Using_astropy

June 1, 2016

```
In [91]: # Just to know last time this was run:
         import time
         print time.ctime()
```

```
Wed Jun  1 17:10:56 2016
```

1 G The astropy package

The Astropy Project is a community effort to develop a single core package for Astronomy in Python and foster interoperability between Python astronomy packages. More informations here: http://www.astropy.org/

!!! WARNING !!!

To install atpy, one must use the –no-deps option when using pip (otherwise updates of numpy may be performed):

pip install -U –no-deps astropy

```
In [92]: %matplotlib inline
         import numpy as np
         import matplotlib.pyplot as plt
```

1.0.1 Constants and Units

```
In [93]: from astropy import constants as const
         from astropy import units as u
         help(const)
```

```
Help on package astropy.constants in astropy:

NAME
    astropy.constants

FILE
    /Users/christophemorisset/anaconda/lib/python2.7/site-packages/astropy/constant

DESCRIPTION
    Contains astronomical and physical constants for use in Astropy or other
```

places.

A typical use case might be::

```
>>> from astropy.constants import c, m_e
>>> # ... define the mass of something you want the rest energy of as m ...
>>> m = m_e
>>> E = m * c**2
>>> E.to('MeV')  # doctest: +FLOAT_CMP
<Quantity 0.510998927603161 MeV>
```

The following constants are available:

Name	Value	Unit	Description
G	6.67384e-11	m3 / (kg s2)	Gravitational constant
L_sun	3.846e+26	W	Solar luminosity
M_earth	5.9742e+24	kg	Earth mass
M_jup	1.8987e+27	kg	Jupiter mass
M_sun	1.9891e+30	kg	Solar mass
N_A	6.02214129e+23	1 / (mol)	Avogadro's number
R	8.3144621	J / (K mol)	Gas constant
R_earth	6378136	m	Earth equatorial radius
R_jup	71492000	m	Jupiter equatorial radius
R_sun	695508000	m	Solar radius
Ryd	10973731.6	1 / (m)	Rydberg constant
a0	5.29177211e-11	m	Bohr radius
alpha	0.00729735257		Fine-structure constant
atmosphere	101325	Pa	Atmosphere
au	1.49597871e+11	m	Astronomical Unit
b_wien	0.0028977721	m K	Wien wavelength displacement law co
c	299792458	m / (s)	Speed of light in vacuum
e	1.60217657e-19	C	Electron charge
eps0	8.85418782e-12	F/m	Electric constant
g0	9.80665	m / s2	Standard acceleration of gravity
h	6.62606957e-34	J s	Planck constant
hbar	1.05457173e-34	J s	Reduced Planck constant
k_B	1.3806488e-23	J / (K)	Boltzmann constant
kpc	3.08567758e+19	m	Kiloparsec
m_e	9.10938291e-31	kg	Electron mass
m_n	1.67492735e-27	kg	Neutron mass
m_p	1.67262178e-27	kg	Proton mass
mu0	1.25663706e-06	N/A2	Magnetic constant
muB	9.27400968e-24	J/T	Bohr magneton
pc	3.08567758e+16	m	Parsec
sigma_T	6.65245873e-29	m2	Thomson scattering cross-section
sigma_sb	5.670373e-08	W / (K4 m2)	Stefan-Boltzmann constant

```
        u      1.66053892e-27        kg        Atomic mass
    ========== ============== ================ ==========================

PACKAGE CONTENTS
    cgs
    constant
    setup_package
    si
    tests (package)

DATA
    G = <Constant name=u'Gravitational constant' value=6...-15 unit='m3 / ...
    L_sun = <Constant name=u'Solar luminosity' value=3.846e+...ence=u"Alle...
    M_earth = <Constant name=u'Earth mass' value=5.9742e+24 un...ence=u"Al...
    M_jup = <Constant name=u'Jupiter mass' value=1.8987e+27 ...ence=u"Alle...
    M_sun = <Constant name=u'Solar mass' value=1.9891e+30 un...ence=u"Alle...
    N_A = <Constant name=u"Avogadro's number" value=6.0221...=2.7e+16 unit...
    R = <Constant name=u'Gas constant' value=8.3144621 u...e-06 unit='J / ...
    R_earth = <Constant name=u'Earth equatorial radius' value=...ence=u"Al...
    R_jup = <Constant name=u'Jupiter equatorial radius' valu...ence=u"Alle...
    R_sun = <Constant name=u'Solar radius' value=695508000.0...ence=u"Alle...
    Ryd = <Constant name=u'Rydberg constant' value=1097373...ty=5.5e-05 un...
    a0 = <Constant name=u'Bohr radius' value=5.2917721092...tainty=1.7e-20...
    absolute_import = _Feature((2, 5, 0, 'alpha', 1), (3, 0, 0, 'alpha', 0...
    alpha = <Constant name=u'Fine-structure constant' value=...rtainty=2.4...
    atmosphere = <Constant name=u'Atmosphere' value=101325 uncertainty=0.0...
    au = <Constant name=u'Astronomical Unit' value=1.4959...0.0 unit='m' r...
    b_wien = <Constant name=u'Wien wavelength displacement la...inty=2.6e-...
    c = <Constant name=u'Speed of light in vacuum' value...tainty=0.0 unit...
    division = _Feature((2, 2, 0, 'alpha', 2), (3, 0, 0, 'alpha', 0), 8192...
    e = <Constant name=u'Electron charge' value=1.602176...tainty=3.5e-27 ...
    eps0 = <Constant name=u'Electric constant' value=8.8541...tainty=0.0 u...
    g0 = <Constant name=u'Standard acceleration of gravit...ainty=0.0 unit...
    h = <Constant name=u'Planck constant' value=6.626069...inty=2.9e-41 un...
    hbar = <Constant name=u'Reduced Planck constant' value=...49334966e-42...
    k_B = <Constant name=u'Boltzmann constant' value=1.380...ty=1.3e-29 un...
    kpc = <Constant name=u'Kiloparsec' value=3.08567758147...tainty=0.0 un...
    m_e = <Constant name=u'Electron mass' value=9.10938291...rtainty=4e-38...
    m_n = <Constant name=u'Neutron mass' value=1.674927351...ainty=7.4e-35...
    m_p = <Constant name=u'Proton mass' value=1.672621777e...ainty=7.4e-35...
    mu0 = <Constant name=u'Magnetic constant' value=1.2566...ainty=0.0 uni...
    muB = <Constant name=u'Bohr magneton' value=9.27400968...inty=2e-31 un...
    pc = <Constant name=u'Parsec' value=3.08567758147e+16...tainty=0.0 uni...
    print_function = _Feature((2, 6, 0, 'alpha', 2), (3, 0, 0, 'alpha', 0)...
    sigma_T = <Constant name=u'Thomson scattering cross-sectio...ainty=1.3...
    sigma_sb = <Constant name=u'Stefan-Boltzmann constant' valu...e-13 uni...
    u = <Constant name=u'Atomic mass' value=1.660538921e...ainty=7.3e-35 u...
    unicode_literals = _Feature((2, 6, 0, 'alpha', 2), (3, 0, 0, 'alpha', ...
```

```
In [94]: # Pretty printing
         print const.c
```

```
  Name   = Speed of light in vacuum
  Value  = 299792458.0
  Uncertainty  = 0.0
  Unit  = m / s
  Reference = CODATA 2010
```

```
In [95]: # .to change the unit
         print const.c.to('pc/yr')
```

```
0.306601393788 pc / yr
```

```
In [96]: # basic operations are managed
         const.c * 2
```

