PyNeb_manual_3

May 5, 2017

1 The atomic data

1.1 Changing the data files used for atomic data

Since, for a given ion, there can be different calculations of atomic data with slightly (or not so slightly!) different results, more than one version of the data can exist for a given ion. The data are stored in and read from different types of files depending on the mechanism of line formation:

- In the case of recombination data, a single xxx_rec_XYZ.fits file contains the line emissivities, in a 2D temperature-density dependent table.
- In the case of collisionally excited lines, the data are organized in different way depending on the file format:
  - The only data format admitted was, originally, fits (extension: .fits). Fits atomic data for a given ion are organized in two files (xxx here represents the ion [e.g., "o_iii" for O III] and XYZ is the acronym of the source paper(s)):
    - the xxx_atom_XYZ.fits file, containing the energies, the statistical weights and the transitions probabilities (the Einstein coefficients Aij)
    - the xxx_coll_XYZ.fits file, containing the temperature-dependent collision strengths.
  - Since version 1.0.1, fits file have being totally replaced by plain ascii tables (extension .dat), which are organized as follows:
    - xxx_levels.dat contain the energy levels and statistical weights. These file are downloaded from NIST and the source papers are encoded within the file and translated by PyNeb into an explicit bibliographic reference.
    - xxx_atom_XYZ.dat contain the transitions probabilities
    - xxx_coll_XYZ.fits contain the temperature-dependent collision strengths
  - Chianti data are also usable if the user installed the Chianti database and set up the XUVTOP environment variable pointing to the corresponding directory.

Each of these file contains information for a finite number of levels N. Specifically, the energy levels and the statistical weights are arrays with N elements (one value per level), while the transition probability tables are triangular N*N matrices with zeros on the main diagonal (transition probabilities are different from zero only for (j -> i) transitions with j > i).

This number is a feature of the particular atomic calculation, not an intrinsic feature of the physical atom, so that an Atom object created within PyNeb will have a number of levels NLevels equal to the number of levels of all the data used (or the minimum of them if either of the atom, coll or levels is different).

You can access the NIST raw data by:

```
In [1]: %matplotlib inline
        %config InlineBackend.figure_format = 'svg'
        import pyneb as pn
        O3 = pn.Atom('O', 3)
```

```
In [2]: O3.NIST
```

```
Out[2]: array([(b'2s2.2p2', b'3P',  0.,      0.   , b'L7288'),
               (b'2s2.2p2', b'3P',  1.,    113.178, b'L7288'),
               (b'2s2.2p2', b'3P',  2.,    306.174, b'L7288'),
               (b'2s2.2p2', b'1D',  2.,  20273.27 , b'L7288'),
               (b'2s2.2p2', b'1S',  0.,  43185.74 , b'L7288'),
               (b'2s.2p3', b'5S*',  2.,  60324.79 , b'L7288')],
              dtype=[('conf', 'S23'), ('term', 'S9'), ('J', '<f8'), ('energy', '<f8'), ('ref', 'S10')])
```

