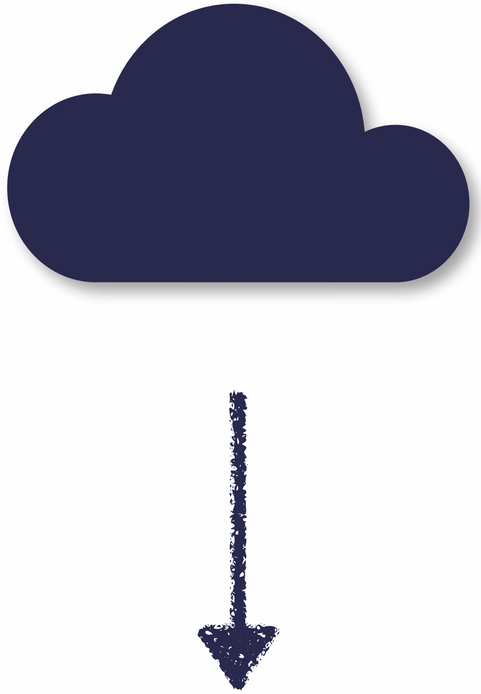
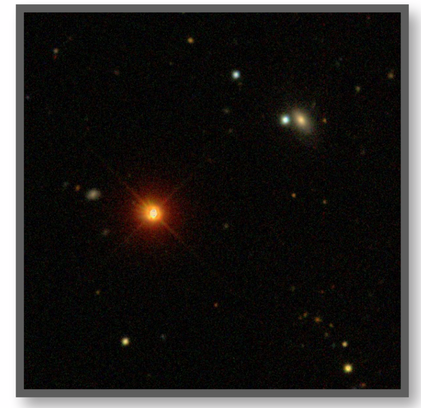
Accessing Heterogeneous Data: An Introduction to Science Data Descriptors

Demitri Muna
University of Texas, San Antonio
@demitrimuna





dotAstronomy Toronto • 22 October 2019

Data Access Should Be Simple.

I think from now on this will be my first slide of any talk I give.

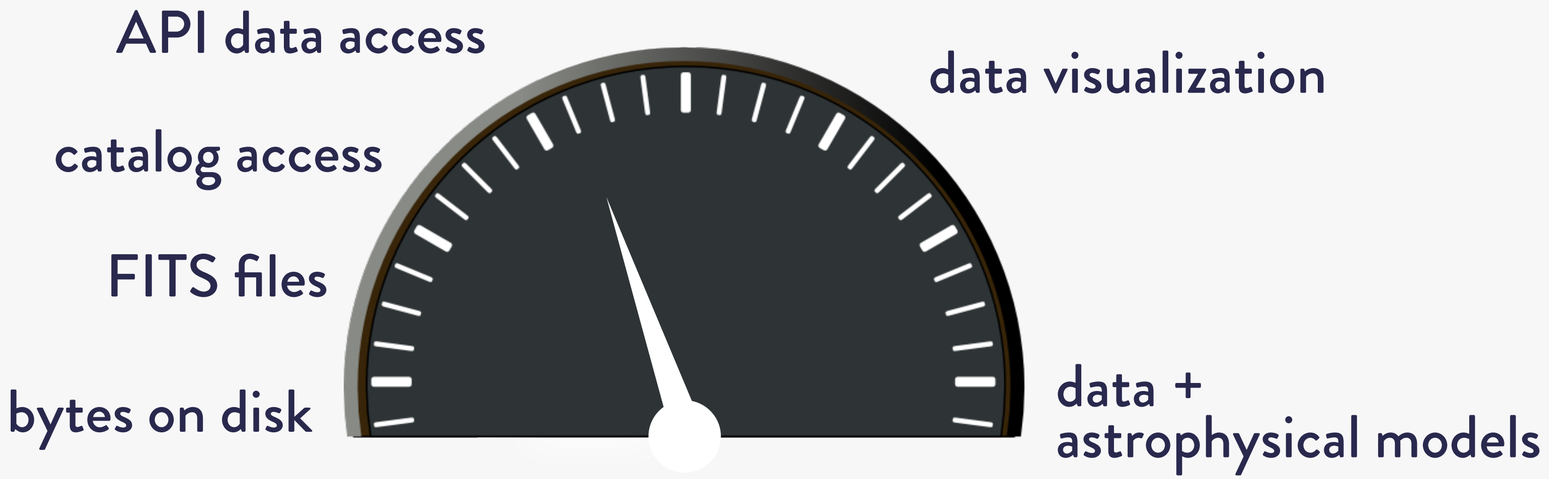
We spend a lot of time here...

...trying to get here.

bytes on disk

data + astrophysical models

Data Access Should Be Simple.

