CubeDynamics: An Inspectable Grammar for Environmental Data Analysis

Ty Tuff
Environmental Data Science Innovation & Impact Lab
Cooperative Institute for Research in Environmental Sciences
University of Colorado Boulder, Boulder, Colorado, USA
ORCID 0000-0001-5249-5197

August 28, 2026

Abstract

Environmental questions that sound simple often compress consequential choices about observations, statistics, transformations, and assumptions into short scripts. Making a script rerunnable supports computational repeatability, but does not by itself reveal what scientific question the code asked. We present the reviewed CubeDynamics 0.1.0 candidate, an open-source Python package that places an inspectable environmental grammar around ordinary scientific Python objects. Source-qualified nouns retain product identity; semantic verbs and explicit modifiers describe transformations; authored order supplies syntax; and state and trace record how the scientific object changes. Numerical work remains with xarray, Dask, and established geospatial libraries. Across reviewed examples, the candidate preserves source and transformation identity, distinguishes meaning-changing and invalid sequences, demonstrates two-noun composition in a South Dakota vignette, and exposes the evidence supporting source interpretation. Its objective is to compress implementation complexity without compressing away scientific meaning.

1 Introduction

The question *Where was it hot and dry?* sounds modest. An analyst could load temperature and precipitation, define two conditions, combine them, and make a map. Yet each step opens another question: Which temperature and precipitation products? Which dates, grids, and temporal resolutions? Is "hot" an absolute threshold or a value relative to a climatology? What counts as "dry"? How are missing observations handled, and what happens when sources do not align? A short question can require substantial computational machinery.

There is a deeper problem. Even perfectly rerunnable code may not make the scientific question inspectable. Consider the same 2×2 temperature field, the same threshold, and the same spatial mean in Figure 1. Thresholding first asks what fraction of cells satisfy the condition. Averaging first asks whether the regional mean satisfies it. Both analyses are computationally reproducible, but they ask different questions of the observations.