You can access the full raw NIST data set by:

```
In [3]: pn.utils.manage_atomic_data.getLevelsNIST('O2') #O2 is lighter than O3...
```

```
Out[3]: array([(b'2s2.2p3', b'4S*',  1.5,       0.   , b'L11267'),
               (b'2s2.2p3', b'2D*',  2.5,   26810.55 , b'L11267'),
               (b'2s2.2p3', b'2D*',  1.5,   26830.57 , b'L11267'),
               (b'2s2.2p3', b'2P*',  1.5,   40468.01 , b'L11267'),
               (b'2s2.2p3', b'2P*',  0.5,   40470.   , b'L11267'),
               (b'2s.2p4', b'4P',  2.5,  119837.21 , b'L11267'),
               (b'2s.2p4', b'4P',  1.5,  120000.43 , b'L11267'),
               (b'2s.2p4', b'4P',  0.5,  120082.86 , b'L11267'),
               (b'2s.2p4', b'2D',  2.5,  165988.46 , b'L11267'),
               (b'2s.2p4', b'2D',  1.5,  165996.5  , b'L11267'),
               (b'2s2.2p2.(3P).3s', b'4P',  0.5,  185235.281, b'L11267'),
               (b'2s2.2p2.(3P).3s', b'4P',  1.5,  185340.577, b'L11267'),
               (b'2s2.2p2.(3P).3s', b'4P',  2.5,  185499.124, b'L11267'),
               (b'2s2.2p2.(3P).3s', b'2P',  0.5,  188888.543, b'L11267'),
               (b'2s2.2p2.(3P).3s', b'2P',  1.5,  189068.514, b'L11267'),
               (b'2s.2p4', b'2S',  0.5,  195710.47 , b'L11267'),
               (b'2s2.2p2.(3P).3p', b'2S*',  0.5,  203942.288, b'L11267'),
               (b'2s2.2p2.(3P).3p', b'4D*',  0.5,  206730.762, b'L11267'),
               (b'2s2.2p2.(3P).3p', b'4D*',  1.5,  206786.286, b'L11267'),
               (b'2s2.2p2.(3P).3p', b'4D*',  2.5,  206877.865, b'L11267'),
               (b'2s2.2p2.(1D).3s', b'2D',  2.5,  206971.68 , b'L11267'),
               (b'2s2.2p2.(1D).3s', b'2D',  1.5,  206972.72 , b'L11267'),
               (b'2s2.2p2.(3P).3p', b'4D*',  3.5,  207002.482, b'L11267'),
               (b'2s2.2p2.(3P).3p', b'4P*',  0.5,  208346.104, b'L11267'),
               (b'2s2.2p2.(3P).3p', b'4P*',  1.5,  208392.258, b'L11267'),
               (b'2s2.2p2.(3P).3p', b'4P*',  2.5,  208484.202, b'L11267'),
               (b'2s2.2p2.(3P).3p', b'2D*',  1.5,  211522.117, b'L11267'),
               (b'2s2.2p2.(3P).3p', b'2D*',  2.5,  211712.732, b'L11267'),
               (b'2s2.2p2.(3P).3p', b'4S*',  1.5,  212161.881, b'L11267'),
               (b'2s.2p4', b'2P',  1.5,  212593.82 , b'L11267'),
               (b'2s.2p4', b'2P',  0.5,  212762.25 , b'L11267'),
               (b'2s2.2p2.(3P).3p', b'2P*',  0.5,  214169.92 , b'L11267'),
               (b'2s2.2p2.(3P).3p', b'2P*',  1.5,  214229.671, b'L11267'),
               (b'2s2.2p2.(1D).3p', b'2F*',  2.5,  228723.84 , b'L11267'),
               (b'2s2.2p2.(1D).3p', b'2F*',  3.5,  228747.45 , b'L11267'),
               (b'2s2.2p2.(1D).3p', b'2D*',  2.5,  229947.07 , b'L11267'),
               (b'2s2.2p2.(1D).3p', b'2D*',  1.5,  229968.44 , b'L11267'),
               (b'2s2.2p2.(1S).3s', b'2S',  0.5,  230609.45 , b'L11267'),
               (b'2s2.2p2.(3P).3d', b'4F',  1.5,  231296.126, b'L11267'),
               (b'2s2.2p2.(3P).3d', b'4F',  2.5,  231350.087, b'L11267'),
               (b'2s2.2p2.(3P).3d', b'4F',  3.5,  231427.97 , b'L11267'),
```

```
 (b'2s2.2p2.(3P).3d', b'4F',  4.5,  231530.246, b'L11267'),
 (b'2s2.2p2.(3P).3d', b'4P',  2.5,  232462.724, b'L11267'),
 (b'2s2.2p2.(1D).3p', b'2P*',  0.5,  232480.44 , b'L11267'),
 (b'2s2.2p2.(1D).3p', b'2P*',  1.5,  232527.09 , b'L11267'),
 (b'2s2.2p2.(3P).3d', b'4P',  1.5,  232535.949, b'L11267'),
 (b'2s2.2p2.(3P).3d', b'4P',  0.5,  232602.492, b'L11267'),
 (b'2s2.2p2.(3P).3d', b'4D',  0.5,  232711.642, b'L11267'),
 (b'2s2.2p2.(3P).3d', b'4D',  1.5,  232745.981, b'L11267'),
 (b'2s2.2p2.(3P).3d', b'4D',  2.5,  232747.562, b'L11267'),
 (b'2s2.2p2.(3P).3d', b'4D',  3.5,  232753.816, b'L11267'),
 (b'2s2.2p2.(3P).3d', b'2F',  2.5,  232796.298, b'L11267'),
 (b'2s2.2p2.(3P).3d', b'2F',  3.5,  232959.21 , b'L11267'),
 (b'2s2.2p2.(3P).3d', b'2P',  1.5,  233430.53 , b'L11267'),
 (b'2s2.2p2.(3P).3d', b'2P',  0.5,  233544.59 , b'L11267'),
 (b'2s2.2p2.(3P).3d', b'2D',  1.5,  234402.797, b'L11267'),
 (b'2s2.2p2.(3P).3d', b'2D',  2.5,  234454.634, b'L11267'),
 (b'2s2.2p2.(3P).4s', b'4P',  0.5,  238627.46 , b'L11267'),
 (b'2s2.2p2.(3P).4s', b'4P',  1.5,  238732.65 , b'L11267'),
 (b'2s2.2p2.(3P).4s', b'4P',  2.5,  238893.96 , b'L11267'),
 (b'2s2.2p2.(3P).4s', b'2P',  0.5,  240330.01 , b'L11267'),
 (b'2s2.2p2.(3P).4s', b'2P',  1.5,  240517.35 , b'L11267'),
 (b'2s2.2p2.(3P).4p', b'2S*',  0.5,  245037.29 , b'L11267')],
dtype=[('conf', 'S23'), ('term', 'S9'), ('J', '<f8'), ('energy', '<f8'), ('ref', 'S10')])
```

You can print the name of the atomic data used for a given ion with:

```
In [4]: print(O3.atomFile)
```

```
o_iii_atom_SZ00-WFD96.dat
```

```
In [5]: print(O3.collFile)
```

```
o_iii_coll_SSB14.dat
```

As the files can be located in different directories (including some from the user), you may also find useful to print out the path:

```
In [6]: print(O3.atomPath)
        print(O3.collPath)
```

```
/home/morisset/anaconda2/envs/py3k6/lib/python3.6/site-packages/pyneb/utils/../atomic_data/
/home/morisset/anaconda2/envs/py3k6/lib/python3.6/site-packages/pyneb/utils/../atomic_data/
```

