STScI | SPACE TELESCOPE SCIENCE INSTITUTE

EXPANDING THE FRONTIERS OF SPACE ASTRONOMY

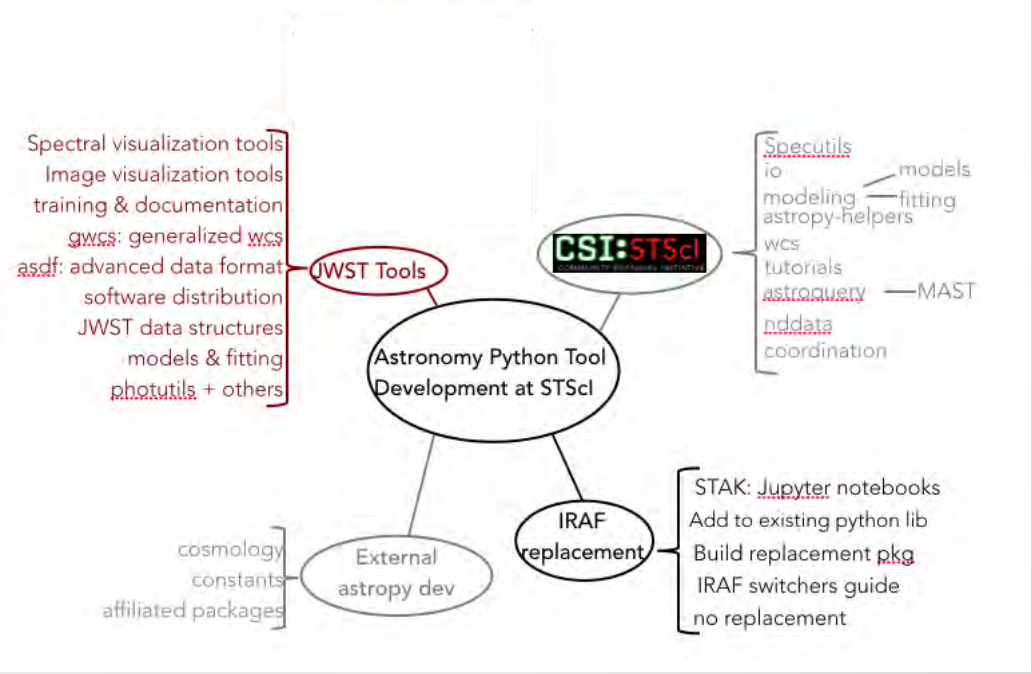
Data Analysis Tools for the HST Community

STUC 2019

Erik Tollerud
Data Analysis Tools Branch
Project Scientist

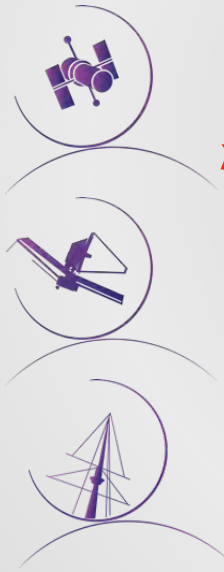
What Are Data Analysis Tools? The Things After the Pipeline

- DATs: Post-Pipeline Tools
  - Analysis Tools
    - Astropy
    - Photutils
    - Specutils
  - Visualization Tools
    - Python Imexam (+ds9)
    - Ginga
    - SpecViz
    - (MOSViz)
    - (CubeViz)



Data Analysis Software at STScI is for the Community

Software that is meant to be used by the astronomy community to do their science. This talk is about some newer developments of interest to the HST user community.



photutils

3. specutils

imexam

astroquery

2. The Astropy Project

FITS The Astronomical Image and Table Format
Flexible Image Transport System

The STAK Project

1.

