PyFITS

2^{nd} *IAG Python Boot Camp*

Daniel Moser

Feb 15, 2017

FITS format

- FITS = *Flexible Image Transport System*
- From = IAU FITS working group (1981)
- Format = Multi-dimensional arrays:
  - 1D spectra, 2D images, 3D+ data cubes;
  - Data saved as Tables (rows and cols.);
  - Metadata stored as Header.
- Currently at version 3.0 (2008).
  Supported in many languages, as *C, IDL, Tcl* and Python.
- pyFITS = STSCI (Space Telescope Science Institute).

Three *magic* works in pyFITS:

- **HDU** = Header Data Unit;
  - Header = Metadata;
  - Data = Data itself;

For this tutorial: easier with UREKA.

Let's follow the tutorial file with IPYTHON (optionally FV and DS9):

```
>>> import pyfits as pf
>>> imname = 'data/spec.fits'
>>> hdulist = pf.open(imname)
>>> hdulist.info()
>>> hdulist.close()
```

HDUlist

- Each **HDU** entry contains a Table+Header structure.
- They are "separately" read.
- There are many ways of doing so (!):

```
>>> hdulist = pf.open(imname)
>>> prihdr = hdulist[0].header  # get first HDU header
>>> pridata = hdulist[0].data   # get first HDU data
>>> hdulist.close()

>>> spechdr = pf.getheader(imname, 0) # get first HDU header
>>> specdata = pf.getdata(imname, 0)  # get first HDU data

>>> specdata, spechdr = pf.getdata(imname, 0, header=True)
```

FITS Header

- FITS Header is the astronomical standard metadata table! Specifies all useful information associate with your data (obtaining conditions, reducing history, etc).
- The header is a table with three possible fields:
  - Keywords - must start with a text character (8 char. long).
  - Value - can be of multiple types (see below).
  - Comment - 80 character long string [Optional].
- The *type* options for value are:
  1. String; ascii characters enclosed in single quotes.
  2. Logical; letter "T" (TRUE) or letter "F" (FALSE).
  3. Integers; signed decimal numbers.
  4. Floating point numbers; similar to integers, with E or D denoting the exponent part.
  5. Complex numbers; specified as (real, imag).

FITS Header

- Manipulating header is easy! It is an Python dictionary!

```
>>> print( spechdr[:10] )
>>> print( spechdr.ascard[:10] )

>>> print( spechdr.keys() )

>>> print( 'INTTIME' in spechdr )
>>> print( 'EXPTIME' in spechdr )
>>> print( spechdr['EXPTIME'] )

>>> cdelt1 = spechdr['CDELT1']
>>> crval1 = spechdr['CRVAL1']
>>> naxis1 = spechdr['NAXIS1']
>>> lbdarr = np.arange(naxis1)*cdelt1+crval1
```

FITS Header

- You can add, edit and erase header entries!

```
>>> specHdr.add_history('IAG PBC example')
>>> print( specHdr[-5] )

>>> print( specHdr['OBSERVER'] )
>>> specHdr.update('OBSERVER', 'Edwin Hubble')
>>> print( specHdr['OBSERVER'] )

>>> print( 'OBSERVER' in specHdr )
>>> specHdr.pop('OBSERVER')
>>> print( 'OBSERVER' in specHdr )

>>> specHdr['OBSERVER'] = ('Edwin Hubble', /
    'That guy at the telescope...')
>>> print( 'OBSERVER' in specHdr )
```

FITS Header and HIERARCH Cards

- Each entry in the FITS header is a card.
- The insertion/sequence of entries can be selected (*COMMENT*, *HISTORY*).
- For keywords longer than 8 characters, there is a convention originated at ESO to facilitate such use:
  - It uses a special keyword HIERARCH with the actual long keyword following.
  - pyFITS supports the use of HIERARCH.

```
>>> spechdr['hierarch iagpyboot'] =  'awesome!'
>>> spechdr['iagpyboot']
'awesome!'
```

FITS 1D Data

- You can easily read and manipulate data!
- You don't need to care about the data structure: Python do it for you!

```
>>> print( specdata.shape )
>>> print( specdata[1000:1050] )
>>> print( np.min(specdata), np.max(specdata) )

>>> plt.plot(lbdarr, specdata, 'o-', ms=2)
>>> plt.plot(plt.xlim(), [1,1], '--', color='gray')
>>> plt.show()

>>> plt.clf()
>>> n, bins, patches = plt.hist(specdata)
>>> plt.show()
>>> plt.close()
```

Saving FITS

- Basically, there are two ways of saving FITS files: writeto and (update+) flush!

```
>>> pf.writeto('spec_modified.fits', specdata, header=spechdr)

>>> f = pf.open('spec_modified.fits', mode='update')
... # making changes in data and/or header
>>> f.flush()  # changes are written back to original.fits
>>> f.close()  # closing the file will also flush any changes and
...            # further writing
```

Hands on!

Exercises:

- FITS 2D+ data = creating a PNG image from a FITS images!!!
- Creating FITS cube from scratch.
- Accessing Tabular Data = VO Table!

Challenges:

1. Generate a JPG image of 'NEW_RGB.FITS'
2. Create a FITS Cube from scratch using PRIMARYHDU()

Other topics

- FITS formats details: e.g.,
  physical_value = BZERO + BSCALE * array_value;
- pyFITS put the opened FITS into memory.
  Caution when working with large files!
- fitscheck = verify and write FITS checksums;
- fitsdiff = analyze and display the differences between two FITS files.
- Also fitsinfo and fitsheader...

References

- Official FITS reference documentation:
  `http://fits.gsfc.nasa.gov`
- Wikipedia article:
  `https://en.wikipedia.org/wiki/FITS`
- Astropython.org PyFits tutorial:
  `https://gist.github.com/phn/3054997`
- Astropy:docs
  `http://astropy.readthedocs.org/en/latest/io/fits/`
- Intern. Virtual Observatory Alliance (IVOA):
  `http://ivoa.net/`