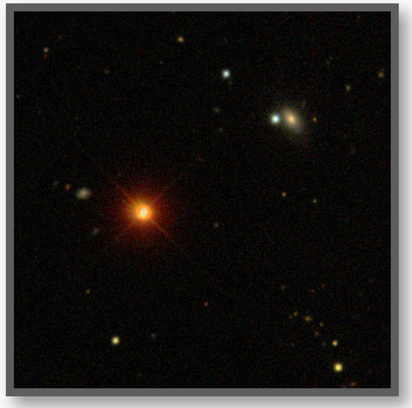


Let’s move the needle forward.

Working With FITS Files.

We're all familiar with this...

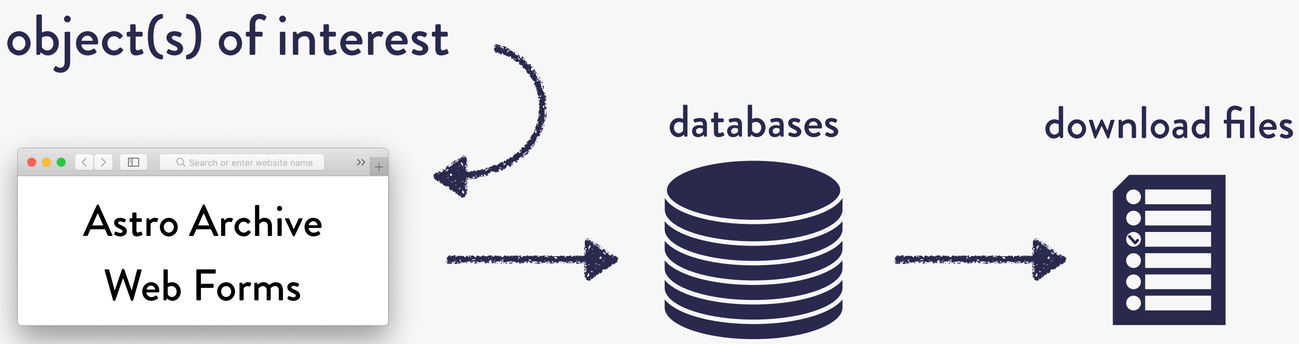
```
from astropy.io import fits

filepath = "data.fits"
hdu_list = fits.open(filepath)
hdu1 = hdu_list[0]
data = hdu1.data

# work with data here...
```



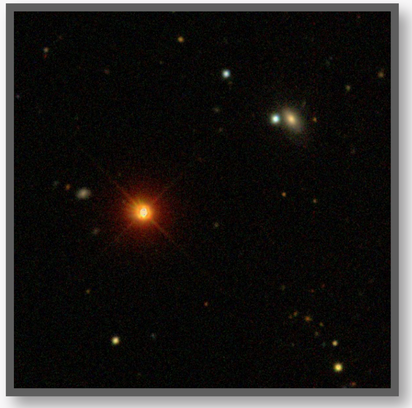
Working With FITS Files.

We're all familiar with this...

...but this skips a few steps.




```
from astropy.io import fits

filepath = "data.fits"
hdu_list = fits.open(filepath)
hdu1 = hdu_list[0]
data = hdu1.data

# work with data here...
```

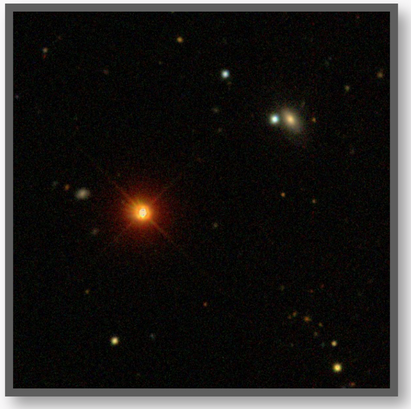


Files Are Implementation Details.

...burn all the files...

object(s) of interest

```
from resolver import ObjectResolver

data = ObjectResolver(...)

# work with data here...
```



A lot of working with data is bookkeeping. This should be abstracted away as much as possible.

There are examples of standardization, but they are almost all still file-based.

Data Access Beyond Web Forms.

An example of abstracting data access from individual files.

Imagine a Python interface to data. Do you need an image from WISE or a spectrum from SDSS? Just query it directly from code.

```
from service.data.SDSS import SDSSSpectrum
from service.data.SDSS import SpectrumQuery
from service.data.WISE import WISEImage
from service.data.GALEX import GALEXImage

spectra = SDSSData.SpectrumSearch(z=[zmin, zmax],
                                  kind='galaxy',
                                  mag=[man_min, mag_max])
for s in spectra:
    print(s.associated_2MASS_sources())
```

the SDSS file doesn't know about 2MASS...

Do you already know which one you need? Get it directly.

```
s = SDSSSpectrum(plate=3463, mjd=55684, fiber=12)
g = GAIASource(source_id=34875987623)
```

File access is completely transparent: this code automatically downloads the files to your computer into a cache (implementation detail). If you have already run the code, the data is found locally.

