# Longitudinal Data Analysis with R

**A Practical Introduction from Theory to Code**

Dr Clemens Jarnach

2026-04-16

# Table of contents

# Introduction

This workshop material accompanies a one-day Grand Union DTP workshop on Longitudinal Data Analysis, taught by Dr Clemens Jarnach at the University of Oxford.

## Why use panel data analysis?

Much of empirical social science is driven by causal questions: we want to know not just whether two things are related, but whether one actually brings about the other. Using purely cross-sectional data, we can compare different units — people, firms, countries — and test whether they differ according to some treatment or variable of interest. But this approach has a well-known limitation: those units may differ in many other respects, and we are unlikely to observe all of them. The result is that we may mistake confounded correlation for causation.

The gold standard for establishing causation is the randomised controlled trial (RCT). By randomly assigning some units to treatment, we ensure that no unobserved characteristics can be systematically correlated with who receives it — treatment and control groups are, on average, identical in all respects except the intervention itself. However, random assignment is often ethically or practically impossible in the social sciences. Consider studying the effects of education, marriage, or childbirth — the ethics committee will not be sympathetic.

Panel data offer a pragmatic middle ground. Rather than comparing different units at a single point in time, we compare the *same unit to itself* across time: a unit before and after some change, in other words "comparing like with like" (Firebaugh 2008). Stable, unobserved characteristics — the factors that would otherwise confound our estimates — drop out of the analysis, because they are constant within each unit. This is not as secure as an RCT, but it is far more feasible for most social science research questions and substantially more credible than naive cross-sectional comparisons.


Figure 1: The same simulated dataset of four groups (colours) fitted by four different models. By the end of this workshop you will be able to understand, estimate, and critically compare these approaches.

This workshop equips you with the conceptual tools and R skills to work with panel data: to move from describing patterns to making defensible causal claims — or at least to being honest about when you cannot.

## Objectives

This workshop introduces the core concepts and methods of longitudinal and panel data analysis using R, equipping participants with the conceptual foundations and practical skills needed to model change over time, address unobserved heterogeneity, and interpret results in a social-scientific context.

## Syllabus

- Panel data fundamentals: structure, advantages, common sources, and typical challenges.
- Introduction to longitudinal modelling: unobserved heterogeneity, within–between variation, and time-varying vs time-invariant predictors.
- Fixed Effects models: assumptions, estimation, interpretation, and extensions.
- Random Effects models: assumptions, estimation, interpretation, and comparisons with FE.
- Exploring longitudinal data in R
- Preparing longitudinal data in R: data reshaping, cleaning, indexing, and handling missingness.
- Model selection and diagnostics: Hausman test, goodness-of-fit considerations, and robustness checks.
- Applied case studies

## Why use R?

Throughout this workshop we use **R**, which has become one of the richest environments for longitudinal and panel data analysis. The core packages (e.g., `plm`, `fixest`, `lme4`) support the full analytical pipeline: data import and reshaping, model estimation, post-estimation diagnostics, and publication-quality output. R's broader ecosystem (e.g. `tidyverse`) also makes it straightforward to integrate data preparation and visualisation into your analysis workflow.

All workshop code, data, and exercises use R. If you have not yet installed the required packages, please do so:

```
install.packages(c(
  "RColorBrewer", "fixest", "forcats", "haven", "janitor", "lme4",
  "lubridate", "plm", "readxl", "srvyr", "stargazer", "tidyverse",
   ↪  "viridis"
))
```

## Key R Packages

R has a rich ecosystem for longitudinal and panel data analysis, spanning data import, reshaping, model estimation, diagnostics, and visualisation. The core packages for this workshop are:

- **Panel model estimation:** plm — the standard package for fixed effects, random effects, and pooled OLS on panel data; includes the Hausman test and a wide range of diagnostics
- **Fast fixed effects:** fixest — high-performance estimation of models with multiple high-dimensional fixed effects; syntax is concise and output integrates well with `stargazer` and `modelsummary`
- **Mixed effects / multilevel models:** lme4 — fits linear and generalised linear mixed models, useful when random effects structure is substantively motivated
- **Regression tables:** stargazer — produces publication-ready LaTeX and HTML tables from model objects
- **Data import:** haven and readxl — read Stata (`.dta`), SPSS (`.sav`), and Excel files directly into R
- **Data wrangling:** tidyverse and janitor — reshaping between wide and long format, cleaning variable names, and handling missing data

- **Date and time handling:** lubridate — parsing and arithmetic on date/time variables common in longitudinal datasets
- **Colour palettes:** RColorBrewer and viridis — perceptually uniform and accessible colour scales for visualisation

## Recommended Reading

### Core Textbooks

- Croissant, Yves, and Giovanni Millo. 2018. Panel Data Econometrics with R. First edition. Hoboken, NJ: Wiley.
- Angrist, Joshua David, and Jörn-Steffen Pischke. 2008. 'Mostly Harmless Econometrics : An Empiricist's Companion'. In Mostly Harmless Econometrics : An Empiricist's Companion, De gruyter eBooks complete., Princeton, NJ: Princeton University Press.
- Singer, Judith D, and John B Willett. 2003. Applied Longitudinal Data Analysis: Modeling Change and Event Occurrence. 1st edn Oxford u.a: Oxford Univ. Pr.
- Wooldridge, Jeffrey M. 2014. Introduction to Econometrics. Europe, Middle East and Africa edition. Andover.

---

### Online Tutorials and Resources

- **Rüttenauer, Tobias.** *Panel Data Analysis.* https://ruettenauer.github.io/Panel-Data-Analysis/

## Acknowledgement

This course profited a lot from teaching materials by Tobias Rüttenauer; Jeffrey M.Wooldridge et al., Yves Croissant & Giovanni Millo; and Judith Singer & John Willett.

# 1 Which Farmers Are More Productive?

Discovering Unobserved Heterogeneity Through Farm Data

```
library(tidyverse)
library(plm)
library(knitr)
library(RColorBrewer)

# Colour palette (Oxford blue + accents)
pal_farms <- brewer.pal(5, "Set2")
```

---

## 1.1 Step 1 — Set the Scene

> **You are economists hired by a Ministry of Agriculture.**

You have been given data on **five farms**. Your task is to answer one policy question:

> **Does increasing the number of workers raise farm productivity (output)?**

Keep it simple for now: *more workers → more output … right?*

---

## 1.2 Step 2 — The Cross-Sectional Evidence

Below is a snapshot of the five farms in a single year.

```r
set.seed(42)

# Unobserved farm quality (alpha_i) - higher-quality farms also
 ↪  hire more workers
quality <- c(75, 70, 63, 52, 35)        # unobserved: A is best, E
 ↪  is worst
farms   <- LETTERS[1:5]
labour_cs <- c(10, 8, 6, 4, 2)          # correlated with quality!

true_beta <- 2.5                          # true effect of one
 ↪  extra worker

# Output = quality + beta * labour + small noise
set.seed(42)
output_cs <- quality + true_beta * labour_cs + rnorm(5, 0, 1.5)
output_cs <- round(output_cs)

df_cs <- tibble(
  Farm   = farms,
  Labour = labour_cs,
  Output = output_cs
)

kable(
  df_cs,
  col.names = c("Farm", "Labour (workers)", "Output (tonnes)"),
  align = "ccc",
  caption = "Table 1 - Cross-sectional snapshot (Year 2)"
)
```

Table 1.1: Table 1 — Cross-sectional snapshot (Year 2)

| Farm | Labour (workers) | Output (tonnes) |
|---|---|---|
| A | 10 | 102 |
| B | 8 | 89 |
| C | 6 | 79 |
| D | 4 | 63 |
| E | 2 | 41 |

```
ols_cs <- lm(Output ~ Labour, data = df_cs)
beta_cs <- round(coef(ols_cs)["Labour"], 2)

ggplot(df_cs, aes(x = Labour, y = Output, colour = Farm, label =
 ↪  Farm)) +
  geom_smooth(aes(group = 1), method = "lm", se = TRUE,
              colour = "grey40", fill = "grey85", linewidth = 0.8)
               ↪  +
  geom_point(size = 5) +
  geom_text(nudge_y = 1.8, fontface = "bold", size = 4.5) +
  scale_colour_brewer(palette = "Set2") +
  annotate("text", x = 8, y = 68,
           label = paste0("OLS slope = ", beta_cs, " tonnes /
            ↪  worker"),
           colour = "grey30", size = 4, hjust = 0) +
  labs(x = "Labour (workers)", y = "Output (tonnes)",
       title = "Cross-sectional relationship: Labour → Output") +
  theme_minimal(base_size = 13) +
  theme(legend.position = "none")
```

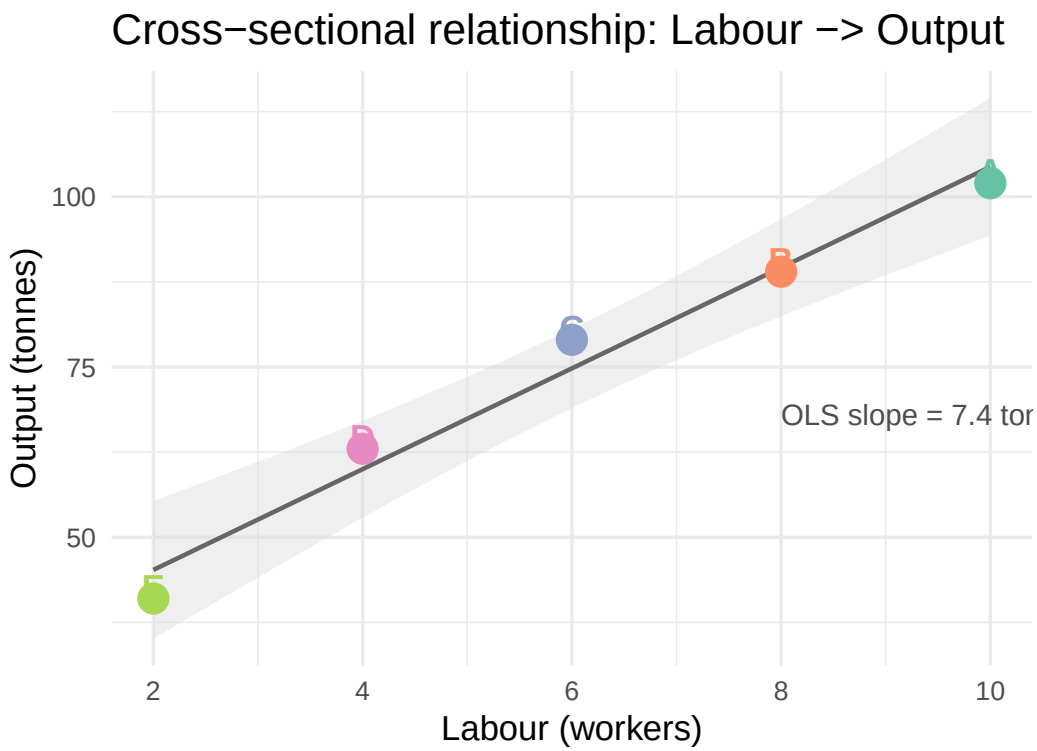


Figure 1.1: Figure 1 — Pooled OLS fit across farms (cross-section)

Discussion — What relationship do you see?

- The slope is **strongly positive**: roughly 7.4 extra tonnes per additional worker.
- A naïve reading suggests labour is highly productive.
- **Policy temptation:** subsidise hiring to boost output.

**But wait — should we trust this number?**

---

## 1.3 Step 3 — Introducing Doubt

Think before you answer

> "Are all farms equally good, independent of how many workers they hire?"

Take 30 seconds to discuss with your neighbour. What might be driving the pattern in Figure 1?

```
df_cs_full <- df_cs |>
  mutate(Quality = quality,
         Quality_label = c("Very high", "High", "Medium", "Low",
          ↪  "Very low"))

kable(
  df_cs_full |> select(Farm, Labour, Quality_label, Output),
  col.names = c("Farm", "Labour (workers)", "Soil Quality
   ↪  (hidden!)", "Output (tonnes)"),
  align = "cccc",
  caption = "Table 2 - The hidden variable revealed"
)
```

Table 1.2: Table 2 — The hidden variable revealed

| Farm | Labour (workers) | Soil Quality (hidden!) | Output (tonnes) |
|---|---|---|---|
| A | 10 | Very high | 102 |
| B | 8 | High | 89 |
| C | 6 | Medium | 79 |

| Farm | Labour (workers) | Soil Quality (hidden!) | Output (tonnes) |
|---|---|---|---|
| D | 4 | Low | 63 |
| E | 2 | Very low | 41 |

```
ggplot(df_cs_full, aes(x = Labour, y = Output, colour =
 ↪  Quality_label, label = Farm)) +
  geom_point(aes(size = Quality), alpha = 0.85) +
  geom_text(nudge_y = 1.8, fontface = "bold", size = 4.5, colour =
   ↪  "grey20") +
  scale_colour_brewer(palette = "YlOrRd", direction = -1,
                      name = "Soil quality") +
  scale_size_continuous(range = c(5, 12), guide = "none") +
  labs(x = "Labour (workers)", y = "Output (tonnes)",
       title = "Larger dots = better soil quality",
       subtitle = "Better farms hire more workers AND produce more
        ↪  - a recipe for bias") +
  theme_minimal(base_size = 13) +
  theme(legend.position = "right")
```

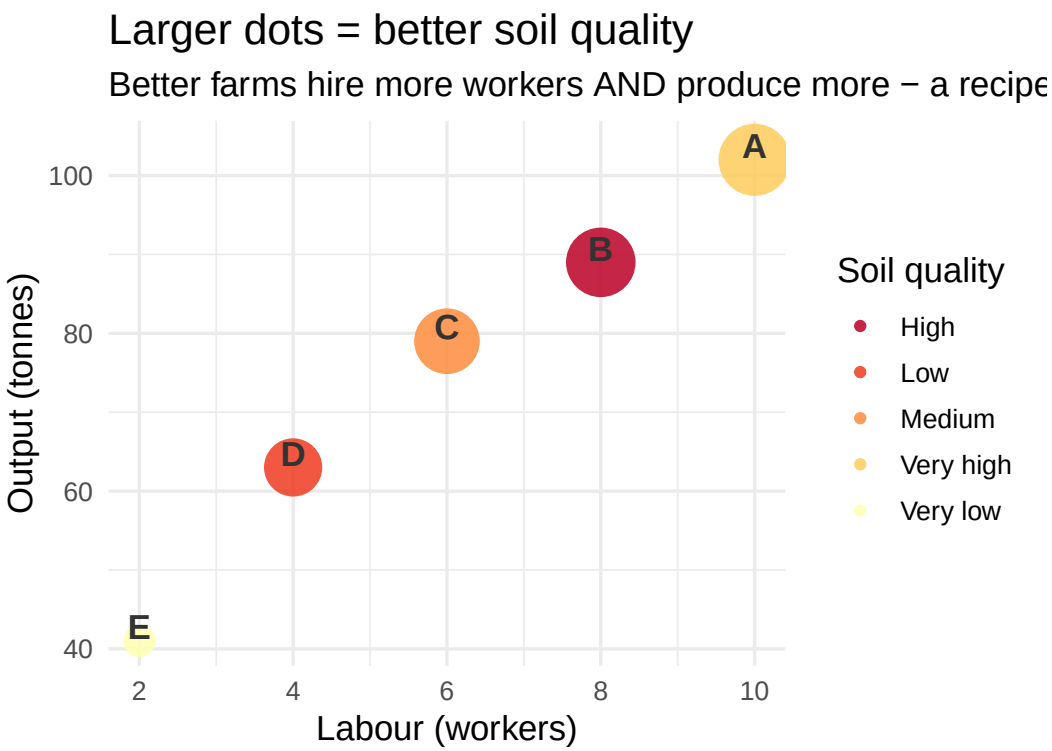


Figure 1.2: Figure 2 — Labour is correlated with unobserved soil quality

⚠ Reveal — The omitted variable bias

The OLS estimate of **7.4** tonnes per worker is **upward biased** because:

$$y_{it} = \alpha + \beta_{\text{OLS}}\, x_{it} + \varepsilon_{it}$$

…but the true model is:

$$y_{it} = \alpha_i + \beta_{\text{true}}\, x_{it} + u_{it}$$

where $\alpha_i$ = soil quality (unobserved). Since better farms hire more workers, $\text{Cov}(x_{it}, \alpha_i) > 0$, so $\hat{\beta}_{\text{OLS}} > \beta_{\text{true}}$.

---

## 1.4 Step 4 — The Panel Data Reveal

Now we track the **same farms over two years**. Each farm changed its workforce between years.

```
set.seed(42)

# Labour levels in year 1 and year 2 (within-farm changes are
 ↪  small)
labour_y1 <- c(6, 5, 4, 3, 1)
labour_y2 <- labour_cs          # same as cross-section year 2

output_y1 <- quality + true_beta * labour_y1 + rnorm(5, 0, 1)
output_y1 <- round(output_y1)

df_panel <- tibble(
  Farm   = rep(farms, 2),
  Year   = rep(c("Year 1", "Year 2"), each = 5),
  Labour = c(labour_y1, labour_y2),
  Output = c(output_y1, output_cs)
)

# Wide format for within-farm differences
df_wide <- df_panel |>
  pivot_wider(names_from = Year, values_from = c(Labour, Output))
   ↪  |>
  mutate(
```

```
    Delta_Labour = `Labour_Year 2` - `Labour_Year 1`,
    Delta_Output = `Output_Year 2` - `Output_Year 1`,
    Within_beta  = round(Delta_Output / Delta_Labour, 2)
  )
```

```
kable(
  df_panel |> arrange(Farm, Year),
  col.names = c("Farm", "Year", "Labour (workers)", "Output
   ↪  (tonnes)"),
  align = "cccc",
  caption = "Table 3 - Panel data: same farms observed over two
   ↪  years"
)
```

Table 1.3: Table 3 — Panel data: same farms observed over two years

| Farm | Year | Labour (workers) | Output (tonnes) |
|---|---|---|---|
| A | Year 1 | 6 | 91 |
| A | Year 2 | 10 | 102 |
| B | Year 1 | 5 | 82 |
| B | Year 2 | 8 | 89 |
| C | Year 1 | 4 | 73 |
| C | Year 2 | 6 | 79 |
| D | Year 1 | 3 | 60 |
| D | Year 2 | 4 | 63 |
| E | Year 1 | 1 | 38 |
| E | Year 2 | 2 | 41 |

```
ggplot(df_panel, aes(x = Labour, y = Output,
                     colour = Farm, group = Farm, label = Farm)) +
  geom_line(linewidth = 1.2, alpha = 0.8) +
  geom_point(aes(shape = Year), size = 4) +
  geom_text(data = df_panel |> filter(Year == "Year 2"),
            nudge_x = 0.2, nudge_y = 0.8,
            fontface = "bold", size = 4) +
  scale_colour_brewer(palette = "Set2") +
  scale_shape_manual(values = c("Year 1" = 16, "Year 2" = 17)) +
  labs(x = "Labour (workers)", y = "Output (tonnes)",
       title = "Within-farm variation: each line connects a farm's
        ↪  two observations",
```

```
       colour = "Farm", shape = "Period") +
  theme_minimal(base_size = 13)
```

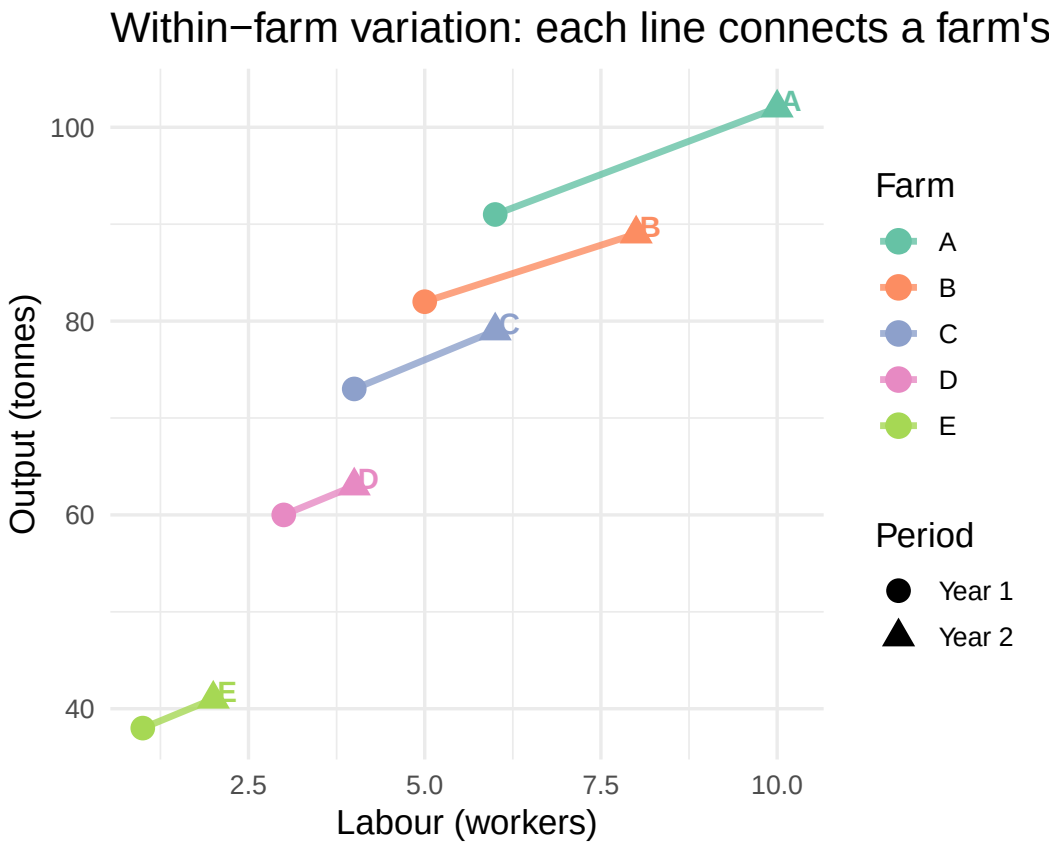


Figure 1.3: Figure 3 — Within-farm changes: each line is one farm over time

```
kable(
  df_wide |>
    select(Farm,
           "Labour (Y1)" = `Labour_Year 1`,
           "Labour (Y2)" = `Labour_Year 2`,
           "Output (Y1)" = `Output_Year 1`,
           "Output (Y2)" = `Output_Year 2`,
           "ΔLabour" = Delta_Labour,
           "ΔOutput" = Delta_Output,
           "ΔOutput/ΔLabour" = Within_beta),
  align = "cccccccc",
```

```
    caption = "Table 4 - Within-farm changes between Year 1 and Year
    ↪  2"
)
```

Table 1.4: Table 4 — Within-farm changes between Year 1 and Year 2

| Farm | Labour (Y1) | Labour (Y2) | Output (Y1) | Output (Y2) | ΔLabour | ΔOut-put | ΔOut-put/ΔLabour |
|---|---|---|---|---|---|---|---|
| A | 6 | 10 | 91 | 102 | 4 | 11 | 2.75 |
| B | 5 | 8 | 82 | 89 | 3 | 7 | 2.33 |
| C | 4 | 6 | 73 | 79 | 2 | 6 | 3.00 |
| D | 3 | 4 | 60 | 63 | 1 | 3 | 3.00 |
| E | 1 | 2 | 38 | 41 | 1 | 3 | 3.00 |

Discussion — What changed?

Compare the **within-farm slope** (Table 4) to the **cross-sectional OLS slope** from Figure 1.

- The within-farm estimates are **much smaller** — close to the true $\beta = 2.5$.
- The cross-sectional estimate of **7.4** was inflated because better farms (high $\alpha_i$) also hire more workers.
- When we look *within* a farm across time, $\alpha_i$ is held constant — quality does not change between Year 1 and Year 2 for the same farm.

"The farm does not change its soil between years. Only its labour does."

---

## 1.5 Step 5 — The Insight: Mundlak (1961)

*What you just discovered is exactly the problem identified by Yair Mundlak in 1961."*

Map the story onto the formal model:

| Story element | Notation |
|---|---|
| Farm output | $y_{it}$ |
| Labour input | $x_{it}$ |

| Story element | Notation |
|---|---|
| Soil quality / skill | $\alpha_i$ |
| Year | $t$ |
| Farm | $i$ |

**The key message:**

> *"We were confusing differences **between farms** with changes **within farms**."*

Between-farm variation is contaminated by $\alpha_i$. Within-farm variation over time is not — because the farm's soil quality is the same in both years.

