was recorded. Predict on profiles the data contains, and say which ones.

4.6 Report it whole

```
Referral completion by disability status, adjusted

  Logistic model, 1,581 consenting cases with complete covariates.
  Adjusted for case category, age band, sex, service requested, department.

  Average marginal effect   -19.4 points   95% CI -26.1 to -13.2
                                           (400 bootstrap resamples, seed 20260729)
  Predicted completion      26.1% with a disability reported
                            45.5% without

  Applied to the 197 cases reporting a disability, the gap is 38 cases that
  did not reach a service and would have at the comparator rate.
```

```
The gap is not constant: it is 22.7 points for health referrals, where
completion is common, and 15.2 points for livelihood support, where it is
not. The odds ratio (0.39) is the same on both.

Observational. Cases were not randomised, and the model adjusts only for
what the register records.
```

The second-to-last paragraph is the one that stops the average being over-applied. A single number is what gets quoted; the sentence that says where it does and does not hold is what keeps the quoting honest.

4.7 What comes next

Every covariate so far has been chosen because it looked relevant. The next lesson adds one that is relevant, defensible and wrong — a variable on the causal path that removes 42% of the gap and improves every fit statistic while doing it.

Part III

Which covariates belong

CHAPTER 5

The covariate that improves every statistic and destroys the answer

Lesson 5 of 8 · 150 min

Add whether a referral was made and AIC falls by 595, pseudo-R^2 quadruples, and 42.5% of the disability gap disappears. Every model-selection criterion says keep it. Every causal consideration says it must come out, and no fit statistic will ever tell you which.

5.1 A covariate that looks obviously right

The referral pathway has a gate in it. A case is consented, then a referral is *made*, then it is accepted, then a service is reached. `referral_made` is recorded, it is strongly related to completion, and adding it to the model is the natural next step.

```python
import statsmodels.formula.api as smf

BASE = ("completed ~ disability + case_category + age_band + sex"
        " + service_requested + admin1")

adjusted = smf.logit(BASE, data=d).fit(disp=0)
plus_gate = smf.logit(BASE + " + referral_made", data=d).fit(disp=0)

for name, m in [("adjusted", adjusted), ("+ referral_made", plus_gate)]:
    print(f"{name:16} AIC {m.aic:7.1f}   pseudo-R2 {m.prsquared:.3f}")
```

```r
adjusted  <- glm(completed ~ disability + case_category + age_band + sex +
                   service_requested + admin1, data = d, family = binomial())
plus_gate <- update(adjusted, . ~ . + referral_made)
AIC(adjusted, plus_gate)
```

Model	Odds ratio	Average marginal effect	AIC	Pseudo-R^2
Crude	0.430	−19.05 pts	2138.5	0.012
Adjusted	0.388	**−19.41 pts**	1999.6	0.089
+ `referral_made`	0.494	**−11.15 pts**	**1404.2**	**0.365**

AIC improves by 595 points. Pseudo-R^2 quadruples. And 42.5% of the effect vanishes.

Every automatic model-selection procedure ever written would keep that variable. Stepwise selection keeps it, AIC keeps it, cross-validated accuracy keeps it, and all of them are answering a question nobody asked.

5.2 Why it removes the effect

```python
print(d.groupby("disability")["referral_made"].agg(["mean", "size"]).round(4))
```

```
d |> summarise(made = mean(referral_made), n = n(), .by = disability)
```

A referral is made for 71.5% of cases with no disability reported and 53.8% of cases with one.

That is not a nuisance difference to be controlled away — **it is part of the thing being measured.** Cases reporting a disability are less likely to reach a service *partly because a referral is less often made for them in the first place.*

Conditioning on `referral_made` asks: among cases where a referral was made, is there still a gap? The answer is yes, 11.2 points — which is a real and useful number about the *acceptance* stage. It is not the effect of disability on reaching a service, and reporting it as though it were understates that effect by 42%.

A covariate on the causal path between exposure and outcome is a mediator, and adjusting for it removes exactly the part of the effect that runs through it.

5.3 Four kinds of covariate, and what each does

The decision is about causal structure, not about statistics, and it has to be made before the model is fitted.

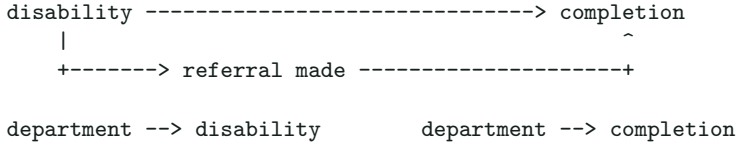
Kind	Sits	Adjust?	If you get it wrong
Confounder	Causes both exposure and outcome	**Yes**	Biased estimate, either direc
Mediator	Between exposure and outcome	**No** (for a total effect)	Effect understated
Collider	Caused by both exposure and outcome	**Never**	Bias created where none exi
Competing cause	Causes only the outcome	Optional	Precision only

Only the first row is a reason to add a variable. The fourth is a reason you may add one, and it buys a smaller standard error rather than a different estimate. The middle two are reasons not to.

A variable's kind is not visible in the data. Confounder, mediator and collider all produce a coefficient that moves when you adjust, and all three improve fit if they predict the outcome. **The only way to tell them apart is to know which came first and what caused what** — which is why the diagram gets drawn before the model is fitted.

5.4 Drawing it before fitting

Three arrows, drawn from what you know about how the pathway works.



Department is a confounder — where a case arises affects both who is registered with a disability and how well the service system works — so it belongs in the model. **Referral made is a mediator** — disability affects it and it affects completion — so it does not.

```
TOTAL_EFFECT = ["disability", "case_category", "age_band", "sex",
                "service_requested", "admin1"]        # confounders only
MECHANISM = TOTAL_EFFECT + ["referral_made"]          # a different question
```

```
# Two named model specifications, two questions, both legitimate.
```

Name the models after the question rather than after the variables. A model called `MECHANISM` will not accidentally be reported as the total effect, and a reviewer can see which one was intended.

5.5 The collider, which is worse

A mediator understates a real effect. A collider manufactures one that is not there, and the mechanism is subtler.

Restricting to closed cases is the most common way it happens here, because a closed case is one where *something* was resolved — which is downstream of both disability and completion.

```python
closed = d[d["case_status"].isin(["closed-resolved", "closed-lost-contact"])]
print(f"closed cases: {len(closed)}")
print(closed.groupby("disability")["completed"].agg(["mean", "size"]).round(4))
```

```r
d |> filter(case_status %in% c("closed-resolved", "closed-lost-contact")) |>
  summarise(rate = mean(completed), n = n(), .by = disability)
```

Sample	n	Gap (average marginal effect)
All consenting cases	1,581	−19.4 points
Closed cases only	**860**	**−21.1 points**

Restricting to closed cases moves the gap to 21.1 points, and the movement is produced by the restriction rather than by anything about disability. Open cases — the ones still in progress — are excluded on a criterion that both variables influence.

Selecting rows is adjusting for a variable. Dropping open cases, keeping only completed forms, analysing only households that answered every question: each is a conditioning step, and each can be a collider. "We restricted the analysis to closed cases for completeness" is a sentence that changes an estimate without anyone noticing it was a modelling decision.

5.6 Report both, and say which is which

Mediation is worth reporting when the mechanism is the programme's question, which here it is: the protection course located the disability gap at referral-making and acceptance, and this decomposition is the quantified version.

```
Referral completion by disability status, decomposition

  Total gap                                  -19.4 points
    of which runs through referral-making    -8.3 points  (42.5%)
    remaining, among referrals made          -11.2 points

  Referrals are made for 53.8% of cases reporting a disability against 71.5%
  of others. Both gates contribute; neither explains the other away.

  The -11.2 figure is conditional on a referral having been made and must not
  be reported as the effect of disability on reaching a service.
```

The last line is the one that keeps the decomposition honest. Both numbers are real, they answer different questions, and the failure mode is not computing the wrong one — it is computing the right one and labelling it as the other.

5.7 What comes next

Every model in this course so far has assumed each row is an independent observation. The next lesson gives the model a term for the thing the rows are grouped inside, and finds three honest routes to the same answer where the naive one was out on its own.

CHAPTER 6

Three honest answers and one that is out on its own

Lesson 6 of 8 · 150 min

Naive OLS, cluster-robust standard errors, a random intercept and school-level aggregation give standard errors of 0.88, 1.46, 1.48 and 1.53 points. Three of those agree and the fourth is the one every default fit produces.

6.1 The same coefficient, four standard errors

The statistics course established that the school feeding programme was assigned to 24 schools and measured on 1,200 children. Here is what each way of handling that produces.

```python
import pandas as pd
import statsmodels.formula.api as smf

naive = smf.ols("rate ~ feeding_programme", data=per_student).fit()
robust = naive.get_robustcov_results(cov_type="cluster",
                                     groups=per_student["school_id"])

mixed = smf.mixedlm("rate ~ feeding_programme", data=per_student,
                    groups=per_student["school_id"]).fit()

by_school = (per_student.groupby(["school_id", "feeding_programme"])["rate"]
             .mean().reset_index())
aggregated = smf.ols("rate ~ feeding_programme", data=by_school).fit()
```

```r
library(lme4); library(sandwich); library(lmtest)

naive      <- lm(rate ~ feeding_programme, data = per_student)
robust     <- coeftest(naive, vcov = vcovCL, cluster = ~school_id)
mixed      <- lmer(rate ~ feeding_programme + (1 | school_id), data = per_student)
aggregated <- lm(rate ~ feeding_programme, data = by_school)
```

Approach	Coefficient	SE	t	n
Naive OLS	+4.93 pts	**0.88**	5.63	1,200
Cluster-robust SE	+4.93 pts	1.46	3.38	1,200
Random intercept	+5.10 pts	1.48	3.44	1,200
School-level OLS	+5.21 pts	1.53	3.41	24

Three of the four agree and one does not. The coefficients span 0.3 points; the three honest standard errors span 0.07; and the naive one is 40% smaller than any of them.

That is the shape to expect. Clustering rarely changes an estimate much and routinely changes its precision a lot, and the three corrections are three routes to the same destination rather than three competing answers.

6.2 What a random intercept adds

Cluster-robust standard errors fix the inference and say nothing about the structure. A random intercept estimates it.

```
print(mixed.summary())
print(f"between-school variance: {float(mixed.cov_re.iloc[0, 0]):.5f}")
print(f"residual variance:       {mixed.scale:.5f}")
```

```
VarCorr(mixed)
```

Model	Between-school variance	Residual variance	ICC
Intercept only	0.00142	0.02042	**0.065**
+ feeding programme	**0.00081**	0.02042	0.038

Feeding explains 42.8% of the between-school variance and none of the within-school variance, which is exactly what a school-level programme should do and is worth checking because it is a way of catching a mis-specified model.

The intercept-only ICC of 0.065 is the same quantity the statistics course computed by hand from mean squares and got 0.060. **Two estimators, two slightly different answers, one conclusion** — about six per cent of the variation in attendance is between schools, which with fifty children per school is enough to quadruple the variance of a naive comparison.

6.3 Choosing among the three

The three corrections are not interchangeable, and the choice is usually decided by the number of clusters and by what varies at which level.

Use	When	Cost
Aggregate to the cluster	Few clusters; the exposure is cluster-level	A big school counts
Cluster-robust SE	Many clusters (30+); you want the individual-level model	Unreliable below ab
Random intercept	You want the variance components, or predictors at both levels	Assumes the random

With 24 clusters, aggregate or fit the random intercept and say which. The cluster-robust sandwich is the standard tool and it is the one that behaves worst here: its asymptotics need more groups than this design has, and it will run without warning you.

When the exposure varies only between clusters, all three converge, which the table above shows. They come apart when a predictor varies *within* clusters — and then the random intercept is doing work the other two cannot.

6.4 Predictors at two levels

This is where a multilevel model stops being a correction and starts being a model.

```
two_level = smf.mixedlm(
    "rate ~ feeding_programme + age_years + disability_reported",
    data=d, groups=d["school_id"]).fit()
print(two_level.summary().tables[1])
```

```
lmer(rate ~ feeding_programme + age_years + disability_reported +
       (1 | school_id), data = d)
```

`feeding_programme` varies only between schools. `age_years` and `disability_reported` vary within them. A single model can carry both, and the random intercept is what lets it: the within-school predictors are estimated from within-school variation and the between-school predictor from between-school variation, without either contaminating the other's standard error.

The failure mode is putting a cluster-level exposure in a model with no cluster term. That is the naive row of the first table, and it is the default that every software package gives you.

6.5 Three levels, and when to stop

Children sit in households, households in villages, villages in districts.

```
# Two grouping levels: households nested within enumeration areas.
smf.mixedlm("outcome ~ x", data=survey,
            groups=survey["ea_id"], re_formula="1").fit()
```

```
lmer(outcome ~ x + (1 | ea_id / household_id), data = survey)
```

Add a level when something is assigned or measured at that level and you have enough units of it. Three levels with six districts at the top will not estimate a district variance worth reporting — the top level needs enough groups for the same reason the multiple-comparisons lesson needed enough tests.

Do not add a level for tidiness. A level with four groups adds a parameter that cannot be estimated and a false sense of having handled something.

6.6 Report it whole

```
Attendance and school feeding, multilevel analysis

  Linear mixed model, 1,200 students in 24 schools.
  rate ~ feeding_programme + (1 | school_id)

  Feeding programme    +5.10 points   95% CI +2.20 to +8.00

  Between-school SD    0.028      Residual SD  0.143
  ICC 0.038 with the programme term; 0.065 without it, so the programme
  accounts for 43% of the between-school variance.

  The naive single-level model gives +4.93 points with a standard error of
  0.88 rather than 1.48. It is not reported: it treats 50 children in one
  school as 50 independent observations.

  Cluster-robust standard errors on the single-level model (+4.93, SE 1.46)
  and school-level aggregation (+5.21, SE 1.53) give the same answer. With
  24 clusters the robust sandwich is at the edge of its assumptions and is
  reported as a check rather than as the headline.

  Observational. Schools were not randomised into the programme.
```

Reporting all three is four lines and it forecloses the obvious challenge — that the result depends on the method. It does not, and showing that is cheaper than arguing it.

6.7 What comes next

This lesson matched the model to how the *programme* was assigned. The next one matches it to how the *sample* was drawn, on a survey where the strata were deliberately unequal — and finds coefficients that cross the significance line in both directions when the weights go in.

Part IV

The design and the fit

CHAPTER 7

Two coefficients cross the line and one crosses back

Lesson 7 of 8 · 150 min

The rural-remote stratum holds 11% of households and a third of the interviews. Fit the model unweighted and livestock livelihoods look like a significant risk factor; weight it and they stop being one, while household size and returnee status start.

7.1 The design the sample was drawn under

```python
import pandas as pd

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

print(frame.groupby("stratum")["households"].sum())
print(survey.groupby("stratum").size())
```

```r
library(dplyr)
frame |> summarise(households = sum(households), .by = stratum)
survey |> count(stratum)
```

Stratum	Households in frame	Share	Interviews	Share
Rural accessible	28,958	51.3%	337	33.8%
Urban	21,270	37.7%	316	31.7%
Rural remote	**6,200**	**11.0%**	**343**	**34.4%**

Rural remote is 11.0% of the population and 34.4% of the sample. That is a deliberate design choice — the survey course explains why you over-sample a small stratum you need estimates for — and it means every unweighted number from this file describes a population that does not exist.

7.2 What the weights do to an estimate

```python
stratum_households = frame.groupby("stratum")["households"].sum()
base_weight = stratum_households / (25 * 14)
interviewed = frame[frame["selected"]].set_index("ea_id")["households_interviewed"]

survey["weight"] = (survey["stratum"].map(base_weight)
                    * 14 / survey["ea_id"].map(interviewed))
assert abs(survey["weight"].sum() - frame["households"].sum()) < 1

unweighted = survey["food_insecure"].mean()
weighted = np.average(survey["food_insecure"], weights=survey["weight"])
print(f"unweighted {unweighted:.1%}   weighted {weighted:.1%}")
```

```r
survey <- survey |>
  mutate(weight = base_weight[stratum] * 14 / households_interviewed)

c(unweighted = mean(survey$food_insecure),
  weighted   = weighted.mean(survey$food_insecure, survey$weight))
```

32.8% unweighted, 29.1% weighted. Rural remote has the highest food insecurity (46.4%), and over-representing it by three times pulls the national figure up by 3.7 points.

A regression fitted on this file without weights inherits exactly that problem, and it is not fixed by adding `stratum` as a covariate — that changes the question from "what is the association in the population" to "what is it within strata". Both are legitimate; only one is what a national estimate means.

7.3 The same model, weighted and not

```python
import statsmodels.api as sm
import statsmodels.formula.api as smf

FORMULA = ("food_insecure ~ improved_water_source + displacement_status"
           " + main_livelihood + household_size")

plain = smf.glm(FORMULA, data=d, family=sm.families.Binomial()).fit(
    cov_type="cluster", cov_kwds={"groups": d["ea_id"]})

weighted = smf.glm(FORMULA, data=d, family=sm.families.Binomial(),
                   freq_weights=d["weight"]).fit(
    cov_type="cluster", cov_kwds={"groups": d["ea_id"]})
```

```r
library(survey)

design <- svydesign(ids = ~ea_id, strata = ~stratum, weights = ~weight,
                    data = survey, nest = TRUE)
svyglm(food_insecure ~ improved_water_source + displacement_status +
         main_livelihood + household_size, design = design,
       family = quasibinomial())
```

Term	Unweighted OR	Weighted OR	Moves?
Improved water source	0.75 (0.55–1.03)	0.74 (0.51–1.08)	—
Displacement: returnee	0.56 (0.25–1.26)	**0.39 (0.15–0.97)**	**becomes significant**
Livelihood: farming	1.55 (1.03–2.34)	1.69 (1.04–2.75)	—
Livelihood: livestock	**1.51 (1.04–2.18)**	1.26 (0.73–2.18)	**stops being significant**
Livelihood: salaried	0.41 (0.22–0.77)	0.45 (0.24–0.87)	—
Household size	1.04 (0.98–1.11)	**1.08 (1.01–1.16)**	**becomes significant**

Three coefficients change which side of the line they sit on, in both directions. Weighting is not a correction that makes everything weaker or everything stronger — it re-weights which households the association is estimated from, and the answer moves wherever those households differ.

Livestock livelihoods are concentrated in the over-sampled remote stratum. Unweighted, they carry a third of the sample's influence; weighted, they carry their real 11%, and the association thins out to nothing.

7.4 Weights, clusters and strata are three separate things

They arrive together and they do three different jobs, and a model can easily get one right and the other two wrong.

Design feature	Fixes	If you omit it
Weights	The estimate	Estimates describe the sample, not the population
Clusters	The standard error	Standard errors too small, as in the last lesson
Strata	The standard error	Standard errors slightly too large — the conservative error

Weights change the coefficient; clusters and strata change its interval. That is the same distinction as adjustment-versus-clustering from lesson 2, arriving from a different direction, and it is the one to hold onto.

```
print(f"clusters: {d['ea_id'].nunique()} enumeration areas")
print(f"strata:   {d['stratum'].nunique()}")
print(f"weights:  {d['weight'].min():.1f} to {d['weight'].max():.1f}")
```

```
# survey::svydesign wants all three; a glm with weights= handles only one.
```

`glm(..., weights=)` is not survey-weighted regression. It applies the weights and computes standard errors as though every observation were independent and the weights were frequencies. Use `survey::svyglm` in R, and in Python pass both `freq_weights` and a cluster covariance — or accept that the interval is wrong and say so.

7.5 When to leave the weights out

Two cases, and neither is "the weights made my result go away".

A within-stratum question. "Among rural remote households, does an improved water source go with lower food insecurity?" is a question about that stratum, and the design weight within a stratum is nearly constant anyway.

A model whose covariates include everything the weights are built from. If stratum and selection probability are fully captured by the covariates, the weighted and unweighted estimates answer the same question and the unweighted one is more precise. **Check it rather than assume it**: fit both, and if they differ materially the covariates did not capture the design.

```
print(f"unweighted {plain.params['main_livelihood[T.livestock]']:.3f}")
print(f"weighted   {weighted.params['main_livelihood[T.livestock]']:.3f}")
```

```
# A large gap between the two is evidence the design is not in the model.
```

A large difference between the weighted and unweighted fit is itself diagnostic, and it is the check to run before deciding the weights are optional.

7.6 Report it whole

```
Food insecurity, household survey 2025
```

```
976 households with complete covariates (of 996 interviewed),
75 enumeration areas, 3 strata.
Survey-weighted logistic regression; weights sum to 56,428 households,
which is the frame total. Standard errors account for clustering by
enumeration area and for stratification.

Weighted prevalence      29.1%   (unweighted 32.8%)

Adjusted odds ratios (weighted):
  Improved water source          0.74   95% CI 0.51 to 1.08
  Returnee household             0.39   95% CI 0.15 to 0.97
  Farming livelihood             1.69   95% CI 1.04 to 2.75
  Livestock livelihood           1.26   95% CI 0.73 to 2.18
  Salaried livelihood            0.45   95% CI 0.24 to 0.87
  Household size, per person     1.08   95% CI 1.01 to 1.16

The unweighted model makes livestock livelihoods significant (OR 1.51,
95% CI 1.04 to 2.18) and household size not. The rural remote stratum is
11% of households and 34% of interviews, and livestock is concentrated
there; the unweighted result is an artefact of the sampling design.

Odds ratios are reported because food insecurity is 29% and the ratios are
moderate; the weighted risk difference for salaried livelihoods is -14.2
points against the observed mix of livelihoods, which is the number in
the summary.
```

The paragraph naming the artefact is what a reviewer will check first, and writing it yourself is better than having it found.

7.7 What comes next

Every model so far has been fitted and read. The last lesson fits one that explains three per cent of its outcome, does not improve on guessing, and is the most useful result in its report — because what it fails to predict is the operational answer.

CHAPTER 8

R^2 of 0.03, and the most useful result in the report

Lesson 8 of 8 · 150 min

Twelve household characteristics explain three per cent of the variation in child MUAC. Targeting screening on the model's worst-predicted 20% finds 31% of the malnourished children, against 20% at random. The model failed, and its failure is the operational answer.

8.1 A model built to answer a targeting question

A programme wants to screen fewer children. If household characteristics predicted which children are malnourished, screening could be targeted at the households most likely to hold them, and the survey carries every variable such a rule would use.

```python
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf

survey = pd.read_csv("household-survey-2025.v1.csv")
d = survey.dropna(subset=["child_muac_mm", "child_age_months", "child_sex",
                          "household_size", "displacement_status",
                          "main_livelihood", "food_insecure",
                          "improved_water_source"])

model = smf.ols(
    "child_muac_mm ~ child_age_months + child_sex + household_size"
    " + displacement_status + main_livelihood + food_insecure"
    " + improved_water_source", data=d).fit()

print(f"n = {int(model.nobs)}   R2 = {model.rsquared:.4f}"
      f"   adjusted R2 = {model.rsquared_adj:.4f}")
print(f"residual SD {np.sqrt(model.scale):.2f}  outcome SD {d['child_muac_mm'].std():.2f}"
    )
```

```r
model <- lm(child_muac_mm ~ child_age_months + child_sex + household_size +
              displacement_status + main_livelihood + food_insecure +
              improved_water_source, data = d)
summary(model)
```

n = 641. R^2 = 0.030, adjusted R^2 = 0.011. Residual SD 15.95 mm against an outcome SD of 16.04 mm.

The model has removed less than one millimetre of the sixteen it started with. One coefficient is distinguishable from zero.

Term	Coefficient	95% CI
Food insecure	**−4.70 mm**	−7.45 to −1.95
Salaried livelihood	−3.49 mm	−8.44 to +1.45
Farming livelihood	−2.76 mm	−6.16 to +0.64
Child age, per month	−0.03 mm	−0.11 to +0.05
Improved water source	−1.09 mm	−3.76 to +1.58

8.2 Do not delete it

The instinct is to drop the model, try more variables, or reach for something non-linear. All three are wrong here, and the reason is that **the question was not "can I model MUAC" — it was "can I target screening".**

```python
d = d.assign(predicted=model.fittedvalues, mam=(d["child_muac_mm"] < 125))
for frac in (0.20, 0.30, 0.50):
    cut = d["predicted"].quantile(frac)
    caught = d.loc[d["predicted"] <= cut, "mam"].sum()
    print(f"screen worst {frac:.0%}: catches {caught}/{d['mam'].sum()}"
          f" = {caught / d['mam'].sum():.1%} of cases")
```

```r
d |> mutate(predicted = fitted(model), mam = child_muac_mm < 125) |>
  arrange(predicted) |>
  summarise(caught = mean(head(mam, n() * 0.2)) * sum(mam))
```

Screening rule	Children screened	MUAC < 125 mm found
Worst-predicted 20%	129	28 of 90 = **31.1%**
Worst-predicted 30%	193	41 of 90 = 45.6%
Worst-predicted 50%	321	54 of 90 = 60.0%
Random 20%	129	**20%** expected

Targeting on the model finds 31% of the malnourished children while screening 20% of them. Screening 20% at random finds 20%. The model is eleven points better than chance and misses seven in ten of the children a programme exists to reach.

That is the finding. Not "the model did not work" — *household characteristics do not identify malnourished children well enough to target screening, so screening has to be blanket.* A programme can act on that sentence, it costs a real amount of money, and it is supported by a model with an R^2 of 0.03.

8.3 Diagnostics that change a decision

Four checks, and each one has a decision attached rather than a threshold.

Fitted values outside the possible range. For a binary outcome fitted with linear regression, this is immediate.

```python
lpm = smf.ols("completed ~ disability + case_category + age_band + sex"
              " + service_requested + admin1", data=referrals).fit()
print(f"fitted values from {lpm.fittedvalues.min():.3f}"
      f" to {lpm.fittedvalues.max():.3f}")
print(f"outside [0, 1]: {((lpm.fittedvalues < 0) | (lpm.fittedvalues > 1)).sum()}")
```

```r
range(fitted(lm(completed ~ ., data = referrals)))
```

Eight of 1,581 fitted probabilities fall below zero, the lowest at −0.060. The logistic model on the same data stays within 0.065 and 0.738. **The decision: use logistic when predictions matter, and the linear probability model when only the average difference does** — the linear model's coefficient is directly a risk difference, which is why it survives in economics.

Residuals against fitted values. Look for a fan shape, which means the spread depends on the level.

```python
import matplotlib.pyplot as plt
plt.scatter(model.fittedvalues, model.resid, s=6)
```

```r
plot(model, which = 1)
```

The decision: heteroscedasticity does not bias the coefficient, it biases the standard error — so the fix is a robust standard error, not a transformed outcome.

Influence. A handful of rows can carry a coefficient.

```python
influence = model.get_influence().cooks_distance[0]
print(f"rows with Cook's D > 4/n: {(influence > 4 / len(d)).sum()}")
```

```r
sum(cooks.distance(model) > 4 / nobs(model))
```

The decision: look at the flagged rows before doing anything. The WASH course's eleven households with a unit error are exactly the kind of row that turns up here, and the answer is to fix the data, not to down-weight the point.

Linearity of a continuous predictor. Fit it in bands and see whether the coefficients step evenly.

```python
d["age_band"] = pd.cut(d["child_age_months"], [0, 12, 24, 36, 48, 60])
print(smf.ols("child_muac_mm ~ C(age_band)", data=d).fit().params.round(2))
```

```r
lm(child_muac_mm ~ cut(child_age_months, c(0, 12, 24, 36, 48, 60)), data = d)
```

The decision: if the bands do not step evenly, the linear term is answering the wrong question, and banding is usually a better answer than a polynomial because the bands are interpretable to the people reading the report.

8.4 What R^2 is and is not

R^2 is the share of variance the model accounts for. In programme data it is routinely small and that is a fact about people, not about the analyst.

It is not a measure of whether a coefficient is right. A randomised trial with a decisive treatment effect can have an R^2 of 0.02, because individual variation swamps a real average difference. Judging a causal estimate by R^2 is a category error.

It is a measure of whether predictions are worth making. That is the use above, and it is the honest one: the targeting question is a prediction question, and R^2 answered it.

Adjusted R^2 penalises added terms, and it went from 0.030 to 0.011 here — the gap between the two is a measure of how many of the twelve variables were paying for themselves. Almost none were.

8.5 Report it whole

```
Predicting child MUAC from household characteristics
```

```
Linear model, 641 children with complete data.
R-squared 0.030, adjusted 0.011. Residual SD 15.95 mm against an outcome
SD of 16.04 mm.

Only food insecurity is distinguishable from zero: -4.70 mm (95% CI -7.45
to -1.95). Household size, displacement status, water source, child sex
and child age are not.

Used as a targeting rule, screening the 20% of children the model ranks
worst would find 31% of the children with MUAC below 125 mm, against 20%
expected from screening 20% at random. Screening half the children would
find 60%.

Recommendation: do not target screening on household characteristics. The
survey's 14.0% prevalence of MUAC below 125 mm is not concentrated in
households that a registration form can identify, and blanket screening
remains the only rule that reaches the caseload.

This model is reported because it does not fit. A negative result about
targeting is a budget decision, and dropping the model would have left the
proposal to assume targeting works.
```

The last paragraph is why the section exists. A model that does not fit is evidence, and the file drawer it usually goes into is where a programme's assumption survives unchallenged.

8.6 What comes next

That is the course. Eight lessons, one idea: **a coefficient is a comparison, and the work is saying which comparison, between which units, adjusted for what, with a standard error that matches how the data arrived.**

The lab puts all four decisions in one analysis. Then *Impact Evaluation Methods* takes on the question this course has deferred in every lesson's last paragraph — which of these comparisons can be written as a cause.

About this edition

This PDF is generated from the same source as the web version of the course, so the two cannot drift. The web edition carries the runnable code, the downloadable datasets and any corrections issued since this file was produced.

> Every dataset referenced in this course is synthetic or fully de-identified. No record traces to a real person.

This course is published in English and in French. The French edition is a professional adaptation using the terminology UNICEF, WHO, WFP, OCHA and national ministries publish in — not a literal translation.

Read and run the course online at data-analysis.cassion.dev/courses/regression-for-programmes

Generated July 29, 2026.