IRAF
Image Reduction and Analysis Facility

I R A F
Image Reduction and Analysis Facility

- IRAF was amazing for its time, and *the* DAT for generations of astronomers

IRAF - Image Reduction and Analysis Facility

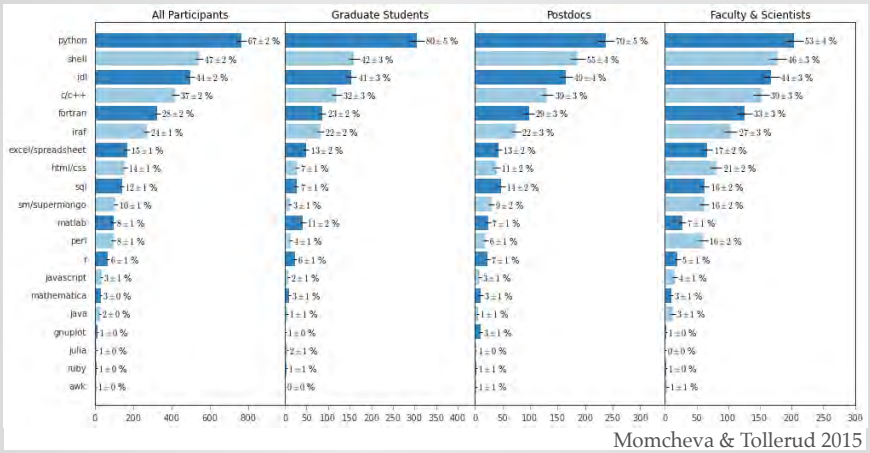
NOAO is transitioning IRAF to an end-of-support state, and has taken NOAO's IRAF distribution offline pending a final copyright and licensing review of the source code. Users interested in new IRAF installations during this review period may wish to consider the following two distributions:

1. The AstroConda Legacy Software Stack
2. The IRAF Community Distribution

STScI will no longer support PyRAF as of Oct 1st, 2019. Users will still be able to access an installation of PyRAF through Astroconda. However, STScI will no longer answer help calls related to PyRAF installation, bugs, or other usage questions. All support currently provided for other packages in Astroconda will continue.

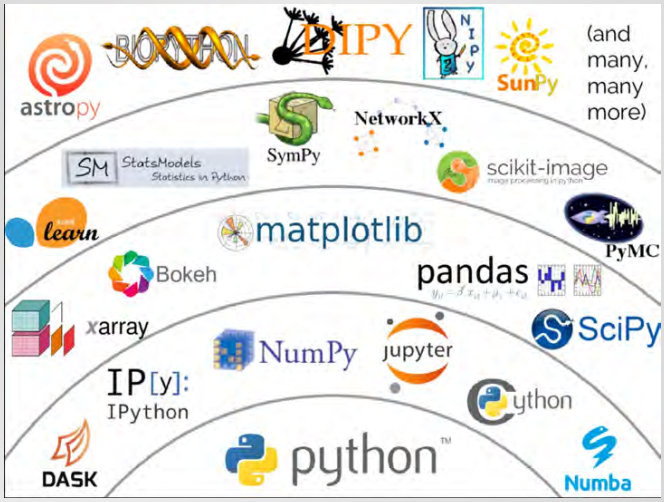
What is the long-term replacement strategy?

Python's Scientific Ecosystem Has what Astro Needs



Momcheva & Tollerud 2015

Python is now the single most-used programming language in astronomy.



And 3rd-most in industry… Largely because of the vibrant scientific and numerical ecosystem (=Data Science):

(Which HST's pipelines helped start!)

IRAF<->Python is not 1-to-1. Hence STAK

Lead: Sara Ogaz

Supporting both STScI's internal scientists and the astronomy community at large (viewers like *you*) requires that the *community* be able to transition from IRAF to the Python-world. This is the purpose of the STAK notebooks

https://stak-notebooks.readthedocs.io/en/latest/

Python Introduction

This notebook will introduce new Python and Astropy users to various aspects of the Python astronomy workflow that may help them to take better advantage of the rest of the STAK notebooks. If you are a new Python or Astropy user, we highly recommend reading through all sections of this notebook. We will discuss:

- Workflow Philosophy, the difference in workflow between IRAF and Python.
- FITS File I/O, opening and updating FITS files in Python.
- IRAF/IDL to Python gotchas.
- Helpful links and additional resources.

Fully supported via the STScI Help Desk

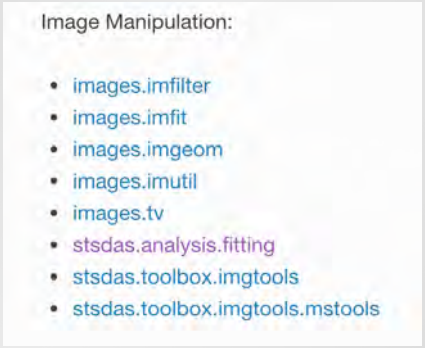


Image Manipulation:

- images.imfilter
- images.imfit
- images.imgeom
- images.imutil
- images.tv
- stsdas.analysis.fitting
- stsdas.toolbox.imgtools
- stsdas.toolbox.imgtools.mstools

(And it is all open for the user community to contribute to and improve:)
https://github.com/spacetelescope/stak-notebooks

STAK Notebooks

Example: IRAF's *images.imfilter* tasks map to several different packages and functions or objects in Python.