Syntax is clear, logical, easy to read without prior knowledge.

Scientific Data Descriptors.

A data descriptor is an identifier that uniquely points to a set of data or single datum. Goals for such a scheme:

- Descriptor easily generated.
- Descriptor reasonably human readable.
- Descriptor is machine readable.
- Does not require data creator to define descriptor.
- Refer only to publicly released data sets.
- File format / data storage agnostic.
- Descriptors are easily citable, searchable.
- Descriptors are stable.

Why Do We Need This?

Data is heterogeneous.

- we standardize on FITS, but data can be organized in multiple ways

Big data.

- as the data volume increases, the ability to learn the formats and interfaces of each data set will get away from us

Cloud computing.

- as out analysis of data moves to the cloud, it may be more efficient to not work with files
- why download 400GB of data to extract a small part?
- our code shouldn't rely on knowing where data is or how to get it

Instant, random access to *everything*.

- if I want a cutout of DES, or five specific columns of ten Gaia objects, I should get that data directly and nothing else

Prior Art.

Research Resource Identifier (RRID)

"RRIDs are persistent and unique identifiers for referencing a research resource and are used for promoting research resource identification and tracking. Catalog numbers change, disappear or can be reused for another resource, but RRIDs always resolve to the same research resource and endure beyond the existence of the research resource itself."

Examples of RRIDs:

• Antibody: Estrogen receptor beta antibody	RRID:AB_10618531
• Cell line: HK-2 [Human kidney]	RRID:CVCL_0302
• Mouse model: TSOD.C-Nidd6	RRID:IMSR_-CARD:280
• Software: ANOVA	RRID:SCR_002427

Reference: https://dknet.org/about/rrid

Prior Art.

- Conceived at NIH meetings in 2010-11.
- Hundreds of bio & genomic journals now request authors to provide RRID identifiers.
- Descriptors are Google-searchable.
- RRIDs don't really fit the model for astronomical data.

Research Resource Identifier (RRID)

"RRIDs are persistent and unique identifiers for referencing a research resource and are used for promoting research resource identification and tracking. Catalog numbers change, disappear or can be reused for another resource, but RRIDs always resolve to the same research resource and endure beyond the existence of the research resource itself."

Examples of RRIDs:

• Antibody: Estrogen receptor beta antibody	RRID:AB_10618531
• Cell line: HK-2 [Human kidney]	RRID:CVCL_0302
• Mouse model: TSOD.C-Nidd6	RRID:IMSR_-CARD:280
• Software: ANOVA	RRID:SCR_002427

Reference: https://dknet.org/about/rrid

Why Not DOIs?

DOI = digital object identifier

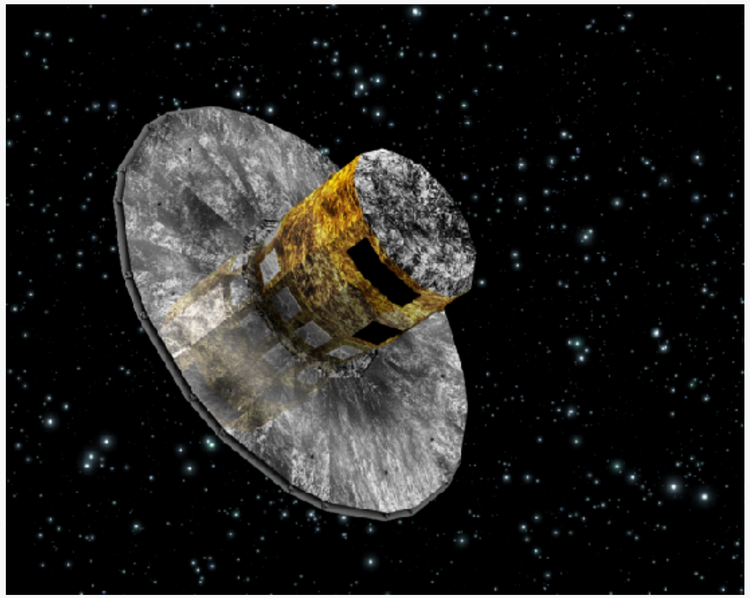
- DOIs are appropriate for the data set itself, not an individual row in a table or gene from a larger sequence.

- Can't expect or wait for data creator to “mint” a DOI. Some won't for just the data, referring one to the methods papers to cite. Other data sets are decades old and the creators have long disbanded.

- Don't want to mint 1.3 billion DOIs for each Gaia source. Identifiers should only be created/stored when annotations have been created.

Example Data Set: Gaia Data Release.

As an example, let's use Gaia. This is an ESA mission to precisely measure the distance to 1% of the stars in the galaxy.



- 1.3 billion records (stars)
- tabular format
- each row has unique integer identifier
- many columns for each row

Scientific Data Descriptor Scheme.

Descriptor utilizes the URN (uniform resource name) scheme, which is a kind of URI (uniform resource identifier).

scientific field

data release / product version

`urn:scidd:astro:gaia:dr2:4038848092638268032`

"science data descriptor"

data source

unique identifier defined in data set

This descriptor would point to the entire record of one object in the Gaia catalog.

The descriptor does not uniquely *locate* data (URL), but can, similar to DOIs:

`http://scidd.org/astro/gaia/dr2/4038848092638268032`

Scientific Data Descriptor Scheme.

Use URL-style fragments to refer to individual columns. Queries can specify subsets of data.

This refers to the columns "ra" and "dec" (coordinates) of the record.

```
urn:scidd:astro:gaia:dr2:4038848092638268032#ra,dec
```

References to subsets of the data can be created using the URL-style query language:

```
urn:scidd:astro:gaia:dr2?ra=10,20&dec=-1,1
```

Example: Gaia Catalog Object.

HTTP request:

```
http://scidd.org/data/gaia/dr2/4038848092638268032
```

Response:


```
{
    "solution_id": 1635721458409799680,
    "source_id": 4038848092638268032,
    "random_index": 171162838,
    "priam_flags": 112001,
    "phot_rp_n_obs": 12,
    "rv_nb_transits": 0,
    "astrometric_n_obs_al": 144,
    "astrometric_n_obs_ac": 0,
    "astrometric_n_good_obs_al": 143,
    "astrometric_n_bad_obs_al": 1,
    "astrometric_params_solved": 31,
    "astrometric_matched_observations": 16,
    . . .
}
```

Example: Gaia Catalog Object, Specific Fields.

HTTP request:

```
http://api.trillianverse.org/data/gaia/dr2/4038848092638268032/dm/ra,dec
```

Response:

```
{
    "ra": 271.7996065838753,
    "dec": -35.388946505502204
}
```

Example: Gaia Catalog Object, Query.

HTTP request:

```
http://api.trillianverse.org/data/gaia/dr2/query?
    ra=12.34&dec=-34.56&radius=70&columns=ra,dec,parallax
```

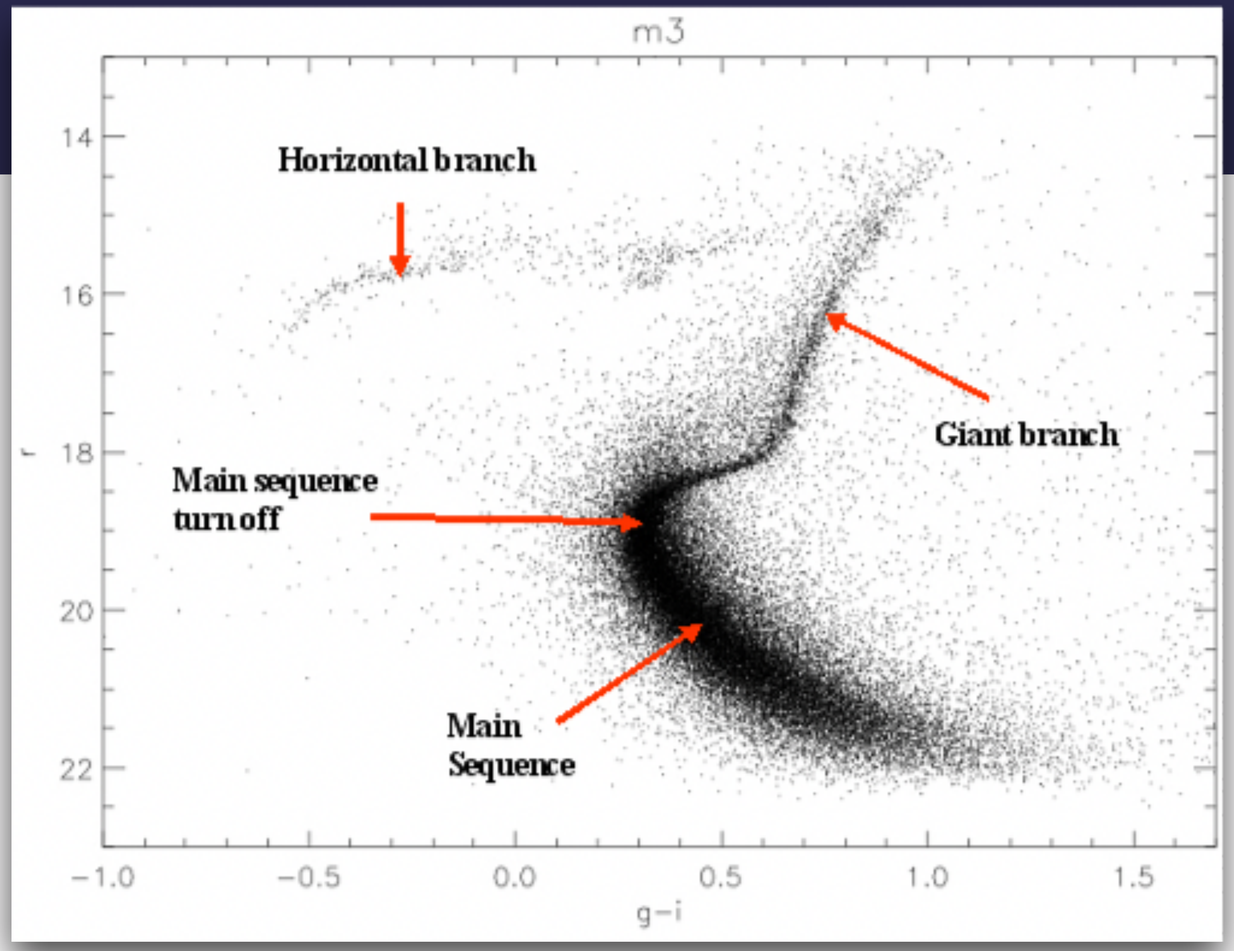
Response:


```
[
    {
        "ra": 12.359836571088403,
        "dec": -34.562767623943586,
        "parallax": null
    },
    {
        "ra": 12.325847098482923,
        "dec": -34.5691975479172,
        "parallax": 1.5041040145727502
    },
    {
        "ra": 12.361496461687954,
        "dec": -34.56199191905366,
        "parallax": 0.43713623958978876
    },
    {
        "ra": 12.332841680244684,
        "dec": -34.55693438849452,
        "parallax": 1.0882559941901266
    }
]
```

Example Use Case: Data Visualization.

M3 Globular Cluster

Plot of all objects identified as “star” from SDSS.



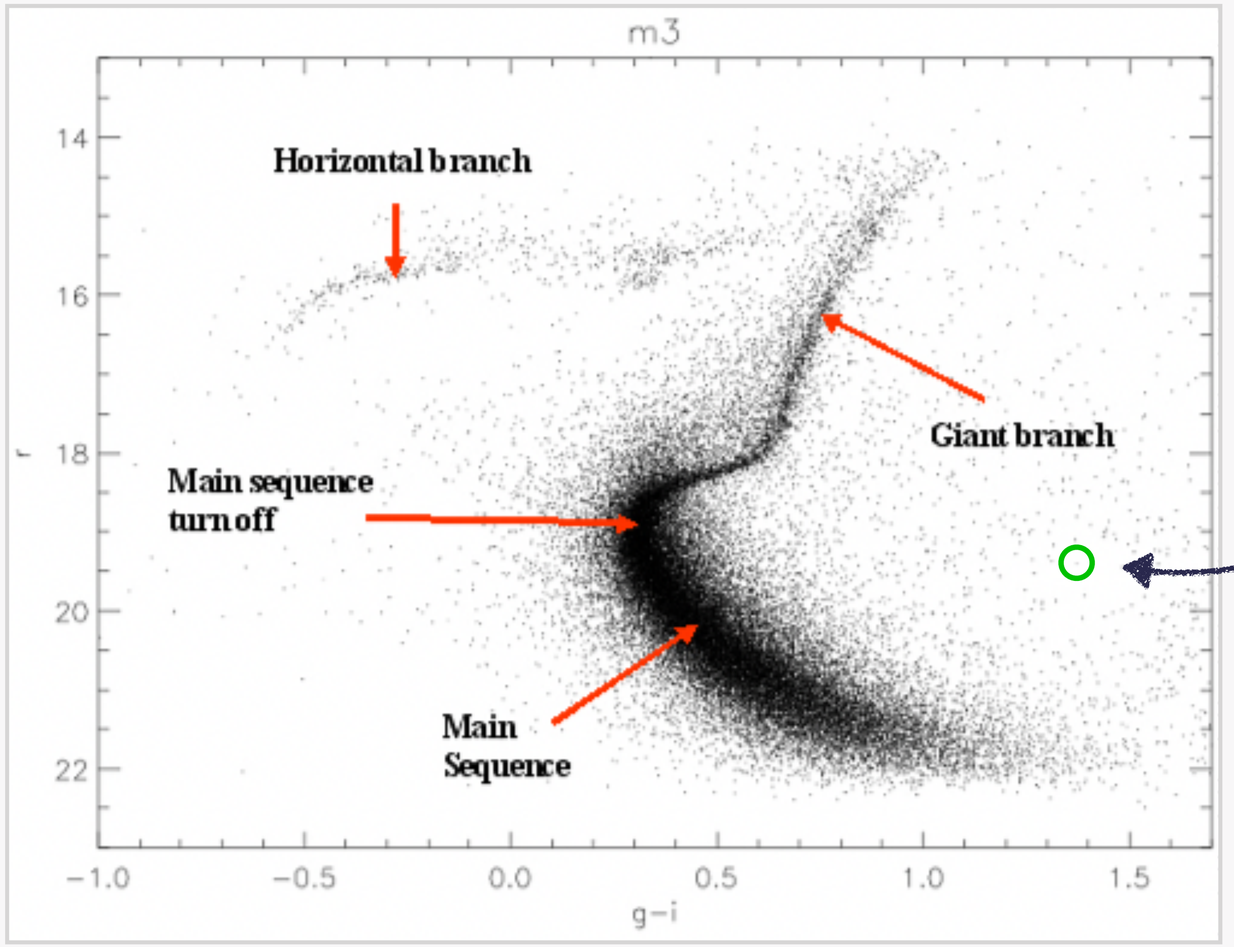
~500,000 stars

Plot credit: Galactic Discovery Project
Photo credit: Sloan Digital Sky Survey

Example Use Case: Data Visualization.

- Two unresolved stars?
- Not a star?
- Star not part of M3?
- Instrument effect?
- Processing error?
- Something new?

M3 Globular Cluster



Clear trends that match physical models visible... but what is this outlier?

Example Use Case: Data Visualization.

M3 Globular Cluster

To study outliers, we look at the individual data point. This requires human intervention, domain knowledge, and expertise.

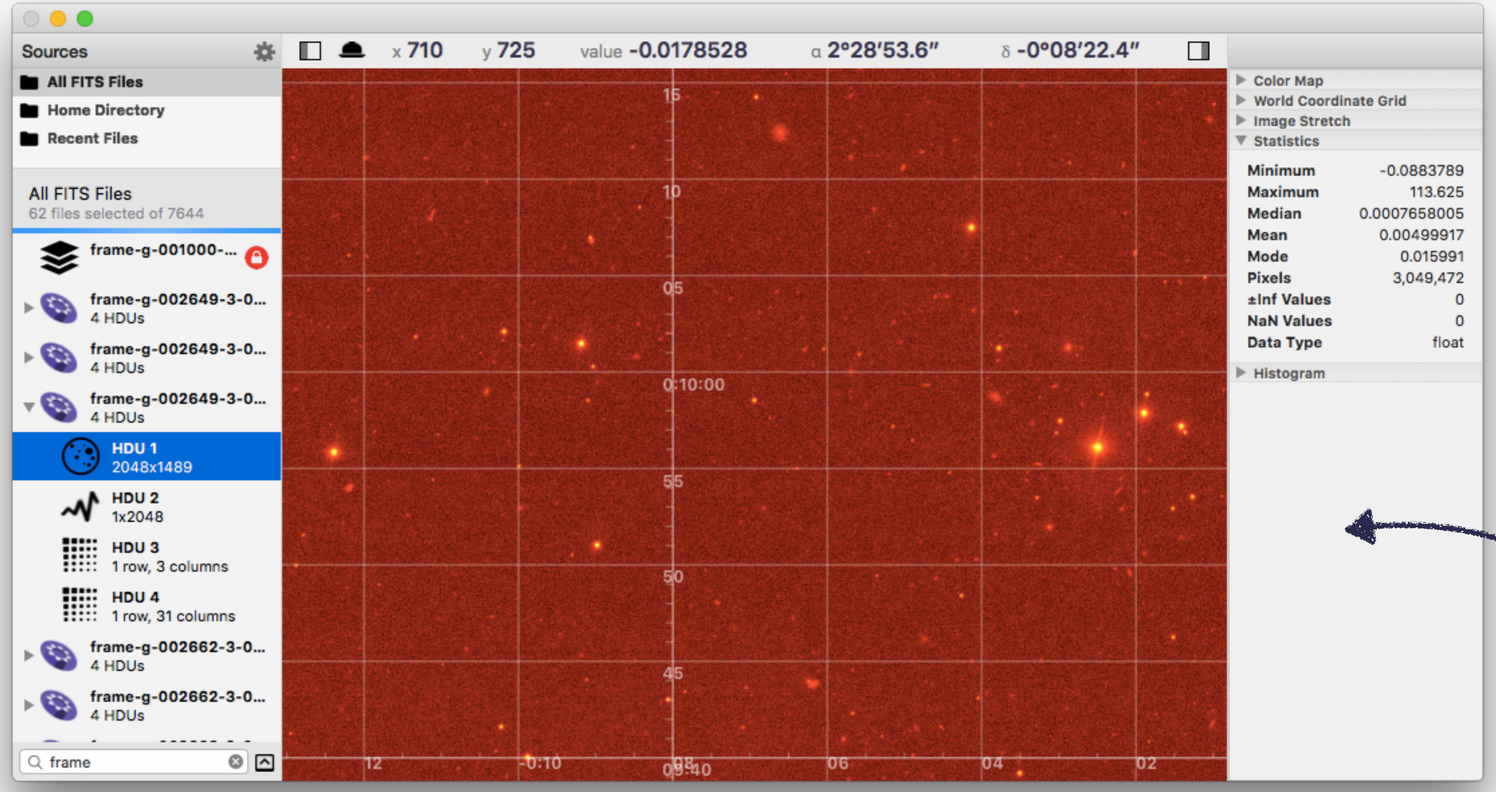
Once the outlier is identified... nothing happens:

- Data set is not updated (releases are static).
- Classification is not updated.
- Bad data is not flagged.

The outlier will need to be identified again by the next scientist.

Community Annotation in User Interfaces.

Annotation from the community can be attached to the identifier.



Integrate community annotation to common data visualization tools.

In a view that displays the data, automatically identify any objects in the data that have annotations.

Community annotations displayed here.

Benefits of Community Annotation.

- User interfaces can attach annotations to any data visualization (e.g. web, desktop).
- Researchers can mark data (“textbook example of x”, “instrument error”, etc.) that others can learn from.
- Automated classification, e.g. bots that combine several data sets and assign likelihood values of model fits
- Educational – “looking over shoulder” of domain expert.
- Dramatically lowers bar for sharing knowledge and research – comments can be made and viewed long before the publication process.
- Serendipity – open a file for one purpose, draw connections from comments from another researcher.

Hypothes.is

A back end for annotation.

Hypothesis is a web plugin that lets you annotate web pages and is focussed on academia. This is widely used in journalism and literature circles.

The objects in their scheme to annotate are web pages, where the identifier is the URL. The science data descriptor takes the same form (URI) and can be used exactly the same way.

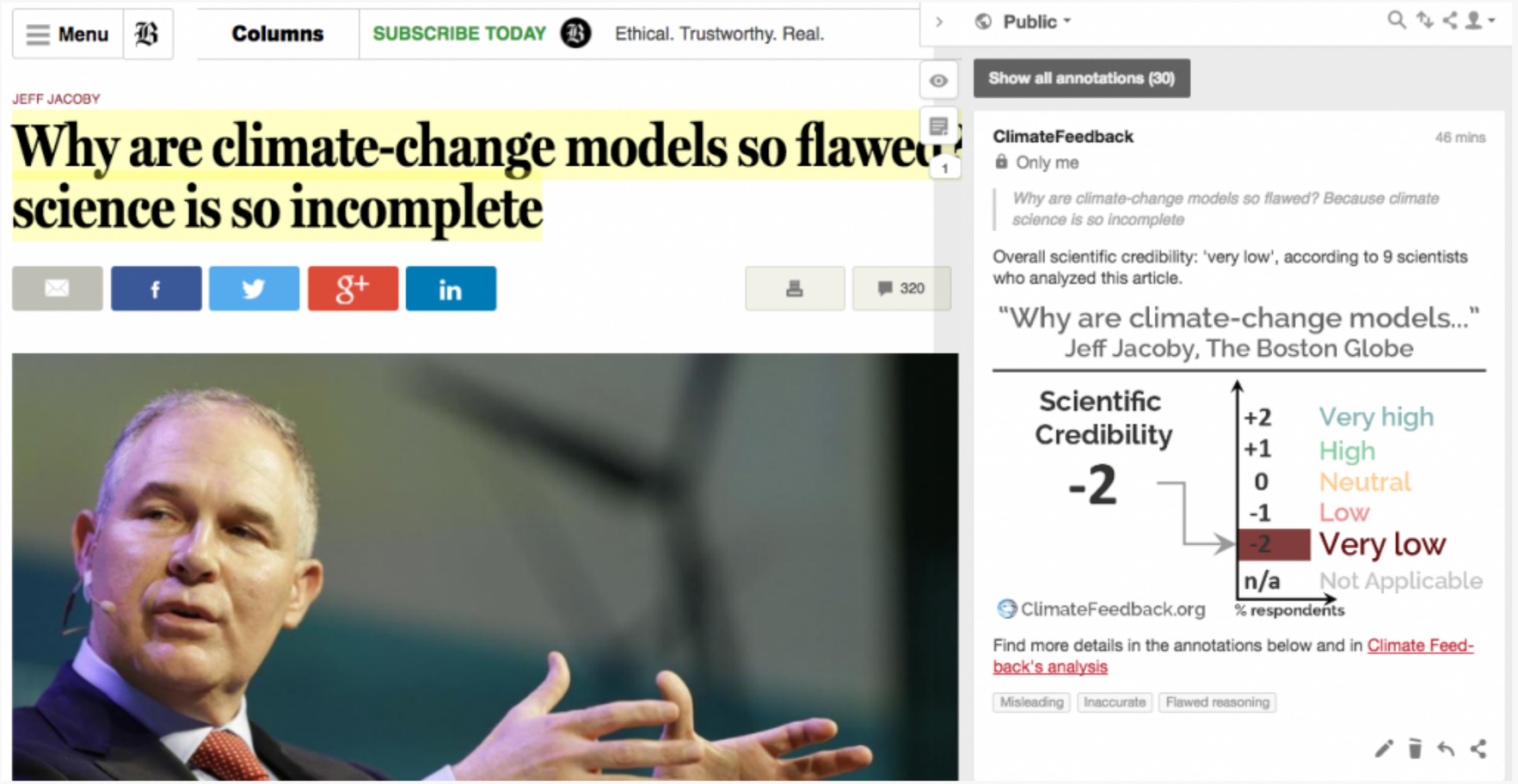
Annotations could be added to (dynamic) web pages, one for each object. Other hooks would need be made for any custom user interface.

https://hypothes.is

Hypothes.is Example.