A set of data files for a given group of ions is called a data set. The default data set (or the adopted data set, if a different one has been set) can be displayed with the command:

```
In [7]: pn.atomicData.getDataFile()
```

```
Out[7]: {'3He2': {'atom': '3he_ii_atom_cloudy.dat', 'coll': '3he_ii_coll_cloudy.dat'},
         'Al2': {'atom': 'al_ii_atom_JSP86-HK87-VVF96-KS86.dat',
          'coll': 'al_ii_coll_KHAF92-TBK85-TBK84.dat'},
         'Ar2': {'atom': 'ar_ii_atom_Bal06.dat', 'coll': 'ar_ii_coll_PB95.dat'},
         'Ar3': {'atom': 'ar_iii_atom_M83-KS86.dat', 'coll': 'ar_iii_coll_GMZ95.dat'},
         'Ar4': {'atom': 'ar_iv_atom_MZ82.dat', 'coll': 'ar_iv_coll_RB97.dat'},
         'Ar5': {'atom': 'ar_v_atom_LL93-MZ82-KS86.dat',
          'coll': 'ar_v_coll_GMZ95.dat'},
```

```
'Ba2': {'atom': 'ba_ii_atom_C04.dat', 'coll': 'ba_ii_coll_SB98.dat'},
'Ba4': {'atom': 'ba_iv_atom_BHQZ95.dat', 'coll': 'ba_iv_coll_SB98.dat'},
'Br3': {'atom': 'br_iii_atom_BH86.dat', 'coll': 'br_iii_coll_S97.dat'},
'Br4': {'atom': 'br_iv_atom_BH86.dat', 'coll': 'br_iv_coll_S97.dat'},
'C1': {'atom': 'c_i_atom_FFS85.dat',
 'coll': 'c_i_coll_JBK87-PA76.dat',
 'rec': 'c_i_rec_P91.func'},
'C2': {'atom': 'c_ii_atom_GMZ98.dat',
 'coll': 'c_ii_coll_BP92.dat',
 'rec': 'c_ii_rec_P91.func'},
'C3': {'atom': 'c_iii_atom_G83-NS78-WFD96.dat',
 'coll': 'c_iii_coll_Bal85.dat',
 'rec': 'c_iii_rec_P91.func'},
'C4': {'atom': 'c_iv_atom_WFD96.dat',
 'coll': 'c_iv_coll_AK04.dat',
 'rec': 'c_iv_rec_P91.func'},
'Ca5': {'atom': 'ca_v_atom_M83-KS86.dat', 'coll': 'ca_v_coll_GMZ95.dat'},
'Cl2': {'atom': 'cl_ii_atom_MZ83.dat', 'coll': 'cl_ii_coll_T04.dat'},
'Cl3': {'atom': 'cl_iii_atom_M83-KS86.dat', 'coll': 'cl_iii_coll_BZ89.dat'},
'Cl4': {'atom': 'cl_iv_atom_KS86-MZ82-EM84.dat',
 'coll': 'cl_iv_coll_GMZ95.dat'},
'Fe2': {'atom': 'fe_ii_atom_B15.dat', 'coll': 'fe_ii_coll_B15.dat'},
'Fe3': {'atom': 'fe_iii_atom_Q96_J00.dat', 'coll': 'fe_iii_coll_Z96.dat'},
'Fe4': {'atom': 'fe_iv_atom_FFRR08.dat', 'coll': 'fe_iv_coll_ZP97.dat'},
'Fe5': {'atom': 'fe_v_atom_Nal00.dat', 'coll': 'fe_v_coll_BGMcL07.dat'},
'Fe6': {'atom': 'fe_vi_atom_CP00.dat', 'coll': 'fe_vi_coll_CP99.dat'},
'Fe7': {'atom': 'fe_vii_atom_WB08.dat', 'coll': 'fe_vii_coll_WB08.dat'},
'H1': {'rec': 'h_i_rec_SH95.hdf5'},
'He1': {'rec': 'he_i_rec_Pal12-Pal13.hdf5'},
'He2': {'rec': 'he_ii_rec_SH95.hdf5'},
'K4': {'atom': 'k_iv_atom_M83-KS86.dat', 'coll': 'k_iv_coll_GMZ95.dat'},
'K5': {'atom': 'k_v_atom_M83-KS86.dat', 'coll': 'k_v_coll_BZL88.dat'},
'Kr3': {'atom': 'kr_iii_atom_BH86.dat', 'coll': 'kr_iii_coll_S97.dat'},
'Kr4': {'atom': 'kr_iv_atom_BH86.dat', 'coll': 'kr_iv_coll_S97.dat'},
'Kr5': {'atom': 'kr_v_atom_BH86.dat', 'coll': 'kr_v_coll_S97.dat'},
'Mg5': {'atom': 'mg_v_atom_GMZ97.dat', 'coll': 'mg_v_coll_BZ94.dat'},
'Mg7': {'atom': 'mg_vii_atom_GMZ97.dat', 'coll': 'mg_vii_coll_LB94-U.dat'},
'N1': {'atom': 'n_i_atom_KS86-WFD96.dat',
 'coll': 'n_i_coll_PA76-DMR76.dat',
 'rec': 'n_i_rec_P91.func'},
'N2': {'atom': 'n_ii_atom_GMZ97-WFD96.dat',
 'coll': 'n_ii_coll_T11.dat',
 'rec': 'n_ii_rec_P91.func'},
'N3': {'atom': 'n_iii_atom_GMZ98.dat',
 'coll': 'n_iii_coll_BP92.dat',
 'rec': 'n_iii_rec_P91.func'},
'N4': {'atom': 'n_iv_atom_WFD96.dat',
 'coll': 'n_iv_coll_RBHB94.dat',
 'rec': 'n_iv_rec_P91.func'},
'N5': {'rec': 'n_v_rec_P91.func'},
'Na4': {'atom': 'na_iv_atom_GMZ97.dat', 'coll': 'na_iv_coll_BZ94.dat'},
'Na6': {'atom': 'na_vi_atom_GMZ97.dat', 'coll': 'na_vi_coll_LB94.dat'},
'Ne2': {'atom': 'ne_ii_atom_Bal06.dat', 'coll': 'ne_ii_coll_GMB01.dat'},
'Ne3': {'atom': 'ne_iii_atom_GMZ97.dat', 'coll': 'ne_iii_coll_McLB00.dat'},
```

```
'Ne4': {'atom': 'ne_iv_atom_BBZ89-BK88.dat', 'coll': 'ne_iv_coll_G81.dat'},
'Ne5': {'atom': 'ne_v_atom_GMZ97-U-BD93.dat', 'coll': 'ne_v_coll_LB94.dat'},
'Ne6': {'atom': 'ne_vi_atom_GMZ98.dat', 'coll': 'ne_vi_coll_ZGP94.dat'},
'Ni3': {'atom': 'ni_iii_atom_B01.dat', 'coll': 'ni_iii_coll_B01.dat'},
'O1': {'atom': 'o_i_atom_WFD96.dat',
 'coll': 'o_i_coll_BK95.dat',
 'rec': 'o_i_rec_P91.func'},
'O2': {'atom': 'o_ii_atom_Z82-WFD96.dat',
 'coll': 'o_ii_coll_P06-T07.dat',
 'rec': 'o_ii_rec_SSB17-B-opt.hdf5'},
'O3': {'atom': 'o_iii_atom_SZ00-WFD96.dat',
 'coll': 'o_iii_coll_SSB14.dat',
 'rec': 'o_iii_rec_P91.func'},
'O4': {'atom': 'o_iv_atom_GMZ98.dat',
 'coll': 'o_iv_coll_BP92.dat',
 'rec': 'o_iv_rec_P91.func'},
'O5': {'atom': 'o_v_atom_H80-NS79.dat',
 'coll': 'o_v_coll_BBDK85.dat',
 'rec': 'o_v_rec_P91.func'},
'O6': {'rec': 'o_vi_rec_P91.func'},
'P2': {'atom': 'p_ii_atom_MZ82.dat', 'coll': 'p_ii_coll_T04.dat'},
'Rb4': {'atom': 'rb_iv_atom_BH86.dat', 'coll': 'rb_iv_coll_S97.dat'},
'Rb5': {'atom': 'rb_v_atom_BH86.dat', 'coll': 'rb_v_coll_S97.dat'},
'Rb6': {'atom': 'rb_vi_atom_BH86.dat', 'coll': 'rb_vi_coll_S97.dat'},
'S2': {'atom': 's_ii_atom_PKW09.dat', 'coll': 's_ii_coll_TZ10.dat'},
'S3': {'atom': 's_iii_atom_PKW09.dat', 'coll': 's_iii_coll_TG99.dat'},
'S4': {'atom': 's_iv_atom_JKD86-DHKD82.dat', 'coll': 's_iv_coll_DHKD82.dat'},
'Se3': {'atom': 'se_iii_atom_BH86.dat', 'coll': 'se_iii_coll_S97.dat'},
'Se4': {'atom': 'se_iv_atom_B05.dat', 'coll': 'se_iv_coll_B05.dat'},
'Si2': {'atom': 'si_ii_atom_BL93-CSB93-N77.dat',
 'coll': 'si_ii_coll_DK91.dat'},
'Si3': {'atom': 'si_iii_atom_M83-OKH88-FW90-KS86.dat',
 'coll': 'si_iii_coll_DK94.dat'},
'Xe3': {'atom': 'xe_iii_atom_BHQZ95.dat', 'coll': 'xe_iii_coll_SB98.dat'},
'Xe4': {'atom': 'xe_iv_atom_BHQZ95.dat', 'coll': 'xe_iv_coll_SB98.dat'},
'Xe6': {'atom': 'xe_vi_atom_BHQZ95.dat', 'coll': 'xe_vi_coll_SB98.dat'}}
```

