RESOURCE SCORE

Res-1: Land resource intensity (from `resource_score1_coal_all.csv`)

Field	Units	Direction	Meaning
`Land_Use_Intensity_ha_per_Mt`	ha/Mt	Higher=worse	Land used per Mt coal (surface mining intensity).
`Plants_Count`	#	Higher=worse	Count of coal plants (fleet size proxy).
`R_P_ratio_years`	years	Higher=worse (lock-in)	Reserves divided by annual production; long R/P ⇒ longer lock-in.

Res-1 composite: mean of normalized sub-scores (invert none by default here).

Res-2: Water (from `resource_score2_coal_all.csv`)

Field	U	Direction	Meaning
`Water_Use_m3`	m	Higher=worse	Total water consumption for coal.

Share_Global_Water_Use	%	Higher=worse	Share of global coal water footprint.

Res-2 composite: mean of normalized sub-scores.

Res-3: Extraction & waste (from resource_score3_coal_all.csv)

Field	Units	Direction	Meaning
Extraction_Share	%	Higher=worse	Country share of global coal extraction.
Coal_Ash_Waste_t	tonnes	Higher=worse	Total ash waste.
Waste_to_Use_Ratio	t/t	Higher=worse	Ash per tonne of coal (quality proxy).

Res-3 composite: mean of normalized sub-scores.

Resource composite (family)

Resource_Composite = mean(**Res-1**, **Res-2**, **Res-3**).

How the script applies this
resource_scores.py

- Token-based inversion for positive fields like restored, replenish, efficiency (if such columns appear).
- Computes Res1_composite, Res2_composite, Res3_composite, then Resource_Composite.

- Saves CSV + charts.

Run

bash

CopyEdit

- `python resource_scores.py`
- `# or, z-score:`
- `python -c "import resource_scores as s; s.compute_resource(norm_method='zscore')"`

-