DD-Coherence - running the DD operator on CSV

0) Identity / scope

- **Recommended OSF title**: *DD-Coherence (Loki) — running the DD operator on CSV time-series*
- **Source repository (URL)**: `github.com/dalozedidier-dot/DD-Coherence` (alias); the README redirects to `dalozedidier-dot/Loki`.
- **Resource type**: *Software*
- **Observed status**: public repository, 5 commits, 100% Python.
- **Version to freeze (Registration snapshot)**:
    - Branch: `main`
    - Tag: `[ …]`
    - Commit hash: `[ …]`
    - Freeze date: `[ …]`

1) Definition (Description / Abstract)

DD-Coherence (Loki) is a tool for executing the **DD operator** on tabular data (CSV). It loads one or more series, applies DD processing using an externalized parameter set (JSON), then produces auditable artifacts: a structured report and derived series. The system is designed to run locally and/or via GitHub Actions (smoke test and batch runs).

2) Materials / components to archive (Materials)

Include in OSF (ideally as a copy/archive, not only as a link):

- **Directories**
    - `dd_coherence_tool/` (engine + scripts + tests + example CSVs)
    - `.github/workflows/` (CI orchestration)
- **Parameter files**: `dd_params.example.json`, `dd_params.small.json`

- **Documentation**: `README.md`, `HOWTO_GIT_FILES.md`

- **Frozen archive (recommended)**: ZIP of the repository at the recorded commit/tag (OSF “Files”).

3) Data (Data / Inputs)

- **Format**: CSV.

- **Structural requirements** (must be explicitly declared in OSF):

    - Target columns: e.g., `'Failure rate'`, `'Avg job run time'` (as in the example command).

    - Minimum length: a “1-row” CSV is marked **skipped** (declared rule).

    - Ordering: DD requires an order (time or sequence); if timestamps are absent, the chosen ordering must be declared as a protocol assumption.

4) Operational procedure (Methods)

Local installation

```
pip install -r dd_coherence_tool/requirements.txt
```

Single run (example)

```
python dd_coherence_tool/scripts/run_dd.py \
  --input dd_coherence_tool/1.csv \
  --outdir out_dd \
  --cols "'Failure rate" "'Avg job run time"
```

Batch run (example)

```
python dd_coherence_tool/scripts/run_dd_batch.py \
  --outdir out_dd \
  --config dd_params.small.json \
  dd_coherence_tool/1.csv dd_coherence_tool/2.csv
dd_coherence_tool/3.csv
```

CI orchestration (GitHub Actions)

- `dd_smoke.yml`: smoke test + possible generation of `dd_components.csv.gz`
- `dd_on_committed_csvs.yml`: DD execution on committed CSVs (`1.csv`, `2.csv`, `3.csv`)

5) Outputs / analytical products (Outputs)

Each run writes (declared outputs):

- `dd_report.json`
- `dd_series.csv`
- `dd_components.csv.gz` (if enabled and if the series is long enough)

6) Validation / verifiability criteria (QA / Validation)

Minimal “success” criteria:

- execution completes without error (exit code 0)
- expected output files are present (list above)
- structural conformance (to be defined): minimal JSON schema for `dd_report.json` + expected CSV format

“Skipped” criterion:

- series too short ⇒ run marked **skipped** (to be tested and documented)

7) License / reuse (License)

- **OSF License field**: must be explicitly filled in.
- If no license is present/displayed in the repository: state “not specified” (as a finding) and, if desired, add a LICENSE file in the frozen version.

8) Limitations (Limitations)

- Without tag/hash + frozen archive, the “software” object is not fixed.

- Without an explicit input/output schema, conformity auditing is reduced to checking file presence.

- The system does not provide causal explanations: outputs are computational observations (measures/breaks/flags) parameter-dependent.