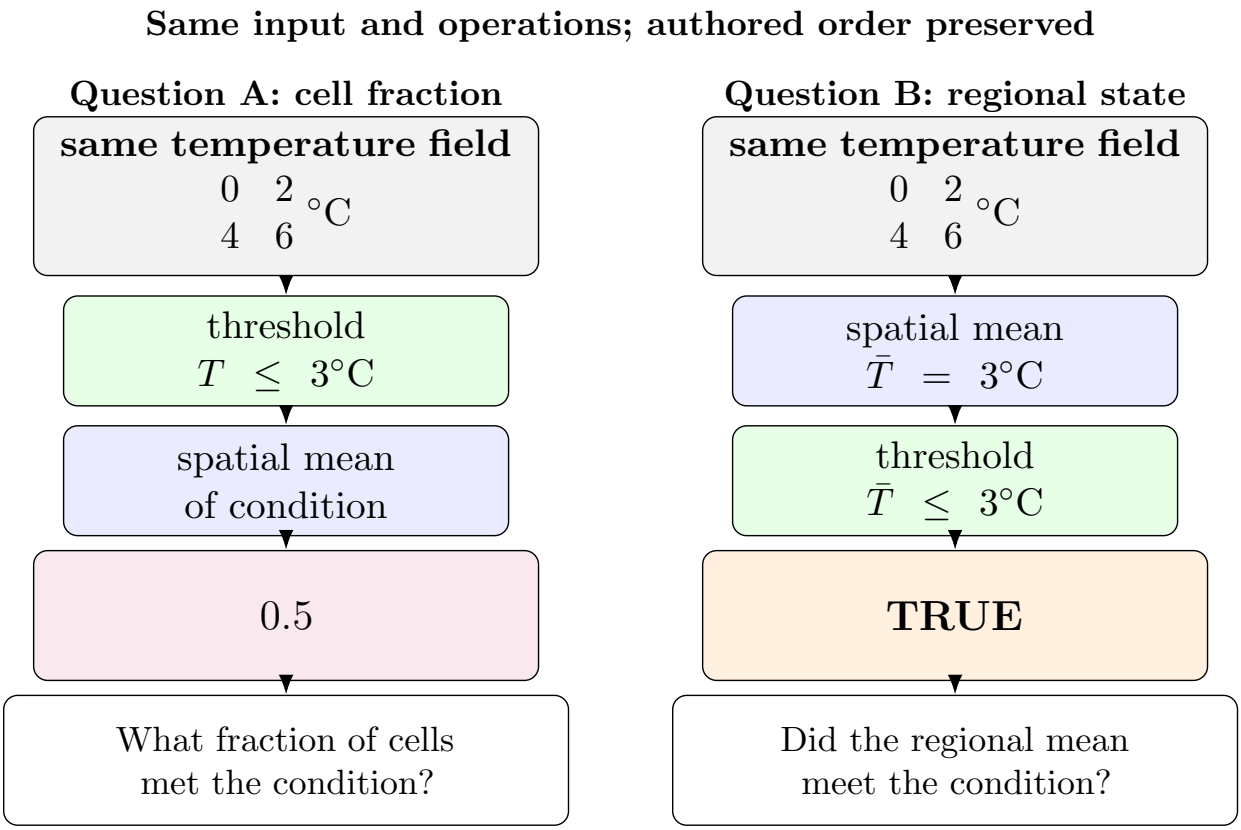

Figure 1: Operation order changes scientific meaning. For the same field $\{0, 2, 4, 6\}\,^{\circ}\mathrm{C}$ and inclusive threshold $T \leq 3\,^{\circ}\mathrm{C}$, threshold-then-mean produces a cell fraction of 0.5, whereas mean-then-threshold produces one regional Boolean state, `TRUE`. Both sequences are reproducible; they answer different questions.

The central concern is therefore not only whether an analysis can be rerun, but what it asked of the observations. Scientific inspectability requires a recoverable account of the inputs, the objects produced by each transformation, the consequences of authored order, and the evidence supporting the path from source to result.

CubeDynamics addresses this problem by treating an environmental pipeline as an executable scientific statement. Around ordinary scientific Python objects, it supplies a source-aware vocabulary, semantic contracts, evolving state, and an ordered trace; it does not replace numerical libraries or scientific judgment. The primary users are environmental scientists and data scientists building reusable analyses from heterogeneous data, together with the research software engineers who support them.

2 From readable computation to scientific grammar

Sequential reasoning has long provided a way to make computation legible. Fry's computational information design described a directional process—acquire, parse, filter, mine, represent, refine, and interact—for moving from data toward interpretation [1]. The value of that precedent is not a

mandatory sequence, but the recognition that a complex computational argument can be expressed as meaningful stages.

The tidyverse later made ordered transformation especially familiar in data analysis: data flows through verbs for filtering, transforming, grouping, and summarizing [2, 3]. Piping and authored order are not CubeDynamics innovations, nor is CubeDynamics a replacement for the tidyverse. It adopts the underlying insight that writing operations in analytical order can make computational reasoning easier to read.

GIS has a related visual precedent in ArcGIS ModelBuilder, which represents geoprocessing workflows as diagrams connecting data and tools, with one process's output becoming another's input [4]. ModelBuilder makes spatial workflow sequence and composition explicit in a visual programming environment. CubeDynamics addresses a different layer: it adds source-aware environmental vocabulary, semantic state, and trace to ordinary Python objects rather than providing a visual geoprocessing system.

Environmental analysis already has mature computational infrastructure. NumPy, xarray, and Dask provide array computation, labels, and deferred execution; rasterio/rioxarray, GeoPandas, and Shapely provide geospatial operations; STAC clients provide asset discovery; and data-cube systems organize Earth observations [5–15]. Domain packages add climate indicators, subsetting and regridding, catalog-driven loading, and model-evaluation recipes [16–19]. CubeDynamics delegates to these systems.

Piping alone, however, says little about the scientific meaning of what moves through the sequence. Temperature is not merely an array, and thresholding can do more than change numerical values: it can turn an observation into a condition. The term *grammar* follows precedents in graphics, tidy data, workflow composition, and scientific data cubes, but here denotes a narrower combination of environmental vocabulary, meaningful composition, and runtime semantic inspection [20–23]. The code is not natural language; the aim is to make an environmental calculation readable as a scientific statement while keeping it executable.