Out[96]:

$5.9958492 \times 10^{8} \ \frac{\mathrm{m}}{\mathrm{s}}$

```
In [97]: np.sqrt(const.c)
```

Out[97]:

$17314.516 \ \frac{\mathrm{m}^{1/2}}{\mathrm{s}^{1/2}}$

```
In [98]: print np.sqrt(const.c)
```

```
17314.5158177 m(1/2) / s(1/2)
```

```
In [99]: # Following the units
         F = (const.G * 3. * const.M_sun * 100 * u.kg) / (2.2 * u.au) ** 2
         print F
```

```
8.22826558512e+21 kg m3 / (AU2 s2)
```

```
In [100]: F
```

Out[100]:

$8.2282656 \times 10^{21} \ \frac{\mathrm{m}^{3}\,\mathrm{kg}}{\mathrm{AU}^{2}\,\mathrm{s}^{2}}$

```
In [101]: # Convert in more classical unit
          print F.to(u.N)
```

```
0.367669392028 N
```

```
In [102]: q = 42.0 * u.meter

In [103]: q**2
```

Out[103]:

$1764\ \mathrm{m}^2$

```
In [104]: # Extract only the value
          (q**2).value

Out[104]: 1764.0

In [105]: # Resolving redondant units
          t = 3.0 * u.kilometer / (130.51 * u.meter / u.second)
          print t
          print t.decompose()

0.0229867443108 km s / m
22.9867443108 s

In [106]: x = 1.0 * u.parsec
          print x.to(u.km)

3.08567758147e+13 km

In [107]: lam = 5007 * u.angstrom

In [108]: print lam.to(u.nm)
          print lam.to(u.micron)

500.7 nm
0.5007 micron

In [109]: # Some transformations needs extra information, available from u.special
          print lam.to(u.eV, equivalencies=u.spectral())

2.47621715438 eV
```

