EMISSIONS SCORE

1) `total_emissions_intensity.csv`

Columns

- `Entity` – country name
- `Code` – ISO3 code (e.g., USA, IND)
- `Year` – data year
- `Coal_CO2_Emissions` – coal-related CO_2 emissions (tCO_2)
- `Coal_Production_TWh` – coal production (TWh)
- `Emissions_Intensity_tCO2_per_TWh` – **score metric** = emissions per unit output

How it's calculated

Emissions Intensity=Coal_CO2_Emissions (tCO_2)Coal_Production_TWh\text{Emissions Intensity}=\frac{\text{Coal_CO2_Emissions (tCO_2)}}{\text{Coal_Production_TWh}}Emissions Intensity=Coal_Production_TWhCoal_CO2_Emissions (tCO_2)

Lower is better (cleaner production per energy produced).

---

2) `absolute_global_emissions_share.csv`

Columns

- `Entity`, `Code`, `Year`
- `Coal_CO2_Emissions` – coal CO_2 (tCO_2)
- `Global_Share` – **score metric** = country share of world coal CO_2

How it's calculated

$$\text{Global Share} = \frac{\text{Country coal } CO_2}{\sum_{\text{all countries}} \text{coal } CO_2}$$

Lower is better (a smaller share of the global coal problem).

---

3) `policy_exempt_emissions.csv`

Columns

- `Entity`, `Code`, `Year`
- `Share_Covered_Carbon_Price` – fraction of national CO_2 covered by a carbon price (tax or ETS), 0–1
- `Policy_Exempt_Emissions` – **score metric** = uncovered fraction

How it's calculated

$$\text{Policy-Exempt} = 1 - \text{Share\_Covered\_Carbon\_Price}$$

Lower is better (fewer emissions left outside carbon pricing).

---

4) `lifecycle_emissions_coverage.csv`

Columns

- `Entity`, `Code`, `Year`
- `Lifecycle_Emissions_Coverage` – **score metric** = fraction (0–1) of CO_2 covered by carbon pricing

How it's calculated

Directly from coverage data (same raw source as #3).
Higher is better (more of the lifecycle emissions are priced/covered).

---

5) `historical_emissions_debt.csv`

Columns

- `Entity`, `Code`
- `Cumulative_Emissions_1850_2023` – **score metric**; cumulative coal CO_2 since 1850 (tCO_2)
- `First_Year` – first year with coal-CO_2 data

How it's calculated
Sum coal CO_2 from 1850 (or the first available year) to 2023:

$$\text{Historical Debt} = \sum_{y=1850}^{2023} \text{coal } CO_2(y)$$

Lower is better (smaller historical contribution).

---

6) `carbon_abatement_readiness.csv`

Columns

- `Entity`, `Code`, `Year`
- `Share_CarbonTax` – fraction (0–1) covered specifically by **carbon tax**
- `Carbon_Abatement_Readiness` – **score metric** (same value)

How it's calculated
Direct fraction under a **carbon tax** (subset of coverage).
Higher is better (a more direct abatement signal).

Normalization and a Composite Score

To combine apples-to-apples, we normalize each small score to **[0, 1]** using **min–max** scaling, aligning direction so **1 = better**:

- **Lower-is-better (invert)**: `intensity`, `global_share`, `policy_exempt`, `historical_debt`
- **Higher-is-better (no invert)**: `lifecycle_coverage`, `abatement_readiness`

Then build a weighted composite:

$$\text{Composite}=\sum_i w_i \cdot \text{NormalizedScore}_i$$

Default weights in the code are **equal (1/6 each)**; pass your own weights if you prefer (they will be renormalized to sum to 1).

Ready-to-run Python module

I prepared a module that:

- loads your six CSVs,
- recomputes (and renames) the six small scores into one table,
- min-max normalizes them (with correct direction),
- computes a composite score,
- saves outputs (raw, normalized, composite), and
- renders a small “dashboard” (4 matplotlib charts) to PNGs.

Download the module:
coal_scores_dashboard.py

How to use it (example)

bash

CopyEdit

```
# In a terminal with Python 3.10+ and matplotlib installed
python coal_scores_dashboard.py \
```

- `  --inputs /path/to/folder/with/six_csvs \`
- `  --out /path/to/output_folder \`
- `  --weights '{"intensity_n":0.35,"global_share_n":0.10,"policy_exempt_n":0.05,"lifecycle_coverage_n":0.30,"historical_debt_n":0.10,"abatement_readiness_n":0.10}'`

Outputs:

- `coal_small_scores_raw.csv` – the six raw metrics per country
- `coal_small_scores_normalized.csv` – normalized metrics (1 = better)
- `coal_composite_scores.csv` – includes the composite score
- `charts/` – `top_composite.png`, `top_lifecycle_coverage.png`, `top_lowest_policy_exempt.png`, `scatter_intensity_vs_coverage.png`

> If any country lacks a needed input (e.g., no coal production), the specific derived value is left blank (NaN) and excluded from plots.

What the module contains (at a glance)

- `compute_small_scores()` – merges the six CSVs into one frame with columns:
  - `intensity`, `global_share`, `policy_exempt`, `lifecycle_coverage`, `historical_debt`, `abatement_readiness`
- `normalize_scores()` – produces `*_n` versions in [0,1] with correct directionality.
- `composite_score()` – weighted sum of the normalized columns.
- `build_dashboard()` – saves 4 matplotlib charts:
  - Top composite countries

    - Top normalized lifecycle coverage
    - Top normalized **low** policy-exempt (already oriented, so “top” = least exempt)
    - Scatter: normalized intensity vs. normalized coverage
- `run_pipeline()` – one-call end-to-end.

Notes and good practice

- **Directionality matters**: ensure lower-is-better metrics are inverted *after* normalization (the module does this for you).
- **Weights**: If you choose the weights from your slide (or any scheme), pass them via `--weights`. The module renormalizes them to sum to 1.
- **Robustness**: Min–max can be sensitive to outliers; for classwork you might also try z-scores and then rescale to [0,1] with a logistic transform, but min–max is most interpretable.
-