The reviewed candidate does not select scientifically appropriate sources, remove observational bias, make products interchangeable, schedule a workflow, certify provider observations, or establish fitness for a decision. Its contribution lies between data access and numerical execution: keeping environmental meaning inspectable while established libraries perform the computation.

3 An inspectable environmental grammar

The grammar combines a readable expression with an inspectable foundation. At the expression level, a source-qualified noun enters a pipe and is transformed by modified verbs in authored order. At the inspection level, evolving state describes the scientific object, trace preserves the sequence, and source records and evidence ground the abstraction. These levels connect the compact analysis to the information needed to interpret it.

3.1 Source-qualified nouns

A noun names the environmental information being analyzed; a required source flavor identifies the provider and product that give it observational meaning. Calls to `data.temperature` with the `prism` or `gridmet` flavor participate in one vocabulary without erasing native variables, units, grids, queries, revisions, or provenance. The source flavor qualifies the noun rather than acting as an interchangeable implementation detail. Nouns return familiar xarray objects, not proprietary containers [24].

3.2 Semantic verbs and scientific modifiers

A verb is a configured callable with a semantic contract. Thresholding changes an observation into a condition; event detection requires an ordered temporal condition with remaining variation; and reductions remove or retain declared dimensions. Verb parameters can be read as adverbial scientific modifiers because they specify how an operation is performed—for example, over which dimension, in which direction, or at which threshold or quantile. They remain ordinary parameters in the reviewed candidate, not a separate interface for adverbs. Moreover, not every argument is a modifier: the second observation supplied to a combining operation such as `overlap` is another scientific operand [24].

Given a temperature object returned by a noun, the core grammar remains small:

```
from cubedynamics import pipe, verbs as v

analysis = (
    pipe(temperature)
    | v.anomaly(dim="time")
    | v.variance(dim="time", keep_dim=False)
)

print(analysis.explain())
result = analysis.unwrap()
```

The noun enters the pipe, the two verbs carry explicit dimensional parameters, and the result remains inspectable before returning to ordinary Python use. Section 4 develops the complete multi-noun example.

3.3 Authored syntax, semantic state, and trace

The order in which verbs and modifiers are written forms the syntax of the analytical statement. CubeDynamics does not rearrange that expression. After each completed stage, the current semantic state describes the scientific object, including its dimensions, units, CRS, spatial and temporal status, remaining variation, source flavor, and provenance. It is a runtime semantic record that constrains compatible next operations and records what information has changed or disappeared.

The trace appends each completed stage in authored order, producing a recoverable account of the

operations, parameters, and state transitions expressed inside the pipe. Inspection methods expose explanations, compatible next steps, validation messages, state, and trace without changing the value or computing lazy arrays solely to infer semantics [24]. The trace is not complete workflow provenance: transformations before `pipe` or after `unwrap` remain outside it.

Some operations commute only under explicit assumptions. With a fixed climatology C, identical linear spatial weights, and identical masks,

$$\text{mean}_s\,(T(s,t) - C(s,t)) = \text{mean}_s\,T(s,t) - \text{mean}_s\,C(s,t). \tag{1}$$

The equality can fail when climatology, masks, interpolation, or weights differ between stages. Order rules therefore explain required, meaning-changing, information-removing, or conditionally equivalent sequences; they do not rewrite them. Authority remains with the researcher [24].

3.4 Statement boundaries, interoperability, and extension

The adoption path is deliberately reversible: xarray object → `pipe` → semantic operations and inspection → `unwrap` → xarray object. Within the linguistic analogy, `unwrap` is a local boundary marker—loosely, a period—between a CubeDynamics-inspected segment and ordinary Python. It does not force computation, certify the result, complete the wider workflow, or prevent the value from entering another pipe [24].

The pipe symbol provides sequencing syntax, while operations such as `overlap` are explicit combining verbs; the reviewed candidate does not define a complete grammar of analytical branching. NumPy- and Dask-backed values retain their established execution behavior when an operation permits it [6, 7]; deferred computation, however, does not prove bounded network access, so transport behavior is evaluated separately. Project-specific callables require no subclass or registration. Promotion into the maintained vocabulary remains a separate review because public names and contracts carry scientific meaning [24].