More in http://docs.astropy.org/en/stable/units/index.html

1.0.2 Data Table

http://docs.astropy.org/en/stable/table/index.html

```
In [110]: from astropy.table import Table
```

```
In [111]: # create a table with non homogeneous types
          a = [1, 4, 5]
          b = [2.0, 5.0, 8.2]
          c = ['x', 'y', 'z']
          t = Table([a, b, c], names=('a', 'b', 'c'), meta={'name': 'first table'})
          print t
```

```
 a   b   c
--- --- ---
  1 2.0   x
  4 5.0   y
  5 8.2   z
```

```
In [112]: # Pretty output
          t
```

```
Out[112]: <Table length=3>
            a      b     c
          int64 float64 str1
          ----- ------- ----
              1     2.0    x
              4     5.0    y
              5     8.2    z
```

```
In [113]: # One can change the output format
          t['b'].format = '7.3f'
          t['a'].format = '{:.4f}'
          # and add units
          t['b'].unit = 's'
          t
```

```
Out[113]: <Table length=3>
            a       b     c
                    s
          int64  float64 str1
          ------ ------- ----
          1.0000   2.000    x
          4.0000   5.000    y
          5.0000   8.200    z
```

```
In [114]: t.show_in_browser(jsviewer=True)
```

```
In [115]: # access the column names
          t.colnames
```

```
Out[115]: ['a', 'b', 'c']

In [116]: # length of the table (number of rows)
          len(t)

Out[116]: 3

In [117]: # Acces one element
          t['a'][1]

Out[117]: 4

In [118]: # Modefy one element
          t['a'][1] = 10
          t

Out[118]: <Table length=3>
             a       b     c
                     s
           int64  float64 str1
          ------- ------- ----
           1.0000   2.000    x
          10.0000   5.000    y
           5.0000   8.200    z

In [119]: # easy add column:
          t['d'] = [1, 2, 3]

In [120]: t

Out[120]: <Table length=3>
             a       b     c     d
                     s
           int64  float64 str1 int64
          ------- ------- ---- -----
           1.0000   2.000    x     1
          10.0000   5.000    y     2
           5.0000   8.200    z     3

In [121]: t.rename_column('a', 'A')
          t

Out[121]: <Table length=3>
             A       b     c     d
                     s
           int64  float64 str1 int64
          ------- ------- ---- -----
           1.0000   2.000    x     1
          10.0000   5.000    y     2
           5.0000   8.200    z     3
```

```
In [122]: t.add_row([-8, -9, 'r', 10])
          t
```

```
Out[122]: <Table length=4>
             A        b      c     d
                      s
           int64   float64 str1 int64
          ------- ------- ---- -----
           1.0000   2.000    x     1
          10.0000   5.000    y     2
           5.0000   8.200    z     3
          -8.0000  -9.000    r    10
```

```
In [123]: t.add_row([-9, 40, 'q', 10])
          t
```

```
Out[123]: <Table length=5>
             A        b      c     d
                      s
           int64   float64 str1 int64
          ------- ------- ---- -----
           1.0000   2.000    x     1
          10.0000   5.000    y     2
           5.0000   8.200    z     3
          -8.0000  -9.000    r    10
          -9.0000  40.000    q    10
```

```
In [124]: # Masked values
          t = Table([a, b, c], names=('a', 'b', 'c'), masked=True)
          t['a'].mask = [True, True, False] # True is for the masked values!!
          t
```

```
Out[124]: <Table masked=True length=3>
            a      b     c
          int64 float64 str1
          ----- ------- ----
             --     2.0    x
             --     5.0    y
              5     8.2    z
```

```
In [125]: # Creat a table from a table
          t2 = Table([t['a']**2, t['b']**2])
          t2
```

```
Out[125]: <Table masked=True length=3>
            a      b
          int64 float64
          ----- -------
             --     4.0
             --    25.0
             25   67.24
```

In [126]:
```python
# Managing columns
from astropy.table import Column
```

In [127]:
```python
# Create a table combining different formats
a = (1, 4)
b = np.array([[2, 3], [5, 6]])  # vector column
c = Column(['x', 'y'], name='axis')
arr = (a, b, c)
t3 = Table(arr)  # Data column named "c" has a name "axis" in that table
t3
```

Out[127]:
```
<Table length=2>
 col0 col1 [2] axis
int64  int64   str1
----- -------- ----
    1   2 .. 3    x
    4   5 .. 6    y
```

In [128]:
```python
# table from a dictionnary
rr = {'a': [1, 4],
      'b': [2.0, 5.0],
      'c': ['x', 'y']}
t4 = Table(rr, names=('a', 'c', 'b'))
t4
```

Out[128]:
```
<Table length=2>
  a    c      b
int64 str1 float64
----- ---- -------
    1    x     2.0
    4    y     5.0
```

In [129]:
```python
# Create table row by row
t5 = Table(rows=[{'a': 5, 'b': 10}, {'a': 15, 'b': 30}])
t5
```

Out[129]:
```
<Table length=2>
  a     b
int64 int64
----- -----
    5    10
   15    30
```

In [130]:
```python
# Numpy structured array
arr = np.array([(1, 2.0, 'x'),
                (4, 5.0, 'y')],
               dtype=[('a', 'i8'), ('b', 'f8'), ('c', 'S2')])
t6 = Table(arr)
t6
```

```
Out[130]: <Table length=2>
            a      b      c
          int64 float64 str2
          ----- ------- ----
              1     2.0    x
              4     5.0    y
```