---

## 1.6 Step 6 — From Intuition to Fixed Effects

```
# Pooled OLS (ignores farm identity)
mod_ols <- lm(Output ~ Labour, data = df_panel)

# Fixed effects (within estimator via plm)
pdata   <- pdata.frame(df_panel, index = c("Farm", "Year"))
mod_fe  <- plm(Output ~ Labour, data = pdata, model = "within")

beta_ols <- round(coef(mod_ols)["Labour"], 3)
beta_fe  <- round(coef(mod_fe)["Labour"], 3)
```

```
tibble(
  Estimator   = c("Pooled OLS", "Fixed Effects (within)"),
  `~ (Labour)` = c(beta_ols, beta_fe),
  `True `     = c(true_beta, true_beta),
  Bias        = c(round(beta_ols - true_beta, 3),
                  round(beta_fe  - true_beta, 3)),
  `Variation used` = c("Between + within farms", "Within farms
   ↪  only")
) |>
  kable(align = "lcccc",
        caption = "Table 5 - Pooled OLS vs Fixed Effects")
```

Table 1.6: Table 5 — Pooled OLS vs Fixed Effects

| Estimator | ˆ (Labour) | True | Bias | Variation used |
|---|---|---|---|---|
| Pooled OLS | 7.291 | 2.5 | 4.791 | Between + within farms |
| Fixed Effects (within) | 2.677 | 2.5 | 0.177 | Within farms only |

```
# Fitted values from FE (add farm means back in for plotting)
df_panel <- df_panel |>
  group_by(Farm) |>
  mutate(
    Labour_dm = Labour - mean(Labour),
    Output_dm = Output - mean(Output),
    Farm_mean_Labour = mean(Labour),
    Farm_mean_Output = mean(Output)
  ) |>
  ungroup()

ggplot(df_panel, aes(x = Labour, y = Output, colour = Farm, group
 ↪  = Farm)) +
  # Farm-level parallel FE lines
  geom_line(aes(y = Farm_mean_Output + beta_fe * (Labour -
   ↪  Farm_mean_Labour)),
            linewidth = 1, linetype = "dashed", alpha = 0.7) +
  # Observed points
  geom_point(aes(shape = Year), size = 4) +
  # Pooled OLS line
  geom_abline(intercept = coef(mod_ols)[1], slope =
   ↪  coef(mod_ols)[2],
              colour = "black", linewidth = 1.1, linetype =
               ↪  "solid") +
  scale_colour_brewer(palette = "Set2") +
  scale_shape_manual(values = c("Year 1" = 16, "Year 2" = 17)) +
  annotate("text", x = 9, y = 76,
           label = paste0("Pooled OLS: ˆ = ", beta_ols),
           colour = "black", fontface = "bold", size = 3.8) +
  annotate("text", x = 9, y = 72,
           label = paste0("Fixed Effects: ˆ = ", beta_fe,
                          " (true   = ", true_beta, ")"),
           colour = "grey30", fontface = "italic", size = 3.8) +
  labs(x = "Labour (workers)", y = "Output (tonnes)",
       title = "Solid line = Pooled OLS  |  Dashed lines =
        ↪  Farm-level FE fits",
```

```
        colour = "Farm", shape = "Period") +
  theme_minimal(base_size = 13)
```

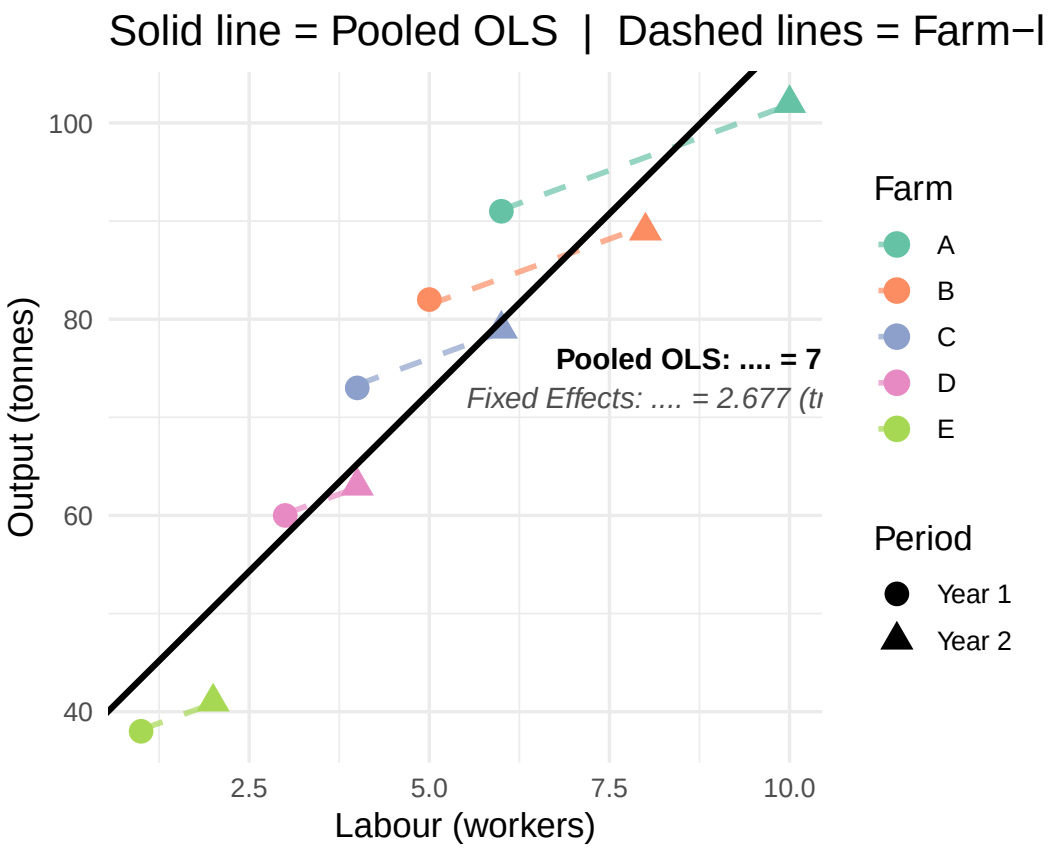


Figure 1.4: Figure 4 — OLS vs within-farm (FE) estimates on the panel data

**Reading the plot**

The **solid black line** is the pooled OLS regression line fitted on all 10 observations (5 farms × 2 years) combined, as if they were independent cross-sectional data points. It ignores farm identity entirely — every observation is treated as a separate, unrelated data point. The slope is therefore driven by **both** between-farm and within-farm variation mixed together. Because high-quality farms have both more labour and more output, the between-farm signal dominates and inflates the slope upward.

The **dashed coloured lines**, by contrast, are the farm-level FE fits — parallel lines with the within slope ($\hat{\beta}_{\text{FE}}$), one per farm, each anchored to that farm's own mean. They use only within-farm variation over time, which is why their slope is

much flatter and closer to the true $\beta$.
The contrast between the steep black line and the shallow dashed lines is the visual payoff of the whole exercise: **same data, very different estimates** — because of what variation each estimator uses.

**!** Policy question — What mistake would we make?

**If we used the cross-sectional (OLS) estimate of 7.291 tonnes per worker:**

- We would **over-subsidise labour** — expecting large output gains from hiring.
- We would **misattribute productivity** to labour inputs, when it is really driven by underlying farm quality ($\alpha_i$).
- The government would transfer resources to farms expecting gains that will not materialise.

**The fixed effects estimate of 2.677 tonnes per worker** — much closer to the true $\beta = 2.5$ — gives the correct policy signal: labour has a real but **much more modest** effect on output.

> *"Fixed effects models do exactly what you just did informally: they hold each unit constant and look only at within-unit variation over time."*

# 2 Panel Data Fundamentals and Fixed Effects

Data Structures, Unobserved Heterogeneity, and the Within Estimator

```
library(tidyverse)
library(plm)
library(knitr)
library(RColorBrewer)
library(broom)
library(lmtest)
library(sandwich)

pal_main  <- brewer.pal(8, "Set2")
pal_rdbu  <- brewer.pal(11, "RdBu")
```

---

## 2.1 Introduction

This chapter develops the conceptual and technical foundations for panel data analysis. We begin by situating panel data within a broader taxonomy of data structures, introduce formal notation, and explain why repeated observations of the same units are so analytically valuable. The central problem — **unobserved heterogeneity** — motivates fixed effects estimation, and we work through its mechanics step by step: the within transformation, the least squares dummy variable approach, and first differencing. The chapter closes with implementation in R and a frank account of the limitations every applied researcher must consider.

---

## 2.2 Step 1 — Types of Data Structures

Before introducing panel data, it helps to place it within a broader taxonomy of data structures commonly used in quantitative social science.

**Cross-sectional data** capture a single snapshot of a population at one point in time. Each unit — person, firm, city — appears once. Cross-sectional analysis is straightforward but cannot distinguish between stable, unit-specific differences and genuine causal effects of a treatment or predictor.

**Pure time series data** track a single unit (e.g., a national economy, a market index) across many time points. These data reveal temporal dynamics but cannot generalise across units.

**Independently pooled cross sections** combine multiple cross-sections taken at different points in time, but the samples at each wave are drawn independently — the same individuals are not followed. Pooling increases sample size and supports comparisons across time, but the lack of unit tracking means we cannot control for unobserved individual characteristics.

**Panel data** (also called longitudinal data) follow the *same* units — individuals, firms, schools, cities, countries — across multiple time periods. This dual structure, cross-sectional and temporal, is what makes panel data analytically powerful.

```
tibble(
  `Data type`        = c("Cross-sectional", "Time series",
                         "Pooled cross-sections", "Panel /
                          ↪ longitudinal"),
  `Units ($N$)`      = c("Many", "One", "Many (different)", "Many
   ↪ (same)"),
  `Time periods ($T$)` = c("One", "Many", "Many", "Many"),
  `Same units tracked?` = c("-", "n/a", "No", "Yes"),
  `Controls unobserved heterogeneity?` = c("No", "No", "No", "Yes
   ↪ (FE/RE)")
) |>
  kable(align = "lcccc",
        caption = "Table 1 - Taxonomy of common data structures in
         ↪ quantitative social science")
```

Table 2.1: Table 1 — Taxonomy of common data structures in quantitative social science

| Data type | Units ($N$) | Time periods ($T$) | Same units tracked? | Controls unobserved heterogeneity? |
|---|---|---|---|---|
| Cross-sectional | Many | One | — | No |
| Time series | One | Many | n/a | No |
| Pooled cross-sections | Many (different) | Many | No | No |
| Panel / longitudinal | Many (same) | Many | Yes | Yes (FE/RE) |

```
# Grid visualisation: rows = units (i), columns = time periods (t)
expand_grid(type = factor(c("Cross-sectional", "Time series",
                            "Pooled cross-sections", "Panel"),
                          levels = c("Cross-sectional", "Time
                           ↪  series",
                                     "Pooled cross-sections",
                                      ↪  "Panel")),
            i = 1:5, t = 1:4) |>
  mutate(
    observed = case_when(
      type == "Cross-sectional"    & t == 2              ~ TRUE,
      type == "Time series"        & i == 1              ~ TRUE,
      type == "Pooled cross-sections"                     ~ (i + t)
↪   %% 2 == 0,
      type == "Panel"                                     ~ TRUE,
      TRUE ~ FALSE
    ),
    same_unit = case_when(
      type == "Pooled cross-sections" ~ "Different units each
↪   wave",
      type == "Panel"                 ~ "Same units across waves",
      TRUE ~ NA_character_
    )
  ) |>
  ggplot(aes(x = factor(t), y = factor(i, levels = 5:1),
             fill = observed, colour = observed)) +
  geom_tile(linewidth = 0.5, width = 0.8, height = 0.8) +
  scale_fill_manual(values  = c("TRUE" = "#4DAF4A", "FALSE" =
   ↪  "grey88"),
```

```
                    labels  = c("TRUE" = "Observed", "FALSE" = "Not
                     ↪  observed"),
                    name    = NULL) +
scale_colour_manual(values = c("TRUE" = "#2b8a2b", "FALSE" =
 ↪  "grey70"),
                      guide  = "none") +
facet_wrap(~ type, ncol = 4) +
labs(x = "Time period (t)", y = "Unit (i)",
     title = "Data structures - which cells are observed?") +
theme_minimal(base_size = 12) +
theme(legend.position = "bottom",
      panel.grid = element_blank(),
      axis.text = element_text(size = 9))
```

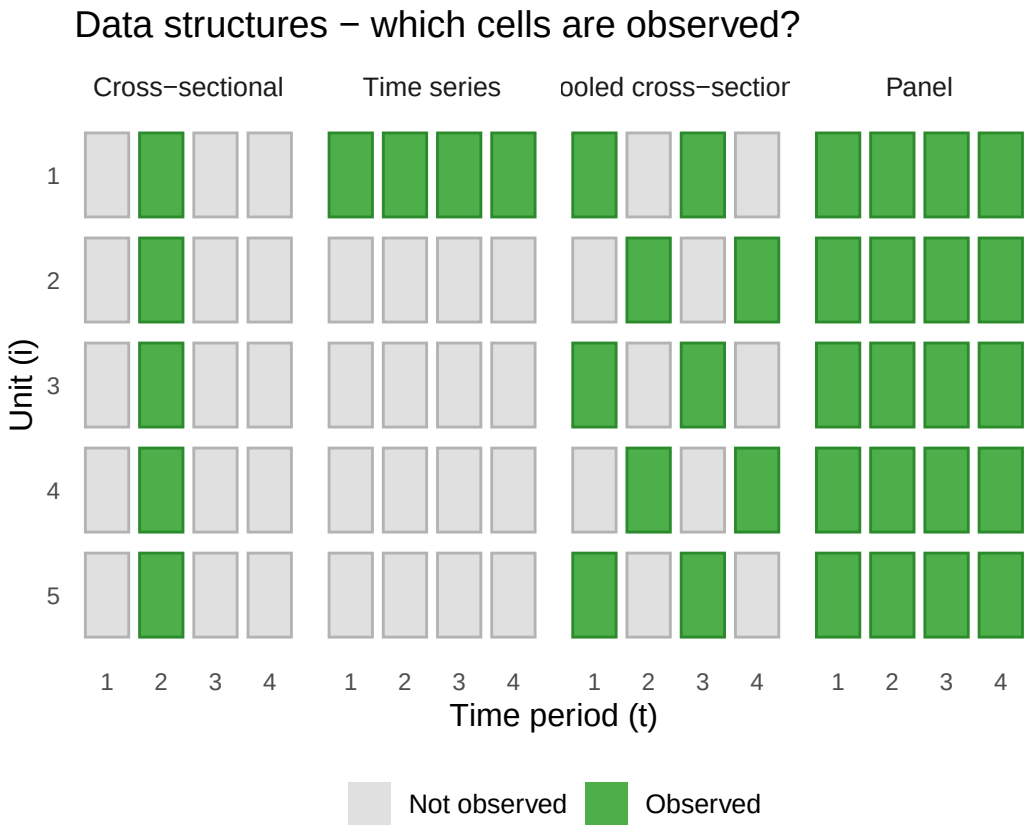


Figure 2.1: Figure 1 — Four data structures: each row is a unit, each column a time period. Filled squares = observed; open squares = not observed or different units.

Key takeaway

The defining feature of panel data is the green column of fully observed cells: **the same units appear at every time period**. This dual structure unlocks within-unit comparisons over time — the engine of fixed effects identification.

---

## 2.3 Step 2 — Panel Data: Structure and Notation

A panel dataset contains observations for $N$ units indexed by $i = 1, \ldots, N$, each observed at $T$ time periods indexed by $t = 1, \ldots, T$. A **balanced panel** has no missing waves: every unit is observed at every time period, yielding $N \times T$ total observations. An **unbalanced panel** has gaps — some units are observed at fewer time points than others.

We write the general unobserved effects model as:

$$y_{it} = \beta_0 + \delta_0 d_t + \mathbf{x}_{it}\beta + a_i + \varepsilon_{it}$$

where:

| Symbol | Meaning |
|---|---|
| $y_{it}$ | Outcome for unit $i$ at time $t$ |
| $\mathbf{x}_{it}$ | $1 \times k$ vector of time-varying regressors |
| $\beta$ | $k \times 1$ parameter vector of interest |
| $d_t$ | Time dummy capturing aggregate trends (e.g., $d_t = 1$ for $t = 2$) |
| $\delta_0$ | Coefficient on time dummy |
| $a_i$ | **Unobserved individual effect** — time-constant, unit-specific |
| $\varepsilon_{it}$ | **Idiosyncratic error** — time-varying, unit-specific |

The term $a_i$ is variously called an unobserved effect, a fixed effect, or unobserved heterogeneity. Whatever the label, its key property is that it does not vary within a unit over time: $a_{it} = a_i$ for all $t$.

```
# Simulate a small balanced panel: 4 countries × 5 years
set.seed(1)
```

```r
panel_demo <- expand_grid(
  country = c("UK", "France", "Germany", "Spain"),
  year    = 2018:2022
) |>
  mutate(
    gdp_growth = round(rnorm(n(), mean = 1.8, sd = 1.2), 2),
    unemployment = round(runif(n(), 4, 12), 1)
  )

kable(panel_demo,
      col.names = c("Country (i)", "Year (t)", "GDP growth (%)",
       ↪  "Unemployment (%)"),
      align = "cccc",
      caption = "Table 2 - A balanced panel: 4 countries × 5 years
       ↪  = 20 observations")
```

Table 2.3: Table 2 — A balanced panel: 4 countries × 5 years = 20 observations

| Country (i) | Year (t) | GDP growth (%) | Unemployment (%) |
|---|---|---|---|
| UK | 2018 | 1.05 | 10.6 |
| UK | 2019 | 2.02 | 9.2 |
| UK | 2020 | 0.80 | 10.3 |
| UK | 2021 | 3.71 | 8.4 |
| UK | 2022 | 2.20 | 8.2 |
| France | 2018 | 0.82 | 10.3 |
| France | 2019 | 2.38 | 4.2 |
| France | 2020 | 2.69 | 7.8 |
| France | 2021 | 2.49 | 9.9 |
| France | 2022 | 1.43 | 9.5 |
| Germany | 2018 | 3.61 | 7.8 |
| Germany | 2019 | 2.27 | 10.9 |
| Germany | 2020 | 1.05 | 7.5 |
| Germany | 2021 | -0.86 | 6.0 |
| Germany | 2022 | 3.15 | 4.6 |
| Spain | 2018 | 1.75 | 4.8 |
| Spain | 2019 | 1.78 | 6.5 |
| Spain | 2020 | 2.93 | 8.1 |
| Spain | 2021 | 2.79 | 9.3 |
| Spain | 2022 | 2.51 | 7.3 |

```
# Introduce missingness to make it unbalanced
panel_unbal <- panel_demo |>
  filter(!(country == "Spain"  & year %in% c(2021, 2022)),
         !(country == "France" & year == 2018))

obs_count <- panel_unbal |>
  count(country, name = "Observations")

kable(obs_count,
      col.names = c("Country", "Observations (of 5 possible)"),
      align = "lc",
      caption = "Table 3 - Unbalanced panel: not every unit
       ↪  observed at every wave")
```

Table 2.4: Table 3 — Unbalanced panel: not every unit observed at every wave

| Country | Observations (of 5 possible) |
|---|---|
| France | 4 |
| Germany | 5 |
| Spain | 3 |
| UK | 5 |

**i** Balanced vs unbalanced panels in R

The `plm` package handles both structures automatically when you declare the panel index via `pdata.frame(df, index = c("unit_id", "time_id"))`. For `feols` from `fixest`, no special declaration is needed — unbalanced panels are handled transparently.

---

## 2.4 Step 3 — Why Panel Data? The Problem of Unobserved Heterogeneity

The core motivation for panel methods is the problem of **unobserved heterogeneity**: units differ in ways we cannot directly measure, and those differences may be correlated with our regressors.

Consider a concrete example. Suppose we want to estimate the effect of local unemployment on city crime rates. A naive cross-sectional regression of crime on unemployment

will likely yield a biased estimate, because cities differ in ways that jointly affect both unemployment and crime — geographic location, historical economic infrastructure, long-standing institutional features — but that are invisible in the data.

The composite error in a pooled OLS model is $v_{it} = a_i + \varepsilon_{it}$. If $a_i$ is correlated with $\mathbf{x}_{it}$, this composite error is also correlated with the regressors. This violates the exogeneity assumption $\text{Cov}(\mathbf{x}_{it}, v_{it}) = 0$, and OLS estimates of $\beta$ become **biased and inconsistent**. The direction and magnitude of the bias depends on the sign and strength of the correlation between $a_i$ and $\mathbf{x}_{it}$. This is sometimes called **heterogeneity bias**, though it is ultimately a form of omitted variable bias: $a_i$ is an omitted confounder.

The key insight that motivates panel methods is this: by observing the same units at multiple time points, we can use *within-unit variation over time* to identify $\beta$, rather than relying on cross-unit comparisons that are contaminated by $a_i$.

```
set.seed(42)
true_beta   <- 0.4     # true effect of unemployment on crime
n_cities    <- 30
n_periods   <- 1       # cross-section only for now

# City-level unobserved factors (urban density, historical
 ↪  disadvantage, etc.)
# Positively correlated with unemployment
city_quality <- rnorm(n_cities, 0, 10)
unem_cs      <- 5 + 0.6 * city_quality + rnorm(n_cities, 0, 2)  #
 ↪  endogenous!
crime_cs     <- 30 + city_quality + true_beta * unem_cs +
 ↪  rnorm(n_cities, 0, 3)

df_ovb <- tibble(city = 1:n_cities, unem = unem_cs, crime =
 ↪  crime_cs,
                 quality = city_quality)

ols_cs  <- lm(crime ~ unem, data = df_ovb)
beta_cs <- round(coef(ols_cs)["unem"], 3)

ggplot(df_ovb, aes(x = unem, y = crime)) +
  geom_point(aes(fill = quality), shape = 21, size = 3.5, alpha =
   ↪  0.85,
             colour = "grey30") +
  geom_smooth(method = "lm", se = TRUE, colour = "#E41A1C",
              fill = "#E41A1C", alpha = 0.15, linewidth = 1.1) +
  geom_abline(intercept = coef(ols_cs)[1] + mean(city_quality),
              slope = true_beta,
```

```
            linetype = "dashed", colour = "grey40", linewidth =
             ↪  0.9) +
  scale_fill_gradientn(colours = rev(pal_rdbu[3:9]),
                       name = "Unobserved\nurban quality") +
  annotate("text", x = max(unem_cs) - 0.5, y = min(crime_cs) + 3,
           label = paste0("\u2014  Cross-sectional OLS slope = ",
            ↪  beta_cs,
                          "  (upward-biased: mixes within- and
                           ↪  between-city variation)"),
           colour = "#E41A1C", fontface = "bold", size = 3.6, hjust
            ↪  = 1) +
  annotate("text", x = max(unem_cs) - 0.5, y = min(crime_cs) - 0.5,
           label = paste0("- - -  True within-city \u03b2 = ",
            ↪  true_beta,
                          "  (unaffected by unobserved city-level
                           ↪  confounders)"),
           colour = "grey40", fontface = "italic", size = 3.6,
            ↪  hjust = 1) +
  labs(x = "Unemployment rate (%)", y = "Crime rate (index)",
       title = "Cross-sectional OLS conflates within- and
        ↪  between-city variation",
       subtitle = "Solid red line: cross-sectional OLS fit (biased)
        ↪  |  Dashed grey line: true causal within-city effect") +
  theme_minimal(base_size = 13)
```

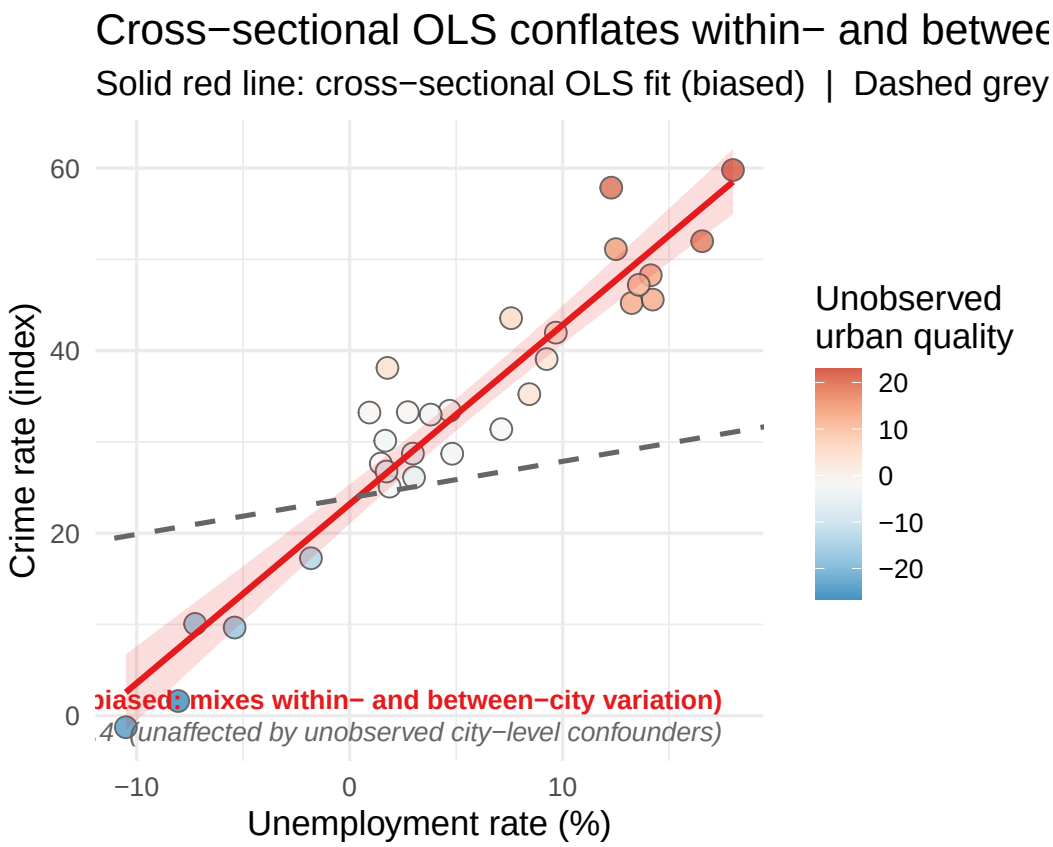


Figure 2.2: Figure 2 — Cross-sectional OLS is upward-biased when unobserved quality is positively correlated with the predictor. The true within-unit relationship (dashed) is much flatter.

**Heterogeneity bias — direction and magnitude**

The OLS omitted variable bias formula tells us:

$$\operatorname{plim} \hat{\beta}_{\text{OLS}} = \beta + \delta_1 \frac{\text{Cov}(x_{it},\, a_i)}{\text{Var}(x_{it})}$$

where $\delta_1$ is the effect of $a_i$ on $y_{it}$. In the crime example:

- $\delta_1 > 0$: cities with higher unobserved disadvantage have more crime.
- $\text{Cov}(\text{unem},\, a_i) > 0$: those same disadvantaged cities also have higher unemployment.

Both terms are positive, so $\hat{\beta}_{\text{OLS}} > \beta_{\text{true}}$. The cross-sectional estimate overstates the causal effect of unemployment.

### 2.4.1 Example: Women's Fertility over Time

*This example is adapted from Wooldridge (2014), Chapter 13.*

A demographer may be interested in the following question: after controlling for education, has the pattern of fertility among women over age 35 changed between 1972 and 1984?

