SI2-SSI: Inquiry-Focused Volumetric Data Analysis Across Scientific Domains: Sustaining and Expanding the yt Community

Matthew Turk, Ben Holtzman, Leigh Orf

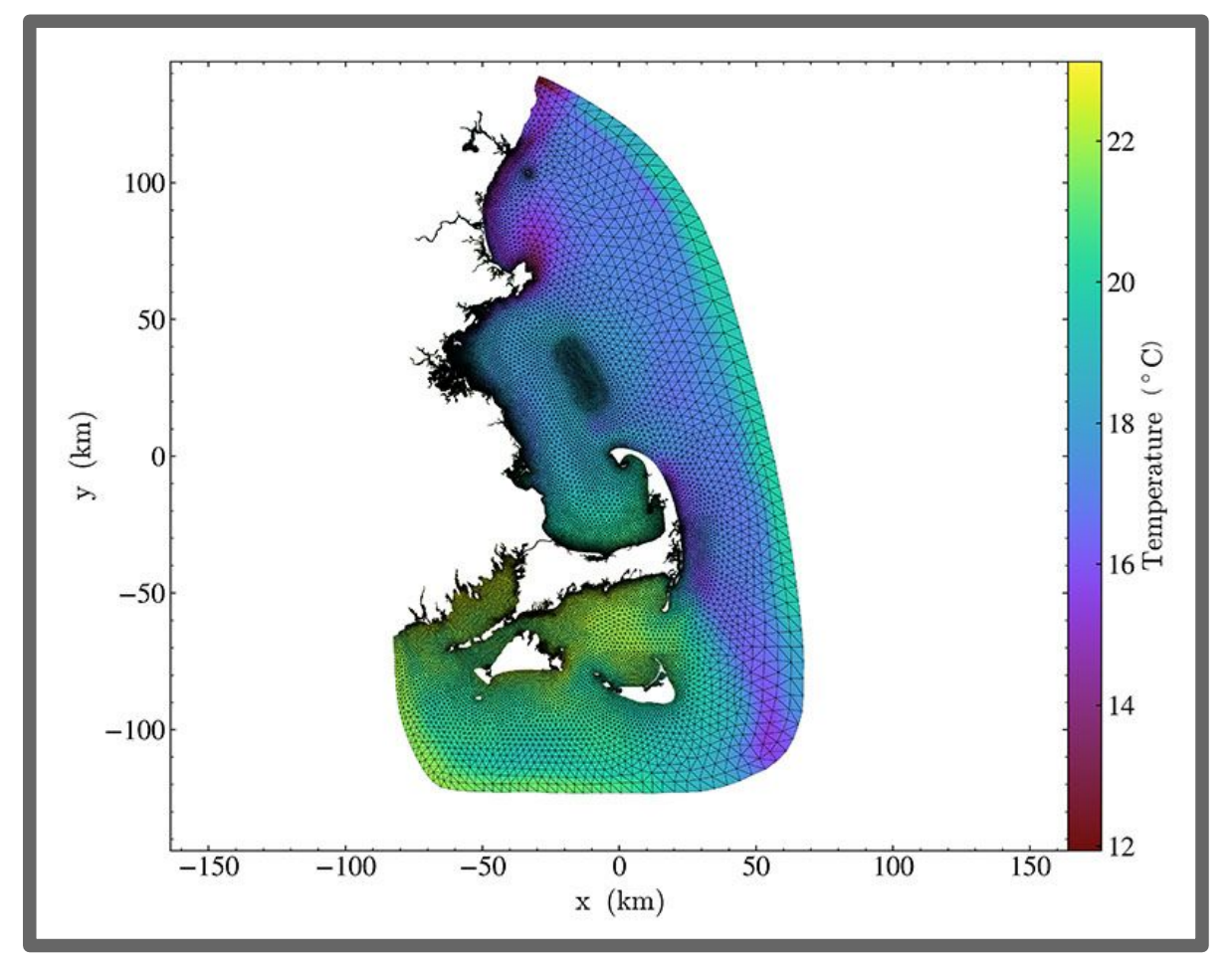
NCSA & iSchool at University of Illinois
`yt-project.org`
`github.com/yt-project`

What is yt?