while the data files set for a particular ion can be displayed by providing an argument to the above; e.g. for [SIV]:

```
In [8]: pn.atomicData.getDataFile('S4')

warng _ManageAtomicData: rec data not available for S4
warng _ManageAtomicData: trc data not available for S4

Out[8]: ('s_iv_atom_JKD86-DHKD82.dat', 's_iv_coll_DHKD82.dat', None, None)
```

The complete inventory of data available for a given ion can be displayed with the command:

```
In [9]: pn.atomicData.getAllAvailableFiles('O2')

Out[9]: ['o_ii_atom.chianti',
         'o_ii_atom_FFT04.dat',
         'o_ii_atom_WFD96.dat',
         'o_ii_atom_Z82-WFD96.dat',
         'o_ii_coll.chianti',
```

```
        'o_ii_coll_Kal09.dat',
        'o_ii_coll_P06-T07.dat',
        'o_ii_coll_P76-McLB93-v1.dat',
        'o_ii_coll_P76-McLB93-v2.dat',
        'o_ii_coll_T07.dat',
        'o_ii_rec_SSB17-A-IR.hdf5',
        'o_ii_rec_SSB17-A-opt.hdf5',
        'o_ii_rec_SSB17-B-IR.hdf5',
        'o_ii_rec_SSB17-B-opt.hdf5',
        'o_ii_rec_SSB17-C-IR.hdf5',
        'o_ii_rec_SSB17-C-opt.hdf5']
```

This method looks for all the "o_ii_*" files in a set of paths which includes the location of the atomic data provided with the package and the current directory (the one from where the python session is running); additional paths can be added with the following command:

```
In [10]: pn.atomicData.addDataFilePath("/tmp")
```

If you want to change several data files at once, it may be worth defining a dictionary with all your preferred atomic data files within your script:

```
In [11]: DataFileDict = {'N1': {'atom': 'n_i_atom_KS86-WFD96.dat', 'coll': 'n_i_coll_PA76-DMR76.dat'},
                         'N2': {'atom': 'n_ii_atom_GMZ97-WFD96.dat', 'coll': 'n_ii_coll_LB94.dat'},
                         'O2': {'atom': 'o_ii_atom_Z82-WFD96.dat', 'coll': 'o_ii_coll_P06-T07.dat'},
                         'O3': {'atom': 'o_iii_atom_FFT04-SZ00.dat', 'coll': 'o_iii_coll_AK99.dat'},
                         'Ne3': {'atom': 'ne_iii_atom_GMZ97.dat', 'coll': 'ne_iii_coll_McLB00.dat'}}
         pn.atomicData.setDataFileDict(DataFileDict)
```

Make sure that all the files listed actually exist, as this is not checked by the code at this level.

1.2 Predefined sets of atomic data

Predefined atomic data sets are provided by PyNeb. Each data set is identified by a label; the one of the default data set can be retrieved by the following command:

```
In [12]: pn.atomicData.defaultDict

Out[12]: 'PYNEB_17_01'
```

To display the complete list of the existing predefined dictionaries, enter:

```
In [13]: pn.atomicData.getAllPredefinedDict()

Out[13]: dict_keys(['IRAF_09_orig', 'IRAF_09', 'PYNEB_13_01', 'PYNEB_14_01', 'PYNEB_14_02', 'PYNEB_14_03',
```

The following command can be used to set one of them:

```
In [14]: pn.atomicData.includeFitsPath()
         pn.atomicData.setDataFileDict("IRAF_09")
```

To revert to the default set:

```
In [15]: pn.atomicData.resetDataFileDict()
         pn.atomicData.removeFitsPath()
```

You can have a look at the other methods of pn.atomicData in the Reference Manual, or using the **help** command.

1.3 Using CHIANTI database

You may have downloaded the CHIANTI database from the website http://www.chiantidatabase.org/chianti_download.html. In this case you can use its data to feed PyNeb. You need first to define the environment variable XUVTOP to point to the directory where the database is on your disk.