It's like a hidden web.

Journalism and fact-checking.



https://web.hypothes.is/journalism/

Additional Use Cases.

Custom User Interfaces.

Anyone can create a “custom archive” for their specific research without the need to host data.

Journal Searches.

Has anyone published anything about this object? A service can map IDs to journal papers.

Education.

Very simple scripts can be written for lessons for students who are new to FITS files.

Cloud Computing.

No need to manage potentially millions of individual files.

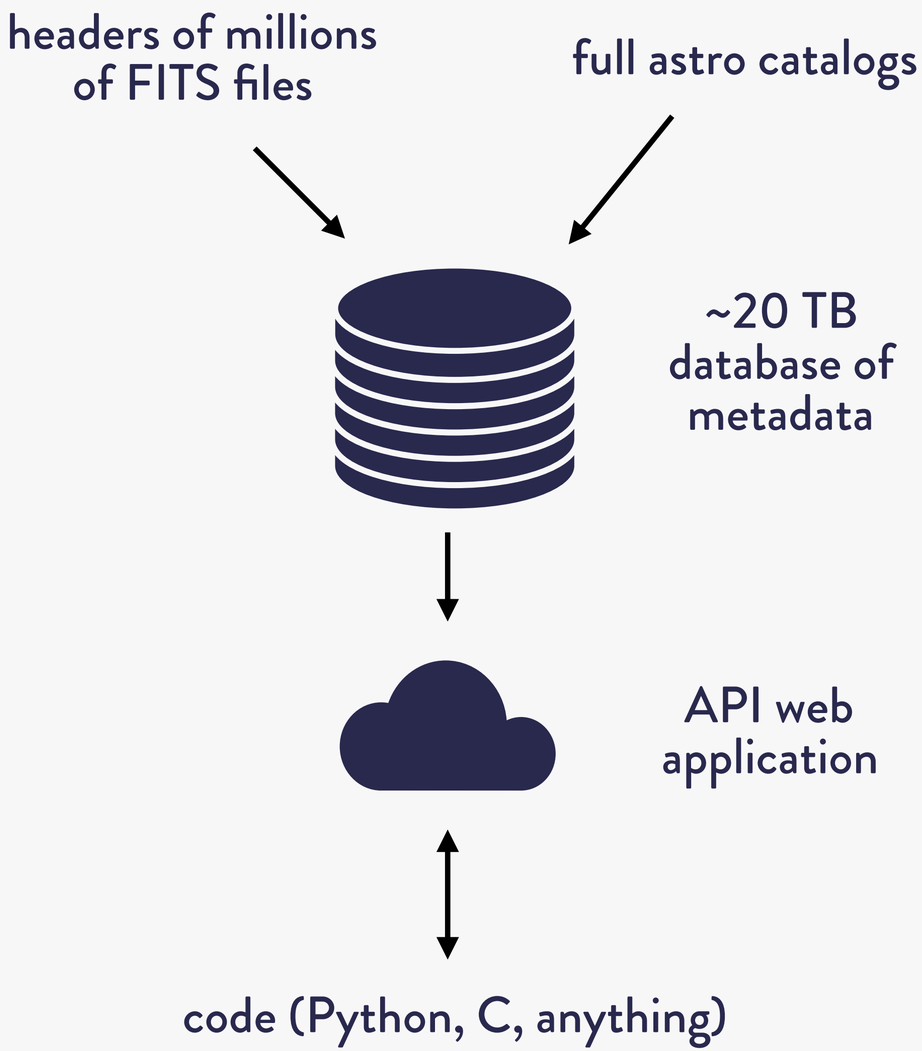
How Would This Be Implemented as a Service?

Or rather, how did I implement this?

Currently Indexed:

- Gaia DR2
- GALEX
- 2MASS
- WISE
- SDSS photometry
- a few others

Many more to be added soon...



Status and Future Work.

A database and web application are serving a few data sets, with more to be added.

Return associated objects: image masks, weight maps, available spectra, cross catalog results, etc.

Extend scheme to models, e.g. PSF for any data set, position.

Extend scheme to regions or objects in images (see Web Annotation Data Model: https://www.w3.org/TR/ annotation-model/).

A specification will be written after some real-world testing.

User feedback and potential use cases welcome now!

Possible Hack Day project?