A Format Independent Data Interface for Astronomy

"...because I don't care what format the spectra is in!!!"

Why "Format Independent"?

While the computer literate among us discuss and debate endlessly the advantages and disadvantages of various data formats, most astronomers just want to get on with doing science. They only care about the format of data if they can't open and interact with their own and communal data seamlessly in their analysis tool of choice.

A uniform Data Program Interface (or Data API) for astronomy simplifies not only the options for the astronomer to interact with data from various sources and in various formats, but also offer the possibility for developers to support multiple formats and sources with no additional development cost.

Context

More and more, astronomy is becoming a data driven rather than observation driven science. Astronomers now care as much about bringing together diverse existing data sets and applying new analysis as about collecting new data and applying existing analysis methods. Both of these depend on interacting with data in a variety of formats and places (FITS, HDF, cloud based, database, files, etc.)

The IVOA has already developed a comprehensive set of standards covering data storage (VOTables), data communication (TAP, SAMP), and data typing (Universal Content Descriptors) to help make interaction with data as seamless as possible. FIDIA can take advantage of that infrastructure where available, but can work around its absence when necessary to provide a consistent experience for the astronomer for all data sets.

What is FIDIA?

FIDIA is a Python package for interacting with data for astronomical objects in a consistent and intuitive manner, regardless of the source or format of the data. It also provides a standardised, high level data model, organising data by astronomical object and physical meaning.

The package accepts that data will always be available in a wide variety of formats and places, and therefore it provides a formulaic way to connect to and import from new data sources such as on the user's computer. The package takes care of mundane details such as memory management, local caching, unit tracking, etc.

The reference package is written in Python, but implementations in other languages are possible.

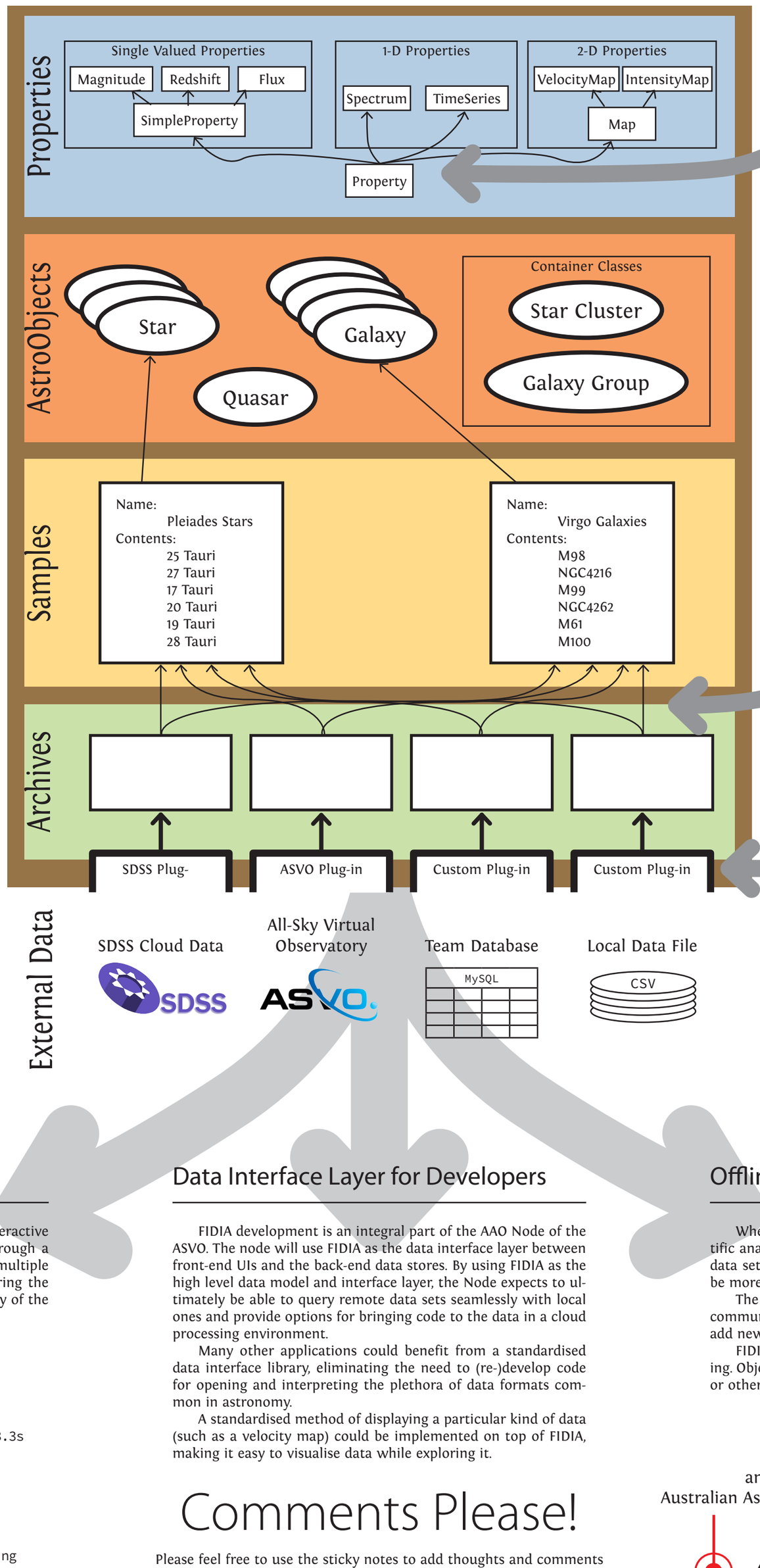
Goals

FIDIA seeks to free the astronomer to think about stars and galaxies and their spectra, magnitudes, velocities and other properties by combining a standardised, high-level data model with community developed interfaces to many common data sources.