Then you can ask PyNeb to use the atomic data from CHIANTI for a given ion:

```
In [16]: pn.atomicData.setDataFile('o_iii_atom.chianti')
         pn.atomicData.setDataFile('o_iii_coll.chianti')
         O3_ch = pn.Atom('O',3)
         print(O3_ch)
```

```
Atom O3 from o_3.wgfa and o_3.scups
```

If you need to include all the available CHIANTI data, you can use the following command:

```
In [17]: atoms = pn.atomicData.getAllAtoms()
         print(atoms)
         print('After including CHIANTI:')
         pn.atomicData.addAllChianti()
         atoms = pn.atomicData.getAllAtoms()
         print(atoms)
```

```
['Al2', 'Ar2', 'Ar3', 'Ar4', 'Ar5', 'Ba2', 'Ba4', 'C1', 'C2', 'C3', 'C4', 'Ca5', 'Cl2', 'Cl3', 'Cl4', 'I
After including CHIANTI:
['Al2', 'Ar2', 'Ar3', 'Ar4', 'Ar5', 'Ba2', 'Ba4', 'C1', 'C2', 'C3', 'C4', 'Ca5', 'Cl2', 'Cl3', 'Cl4', 'I
```

1.4 Creating new atomic data files

The following is an example of a 6-level atom file of [O III], containing the matrix of transition probabilities. The energy levels and statistical weigths are not in this file. This file is o_iii_atom_FFT04.dat.

```
Aij
1/s 1/s 1/s 1/s 1/s 1/s
0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
2.5960000e-05 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
3.0300000e-11 9.6320000e-05 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
2.3220000e-06 6.9510000e-03 2.0290000e-02 0.0000000e+00 0.0000000e+00 0.0000000e+00
0.0000000e+00 2.2550000e-01 6.9980000e-04 1.6850000e+00 0.0000000e+00 0.0000000e+00
2.1910000e-03 2.3080000e+02 5.7650000e+02 5.7770000e-03 3.7600000e-11 0.0000000e+00
*** ATOM oxygen
*** SPECTRUM 3
*** GSCONFIG unknown
*** SOURCE3 Froese Fischer et al 2004, ADNDT 87, 1
*** NOTE3 A-values from all levels
```

The effective collision strengths are a function of electron temperature (a collective property of the electron distribution) and are obtained as the average over a Maxwellian distribution of the collision strengths, which depend on the energy. (Note: in this document, we will often refer to the effective collision strengths as collision strengths for short, although this is not strictly correct.) They are usually published for a handful of T values and must be interpolated to get the collision strength at the desired values. As a result, there is a whole 1-D array of collision strengths, with one element for each tabulated temperature value, for each transition j -> i with j > i. The existence of this 3rd dimension prevents the data from being simply stored in a matrix as transition probabilities are. Instead, each transition is presented in a line, as in the following example:

```
*** OIII collision strengths data
0 0 3.699e+00 4.000e+00 4.301e+00 4.477e+00
1 2 5.240e-01 5.648e-01 6.007e-01 6.116e-01
1 3 2.469e-01 2.766e-01 3.106e-01 3.264e-01
1 4 2.347e-01 2.693e-01 3.094e-01 3.256e-01
1 5 4.094e-02 4.069e-02 4.299e-02 4.424e-02
1 6 1.130e-01 1.239e-01 1.346e-01 1.373e-01
2 3 1.210e+00 1.330e+00 1.451e+00 1.499e+00
2 4 7.067e-01 8.108e-01 9.313e-01 9.802e-01
2 5 1.228e-01 1.223e-01 1.294e-01 1.332e-01
2 6 3.390e-01 3.717e-01 4.038e-01 4.119e-01
3 4 1.188e+00 1.363e+00 1.564e+00 1.645e+00
3 5 2.045e-01 2.046e-01 2.170e-01 2.235e-01
3 6 5.650e-01 6.195e-01 6.730e-01 6.865e-01
4 5 4.544e-01 5.661e-01 6.230e-01 6.219e-01
4 6 0.000e+00 0.000e+00 0.000e+00 0.000e+00
5 6 0.000e+00 0.000e+00 0.000e+00 0.000e+00
*** T_UNIT log(K)
*** ATOM oxygen
*** SPECTRUM 3
*** GSCONFIG p2
*** SOURCE1 Palay, E. et al. 2012, MNRAS Letters, 423, L35
*** NOTE1 All collision strengths from levels up to 5
*** SOURCE2 Aggarwal and Keenan 1999, ApJS 123, 311
*** NOTE2 Collision strengths of 6-1, 6-2, 6-3
```

The first row of the data block (0 0 ...) contains, from the third element on, the array of electron temperatures for which the collision strengths are tabulated (in log10(K) or K/10000, depending on the particular data set).

The **original fits data** for collision strengths support, by default, a Chebyshev polynomial interpolation: each transition is described by a Chebyshev polynomial of order n that interpolates the tabulated values. The default value for the Chebyshev order is the number of temperature values, but other order can be specified when calling pn.writeColl. If, for some reason, a Chebyshev interpolation is not desired, the data can also be interpolated linearly, when the instantiation of the corresponding Atom is done.

The **ascii data** (those currently used by default) only support linear interpolation among tabulated values.

Notice the use of **SOURCE** and **NOTE** keywords to store the data references. It is very important to fill in these keywords and to give a descriptive name to the file, i.e. a name that reflects the sources of the data. When an **Atom** object is built, this information is stored and is retrievable thereafter through:

```
In [18]: O3.printSources()
```

```
O3: A values for 4-2 and 4-3: Storey and Zeippen 2000, 312, 813
O3: All other A values: Wiese, Fuhr & Deters, 1996, JPCRD, Monograph 7, 403
O3: Energy levels:
    Ref. b'7288' of NIST 2014 (try this: http://physics.nist.gov/cgi-bin/ASBib1/get_ASBib_ref.cgi?db=el&
O3: Collision strengths: Storey, P. J., Sochi, T., & Badnell, N. R. 2014, A&A, 441, 3028
```