The dataset **FERTIL1**, similar to that used by Sander (1992), comes from the National Opinion Research Center's General Social Survey for the even years from 1972 to 1984. We model the total number of children born to a woman (`kids`). Year dummy variables capture trends in fertility net of education, age, race, and region.

```
set.seed(2024)

# ~1,100 women, surveyed in even years 1972-1984 (pooled
 ↪  cross-section)
n_total <- 1100
years_f <- c(1972, 1974, 1976, 1978, 1980, 1982, 1984)

# Year-specific baseline fertility (declining sharply post-1980)
year_effect <- c(0, -0.03, -0.07, -0.10, -0.18, -0.52, -0.54)
names(year_effect) <- as.character(years_f)

df_fertil <- tibble(
  id    = 1:n_total,
  year  = sample(years_f, n_total, replace = TRUE),
  educ  = round(pmax(0, pmin(20, rnorm(n_total, 12.5, 2.5)))),
  age   = round(runif(n_total, 35, 55)),
  black = rbinom(n_total, 1, 0.12),
  east  = rbinom(n_total, 1, 0.2),
  west  = rbinom(n_total, 1, 0.2)
) |>
  mutate(
    yr_eff = year_effect[as.character(year)],
    # True DGP: kids = f(educ, age, time trend) + noise
    kids_raw = 3.5 - 0.13 * educ +
               0.18 * (age - 35) - 0.004 * (age - 35)^2 +
               0.3 * black + yr_eff +
               rnorm(n_total, 0, 0.8),
    kids = pmax(0, round(kids_raw))
  )
```

```
# Add year dummies and run regression
df_fertil <- df_fertil |>
  mutate(year_f = factor(year, levels = sort(unique(year))))

mod_fertil <- lm(kids ~ year_f + educ + age + I(age^2) + black +
 ↪  east + west,
                 data = df_fertil)
```

```
coef_df <- broom::tidy(mod_fertil, conf.int = TRUE) |>
  filter(str_starts(term, "year_f")) |>
  mutate(year = as.integer(str_remove(term, "year_f")))

# Add 1972 reference (0 by construction)
coef_df <- bind_rows(
  tibble(year = 1972, estimate = 0, conf.low = 0, conf.high = 0),
  coef_df
)

ggplot(coef_df, aes(x = year, y = estimate)) +
  geom_hline(yintercept = 0, linetype = "dashed", colour =
   ↪  "grey60") +
  geom_ribbon(aes(ymin = conf.low, ymax = conf.high),
              fill = "#4DAF4A", alpha = 0.2) +
  geom_line(colour = "#4DAF4A", linewidth = 1.2) +
  geom_point(size = 4, colour = "#2b8a2b") +
  geom_text(aes(label = round(estimate, 2)),
            vjust = -1, size = 3.4, colour = "grey30") +
  scale_x_continuous(breaks = years_f) +
  labs(x = "Survey year", y = "Change in kids (relative to 1972)",
       title = "Fertility declined sharply in the early 1980s",
       subtitle = "Year dummy coefficients from OLS regression
        ↪  (base = 1972)") +
  theme_minimal(base_size = 13)
```

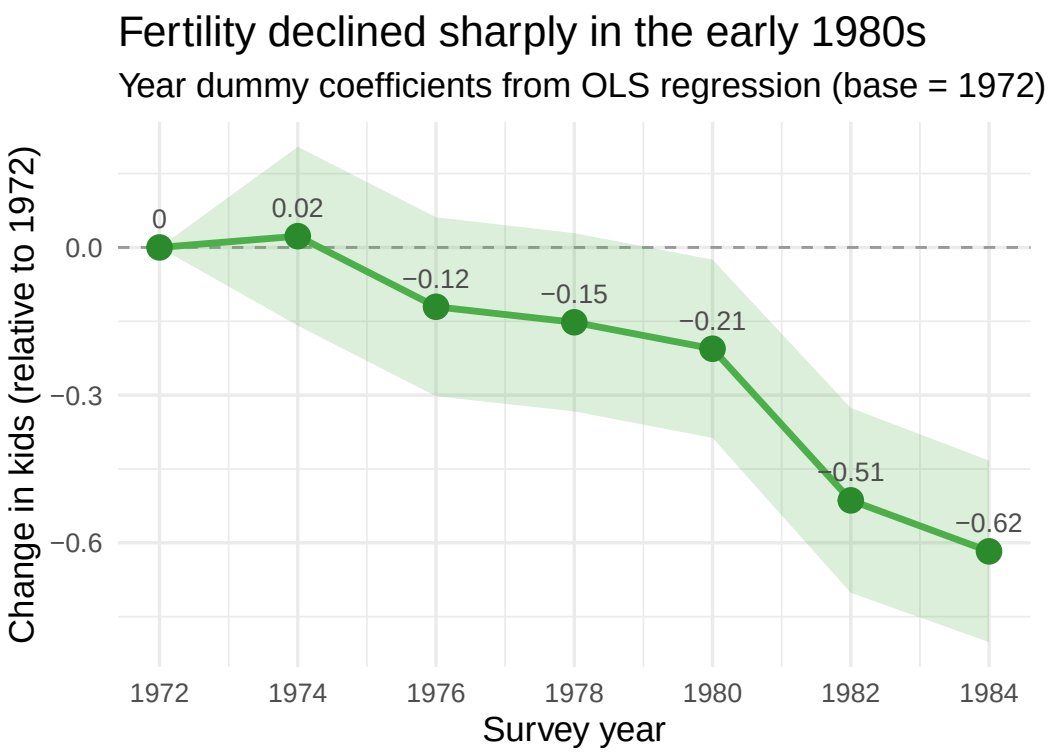


Figure 2.3: Figure 3 — Year dummy coefficients: net change in completed fertility relative to 1972, after controlling for education, age, and demographics.

```
broom::tidy(mod_fertil) |>
  filter(term %in% c("educ", "age", "I(age^2)", "black",
                     "year_f1982", "year_f1984")) |>
  mutate(term = recode(term,
    educ = "Education (years)", age = "Age", `I(age^2)` = "Age²",
    black = "Black (=1)", year_f1982 = "Year = 1982", year_f1984 =
     ↪  "Year = 1984")) |>
  select(Term = term, `^` = estimate, SE = std.error, `p-value` =
   ↪  p.value) |>
  mutate(across(where(is.numeric), \(x) round(x, 4))) |>
  kable(align = "lccc",
        caption = "Table 4 - Selected coefficients from fertility
         ↪  regression")
```

Table 2.5: Table 4 — Selected coefficients from fertility regression

| Term | ˆ | SE | p-value |
|---|---|---|---|
| Year = 1982 | -0.5138 | 0.0957 | 0 |
| Year = 1984 | -0.6174 | 0.0940 | 0 |
| Education (years) | -0.1389 | 0.0104 | 0 |
| Age | 0.5066 | 0.0788 | 0 |
| Age$^2$ | -0.0046 | 0.0009 | 0 |
| Black (=1) | 0.3440 | 0.0793 | 0 |

**Reading the results**

The base year is 1972. The coefficient on `year = 1982` implies that, holding education, age, and other factors fixed, a woman had on average about **0.52 fewer children** in 1982 than in 1972. That is a large drop: 100 comparable women in 1982 are predicted to have roughly 52 fewer children than 100 women in 1972.
Critically, this is **net of education**: rising average education levels (from 12.2 to 13.3 years over this period) would by themselves reduce fertility. The year dummies capture *additional* fertility decline not explained by observable controls — likely reflecting changing social norms, access to contraception, and labour-market opportunities.
This type of analysis — pooled cross-sections with year dummies — underpins the **difference-in-differences** estimator, where one group is "treated" by a policy change while another serves as a control.

---

## 2.5 Step 4 — Pooled OLS (And Why It Fails)

Consider a two-period model ($t = 1, 2$) for 46 US cities, observed in 1982 and 1987. We want to estimate the effect of unemployment on crime:

$$\text{crime\_rate}_{it} = \beta_0 + \delta_0 \, \text{d87}_t + \beta_1 \, \text{unemployment}_{it} + a_i + \varepsilon_{it}$$

where $\text{d87}_t$ is a dummy equal to 1 in 1987 and 0 in 1982.

If we simply pool the two years and run OLS, we must assume that $a_i$ — city-specific, time-invariant factors such as geography, historical culture, or institutional legacies — is uncorrelated with unemployment. That assumption is almost certainly false. Cities with unfavourable structural conditions may persistently exhibit both higher unemployment

and higher crime. Pooled OLS will then conflate the within-city effect of unemployment

```
set.seed(99)
true_beta_crime <- 2.0   # true within-city effect of unemployment
 ↪  on crime
n_cities_c      <- 46

# Unobserved city quality (structural disadvantage) - positively
# correlated with unemployment AND crime
city_fe <- rnorm(n_cities_c, 0, 8)

# Labour input matrix: period 1 (1982) and period 2 (1987)
unem_82 <- 7 + 0.7 * city_fe + rnorm(n_cities_c, 0, 1.5)
unem_87 <- unem_82 + rnorm(n_cities_c, -0.5, 1.2)   # slight drop
 ↪  by 1987

# Crime rate: driven by city FE + unemployment + noise
crime_82 <- 40 + city_fe + true_beta_crime * unem_82 +
 ↪  rnorm(n_cities_c, 0, 4)
crime_87 <- 40 + city_fe + true_beta_crime * unem_87 +
            5 +  # national crime rise 1982-1987 (d87 = 1)
            rnorm(n_cities_c, 0, 4)

df_crime <- tibble(
  city  = rep(1:n_cities_c, 2),
  year  = rep(c(1982, 1987), each = n_cities_c),
  d87   = rep(c(0, 1), each = n_cities_c),
  unem  = c(unem_82, unem_87),
  crime = c(crime_82, crime_87),
  city_fe = rep(city_fe, 2)
)
```

```
mod_pooled <- lm(crime ~ d87 + unem, data = df_crime)
beta_pool  <- round(coef(mod_pooled)["unem"], 3)

tibble(
  Estimator    = "Pooled OLS",
  `~ (unem)`  = beta_pool,
  `True `     = true_beta_crime,
  Bias         = round(beta_pool - true_beta_crime, 3),
  Note         = "Between + within variation (contaminated by a )"
```

```
) |>
  kable(align = "lcccc",
        caption = "Table 5 - Pooled OLS: biased because city
         ↪  heterogeneity is omitted")
```

Table 2.6: Table 5 — Pooled OLS: biased because city heterogeneity is omitted

| Estimator | ˆ (unem) | True | Bias | Note |
|---|---|---|---|---|
| Pooled OLS | 3.513 | 2 | 1.513 | Between + within variation (contaminated by a ) |

```
df_crime_means <- df_crime |>
  group_by(city) |>
  mutate(mean_unem = mean(unem), mean_crime = mean(crime)) |>
  ungroup()

ggplot(df_crime_means, aes(x = unem, y = crime)) +
  # Connect two observations of the same city with thin grey lines
  geom_line(aes(group = city), colour = "grey75", linewidth = 0.5,
   ↪  alpha = 0.6) +
  # Year points
  geom_point(aes(shape = factor(year), colour = city_fe),
             size = 2.5, alpha = 0.85) +
  # Pooled OLS
  geom_smooth(method = "lm", se = FALSE, colour = "#E41A1C",
              linewidth = 1.3, linetype = "solid") +
  # True within-city slope through grand mean
  geom_abline(
    intercept = mean(df_crime$crime) - true_beta_crime *
     ↪  mean(df_crime$unem),
    slope = true_beta_crime,
    colour = "grey30", linewidth = 1.1, linetype = "dashed"
  ) +
  scale_colour_gradientn(colours = rev(pal_rdbu[3:9]),
                         name = "City FE\n(unobserved)") +
  scale_shape_manual(values = c("1982" = 16, "1987" = 17), name =
   ↪  "Year") +
  annotate("text", x = max(df_crime$unem) - 0.3,
           y = min(df_crime$crime) + 4,
           label = paste0("OLS: ˆ = ", beta_pool, "  (biased ↑)"),
```

```
          colour = "#E41A1C", fontface = "bold", size = 3.6, hjust
           ↪  = 1) +
  annotate("text", x = max(df_crime$unem) - 0.3,
          y = min(df_crime$crime) + 0.2,
          label = paste0("True within-city   = ", true_beta_crime),
          colour = "grey30", fontface = "italic", size = 3.6,
           ↪  hjust = 1) +
  labs(x = "Unemployment rate (%)", y = "Crime rate (index)",
       title = "Pooled OLS is biased - city heterogeneity inflates
        ↪  the slope",
       subtitle = "Each pair of points connected by a grey line is
        ↪  the same city in 1982 and 1987") +
  theme_minimal(base_size = 13)
```

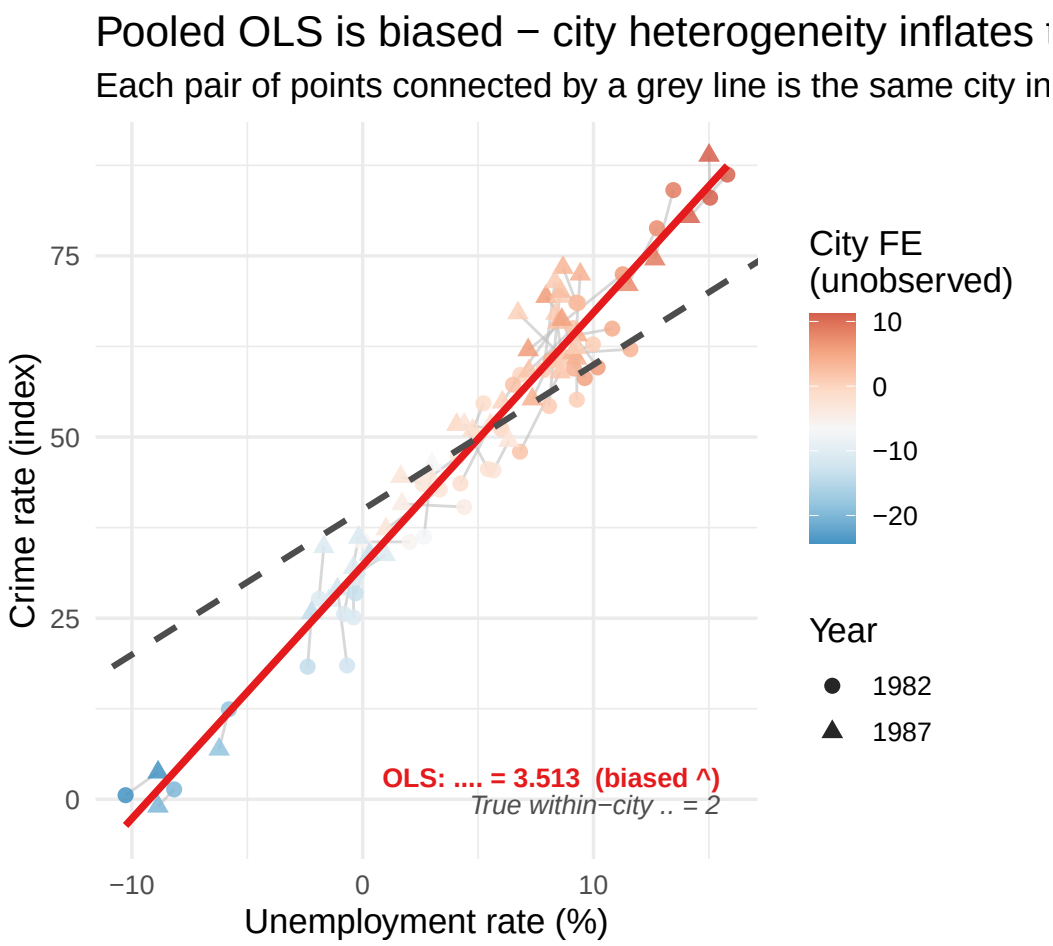


Figure 2.4: Figure 4 — Pooled OLS conflates cross-city differences (driven by unobserved city quality) with within-city changes. The true within-city slope (dashed) is much flatter.

### 2.5.1 Example: Crime Rates and Unemployment (Two-Period Panel)

*This example is adapted from Wooldridge (2014), Chapter 13.*

The dataset **CRIME2** contains crime and unemployment rates for 46 US cities in 1982 and 1987. The two periods need not be adjacent; what matters is that the **same units** are observed at both points in time.

One response to the omitted-variable problem is to add further controls (age distribution, gender distribution, education levels, law enforcement effort), but many relevant factors are difficult to measure directly. Panel data offer a more principled solution. We decompose the unobserved determinants of the outcome into two types: those that are time-constant and those that vary over time.

The time dummy $d_t$ captures aggregate time trends common to all cities. The term $a_i$ captures all **unobserved, time-constant factors** that affect crime. Because it carries no $t$ subscript, it is fixed within each unit over time. The idiosyncratic error $\varepsilon_{it}$ represents factors that are both time-varying and unit-specific.

⚠ Why pooled OLS fails here

Pooled OLS on this data produces a biased estimate because:

1. Cities with historically high disadvantage ($a_i$ large) have both high crime *and* high unemployment.
2. When we pool observations from both years without controlling for city identity, the OLS line is tilted up by this between-city confounding.
3. The coefficient $\widehat{\beta}_{\text{OLS}}$ captures the correlation between unemployment and crime **across cities**, not the causal effect of unemployment *changes within a city*.

Adding more observable controls helps, but cannot fully remove the bias if important determinants of crime — policing culture, economic geography, historical disinvestment — remain unmeasured.

---

## 2.6 Step 5 — The Within Transformation and Fixed Effects Estimation

The fixed effects (FE) estimator eliminates $a_i$ by exploiting the fact that it is time-constant. Two algebraically equivalent approaches achieve this.

To build intuition before the algebra, consider the figures below. Introducing a **binary control variable** (left) partitions the data into discrete groups and compares observations *within* each group. **Fixed effects** (right) generalise this idea: rather than a single binary split, every unit gets its own intercept, so all comparisons are made *within units* over time. The identifying variation comes entirely from how each unit changes, not from how units differ from one another.

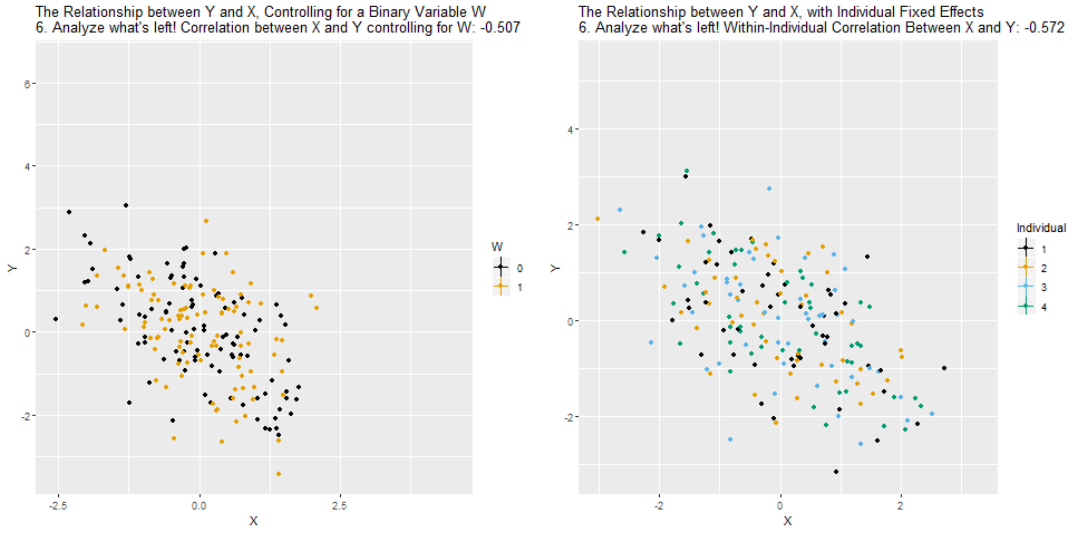


Illustration by Huntington-Klein (2025): Introducing a binary control variable (left) and introducing fixed effects (right)

### 2.6.1 5.1 The Within (De-meaning) Transformation

For each unit $i$, compute the time-mean of every variable:

$$\bar{y}_i = \frac{1}{T}\sum_{t=1}^{T} y_{it}, \qquad \bar{\mathbf{x}}_i = \frac{1}{T}\sum_{t=1}^{T} \mathbf{x}_{it}$$

Subtracting the unit mean from each observation yields the **within-transformed** model:

$$y_{it} - \bar{y}_i = (\mathbf{x}_{it} - \bar{\mathbf{x}}_i)\beta + (\varepsilon_{it} - \bar{\varepsilon}_i)$$

Because $a_i$ is constant, $a_i - \bar{a}_i = 0$: the unobserved effect cancels exactly. OLS on the demeaned data yields the **within estimator**, which is consistent even when $a_i$ and $\mathbf{x}_{it}$ are correlated, provided the idiosyncratic errors satisfy strict exogeneity: $\mathrm{E}(\varepsilon_{it} \mid \mathbf{x}_{i1}, \ldots, \mathbf{x}_{iT}, a_i) = 0$.

```
# Show the demeaning step-by-step for first 5 cities
df_demean_demo <- df_crime |>
  group_by(city) |>
  mutate(
    unem_mean  = mean(unem),
    crime_mean = mean(crime),
```

```
    unem_dm    = unem  - unem_mean,
    crime_dm   = crime - crime_mean
  ) |>
  ungroup()

# Display a few rows
df_demean_demo |>
  filter(city <= 4) |>
  select(city, year, unem, unem_mean, unem_dm, crime, crime_mean,
   ↪  crime_dm) |>
  mutate(across(where(is.numeric), \(x) round(x, 2))) |>
  kable(col.names = c("City", "Year", "Unem", "Ū_i", "Unem − Ū_i",
                      "Crime", "C̄_i", "Crime − C̄_i"),
        align = "cccccccc",
        caption = "Table 6 - De-meaning: each observation expressed
         ↪  as deviation from its city mean")
```

Table 2.7: Table 6 — De-meaning: each observation expressed as deviation from its city mean

| City | Year | Unem | Ū_i | Unem − Ū_i | Crime | C̄_i | Crime − C̄_i |
|---|---|---|---|---|---|---|---|
| 1 | 1982 | 6.82 | 7.69 | -0.87 | 47.97 | 53.46 | -5.49 |
| 2 | 1982 | 8.39 | 8.83 | -0.44 | 61.05 | 60.95 | 0.10 |
| 3 | 1982 | 9.98 | 9.16 | 0.81 | 62.80 | 61.62 | 1.18 |
| 4 | 1982 | 9.25 | 8.97 | 0.29 | 68.54 | 70.96 | -2.42 |
| 1 | 1987 | 8.55 | 7.69 | 0.87 | 58.95 | 53.46 | 5.49 |
| 2 | 1987 | 9.26 | 8.83 | 0.44 | 60.86 | 60.95 | -0.10 |
| 3 | 1987 | 8.35 | 9.16 | -0.81 | 60.44 | 61.62 | -1.18 |
| 4 | 1987 | 8.68 | 8.97 | -0.29 | 73.38 | 70.96 | 2.42 |

```
mod_fe_dm  <- lm(crime_dm ~ unem_dm, data = df_demean_demo)
beta_fe_dm <- round(coef(mod_fe_dm)["unem_dm"], 3)

# Build a label data frame for the FE slope annotation (bottom
 ↪  panel only)
label_df <- data.frame(
  x     = Inf,
  y     = -Inf,
  label = paste0("FE slope = ", beta_fe_dm),
  panel = factor(
    "De-meaned data\n(within-city variation only)",
```

```
    levels = c("Original data\n(between + within variation)",
               "De-meaned data\n(within-city variation only)")
  )
)

# Build a label data frame for the OLS slope annotation (top panel
 ↪ only)
mod_ols    <- lm(crime ~ unem, data = df_crime)
beta_ols   <- round(coef(mod_ols)["unem"], 3)
label_ols_df <- data.frame(
  x     = Inf,
  y     = -Inf,
  label = paste0("OLS slope = ", beta_ols),
  panel = factor(
    "Original data\n(between + within variation)",
    levels = c("Original data\n(between + within variation)",
               "De-meaned data\n(within-city variation only)")
  )
)

# Combine original and de-meaned data into one long frame for
 ↪ faceting
df_plot_combined <- bind_rows(
  df_crime |>
    transmute(x = unem, y = crime,
              panel = "Original data\n(between + within
               ↪ variation)"),
  df_demean_demo |>
    transmute(x = unem_dm, y = crime_dm,
              panel = "De-meaned data\n(within-city variation
               ↪ only)")
) |>
  mutate(panel = factor(panel,
                        levels = c("Original data\n(between +
                         ↪ within variation)",
                                   "De-meaned data\n(within-city
                                    ↪ variation only)")))

ggplot(df_plot_combined, aes(x = x, y = y)) +
  # Reference lines at zero for the de-meaned panel:
  # x = 0 means "unemployment is at this city's average"; y = 0
   ↪ means "crime is at this city's average"
  geom_hline(data = \(d) filter(d, str_detect(panel, "De-meaned")),
```

```
            yintercept = 0, colour = "grey60", linetype =
             ↪  "dashed", linewidth = 0.5) +
geom_vline(data = \(d) filter(d, str_detect(panel, "De-meaned")),
            xintercept = 0, colour = "grey60", linetype =
             ↪  "dashed", linewidth = 0.5) +
geom_point(aes(colour = panel), alpha = 0.55, size = 2) +
# Regression line: pooled OLS on raw data (top) vs FE / within
 ↪  estimator on de-meaned data (bottom)
geom_smooth(method = "lm", se = FALSE,
             aes(colour = panel), linewidth = 1.1) +
scale_colour_manual(values = c(
  "Original data\n(between + within variation)"  = "#E41A1C",
  "De-meaned data\n(within-city variation only)" = "#2b8a2b"
), guide = "none") +
# Annotate slopes in the corner of each panel
geom_text(data = label_ols_df, aes(x = x, y = y, label = label),
          colour = "#E41A1C", fontface = "bold", size = 3.4,
          hjust = 1.05, vjust = -0.5, inherit.aes = FALSE) +
geom_text(data = label_df, aes(x = x, y = y, label = label),
          colour = "#2b8a2b", fontface = "bold", size = 3.4,
          hjust = 1.05, vjust = -0.5, inherit.aes = FALSE) +
facet_wrap(~ panel, ncol = 1, scales = "free") +
labs(
  x     = NULL,
  y     = NULL,
  title = "The within transformation removes between-city
   ↪  differences",
  # Shared axis hint placed via subtitle; actual labels set
   ↪  per-panel below
  subtitle = "Top: unemployment rate (%) vs crime rate - Bottom:
   ↪  deviation from each city's own mean (Δ from city average)"
) +
theme_minimal(base_size = 13) +
theme(plot.subtitle = element_text(size = 10, colour = "grey40"))
```

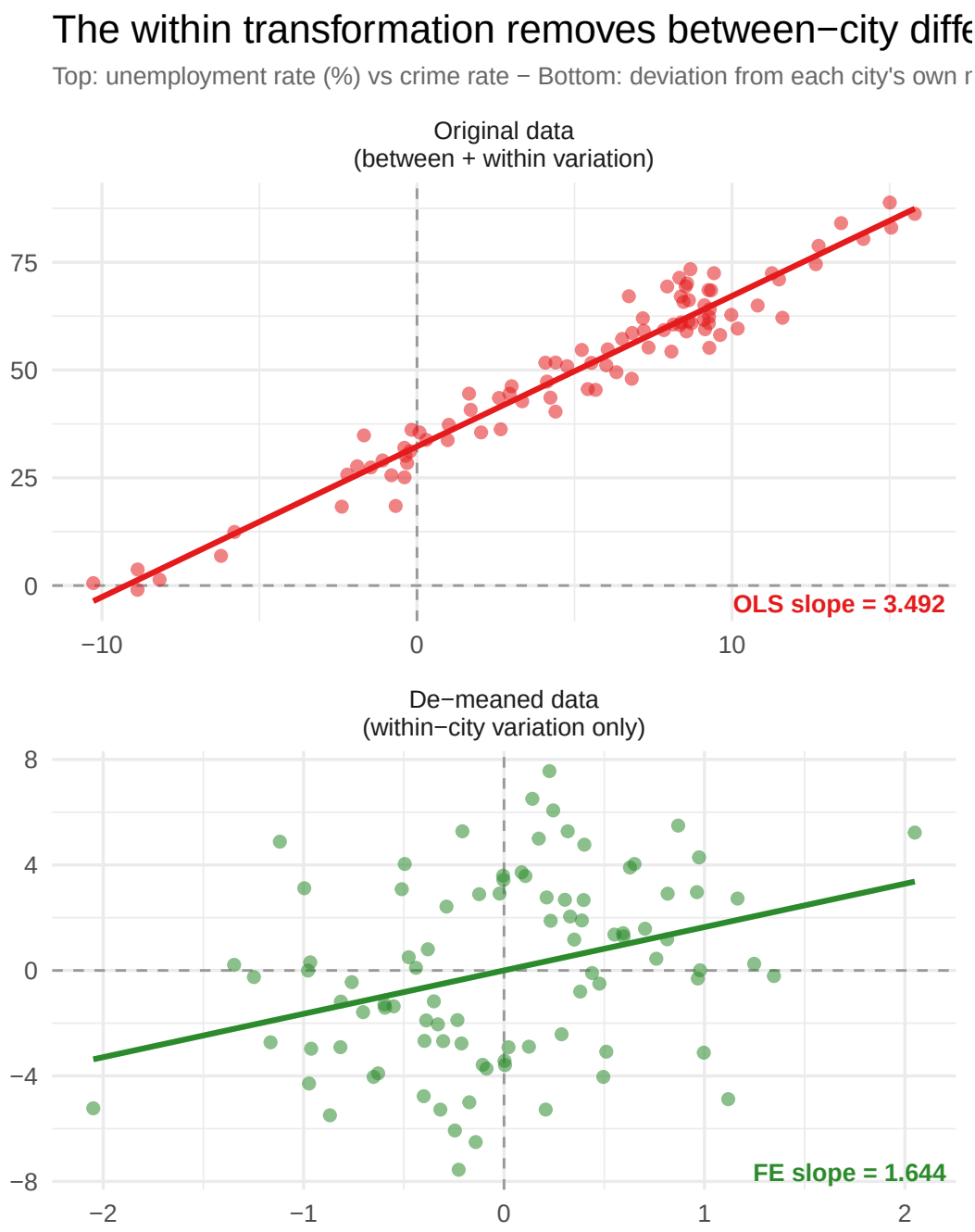


Figure 2.5: Figure 5 — The within transformation. **Top panel:** raw data in which each point is one city-year observation. The OLS regression line (red) conflates differences *between* cities (e.g. high-crime cities may always have high unemployment) with changes *within* cities over time. **Bottom panel:** after de-meaning, each point shows how far that city-year deviates from its own time-average — the origin (0, 0) represents a city's average unemployment and average crime rate. The FE regression line (green) passes through the origin by construction and is estimated purely from within-city year-on-year changes, with all time-constant city characteristics removed.

### 2.6.2 5.2 The Least Squares Dummy Variable (LSDV) Approach

An equivalent approach adds an indicator variable for each unit $i > 1$, giving every unit its own intercept:

$$y_{it} = \sum_{i=1}^{N} \alpha_i D_i + \mathbf{x}_{it}\beta + \varepsilon_{it}$$

where $D_i$ is a dummy equal to 1 for unit $i$ and 0 otherwise. The LSDV and within estimators yield numerically **identical** coefficients $\hat{\beta}$. In practice, the LSDV approach becomes computationally expensive when $N$ is large, because it requires estimating $N-1$ additional parameters and constructing a high-dimensional design matrix. The within transformation is generally preferred.

> **i** Why are they numerically identical?
>
> Adding $N$ unit dummies is equivalent to projecting out (partialling out) all unit-level variation from both $y_{it}$ and $\mathbf{x}_{it}$. The Frisch-Waugh-Lovell theorem guarantees that the coefficient on $\mathbf{x}_{it}$ in the full LSDV regression equals the coefficient from the regression of demeaned $y$ on demeaned $\mathbf{x}$. Same information, same estimator — just different computational routes.

---

## 2.7 Step 6 — First Differencing

For the special case of $T = 2$, a particularly transparent approach is **first differencing**. Subtracting the $t = 1$ equation from the $t = 2$ equation:

$$\Delta y_i = \delta_0 + \beta_1 \Delta \text{unemployment}_i + \Delta \varepsilon_i$$

where $\Delta$ denotes the change from period 1 to period 2. Because $a_i$ is time-constant, $a_{i2} - a_{i1} = 0$ and the unobserved effect drops out exactly. OLS on the differenced equation is unbiased and consistent under the same strict exogeneity assumption.

For $T = 2$, first differencing and the within estimator are **numerically identical** — they produce the same $\hat{\beta}_1$. For $T > 2$, they diverge: first differencing uses consecutive changes, while the within estimator uses deviations from the overall unit mean. The choice between them then depends on assumptions about the serial correlation structure of $\varepsilon_{it}$.

More generally, for $T \geq 2$, the first-difference estimator is:

$$\Delta y_{it} = \Delta \mathbf{x}_{it}\beta + \Delta \varepsilon_{it}, \quad t = 2, \dots, T$$

```
# Reshape to wide, compute differences
df_fd <- df_crime |>
  select(city, year, unem, crime) |>
  pivot_wider(names_from = year, values_from = c(unem, crime)) |>
  mutate(
    delta_unem  = unem_1987  - unem_1982,
    delta_crime = crime_1987 - crime_1982
  )

mod_fd     <- lm(delta_crime ~ delta_unem, data = df_fd)
beta_fd    <- round(coef(mod_fd)["delta_unem"], 3)

# Within (FE) via plm for comparison
pdata_c    <- pdata.frame(df_crime, index = c("city", "year"))
mod_fe_plm <- plm(crime ~ unem + d87, data = pdata_c, model =
 ↪  "within")
beta_fe_c  <- round(coef(mod_fe_plm)["unem"], 3)

tibble(
  Estimator = c("Pooled OLS", "First Difference", "Fixed Effects
   ↪  (within)"),
  `^ (unem)` = c(beta_pool, beta_fd, beta_fe_c),
  `True `    = true_beta_crime,
  Bias       = round(c(beta_pool, beta_fd, beta_fe_c) -
   ↪  true_beta_crime, 3),
  Equivalence = c("-",
                  "= FE for T = 2",
                  "= FD for T = 2")
) |>
  kable(align = "lcccc",
        caption = "Table 7 - First Difference and FE are
         ↪  numerically identical when T = 2")
```

Table 2.8: Table 7 — First Difference and FE are numerically identical when T = 2

| Estimator | ^ (unem) | True | Bias | Equivalence |
|---|---|---|---|---|
| Pooled OLS | 3.513 | 2 | 1.513 | — |
| First Difference | 2.914 | 2 | 0.914 | = FE for T = 2 |
| Fixed Effects (within) | 2.914 | 2 | 0.914 | = FD for T = 2 |

```
ggplot(df_fd, aes(x = delta_unem, y = delta_crime)) +
  geom_hline(yintercept = 0, colour = "grey70", linetype =
   ↪  "dashed") +
  geom_vline(xintercept = 0, colour = "grey70", linetype =
   ↪  "dashed") +
  geom_point(colour = "#4DAF4A", alpha = 0.75, size = 2.5) +
  geom_smooth(method = "lm", se = TRUE, colour = "#2b8a2b",
              fill = "#4DAF4A", alpha = 0.15, linewidth = 1.1) +
  annotate("text", x = max(df_fd$delta_unem) - 0.2,
           y = min(df_fd$delta_crime) + 3,
           label = paste0("FD slope = ", beta_fd,
                          "  ( true  = ", true_beta_crime, ")"),
           colour = "#2b8a2b", fontface = "bold", size = 3.5, hjust
            ↪  = 1) +
  labs(x = "Δ Unemployment (1987 - 1982)", y = "Δ Crime rate (1987
   ↪  - 1982)",
       title = "First-differenced regression: within-city changes
        ↪  only",
       subtitle = "City-level unobserved heterogeneity has been
        ↪  differenced away") +
  theme_minimal(base_size = 13)
```

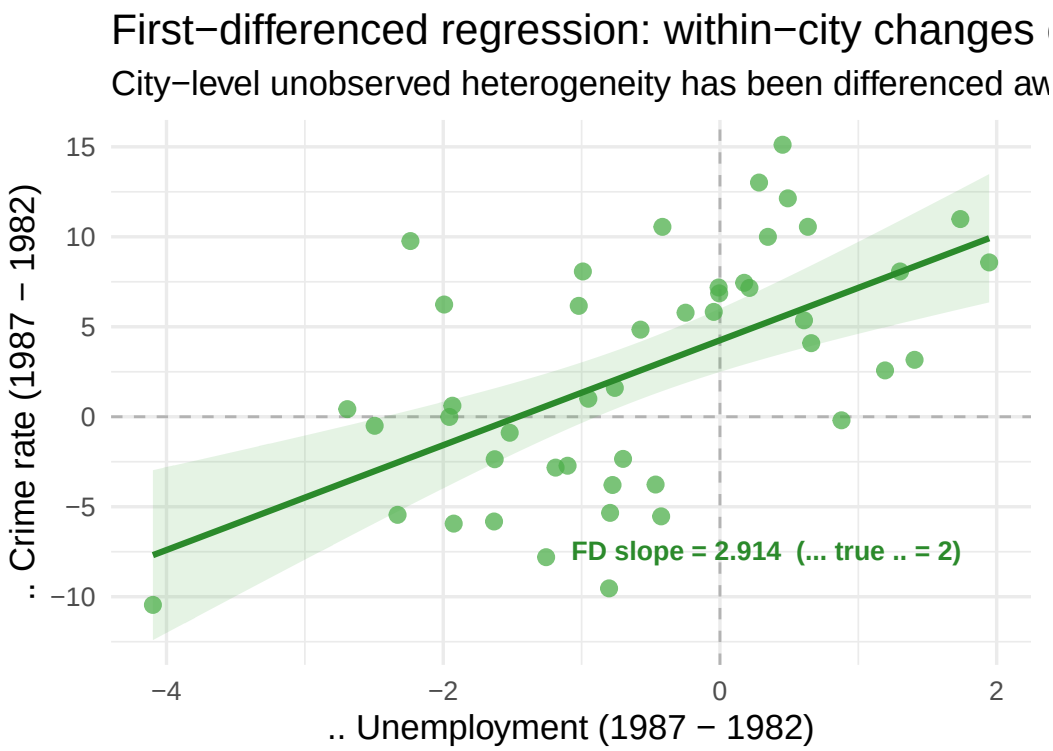


Figure 2.6: Figure 6 — First-differenced regression: changes in unemployment predict changes in crime. Comparing cities to themselves eliminates all time-constant confounders.

When to choose first differencing over the within estimator

Both FD and FE (within) are consistent under strict exogeneity. For $T = 2$ they are identical, so the choice is irrelevant. For $T > 2$:

- **FE** is more efficient when $\varepsilon_{it}$ is serially uncorrelated (standard assumption).
- **FD** is more efficient when $\varepsilon_{it}$ follows a **random walk** — i.e., when shocks are permanent.
- If neither assumption is clearly supported, FE is the default in social science, with clustered standard errors to handle any residual serial correlation.

---

## 2.8 Step 7 — Implementing Fixed Effects in R

The crime rate example provides a clean illustration. The model is:

$$\text{crime\_rate}_{it} = \beta_0 + \delta_0\, \text{d87}_t + \beta_1\, \text{unemployment}_{it} + a_i + \varepsilon_{it}$$

**LSDV via `lm()`**: Each city receives its own intercept through `factor(city)`, which R expands into $N-1$ binary indicator variables (one city serves as the reference category, omitted to avoid perfect collinearity).

```
# LSDV - correct but generates N-1 city dummies
mod_lsdv <- lm(crime ~ d87 + unem + factor(city), data = df_crime)
beta_lsdv <- round(coef(mod_lsdv)["unem"], 3)
cat("LSDV estimate (lm):", beta_lsdv, "\n")
```

```
LSDV estimate (lm): 2.914
```

**Within estimator via `plm()`**: Setting `model = "within"` instructs `plm` to apply the within transformation: it demeaning all variables by unit and estimates $\beta$ from the residual variation. This is equivalent to LSDV but does not construct the full dummy matrix. For very large $N$, the `fixest` package's `feols()` function is even faster and clusters standard errors by default.

```
# Declare panel structure
pdata_fe   <- pdata.frame(df_crime, index = c("city", "year"))

# Within (FE) estimator
mod_fe_plm2  <- plm(crime ~ unem + d87, data = pdata_fe, model =
 ↪  "within")
beta_plm     <- round(coef(mod_fe_plm2)["unem"], 3)
cat("plm (within) estimate:", beta_plm, "\n")
```

```
plm (within) estimate: 2.914
```

```
cat("True beta:            ", true_beta_crime, "\n")
```

```
True beta:             2
```

```
tibble(
  Estimator    = c("Pooled OLS", "LSDV (lm)", "Within (plm)",
   ↪  "First Difference"),
  `~ (unem)`  = c(beta_pool, beta_lsdv, beta_plm, beta_fd),
  `True `     = true_beta_crime,
  Bias         = round(c(beta_pool, beta_lsdv, beta_plm, beta_fd) -
                       true_beta_crime, 3),
```

```
    `Approach`    = c("No FE", "N-1 dummies", "Within transform", "Δ
     ↪  regression")
) |>
  kable(align = "lcccc",
        caption = "Table 8 - All four estimators on the crime
         ↪  panel: LSDV and plm(within) are identical")
```

Table 2.9: Table 8 — All four estimators on the crime panel: LSDV and plm(within) are identical

| Estimator | ˆ (unem) | True | Bias | Approach |
|---|---|---|---|---|
| Pooled OLS | 3.513 | 2 | 1.513 | No FE |
| LSDV (lm) | 2.914 | 2 | 0.914 | N−1 dummies |
| Within (plm) | 2.914 | 2 | 0.914 | Within transform |
| First Difference | 2.914 | 2 | 0.914 | Δ regression |

**i** Key implementation points

- **LSDV and `plm(model = "within")`** produce numerically identical $\hat{\beta}_{\text{unem}}$ — the displayed numbers may differ by rounding only.
- **Time-invariant predictors** (e.g., region, whether a city is coastal) are collinear with the city fixed effects and are automatically dropped. If estimating time-invariant effects matters, use Random Effects or a hybrid model.
- **Standard errors**: OLS standard errors assume independent errors. For panel data, cluster at the unit level using `coeftest(mod, vcov = vcovHC(mod, cluster = "group"))` from the `sandwich` and `lmtest` packages.

```
coeftest(mod_fe_plm2, vcov = vcovHC(mod_fe_plm2, cluster =
 ↪  "group"))
```

```
t test of coefficients:

     Estimate Std. Error t value  Pr(>|t|)
unem  2.91362    0.51952  5.6083 1.266e-06 ***
d87   4.25018    0.80040  5.3101 3.442e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Fixed effects estimation is not without cost. Four limitations deserve particular attention.

## 2.9 Step 8 — Limitations and Practical Considerations

⚠ 1. Loss of between-unit variation

The within transformation discards **all information about differences between units**. If $\mathbf{x}_{it}$ varies little within units over time — common for slowly-changing variables like educational attainment or institutional quality — the within estimator will have high variance and low precision, even if the sample is large in the cross-sectional dimension.
As a rule of thumb: the smaller the within-unit variance relative to the between-unit variance (low *intra-class correlation*), the more imprecise FE estimates become.

⚠ 2. Time-varying confounders

Fixed effects control only for *time-constant* unobservables. If important confounders **change over time** (e.g., a new policing policy enacted mid-panel, a firm-level restructuring), $a_i$ does not absorb them, and $\widehat{\beta}_1$ remains biased. Researchers should always assess whether the key identification assumption — no time-varying confounders correlated with the regressor — is credible in their setting.

⚠ 3. Serial correlation

Observations on the same unit across time are not independent. Standard OLS standard errors, which assume independent errors, will be **too small**, yielding artificially narrow confidence intervals and inflated test statistics. The standard remedy is to cluster standard errors at the unit level. `feols` does this by default; when using `lm()`, apply `coeftest()` with clustered variance-covariance estimates.

⚠ 4. Attrition and data collection challenges

Panel data are more costly to collect than cross-sections. **Attrition** — units dropping out of the panel between waves — is a pervasive practical problem. If attrition is non-random (e.g., more successful firms survive, sicker patients drop out of health studies), the observed panel becomes a selected sample, and estimates may not generalise to the full population.

```
tibble(
  Limitation              = c("Loss of between variation",
```

```
                                "Time-varying confounders",
                                "Serial correlation",
                                "Attrition"),
  `When it matters most`  = c("Slowly-changing predictors (educ,
   ↪  institutions)",
                                "Policy changes mid-panel;
                                 ↪  structural shocks",
                                "Long panels (T   2); persistent
                                 ↪  processes",
                                "Voluntary surveys; firm/patient
                                 ↪  panels"),
  `Practical remedy`      = c("RE or hybrid model; report
   ↪  within-variance ratio",
                                "Include time × covariate
                                 ↪  interactions; DiD design",
                                "Cluster SEs at unit level (default
                                 ↪  in feols)",
                                "Test for non-random attrition; IPW
                                 ↪  correction")
) |>
  kable(align = "lll",
        caption = "Table 9 - FE limitations and practical
         ↪  responses")
```

Table 2.10: Table 9 — FE limitations and practical responses

| Limitation | When it matters most | Practical remedy |
|---|---|---|
| Loss of between variation | Slowly-changing predictors (educ, institutions) | RE or hybrid model; report within-variance ratio |
| Time-varying confounders | Policy changes mid-panel; structural shocks | Include time × covariate interactions; DiD design |
| Serial correlation | Long panels (T 2); persistent processes | Cluster SEs at unit level (default in feols) |
| Attrition | Voluntary surveys; firm/patient panels | Test for non-random attrition; IPW correction |

---

## 2.10 When to Use Panel Methods

Panel methods are most valuable when:

- The research question concerns **causal effects** of a time-varying treatment or predictor on an outcome.
- There are plausible **time-constant confounders** that cannot be directly measured or controlled for.
- The panel has enough **within-unit variation** to identify the parameter of interest.

They are less suitable when:

- The key predictors of interest are **time-invariant** (FE will drop them; consider RE or between-effects models).
- The panel is very short ($T$ is small) and within-unit variation is minimal.
- **Attrition is severe** and likely non-random.

---

## 2.11 Summary

In this chapter you learned:

1. **Panel data** follow the same units across time, enabling within-unit identification that is immune to time-constant unobserved heterogeneity.
2. **Pooled OLS** is biased when $\text{Cov}(a_i, \mathbf{x}_{it}) \neq 0$ — a near-universal condition in observational data.
3. **The within transformation** (de-meaning) eliminates $a_i$ algebraically, yielding the FE estimator. The LSDV approach (unit dummies) is equivalent but computationally expensive for large $N$.
4. **First differencing** achieves the same elimination by taking consecutive changes. For $T = 2$, FD and FE are numerically identical.
5. **In R**, `feols(y ~ x | unit, data)` from `fixest` is the recommended implementation: fast, handles large panels, and clusters standard errors by default.
6. **FE has real costs**: it discards between-unit variation, cannot estimate time-invariant effects, and requires clustered standard errors to be valid. Random Effects — the subject of the next chapter — offers a partial remedy.

# 3 An R Refresher

Essential Programming Skills for Longitudinal Data Analysis

Before we run models, we need to ensure you're comfortable with R itself. This chapter introduces the R programming concepts you will need throughout the workshop. If you are already comfortable with R, treat it as a quick reference. If you are new to R, work through it carefully — the material here underpins everything that follows.

> Running code
>
> All code blocks in this workshop can be run interactively. Open the `.qmd` source file in RStudio, place your cursor inside a code chunk, and press **Ctrl+Enter** (Windows/Linux) or **Cmd+Enter** (Mac) to run a single line, or **Ctrl+Shift+Enter** to run the entire chunk.

---

## 3.1 Basic Operations and Assignment

R can be used as a calculator, but its real power lies in storing and manipulating values. The assignment operator `<-` binds a value to a name in the current environment.

```
1 + 3
```

```
[1] 4
```

```
# arithmetic evaluation
```

```
2^8             # exponentiation
```

```
[1] 256
```

```
17 %% 5          # modulo (remainder)
```

```
[1] 2
```

```
17 %/% 5         # integer division
```

```
[1] 3
```

```
a <- 9           # assignment: bind 9 to the name 'a'
```

```
sqrt(a)          # apply a function to an object
```

```
[1] 3
```

```
b <- sqrt(a)
```

```
a == b           # logical comparison: are they equal?
```

```
[1] FALSE
```

```
a != b           # not equal
```

```
[1] TRUE
```

```
a > 5            # greater than
```

```
[1] TRUE
```

```
ls()             # list all objects in the current environment
```

```
[1] "a"          "b"          "panel_data"
```

> **Note**
>
> R is case-sensitive: `A` and `a` are different objects. Object names can contain letters, digits, `.` and `_`, but must start with a letter or `.`.

---

## 3.2 Data Types

Every value in R has a type. The most common scalar types are:

```
  class(3.14)         # numeric (double)
```

```
[1] "numeric"
```

```
  class(42L)          # integer (note: I used the L suffix to tell R
   ↪  this is an integer)
```

```
[1] "integer"
```

```
  class("hello")      # character
```

```
[1] "character"
```

```
  class(TRUE)         # logical
```

```
[1] "logical"
```

```
  class(2 + 3i)       # complex
```

```
[1] "complex"
```

```
  # Type coercion
  as.numeric("3.14")
```

```
[1] 3.14
```

```
  as.integer(42)
```

```
[1] 42
```

```
  as.character(100)
```

```
[1] "100"
```

```
as.logical(0)       # 0 is FALSE; anything non-zero is TRUE
```

```
[1] FALSE
```

```
is.na(NA)           # test for missing value
```

```
[1] TRUE
```

---

## 3.3 Vectors

A vector is the fundamental data structure in R — an ordered collection of values of the **same type**. Almost all operations in R are vectorised, meaning they apply element-wise without explicit loops.

```
x <- c(1, 3, 5)          # combine values into a vector
y <- c("one", "three", "five")
```

```
# Indexing (1-based)
x[1]                     # first element
```

```
[1] 1
```

```
x[c(1, 3)]               # first and third elements
```

```
[1] 1 5
```

```
x[-2]                    # all except the second
```

```
[1] 1 5
```

```
# Sequences
a <- 1:10
b <- seq(from = 0, to = 1, by = 0.25)
c_rep <- rep(c(1, 2), times = 3)
c_rep
```

```
[1] 1 2 1 2 1 2
```

```
# Vectorised operations - applied element-wise
x * 2
```

```
[1]  2  6 10
```

```
x + c(10, 20, 30)
```

```
[1] 11 23 35
```

```
# Logical operations on vectors
x > 2
```

```
[1] FALSE  TRUE  TRUE
```

```
any(x > 2)                # is any element > 2?
```

```
[1] TRUE
```

```
all(x > 2)                # are all elements > 2?
```

```
[1] FALSE
```

```
which(x > 2)              # indices where condition is TRUE
```

```
[1] 2 3
```

```
# Common summary functions
length(x)
```

```
[1] 3
```

```
sum(x)
```

```
[1] 9
```

```
mean(x)
```

```
[1] 3
```

```
sd(x)
```

```
[1] 2
```

```
min(x); max(x)
```

```
[1] 1
```

```
[1] 5
```

---

## 3.4 Matrices

A matrix is a two-dimensional vector: all elements share the same type, and values are arranged in rows and columns. In longitudinal data analysis, matrices appear in covariance and correlation structures, variance-component decompositions, and balanced panel layouts — understanding matrix indexing is useful for inspecting model objects and constructing derived quantities.

```
m <- matrix(1:25, nrow = 5, ncol = 5)
m
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11   16   21
[2,]    2    7   12   17   22
[3,]    3    8   13   18   23
[4,]    4    9   14   19   24
[5,]    5   10   15   20   25
```

```
# Element access
m[1, 2]                    # row 1, column 2
```

```
[1] 6
```

```
m[1, ]                     # entire first row
```

```
[1]  1  6 11 16 21
```

```
  m[, 2]                    # entire second column
```

```
[1]  6  7  8  9 10
```

```
  m[2:3, 3:5]               # submatrix
```

```
     [,1] [,2] [,3]
[1,]   12   17   22
[2,]   13   18   23
```

```
  # Dimensions
  nrow(m); ncol(m)
```

```
[1] 5
```

```
[1] 5
```

```
  dim(m)
```

```
[1] 5 5
```

```
  # Matrix operations - useful for covariance and correlation
   ↪  matrices in panel diagnostics
  t(m)                      # transpose
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
[4,]   16   17   18   19   20
[5,]   21   22   23   24   25
```

```
  m %*% t(m)                # matrix multiplication (not
   ↪  element-wise!)
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]  855  910  965 1020 1075
[2,]  910  970 1030 1090 1150
```

```
[3,]  965 1030 1095 1160 1225
[4,] 1020 1090 1160 1230 1300
[5,] 1075 1150 1225 1300 1375
```

```
diag(m)                  # diagonal elements
```

```
[1]  1  7 13 19 25
```

```
# Compute a correlation matrix - useful for checking
 ↪  multicollinearity among panel predictors
vars <- data.frame(
  gdp_growth = c(1.2, 2.3, 1.8, 3.1, 0.9, 2.7),
  trade_open  = c(0.4, 0.9, 0.6, 1.1, 0.3, 0.8),
  population  = c(10,  12,  11,  14,  9,   13)
)
cor(vars)                # pairwise correlations between
 ↪  panel-level variables
```

```
            gdp_growth trade_open population
gdp_growth   1.0000000  0.9673915  0.9968896
trade_open   0.9673915  1.0000000  0.9605857
population   0.9968896  0.9605857  1.0000000
```

---

## 3.5 Lists

Lists are R's most flexible container: each element can hold any object of any type or length. Many R functions return lists (model output from `plm`, `lme4`, or `fixest`, etc.), so you need to know how to navigate them.

```
person <- list(
  name    = "Alice",
  age     = 34,
  scores  = c(88, 92, 79),
  active  = TRUE
)
```

```
# Access by name
person$name
```

```
[1] "Alice"
```

```
person[["age"]]
```

```
[1] 34
```

```
# Access by position
person[[3]]               # third element (the scores vector)
```

```
[1] 88 92 79
```

```
person[[3]][2]            # second score
```

```
[1] 92
```

```
# Inspect structure
str(person)
```

```
List of 4
 $ name  : chr "Alice"
 $ age   : num 34
 $ scores: num [1:3] 88 92 79
 $ active: logi TRUE
```

```
length(person)
```

```
[1] 4
```

```
names(person)
```

```
[1] "name"   "age"    "scores" "active"
```

---

## 3.6 Data Frames

A data frame is the standard rectangular data structure in R: a list of vectors of equal length, where each vector is a column. Think of it as a spreadsheet or database table. Longitudinal datasets are naturally represented as data frames, where each row is an observation for a given unit (individual, country, firm) at a given time point.

```
df <- data.frame(
  id     = 1:5,
  name   = c("Alice", "Bob", "Clara", "David", "Eva"),
  degree = c(3, 7, 2, 5, 4),
  active = c(TRUE, TRUE, FALSE, TRUE, FALSE),
  stringsAsFactors = FALSE
)

df
```

```
  id  name degree active
1  1 Alice      3   TRUE
2  2   Bob      7   TRUE
3  3 Clara      2  FALSE
4  4 David      5   TRUE
5  5   Eva      4  FALSE
```

```
# Column access
df$name
```

```
[1] "Alice" "Bob"   "Clara" "David" "Eva"
```

```
df[["degree"]]
```

```
[1] 3 7 2 5 4
```

```
# Row and column indexing
df[1, ]                   # first row
```

```
  id  name degree active
1  1 Alice      3   TRUE
```

```
df[, 3]                   # third column
```

```
[1] 3 7 2 5 4
```

```
df[df$active, ]           # rows where active == TRUE
```

```
  id  name degree active
1  1 Alice      3   TRUE
2  2   Bob      7   TRUE
4  4 David      5   TRUE
```

```
df[df$degree > 3, c("name", "degree")]
```

```
   name degree
2   Bob      7
4 David      5
5   Eva      4
```

```
# Useful inspection functions
nrow(df); ncol(df)
```

```
[1] 5
```

```
[1] 4
```

```
head(df, 3)
```

```
  id  name degree active
1  1 Alice      3   TRUE
2  2   Bob      7   TRUE
3  3 Clara      2  FALSE
```

```
str(df)
```

```
'data.frame':   5 obs. of  4 variables:
 $ id    : int  1 2 3 4 5
 $ name  : chr  "Alice" "Bob" "Clara" "David" ...
 $ degree: num  3 7 2 5 4
 $ active: logi  TRUE TRUE FALSE TRUE FALSE
```

```
summary(df)
```

```
       id        name                degree      active
 Min.   :1   Length:5           Min.   :2.0   Mode :logical
 1st Qu.:2   Class :character   1st Qu.:3.0   FALSE:2
 Median :3   Mode  :character   Median :4.0   TRUE :3
 Mean   :3                      Mean   :4.2
 3rd Qu.:4                      3rd Qu.:5.0
 Max.   :5                      Max.   :7.0
```

---

## 3.7 Control Flow

### 3.7.1 Conditionals

```r
x <- 15

if (x > 10) {
  cat("x is greater than 10\n")
} else if (x == 10) {
  cat("x is exactly 10\n")
} else {
  cat("x is less than 10\n")
}
```

```
x is greater than 10
```

```r
# ifelse() - vectorised conditional
scores <- c(55, 72, 48, 91, 60)
ifelse(scores >= 60, "pass", "fail")
```

```
[1] "fail" "pass" "fail" "pass" "pass"
```

### 3.7.2 Loops

In R, explicit loops are often avoidable thanks to vectorisation, but they are useful for iterative tasks and are worth knowing.

```r
# for loop
for (i in 1:5) {
  cat("Iteration:", i, "\n")
```

```
}
```

```
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
Iteration: 5
```

```
# while loop
count <- 0
while (count < 3) {
  count <- count + 1
  cat("count is", count, "\n")
}
```

```
count is 1
count is 2
count is 3
```

```
# break and next
for (i in 1:10) {
  if (i == 4) next       # skip this iteration
  if (i == 7) break      # exit the loop
  cat(i, "")
}
```

```
1 2 3 5 6
```

### 3.7.3 Apply Functions

The `apply` family offers a concise way to apply a function over a vector, list, or matrix — often replacing explicit loops.

```
m <- matrix(1:12, nrow = 3)
```

```
apply(m, 1, sum)          # row sums
```

```
[1] 22 26 30
```

```
apply(m, 2, mean)          # column means
```

```
[1]  2  5  8 11
```

```
nums <- list(a = 1:4, b = 5:10, c = 11:15)
sapply(nums, mean)         # returns a named vector
```

```
   a    b    c
 2.5  7.5 13.0
```

```
lapply(nums, length)       # returns a list
```

```
$a
[1] 4

$b
[1] 6

$c
[1] 5
```

---

## 3.8 Writing Functions

Functions are the building blocks of reusable code. When you find yourself repeating the same operations, write a function.

```
# Basic function definition
greet <- function(name, greeting = "Hello") {
  paste(greeting, name)
}

greet("Alice")
```

```
[1] "Hello Alice"
```

```
greet("Bob", greeting = "Welcome")
```

```
[1] "Welcome Bob"
```

```
# A more practical example: normalise a vector to [0, 1]
normalise <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}
```

```
gdp_growth <- c(1.3, 3.7, 0.2, 4.9, 2.4)
normalise(gdp_growth)
```

```
[1] 0.2340426 0.7446809 0.0000000 1.0000000 0.4680851
```

```
# Functions can return multiple values via a list
describe <- function(x) {
  list(n = length(x), mean = mean(x), sd = sd(x), range = range(x))
}

describe(gdp_growth)
```

```
$n
[1] 5

$mean
[1] 2.5

$sd
[1] 1.866815

$range
[1] 0.2 4.9
```

---

## 3.9 Data Wrangling with the Tidyverse

The tidyverse is a collection of R packages designed around a consistent philosophy of tidy data and readable code. The core package for data manipulation is **dplyr**, which provides a small set of verbs that cover the vast majority of data-wrangling tasks.

```
library(tidyverse)
```

### 3.9.1 The Pipe Operator

The pipe `|>` (base R, since 4.1) or `%>%` (magrittr/tidyverse) passes the result of one expression as the first argument of the next. It allows you to write a sequence of transformations in the order they happen, which is much easier to read than nested function calls.

```
# Without pipe - read inside out
round(mean(c(1, 2, 3, 4, 5)), digits = 2)
```

```
[1] 3
```

```
# With pipe - read left to right
c(1, 2, 3, 4, 5) |> mean() |> round(digits = 2)
```

```
[1] 3
```

### 3.9.2 A Working Dataset

We'll use a small cross-sectional dataset of survey respondents, representative of the kind of individual-level data you might encounter in longitudinal research.

```
persons <- tibble(
  id          = 1:8,
  name        = c("Alice","Bob","Clara","David","Eva","Frank",
   ↪  "Grace","Hugo"),
  country     = c("UK","France","UK","Germany","France","UK",
   ↪  "Germany","France"),
  age         = c(32, 45, 28, 51, 38, 23, 47, 35),
  income      = c(42000, 68000, 35000, 72000, 55000, 28000, 61000,
   ↪  49000)
)
```

```
persons
```

```
# A tibble: 8 x 5
     id name  country   age income
  <int> <chr> <chr>   <dbl>  <dbl>
1     1 Alice UK         32  42000
2     2 Bob   France     45  68000
3     3 Clara UK         28  35000
```

```
4     4 David Germany    51  72000
5     5 Eva   France     38  55000
6     6 Frank UK         23  28000
7     7 Grace Germany    47  61000
8     8 Hugo  France     35  49000
```

### 3.9.3 filter() — Keep Rows

```
# Keep only UK respondents
persons |> filter(country == "UK")
```

```
# A tibble: 3 x 5
     id name  country   age income
  <int> <chr> <chr>   <dbl>  <dbl>
1     1 Alice UK         32  42000
2     3 Clara UK         28  35000
3     6 Frank UK         23  28000
```

```
# Multiple conditions
persons |> filter(country == "France", age > 40)
```

```
# A tibble: 1 x 5
     id name  country   age income
  <int> <chr> <chr>   <dbl>  <dbl>
1     2 Bob   France     45  68000
```

```
# Using %in% for multiple values
persons |> filter(country %in% c("UK", "Germany"))
```

```
# A tibble: 5 x 5
     id name  country   age income
  <int> <chr> <chr>   <dbl>  <dbl>
1     1 Alice UK         32  42000
2     3 Clara UK         28  35000
3     4 David Germany    51  72000
4     6 Frank UK         23  28000
5     7 Grace Germany    47  61000
```

### 3.9.4 select() — Choose Columns

```
persons |> select(name, country, income)
```

```
# A tibble: 8 x 3
  name  country income
  <chr> <chr>    <dbl>
1 Alice UK       42000
2 Bob   France   68000
3 Clara UK       35000
4 David Germany  72000
5 Eva   France   55000
6 Frank UK       28000
7 Grace Germany  61000
8 Hugo  France   49000
```

```
# Drop a column with -
persons |> select(-id)
```

```
# A tibble: 8 x 4
  name  country   age income
  <chr> <chr>   <dbl>  <dbl>
1 Alice UK         32  42000
2 Bob   France     45  68000
3 Clara UK         28  35000
4 David Germany    51  72000
5 Eva   France     38  55000
6 Frank UK         23  28000
7 Grace Germany    47  61000
8 Hugo  France     35  49000
```

```
# Rename while selecting
persons |> select(name, cntry = country, annual_income = income)
```

```
# A tibble: 8 x 3
  name  cntry   annual_income
  <chr> <chr>           <dbl>
1 Alice UK              42000
2 Bob   France          68000
3 Clara UK              35000
4 David Germany         72000
5 Eva   France          55000
```

```
6 Frank UK              28000
7 Grace Germany         61000
8 Hugo  France          49000
```

### 3.9.5 mutate() — Create or Modify Columns

```
persons |>
  mutate(
    older       = age >= 40,
    income_z    = (income - mean(income)) / sd(income),   #
     ↪ standardise
    label       = paste0(name, " (", country, ")")
  )
```

```
# A tibble: 8 x 8
     id name  country   age income older income_z label
  <int> <chr> <chr>   <dbl>  <dbl> <lgl>    <dbl> <chr>
1     1 Alice UK         32  42000 FALSE   -0.591 Alice (UK)
2     2 Bob   France     45  68000 TRUE     1.07  Bob (France)
3     3 Clara UK         28  35000 FALSE   -1.04  Clara (UK)
4     4 David Germany    51  72000 TRUE     1.33  David (Germany)
5     5 Eva   France     38  55000 FALSE    0.240 Eva (France)
6     6 Frank UK         23  28000 FALSE   -1.49  Frank (UK)
7     7 Grace Germany    47  61000 TRUE     0.623 Grace (Germany)
8     8 Hugo  France     35  49000 FALSE   -0.144 Hugo (France)
```

### 3.9.6 arrange() — Sort Rows

```
persons |> arrange(desc(income))       # highest income first
```

```
# A tibble: 8 x 5
     id name  country   age income
  <int> <chr> <chr>   <dbl>  <dbl>
1     4 David Germany    51  72000
2     2 Bob   France     45  68000
3     7 Grace Germany    47  61000
4     5 Eva   France     38  55000
5     8 Hugo  France     35  49000
6     1 Alice UK         32  42000
7     3 Clara UK         28  35000
8     6 Frank UK         23  28000
```

```
  persons |> arrange(country, age)       # sort by two columns
```

```
# A tibble: 8 x 5
     id name  country   age income
  <int> <chr> <chr>   <dbl>  <dbl>
1     8 Hugo  France     35  49000
2     5 Eva   France     38  55000
3     2 Bob   France     45  68000
4     7 Grace Germany    47  61000
5     4 David Germany    51  72000
6     6 Frank UK         23  28000
7     3 Clara UK         28  35000
8     1 Alice UK         32  42000
```

### 3.9.7 summarise() and group_by() — Aggregation

```
  # Overall summary
  persons |>
    summarise(
      n           = n(),
      mean_income = mean(income),
      sd_income   = sd(income),
      max_age     = max(age)
    )
```

```
# A tibble: 1 x 4
      n mean_income sd_income max_age
  <int>       <dbl>     <dbl>   <dbl>
1     8       51250    15655.      51
```

```
  # Summary by group - useful for comparing units in panel data
  persons |>
    group_by(country) |>
    summarise(
      n           = n(),
      mean_income = mean(income),
      mean_age    = mean(age)
    ) |>
    arrange(desc(mean_income))
```

```
# A tibble: 3 x 4
```

```
  country     n mean_income mean_age
  <chr>   <int>       <dbl>    <dbl>
1 Germany     2       66500       49
2 France      3       57333.      39.3
3 UK          3       35000       27.7
```

### 3.9.8 Joining Tables

Joins combine two data frames on a shared key column — essential in longitudinal work when merging panel waves with time-invariant attributes, or linking datasets from different sources.

```
# Panel waves: repeated observations per person across two years
waves <- tibble(
  id    = c(1, 1, 2, 2, 3, 3, 4, 4),
  year = c(2020, 2021, 2020, 2021, 2020, 2021, 2020, 2021),
  income_obs = c(40000, 42000, 65000, 68000, 33000, 35000, 70000,
   ↪  72000)
)

# Attach time-invariant person attributes to the panel
waves |>
  left_join(persons |> select(id, name, country),
            by = "id") |>
  arrange(id, year)
```

```
# A tibble: 8 x 5
     id  year income_obs name  country
  <dbl> <dbl>      <dbl> <chr> <chr>
1     1  2020      40000 Alice UK
2     1  2021      42000 Alice UK
3     2  2020      65000 Bob   France
4     2  2021      68000 Bob   France
5     3  2020      33000 Clara UK
6     3  2021      35000 Clara UK
7     4  2020      70000 David Germany
8     4  2021      72000 David Germany
```

### 3.9.9 Reshaping Data

```
# Wide to long (pivot_longer) - core operation when converting
 ↪ panel data to long format
persons |>
  select(name, age, income) |>
  pivot_longer(cols = c(age, income),
               names_to  = "variable",
               values_to = "value")
```

```
# A tibble: 16 x 3
   name  variable value
   <chr> <chr>    <dbl>
 1 Alice age         32
 2 Alice income   42000
 3 Bob   age         45
 4 Bob   income   68000
 5 Clara age         28
 6 Clara income   35000
 7 David age         51
 8 David income   72000
 9 Eva   age         38
10 Eva   income   55000
11 Frank age         23
12 Frank income   28000
13 Grace age         47
14 Grace income   61000
15 Hugo  age         35
16 Hugo  income   49000
```

```
# Long to wide (pivot_wider) - useful when reshaping repeated
 ↪ measures to one row per unit
waves |>
  select(id, year, income_obs) |>
  pivot_wider(names_from = year, values_from = income_obs,
              names_prefix = "income_")
```

```
# A tibble: 4 x 3
     id income_2020 income_2021
  <dbl>       <dbl>       <dbl>
1     1       40000       42000
2     2       65000       68000
3     3       33000       35000
```

```
4       4         70000         72000
```

---

## 3.10 Reading and Writing Data

```
# CSV files
df <- read_csv("data/panel_data.csv")       # tidyverse
 ↪  (recommended)
df <- read.csv("data/panel_data.csv")       # base R
```

```
write_csv(df, "data/panel_data_clean.csv")
```

```
# Excel files
library(readxl)
df <- read_excel("data/data.xlsx", sheet = 1)
```

```
# SPSS, Stata, SAS
library(haven)
df <- read_spss("data/survey.sav")
df <- read_dta("data/survey.dta")
```

```
# R's native format
saveRDS(df, "data/df.rds")
df <- readRDS("data/df.rds")
```

---

## 3.11 Quick Reference

| Task | Function |
|---|---|
| Assign a value | `x <- value` |
| Create a vector | `c(1, 2, 3)` |
| Create a sequence | `1:10`, `seq(0, 1, 0.1)` |
| Create a matrix | `matrix(data, nrow, ncol)` |

| Task | Function |
|---|---|
| Create a data frame | `data.frame()` / `tibble()` |
| Check an object's type | `class()`, `typeof()` |
| Check dimensions | `dim()`, `nrow()`, `ncol()`, `length()` |
| Inspect structure | `str()`, `summary()`, `head()` |
| Filter rows | `filter()` |
| Select columns | `select()` |
| Create columns | `mutate()` |
| Sort rows | `arrange()` |
| Aggregate | `group_by()` + `summarise()` |
| Join tables | `left_join()`, `inner_join()` |
| Reshape wide → long | `pivot_longer()` |
| Reshape long → wide | `pivot_wider()` |

# 4 Working with Panel Data in R

Structure, Reshaping, Indexing, and Missingness

Before estimating any model, we need to understand the two fundamental shapes that panel data can take, know how to move between them, and be able to detect and address the most common data quality problems. This chapter covers all three.

---

## 4.1 Two Shapes of Panel Data

Panel data — repeated observations on the same units over time — can be stored in two distinct formats. Understanding the difference is essential before you can do anything else.

### 4.1.1 Person-level (wide) format

In **wide** format, each row represents one unit (a person, firm, or country) and each wave of measurement occupies a separate column. This is how many surveys distribute their data and how non-specialists tend to organise spreadsheets.

```
wide_example <- tibble(
  id     = 1:4,
  gender = c("F", "M", "F", "M"),
  wage_2020 = c(2.1, 3.4, 1.8, 2.9),
  wage_2021 = c(2.3, 3.6, 1.9, 3.1),
  wage_2022 = c(2.4, 3.5, 2.2, 3.3),
  educ_2020 = c(12, 16, 14, 12),
  educ_2021 = c(12, 16, 16, 12),
  educ_2022 = c(14, 16, 16, 14)
)
```

```
  wide_example
```

```
# A tibble: 4 x 8
     id gender wage_2020 wage_2021 wage_2022 educ_2020 educ_2021
      ↪  educ_2022
  <int> <chr>      <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
   ↪  <dbl>
1     1 F            2.1       2.3       2.4        12        12
 ↪  14
2     2 M            3.4       3.6       3.5        16        16
 ↪  16
3     3 F            1.8       1.9       2.2        14        16
 ↪  16
4     4 M            2.9       3.1       3.3        12        12
 ↪  14
```

Each person appears on exactly **one row**. Repeated measurements are spread across columns named by variable and wave (`wage_2020`, `wage_2021`, …). This makes it easy to compute within-person changes (subtract column A from column B), but it is awkward for modelling: most statistical functions in R expect one observation per row.

### 4.1.2 Person-period (long) format

In **long** format, each row is one observation — one unit at one point in time. A person observed at four time points occupies four rows.

```
  long_example <- wide_example |>
    pivot_longer(
      cols      = starts_with("wage_") | starts_with("educ_"),
      names_to  = c(".value", "year"),
      names_sep = "_"
    ) |>
    mutate(year = as.integer(year))

  long_example
```

```
# A tibble: 12 x 5
      id gender  year  wage  educ
   <int> <chr>  <int> <dbl> <dbl>
 1     1 F       2020   2.1    12
 2     1 F       2021   2.3    12
 3     1 F       2022   2.4    14
```

```
 4     2 M       2020   3.4    16
 5     2 M       2021   3.6    16
 6     2 M       2022   3.5    16
 7     3 F       2020   1.8    14
 8     3 F       2021   1.9    16
 9     3 F       2022   2.2    16
10     4 M       2020   2.9    12
11     4 M       2021   3.1    12
12     4 M       2022   3.3    14
```

This is the format required by virtually all panel model packages in R (`plm`, `fixest`, `lme4`). Each row has a unit identifier (`id`), a time identifier (`year`), and then the variables measured at that wave.

> **Which format does your data arrive in?**
>
> Survey data is often wide: one row per respondent, with wave-specific suffixes. Administrative records are usually long: each event or measurement is a separate record. Knowing which you have determines your first step.

---

## 4.2 Reading Panel Data into R

### 4.2.1 CSV and tabular data

The workshop dataset — 200 individuals tracked annually from 2016 to 2023 — is stored as a CSV:

```
panel_data <- read_csv("files/panel_data.csv", show_col_types =
 ↪  FALSE)

glimpse(panel_data)
```

```
Rows: 1,600
Columns: 11
$ id         <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3,
 ↪  3, 3,~
$ year       <dbl> 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2016,
 ↪  2017,~
$ period     <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 3, 4, 5, 6, 7, 8, 1, 2,
 ↪  3, 4,~
```

```
$ gender     <chr> "Female", "Female", "Female", "Female", "Female",
 ↪  "Female",~
$ sector     <chr> "Public", "Public", "Public", "Public", "Public",
 ↪  "Public",~
$ education  <dbl> 10, 10, 11, 11, 11, 12, 12, 12, 11, 11, 12, 12, 12,
 ↪  13, 13,~
$ experience <dbl> 6, 7, 8, 9, 10, 11, 12, 13, 6, 7, 8, 9, 10, 11, 12,
 ↪  13, 10,~
$ union      <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
 ↪  0, 0,~
$ ability    <dbl> -2.0009292, -2.0009292, -2.0009292, -2.0009292,
 ↪  -2.0009292,~
$ wage       <dbl> 5.883, 8.100, 7.670, 6.753, 7.688, 10.231, NA, 8.301,
 ↪  17.21~
$ log_wage   <dbl> 1.772119, 2.091868, 2.037348, 1.909919, 2.039721,
 ↪  2.325465,~
```

The dataset is already in **long format**: each row is one person-year. The key identifiers are `id` (person) and `year` (time).

### 4.2.2 Stata and SPSS files

Many longitudinal datasets are distributed in Stata (`.dta`) or SPSS (`.sav`) format. The `haven` package reads both directly, preserving value labels:

```
library(haven)

# Stata
df_stata <- read_dta("your_data.dta")

# SPSS
df_spss  <- read_sav("your_data.sav")

# Convert labelled columns to factors
df_stata <- df_stata |> mutate(across(where(is.labelled),
 ↪  as_factor))
```

### 4.2.3 Excel files

```
library(readxl)

df_xl <- read_excel("your_data.xlsx", sheet = "Panel", na = c("",
 ↪  "NA", "."))
```

---

## 4.3 Inspecting the Panel Structure

Before modelling, always verify that your data is what you think it is.

### 4.3.1 How many units? How many periods?

```
cat("Individuals :", n_distinct(panel_data$id),   "\n")
```

```
Individuals : 200
```

```
cat("Time periods:", n_distinct(panel_data$year),  "\n")
```

```
Time periods: 8
```

```
cat("Total rows  :", nrow(panel_data),             "\n")
```

```
Total rows  : 1600
```

```
cat("Year range  :", min(panel_data$year), "-",
 ↪  max(panel_data$year), "\n")
```

```
Year range  : 2016 - 2023
```

### 4.3.2 Is the panel balanced?

A **balanced** panel has the same number of time points for every unit. An **unbalanced** panel does not — some units are observed less frequently. The distinction matters for estimation.

```
obs_per_person <- panel_data |>
  count(id, name = "n_obs")

obs_per_person |>
  count(n_obs, name = "n_individuals") |>
  kable(
    col.names = c("Observations per person", "Number of
     ↪  individuals"),
    caption   = "Table 1 - Panel balance check"
  )
```

Table 4.1: Table 1 — Panel balance check

| Observations per person | Number of individuals |
|---|---|
| 8 | 200 |

```
ggplot(obs_per_person, aes(x = n_obs)) +
  geom_bar(fill = "#045a8d", width = 0.6) +
  scale_x_continuous(breaks = 1:10) +
  labs(x = "Observations per individual", y = "Count",
       title = "A balanced panel: every individual has exactly 8
        ↪  observations") +
  theme_minimal(base_size = 13)
```

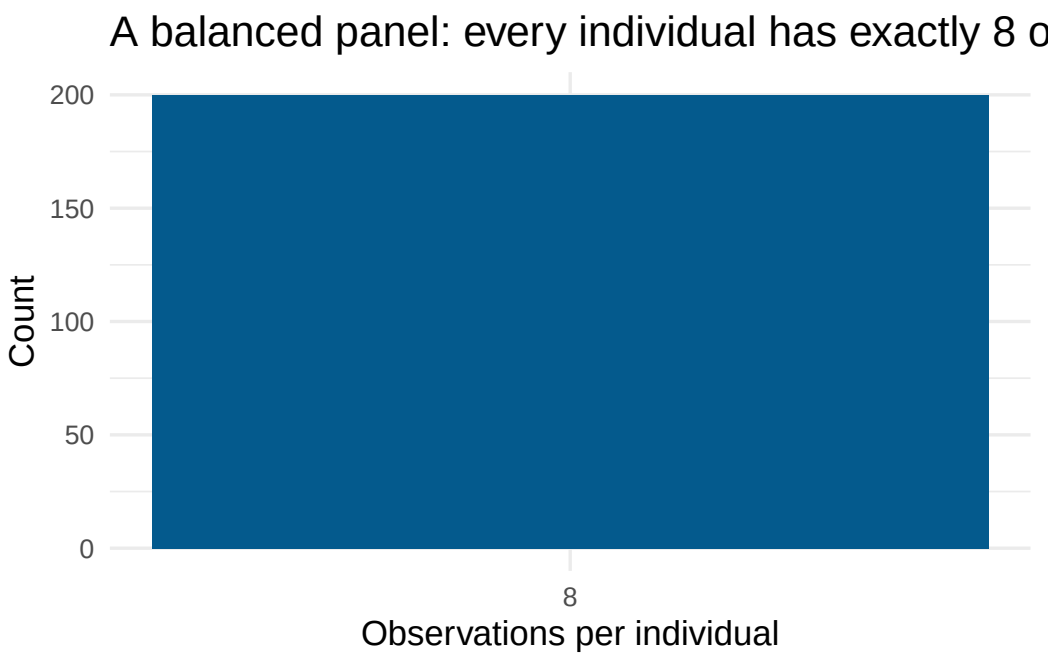


Figure 4.1: Figure 1 — Number of observations per individual (balanced panel)

### 4.3.3 Unit and time coverage at a glance

For small panels it can be useful to visualise which units are observed at which times — a "swimmer plot" or tile chart that reveals gaps immediately.

```
set.seed(7)
sample_ids <- sample(unique(panel_data$id), 30)

panel_data |>
  filter(id %in% sample_ids) |>
  mutate(id = factor(id, levels = rev(sort(sample_ids)))) |>
  ggplot(aes(x = year, y = id)) +
  geom_tile(fill = "#045a8d", colour = "white", linewidth = 0.3) +
  scale_x_continuous(breaks = 2016:2023) +
  labs(x = "Year", y = "Individual ID",
       title = "Observation grid - filled = observed, gap =
        ↪  missing") +
  theme_minimal(base_size = 11) +
  theme(axis.text.y = element_text(size = 7),
        panel.grid  = element_blank())
```

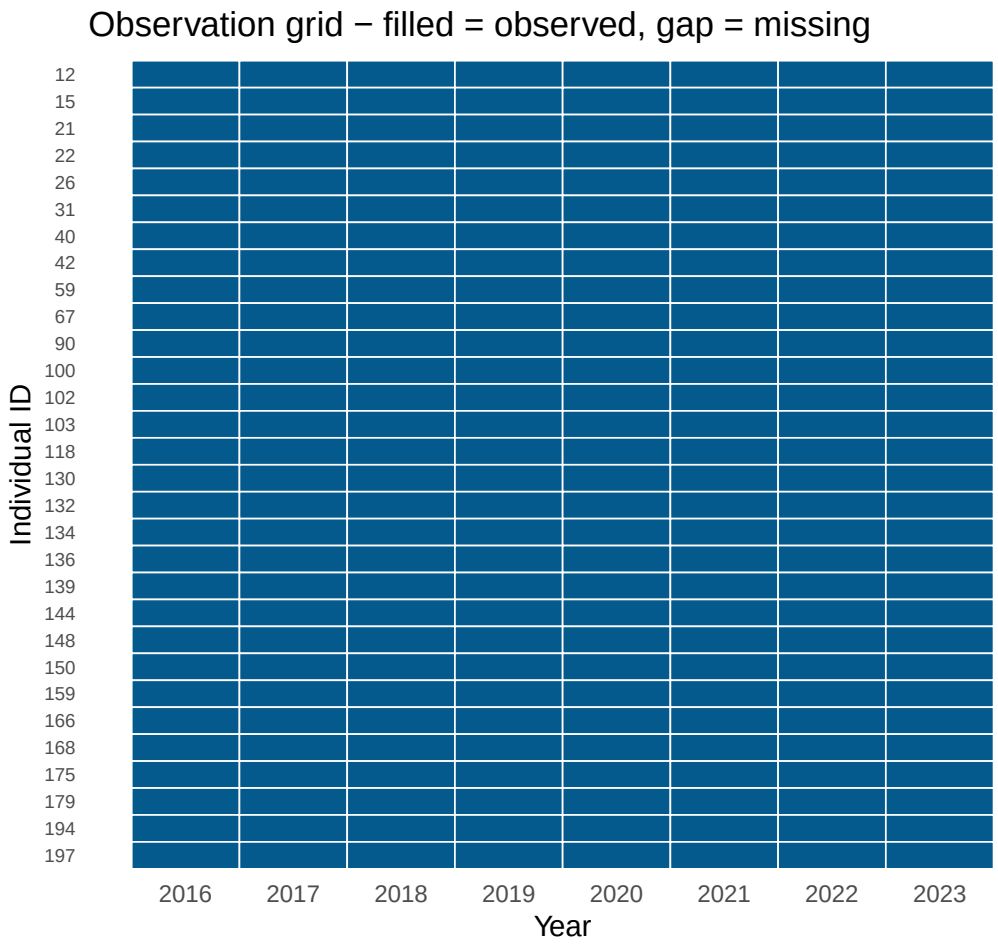


Figure 4.2: Figure 2 — Observation coverage for a random sample of 30 individuals

---

## 4.4 Cleaning Panel Data

### 4.4.1 Standardise variable names

The `janitor` package's `clean_names()` converts all column names to `snake_case`, removes special characters, and ensures uniqueness. Get into the habit of calling it immediately after reading data.

```
# Requires: install.packages("janitor")
library(janitor)
```

```
# Simulate messy column names from an external file
messy <- tibble(
  `Person ID` = 1:3,
  `Survey Year` = 2020:2022,
  `Log Wage (£)` = c(7.2, 7.4, 7.3),
  `Union Member?` = c(1, 0, 1)
)

messy |> clean_names()
```

### 4.4.2 Parse and handle date variables

Many longitudinal datasets record time as a date string rather than an integer year. `lubridate` provides a consistent grammar for parsing and computing with dates.

```
date_example <- tibble(
  id           = c(1, 1, 2, 2),
  interview_dt = c("2020-03-15", "2022-09-01", "2020-06-22",
   ↪  "2022-11-30")
)

date_example <- date_example |>
  mutate(
    interview_dt = ymd(interview_dt),           # parse to Date
    year         = year(interview_dt),          # extract year
    month        = month(interview_dt, label = TRUE),
    wave_gap_days = as.integer(
      interview_dt - lag(interview_dt), units = "days"
    )
  )

date_example
```

```
# A tibble: 4 x 5
     id interview_dt  year month wave_gap_days
  <dbl> <date>       <dbl> <ord>         <int>
1     1 2020-03-15    2020 Mar               0
2     1 2022-09-01    2022 Sep               0
3     2 2020-06-22    2020 Jun               0
4     2 2022-11-30    2022 Nov               0
```

### 4.4.3 Recode and relabel variables

```
panel_data <- panel_data |>
  mutate(
    gender    = factor(gender, levels = c("Male", "Female")),
    sector    = factor(sector, levels = c("Public", "Private")),
    union_lbl = if_else(union == 1, "Union member", "Non-member")
  )

panel_data |>
  count(gender, sector) |>
  kable(caption = "Table 2 - Cross-tabulation: gender × sector")
```

Table 4.2: Table 2 — Cross-tabulation: gender × sector

| gender | sector | n |
|---|---|---|
| Male | Public | 224 |
| Male | Private | 528 |
| Female | Public | 240 |
| Female | Private | 608 |

---

## 4.5 Reshaping: Wide Long

### 4.5.1 Long to wide with `pivot_wider`

Sometimes you need wide format — for example to compute person-level change scores or to merge with a cross-sectional dataset.

```
wages_wide <- panel_data |>
  select(id, year, wage) |>
  pivot_wider(
    names_from  = year,
    values_from = wage,
    names_prefix = "wage_"
  )

head(wages_wide, 5)
```

```
# A tibble: 5 x 9
     id wage_2016 wage_2017 wage_2018 wage_2019 wage_2020 wage_2021
      ↪  wage_2022
  <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
   ↪  <dbl>
1     1      5.88      8.1       7.67      6.75      7.69      10.2
 ↪  NA
2     2     17.2      19.0      13.8      18.3      17.1      23.2
 ↪  20.3
3     3     25.5      20.2      21.6      NA        27.4      32.5
 ↪  25.7
4     4     33.4      31.6      38.9      42.2      40.4      49.1
 ↪  45.6
5     5      6.36      7.32      8.17     10.2       6.18      6.91
 ↪  9.41
# i 1 more variable: wage_2023 <dbl>
```

You can now compute within-person change between any two waves:

```
wages_wide <- wages_wide |>
  mutate(wage_change_16_23 = wage_2023 - wage_2016)

wages_wide |>
  select(id, wage_2016, wage_2023, wage_change_16_23) |>
  slice_head(n = 6) |>
  kable(
    col.names = c("ID", "Wage 2016", "Wage 2023", "Change
     ↪  2016→2023"),
    caption   = "Table 3 - Within-person wage change (wide format)"
  )
```

Table 4.3: Table 3 — Within-person wage change (wide format)

| ID | Wage 2016 | Wage 2023 | Change 2016→2023 |
|---|---|---|---|
| 1 | 5.883 | 8.301 | 2.418 |
| 2 | 17.217 | 18.898 | 1.681 |
| 3 | 25.461 | 26.789 | 1.328 |
| 4 | 33.405 | 46.068 | 12.663 |
| 5 | 6.365 | 9.512 | 3.147 |
| 6 | 9.270 | 8.428 | -0.842 |

### 4.5.2 Wide to long with `pivot_longer`

To go back to long format, or when your data arrives wide:

```
wages_long_again <- wages_wide |>
  select(id, starts_with("wage_20")) |>      # drop the change
   ↪  score
  pivot_longer(
    cols         = starts_with("wage_20"),
    names_to     = "year",
    names_prefix = "wage_",
    values_to    = "wage"
  ) |>
  mutate(year = as.integer(year))

head(wages_long_again, 8)
```

```
# A tibble: 8 x 3
     id  year  wage
  <dbl> <int> <dbl>
1     1  2016  5.88
2     1  2017  8.1
3     1  2018  7.67
4     1  2019  6.75
5     1  2020  7.69
6     1  2021 10.2
7     1  2022 NA
8     1  2023  8.30
```

### 4.5.3 Reshaping multiple variables simultaneously

When several variables are wave-stamped (e.g., `wage_2020`, `educ_2020`, `wage_2021`, `educ_2021`), use the `".value"` sentinel in `names_to` to split the column name into variable name and wave:

```
# Reconstruct a multi-variable wide dataset to demonstrate
multi_wide <- panel_data |>
  select(id, year, wage, education) |>
  filter(id <= 3) |>
  pivot_wider(
    names_from  = year,
    values_from = c(wage, education)
```

```
  )

# Pivot back to long, splitting "wage_2016" into variable="wage",
 ↪  year="2016"
multi_wide |>
  pivot_longer(
    cols      = -id,
    names_to  = c(".value", "year"),
    names_sep = "_(?=\\d{4}$)"           # split on _ before a
     ↪  4-digit year
  ) |>
  mutate(year = as.integer(year)) |>
  arrange(id, year)
```

```
# A tibble: 24 x 4
      id  year  wage education
   <dbl> <int> <dbl>     <dbl>
 1     1  2016  5.88        10
 2     1  2017  8.1         10
 3     1  2018  7.67        11
 4     1  2019  6.75        11
 5     1  2020  7.69        11
 6     1  2021 10.2         12
 7     1  2022 NA           12
 8     1  2023  8.30        12
 9     2  2016 17.2         11
10     2  2017 19.0         11
# i 14 more rows
```

---

## 4.6 Indexing: Creating a `pdata.frame`

The `plm` package works with **panel data frames** — objects that know which columns identify units and time. Creating one is the gateway to all of `plm`'s modelling and diagnostic functions.

```
pdata <- pdata.frame(panel_data, index = c("id", "year"))

# The class is both a pdata.frame and a data.frame
class(pdata)
```

```
[1] "pdata.frame" "data.frame"
```

### 4.6.1 Inspecting the panel index

```
pdim(pdata)
```

```
Balanced Panel: n = 200, T = 8, N = 1600
```

`pdim()` reports the panel dimensions and whether it is balanced. The `T` here is the number of time periods, `N` the number of individuals.

### 4.6.2 Within and between variation

A central diagnostic in panel analysis is how much variation in your key variables is *within* units (across time) versus *between* units (across individuals). `plm::Within()` and `between()` compute these directly.

```
within_between <- panel_data |>
  summarise(
    across(
      c(wage, education, experience),
      list(
        total_sd  = \(x) round(sd(x, na.rm = TRUE), 3),
        between_sd = \(x) round(sd(tapply(x, id, mean, na.rm =
         ↪  TRUE),
                                   na.rm = TRUE), 3),
        within_sd  = \(x) round(mean(tapply(x, id,
                                   \(v) sd(v, na.rm = TRUE),
                                   simplify = TRUE), na.rm = TRUE),
                                    ↪  3)
      )
    )
  ) |>
  pivot_longer(everything(), names_to = c("Variable", ".value"),
               names_sep = "_(?=total|between|within)") |>
  rename(`Total SD` = total_sd,
         `Between SD` = between_sd,
         `Within SD` = within_sd)

kable(within_between,
```

```
    caption = "Table 4 - Decomposition of variance: total,
    ↪  between, and within")
```

Table 4.4: Table 4 — Decomposition of variance: total, between, and within

| Variable | Total SD | Between SD | Within SD |
|---|---|---|---|
| wage | 9.933 | 9.338 | 3.355 |
| education | 2.495 | 2.376 | 0.830 |
| experience | 4.779 | 4.203 | 2.449 |

Reading the variance decomposition

- **Between SD** — how much the *person-average* of a variable varies across individuals. This is the variation exploited by cross-sectional (OLS) models.
- **Within SD** — how much a variable varies *within* the same individual over time. This is the variation used by the Fixed Effects estimator.

A variable with low within-SD and high between-SD (e.g., gender) carries little

```
within_between |>
  pivot_longer(c(`Between SD`, `Within SD`),
               names_to = "Component", values_to = "SD") |>
  ggplot(aes(x = Variable, y = SD, fill = Component)) +
  geom_col(position = "dodge", width = 0.6) +
  scale_fill_manual(values = c("Between SD" = "#045a8d",
                               "Within SD"  = "#74add1")) +
  labs(x = NULL, y = "Standard deviation", fill = NULL,
       title = "Within vs between variation",
       subtitle = "FE uses within; OLS mixes both") +
  theme_minimal(base_size = 13) +
  theme(legend.position = "top")
```

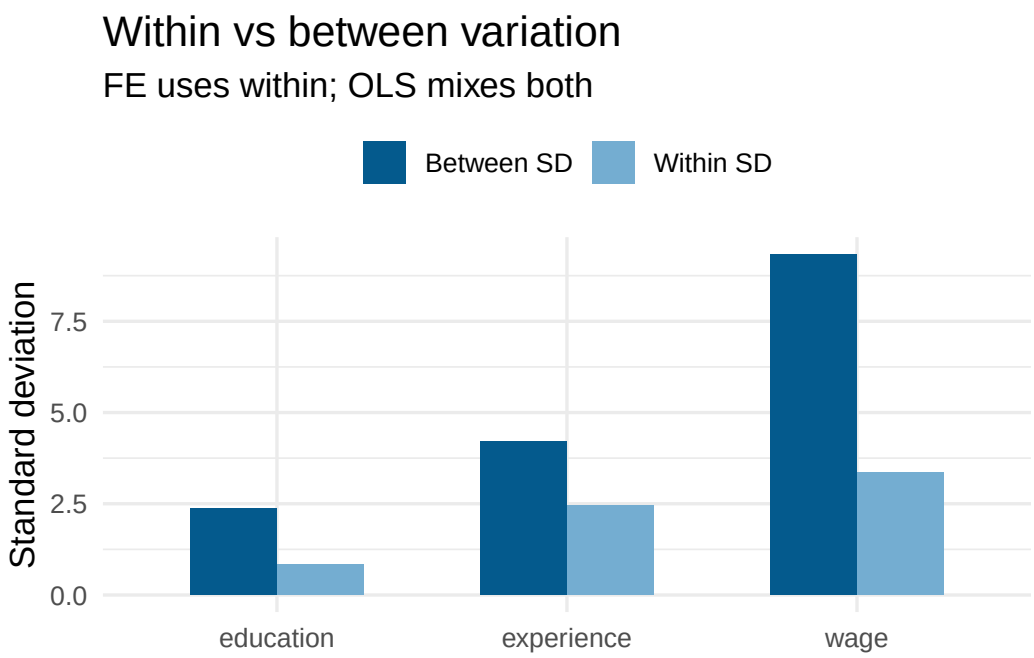


Figure 4.3: Figure 3 — Within vs between standard deviation for key variables

---

## 4.7 Handling Missingness

Real panel datasets almost always have missing values. The source and pattern of missingness determines how seriously it threatens your estimates.

### 4.7.1 Types of missingness

| Type | Definition | Implication |
|---|---|---|
| **MCAR** | Missing completely at random — missingness is unrelated to any variable | Listwise deletion is unbiased (just less efficient) |
| **MAR** | Missing at random — missingness depends on *observed* variables | Multiple imputation or FIML can recover unbiased estimates |
| **MNAR** | Missing not at random — missingness depends on the *unobserved* value itself | Requires a selection model; listwise deletion is biased |

### 4.7.2 Detecting missingness

```
missing_summary <- panel_data |>
  summarise(across(everything(),
                   \(x) sum(is.na(x)))) |>
  pivot_longer(everything(),
               names_to  = "Variable",
               values_to = "N missing") |>
  mutate(`% missing` = round(`N missing` / nrow(panel_data) * 100,
   ↪  2)) |>
  filter(`N missing` > 0)

kable(missing_summary,
      caption = "Table 5 - Missing values by variable")
```

Table 4.6: Table 5 — Missing values by variable

| Variable | N missing | % missing |
|---|---|---|
| wage | 48 | 3 |
| log_wage | 48 | 3 |

### 4.7.3 Is missingness systematic?

The first question to ask: is missingness in the outcome related to observed predictors? If it is, listwise deletion may introduce bias.

```
panel_data |>
  arrange(id, year) |>
  group_by(id) |>
  mutate(
    missing_next_wage = is.na(dplyr::lead(wage))
  ) |>
  ungroup() |>
  filter(!is.na(missing_next_wage)) |>
  group_by(year, missing_next_wage) |>
  summarise(mean_wage = mean(wage, na.rm = TRUE), .groups = "drop")
   ↪  |>
  ggplot(aes(x = year, y = mean_wage,
             colour = missing_next_wage, group =
              ↪  missing_next_wage)) +
```

```
  geom_line(linewidth = 1.1) +
  geom_point(size = 3) +
  scale_colour_manual(values = c("FALSE" = "#045a8d", "TRUE" =
   ↪  "#d73027"),
                      labels  = c("Wage observed next year",
                                  "Wage missing next year")) +
  labs(x = NULL, y = "Mean wage (£'000s/month)", colour = NULL,
       title = "Do lower earners drop out more often?",
       subtitle = "Red = individuals who are missing the following
        ↪  year") +
  theme_minimal(base_size = 13) +
  theme(legend.position = "bottom")
```

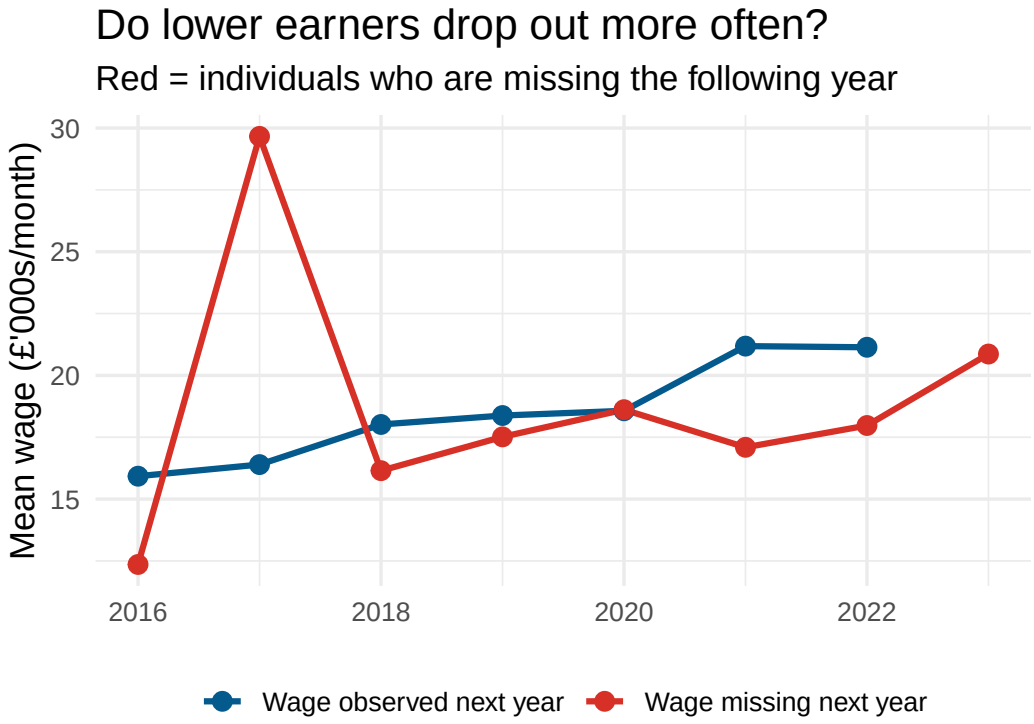


Figure 4.4: Figure 4 — Mean wage by year and missingness in the following year

### 4.7.4 Listwise deletion (complete cases)

The default in most R model functions. Observations with any `NA` on the modelled variables are dropped. This is valid only under MCAR.

```
panel_complete <- panel_data |> drop_na(wage, education,
 ↪  experience, union)

cat("Rows before:", nrow(panel_data),    "\n")
```

```
Rows before: 1600
```

```
cat("Rows after :", nrow(panel_complete), "\n")
```

```
Rows after : 1552
```

```
cat("Dropped    :", nrow(panel_data) - nrow(panel_complete), "\n")
```

```
Dropped    : 48
```

### 4.7.5 Carrying observations forward (`fill`)

In some panel datasets, a variable is recorded only when it changes (e.g., a diagnosis or employment status). `tidyr::fill()` propagates the last observed value forward (or backward) within each unit.

```
fill_example <- tibble(
  id     = c(1, 1, 1, 1, 2, 2, 2, 2),
  year   = rep(2020:2023, 2),
  status = c("Employed", NA, NA, "Unemployed",
             "Student",  NA, "Employed", NA)
)

fill_example |>
  group_by(id) |>
  fill(status, .direction = "down") |>
  ungroup()
```

```
# A tibble: 8 x 3
     id  year status
  <dbl> <int> <chr>
1     1  2020 Employed
2     1  2021 Employed
3     1  2022 Employed
4     1  2023 Unemployed
5     2  2020 Student
```

```
6     2  2021 Student
7     2  2022 Employed
8     2  2023 Employed
```

### 4.7.6 Checking panel balance after dropping missing rows

Dropping rows due to missingness can **unbalance** a previously balanced panel. Always recheck after listwise deletion:

```
pdata_complete <- pdata.frame(panel_complete, index = c("id",
 ↪  "year"))
pdim(pdata_complete)
```

```
Unbalanced Panel: n = 200, T = 6-8, N = 1552
```

⚠ Unbalanced panels and fixed effects

An unbalanced panel is not invalid — `plm` and `fixest` both handle it correctly. But it is worth understanding *why* balance was lost: if high-wage or low-wage individuals are disproportionately missing, your within-estimator still operates on a selected sample.

### 4.7.7 Flagging and summarising incomplete spells

ℹ What is a "spell"?

In the context of longitudinal data and panel studies, a **spell** refers to a continuous period of time that an individual spends in a particular state — employed, unemployed, enrolled in education, living in a given region, and so on.
Think of it as a single "chapter" or "streak" in someone's data: a spell begins when the individual enters a state and ends when they leave it. A person may have multiple spells of the same type over the observation window (e.g., two separate employment spells separated by a period of unemployment).
In the code below, a *complete spell* means an individual is observed in every time period in the dataset — their "chapter" has no missing pages.

```
incomplete <- panel_data |>
  group_by(id) |>
  summarise(
    n_obs   = n(),
```

```
    n_miss  = sum(is.na(wage)),
    complete = n_obs == max(panel_data |>
                 count(id) |> pull(n))
  ) |>
  ungroup()

incomplete |>
  count(complete) |>
  mutate(label = if_else(complete, "Complete spells", "Incomplete
   ↪  spells")) |>
  kable(col.names = c("Complete", "N individuals", "Label"),
        caption = "Table 6 - Complete vs incomplete individual
         ↪  spells")
```

Table 4.7: Table 6 — Complete vs incomplete individual spells

| Complete | N individuals | Label |
|---|---|---|
| TRUE | 200 | Complete spells |

## 4.8 Summary

| Task | Key function(s) |
|---|---|
| Read CSV / Stata / SPSS / Excel | `read_csv()`, `read_dta()`, `read_sav()`, `read_excel()` |
| Standardise column names | `janitor::clean_names()` |
| Parse dates | `lubridate::ymd()`, `year()`, `month()` |
| Wide → long | `tidyr::pivot_longer()` |
| Long → wide | `tidyr::pivot_wider()` |
| Check balance | `plm::pdim()` |
| Detect missing | `is.na()`, `drop_na()`, `naniar::gg_miss_var()` |
| Carry values forward | `tidyr::fill()` |
| Panel index | `plm::pdata.frame(df, index = c("id", "time"))` |
| Within/between split | `plm::Within()`, `plm::between()` |

> ! Before fitting any model, always verify
>
> 1. Your data is in **long format** (one row per unit × time).
> 2. The unit and time **identifiers are correct** and uniquely index each row.
> 3. The panel is **balanced** — or you understand why it isn't.
> 4. You know the **direction and magnitude of missingness** in your outcome.
> 5. You have decomposed key variables into **within and between components** to understand what variation your chosen estimator will exploit.

---

## 4.9 Common Panel Data Sources

Panel and longitudinal datasets come from a wide range of sources. Below is a selection of well-known survey-based panels and process-generated datasets with a longitudinal dimension.

### 4.9.1 Household and individual panel surveys

These are purpose-designed surveys that track the same individuals or households over many waves:

- **PSID** — Panel Study of Income Dynamics (USA); one of the longest-running household panel surveys, covering income, wealth, employment, and health since 1968.
- **NLSY79** — National Longitudinal Survey of Youth 1979 cohort (USA); tracks labour market experiences, education, and family formation.
- **Understanding Society** — UK Household Longitudinal Study; follows ~40,000 households annually on a broad range of social and economic outcomes.
- **Millennium Cohort Study** — Follows ~19,000 children born in the UK in 2000–01 through childhood, adolescence, and into adulthood.
- **SOEP** — Socio-Economic Panel Study (Germany); annual household survey running since 1984, covering income, employment, health, and well-being.
- **pairfam** — Panel Analysis of Intimate Relationships and Family Dynamics (Germany); focuses on partnership, fertility, and intergenerational relations.
- **SHARE** — Survey of Health, Ageing and Retirement in Europe; multi-country panel of individuals aged 50+.

### 4.9.2 Process-generated longitudinal data

Administrative records and monitoring systems often produce data with a natural longitudinal structure, even though they were not designed as surveys:

- **Our World in Data — COVID-19** — Country-day panel of cases, deaths, vaccinations, and policy responses; well-documented and regularly updated.
- **UK Census (Nomis)** — Area-level statistics from the decennial UK Census; geographic units can be tracked across census years.
- **London Air Quality Network** — Station-level time series of pollutant concentrations across London.
- **Copernicus Tree Cover Density** — Raster-based estimates of tree cover density across Europe; repeated status maps enable change analysis over time.

### 4.9.3 Practical considerations

**Merging across waves.** Many panel datasets are distributed as a separate file per wave (each file is cross-sectional), which must be stacked or merged to create a panel. This is a common and time-consuming step. Some data providers or the research community have published pre-written merge scripts to simplify the process — see for example the Understanding Society syntax library or the SOEP teaching materials.

**Panel attrition.** Participation in surveys tends to fall over successive waves. Individuals drop out for a variety of reasons — non-contact, refusal, migration, or death — and those who remain may be systematically different from those who leave. If attrition is related to the outcome of interest, estimates based on the surviving sample can be biased. Always check how many units are observed across all waves and consider whether attrition looks random.

# 5 Fixed or Random — Which Model Fits?

From Fixed Effects to Random Effects, and How to Choose

```
library(tidyverse)
library(plm)
library(knitr)
library(RColorBrewer)

pal_farms <- brewer.pal(5, "Set2")
```

---

## 5.1 Recap — What We Learned in Exercise 1

In the first exercise we tracked five farms over two years and found:

- **Pooled OLS** was badly biased — it confused high-quality farms with productive farms.
- **Fixed Effects (FE)** eliminated that bias by using *within-farm* variation only: holding each farm constant and asking "when this farm hired more workers, did it produce more?"
- The FE estimate ($\hat{\beta}_{\text{FE}} \approx 2.5$) was close to the true value; OLS was far above it.

> **The FE trade-off**
>
> FE is powerful because it controls for **all** time-invariant unobserved heterogeneity — soil quality, management culture, location — without ever measuring it. But that power comes at a cost:
>
> > *FE can only use within-unit variation. Anything that does not change over time within a unit cannot be estimated.*
>
> If you want to know whether **organic certification** (a fixed farm characteristic)

affects productivity, FE cannot help — it has already "swept out" all between-farm differences.

This exercise asks: **is there an alternative that uses more of the data and can still estimate time-invariant effects?**

---

## 5.2 Step 1 — Extending the Panel to Four Years

We now follow the same five farms across **four years** instead of two. Each farm changes its labour input modestly from year to year.

```
set.seed(42)
true_beta <- 2.5
farms     <- LETTERS[1:5]
years     <- paste0("Year ", 1:4)

# Unobserved soil quality - correlated with labour (same as
 ↪  Exercise 1)
# Farm A is best, E is worst; better farms hire more workers
quality_a <- c(75, 70, 63, 52, 35)

# Labour matrix: rows = farms, columns = years
# Better farms (A, B) expand labour over time; worse farms stay
 ↪  small
labour_mat <- matrix(
  c( 6,  7,  8, 10,   # Farm A
     5,  5,  6,  8,   # Farm B
     4,  5,  4,  6,   # Farm C
     3,  3,  4,  4,   # Farm D
     1,  2,  1,  2),  # Farm E
  nrow = 5, byrow = TRUE
)

# Output = quality + beta*labour + noise
# as.vector(labour_mat) goes column-by-column: all farms in Year 1,
 ↪  then Year 2, …
# matching rep(years, each=5) and rep(farms, times=4)
set.seed(42)
output_mat_a <- matrix(0, nrow = 5, ncol = 4)
```

```
for (t in 1:4) {
  output_mat_a[, t] <- quality_a + true_beta * labour_mat[, t] +
 ↪  rnorm(5, 0, 1.5)
}
output_mat_a <- round(output_mat_a)

df_panel <- tibble(
  Farm   = rep(farms, times = 4),
  Year   = rep(years, each  = 5),
  t      = rep(1:4,   each  = 5),
  Labour = as.vector(labour_mat),
  Output = as.vector(output_mat_a)
)
```

```
kable(
  df_panel |> select(Farm, Year, Labour, Output),
  col.names = c("Farm", "Year", "Labour (workers)", "Output
   ↪  (tonnes)"),
  align = "cccc",
  caption = "Table 1 - Extended panel: 5 farms × 4 years = 20
   ↪  observations"
)
```

Table 5.1: Table 1 — Extended panel: 5 farms × 4 years = 20 observations

| Farm | Year | Labour (workers) | Output (tonnes) |
|---|---|---|---|
| A | Year 1 | 6 | 92 |
| B | Year 1 | 5 | 82 |
| C | Year 1 | 4 | 74 |
| D | Year 1 | 3 | 60 |
| E | Year 1 | 1 | 38 |
| A | Year 2 | 7 | 92 |
| B | Year 2 | 5 | 85 |
| C | Year 2 | 5 | 75 |
| D | Year 2 | 3 | 63 |
| E | Year 2 | 2 | 40 |
| A | Year 3 | 8 | 97 |
| B | Year 3 | 6 | 88 |
| C | Year 3 | 4 | 71 |
| D | Year 3 | 4 | 62 |
| E | Year 3 | 1 | 37 |
| A | Year 4 | 10 | 101 |

| Farm | Year | Labour (workers) | Output (tonnes) |
|---|---|---|---|
| B | Year 4 | 8 | 90 |
| C | Year 4 | 6 | 74 |
| D | Year 4 | 4 | 58 |
| E | Year 4 | 2 | 42 |

```
ggplot(df_panel, aes(x = t, y = Output, colour = Farm, group =
 ↪  Farm)) +
  geom_line(linewidth = 1.1) +
  geom_point(size = 3) +
  scale_x_continuous(breaks = 1:4, labels = years) +
  scale_colour_brewer(palette = "Set2") +
  labs(x = NULL, y = "Output (tonnes)",
       title = "Output trajectories - farms are persistently
        ↪  ordered by soil quality") +
  theme_minimal(base_size = 13)
```

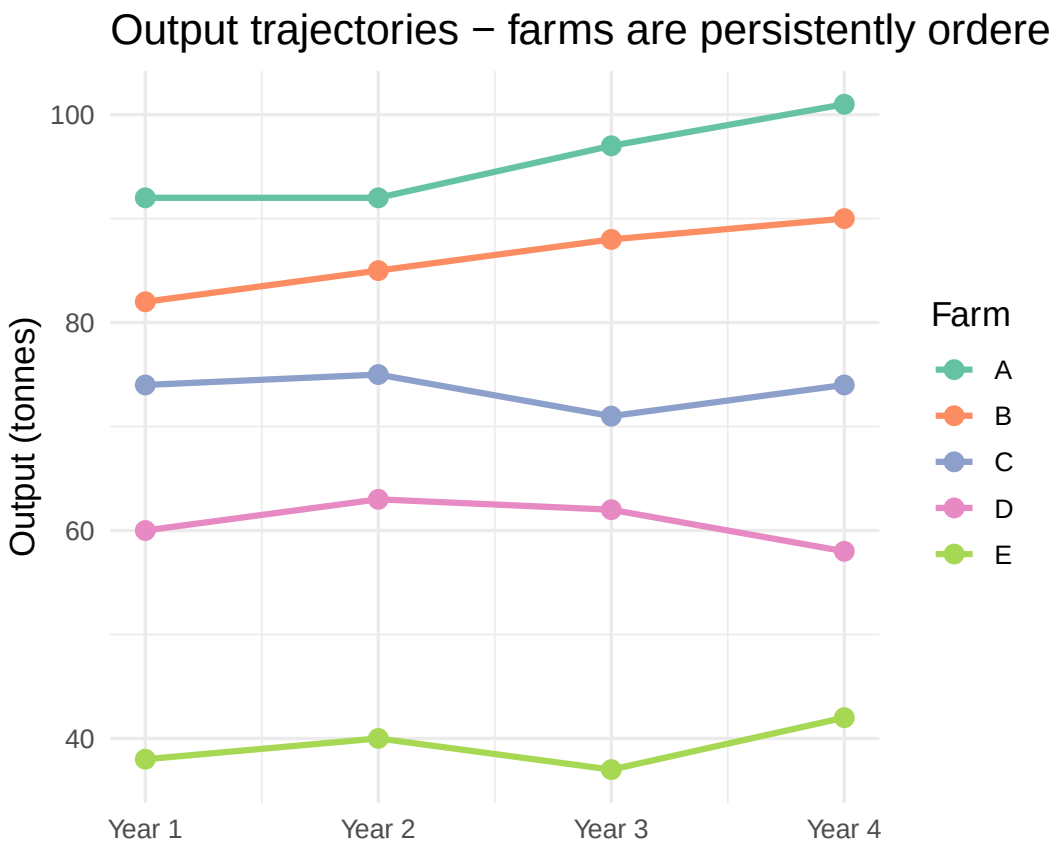


Figure 5.1: Figure 1 — Farm output trajectories over four years

What do you notice about the ordering?

Farm A always leads, Farm E always trails. This persistent ordering reflects **unobserved soil quality** ($\alpha_i$), which is fixed across time. FE controls for this perfectly — but by doing so it also discards all the *between-farm* information.

---

## 5.3 Step 2 — Introducing the Random Effects Model

The **Random Effects (RE) model** offers a middle path. Instead of eliminating $\alpha_i$ by differencing it away, it treats $\alpha_i$ as a **random variable** drawn from a distribution:

$$y_{it} = \mu + \beta\, x_{it} + \underbrace{\alpha_i + u_{it}}_{\varepsilon_{it}}$$

where:

| Symbol | Meaning |
|---|---|
| $\mu$ | Grand intercept |
| $\beta$ | Effect of labour (what we want) |
| $\alpha_i$ | Farm-level random intercept (unobserved heterogeneity) |
| $u_{it}$ | Idiosyncratic noise |

The composite error $\varepsilon_{it} = \alpha_i + u_{it}$ has a **block structure**: observations from the same farm are correlated because they share $\alpha_i$. RE uses Generalised Least Squares (GLS) to exploit this structure, weighting within and between variation optimally.

The critical assumption

RE requires:

$$\mathrm{Cov}(\alpha_i,\, x_{it}) = 0$$

> *The unobserved farm-level characteristic must be **uncorrelated** with the predictor.*

If this fails — as it did in Exercise 1, where better farms (high $\alpha_i$) also hired more workers (high $x_{it}$) — RE inherits the same omitted-variable bias as OLS. FE, by contrast, is consistent regardless.

## 5.4 Step 3 — Scenario A: When RE Fails (Correlated Heterogeneity)

This is our **original farm data** from Exercise 1, now extended to four years. Soil quality is strongly **correlated** with labour: better farms (high $\alpha_i$) consistently hire more workers.

```
pdata_a <- pdata.frame(df_panel, index = c("Farm", "Year"))

mod_ols_a <- lm(Output ~ Labour, data = df_panel)
mod_fe_a  <- plm(Output ~ Labour, data = pdata_a, model = "within")
mod_re_a  <- plm(Output ~ Labour, data = pdata_a, model = "random")

beta_ols_a <- round(coef(mod_ols_a)["Labour"], 3)
beta_fe_a  <- round(coef(mod_fe_a)["Labour"],  3)
beta_re_a  <- round(coef(mod_re_a)["Labour"],  3)
```

```
tibble(
  Estimator        = c("Pooled OLS", "Fixed Effects", "Random
   ↪  Effects"),
  `^ (Labour)`     = c(beta_ols_a, beta_fe_a, beta_re_a),
  `True  `         = true_beta,
  Bias             = c(round(beta_ols_a - true_beta, 3),
                       round(beta_fe_a  - true_beta, 3),
                       round(beta_re_a  - true_beta, 3)),
  `Variation used` = c("Between + within",
                       "Within only",
                       "Within + between (weighted)")
) |>
  kable(align = "lcccc",
        caption = "Table 2 - Scenario A: heterogeneity correlated
         ↪  with labour")
```

Table 5.3: Table 2 — Scenario A: heterogeneity correlated with labour

| Estimator | ^ (Labour) | True | Bias | Variation used |
|---|---|---|---|---|
| Pooled OLS | 7.925 | 2.5 | 5.425 | Between + within |
| Fixed Effects | 2.000 | 2.5 | -0.500 | Within only |

| Estimator | ˆ (Labour) | True | Bias | Variation used |
|---|---|---|---|---|
| Random Effects | 4.065 | 2.5 | 1.565 | Within + between (weighted) |

```
df_panel_a <- df_panel |>
  group_by(Farm) |>
  mutate(Farm_mean_L = mean(Labour), Farm_mean_O = mean(Output)) |>
  ungroup()

ggplot(df_panel_a, aes(x = Labour, y = Output, colour = Farm, group
↪  = Farm)) +
  # FE: parallel within-slope lines, one per farm
  geom_line(aes(y = Farm_mean_O + beta_fe_a * (Labour -
  ↪  Farm_mean_L)),
            linewidth = 1, linetype = "dashed", alpha = 0.7) +
  geom_point(size = 3.5, alpha = 0.85) +
  # Pooled OLS
  geom_abline(intercept = coef(mod_ols_a)[1], slope = beta_ols_a,
              colour = "black", linewidth = 1.1) +
  # RE (single-slope GLS line)
  geom_abline(intercept = coef(mod_re_a)[1], slope = beta_re_a,
              colour = "#E41A1C", linewidth = 1.1, linetype =
               ↪  "dotdash") +
  scale_colour_brewer(palette = "Set2") +
  annotate("text", x = 9.2, y = 100,
           label = paste0("OLS: ˆ = ", beta_ols_a),
           colour = "black", fontface = "bold", size = 3.6) +
  annotate("text", x = 9.2, y = 96,
           label = paste0("RE:  ˆ = ", beta_re_a),
           colour = "#E41A1C", fontface = "bold", size = 3.6) +
  annotate("text", x = 9.2, y = 92,
           label = paste0("FE:  ˆ = ", beta_fe_a,
                          "  ← closest to truth (", true_beta,
                           ↪  ")"),
           colour = "grey30", fontface = "italic", size = 3.6) +
  labs(x = "Labour (workers)", y = "Output (tonnes)",
       title = "Scenario A - RE is biased when Cov( , x )   0",
       subtitle = "Solid = OLS  |  Dot-dash = RE  |  Dashed lines =
        ↪  FE (per farm)",
       colour = "Farm") +
  theme_minimal(base_size = 13)
```

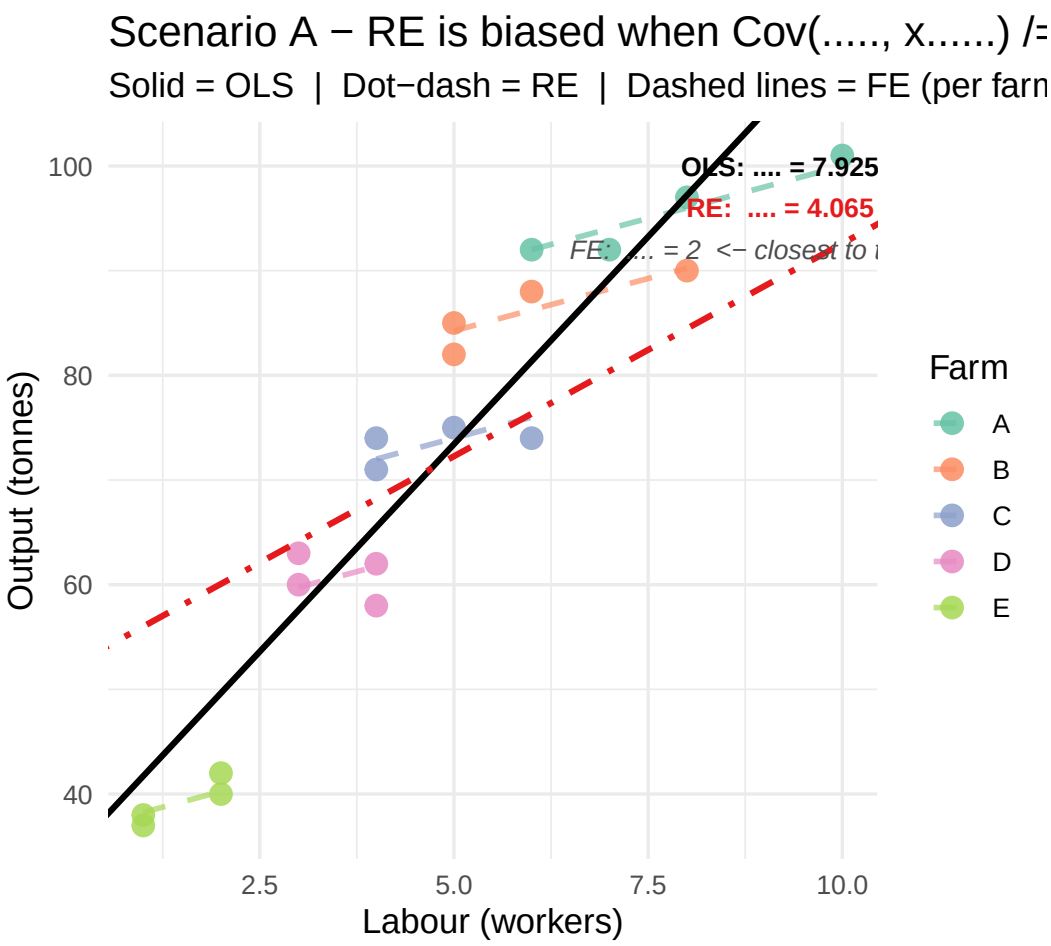


Figure 5.2: Figure 2 — Scenario A: OLS, FE, and RE fits on correlated data

**Interpret the results**

In Scenario A:

- **OLS** is the most biased ($\hat{\beta}_{\text{OLS}} \approx 7.925$) — it ignores farm identity entirely and conflates quality differences with labour effects.
- **RE** is better than OLS but **still biased** ($\hat{\beta}_{\text{RE}} \approx 4.065$): it borrows some of the contaminated between-farm signal because it cannot distinguish "this farm is productive because it has good soil" from "this farm is productive because it has more workers."
- **FE** is closest to the truth ($\hat{\beta}_{\text{FE}} \approx 2$) because it discards between-farm variation and uses only within-farm changes over time.

The RE estimate sits *between* OLS and FE because RE is a weighted average of within and between estimators. When the between estimator is biased (as here),

RE is partially contaminated.

---

## 5.5 Step 4 — Scenario B: When RE Succeeds (Uncorrelated Heterogeneity)

Now imagine a **policy experiment**: the Ministry of Agriculture conducted a randomised trial, **randomly assigning additional workers** to farms. The randomisation breaks the link between soil quality and labour: a farm's soil quality no longer predicts how many workers it has.

In this scenario, soil quality for farms A–E is 55, 50, 60, 75, 45. Notice that Farm D — which has the fewest workers on average — now has the *best* soil. The correlation between quality and mean labour is near zero.

```
# Quality uncorrelated with mean labour: constructed to be
 ↪  orthogonal
# Farm D (low labour) now has the best soil; Farm A (high labour)
 ↪  does not
quality_b <- c(55, 50, 60, 75, 45)

cat("Correlation of quality with farm mean labour:\n")
```

```
Correlation of quality with farm mean labour:
```

```
cat("  Scenario A:", round(cor(quality_a, rowMeans(labour_mat)),
 ↪  3), "\n")
```

```
  Scenario A: 0.977
```

```
cat("  Scenario B:", round(cor(quality_b, rowMeans(labour_mat)),
 ↪  3), "\n")
```

```
  Scenario B: 0.016
```

```
set.seed(77)
output_mat_b <- matrix(0, nrow = 5, ncol = 4)
for (t in 1:4) {
```

```
  output_mat_b[, t] <- quality_b + true_beta * labour_mat[, t] +
 ↪  rnorm(5, 0, 1.5)
}
output_mat_b <- round(output_mat_b)

df_panel_b <- tibble(
  Farm   = rep(farms, times = 4),
  Year   = rep(years, each  = 5),
  t      = rep(1:4,   each  = 5),
  Labour = as.vector(labour_mat),
  Output = as.vector(output_mat_b)
)

pdata_b <- pdata.frame(df_panel_b, index = c("Farm", "Year"))

mod_ols_b <- lm(Output ~ Labour, data = df_panel_b)
mod_fe_b  <- plm(Output ~ Labour, data = pdata_b, model = "within")
mod_re_b  <- plm(Output ~ Labour, data = pdata_b, model = "random")

beta_ols_b <- round(coef(mod_ols_b)["Labour"], 3)
beta_fe_b  <- round(coef(mod_fe_b)["Labour"],  3)
beta_re_b  <- round(coef(mod_re_b)["Labour"],  3)
```

```
tibble(
  Estimator        = c("Pooled OLS", "Fixed Effects", "Random
   ↪  Effects"),
  `~ (Labour)`     = c(beta_ols_b, beta_fe_b, beta_re_b),
  `True `          = true_beta,
  Bias             = c(round(beta_ols_b - true_beta, 3),
                       round(beta_fe_b  - true_beta, 3),
                       round(beta_re_b  - true_beta, 3)),
  `SE advantage`   = c("-",
                       "Less efficient (within only)",
                       "More efficient (within + between)")
) |>
  kable(align = "lcccc",
        caption = "Table 3 - Scenario B: heterogeneity uncorrelated
         ↪  with labour")
```

Table 5.4: Table 3 — Scenario B: heterogeneity uncorrelated with labour

| Estimator | ˆ (Labour) | True | Bias | SE advantage |
|---|---|---|---|---|
| Pooled OLS | 2.461 | 2.5 | -0.039 | — |
| Fixed Effects | 2.949 | 2.5 | 0.449 | Less efficient (within only) |
| Random Effects | 2.940 | 2.5 | 0.440 | More efficient (within + between) |

```
df_panel_b2 <- df_panel_b |>
  group_by(Farm) |>
  mutate(Farm_mean_L = mean(Labour), Farm_mean_O = mean(Output)) |>
  ungroup()

ggplot(df_panel_b2, aes(x = Labour, y = Output, colour = Farm,
 ↪  group = Farm)) +
  geom_line(aes(y = Farm_mean_O + beta_fe_b * (Labour -
   ↪  Farm_mean_L)),
            linewidth = 1, linetype = "dashed", alpha = 0.7) +
  geom_point(size = 3.5, alpha = 0.85) +
  geom_abline(intercept = coef(mod_ols_b)[1], slope = beta_ols_b,
              colour = "black", linewidth = 1.1) +
  geom_abline(intercept = coef(mod_re_b)[1], slope = beta_re_b,
              colour = "#E41A1C", linewidth = 1.1, linetype =
               ↪  "dotdash") +
  scale_colour_brewer(palette = "Set2") +
  annotate("text", x = 9, y = 94,
           label = paste0("OLS: ˆ = ", beta_ols_b),
           colour = "black", fontface = "bold", size = 3.6) +
  annotate("text", x = 9, y = 90.5,
           label = paste0("RE:  ˆ = ", beta_re_b),
           colour = "#E41A1C", fontface = "bold", size = 3.6) +
  annotate("text", x = 9, y = 87,
           label = paste0("FE:  ˆ = ", beta_fe_b,
                          "  (true  = ", true_beta, ")"),
           colour = "grey30", fontface = "italic", size = 3.6) +
  labs(x = "Labour (workers)", y = "Output (tonnes)",
       title = "Scenario B - All three are consistent; RE is more
        ↪  efficient",
       subtitle = "Solid = OLS  |  Dot-dash = RE  |  Dashed lines =
        ↪  FE (per farm)",
       colour = "Farm") +
  theme_minimal(base_size = 13)
```

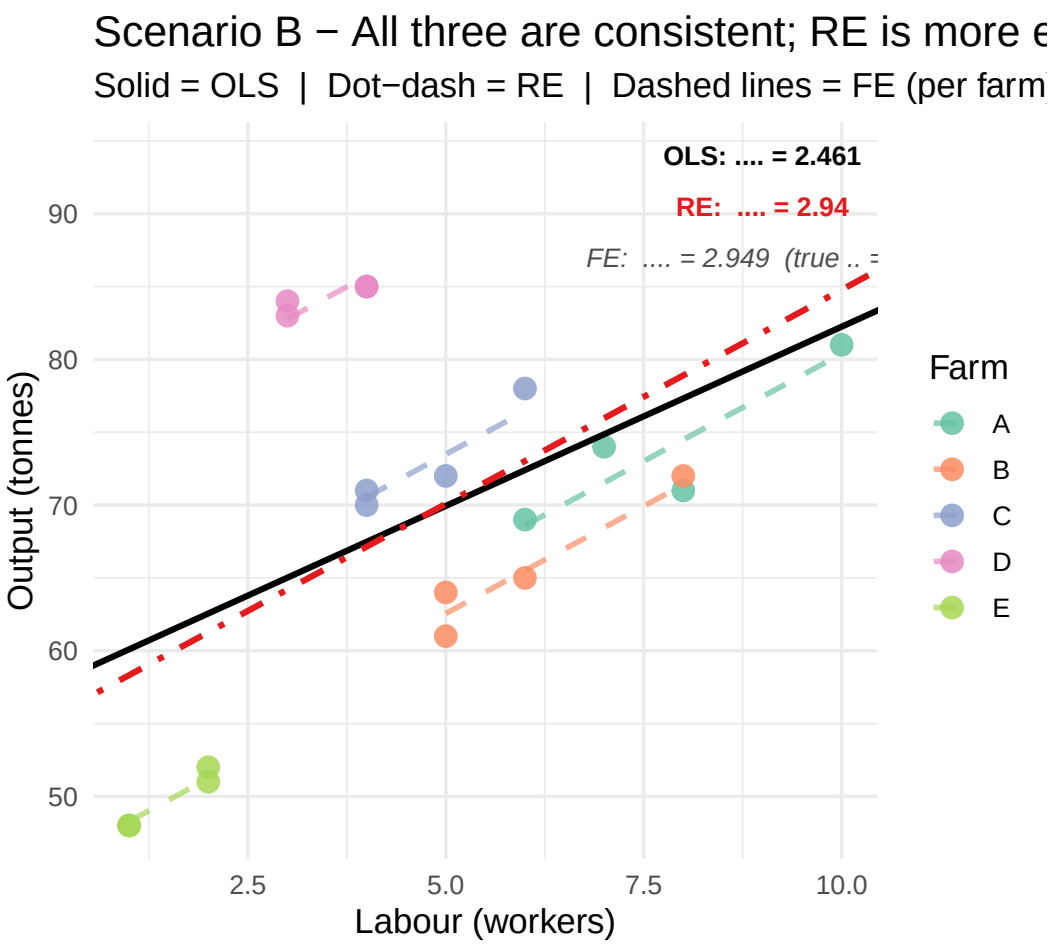


Figure 5.3: Figure 3 — Scenario B: all estimators are consistent when Cov( , x ) = 0

**What changed?**

When $\text{Cov}(\alpha_i, x_{it}) \approx 0$:

- **OLS** is now consistent (no omitted-variable bias) but ignores the error correlation structure — inefficient.
- **FE** is consistent but still discards between-farm information — less precise.
- **RE** is consistent *and* more efficient because it exploits both within and between variation, weighted optimally by the GLS transformation.

*When the RE assumption holds, RE is the preferred estimator — unbiased and more precise than FE.*

## 5.6 Step 5 — The Unique Advantage: Time-Invariant Predictors

One practical reason to prefer RE over FE: **FE cannot estimate the effect of any variable that does not change within a unit over time**.

Suppose the Ministry also collected data on each farm's **ownership type** — whether the farm is family-owned (0) or a cooperative (1). This is fixed across the four years.

```
# Farms A, C, E are family-owned; B and D are cooperatives
ownership <- c(0, 1, 0, 1, 0)
names(ownership) <- farms

df_panel_b <- df_panel_b |>
  mutate(Ownership = ownership[Farm])

pdata_b_own <- pdata.frame(df_panel_b, index = c("Farm", "Year"))

# FE: ownership is collinear with farm dummies - dropped
mod_fe_own <- plm(Output ~ Labour + Ownership,
                  data = pdata_b_own, model = "within")

# RE: can estimate ownership effect
mod_re_own <- plm(Output ~ Labour + Ownership,
                  data = pdata_b_own, model = "random")
```

```
tibble(
  Parameter     = c("Labour ( )", "Ownership - cooperative ( )"),
  `FE estimate` = c(round(coef(mod_fe_own)["Labour"], 3),
                    "- (absorbed by farm FE)"),
  `RE estimate` = c(round(coef(mod_re_own)["Labour"],    3),
                    round(coef(mod_re_own)["Ownership"],  3))
) |>
  kable(align = "lcc",
        caption = "Table 4 - FE cannot identify time-invariant
         ↪  predictors; RE can")
```

Table 5.5: Table 4 — FE cannot identify time-invariant predictors; RE can

| Parameter | FE estimate | RE estimate |
|---|---|---|
| Labour ( ) | 2.949 | 2.941 |
| Ownership — cooperative ( ) | — (absorbed by farm FE) | 9.213 |

**Why FE cannot estimate time-invariant variables**

Fixed effects can be estimated in several ways — including the within-estimator (demeaning), first-differencing, or explicitly including unit dummy variables — but they all produce the same fundamental result: only *within-unit* variation is used for identification.
The most common approach, within-demeaning, subtracts each unit's time-mean from every observation:

$$\tilde{y}_{it} = y_{it} - \bar{y}_i, \quad \tilde{x}_{it} = x_{it} - \bar{x}_i$$

A time-invariant variable like ownership has $x_{it} = x_i$ for all $t$, so after demeaning: $\tilde{x}_{it} = x_i - x_i = 0$. The variable is literally zeroed out — it provides no variation to estimate from. The same logic applies regardless of which estimation approach is used: if a variable never changes within a unit, no within-unit estimator can identify its effect.
RE applies only a *partial* transformation using the estimated intra-class correlation, leaving some between-unit signal intact. That between-unit signal identifies $\beta_2$.

> *This is a key practical reason RE is preferred in research designs that include important time-invariant covariates (e.g., country-level institutions, individual characteristics such as gender or ethnicity, historical conditions).*

---

## 5.7 Step 6 — The Hausman Test: Choosing Between FE and RE

If you are uncertain whether $\text{Cov}(\alpha_i, x_{it}) = 0$, the **Hausman test** gives a formal answer.

**Logic:**

- Under $H_0$: RE assumption holds → both FE and RE are consistent, but RE is more efficient.
- Under $H_1$: RE assumption fails → FE is consistent; RE is biased.

The test statistic compares the FE and RE coefficient vectors. A large difference (relative to the sampling variance) rejects $H_0$ and points to FE.

$$H = (\hat{\beta}_{\text{FE}} - \hat{\beta}_{\text{RE}})^\top \left[\text{Var}(\hat{\beta}_{\text{FE}}) - \text{Var}(\hat{\beta}_{\text{RE}})\right]^{-1} (\hat{\beta}_{\text{FE}} - \hat{\beta}_{\text{RE}}) \sim \chi^2(k)$$

```
ht_a <- phtest(mod_fe_a, mod_re_a)   # Scenario A: expect rejection
ht_b <- phtest(mod_fe_b, mod_re_b)   # Scenario B: expect no
 ↪  rejection
```

```
tibble(
  Scenario       = c("A - Cov( , x )   0\n(quality correlated with
   ↪  labour)",
                     "B - Cov( , x )   0\n(randomised labour
                      ↪  assignment)"),
  ` ² statistic` = c(round(ht_a$statistic, 3), round(ht_b$statistic,
   ↪  3)),
  df             = c(ht_a$parameter,            ht_b$parameter),
  `p-value`      = c(round(ht_a$p.value, 4),    round(ht_b$p.value,
   ↪  4)),
  Decision       = c("Reject H  → use FE",
                     "Fail to reject H  → RE preferred")
) |>
  kable(align = "lcccc",
        caption = "Table 5 - Hausman test results for both
         ↪  scenarios")
```

Table 5.6: Table 5 — Hausman test results for both scenarios

| Scenario | $^2$ statistic | df | p-value | Decision |
|---|---|---|---|---|
| A — Cov( , x )  0 (quality correlated with labour) | 8.137 | 1 | 0.0043 | Reject H  → use FE |
| B — Cov( , x )  0 (randomised labour assignment) | 0.010 | 1 | 0.9202 | Fail to reject H  → RE preferred |

⚠ Interpreting the Hausman test carefully

The Hausman test has real limitations in practice:

1. **Low power in short panels.** With few time periods or few units, the test may fail to detect genuine correlation. A non-rejection does not guarantee RE is safe.
2. **It only tests one direction.** The test checks whether RE is *as consistent as* FE. But if FE itself is the wrong model (e.g., non-linear effects, wrong functional form), the test can give misleading guidance.

3. **Practical knowledge matters.** If your theory or prior work strongly suggests that unobserved heterogeneity correlates with your predictor, prefer FE regardless of the test result.

   *Use the Hausman test as one input, not the final word.*

---

## 5.8 Step 7 — Side-by-Side: All Estimators Compared