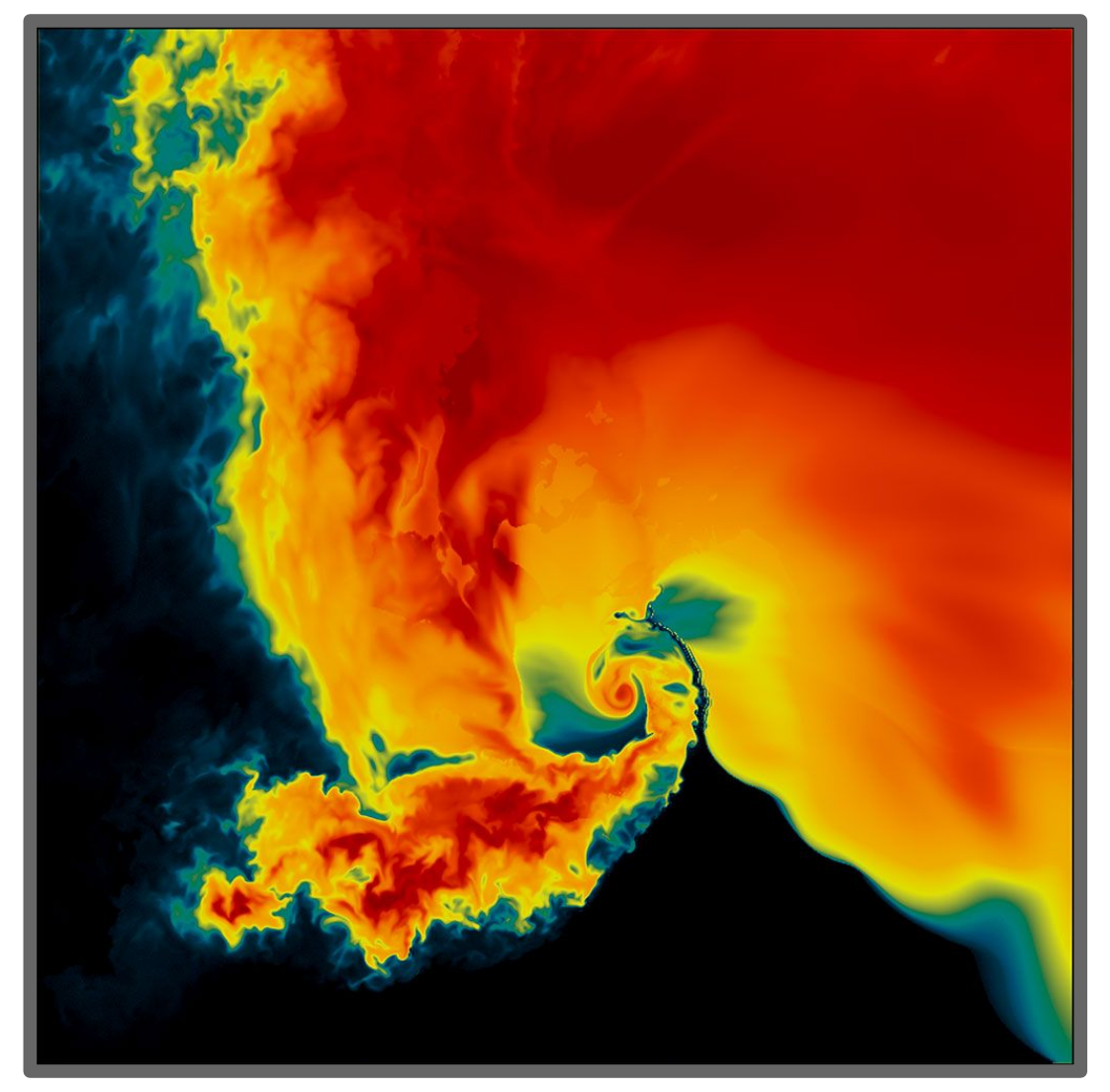
yt is designed to guide scientific inquiry (analysis, visualization, simulation) through physically-motivated understanding. It is released under the BSD license, developed completely in the open, and is designed to present a library of loosely-coupled components that can be easily integrated with other Python tools.