1.5 Plotting atomic data

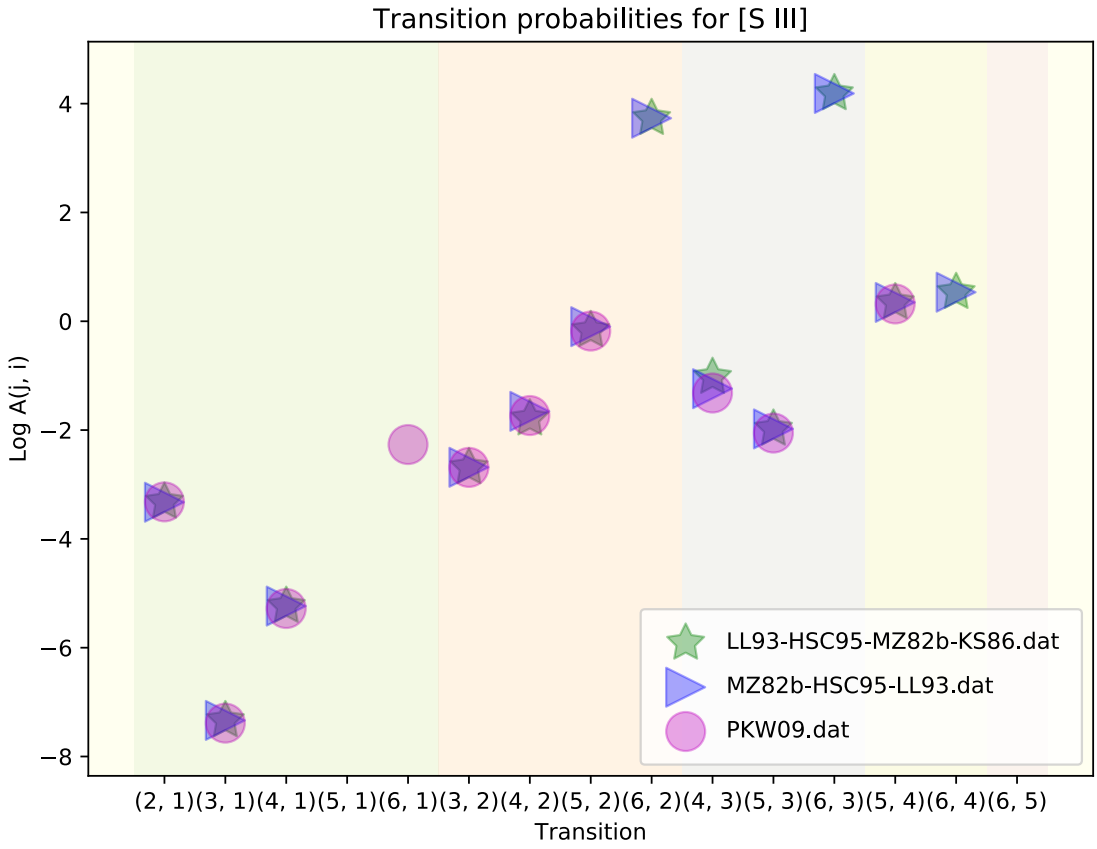
The atomic data available can be plotted using the methods of the **DataPlot** class. The class must be instantiated by specifying which data sets are to be plotted. The available methods are **plotA**, **plotRelA** and **plotAllA** to plot transition probabilities, and **plotOmega** to plot collision strengths.

You must first create an instantiation of DataPlot for a given atom, e.g.:

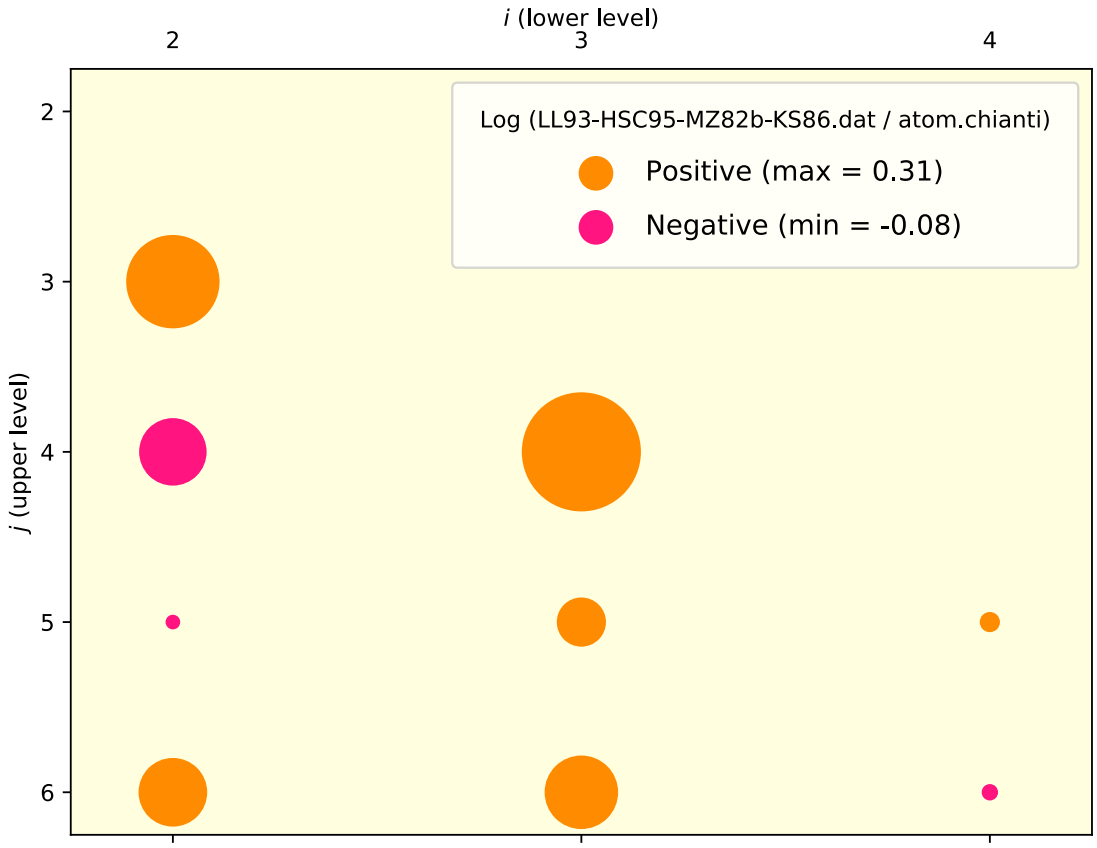
```
In [19]: dp_O3 = pn.DataPlot('O', 3)
         dp_S3 = pn.DataPlot('S', 3)

warng _ManageAtomicData: trc data not available for O3
warng _ManageAtomicData: rec data not available for S3
warng _ManageAtomicData: trc data not available for S3
```