4 From analytical statements to environmental questions

A piped statement contributes to an analysis; the analysis addresses a question; and its result may eventually become evidence considered in a decision. CubeDynamics operates at the first two levels: it makes the analytical statement inspectable without assuming decision authority.

The South Dakota Working Lands vignette gives the opening question a bounded definition: where July 2024 maximum temperature exceeded each cell's 0.75 quantile for that month while precipitation was at or below 0.1 mm. It expresses the two conditions as separate branches:

```
from cubedynamics import data, pipe, verbs as v

query = dict(
    source="prism", bbox=[-101.2, 43.7, -100.4, 44.3],
    start="2024-07-01", end="2024-07-31",
)
temperature = data.temperature(**query, statistic="maximum")
precipitation = data.precipitation(**query)
```

```
warm = (pipe(temperature) | v.quantile_state(
    quantile=0.75, direction="above", name="warm_july_day"
)).unwrap()
dry = (pipe(precipitation) | v.threshold_state(
    threshold=0.1, direction="below", name="trace_or_no_rain"
)).unwrap()

analysis = (
    pipe(warm)
    | v.overlap(dry, name="warm_and_dry")
    | v.mean(dim="time", keep_dim=False)
)
report = analysis.validate()
frequency = analysis.unwrap() * 100
```

Temperature and precipitation enter separate inspected branches and become explicitly defined warm and dry conditions. Each branch has its own trace before `unwrap`; the final pipe records the subsequent `overlap` and temporal `mean`. The quantile, threshold, direction, and dimension parameters keep the definitions visible without implying that one final trace contains the complete histories of both branches.

The reviewed fixture contains 31 days on a 15×19 central South Dakota grid. Here, "warm" denotes days above each cell's 0.75 quantile calculated from that same July period; it is not an anomaly relative to a longer climatology. "Dry" denotes precipitation at or below 0.1 mm. Their overlap produces co-occurrence frequencies from 6.5% to 22.6% [24]. The result is deliberately narrow: the frequency with which those two conditions occurred together. It does not diagnose drought, land sensitivity, vegetation stress, harm, causation, economic impact, or risk.

5 Abstraction must remain grounded

Suppose the grammar succeeds. Retrieval, variable selection, coordinate handling, subsetting, metadata interpretation, provenance, and validation can recede behind a short call to `temperature` with the `prism` source flavor. That concision is powerful, but the hidden complexity has not disappeared; it has moved into an abstraction. Without a recoverable path downward, readable code could make the science less inspectable rather than more.

This reversal motivates the design maxim:

> *The shorter the analytical expression becomes, the stronger the evidence underneath it must become.*

The grammar therefore connects two directions. Above it, analytical statements contribute to scientific questions. Below it, nouns retain source identity, retrieval procedures, provenance, and bounded source QA, while verbs retain explicit parameters, semantic contracts, implementations, and behavioral tests. State and trace link the readable expression to this inspectable foundation.

5.1 A common noun does not imply source equivalence

A shared noun is a common entry point, not a claim of observational equivalence. The reviewed candidate implements this principle for PRISM and gridMET temperature, products constructed with different methods and grids [25, 26]. Table 1 lists distinctions retained by their adapters.

Table 1: One noun can preserve scientifically distinct source identities. Selected PRISM and gridMET differences remain visible rather than being silently harmonized [24, 27–29].

Property	PRISM flavor	gridMET flavor
Daily statistics	Maximum, minimum, and mean	Maximum and minimum; no native mean
Native variables and units	`tmax`, `tmin`, `tmean`; degrees Celsius	`tmmx`, `tmmn`; kelvin
Grid and documented span	Approximately 4 km; 1981–present	1/24 degree; 1979–present
Revision behavior	Recent daily grids may be remodeled	Recent observations progress through maturity stages
Reviewed QA scope	Bounded temperature extract	Bounded maximum-temperature extract

Comparability still depends on statistic, units, grid, temporal support, revision, and provenance. Sources with different measurands, such as near-surface air temperature and satellite land-surface temperature, require distinct nouns or mandatory qualifiers. Unit conversion and regridding can create computational compatibility without establishing scientific interchangeability. The common noun creates a common entry point; it does not make source substitution an implicit software decision.

