PyNeb_manual_8

June 2, 2020

[1]:
```
import pyneb as pn
```

0.1 Determination of elemental abundances using ICFs

The determination of elemental abundances is complicated by the fact that some ions are not observed. To take this into account when computing the total abundances, it is necessary to use ionization correction factors (ICFs). ICFs are expressions used to correct the abundance of observed ions for unseen ions to get the total elemental abundance of a given element:

$$X(elem) = X(ion) * icf,$$

where $X(ion)$ may be the abundance of one single ion or the sum of several ions of the element. These expressions are generally obtained empirically or semiempirically, based on the results of photoionization models or a comparison between the ionization potentials of different ions. Most expressions have been devised for a specific kind of object (e.g., PNe, HII regions, etc) and should not be applied to objects of a different kind.

In PyNeb, an icf formula is identified by an element ("elem", e.g. "O"), which is the element of which the total abundance is seeked for; the ion or ions whose abundance is corrected ("atom", e.g. "O2+O3"); and the icf proper, which is an expression involving the abundances of other ions, assumed to be known ("icf", e.g. "1 + 0.5 * He3 / (He2 + He3)"). In addition, each icf expressions is identified by a label and holds the original reference and a brief comment specifying its intended field of application. The label is formed by an acronym of the paper and the equation number of the icf within the paper.

The following snippet illustrates how PyNeb manages icfs:

[2]:
```
atom_abun = {'O2': 0.001, 'O3': 0.002, 'Ne3': 1.2e-5}
icf = pn.ICF()
print(icf.getAvailableICFs('Ne'))   # lists all the available recipes for Ne
```

```
{'Ne': ['direct_Ne.23', 'direct_Ne.235', 'direct_Ne.2345', 'direct_Ne.345',
'direct_Ne.2356', 'TPP77_15', 'PHCD07_12', 'PHCD07_13', 'KB94_A27',
'KB94_A28.6', 'KB94_A28.8', 'KB94_A28.10', 'KB94_A28.10b', 'KH01_4d', 'PC69_40',
'S78_265b', 'DIMS14_17a', 'DIMS14_17b', 'DIMS14_17c', 'DIMS14_20', 'Ial06_19a',
'Ial06_19b', 'Ial06_19c']}
```

[3]:
```
elem_abun = icf.getElemAbundance(atom_abun, icf_list=['TPP77_15']) # Computes␣
 ↪the Ne abundance with the TPP06 recipe
print(elem_abun)
```

```
{'TPP77_15': 1.8e-05}
```

[4]:
```
elem_abun = icf.getElemAbundance(elem_abun)  # performs all the possible␣
↪element abundance computations given the input ionic abundance set and␣
↪available icf
print(elem_abun)
```

```
warng ICF: 1./(1. - abun["N2"] / elem_abun["KB94_A1.10b"]) cannot be evaluated
for KB94_A30.10b
{'direct_He.23': nan, 'direct_N.23': nan, 'direct_N.234': nan, 'direct_N.2345':
nan, 'direct_O.23': nan, 'direct_O.234': nan, 'direct_O.2345': nan,
'direct_S.23': nan, 'direct_S.234': nan, 'direct_S.2345': nan, 'direct_Ne.23':
nan, 'direct_Ne.235': nan, 'direct_Ne.2345': nan, 'direct_Ne.345': nan,
'direct_Ne.2356': nan, 'direct_Mg.45': nan, 'direct_Ar.23': nan,
'direct_Ar.234': nan, 'direct_Ar.345': nan, 'direct_Cl.23': nan, 'direct_Cl.34':
nan, 'direct_Cl.234': nan, 'TPP77_13': nan, 'TPP77_14': nan, 'TPP77_15': nan,
'PHCD07_12': nan, 'PHCD07_13': nan, 'PHCD07_16': nan, 'PHCD07_17': nan,
'ITL94_19': nan, 'ITL94_20': nan, 'PTPR92_21': nan, 'KB94_A0': nan, 'KB94_A6':
nan, 'KB94_A8': nan, 'KB94_A10': nan, 'KB94_A10b': nan, 'KB94_A12': nan,
'KB94_A13.10': nan, 'KB94_A13.10b': nan, 'KB94_A16': nan, 'KB94_A19': nan,
'KB94_A21': nan, 'KB94_A26': nan, 'KB94_A27': nan, 'KB94_A28.6': nan,
'KB94_A28.8': nan, 'KB94_A28.10': nan, 'KB94_A28.10b': nan, 'KB94_A30.0': nan,
'KB94_A32': nan, 'KB94_A36.6': nan, 'KB94_A36.8': nan, 'KB94_A36.10': nan,
'KB94_A36.10b': nan, 'KH01_4a': nan, 'KH01_4b': nan, 'KH01_4c': nan, 'KH01_4d':
nan, 'KH01_4e': nan, 'KH01_4f': nan, 'KH01_4g': nan, 'KH01_4txt': nan,
'Ial06_16': nan, 'Ial06_17': nan, 'PC69_40': nan, 'S78_265b': nan, 'RR05_2':
nan, 'RR05_3': nan, 'RR05_4': nan, 'GKA07_1.p269': nan, 'mGKA07-PTPR92_p269':
nan, 'DIMS14_10': nan, 'DIMS14_12': nan, 'DIMS14_14': nan, 'DIMS14_14b': nan,
'DIMS14_17a': nan, 'DIMS14_17b': nan, 'DIMS14_17c': nan, 'DIMS14_20': nan,
'DIMS14_23': nan, 'DIMS14_26': nan, 'DIMS14_29': nan, 'DIMS14_29b': nan,
'DIMS14_32': nan, 'DIMS14_35': nan, 'DIMS14_36': nan, 'DIMS14_39': nan,
'Ial06_18a': nan, 'Ial06_18b': nan, 'Ial06_18c': nan, 'Ial06_19a': nan,
'Ial06_19b': nan, 'Ial06_19c': nan, 'Ial06_21a': nan, 'Ial06_21b': nan,
'Ial06_21c': nan, 'Ial06_22a': nan, 'Ial06_22b': nan, 'Ial06_22c': nan,
'Ial06_24a': nan, 'Ial06_24b': nan, 'Ial06_24c': nan, 'Ial06_20a': nan,
'Ial06_20b': nan, 'Ial06_20c': nan, 'Ial06_23a': nan, 'Ial06_23b': nan,
'Ial06_23c': nan, 'KB94_A1.6': nan, 'KB94_A1.8': nan, 'KB94_A1.10': nan,
'KB94_A30.10': nan}
```

