AstroAsciiData Table Handling in Python

Abstract

Tabulated character strings and numbers in text files are one of the most widespread data exchange format in astronomy and science in general.
Presented here is the version 1.0 of the Python module AstroAsciiData.
This module is a lightweight framework that provides a convenient access to table data and supports the common core of writing scripts to read, manipulate and write the data out.
Well formed tables are automatically parsed and transformed into objects, which allows a very convenient and intuitive data manipulation.
The new version 1.0 of the AstroAsciiData module uses metadata stored in the header of the ubiquitous SExtractor output catalogues and provides transformations to other data formats such as fits or html.

The AstroAsciiData Module

Handling tabulated data is a very common task that can be handled in many ways. Probably just about every scientist or computer professional in the scientific world has developed their own set of methods and code-snippets to deal with ASCII tables. Some may use the awk scripting language, others convert tables to FITS format and apply tools for accessing FITS tables, and some deploy IDL scripts. Common to all these approaches is their inflexibility and need for customisation - for example the preparation of format files for the transformation to FITS or the imposed rigid formatting of the ASCII tables themselves.

When several concurrent projects at the ST-ECF made intensive use of ASCII tables in a Python environment (such as aXe, aXe2web and the ingestion of a number of GOODS catalogues into the archive databases), it seemed prudent to develop a general purpose tool which could be used internally and also be distributed to a wider community.

Although reading and interpreting of text files is quite simple in Python, care must be taken to ensure that all comment-lines are escaped, the lines of data are split correctly, values are cast to the right data type and the null-value of the current table is interpreted correctly further down the line. AstroAsciiData aims to make this mundane task even more convenient, so that a few lines of Python code are enough to extract the desired data in the right format.

The AstroAsciiData module can be used interactively, within small Python scripts and also in large data reduction programs which use IRAF and MIDAS tasks via PyRAF and PyMidas, respectively. The boxes below and to the right give usage examples of interactive sessions with a simple table and a table with SExtractor headers.

The development of the AstroAsciiData module is taking place as part of the AstroLib project, an open source effort to develop general astronomical utilities akin to those available in the IDL ASTRON package.

We invite everyone to download the module from the AstroAsciiData homepage or directly from the subversion repository. Try it with your favourite tabulated data and mail questions, criticisms and improvement suggestions to AstroAsciiData@stecf.org

Simple usage example

example.txt:

```
#
# Some objects in the GOODS field
#
unknown 189.2207323  62.2357983 26.87 0.32
 galaxy 189.1408929  62.2376331 24.97 0.15
   star 189.1409453  62.1696844 25.30 0.12
 galaxy 188.9014716  62.2037839 25.95 0.20
```

Interactive session:

```
>>> import asciidata                                                    (1)
>>> example = asciidata.open('example.txt')                            (2)
>>> print str(example)                                                  (3)
#
# Some objects in the GOODS field
#
unknown  189.2207323  62.2357983  26.87  0.32
 galaxy  189.1408929  62.2376331  24.97  0.15
   star  189.1409453  62.1696844  25.30  0.12
 galaxy  188.9014716  62.2037839  25.95  0.20
>>> sum1=0.0                                                            (4)
>>> sum2=0.0
>>> for index in range(example.nrows):
...     sum1 += example[1][index]
...     sum2 += example[2][index]
...
>>> ave1 = sum1/example.nrows
>>> ave2 = sum2/example.nrows
>>> print ave1, ave
2189.101010525 62.211724925
>>> for index in range(example.nrows):                                  (5)
...     example['diff1'][index] = example[1][index] - ave1
...     example['diff2'][index] = example[2][index] - ave2
...
>>> print str(example)
#
# Some objects in the GOODS field
#
unknown  189.2207323  62.2357983  26.87  0.32  1.197218e-01  2.407338e-02
 galaxy  189.1408929  62.2376331  24.97  0.15  3.988237e-02  2.590817e-02
   star  189.1409453  62.1696844  25.30  0.12  3.993477e-02 -4.204052e-02
 galaxy  188.9014716  62.2037839  25.95  0.20 -1.995389e-01 -7.941025e-03
>>> example.writeto('example_mod.txt')                                  (6)
```

1. Within Python, the AstroAsciiData module is imported
2. Load the table
3. The table is inspected inside Python
4. As a first application, the user computes the average from the numbers in the second and third row
5. Then the user computes and saves the differences between the average and the individual values
6. The modified ASCII table is written to the file 'example_mod.txt'

Features of version 1.0

- Imports all reasonably well-formed Ascii tables
- Column-first access
- Easy creation and manipulation of tables, columns, rows and attached comments
- Retains formatting of data values
- Support for SExtractor style headers
- Column sorting
- Interchangeable comment character, column delimiter and null value
- Exports to
  - Ascii
  - Numarray
  - FITS table
  - HTML table format
  - LaTeX table format

Features for future versions

- Even more Pythonic
- Support for more external formats
- ?? (We are open for suggestions)

Example with SExtractor header

AstroAsciiData tries to make use of header meta-data if possible.

As one of the most common astronomical ASCII tables the Source Extractor output format is automatically recognised and interpreted so that column names can be used immediately.

```
>>> import asciidata
>>> tab = asciidata.open('h_goods_sb_r1.1z.cat')                        (1)
>>> tab['ID_IAU'][0]                                                    (2)
'J033155.16-274540.0'
>>> len(tab)
104
>>> newtab = tab[:28]                                                   (3)
>>> len(newtab)
28
>>> print newtab.info()                                                 (4)
File:       h_goods_sb_r1.1z.cat
Ncols:      28
Nrows:      38
Delimiter:  None
Null value: ['Null', 'NULL', 'None', '*']
comment_char:    #
Column name:         ID_IAU
Column type:         <type 'str'>
Column format:       ['% 19s', '%19s']
Column null value :  ['Null']
Column comment :  IAU style ID, remember to add a acronym at some point
Column name:         ALPHA_J2000
Column type:         <type 'float'>
Column format:       ['% 10.7f', '%11s']
Column null value :  ['Null']
Column unit :  deg
...
Column name:         FLUX_RADIUS
Column type:         <type 'float'>
Column format:       ['% 6.3f', '%7s']
Column null value :  ['Null']
Column name:         FLUX_RADIUS1
Column type:         <type 'float'>
Column format:       ['% 6.3f', '%7s']
Column null value :  ['Null']
Column name:         FLUX_RADIUS2
Column type:         <type 'float'>
Column format:       ['% 6.3f', '%7s']
Column null value :  ['Null']
Column name:         FWHM_IMAGE
Column type:         <type 'float'>
Column format:       ['% 4.2f', '%5s']
Column null value :  ['Null']
Column name:         CLASS_STAR
Column type:         <type 'float'>
Column format:       ['% 4.2f', '%5s']
Column null value :  ['Null']

>>> newtab['ID_IAU'].set_colcomment('maybe we need an acronym at some point') (5)
>>> newtab['ID_IAU'].get_colcomment()
'maybe we need an acronym at some point'

>>> elarray = newtab['ELONGATION'].tonumarray()                         (6)
>>> elarray
array([ 1.034,  1.079,  1.092,  1.114,  1.129,  1.149,  1.197,  1.198,
        1.2  ,  1.204,  1.218,  1.241,  1.243,  1.246,  1.271,  1.283,
        1.312,  1.33 ,  1.335,  1.364,  1.374,  1.374,  1.377,  1.386,
        1.406,  1.411,  1.535,  1.545,  1.587,  1.599,  1.61 ,  1.649,
        1.671,  1.725,  1.889,  2.193,  2.532,  2.845])

>>> newtab.sort('ELLIPTICITY')
>>> newtab.newdelimiter('|')
>>> newtab.writeto('h_goods_sb_r1.1z.elip-sorted.cat')                  (7)
```

1. Import AstroAsciiData module, read a catalogue
2. Columns are immediately accessible through their names
3. Cut out first 28 out of 104 columns
4. Check table information
5. Correct spelling mistake in header comment
6. Cast column values into an array
7. Sort table by column values, change delimiter character, write to file.

Vector output in the SExtractor table get additional running numbers assigned to the column name for each column after the first, as with FLUX RADIUS to the left.

Links

AstroAsciiData Homepage	http://www.stecf.org/software/PYTHONtools/astroasciidata
Astrolib @ Scipy.org	http://www.scipy.org/AstroLib
Subversion repository	http://astropy.scipy.org/svn/astrolib/trunk/asciidata/

Jonas Haase

Martin Kümmel

Space Telescope European Coordinating Facility
c/o European Southern Observatory
Karl-Schwarzschild-Strasse 2
D-85748 Garching bei München
Germany
AstroAsciiData@stecf.org