5.2 Evidence beneath the abstraction

Concise source access rests on evidence for three distinct questions: Were bytes retrieved through the claimed access mode? Did the adapter interpret variables and coordinates as intended? Has the integration accumulated enough reviewed evidence for a bounded scope? Successful retrieval does not prove correct interpretation, and reviewed interpretation does not guarantee a currently available endpoint.

A serving revision identifies the CubeDynamics adapter and interpretation contract, not a provider's scientific version. Schema fingerprints make structural interpretation inspectable without hashing values or forcing lazy computation. Checksum-controlled fixtures support stable offline tests, while live checks are recorded separately. Table 2 summarizes the evidence without extending it into a certification of provider observations [24].

The trace complements these source records by recording completed operations, parameters, semantic states, dimensions, units, and messages. General environment capture and provenance outside the pipe remain separate responsibilities [24, 30].

Table 2: Concise source access rests on bounded evidence. Each evidence group supports a specific integration claim while stating what remains outside that claim.

Evidence group	What is checked	Boundary
Retrieval and live health	Small request, response identity, and access behavior	Availability can change independently of reviewed interpretation
Fixture and provenance	Reviewed extract, checksum, query bounds, and upstream identity	Establishes the tested bytes, not all places or periods
Structural interpretation	Variables, dimensions, coordinates, units, CRS, masks, and schema fingerprint	Tests the adapter contract without validating observations
Numerical and visual review	Physical ranges, source-specific relationships, and reviewed plots	Checks plausibility and adapter interpretation; does not establish observational accuracy or decision fitness

6 Evaluation

The evaluation is organized around the claims made for the grammar. Existing tests, vignettes, and QA records provide the evidence; they do not establish user productivity, usability, decision quality, or superiority to direct xarray analysis.

6.1 Source and transformation recoverability

The observational-data and executable-vignette checks show that noun outputs retain source flavor, provider and product information, native variables, query, units, grid metadata, serving interpretation, and provenance. Completed pipe stages retain operation names, modifier parameters, states, and authored order. The inspection interface exposes this record before `unwrap` returns the underlying object; operations outside the pipe are intentionally absent [24].

6.2 Semantic transitions, order, and invalid operations

Figure 1 demonstrates the analytical consequence of changing order; the grammar-equivalence and expected-failure checks establish the software behavior by preserving authored sequence and distinguishing observations from conditions and summaries. In particular, `detect_events` rejects a continuous field, and removal of temporal variation prevents a remaining length-one coordinate from being treated as event-ready time. Conditional-equivalence checks record the assumptions under which linear anomaly and mean operations commute. These results support semantic diagnostics, not completeness of the rule catalog [24].

6.3 Source distinction and evidence boundary

PRISM and gridMET participate in the same temperature noun while retaining the differences summarized in Table 1. Their reviewed fixtures do not overlap, so the evaluation demonstrates source distinction rather than product agreement. Source QA reports `PASS_WITH_CAVEATS` for bounded PRISM temperature, gridMET maximum-temperature, and Sentinel-2 reflectance/NDVI

integrations. The records preserve limitations, including gridMET's annual-file fallback and the Sentinel-2 fixture's lack of pixel-level cloud masking. On August 27, 2026, reviewed temperature integrations were `VALIDATED` while recorded live health was `STALE`, illustrating the intended evidence boundary [24].

6.4 Multi-noun composition

The Working Lands analysis in Section 4 tests whether two source-qualified observations can be transformed in separate inspected branches and then composed. Decision QA checks fixture provenance, dimensions, finite values, physical ranges, exact coordinate alignment, agreement between piped and direct calculations, and a bounded percentage result. These checks support the reported co-occurrence summary and the branch and final-pipe records; they do not extend it into a drought, impact, or risk assessment [24].

Across the reviewed snapshot, the five-module offline publication suite passes and twelve supported notebooks execute: eight core lessons, three real-data noun lessons, and the Working Lands analysis. These are release checks supporting the claim-level evidence above, not the contribution by themselves [24].