Python arrays versus `numpy` arrays as input

There is a slightly subtle issue that is important to understand in the way that Table objects are created. Any data input that looks like a Python list (including a tuple) is considered to be a list of columns. In contrast an homogeneous numpy array input is interpreted as a list of rows:

```
In [131]: arr = np.array([(1, 2.0, 'x'),
                          (4, 5.0, 'y')],
                         dtype=[('a', 'i8'), ('b', 'f8'), ('c', 'S2')])
          t6 = Table(arr, copy=False) # pinting to the original data
          arr['a'][0] = 99
          print arr
          print t6

[(99, 2.0, 'x') (4, 5.0, 'y')]
 a   b   c
--- --- ---
 99 2.0   x
  4 5.0   y


In [132]: t6.columns

Out[132]: TableColumns([('a', <Column name='a' dtype='int64' length=2>
                         99
                          4), ('b', <Column name='b' dtype='float64' length=2>
                         2.0
                         5.0), ('c', <Column name='c' dtype='str2' length=2>
                         x
                         y)])

In [133]: t6.colnames

Out[133]: ['a', 'b', 'c']

In [134]: # One can obtain a numpy structured array from a Table
          np.array(t6)

Out[134]: array([(99, 2.0, 'x'), (4, 5.0, 'y')],
                dtype=[('a', '<i8'), ('b', '<f8'), ('c', 'S2')])

In [135]: arr = np.arange(3000).reshape(100, 30)  # 100 rows x 30 columns array
          t = Table(arr)
          print t
```

```
col0 col1 col2 col3 col4 col5 col6 ... col23 col24 col25 col26 col27 col28 col29
---- ---- ---- ---- ---- ---- ---- ... ----- ----- ----- ----- ----- ----- -----
   0    1    2    3    4    5    6 ...    23    24    25    26    27    28    29
  30   31   32   33   34   35   36 ...    53    54    55    56    57    58    59
  60   61   62   63   64   65   66 ...    83    84    85    86    87    88    89
  90   91   92   93   94   95   96 ...   113   114   115   116   117   118   119
 120  121  122  123  124  125  126 ...   143   144   145   146   147   148   149
 150  151  152  153  154  155  156 ...   173   174   175   176   177   178   179
 180  181  182  183  184  185  186 ...   203   204   205   206   207   208   209
 210  211  212  213  214  215  216 ...   233   234   235   236   237   238   239
 240  241  242  243  244  245  246 ...   263   264   265   266   267   268   269
 270  271  272  273  274  275  276 ...   293   294   295   296   297   298   299
 ...  ...  ...  ...  ...  ...  ...  ...   ...   ...   ...   ...   ...   ...   ...
2670 2671 2672 2673 2674 2675 2676 ...  2693  2694  2695  2696  2697  2698  2699
2700 2701 2702 2703 2704 2705 2706 ...  2723  2724  2725  2726  2727  2728  2729
2730 2731 2732 2733 2734 2735 2736 ...  2753  2754  2755  2756  2757  2758  2759
2760 2761 2762 2763 2764 2765 2766 ...  2783  2784  2785  2786  2787  2788  2789
2790 2791 2792 2793 2794 2795 2796 ...  2813  2814  2815  2816  2817  2818  2819
2820 2821 2822 2823 2824 2825 2826 ...  2843  2844  2845  2846  2847  2848  2849
2850 2851 2852 2853 2854 2855 2856 ...  2873  2874  2875  2876  2877  2878  2879
2880 2881 2882 2883 2884 2885 2886 ...  2903  2904  2905  2906  2907  2908  2909
2910 2911 2912 2913 2914 2915 2916 ...  2933  2934  2935  2936  2937  2938  2939
2940 2941 2942 2943 2944 2945 2946 ...  2963  2964  2965  2966  2967  2968  2969
2970 2971 2972 2973 2974 2975 2976 ...  2993  2994  2995  2996  2997  2998  2999
Length = 100 rows
```

```
In [136]: t.show_in_browser(jsviewer=True)
```

```
In [137]: # create a simple table to play with
          arr = np.arange(15).reshape(5, 3)
          t = Table(arr, names=('a', 'b', 'c'), meta={'keywords': {'key1': 'val1'}}
          t
```

```
Out[137]: <Table length=5>
            a     b     c
          int64 int64 int64
          ----- ----- -----
              0     1     2
              3     4     5
              6     7     8
              9    10    11
             12    13    14
```

```
In [138]: t['a'] = [1, -2, 3, -4, 5]  # Set all
          t
```

```
Out[138]: <Table length=5>
            a     b     c
```

```
int64 int64 int64
----- ----- -----
    1     1     2
   -2     4     5
    3     7     8
   -4    10    11
    5    13    14
```

In [139]:
```
t['a'][2] = 30 # set one
t
```

Out[139]:
```
<Table length=5>
  a     b     c
int64 int64 int64
----- ----- -----
    1     1     2
   -2     4     5
   30     7     8
   -4    10    11
    5    13    14
```

In [140]:
```
# set one row
t[1] = (8, 9, 10)
t
```

Out[140]:
```
<Table length=5>
  a     b     c
int64 int64 int64
----- ----- -----
    1     1     2
    8     9    10
   30     7     8
   -4    10    11
    5    13    14
```

In [141]:
```
# Set a whole column
t['a'] = 99
t
```

Out[141]:
```
<Table length=5>
  a     b     c
int64 int64 int64
----- ----- -----
   99     1     2
   99     9    10
   99     7     8
   99    10    11
   99    13    14
```

```
In [142]: # Add a column
          t.add_column(Column(np.array([1,2,3,4,5]), name='d'))
          t
```

```
Out[142]: <Table length=5>
            a     b     c     d
          int64 int64 int64 int64
          ----- ----- ----- -----
             99     1     2     1
             99     9    10     2
             99     7     8     3
             99    10    11     4
             99    13    14     5
```

```
In [143]: # remove a column
          t.remove_column('b')
          t
```

```
Out[143]: <Table length=5>
            a     c     d
          int64 int64 int64
          ----- ----- -----
             99     2     1
             99    10     2
             99     8     3
             99    11     4
             99    14     5
```

```
In [144]: # add a row
          t.add_row([-8, -9, 10])
          t
```

```
Out[144]: <Table length=6>
            a     c     d
          int64 int64 int64
          ----- ----- -----
             99     2     1
             99    10     2
             99     8     3
             99    11     4
             99    14     5
             -8    -9    10
```

```
In [145]: # Remove some rows
          t.remove_rows([1, 2])
          t
```

```
Out[145]: <Table length=4>
            a     c     d
```

```
int64 int64 int64
----- ----- -----
   99     2     1
   99    11     4
   99    14     5
   -8    -9    10
```