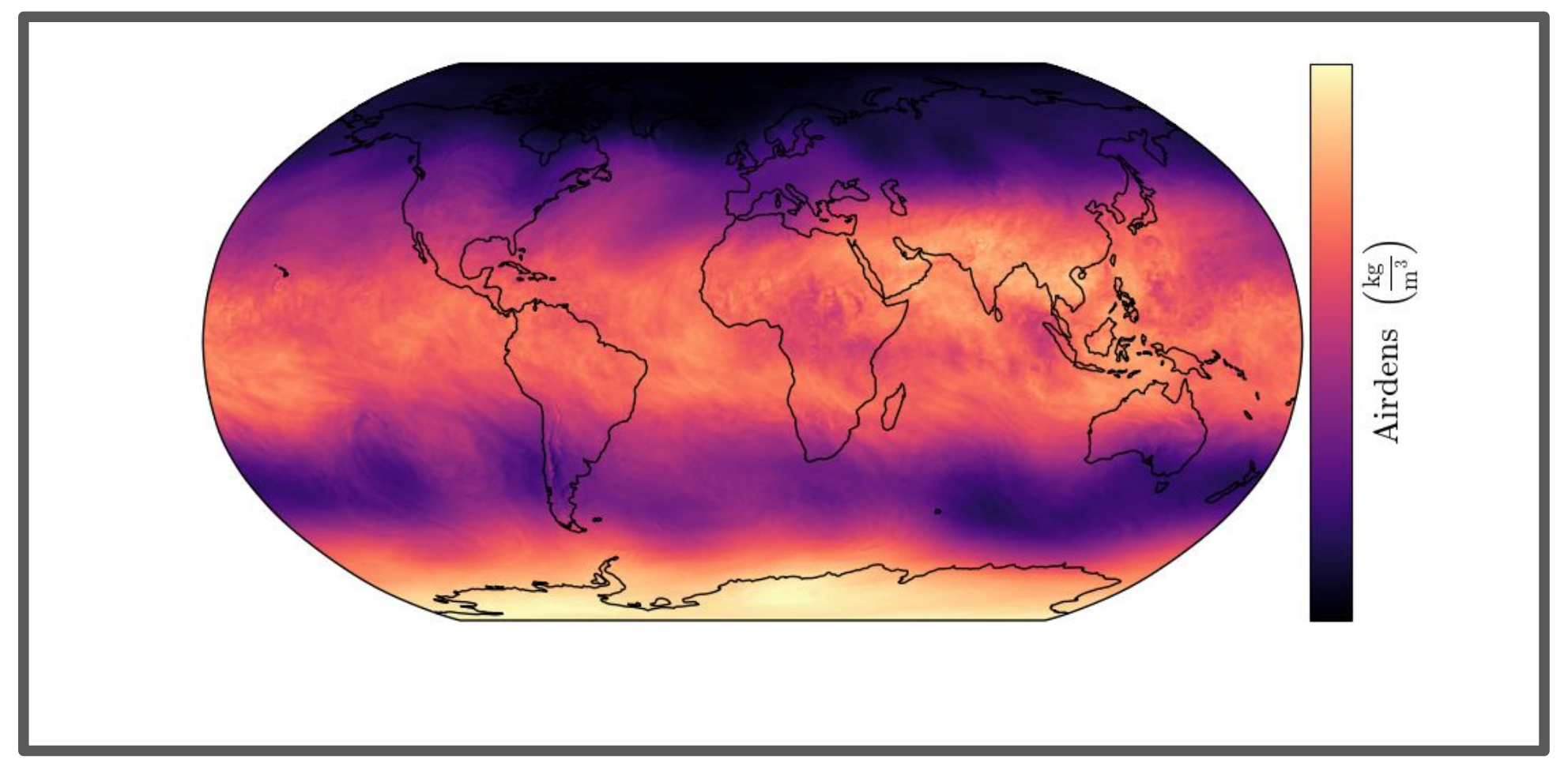
A reference implementation in yt will create a development standard for expansion into new physical domains as a strategy to grow the codebase and create an accessible and extensible framework. Our goal is to make the domain context system pluggable and easily extensible without requiring knowledge of yt internals.

FVCOM Ocean Forecast Model of the North Atlantic Coast (NOAA, UMass Dartmouth)



Predicted Weather Radar from Tornadogenesis Simulation Leigh Orf (University of Wisconsin)





Global Air Density Projection
Madicken Munk, University of Illinois
Data: GMAO at NASA, fluid.nccs.nasa.gov/weather/

Expanding Beyond Astrophysics

From a user standpoint, we can identify the pattern and libraries different domains use to read in, unpack, and pull out the values and fields they want to then visualize and analyze in yt.