7 Discussion

CubeDynamics demonstrates that concise environmental code need not make scientific interpretation harder to recover. Its contribution is not the pipe syntax itself, but the connection between readable expressions and an inspectable account of their scientific objects, transformations, sources, and evidence. The linguistic framing is functional rather than decorative: vocabulary identifies objects and operations, syntax affects meaning, state constrains what can follow, and trace preserves the authored construction.

That connection becomes more important as workflows are reused, automated, and potentially assembled by software agents, because more interpretation moves into defaults, adapters, parameter choices, and operation order. Inspectable metadata cannot establish that those choices are scientifically appropriate, but it can make them available for review.

A larger motivation is the many-dataset synthesis problem: what would it take to construct a scientifically inspectable analysis from tens or, illustratively, 100 heterogeneous environmental datasets? This is a design challenge, not a benchmark or current capability claim. Such sources may differ in resolution, coordinates, units, observation processes, missing-data conventions, update schedules, provenance, uncertainty, and review status. The aspiration is not a five-line program, but a readable, potentially page-sized analytical specification in which source-qualified branches remain distinct or combine where scientifically justified. A scientist should be able to read that argument, a computer to execute it, and a reviewer to descend from any abstraction toward its parameters, state, trace, source, procedure, provenance, and evidence. The reviewed 0.1.0 candidate does not demonstrate this scale or a complete branch-and-join grammar; the two-noun Working Lands analysis is only a bounded first case.

The approach remains limited. Linear pipes do not naturally represent every branching or multi-cube

analysis; semantic states simplify richer scientific reasoning; source evidence ages; and complete reconstruction still requires environments, inputs, revisions, and provenance outside the pipe [31–33]. The present evaluation establishes implementation behavior and evidence boundaries. Comparative studies would be required to claim that the grammar produces faster analysis, fewer mistakes, improved usability, more readable large syntheses, or better decisions.

8 Conclusion

The reviewed CubeDynamics 0.1.0 candidate shows that an environmental pipeline can resemble an executable scientific statement while remaining ordinary scientific Python. It preserves a path from readable expression to authored transformations, source identity, and bounded supporting evidence. The aim is not to make difficult environmental science look simple, but to make its complexity readable without losing the ability to inspect and verify what lies underneath.

9 Availability and release status

Source code and documentation are available at `https://github.com/CU-ESIIL/cubedynamics` and `https://cu-esiil.github.io/cubedynamics/`. The reviewed repository snapshot is commit `862a80aed8a2781b40e6e5293fd6cfbcba887aa4`, which declares version 0.1.0 with alpha development status, Python 3.9 or later, and the MIT License. At the manuscript date, this candidate was neither a tagged release nor available through the public PyPI package path; an independent outside-user installation test therefore could not install it as a public package. Repository and reviewed-wheel installation remain development paths, not evidence of public distribution. The citation metadata contains no release DOI [24].

[*To do*: Before submission, tag and archive the reviewed state, independently verify public installation, insert the archive identifier, add the assigned DOI, and synchronize final citation metadata.]

Generative AI transparency statement

The author used OpenAI Codex for LaTeX diagnostics, repository and documentation checks, structural and stylistic editing, and figure design. The author supplied and reviewed the scientific and software content; Codex did not generate observational data or assign validation outcomes. The author verified retained claims and citations and accepts responsibility for the manuscript.

Author contributions

Ty Tuff: Conceptualization, Methodology, Software, Validation, Data curation, Visualization, Writing—original draft, Writing—review and editing, and Project administration.

Competing interests

The author declares no competing interests.

Acknowledgements

The author acknowledges the maintainers of the scientific Python libraries on which CubeDynamics depends and the organizations that provide the environmental data products used in its validation examples.

References

[1] Ben Fry. *Visualizing Data: Exploring and Explaining Data with the Processing Environment*. O'Reilly Media, 2007. ISBN 978-0-596-51455-6. URL https://www.oreilly.com/library/view/visualizing-data/9780596514556/.

[2] Hadley Wickham. Tidy data. *Journal of Statistical Software*, 59(10):1–23, 2014. doi: 10.18637/jss.v059.i10. URL https://doi.org/10.18637/jss.v059.i10.

[3] Hadley Wickham, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, et al. Welcome to the Tidyverse. *Journal of Open Source Software*, 4(43): 1686, 2019. doi: 10.21105/joss.01686. URL https://doi.org/10.21105/joss.01686.

[4] Esri. What is ModelBuilder? ArcGIS Pro documentation, 2026. URL https://pro.arcgis.com/en/pro-app/latest/help/analysis/geoprocessing/modelbuilder/what-is-modelbuilder-.htm. Accessed August 28, 2026.

[5] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, et al. Array programming with NumPy. *Nature*, 585:357–362, 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

[6] Stephan Hoyer and Joe Hamman. xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*, 5(1):10, 2017. doi: 10.5334/jors.148. URL https://doi.org/10.5334/jors.148.

[7] Matthew Rocklin. Dask: Parallel computation with blocked algorithms and task scheduling. In *Proceedings of the 14th Python in Science Conference*, pages 126–132, 2015. doi: 10.25080/Majora-7b98e3ed-013. URL https://doi.org/10.25080/Majora-7b98e3ed-013.

[8] Sean Gillies and contributors. Rasterio: Geospatial raster I/O for Python programmers. Computer software, 2013–present. URL https://github.com/rasterio/rasterio.

[9] Alan D. Snow, R. Braun, RichardScottOZ, Martin Raspaud, David Brochart, Kirill Kouzoubov, et al. rioxarray, version 0.22.0. Computer software, 2026. URL https://doi.org/10.5281/zenodo.18892731.

[10] Kelsey Jordahl, Joris Van den Bossche, Martin Fleischmann, Jacob Wasserman, James McBride, Jeffrey Gerard, et al. geopandas/geopandas, version 0.8.1. Computer software, 2020. URL https://doi.org/10.5281/zenodo.3946761.

[11] Sean Gillies, Casper van der Wel, Joris Van den Bossche, et al. Shapely, version 2.1.1. Computer software, 2025. URL https://doi.org/10.5281/zenodo.5597138.

[12] SpatioTemporal Asset Catalog community. SpatioTemporal Asset Catalog specification, version 1.1.0. Software specification, 2024. URL https://github.com/radiantearth/stac-spec/releases/tag/v1.1.0.

[13] Gregory Giuliani, Bruno Chatenoux, Andrea De Bono, Denisa Rodila, Jean-Philippe Richard, Karin Allenbach, Hy Dao, and Pascal Peduzzi. Building an Earth Observations Data Cube: lessons learned from the Swiss Data Cube (SDC) on generating Analysis Ready Data (ARD). *Big Earth Data*, 1(1–2):100–117, 2017. doi: 10.1080/20964471.2017.1398903. URL https://doi.org/10.1080/20964471.2017.1398903.

[14] Meng Lu, Marius Appel, and Edzer Pebesma. Multidimensional arrays for analysing geoscientific data. *ISPRS International Journal of Geo-Information*, 7(8):313, 2018. doi: 10.3390/ijgi7080313. URL https://doi.org/10.3390/ijgi7080313.

[15] Marius Appel and Edzer Pebesma. On-demand processing of data cubes from satellite image collections with the gdalcubes library. *Data*, 4(3):92, 2019. doi: 10.3390/data4030092. URL https://doi.org/10.3390/data4030092.

[16] Pascal Bourgault, David Huard, Trevor James Smith, Travis Logan, Abel Aoun, Juliette Lavoie, et al. xclim: xarray-based climate data analytics. *Journal of Open Source Software*, 8(85):5415, 2023. doi: 10.21105/joss.05415. URL https://doi.org/10.21105/joss.05415.

[17] clisops contributors. clisops: Climate simulation operations. Computer software and documentation, 2026. URL https://clisops.readthedocs.io/.

[18] intake-esm contributors. intake-esm: Cataloging and loading earth system model data. Computer software, 2026. URL https://github.com/intake/intake-esm.

[19] Mattia Righi, Bouwe Andela, Veronika Eyring, Axel Lauer, Valeriu Predoi, Manuel Schlund, et al. Earth system model evaluation tool (ESMValTool) v2.0—technical overview. *Geoscientific Model Development*, 13:1179–1199, 2020. doi: 10.5194/gmd-13-1179-2020. URL https://doi.org/10.5194/gmd-13-1179-2020.