Then you can plot the different As and Omegas :

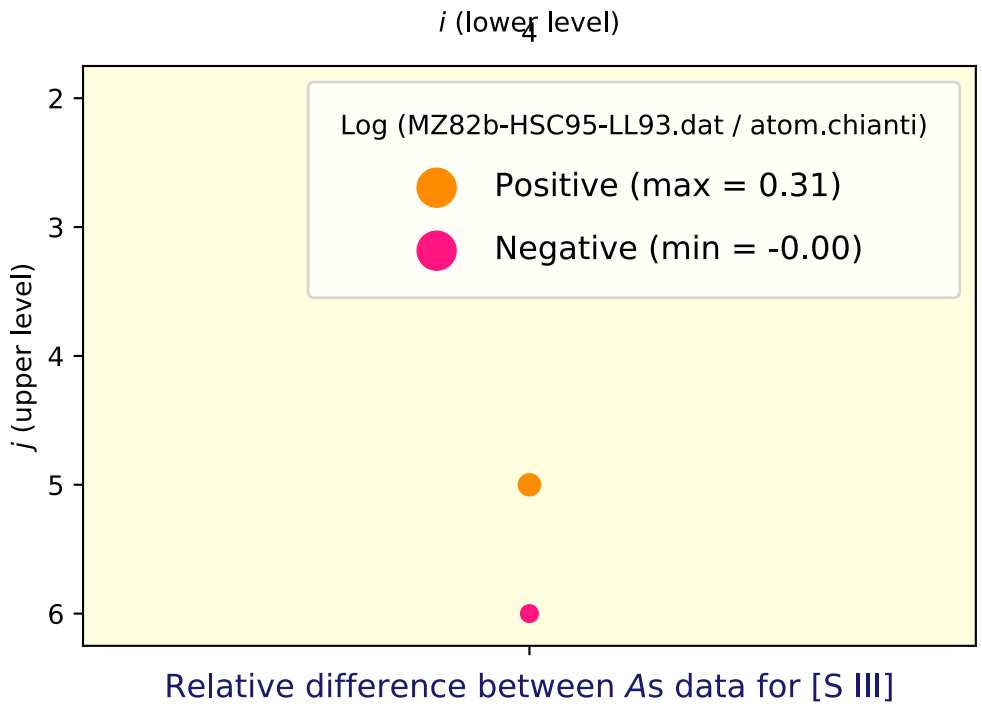
```
In [20]: dp_S3.plotA(figsize=(8, 6)) # transition probabilities plot

/home/morisset/anaconda2/envs/py3k6/lib/python3.6/site-packages/matplotlib/cbook.py:136: MatplotlibDepr
  warnings.warn(message, mplDeprecation, stacklevel=1)

warng DataPlot.plotA: Problem in plotting A
```

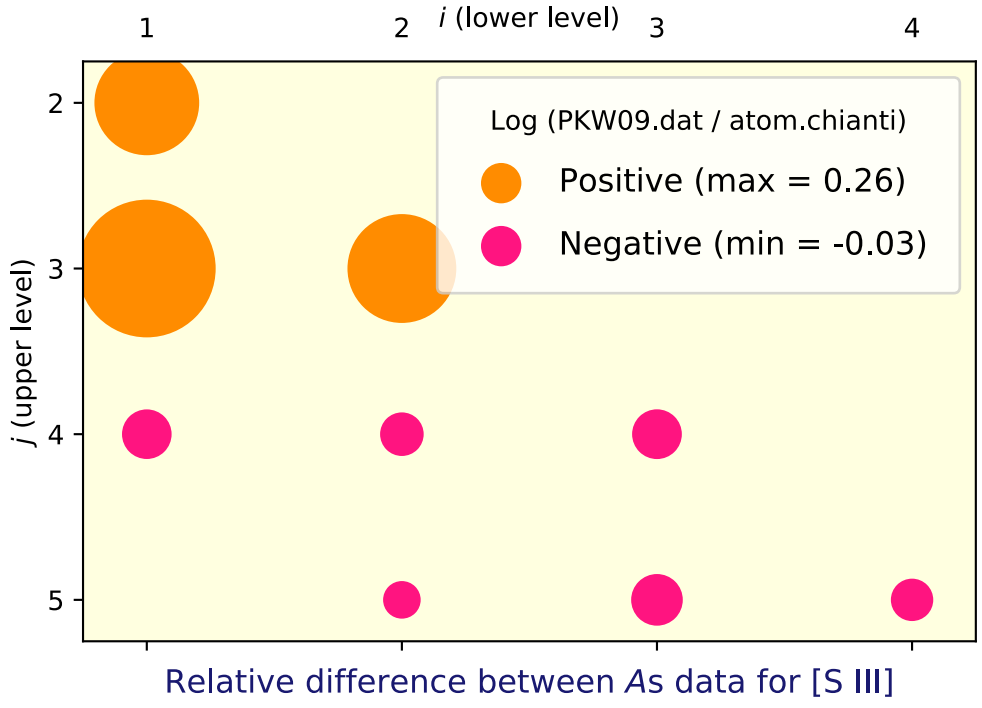



```
In [21]: dp_S3.plotRelA(figsize=(8, 6)) # relative transition probabilities plot

/home/morisset/anaconda2/envs/py3k6/lib/python3.6/site-packages/matplotlib/cbook.py:136: MatplotlibDepr
  warnings.warn(message, mplDeprecation, stacklevel=1)
```