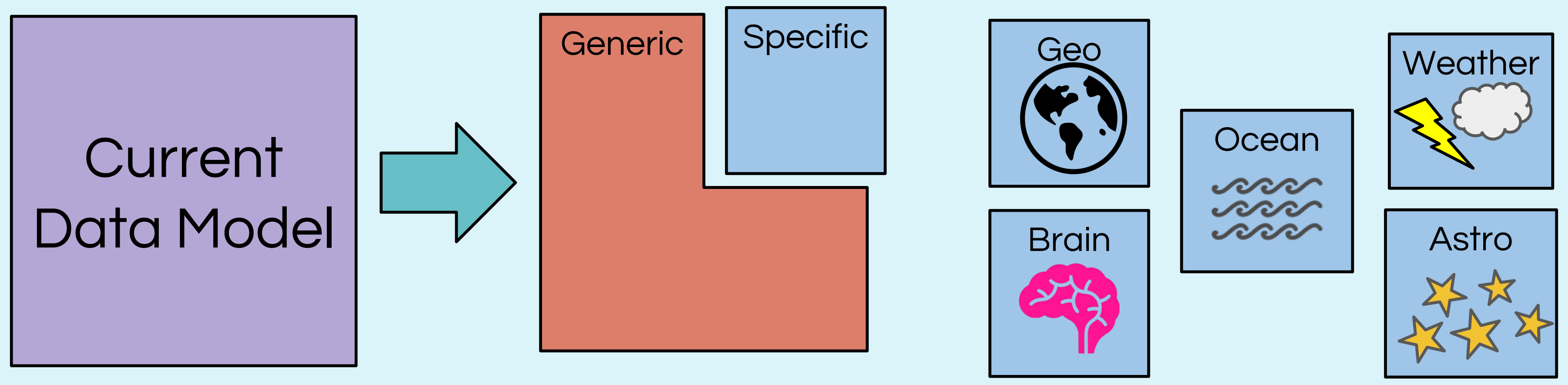
From a developer standpoint, we can identify overarching assumptions and design choices that can inform what needs to be abstracted out, or added to the code base to make yt more extendable and accessible to new domains.

Future versions will include only the core yt functionality, with all astronomy-specific analysis modules shipped in the external `yt_astro_analysis` package.

yt Reference Implementation

- yt has grown organically within the astrophysics domain with needed functionality leading development, and astro specific attributes referenced through the code.
- Removing the astro specifics from the general functionality in yt and adding domain agnostic attributes will create space for a more general mental model as the foundation of yt.
- Relocating the astro code to its own module with other scientific domains will allow users to find and use attributes that are tailored to their domain, without astro interference.