[20] Leland Wilkinson. *The Grammar of Graphics*. Springer, 2 edition, 2005. doi: 10.1007/0-387-28695-0. URL https://doi.org/10.1007/0-387-28695-0.

[21] Hadley Wickham. A layered grammar of graphics. *Journal of Computational and Graphical Statistics*, 19(1):3–28, 2010. doi: 10.1198/jcgs.2009.07098. URL https://doi.org/10.1198/jcgs.2009.07098.

[22] Ewa Deelman, Karan Vahi, Gideon Juve, Mats Rynge, Scott Callaghan, Philip J. Maechling, Rajiv Mayani, Weiwei Chen, Rafael Ferreira da Silva, Miron Livny, and Kent Wenger. Pegasus, a workflow management system for science automation. *Future Generation Computer Systems*, 46:17–35, 2015. doi: 10.1016/j.future.2014.10.008. URL https://doi.org/10.1016/j.future.2014.10.008.

[23] Peter Amstutz, Michael R. Crusoe, Nebojša Tijanić, Brad Chapman, John Chilton, Michael Heuer, Andrey Kartashov, Dan Leehr, Hervé Ménager, Maya Nedeljkovich, Matt Scales, Stian Soiland-Reyes, and Luka Stojanovic. Common workflow language, v1.0. Software specification, 2016. URL https://doi.org/10.6084/m9.figshare.3115156.v2.

[24] Ty Tuff. CubeDynamics: A composable grammar for spatiotemporal science, development snapshot for version 0.1.0. Computer software and accompanying documentation, 2026. URL https://cu-esiil.github.io/cubedynamics/. Unreleased repository snapshot at commit 862a80aed8a2781b40e6e5293fd6cfbcba887aa4; accessed August 28, 2026.

[25] Christopher Daly, Michael Halbleib, Joseph I. Smith, Wayne P. Gibson, Matthew K. Doggett, George H. Taylor, Jan Curtis, and Phillip P. Pasteris. Physiographically sensitive mapping of climatological temperature and precipitation across the conterminous United States. *International Journal of Climatology*, 28(15):2031–2064, 2008. doi: 10.1002/joc.1688. URL https://doi.org/10.1002/joc.1688.

[26] John T. Abatzoglou. Development of gridded surface meteorological data for ecological applications and modelling. *International Journal of Climatology*, 33(1):121–131, 2013. doi: 10.1002/joc.3413. URL https://doi.org/10.1002/joc.3413.

[27] PRISM Climate Group, Oregon State University. PRISM time series data and dataset documentation, 2026. URL https://prism.oregonstate.edu/recent/. Accessed August 28, 2026.

[28] Climatology Lab. gridmet: Gridded surface meteorological data, 2026. URL https://www.climatologylab.org/gridmet.html. Accessed August 28, 2026.

[29] Google Earth Engine Data Catalog. GRIDMET: University of idaho gridded surface meteorological dataset, 2026. URL https://developers.google.com/earth-engine/datasets/catalog/IDAHO_EPSCOR_GRIDMET. Accessed August 28, 2026.

[30] PROV-O: The PROV ontology. W3C Recommendation, 2013. URL https://www.w3.org/TR/prov-o/.

[31] Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. Ten simple rules for reproducible computational research. *PLoS Computational Biology*, 9(10):e1003285, 2013. doi: 10.1371/journal.pcbi.1003285. URL https://doi.org/10.1371/journal.pcbi.1003285.

[32] Greg Wilson, Jennifer Bryan, Karen Cranston, Justin Kitzes, Lex Nederbragt, and Tracy K. Teal. Good enough practices in scientific computing. *PLoS Computational Biology*, 13(6): e1005510, 2017. doi: 10.1371/journal.pcbi.1005510. URL https://doi.org/10.1371/journal.pcbi.1005510.

[33] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, et al. The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, 3:160018, 2016. doi: 10.1038/sdata.2016.18. URL https://doi.org/10.1038/sdata.2016.18.