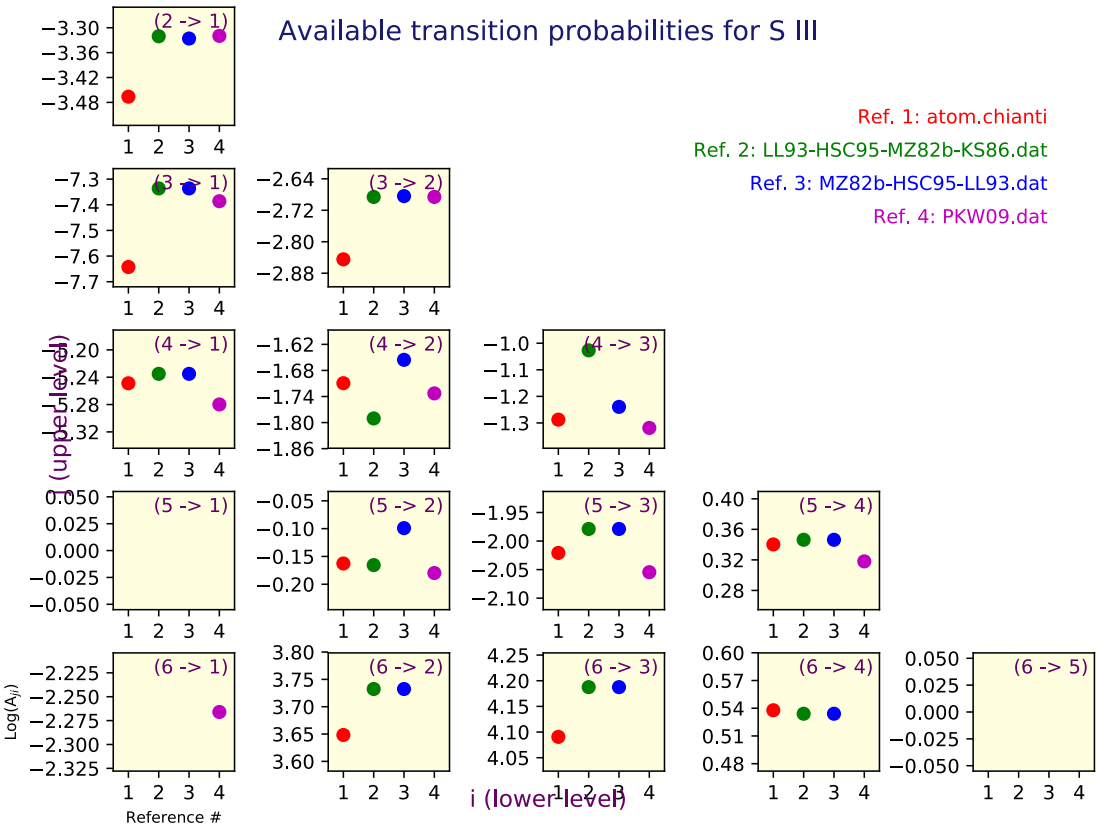
Relative difference between *A*s data for [S III]






```
In [22]: dp_S3.plotAllA(figsize=(8, 6))
```

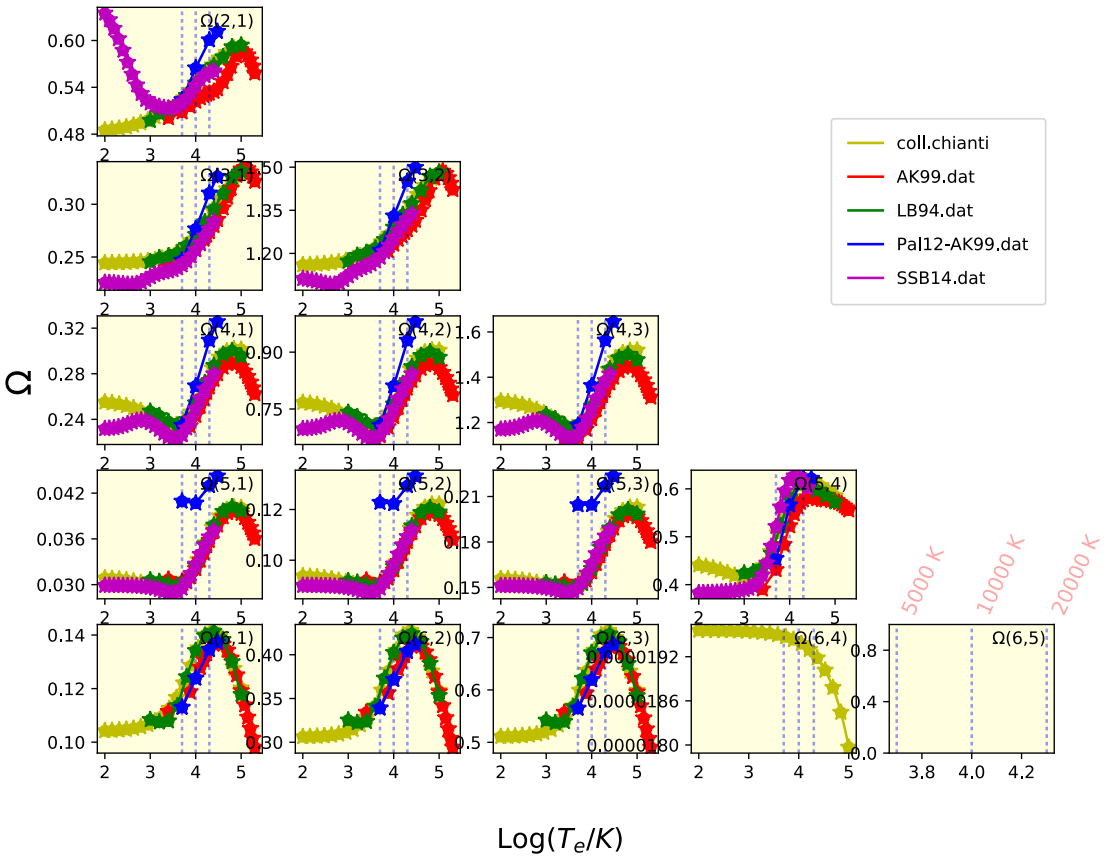
```
/home/morisset/anaconda2/envs/py3k6/lib/python3.6/site-packages/matplotlib/cbook.py:136: MatplotlibDepre
  warnings.warn(message, mplDeprecation, stacklevel=1)
```



In [23]:
```
dp_O3.plotOmega(figsize=(10, 8)) # collision strength plot
```

```
/home/morisset/anaconda2/envs/py3k6/lib/python3.6/site-packages/matplotlib/cbook.py:136: MatplotlibDepre
  warnings.warn(message, mplDeprecation, stacklevel=1)
```

[O III] collision strengths



In [ ]: