

Life-Cycle Assessment (LCA) Tool – Project Report

1 Overview

The **LCA-tool** is a lightweight, modular Python package that helps engineers and sustainability practitioners quantify and visualise the environmental impacts of manufactured products across their entire life-cycle.

It supports:

- **Flexible data ingestion** – CSV, Excel, or JSON for product data; JSON for impact-factor libraries
 - **Impact calculations** – Carbon, energy, water, and waste, normalised on a mass basis and rolled-up by stage or product
 - **Rich visual analytics** – Pie, stacked-bar, radar, and correlation heat-maps, rendered with Matplotlib/Seaborn (GUI-less for CI)
 - **Turn-key reporting** – Auto-export of results to Excel/CSV and generation of a multi-page PDF report
 - **Robust unit-tests** – Pytest suite guarding I/O, validation, and plot creation (headless)
-

2 Architecture & Module Walkthrough

```
lca-tool/
├── data/raw      # raw sample_data.csv, impact_factors.json
├── results/      # generated XLSX, PNG, PDF (git-ignored)
├── src/
│   ├── data_input.py # ⇒ DataInput
│   ├── calculations.py # ⇒ LCACalculator
│   └── visualization.py # ⇒ LCAVisualizer
├── tests/
│   ├── test_data_input.py
│   └── test_visualization.py
├── notebooks/
│   └── lca_analysis.ipynb # demo notebook / exploratory dev
```

2.1 DataInput

- **Responsibilities**
 - Read product datasets (read_data)
 - Validate structure + numeric integrity (validate_data)
 - Load impact-factor libraries (read_impact_factors)
- **Key validation rules**
 - Presence of 14 required columns
 - Numeric coercion of mass/energy/rate columns
 - Sum of recycling_rate + landfill_rate + incineration_rate ≈ 1 ($|\text{error}| < 1 \times 10^{-3}$)

2.2 LCACalculator

- **Responsibilities**
 - Merge product rows with stage/material impact factors
 - Compute **absolute impacts** & **normalised impacts**
 - Produce comparison tables across product IDs
- **Algorithm**
 - Loop rows \rightarrow look-up (material_type, life_cycle_stage) factors
 - Impact = $\text{quantity} \times \text{factor}$ + direct measurement (e.g. carbon_footprint_kg_co2e)

2.3 LCAVisualizer

Headless plotting (uses matplotlib with Agg backend inside tests):

Method	Figure	Purpose
plot_impact_breakdown	Pie	Share of impact by material (or any grouping)
plot_life_cycle_impacts	$4 \times$ bar	Stage-wise impacts for a product
plot_product_comparison	Radar	Normalised multi-impact comparison
plot_end_of_life_breakdown	Stacked bar	Recycling vs landfill vs incineration
plot_impact_correlation	Heatmap	Correlation between impact categories

2.4 Directory & File Roles

Path	Kind	What it contains / does
data/raw/	Input data	sample_data.csv (row-level product inventory) and impact_factors.json (per-kg emission factors). Read-only – treat as golden source.
results/	Build artefacts	Populated at runtime with Excel, CSV summaries, PNG plots, and the PDF report. Added to .gitignore to keep the repo clean.
src/data_input.py	Core module	DataInput – parses/validates raw tables and factor libraries.

Path	Kind	What it contains / does
src/calculations.py	Core module	LCACalculator – merges raw data with factors and computes impacts, normalisation, comparisons.
src/visualization.py	Core module	LCAVisualizer – generates headless Matplotlib/Seaborn figures.
tests/	Unit-tests	test_data_input.py + test_visualization.py – CI guards for I/O + plotting.
notebooks/lca_analysis.ipynb	Exploratory notebook	End-to-end demo: ingest → calc → plot → export. Also served as a scratch-pad for bug-fixes before promoting changes into src/.

“Path travelled” during debugging

1. `** → **` – fixed Seaborn style call.
2. `** → **` – relaxed EoL-rate check, added numeric coercion & row-wise logging.
3. **Notebook experimentation** – validated fixes live, then copied into src/.
4. `**Re-ran **` – confirmed 5/5 plot tests and 4/4 data tests green.

3 Test-Driven Development & Debug-Diary

Milestone 1 – Visualisation tests

- **Symptom:** All five tests/test_visualization.py failed with `OSError: 'seaborn' is not a valid package style`.
- **Root cause:** visualization.py called `plt.style.use('seaborn')`; the minimal CI image lacks external seaborn style files.
- **Fix:** Replaced the call with `sns.set_theme()` & forced `matplotlib.use('Agg')` in test-file. Visualisation tests now **pass**.

Milestone 2 – Data-validation test

- **Symptom:** test_validate_data failed; row rates summed to 0 (transport rows had zeros).
- **Design decision:** In real data, transportation rows often have 0 for EoL rates. Updated validate_data to *ignore* rows where all three rates are zero.
- **Additional enhancements:**
 - Added per-row error logging for easier debugging.
 - Cleaned numeric-type coercion via `pd.to_numeric(..., errors="coerce")`.

Milestone 3 – House-keeping

- **Warning:** Pandas FutureWarning on dtype mismatch during test; acceptable for now but cast explicitly during validation.
- **Tests:** All nine unit tests now pass (pytest -q ⇒ 9 passed).

4 Manual QA Checklist

Check	Status
pytest green	✓
Notebook lca_analysis.ipynb executes top-to-bottom	✓
results/ auto-populates .xlsx, .csv, .png, .pdf	✓
requirements.txt minimal & frozen	✓ (pandas, matplotlib, seaborn, openpyxl, pytest)
README.md updated with usage examples	✓
src/ is importable as package (pip install -e .)	✓
.flake8 / black formatting	<i>optional</i>

5 Reflection

“Data fights intuition.” Developing this tool exposed several subtle assumptions (e.g., EoL rates for non-EoL rows) that would have gone unnoticed without rigorous testing. TDD proved invaluable, turning each bug into a documented learning opportunity and leaving us with a maintainable code-base that others can extend confidently.

Author

Zübeyr Aksu, 2025-06-13