linearmodels

Panel data: the same countries or firms, followed over time

Python cheatsheet · MATH60033A · Quantitative Methods in IB · used in session 6

```
from linearmodels.panel import PanelOLS,
                               RandomEffects
```

a panel needs a two-level index: entity, then time

1 · Shape the panel

Same structure as any DataFrame, plus an index that says who and when.

long format: one row = one entity in one year

Austria	2010		
Austria	2011		
Austria	2012		
Belgium	2010		
Belgium	2011		
Belgium	2012		
Canada	2010		
Canada	2011		
Canada	2012		
entity	**time**	y, x	

`set_index(["country", "year"])`

tells linearmodels which rows are the same entity

missing years? fine: the panel is unbalanced

`panel = d.dropna(subset=cols).set_index(["country", "year"])`

drop the gaps in what you use, then index by **entity, then time** — in that order.

`y = panel["grossdebt"]`

the outcome, a Series with the two-level index.

`X = panel[["busfreedom", "finfreedom", "invfreedom"]]`

the regressors, a DataFrame with the same index.

`panel.index.get_level_values(0).nunique()`

how many entities (here 16 countries; 23 years; 314 rows).

`panel.groupby(level=0).size()`

rows per entity: is the panel balanced?

> Index in the wrong order — time first — and every "country effect" becomes a year effect.

2 · Fixed effects — PanelOLS

`fixed = PanelOLS(y, X, entity_effects=True).fit()`

one intercept per country, absorbed rather than printed. Each slope now compares a country **with itself** over time.

`PanelOLS(y, X, entity_effects=True, time_effects=True)`

two-way: also one intercept per year, which removes shocks every country shared.

`PanelOLS.from_formula("y ~ x1 + x2 + EntityEffects", data=panel)`

the same model, written as a formula.

`print(fixed)`

the table: coefficients, standard errors, R^2 within / between / overall.

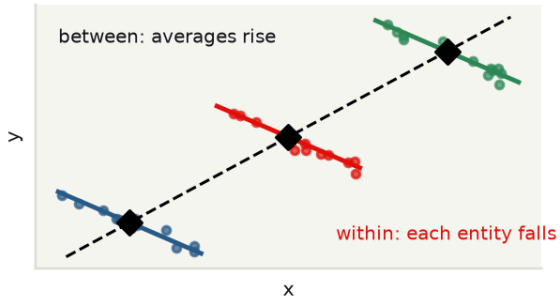
`fixed.params · fixed.std_errors · fixed.pvalues`

the numbers, as Series.

> A fixed-effects slope answers a **within** question: when *this* country's x rises, what happens to *its* y?

3 · Within against between

An illustration: three entities, each followed over time (coloured lines); the diamonds are their averages.



`fixed.rsquared_within · fixed.rsquared_between`

how much of the variation **inside** countries, and **across** country averages, the model accounts for.

> Anything constant within a country — geography, legal origin — is absorbed: fixed effects cannot estimate its effect.

4 · Random effects, and choosing

fixed effects	**random effects**
one intercept per entity, absorbed: only change within an entity is used	intercepts drawn at random, assumed unrelated to x: more precise, riskier

Hausman: do they disagree beyond noise?

yes (small p) → keep fixed effects

`random_ = RandomEffects(y, X).fit()`

treats the country intercepts as random draws: uses within **and** between variation.

`diff = fixed.params - random_.params`

the Hausman test, as in session 6: compare the two sets of slopes…

`vdiff = fixed.cov - random_.cov`

…against the difference of their covariance matrices…

`H = diff @ np.linalg.pinv(vdiff) @ diff`

…a χ^2 with one degree of freedom per slope.

```
p = stats.chi2.sf(H, len(diff))
```

> Choose on the assumption, never on which model gives the significant answer.

5 · Standard errors that respect the panel

`.fit(cov_type="clustered", cluster_entity=True)`

years of the same country are not independent: cluster by entity. The estimates do not move; the standard errors do — here they roughly double.

`.fit(cov_type="robust")`

heteroskedasticity-robust only, without clustering.

6 · When linearmodels complains

`ValueError: ... MultiIndex`

the data has no (entity, time) index: `set_index` first.

`AbsorbingEffectError`

a regressor never changes within an entity: fixed effects have absorbed it. Drop it, or pass `drop_absorbed=True` to `PanelOLS`.

`matrix not positive definite (Hausman)`

use `np.linalg.pinv`, as the deck does, not `inv`.