```
estimates_long <- tibble(
  Estimator = rep(c("Pooled OLS", "Fixed Effects", "Random
   ↪  Effects"), 2),
  Scenario  = rep(c("A: Cov( , x )  0\n(FE needed)",
                    "B: Cov( , x )  0\n(RE preferred)"), each = 3),
  Beta      = c(beta_ols_a, beta_fe_a, beta_re_a,
                beta_ols_b, beta_fe_b, beta_re_b)
) |>
  mutate(
    Estimator = factor(Estimator,
                       levels = c("Pooled OLS", "Random Effects",
                        ↪  "Fixed Effects"))
  )

ggplot(estimates_long,
       aes(x = Beta, y = Estimator, colour = Estimator, shape =
        ↪  Estimator)) +
  geom_vline(xintercept = true_beta, linetype = "dashed",
             colour = "grey40", linewidth = 0.8) +
  geom_point(size = 5) +
  annotate("text", x = true_beta + 0.05, y = 0.55,
           label = paste0("True   = ", true_beta),
           colour = "grey40", size = 3.5, hjust = 0) +
  scale_colour_manual(values = c("Pooled OLS"    = "#666666",
                                 "Fixed Effects"  = "#4DAF4A",
                                 "Random Effects" = "#E41A1C")) +
  scale_shape_manual(values  = c("Pooled OLS"    = 15,
                                 "Fixed Effects"  = 17,
                                 "Random Effects" = 16)) +
  facet_wrap(~ Scenario, ncol = 2) +
```

```
  labs(x = "Estimated   (Labour → Output)", y = NULL,
       title = "Proximity to the true   across scenarios and
        ↪  estimators") +
  theme_minimal(base_size = 13) +
  theme(legend.position = "bottom", legend.title = element_blank())
```

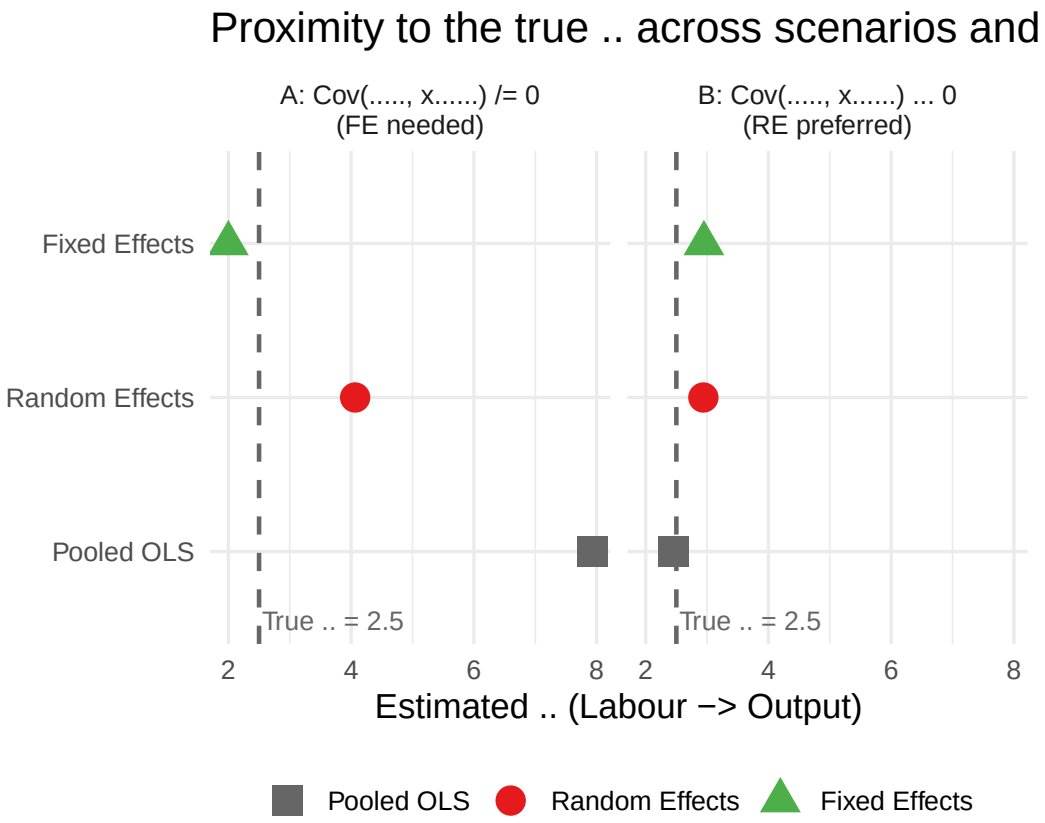


Figure 5.4: Figure 4 — Estimates from all three models across both scenarios

```
tibble(
  Estimator = c("Pooled OLS", "Fixed Effects (within)", "Random
   ↪  Effects (GLS)"),
  `Sc A ^^`  = c(beta_ols_a, beta_fe_a, beta_re_a),
  `Sc A Bias`= c(round(beta_ols_a - true_beta, 2),
                 round(beta_fe_a  - true_beta, 2),
                 round(beta_re_a  - true_beta, 2)),
  `Sc B ^^`  = c(beta_ols_b, beta_fe_b, beta_re_b),
  `Sc B Bias`= c(round(beta_ols_b - true_beta, 2),
                 round(beta_fe_b  - true_beta, 2),
```

```
                round(beta_re_b  - true_beta, 2))
) |>
  kable(align = "lcccc",
        col.names = c("Estimator",
                      "Sc A: ^", "Sc A: Bias",
                      "Sc B: ^", "Sc B: Bias"),
        caption = paste0("Table 6 - All estimators: true   = ",
         ↪  true_beta))
```

Table 5.7: Table 6 — All estimators: true  = 2.5

| Estimator | Sc A: ^ | Sc A: Bias | Sc B: ^ | Sc B: Bias |
|---|---|---|---|---|
| Pooled OLS | 7.925 | 5.42 | 2.461 | -0.04 |
| Fixed Effects (within) | 2.000 | -0.50 | 2.949 | 0.45 |
| Random Effects (GLS) | 4.065 | 1.57 | 2.940 | 0.44 |

Reading Figure 4

- **Left panel (Scenario A):** Only FE (green triangle) lands near the dashed line. OLS and RE are biased upward because they absorb between-farm variation contaminated by soil quality.
- **Right panel (Scenario B):** All three are close to the true $\beta$. RE (red circle) and FE are both consistent; RE's advantage here is efficiency (smaller standard errors), not bias correction. OLS is also close because randomisation removed the confounding.

---

## 5.9 Step 8 — Decision Framework

Use the table below as a starting point when choosing between FE and RE in practice.

| Question | FE | RE |
|---|---|---|
| Is $\text{Cov}(\alpha_i, x_{it}) = 0$ plausible? | Not required | Required |
| Can I estimate time-invariant predictors? | No | Yes |

| Question | FE | RE |
|---|---|---|
| Can I estimate effects of variables with little within-unit variation? | Poorly | Better |
| Is efficiency a concern (small $T$, large $N$)? | Less efficient | More efficient |
| Hausman test rejects $H_0$? | Preferred | Inconsistent |
| Research design uses random assignment? | Either works | RE preferred |
| Observational data with likely self-selection? | Preferred | Only if assumption is defensible |

A researcher's heuristic

> *"Start with FE. It is the conservative, credible choice. Use RE only when you have a strong theoretical or empirical reason to believe that unobserved heterogeneity is uncorrelated with your regressors — and even then, always report the Hausman test and a FE robustness check side by side."*

---

## 5.10 Summary

In this exercise you discovered:

1. **Fixed Effects** eliminates unobserved heterogeneity by exploiting within-unit variation over time — but at the cost of discarding between-unit information and losing all time-invariant predictors.
2. **Random Effects** treats heterogeneity as part of the error and uses GLS to efficiently combine within and between variation — but requires the strong assumption $\text{Cov}(\alpha_i, x_{it}) = 0$.
3. **The Hausman test** is the standard diagnostic for choosing between FE and RE. A significant result favours FE; a non-significant result makes RE defensible, though not guaranteed safe.
4. **When the RE assumption holds**, RE dominates: unbiased, efficient, and able to estimate time-invariant predictors that FE cannot touch.

5. **In practice**, begin with FE for credibility, and treat RE as a robustness check or primary specification only when the research design (e.g., random assignment) justifies it.

**!** Extension: The Mundlak Device

A powerful synthesis due to Mundlak (1978): include the **within-unit means of all time-varying regressors** as additional controls in the RE model:

$$y_{it} = \mu + \beta\, x_{it} + \gamma\, \bar{x}_i + \alpha_i^* + u_{it}$$

This "Mundlak device" soaks up the correlation between $\alpha_i$ and $x_{it}$, making $\hat{\beta}$ numerically equivalent to the FE estimator — while the RE framework still allows time-invariant variables to enter through $\bar{x}_i$. It is increasingly recommended as a best-of-both-worlds specification, and provides a further diagnostic: if $\hat{\gamma} \neq 0$, the RE assumption was violated.

# 6 Summary

This workshop introduced the core logic and methods of longitudinal data analysis using R. The sections below provide a brief conceptual recap of the three main modelling approaches covered: OLS, Fixed Effects, and Random Effects.

## 6.1 1. Ordinary Least Squares (OLS)

**Equation**

$$y_{it} = \beta_0 + \beta_1 x_{it} + u_{it}$$

- $i$ = individual (person, firm, country)
- $t$ = time
- $u_{it}$ = error term

**Key idea**

OLS ignores the panel structure. It treats all observations as independent.

**Unique features**

- One common intercept $\beta_0$ for all units
- No explicit handling of unobserved individual heterogeneity
- Assumes:

$$\text{Cov}(x_{it}, u_{it}) = 0$$

**Problem in panel data**

If there are unobserved individual effects ($\alpha_i$) that affect $y_{it}$ and are correlated with $x_{it}$, then: - They get absorbed into $u_{it}$ - → Omitted variable bias

---

## 6.2 2. Fixed Effects (FE)

**Equation**

$$y_{it} = \alpha_i + \beta_1 x_{it} + u_{it}$$

- $\alpha_i$ = individual-specific intercept (fixed effect)

**Equivalent (demeaned / "within" form)**

$$y_{it} - \bar{y}_i = \beta_1 (x_{it} - \bar{x}_i) + (u_{it} - \bar{u}_i)$$

**Key idea**

Each unit gets its own intercept, and these intercepts can be correlated with $x_{it}$.

**Unique features**

- Controls for all time-invariant unobserved heterogeneity
- Uses only within-individual variation
- Allows:

$$\text{Cov}(x_{it}, \alpha_i) \neq 0$$

**Consequences**

- Cannot estimate coefficients on time-invariant variables
- More robust than OLS in panel settings

---

## 6.3 3. Random Effects (RE)

**Equation**

$$y_{it} = \beta_0 + \beta_1 x_{it} + \alpha_i + u_{it}$$

- $\alpha_i$ = random individual effect

**Error structure (composite error term)**

$$\varepsilon_{it} = \alpha_i + u_{it}$$

Individual-specific effects are captured by this composite error term. Rather than estimating a separate intercept for each unit, RE assumes that individual intercepts are drawn from a random distribution of possible intercepts — meaning $\alpha_i$ is not a fixed parameter but a random variable with some distribution.

**Key assumption**

$$\text{Cov}(x_{it}, \alpha_i) = 0$$

**Key idea**

The individual effect is treated as a random variable, not a fixed parameter.

**Unique features**

- Uses both:
  - within variation (like FE)
  - between variation (differences across individuals)
- Estimated via GLS (Generalized Least Squares)
- Implicitly applies partial pooling

**Consequences**

- Can estimate time-invariant variables
- More efficient than FE if assumptions hold
- Biased if $\alpha_i$ is correlated with $x_{it}$

---

## 6.4 Side-by-Side Summary

| Feature | OLS | Fixed Effects | Random Effects |
|---|---|---|---|
| Intercept | Single $\beta_0$ | $\alpha_i$ for each unit | $\beta_0 + \alpha_i$ |
| Unobserved heterogeneity | Ignored | Controlled (fixed) | Modeled (random) |
| $\text{Corr}(x_{it}, \alpha_i)$ | Not allowed | Allowed | Not allowed |
| Variation used | All (naively) | Within only | Within + between |

| Feature | OLS | Fixed Effects | Random Effects |
|---|---|---|---|
| Time-invariant variables | Yes | No | Yes |
| Bias risk | High for panel data | Low | Depends on assumption |

---

## 6.5 Model Comparison: Simpson's Paradox

The figure below illustrates how each modelling strategy handles unobserved individual heterogeneity, using a synthetic dataset designed to produce **Simpson's Paradox**: the aggregate (OLS) trend is negative, yet the true within-unit relationship is positive.

Each panel shows the same six simulated units (coloured points). The lines show what each estimator recovers, and each panel is annotated with its slope estimate $\hat{\beta}_1$:

- **OLS** — a single regression line ignoring group membership; the negative between-unit confound dominates.
- **Fixed Effects** — parallel within-unit lines (common slope, unit-specific intercepts); the true positive within-unit relationship is recovered.
- **Random Effects** — GLS estimator that uses a weighted mix of within- and between-unit variation. When $\text{Cov}(x_{it}, \alpha_i) \neq 0$ (as here), the slope is biased toward the OLS estimate — the key trade-off versus FE.

```
if (!requireNamespace("lme4", quietly = TRUE))
 ↪  install.packages("lme4")

library(tidyverse)
library(lme4)
library(RColorBrewer)

set.seed(42)

n_units <- 6
n_obs   <- 5   # few obs per unit →    0.70, making RE visibly
 ↪  distinct from FE

# Simpson's paradox: between-unit trend is negative, within-unit
 ↪  trend is positive.
```

```
# Smaller unit spread + higher within noise keeps   well below 1, so
 ↪  RE   FE.
unit_x0 <- seq(6.5, 3.5, length.out = n_units)  # unit mean x (high
 ↪  → low)
unit_y0 <- seq(3.5, 6.5, length.out = n_units)  # unit mean y (low
 ↪  → high)

df <- map_dfr(seq_len(n_units), function(i) {
  x <- unit_x0[i] + rnorm(n_obs, 0, 0.70)
  y <- unit_y0[i] + 0.85 * (x - unit_x0[i]) + rnorm(n_obs, 0, 0.80)
  tibble(unit = factor(paste0("Unit ", i)), x = x, y = y)
})

unit_levels <- levels(df$unit)
xs <- seq(min(df$x) - 0.2, max(df$x) + 0.2, length.out = 80)

# ---- OLS ----
ols_fit   <- lm(y ~ x, data = df)
ols_slope <- as.numeric(coef(ols_fit)["x"])
ols_lines <- tibble(
  x = xs, y = as.numeric(coef(ols_fit)[1]) + ols_slope * xs,
  unit = "Overall", model = "OLS"
)

# ---- Fixed Effects ----
fe_fit   <- lm(y ~ x + unit, data = df)
fe_slope <- as.numeric(coef(fe_fit)["x"])
fe_ints  <- setNames(
  c(coef(fe_fit)["(Intercept)"],
    coef(fe_fit)["(Intercept)"] + coef(fe_fit)[paste0("unit",
     ↪  unit_levels[-1])]),
  unit_levels
)
fe_lines <- map_dfr(unit_levels, function(u) {
  tibble(x = xs, y = as.numeric(fe_ints[u]) + fe_slope * xs, unit
   ↪  = u, model = "Fixed Effects")
})

# ---- Random Effects (random intercept, ML) ----
# With our data, Cov(x_it,  _i)   0, so RE is biased: its slope will
 ↪  sit between
# the FE slope (within only) and the OLS slope (within + between
 ↪  confounded).
```

```r
re_fit   <- lmer(y ~ x + (1 | unit), data = df, REML = FALSE)
re_slope <- as.numeric(fixef(re_fit)["x"])
re_int   <- as.numeric(fixef(re_fit)["(Intercept)"])
re_blup  <- data.frame(
  unit = rownames(ranef(re_fit)$unit),
  ran  = ranef(re_fit)$unit[[1]]
)
re_lines <- map_dfr(unit_levels, function(u) {
  ri <- re_blup$ran[re_blup$unit == u]
  tibble(x = xs, y = re_int + ri + re_slope * xs, unit = u, model =
   ↪  "Random Effects")
})

# ---- Combine ----
model_levels <- c("OLS", "Fixed Effects", "Random Effects")

lines_all <- bind_rows(ols_lines, fe_lines, re_lines) %>%
  mutate(model = factor(model, levels = model_levels))

scatter_all <- map_dfr(model_levels, function(m) {
  df %>% mutate(model = factor(m, levels = model_levels))
})

# ---- Slope annotations (one per panel) ----
slope_ann <- tibble(
  model = factor(model_levels, levels = model_levels),
  slope = c(ols_slope, fe_slope, re_slope),
  label = paste0("hat(beta)[1] == ", round(c(ols_slope, fe_slope,
   ↪  re_slope), 2))
)

# ---- Colours ----
unit_colours <- c(
  setNames(brewer.pal(n_units, "Set2"), unit_levels),
  "Overall" = "black"
)

# ---- Plot ----
ggplot() +
  geom_point(
    data = scatter_all,
    aes(x = x, y = y, colour = unit),
    size = 2.0, alpha = 0.70
```

```
  ) +
  geom_line(
    data = lines_all %>% filter(unit != "Overall"),
    aes(x = x, y = y, colour = unit),
    linewidth = 0.85
  ) +
  geom_line(
    data = lines_all %>% filter(unit == "Overall"),
    aes(x = x, y = y),
    colour = "black", linewidth = 1.4, linetype = "dashed"
  ) +
  geom_label(
    data  = slope_ann,
    aes(label = label),
    x = Inf, y = -Inf, hjust = 1.08, vjust = -0.5,
    size = 3.6, parse = TRUE,
    fill = alpha("white", 0.85), colour = "grey20", label.size =
     ↪  0.3
  ) +
  scale_colour_manual(values = unit_colours) +
  facet_wrap(
    ~model, nrow = 1,
    labeller = as_labeller(c(
      "OLS"            = "Pooled OLS\n(single line, ignores
       ↪  group/unit structure)",
      "Fixed Effects"  = "Fixed Effects \neach unit i: own
       ↪  intercept, common slope",
      "Random Effects" = "Random Effects\n(shrunk intercepts,
       ↪  common slope: within + between)"
    ))
  ) +
  labs(
    title    = "Regression Models for Panel Data",
    subtitle = "Example: Simpson's Paradox",
    x      = "Predictor (x)",
    y      = "Outcome (y)",
    colour = NULL
  ) +
  theme_minimal(base_size = 11) +
  theme(
    strip.text       = element_text(face = "bold", size = 10),
    legend.position  = "bottom",
    legend.key.width = unit(1.5, "lines"),
```

```
    plot.title       = element_text(face = "bold", size = 13),
    plot.subtitle    = element_text(size = 10, colour = "grey40"),
    panel.grid.minor = element_blank()
  )
```

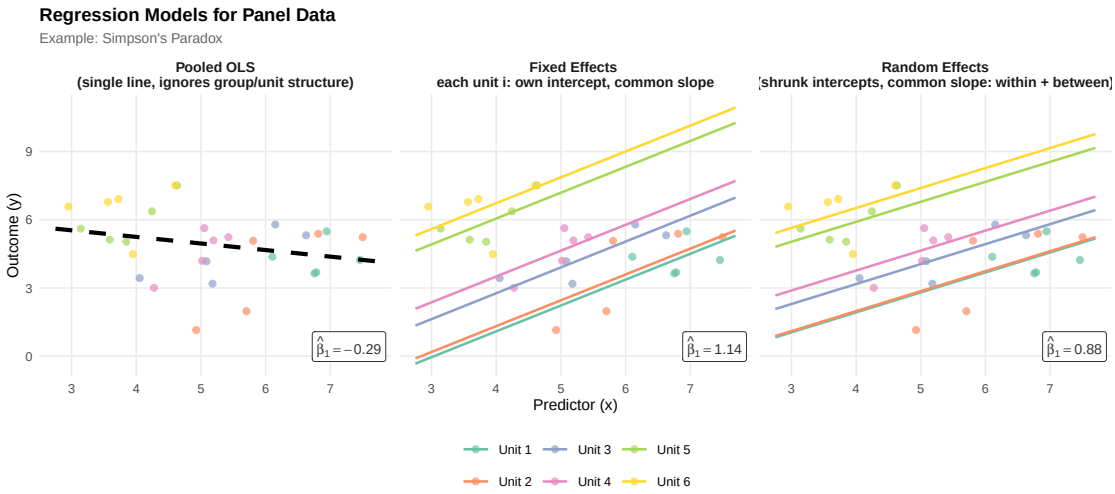


Figure 6.1: Simpson's Paradox in panel data. OLS recovers the spurious negative aggregate trend; unit-level methods reveal the true positive within-unit relationship. Random Effects partially pools intercepts toward the grand mean relative to FE.

# How to Cite

You can cite this tutorial book as:

Jarnach, Clemens. 2026. Longitudinal Data Analysis with R: A Practical Introduction from Theory to Code. Oxford. https://github.com/clemensjarnach

# References


Angrist, J. D., & Pischke, J.-S. (2008). *Mostly harmless econometrics: An empiricist's companion*. Princeton University Press.

Croissant, Y., & Millo, G. (2018). *Panel data econometrics with R* (1st ed.). Wiley.

Fieldhouse, E., Green, J., Evans, G., Mellon, J., Prosser, C., Bailey, R., deGeus, R., Schmitt, H., & van der Eijk, C. (2023). *British Election Study 2014–2023 Internet Panel Waves 1–21* [Data series]. UK Data Service.

Firebaugh, G. (2008). *Seven rules for social research*. Princeton University Press. https://doi.org/10.1515/9780691190433

Fletcher, R., Andı, S., Badrinathan, S., Eddy, K. A., Kalogeropoulos, A., Mont'Alverne, C., Robertson, C. T., Ross Arguedas, A., Schulz, A., Toff, B., & Nielsen, R. K. (2024). The link between changing news use and trust: Longitudinal analysis of 46 countries. *Journal of Communication*, jqae044. https://doi.org/10.1093/joc/jqae044

Hsiao, C. (2014). *Analysis of panel data* (3rd ed.). Cambridge University Press.

Huntington-Klein, N. (2025). *The effect: An introduction to research design and causality* (2nd ed.). CRC Press.

Kearney, M. (2017). Cross-lagged panel analysis. In M. Allen (Ed.), *The SAGE encyclopedia of communication research methods* (Vol. 4). SAGE Publications.

Singer, J. D., & Willett, J. B. (2003). *Applied longitudinal data analysis: Modeling change and event occurrence*. Oxford University Press.

Wooldridge, J. M. (2014). *Introduction to econometrics* (Europe, Middle East & Africa ed.). Cengage Learning.

# Appendix

## Panel Model Comparison

The figure below illustrates four regression strategies for panel data, all fitted to the same simulated dataset of four groups (shown in different colours). The models differ substantially in how they account for group structure, and consequently in how well they recover the underlying within-group relationship.


Figure 6.2: Regression models for grouped/panel data. Each panel uses the same simulated dataset of four groups (colours). Pooled OLS ignores group structure and recovers a badly biased slope; Fixed Effects removes the confound by demeaning within groups; Random Effects partially pools information across groups; Mixed Effects allows both intercepts and slopes to vary by group.

**Pooled OLS** treats all observations as if they came from a single population, ignoring group membership entirely. When groups differ systematically in both the outcome and the predictor, the resulting slope is confounded and can be severely biased — a classic manifestation of Simpson's Paradox.

**Fixed Effects (FE)** addresses this by demeaning within each group (or equivalently, including a group-level intercept for every unit). This removes all time-invariant unobserved heterogeneity, so only within-group variation identifies the slope. It is the primary method covered in this workshop.

**Random Effects (RE)** assumes that group-level deviations are drawn from a common distribution and are uncorrelated with the predictors. This is a stronger assumption than FE, but it allows time-invariant predictors to be included and can be more efficient when

the assumption holds. The Hausman test is the standard tool for choosing between FE and RE.

**Mixed Effects (ME)** — also called multilevel or hierarchical models — goes one step further by allowing both intercepts *and* slopes to vary across groups. This is a rich and flexible framework that sits beyond the scope of this introductory workshop.

## Presentation Slides

Below are the slides for this module. You can view them in full screen here.

## Notation Reference

This appendix provides a quick reference for common mathematical and statistical symbols used throughout the workshop materials, along with their LaTeX commands for use in write-ups and presentations.

| Symbol | LaTeX Command | Common Statistical Use |
|---|---|---|
| | `\alpha` | Significance level / Type I error rate |
| | `\beta` | Regression coefficients / Type II error rate |
| | `\gamma` | Gamma distribution / Euler-Mascheroni constant |
| | `\delta` | Small change / Dirac delta function |
| | `\varepsilon` | Error term (residuals) in regression (variant form) |
| | `\epsilon` | Error term (residuals) in regression |
| | `\mu` | Population mean |
| | `\pi` | Population proportion / Probability of success |
| | `\rho` | Population correlation coefficient |
| | `\sigma` | Population standard deviation |
| | `\tau` | Kendall's Tau / Precision (inverse variance) |
| | `\theta` | General unknown parameter |
| $\Sigma$ | `\sum` | Summation operator |
| | `\prod` | Product operator |

# Dummy Variable Trap

The **Dummy Variable Trap** is a scenario in which independent variables are highly collinear—specifically, when one variable can be predicted perfectly from the others. In technical terms, it is a case of **perfect multicollinearity**.

Here is how it happens and how to avoid it.

## How the Trap is Set

When you have a categorical variable (like "Gender" or "Season") and you want to include it in a regression, you create dummy variables (0 or 1).

If you have a category like **Season** (Spring, Summer, Fall, Winter) and you include a dummy for **all four** categories *plus* an intercept ($\beta_0$) in your model, you have entered the trap.

## The Math Behind the Failure

A standard regression model includes a constant (the intercept), which is essentially a column of 1s. If you add up the values of all your dummy variables for a specific observation, they will always equal 1 (because every observation must belong to exactly one category).

$$Spring + Summer + Fall + Winter = 1$$

Since the sum of your dummies equals your intercept, the computer cannot mathematically distinguish between the effect of the intercept and the combined effect of the dummies. This makes the matrix non-invertible, and your software will either: 1. Crash/return an error. 2. Automatically drop one of the variables.

## How to Escape the Trap: The $(n-1)$ Rule

To avoid perfect multicollinearity, you must always include **one fewer dummy variable than there are categories**.

- **If you have 2 categories (Male/Female):** Include 1 dummy.
- **If you have 4 categories (Seasons):** Include 3 dummies.
- **If you have 50 cities:** Include 49 dummies.

## The "Reference" Group

The category you leave out becomes the **Reference Group** (or Baseline). * If you leave out "Winter," the coefficients for Spring, Summer, and Fall will represent the difference in the outcome *compared to Winter*. * The Intercept ($\beta_0$) will represent the average value for Winter.

In your previous R code example: `lm(crime_rate ~ factor(city)...)` R automatically handles this for you by picking the first city (alphabetically) and dropping it to serve as the reference group, effectively saving you from the trap!

Firebaugh, Glenn. 2008. *Seven Rules for Social Research*. Princeton University Press. https://doi.org/10.1515/9780691190433.

Huntington-Klein, Nick. 2025. *The Effect : An Introduction to Research Design and Causality*. Second edition. Boca Raton, FL ; CRC Press.

Wooldridge, Jeffrey M. 2014. *Introduction to Econometrics*. Europe, Middle East and Africa edition. Andover.