(And all the code is runnable if you download the notebook)

images.imfilter

The images.imfilter IRAF package provides an assortment of image filtering and convolution tasks.

Notes

For questions or comments please see our github page. **We encourage and appreciate user feedback.**

Most of these notebooks rely on basic knowledge of the Astropy FITS I/O module. If you are unfamiliar with this module please see the Astropy FITS I/O user documentation **before using this documentation.**

Python replacements for the images.imfilter tasks can be found in the Astropy and Scipy packages. Astropy convolution offers two convolution options, `convolve()` is better for small kernels, and `convolve_fft()` is better for larger kernels, please see the Astropy convolution doc page and Astropy Convolution How to for more details. For this notebook, we will use `convolve`. Check out the list of kernels and filters avaialble for Astropy, and Scipy

Although `astropy.convolution` is built on `scipy`, it offers several advantages: * can handle NaN values * improved options for boundaries * provided built in kernels

So when possible, we will be using `astropy.convolution` functions in this notebook. The ability to

Drizzlepac Notebooks

https://spacetelescope.github.io/notebooks/

Note: The notebook in this repository 'Initializtion.ipynb' goes over many of the basic concepts such as the setup of the environment/package installation and should be read first if you are new to HST images, DrizzlePac, or Astroquery.

Introduction

Even though Hubble has a small field of view, satellites are commonly captured in images. The cosmic ray rejection algorithm in Astrodrizzle is not well suited to eliminate satellite trails, and the affected adjacent pixels that make up their wings leave ugly blemishes in stacked images.

To fix this problem, the pixels around satellite trails need to be marked as bad in the affected images. There are several ways to do this. The ACS Team has developed an algorithm to automatically detect and mask satellite trails. This is the easiest and most convenient way. Masks can also be made manually using DS9 regions. While not as convenient, making masks manually allows you to mask not only satellites, but also any other anomalies with odd shapes (e.g. dragon's breath, glint, blooming).

Both methods are explained below.

In [1]:
```
import os
import shutil

from astropy.io import fits
from astroquery.mast import Observations
from astropy.visualization import astropy_mpl_style,LogStretch,ImageNormalize,LinearStretch
from IPython.display import Image
import matplotlib.pyplot as plt
import pyregion

import acstools
from acstools.satdet import detsat, make_mask, update_dq
from drizzlepac.astrodrizzle import AstroDrizzle as adriz

%matplotlib inline
```

```
The following tasks in the acstools package can be run with TEAL:
        acs2d                  acs_destripe          acs_destripe_plus
        acsccd                    acscte             acscteforwardmodel
        acsrej                    acssum                   calacs
PixCteCorr is no longer supported. Please use acscte.
The following task in the stsci.skypac package can be run with TEAL:
                                  skymatch
The following tasks in the drizzlepac package can be run with TEAL:
```

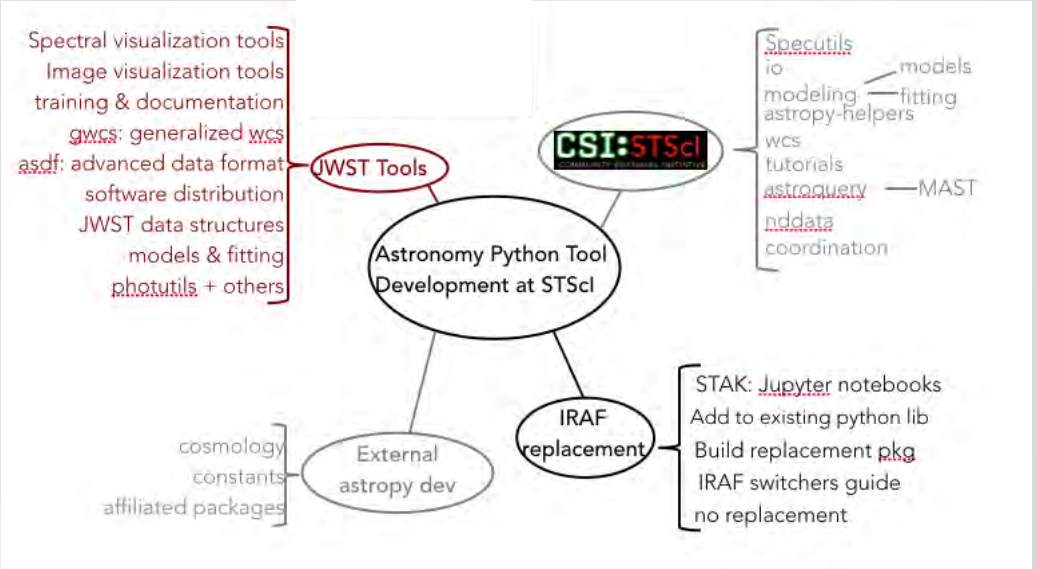
Inspired by this, a larger effort continues to write and serve Jupyter notebooks that show specific use cases to show the community how to use the Python-based DATs on HST data.

New Development: Deep Synergy w/ JWST DATs

w/Harry Ferguson, Susan Kassin

JWST DATs are almost entirely in Python, and meet many of HST's needs with only minor additions.

- Data Analysis Tools
  - Astropy
  - Photutils (w/Astropy)
  - Specutils (w/Astropy)
  - WebbPSF
- Visualization Tools
  - Python Imexam (+ds9)
  - Ginga
  - SpecViz
  - (MOSViz)
  - (CubeViz)



STScI Community Software is Built on a Shared Foundation

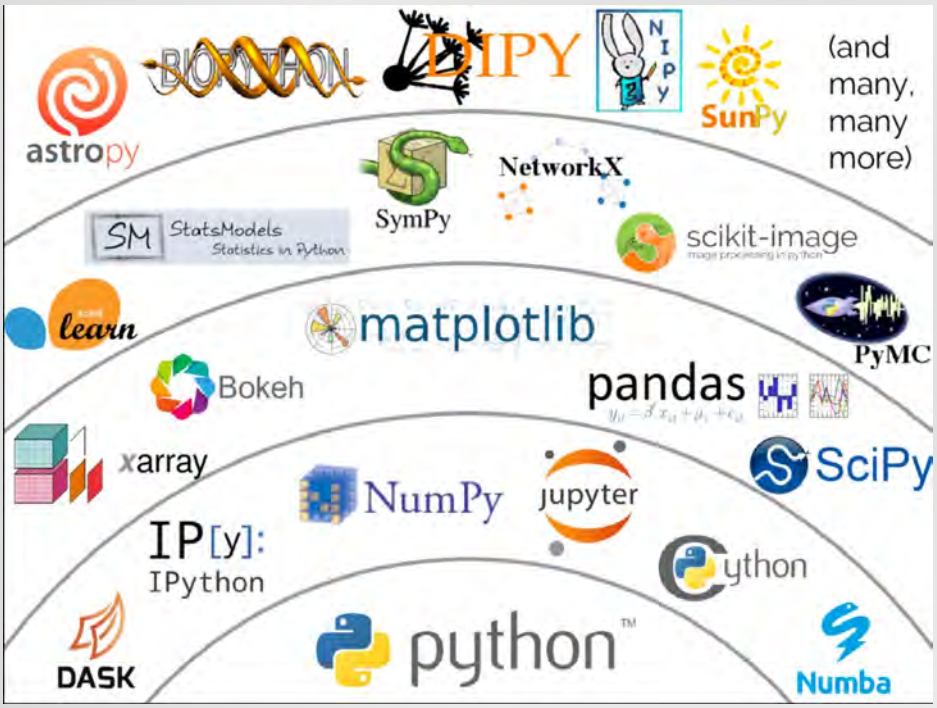


Diagram credit: Jake vanderPlas

Openness leads to sharing the load

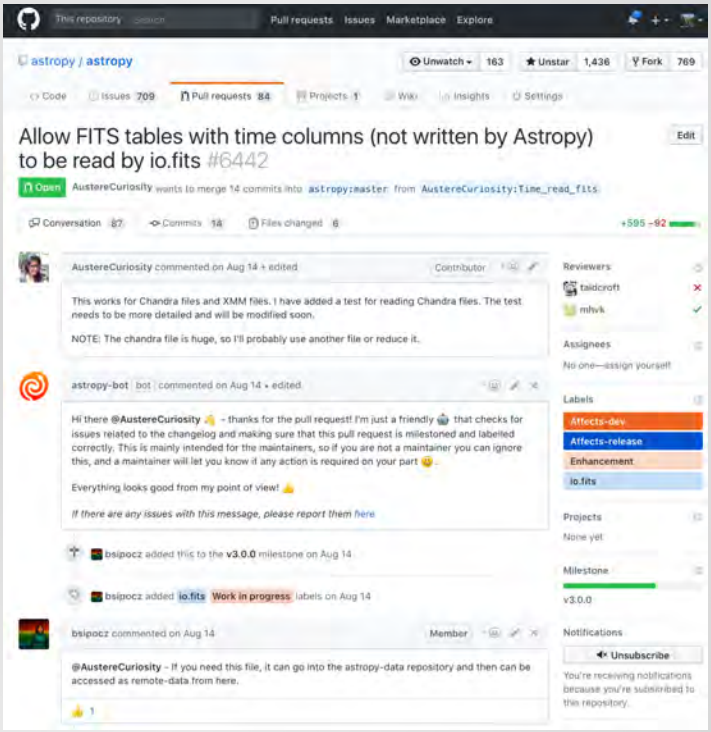
Core Shared Philosophy:

Open Development (≠ Open Source)

What is Open Development?

- A way to build software that emphasizes processes where every step is done publicly.
- Anyone, internal or external, can participate as a 1st-class citizen.
- This makes the software also *by* the astronomy community. (Like **you**!)

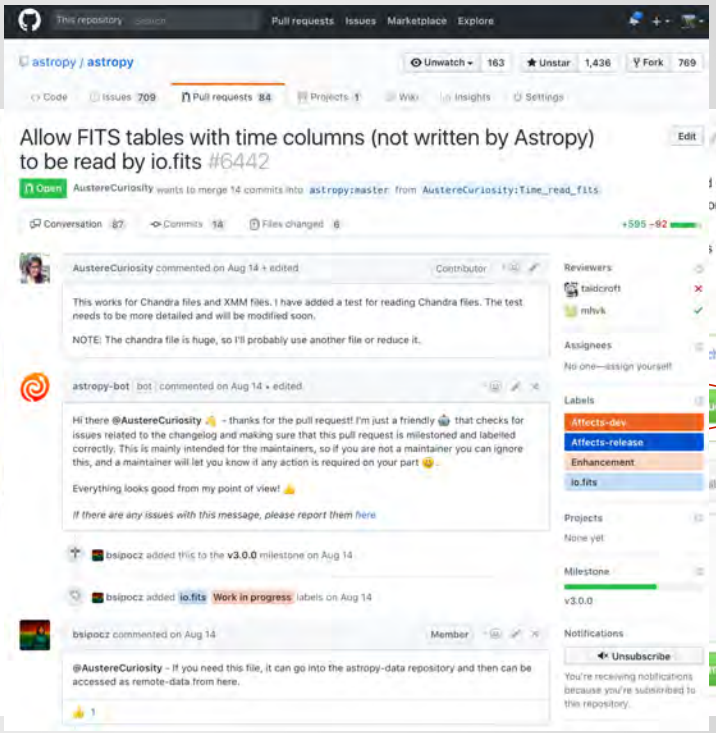
GitHub



What is Open Development?

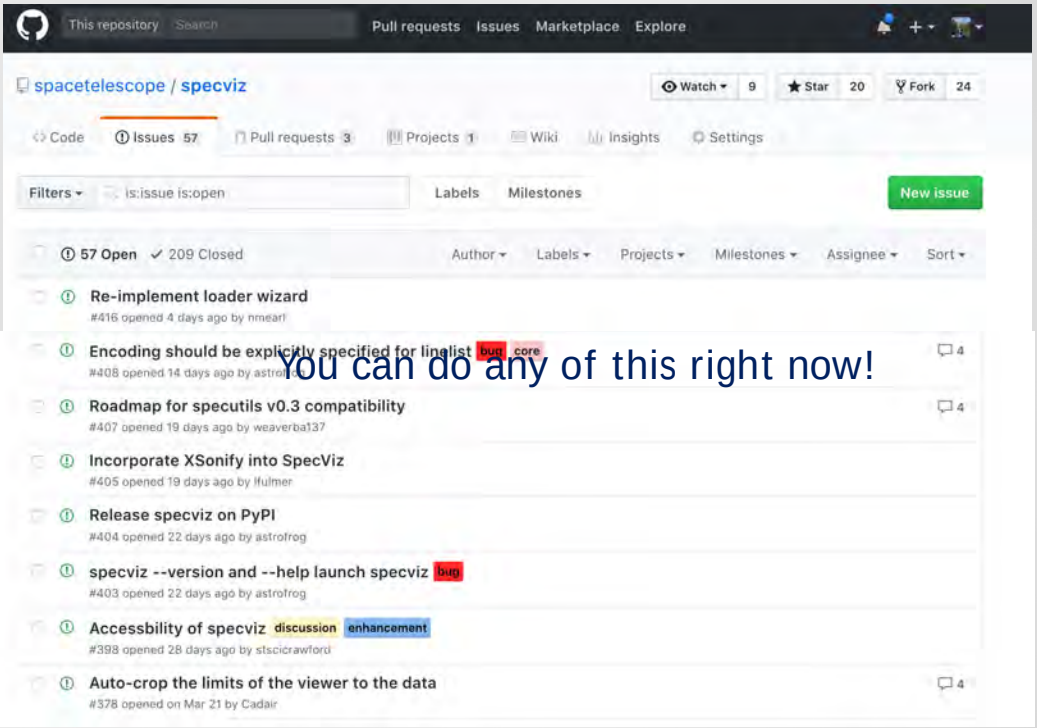
- Starting from a Pull Request… Anyone can review
- Commenters may be scientists
- Or engineers
- The original author then chooses how to update it
- The maintainers just hit “merge”. The user has become a contributor.

GitHub



Open Development is also Open *Planning*

- Design and planning of these tools also occurs in the open, in the same place the code lives.
- Contributions/ opinions are accepted from anyone.



You can do any of this right now!

Astropy

http://docs.astropy.org

- Data Analysis Tools
  - **Astropy**
  - Photutils (w/Astropy)
  - Specutils (w/Astropy)
  - WebbPSF
- Visualization Tools
  - Python Imexam (+ds9)
  - Ginga
  - SpecViz
  - (MOSViz)
  - (CubeViz)

- Units and "Quantities" (arrays with units that act the way you'd expect). Integrated with comprehensive astro-appropriate physical constants
- Date/time good to nanoseconds over a Hubble time
- Celestial coordinates and their transformations
- Table manipulation, including many arcane astro formats

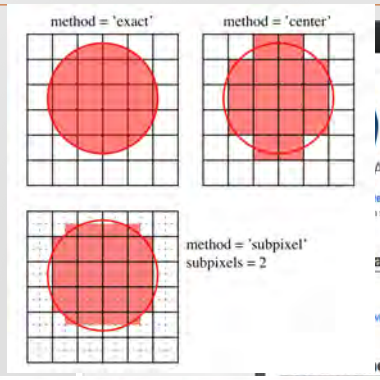
- *nddata*: Image analysis and interoperability data structures
- Astro-appropriate convolution
- WCS (pixel ↔ sky mapping)
- Extensible I/O: **FITS**, VOTable, hdf5, custom
- Astro-relevant data models and compatible fitting/optimization
- Common Astrostatistics tools
- Cosmology tools

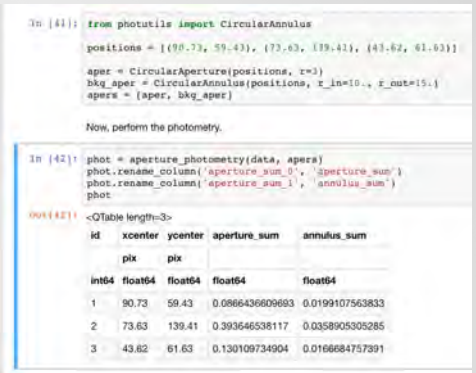
photutils

An Astropy Package for Photometry

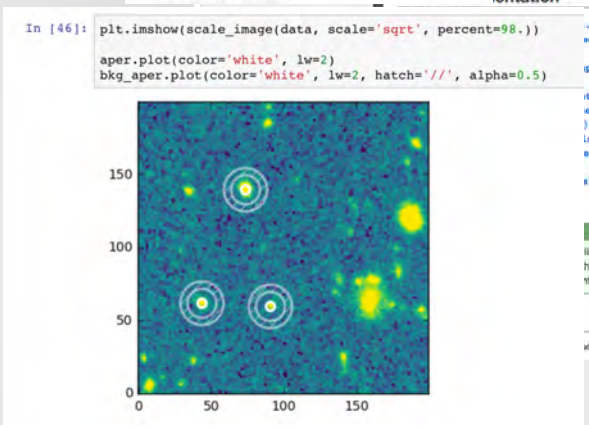
https://photutils.readthedocs.io

- Data Analysis Tools
  - Astropy
  - **Photutils** (w/Astropy)
  - Specutils (w/Astropy)
  - WebbPSF
- Visualization Tools
  - Python Imexam (+ds9)
  - Ginga
  - SpecViz
  - (MOSViz)
  - (CubeViz)





id	xcenter	ycenter	aperture_sum	annulus_sum
	pix	pix		
int64	float64	float64	float64	float64
1	90.73	59.43	0.0866436609693	0.0199107563833
2	73.63	139.41	0.393646538117	0.0358905305285
3	43.62	61.63	0.130109734904	0.0166684757391



A Demo of *photutils* on ACS data

photutils
An Astropy Package for Photometry

Aperture Photometry with `photutils`

Not shown here: Crowded-Field PSF Photometry (a la DAOPHOT/DOLPHOT) in the same framework.

What is aperture photometry?

The most common method to measure the flux from a celestial source is aperture photometry,. This kind of photometry measures the amount of flux within a region of defined shape and size (an aperture) surrounding a source. The ideal aperture would capture all of the flux emitted from the desired source, and none of the flux emitted from the surrounding sky or nearby sources. Especially when performing photometry on image data that includes a number of sources with varying size and shape, it is important to perform aperture corrections to account for imperfect apertures and better constrain photometric errors.

The `photutils` package provides tools for performing photometry with apertures of various shapes.

What does this tutorial include?

This tutorial covers how to perform aperture photometry with `photutils`, including the following methods:

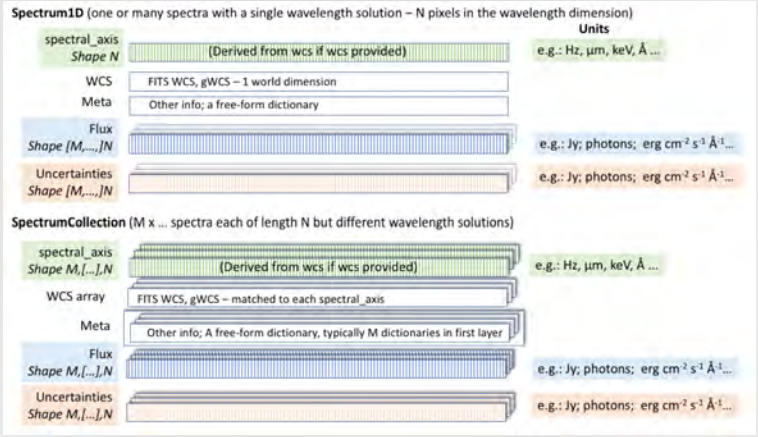
- Creating Apertures
  - Circular Apertures
  - Elliptical Apertures
  - Sky Apertures with WCS

Specutils

https://specutils.readthedocs.io

An Astropy-coordinated package with data structures and standard analysis functions for spectroscopy.

- Data Analysis Tools
  - Astropy
  - Photutils (w/Astropy)
  - **Specutils** (w/Astropy)
  - WebbPSF
- Visualization Tools
  - Python Imexam (+ds9)
  - Ginga
  - SpecViz
  - (MOSViz)
  - (CubeViz)



- Pythonic data structures of spectra^

- Analysis functions ->
  - Flux, Centroids, FWHM
  - Continuum fitting/ subtraction
  - Spectral arithmetic, respecting units
  - Line modeling

A Demo of *specutils* on STIS data

```
[ ]: import numpy as np

%matplotlib inline
from matplotlib import pyplot as plt

from IPython import display

from astropy import units as u
from astropy import modeling
from astropy.table import Table


from astropy.visualization import quantity_support
quantity_support()

import specutils
from specutils import analysis, fitting
```

Start by Loading a specisif PID's STIS dataset from MAST

```
from astroquery.mast import Observations

obses = Observations.query_criteria(proposal_id=9117)
```

```
sub_obs = obses[(obses['target_name']=='M82-A-POS1')&(obses['filters']=='G750M')]
sub_obs
```

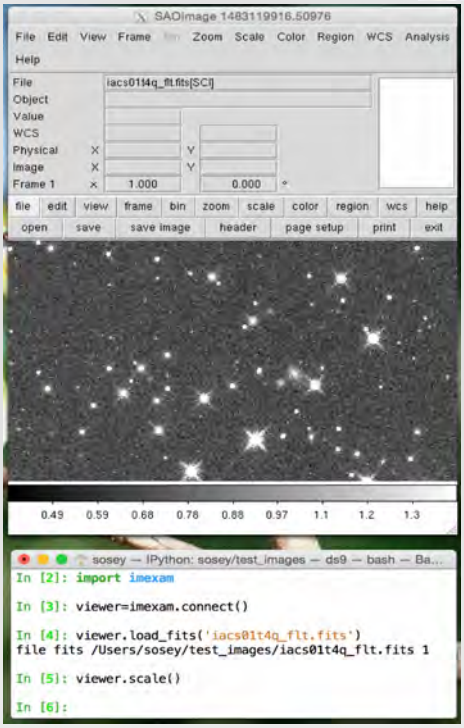
```
prods = Observations.get_product_list(sub_obs[0])
prods
```

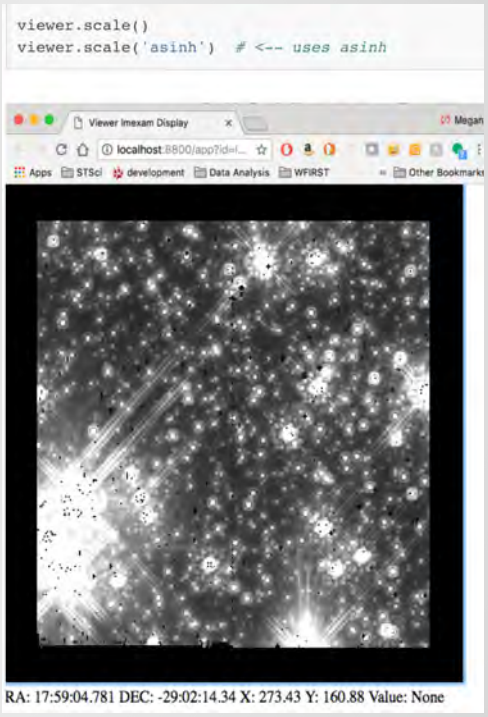
Imexam (in Python)

imexam

Lead: Megan Sosey

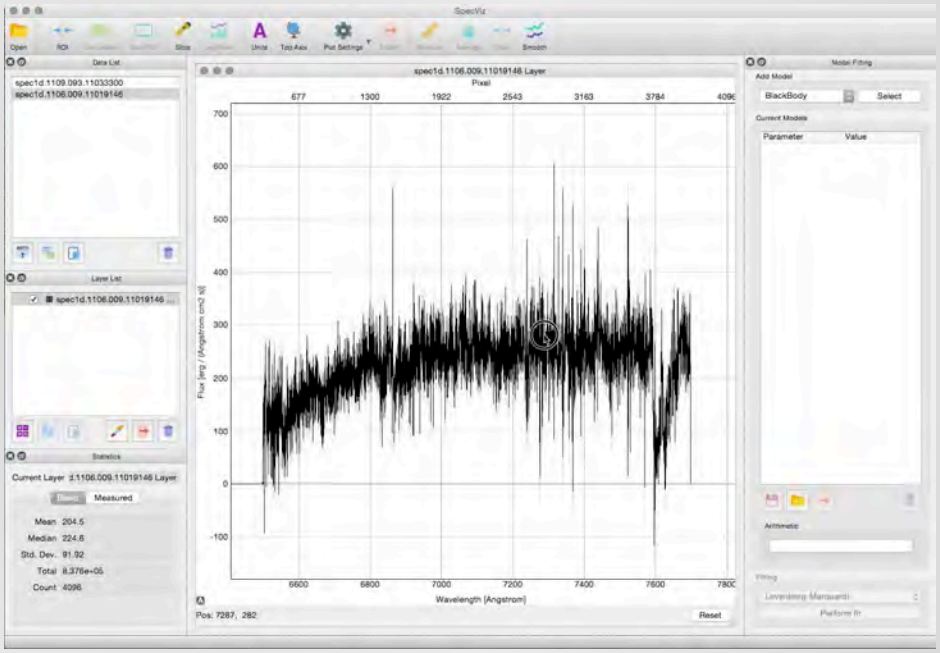
- Data Analysis Tools
  - Astropy
  - Photutils (w/Astropy)
  - **Specutils** (w/Astropy)
  - WebbPSF
- Visualization Tools
  - **Python Imexam (+ds9)**
  - Ginga
  - SpecViz
  - (MOSViz)
  - (CubeViz)





Specviz

- Data Analysis Tools
  - Astropy
  - Photutils (w/Astropy)
  - **Specutils** (w/Astropy)
  - WebbPSF
- Visualization Tools
  - Python Imexam (+ds9)
  - Ginga
  - **SpecViz**
  - (MOSViz)
  - (CubeViz)



But not everything comes for free

- Some of these tools require more directed effort to apply newly-developed JWST tools to HST.
  - Grism tools have been developed but tend to be problem-specific. JWST is more general but a lot of details are missing for HST
  - Sophisticated PSF modeling tools exist for for JWST (and WFIRST), and could replace tinytim, but need work to be backported and tested against HST's cameras.
  - Tools for *empirical* PSFs for astrometry and photometry are being developed for JWST from knowledge of HST, but need customization for HST.
  - (Tour favorite problem could go here!)
  - We may be able to do this if you tell us it's a priority, but we want to know what you want.


This is how STScI is helping build a new generation of data analysis tools that are not just for science with HST, but also *by* and *for*, its *community*.
But we need your input! Comments/questions?
Possibilities include:
Should we focus on grism tools/aXe replacement?
Or better PSFs/tinytim replacement?