The first line above defines a set of ionic abundances; the second defines an ICF object; the third lists the label of all the available recipes for Ne; in the fourth, a specific one is selected to compute the desired abundance; in the fifth line, all the possible element abundance computations given the present icf set are performed (these may include icfs not suitable for the object under study).

Additionally, PyNeb provides a series of single-line commands to explore the icf collection and its source papers; e.g.:

[5]:
```
print(icf.getAvailableICFs())
```

```
{'Ar': ['direct_Ar.23', 'direct_Ar.234', 'direct_Ar.345', 'PHCD07_16',
'PHCD07_17', 'ITL94_19', 'ITL94_20', 'KB94_A30.0', 'KB94_A30.10',
'KB94_A30.10b', 'KB94_A32', 'KH01_4g', 'KH01_4txt', 'DIMS14_35', 'DIMS14_36',
'Ial06_22a', 'Ial06_22b', 'Ial06_22c', 'Ial06_23a', 'Ial06_23b', 'Ial06_23c'],
'C': ['KB94_A12', 'KB94_A13.10', 'KB94_A13.10b', 'KB94_A16', 'KB94_A19',
'KB94_A21', 'KB94_A26', 'DIMS14_39'], 'Cl': ['direct_Cl.23', 'direct_Cl.34',
'direct_Cl.234', 'KH01_4f', 'GKA07_1.p269', 'mGKA07-PTPR92_p269', 'DIMS14_29',
'DIMS14_29b', 'DIMS14_32', 'Ial06_21a', 'Ial06_21b', 'Ial06_21c'], 'Fe':
['RR05_2', 'RR05_3', 'RR05_4', 'Ial06_24a', 'Ial06_24b', 'Ial06_24c'], 'He':
['direct_He.23', 'PTPR92_21', 'KH01_4a', 'DIMS14_10'], 'Mg': ['direct_Mg.45'],
'N': ['direct_N.23', 'direct_N.234', 'direct_N.2345', 'TPP77_14', 'KB94_A0',
'KB94_A1.6', 'KB94_A1.8', 'KB94_A1.10', 'KH01_4c', 'DIMS14_14', 'DIMS14_14b',
'Ial06_18a', 'Ial06_18b', 'Ial06_18c'], 'Ne': ['direct_Ne.23', 'direct_Ne.235',
'direct_Ne.2345', 'direct_Ne.345', 'direct_Ne.2356', 'TPP77_15', 'PHCD07_12',
'PHCD07_13', 'KB94_A27', 'KB94_A28.6', 'KB94_A28.8', 'KB94_A28.10',
'KB94_A28.10b', 'KH01_4d', 'PC69_40', 'S78_265b', 'DIMS14_17a', 'DIMS14_17b',
'DIMS14_17c', 'DIMS14_20', 'Ial06_19a', 'Ial06_19b', 'Ial06_19c'], 'O':
['direct_O.23', 'direct_O.234', 'direct_O.2345', 'TPP77_13', 'KB94_A6',
'KB94_A8', 'KB94_A10', 'KB94_A10b', 'KH01_4b', 'Ial06_16', 'Ial06_17',
'DIMS14_12'], 'S': ['direct_S.23', 'direct_S.234', 'direct_S.2345',
'KB94_A36.6', 'KB94_A36.8', 'KB94_A36.10', 'KB94_A36.10b', 'KH01_4e',
'DIMS14_23', 'DIMS14_26', 'Ial06_20a', 'Ial06_20b', 'Ial06_20c']}
```

[6]:
```
print(icf.getExpression('KH01_4g'))   # returns the analytical expression of␣
↪the icf identified by the label KH01_4g
```

```
Ar = (Ar3 + Ar4) * (He2 + He3) / He2 / (1 -  N2 / (KH01_4c))
```

[7]:
```
print(icf.getReference('KH01_4g'))  # returns the bibliographic reference of␣
↪the source paper
```

```
Kwitter & Henry 2001, ApJ, 562, 804
```

[8]:
```
print(icf.getURL('KH01_4g'))   # returns the ADS URL of the source paper
```

```
http://adsabs.harvard.edu/abs/2001ApJ…562..804K
```

PyNeb provides a large set of icfs compiled from the literature; each icf is stored together with the bibliographic reference, the URL and a comment with further details. Further expressions are being added continuously, and a special function (**addICF**) enables users to add customized expressions to the collection.