In [146]:
```
# sort the Table using one column
t.sort('c')
t
```

Out[146]:
```
<Table length=4>
  a     c     d
int64 int64 int64
----- ----- -----
   -8    -9    10
   99     2     1
   99    11     4
   99    14     5
```

In [147]:
```
%%writefile tab1.dat
name    obs_date    mag_b  mag_v
M31     2012-01-02  17.0   17.5
M31     2012-01-02  17.1   17.4
M101    2012-01-02  15.1   13.5
M82     2012-02-14  16.2   14.5
M31     2012-02-14  16.9   17.3
M82     2012-02-14  15.2   15.5
M101    2012-02-14  15.0   13.6
M82     2012-03-26  15.7   16.5
M101    2012-03-26  15.1   13.5
M101    2012-03-26  14.8   14.3
```

```
Overwriting tab1.dat
```

In [148]:
```
# directly read a Table from an ascii file
obs = Table.read('tab1.dat', format='ascii')
```

In [149]:
```
print obs
```

```
name  obs_date  mag_b mag_v
---- ---------- ----- -----
 M31 2012-01-02  17.0  17.5
 M31 2012-01-02  17.1  17.4
M101 2012-01-02  15.1  13.5
 M82 2012-02-14  16.2  14.5
 M31 2012-02-14  16.9  17.3
 M82 2012-02-14  15.2  15.5
```

```
M101 2012-02-14  15.0  13.6
 M82 2012-03-26  15.7  16.5
M101 2012-03-26  15.1  13.5
M101 2012-03-26  14.8  14.3
```

```
In [150]: # Group data
          obs_by_name = obs.group_by('name')
          obs_by_name

Out[150]: <Table length=10>
          name  obs_date   mag_b   mag_v
          str4   str10    float64 float64
          ---- ---------- ------- -------
          M101 2012-01-02    15.1    13.5
          M101 2012-02-14    15.0    13.6
          M101 2012-03-26    15.1    13.5
          M101 2012-03-26    14.8    14.3
           M31 2012-01-02    17.0    17.5
           M31 2012-01-02    17.1    17.4
           M31 2012-02-14    16.9    17.3
           M82 2012-02-14    16.2    14.5
           M82 2012-02-14    15.2    15.5
           M82 2012-03-26    15.7    16.5

In [151]: print obs_by_name.groups.keys
```

```
name
----
M101
 M31
 M82
```

```
In [152]: # Using 2 keys to group
          print obs.group_by(['name', 'obs_date']).groups.keys
```

```
name  obs_date
---- ----------
M101 2012-01-02
M101 2012-02-14
M101 2012-03-26
 M31 2012-01-02
 M31 2012-02-14
 M82 2012-02-14
 M82 2012-03-26
```

```
In [153]: # Extracting a group
          print obs_by_name.groups[1]
```

```
name  obs_date  mag_b mag_v
---- ---------- ----- -----
 M31 2012-01-02  17.0  17.5
 M31 2012-01-02  17.1  17.4
 M31 2012-02-14  16.9  17.3
```

```
In [154]: # Using a mask to select entries
          mask = obs_by_name.groups.keys['name'] == 'M101'
          print mask
          print obs_by_name.groups[mask]
```

```
[ True False False]
name  obs_date  mag_b mag_v
---- ---------- ----- -----
M101 2012-01-02  15.1  13.5
M101 2012-02-14  15.0  13.6
M101 2012-03-26  15.1  13.5
M101 2012-03-26  14.8  14.3
```

```
In [155]: # Some functions can be applied to the elements of a group
          obs_mean = obs_by_name.groups.aggregate(np.mean)
          print obs_mean
```

```
name mag_b mag_v
---- ----- ------
M101  15.0 13.725
 M31  17.0   17.4
 M82  15.7   15.5
```

```
In [156]: print obs_by_name['name', 'mag_v', 'mag_b'].groups.aggregate(np.mean)
```

```
name mag_v  mag_b
---- ------ -----
M101 13.725  15.0
 M31   17.4  17.0
 M82   15.5  15.7
```

```
In [157]: # creat a new Table on the fly
          obs1 = Table.read("""name    obs_date    mag_b  logLx
          M31     2012-01-02  17.0   42.5
          M82     2012-10-29  16.2   43.5
          M101    2012-10-31  15.1   44.5""", format='ascii')
```

```
In [158]: # this is used to stack Tables
          from astropy.table import vstack
```

```
In [159]: tvs = vstack([obs, obs1])
          tvs
```

```
Out[159]: <Table masked=True length=13>
          name  obs_date   mag_b   mag_v   logLx
          str4   str10    float64 float64 float64
          ---- ---------- ------- ------- -------
           M31 2012-01-02    17.0    17.5      --
           M31 2012-01-02    17.1    17.4      --
          M101 2012-01-02    15.1    13.5      --
           M82 2012-02-14    16.2    14.5      --
           M31 2012-02-14    16.9    17.3      --
           M82 2012-02-14    15.2    15.5      --
          M101 2012-02-14    15.0    13.6      --
           M82 2012-03-26    15.7    16.5      --
          M101 2012-03-26    15.1    13.5      --
          M101 2012-03-26    14.8    14.3      --
           M31 2012-01-02    17.0      --    42.5
           M82 2012-10-29    16.2      --    43.5
          M101 2012-10-31    15.1      --    44.5
```