Metadata and Data Model

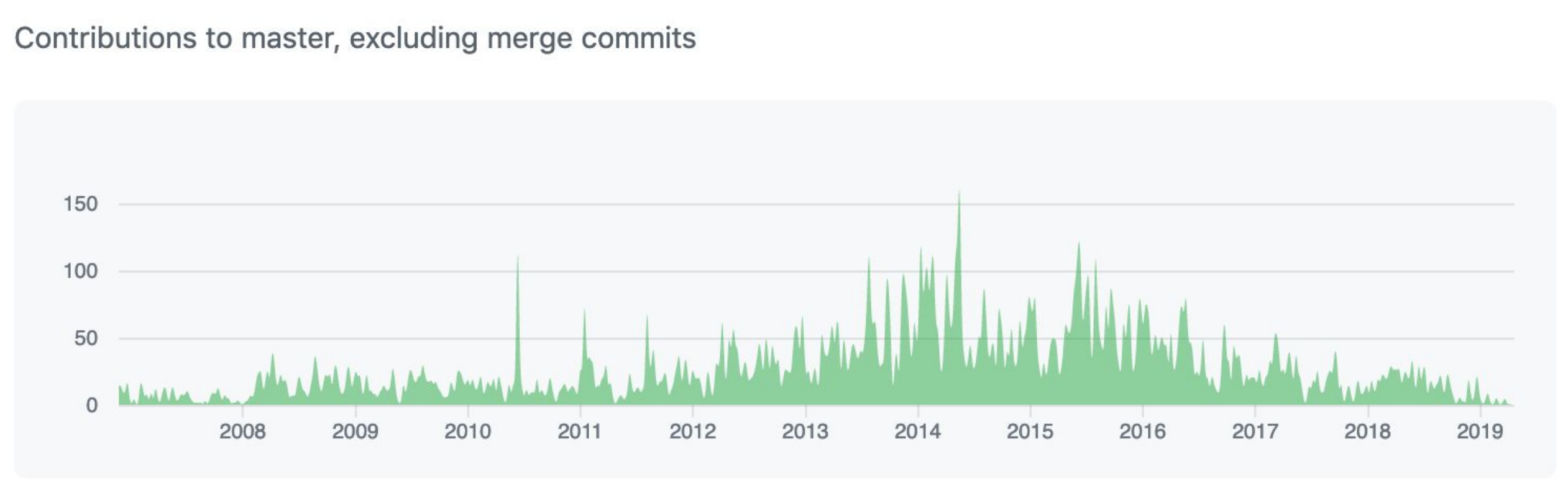


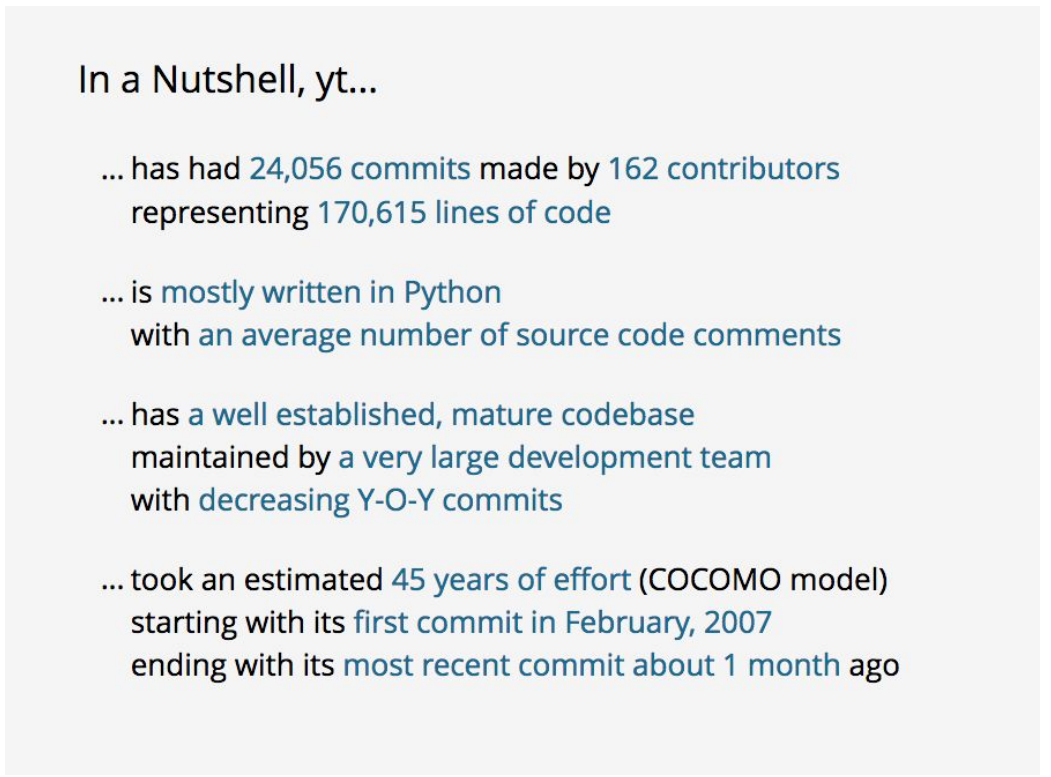
Development Highlights

- Since the yt 3.5.1 release in February 2019, yt has had:
  - 980 new commits to the repository
  - With contributions from 37 unique individuals, 15 of those are first-time contributors
  - 195 new pull requests have been submitted to the repository, and 150 of those have been merged
- New features include support for using unyt instead of yt.units, two new frontends (adaptahop and amrvac), support for cf-compliant data, exporting data to firefly, and support for geographically-located data support with cartopy

yt Community Numbers

Contributions to master, excluding merge commits





In a Nutshell, yt...