Analysis code written using FIDIA will be highly portable, making it easier to apply existing analysis methods to new data, or bring existing data into new analysis.

Following the success of astropy, FIDIA invites community contributions to reduce the need for astronomers to re-invent data import code or develop new data models. Plug-ins for data access can read both local and remote data, eliminating the need to download large remote data sets, and allowing code to be uploaded for "bring code to the data" style execution.

FIDIA



Object Oriented Data Model

The data model is object oriented, with classes of objects inheriting from more general classes. Inheritance reduces repetition of code, making new functionality easy to deploy across the data model.

Properties are the quantities associated with AstroObjects. These are smart objects that include

- units
- errors
- upper or lower limits as necessary
- description of what they are
- comments
- dimensionality

AstroObjects such as Stars, Galaxies, or Quasars define the standardised vocabulary of properties (high level data model) that an astronomer might expect to find for each, reducing the need to troll through a schema browser.

In some cases, AstroObjects may be able to infer properties not explicitly present.

Built in cross matching

Any time data from multiple archives are brought together, they must be cross matched. Therefore, tools for doing cross matching are critical to this data interface. The following strategies are proposed:

- Attaching a new archive to the sample includes the cross matching
- There are default/standard schemes available, such as match on ID or position with tolerance
- Cross matching can be done in a user defined way by providing a custom cross-matcher.

The Sample is the basic access point for FIDIA.

Samples define collections of AstroObjects and add and maintain connections to archives of data. It is through the sample that objects and their properties ("the data") are accessed.

Archives handle accessing the data through an associated plug-in as needed, and may cache data locally for remote data sources.

Plug-ins themselves are programmatic descriptions of how to retrieve a given piece of data (something that might have to be written anyway if not already present). Plug-ins can execute arbitrary Python code to access the necessary data.

Plug-in Architecture

Simple community contributed or custom plug-ins connect arbitrary data to the data model:

```
@redshift
def z_helio(self):
    data = np.genfromtext("my_data.dat")
    # Redshift in the third column
    return data[2]

@magnitude.r_band
def sdss_r_mag(self):
    res = sdss_sql("SELECT mag_r "
        + "FROM PhotoObj "
        + "WHERE PhotoObjID = " + self.id)
    return res[0]
```

Data to be accessed can be in an arbitrary format, remote or local. It might be accessed through a web-page or VO protocol.

Interactive Data Exploration

FIDIA is being developed with a focus on easy, interactive data exploration. Rather than spending time trolling through a schema browser or figuring out how to bring together multiple data sources, time should be spent analysing and exploring the data. Introspection in e.g. iPython allows for self discovery of the data model.

An Example session:

```
>>> mysample = \
    Sample.new_sample_from_archive('dynamo')
>>> mysample
['HfluxLz_15-3', 'MfluxLz_21-1']
>>> mysample['HfluxLz_15-3']
Galaxy "HfluxLz_15-3" at 15h34m13.4s -04d34m03.3s
>>> mysample['HfluxLz_15-3'].redshift
0.05432
>>> mysample.add_archive_for_existing('SDSS')
>>> mysample['HfluxLz_15-3'].image['SDSS-r']
<<image>>
>>> mysample['HfluxLz_15-3'].redshift
WARNING: Multiple redshifts availabe, displaying
from primary Archive "DYNAMO";
0.05432
>>> mysample['HfluxLz_15-3'].redshift.all
['dynamo': 0.05432, 'sdss': 0.054291]
```

Data Interface Layer for Developers

FIDIA development is an integral part of the AAO Node of the ASVO. The node will use FIDIA as the data interface layer between front-end UIs and the back-end data stores. By using FIDIA as the high level data model and interface layer, the Node expects to ultimately be able to query remote data sets seamlessly with local ones and provide options for bringing code to the data in a cloud processing environment.

Many other applications could benefit from a standardised data interface library, eliminating the need to (re-)develop code for opening and interpreting the plethora of data formats common in astronomy.

A standardised method of displaying a particular kind of data (such as a velocity map) could be implemented on top of FIDIA, making it easy to visualise data while exploring it.

Offline Processing and Analysis

When used as a basis for implementing algorithms for scientific analysis, FIDIA makes analysis code highly portable to other data sets. New analysis methods written on top of FIDIA should be more attractive for other astronomers.

The standardised data model of FIDIA is extensible through a community driven process similar to astropy, making it easy to add new functionality.

FIDIA is designed to be reasonably efficient in offline processing. Objects of the data model are created only as needed. Remote or other data that is costly to access is cached locally during.

Comments Please!

Please feel free to use the sticky notes to add thoughts and comments

Poster by:
Andy Green
andrew.green@aao.gov.au
Australian Astronomical Observatory