There is a lot of possibilities of joining Tables, see http://docs.astropy.org/en/stable/table/operations.html

```
In [160]: t = Table.read("ftp://cdsarc.u-strasbg.fr/pub/cats/J/other/RMxAA/45.261/c
                         format='ascii.cds',
                         readme='ftp://cdsarc.u-strasbg.fr/pub/cats/J/other/RMxAA/4

Downloading ftp://cdsarc.u-strasbg.fr/pub/cats/J/other/RMxAA/45.261/digeda.dat [Don
Downloading ftp://cdsarc.u-strasbg.fr/pub/cats/J/other/RMxAA/45.261/ReadMe [Done]
```

```
In [161]: t
```

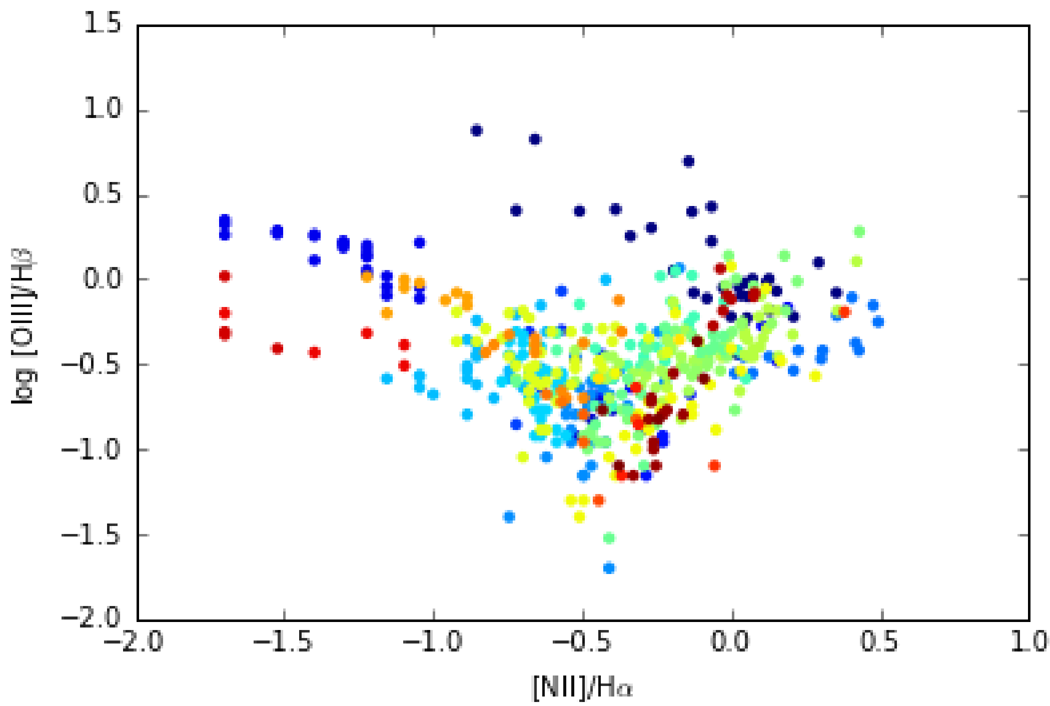
```
Out[161]: <Table masked=True length=1061>
          ObsID   Pos    I3727   I4363    IHb    I4959  ... MType  Slit Region Gal
                   pc                                   ...
          int64 float64 float64 float64 float64 float64 ... int64 int64 int64  int6
          ----- ------- ------- ------- ------- ------- ... ----- ----- ------ ----
              1    0.03      --      --     1.0     0.2 ...    12     3      1
              2    0.03      --      --     1.0    0.33 ...    12     3      1
              3    0.05      --      --     1.0    0.32 ...    12     3      1
              4    0.06      --      --     1.0    0.12 ...    12     3      1
              5    0.07      --      --     1.0    0.27 ...    12     3      1
              6    0.12      --      --     1.0    0.31 ...    12     3      1
              7    0.13      --      --     1.0    0.29 ...    12     3      1
              8    0.15      --      --     1.0     0.3 ...    12     3      1
              9    0.15      --      --     1.0    0.57 ...    12     3      1
            ...     ...     ...     ...     ...     ... ...   ...   ...    ...    ..
           1052    -1.0      --      --    0.35      -- ...     2     3      3    9
```

1053	-1.0	--	--	0.35	--	...	2	3	3	9
1054	-1.0	--	--	0.35	--	...	2	3	3	9
1055	-1.0	--	--	0.35	--	...	2	3	1	9
1056	-1.0	--	--	0.35	--	...	2	3	3	9
1057	-1.0	--	--	0.35	--	...	2	3	1	9
1058	-1.0	--	--	0.35	--	...	2	3	3	9
1059	-1.0	--	--	0.35	--	...	2	3	3	9
1060	-1.0	--	--	0.35	--	...	2	3	1	9
1061	-1.0	--	--	0.35	--	...	2	3	3	9