...has had 24,056 commits made by 162 contributors
representing 170,615 lines of code

...is mostly written in Python
with an average number of source code comments

...has a well established, mature codebase
maintained by a very large development team
with decreasing Y-O-Y commits

...took an estimated 45 years of effort (COCOMO model)
starting with its first commit in February, 2007
ending with its most recent commit about 1 month ago

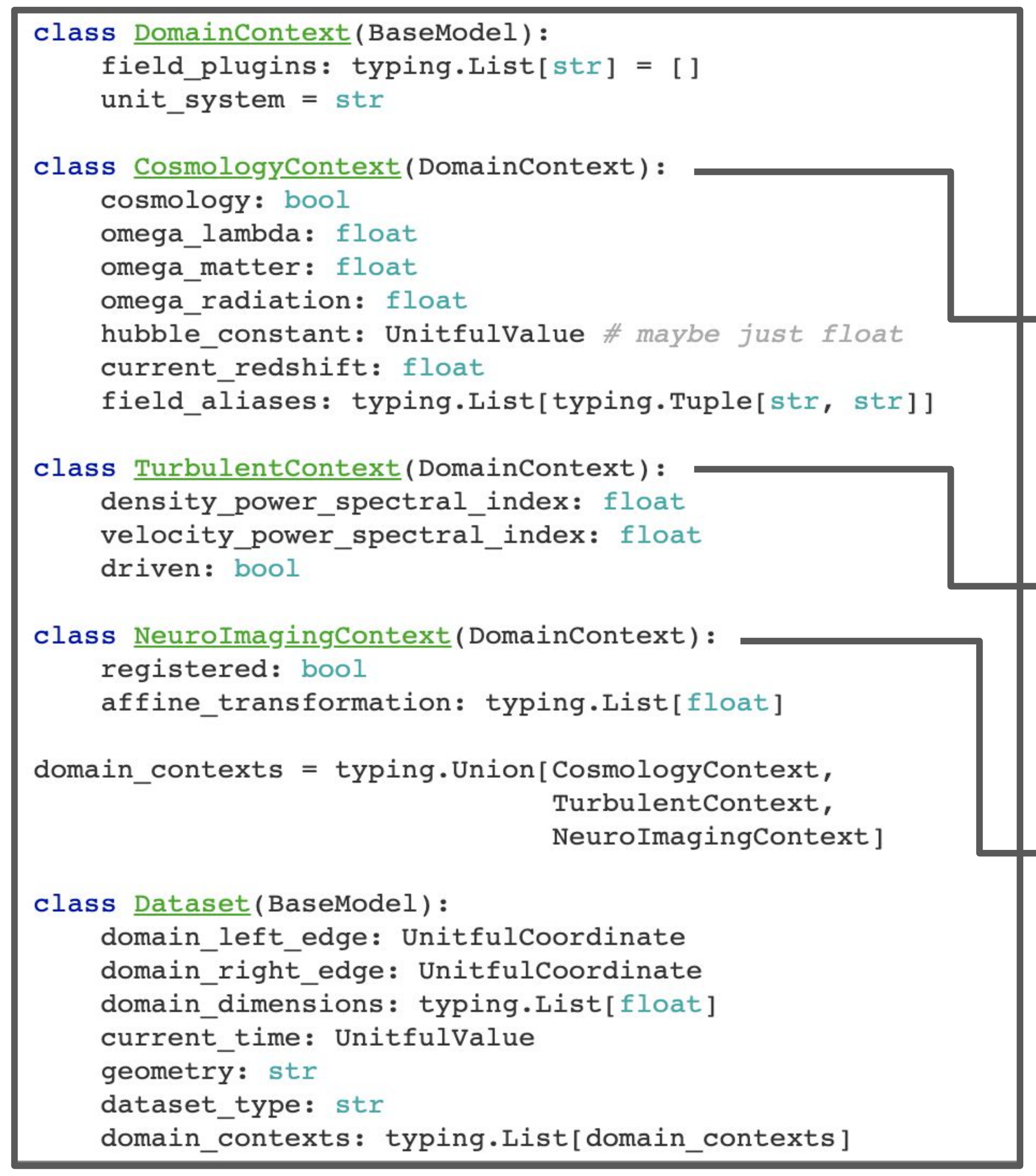
This work is supported by NSF SI2-SSI OAC-1663914 (`goo.gl/6w25zy`)

Data Representation

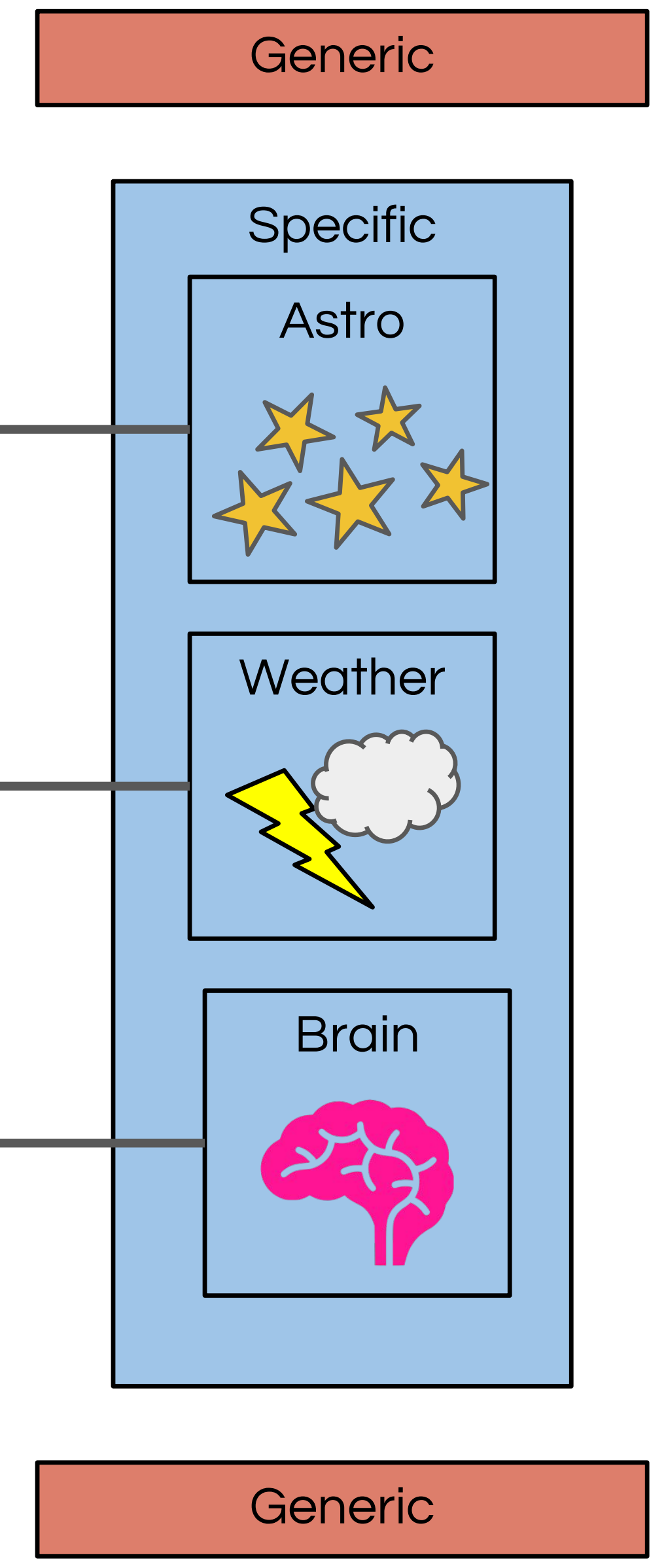
Domain specific file formats outline a number of interesting challenges as they are the entry point for loading data into yt. Efforts to accommodate file formats have encouraged development to tackle:

- Dependencies
- Code redundancy
- Encoding
- Metadata

This requires understanding common file formats, other software packages used, and the method behind the data storage.



```
class DomainContext(BaseModel):
    field_plugins: typing.List[str] = []
    unit_system = str

class CosmologyContext(DomainContext):
    cosmology: bool
    omega_lambda: float
    omega_matter: float
    omega_radiation: float
    hubble_constant: UnitfulValue # maybe just float
    current_redshift: float
    field_aliases: typing.List[typing.Tuple[str, str]]

class TurbulentContext(DomainContext):
    density_power_spectral_index: float
    velocity_power_spectral_index: float
    driven: bool

class NeuroImagingContext(DomainContext):
    registered: bool
    affine_transformation: typing.List[float]

domain_contexts = typing.Union[CosmologyContext,
                               TurbulentContext,
                               NeuroImagingContext]

class Dataset(BaseModel):
    domain_left_edge: UnitfulCoordinate
    domain_right_edge: UnitfulCoordinate
    domain_dimensions: typing.List[float]
    current_time: UnitfulValue
    geometry: str
    dataset_type: str
    domain_contexts: typing.List[domain_contexts]
```



Acknowledgements

We would like to thank major yt contributors including Nathan Goldbaum, and the larger yt community for their sustained effort. This poster was developed in collaboration with community members Madicken Munk, Samantha Walkow and Kacper Kowalik.