In [162]:
```
t.show_in_browser(jsviewer=True)
```

In [163]:
```
plt.scatter(np.log10(t['I6583']), np.log10(t['I5007']), c=t['RefN'], edge
plt.xlabel(r'[NII]/H$\alpha$')
plt.ylabel(r'log [OIII]/H$\beta$')
```

Out[163]:
```
&lt;matplotlib.text.Text at 0x114fe0290&gt;
```



In [164]:
```
t = Table.read("ftp://cdsarc.u-strasbg.fr/pub/cats/VII/253/snrs.dat",
readme="ftp://cdsarc.u-strasbg.fr/pub/cats/VII/253/ReadMe",
format="ascii.cds")
```

```
Downloading ftp://cdsarc.u-strasbg.fr/pub/cats/VII/253/snrs.dat [Done]
Downloading ftp://cdsarc.u-strasbg.fr/pub/cats/VII/253/ReadMe [Done]
```

```
In [165]: t

Out[165]: <Table masked=True length=274>
              SNR       RAh   RAm   RAs  DE-  ... u_S(1GHz) Sp-Index u_Sp-Index   Na
                         h    min    s        ...
             str11     int64 int64 int64 str1 ...    str1    float64     str1      st
          ----------- ----- ----- ----- ---- ... --------- -------- ---------- ----
          G000.0+00.0    17    45    44    - ...         ?      0.8          ? Sgr
          G000.3+00.0    17    46    15    - ...        --      0.6         --
          G000.9+00.1    17    47    21    - ...         ?       --          v
          G001.0-00.1    17    48    30    - ...        --      0.6          ?
          G001.4-00.1    17    49    39    - ...         ?       --          ?
          G001.9+00.3    17    48    45    - ...        --      0.6         --
          G003.7-00.2    17    55    26    - ...        --     0.65         --
          G003.8+00.3    17    52    55    - ...         ?      0.6         --
          G004.2-03.5    18     8    55    - ...         ?      0.6          ?
                  ...   ...   ...   ...  ... ...       ...      ...        ...
          G356.3-00.3    17    37    56    - ...         ?       --          ?
          G356.3-01.5    17    42    35    - ...         ?       --          ?
          G357.7-00.1    17    40    29    - ...        --      0.4         --  MSH
          G357.7+00.3    17    38    35    - ...        --      0.4          ?
          G358.0+03.8    17    26     0    - ...         ?       --          ?
          G358.1+00.1    17    37     0    - ...         ?       --          ?
          G358.5-00.9    17    46    10    - ...         ?       --          ?
          G359.0-00.9    17    46    50    - ...        --      0.5         --
          G359.1-00.5    17    45    30    - ...        --      0.4          ?
          G359.1+00.9    17    39    36    - ...         ?       --          ?

In [166]: t.show_in_browser(jsviewer=True)

In [167]: t[0:10].write('tab_cds1.tex', format='latex')

In [168]: !cat tab_cds1.tex

\begin{table}
\begin{tabular}{cccccccccccccccccc}
SNR & RAh & RAm & RAs & DE- & DEd & DEm & MajDiam & --- & MinDiam & u_MinDiam & typ
 & $\mathrm{h}$ & $\mathrm{min}$ & $\mathrm{s}$ &  & $\mathrm{{}^{\circ}}$ & $\math
G000.0+00.0 & 17 & 45 & 44 & - & 29 & 0 & 3.5 & x & 2.5 & -- & S & -- & 100.0 & ? &
G000.3+00.0 & 17 & 46 & 15 & - & 28 & 38 & 15.0 & x & 8.0 & -- & S & -- & 22.0 & --
G000.9+00.1 & 17 & 47 & 21 & - & 28 & 9 & 8.0 & -- & -- & -- & C & -- & 18.0 & ? &
G001.0-00.1 & 17 & 48 & 30 & - & 28 & 9 & 8.0 & -- & -- & -- & S & -- & 15.0 & -- &
G001.4-00.1 & 17 & 49 & 39 & - & 27 & 46 & 10.0 & -- & -- & -- & S & -- & 2.0 & ? &
G001.9+00.3 & 17 & 48 & 45 & - & 27 & 10 & 1.5 & -- & -- & -- & S & -- & 0.6 & -- &
G003.7-00.2 & 17 & 55 & 26 & - & 25 & 50 & 14.0 & x & 11.0 & -- & S & -- & 2.3 & --
G003.8+00.3 & 17 & 52 & 55 & - & 25 & 28 & 18.0 & -- & -- & -- & S? & -- & 3.0 & ?
G004.2-03.5 & 18 & 8 & 55 & - & 27 & 3 & 28.0 & -- & -- & -- & S & -- & 3.2 & ? & (
G004.5+06.8 & 17 & 30 & 42 & - & 21 & 29 & 3.0 & -- & -- & -- & S & -- & 19.0 & --
\end{tabular}
```

```
\end{table}
```

```
In [169]: t[10:20].write('tab_cds1.ascii', format='ascii', delimiter='|', formats={

In [170]: !cat tab_cds1.ascii

SNR|RAh|RAm|RAs|DE-|DEd|DEm|MajDiam|---|MinDiam|u_MinDiam|type|l_S(1GHz)|S(1GHz)|u_
G004.8+06.2|17|33|25|-|21|34|18.0|--|--|--|S|--|3.0|--|0.6|--|--
G005.2-02.6|18|7|30|-|25|45|18.0|--|--|--|S|--|2.6|?|0.6|?|--
G005.4-01.2|18|2|10|-|24|54|35.0|--|--|--|C?|--|35.0|?|0.2|?|Milne 56
G005.5+00.3|17|57|4|-|24|0|15.0|x|12.0|--|S|--|5.5|--|0.7|--|--
G005.9+03.1|17|47|20|-|22|16|20.0|--|--|--|S|--|3.3|?|0.4|?|--
G006.1+00.5|17|57|29|-|23|25|18.0|x|12.0|--|S|--|4.5|--|0.9|--|--
G006.1+01.2|17|54|55|-|23|5|30.0|x|26.0|--|F|--|4.0|?|0.3|?|--
G006.4-00.1|18|0|30|-|23|26|48.0|--|--|--|C|--|310.0|--|--|v|W28
G006.4+04.0|17|45|10|-|21|22|31.0|--|--|--|S|--|1.3|?|0.4|?|--
G006.5-00.4|18|2|11|-|23|34|18.0|--|--|--|S|--|27.0|--|0.6|--|--


In [171]: t[10:20].write('tab_cds2.ascii', format='ascii.fixed_width', delimiter='|

In [172]: !cat tab_cds2.ascii

|         SNR | RAh | RAm | RAs | DE- | DEd | DEm | MajDiam | --- | MinDiam | u_Mir
| G004.8+06.2 |  17 |  33 |  25 |   - |  21 |  34 |    18.0 |  -- |      -- |
| G005.2-02.6 |  18 |   7 |  30 |   - |  25 |  45 |    18.0 |  -- |      -- |
| G005.4-01.2 |  18 |   2 |  10 |   - |  24 |  54 |    35.0 |  -- |      -- |
| G005.5+00.3 |  17 |  57 |   4 |   - |  24 |   0 |    15.0 |   x |    12.0 |
| G005.9+03.1 |  17 |  47 |  20 |   - |  22 |  16 |    20.0 |  -- |      -- |
| G006.1+00.5 |  17 |  57 |  29 |   - |  23 |  25 |    18.0 |   x |    12.0 |
| G006.1+01.2 |  17 |  54 |  55 |   - |  23 |   5 |    30.0 |   x |    26.0 |
| G006.4-00.1 |  18 |   0 |  30 |   - |  23 |  26 |    48.0 |  -- |      -- |
| G006.4+04.0 |  17 |  45 |  10 |   - |  21 |  22 |    31.0 |  -- |      -- |
| G006.5-00.4 |  18 |   2 |  11 |   - |  23 |  34 |    18.0 |  -- |      -- |
```

The astropy Table can also read FITS files (if containing tables), VO tables and hdf5 format. See more there: http://docs.astropy.org/en/stable/io/unified.html

1.0.3 Time and Dates

The astropy.time package provides functionality for manipulating times and dates. Specific emphasis is placed on supporting time scales (e.g. UTC, TAI, UT1, TDB) and time representations (e.g. JD, MJD, ISO 8601) that are used in astronomy and required to calculate, e.g., sidereal times and barycentric corrections. It uses Cython to wrap the C language ERFA time and calendar routines, using a fast and memory efficient vectorization scheme. More here: http://docs.astropy.org/en/stable/time/index.html

1.0.4 Coordinates

The coordinates package provides classes for representing a variety of celestial/spatial coordinates, as well as tools for converting between common coordinate systems in a uniform way.

```
In [173]: from astropy import units as u
          from astropy.coordinates import SkyCoord
```

```
In [174]: c = SkyCoord(ra=10.5*u.degree, dec=41.2*u.degree, frame='icrs')
          c
```

```
Out[174]: <SkyCoord (ICRS): (ra, dec) in deg
              (10.5, 41.2)>
```

```
In [175]: c = SkyCoord('00 42 00 +41 12 00', 'icrs', unit=(u.hourangle, u.deg))
          c
```

```
Out[175]: <SkyCoord (ICRS): (ra, dec) in deg
              (10.5, 41.2)>
```

```
In [176]: print c.ra, c.dec
```

```
10d30m00s 41d12m00s
```

```
In [177]: c.to_string('decimal')
```

```
Out[177]: u'10.5 41.2'
```

```
In [178]: print c.dec.to_string(format='latex')
```

```
$41^\circ12{}^\prime00{}^{\prime\prime}$
```

$$41^\circ12{}^\prime00{}^{\prime\prime}$$

1.0.5 Modeling

astropy.modeling provides a framework for representing models and performing model evaluation and fitting. It currently supports 1-D and 2-D models and fitting with parameter constraints.

It is designed to be easily extensible and flexible. Models do not reference fitting algorithms explicitly and new fitting algorithms may be added without changing the existing models (though not all models can be used with all fitting algorithms due to constraints such as model linearity).

The goal is to eventually provide a rich toolset of models and fitters such that most users will not need to define new model classes, nor special purpose fitting routines (while making it reasonably easy to do when necessary).

1.0.6 Convolution and filtering

astropy.convolution provides convolution functions and kernels that offers improvements compared to the scipy scipy.ndimage convolution routines, including:

- Proper treatment of NaN values
- A single function for 1-D, 2-D, and 3-D convolution
- Improved options for the treatment of edges
- Both direct and Fast Fourier Transform (FFT) versions
- Built-in kernels that are commonly used in Astronomy

More on http://docs.astropy.org/en/stable/convolution/index.html

```
In [ ]:
```