PyNeb_manual_6

May 5, 2017

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import pyneb as pn
```

0.1 The EmissionLine class

This is the class characterizing emission lines. An emission line is identified by an element and a spectrum (which identify the emitting ion), a wavelength in Angstrom, a blend flag, a label in the standard PyNeb format, an observed intensity, a reddening-corrected intensity, an expression describing how the intensity depends on the included wavelengths, an observational error and an error on the corrected intensity. Other programs determine one or more of these values.

To instantiate an Emission Line object, use the following:

```
In [2]: line = pn.EmissionLine('O', 3, 5007, obsIntens=[1.4, 1.3])
        line2 = pn.EmissionLine(label = 'O3_5007A', obsIntens=320, corrected=True)
```

obsIntens is a value, a list or a numpy array of values corresponding to the observed intensity of the given emission line.

```
In [3]: print(line)
```

```
Line O3 O3_5007A
```

To know how the label of a given line is exactly spelled, you can print the dictionary pn.LINE_LABEL_LIST

```
In [4]: print(pn.LINE_LABEL_LIST['O3'])
```

```
['4931A', '4959A', '5007A', '2315A', '2321A', '2331A', '4363A', '1658A', '1661A', '1666A', '2497A', '58
```

It is possible to instantiate a line not contained in the pn.LINE_LABEL_LIST. In this case a warning is issued, but the code doesn't stop.

The observed intensity is stored in **line.obsIntens** and the extinction-corrected intensity is stored in **line.corrIntens**. **line.corrIntens** is set to 0.0 when the line is instantiated, unless the parameter corrected is set to **True**, in which case the observed value **obsIntens** is copied to the **corrIntens** slot (the same applies for **corrError**, which is set to **obsError**).

The **corrIntens** value can also be computed using an instantiation of the **pn.RedCorr** class:

```
In [5]: redcorr = pn.RedCorr(E_BV = 0.87, law = 'F99')
```

```
In [6]: line.correctIntens(redcorr) #redcorr is used to compute line.corrIntens
```

The line information is printed using:

```
In [7]: line.printLine()
```

```
Line O3 O3_5007A evaluated as L(5007)
Observed intensity: [ 1.4  1.3]
Observed error: [ 0.  0.]
Corrected intensity: [ 22.58352855  20.97041937]
Corrected error: [ 0.  0.]
```

Most of the times, users will not need to define or manipulate EmissionLine objects, since most of the work on the EmissionLine objects will be performed from the Observation class (read data, extinction correction); see next section.

WARNING: Note that the wavelengths assigned to EmissionLine objects are simply the numerical part of the label:

```
In [9]: Hb1 = pn.EmissionLine(label='H1r_4861A').wave
        print(Hb1)
```

```
4861.0
```

whereas pn.Atom computes them as the difference from energy levels:

```
In [10]: Hb2 = pn.RecAtom('H', 1).getWave(4, 2)
         print(Hb2)
```

```
4861.33165987
```

This can cause small errors when both methods are used simultaneously. For instance, the extinction correction at Hb1 is slightly different from the expected value of 1:

```
In [13]: rc = pn.RedCorr()
         rc.E_BV = 1.34
         rc.law = 'F99'
         rc.getCorrHb(Hb1)
```

```
Out[13]: 1.000410798530041
```

This happens because the ExtCorr uses the precise Hβ value computed from energy levels. If this roundoff error exceeds your tolerance, a possible workaround is forcing the emission line to have exactly the wavelength computed from the energy levels:

```
In [16]: O3w = pn.EmissionLine('O', 3, wave=5007).wave
         print(rc.getCorrHb(O3w))
```

```
0.836202049828
```

```
In [18]: Hb11 = pn.EmissionLine('H', 1, wave=Hb2).wave
         print(rc.getCorrHb(Hb11))
         # This will generate a warning (as the transition is not included in the inventory for the spe
         # but the code won't stop.
```

```
warng EmissionLine: Atom H1 not valid
1.0
```

0.2 The Observation class: reading and dealing with observations

0.2.1 Reading observation from a file

pn.Observation is the class characterizing observation records. An observation record is composed of an object identifier, the observed intensity of one or more emission lines, and, optionally, the dereddened line intensities and the identifier of the extinction law used, and the value of c(Hbeta).

Observations can be initialized by reading data files, which can be organized with different emission lines either in rows or columns (usually, in a survey of many objects with few emission lines emission lines change across columns; and in a high-resolution observation of a small sample of objects lines change across rows).

The following is an example of how to define an observation:

```
In [21]: obs = pn.Observation()
```

```
In [22]: %%writefile observations1.dat
         LINE SMC_24
         S4_10.5m   7.00000
         Ne2_12.8m  8.3000
         Ne3_15.6m 34.10
         S3_18.7m  10.
         O2_3726A  39.700
         O2_3729A  18.600
         Ne3_3869A 18.90
         Ne3_3968A  6.4
         S2_4069A   0.85
         S2_4076A   0.450
         O3_4363A   4.36
         H1r_4861A 100.00
         O3_5007A 435.09
         N2_5755A   0.510000
         S3_6312A   0.76
         O1_6300A   1.69
         O1_6364A   0.54
         N2_6548A   6.840000
         H1r_6563A   3.45
         N2_6584A  19.00
         S2_6716A   1.220000
         S2_6731A   2.180000
         Ar3_7136A  4.91
         O2_7319A+   6.540000
         O2_7330A+   5.17
```

```
Overwriting observations1.dat
```

```
In [23]: obs.readData('observations1.dat', fileFormat='lines_in_rows', err_default=0.05) # fill obs wit
```

```
In [24]: obs.printIntens(returnObs=True)
```

```
S4_10.5m [ 7.]
Ne2_12.8m [ 8.30000019]
Ne3_15.6m [ 34.09999847]
S3_18.7m [ 10.]
O2_3726A [ 39.70000076]
O2_3729A [ 18.60000038]
Ne3_3869A [ 18.89999962]
Ne3_3968A [ 6.4000001]
S2_4069A [ 0.85000002]
S2_4076A [ 0.44999999]
O3_4363A [ 4.36000013]
H1r_4861A [ 100.]
O3_5007A [ 435.08999634]
N2_5755A [ 0.50999999]
S3_6312A [ 0.75999999]
O1_6300A [ 1.69000006]
O1_6364A [ 0.54000002]
N2_6548A [ 6.84000015]
H1r_6563A [ 3.45000005]
```

```
N2_6584A [ 19.]
S2_6716A [ 1.22000003]
S2_6731A [ 2.18000007]
Ar3_7136A [ 4.90999985]
O2_7319A+ [ 6.53999996]
O2_7330A+ [ 5.17000008]
```

```
In [25]: obs.extinction.law = 'CCM89'  # define the extinction law from Cardelli et al.
         obs.correctData()                        # the dereddened data are computed
```

The data can be read by the readData method as above or directly while instantiating the object:

```
In [26]: obs = pn.Observation('observations1.dat', fileFormat='lines_in_rows', corrected=True)
```

The format of the data file from which the emission line intensities are read can be one of three kinds: "lines_in_rows" as above, or "lines_in_cols" like this one:

```
In [27]: %%writefile observations2.dat
         NAME O2_3726A  O2_3726Ae O2_3729A O2_3729Ae
         NGC3132 0.93000   0.05000   0.17224200 0.10
         IC418 1.28000   0.05000   0.09920000 0.05
         M33 0.03100   0.080     0.03100    0.10

Writing observations2.dat
```

```
In [28]: obs2 = pn.Observation('observations2.dat', fileFormat='lines_in_cols', corrected=True)
```

or fileFormat='lines_in_rows_err_cols' (errors labeled "err". Don't name an observation "err"!) like this one:

```
In [29]: %%writefile observations3.dat
         LINE     TT   err  TT2 err TT3 err
         cHbeta   1.2  0.0  1.5 0.2 1.1 0.2
         O3_5007A 1.5  0.15 1.3  .2 1.1 0.1
         H1_6563A 2.89 0.05 1.6 0.3 1.3 0.1
         N2_6584A 1.   0.20 0.3 0.5 1.5 0.1

Writing observations3.dat
```

```
In [31]: #obs3 = pn.Observation('observations3.dat', fileFormat='lines_in_rows_err_cols', corrected=Fal
```

The delimiter between the columns is any sequence of spaces or TAB, but it can be changed using the delimiter parameter. The line names are defined by a label starting with the name of the atom ('O2'), followed by an underscore, followed by a wavelength and ending with a unit ('A' or 'm'). The list of all the lines managed by PyNeb, ordered by atoms, is obtained by entering:

```
In [32]: for atom in pn.LINE_LABEL_LIST:
             print(atom, pn.LINE_LABEL_LIST[atom])

H1r ['1216A', '1026A', '973A', '6563A', '4861A', '4341A', '4102A', '3970A', '3889A', '3835A', '3798A',
He1r ['5876A', '2945A', '3188A', '3614A', '3889A', '3965A', '4026A', '4121A', '4388A', '4438A', '4471A'
He2r ['1640A', '1215A', '1084A', '4686A', '3203A', '6560A', '5411A', '4859A']
3He2 ['3.50c']
Al2 ['2674A', '2670A', '2661A', '1671A', '4451A', '4463A', '4488A', '164.2m', '54.1m', '80.7m']
Ar2 ['7.0m']
Ar3 ['7136A', '7751A', '8036A', '3005A', '3109A', '3154A', '5192A', '9.0m', '6.37m', '21.8m']
Ar4 ['4740A', '4711A', '2868A', '7263A', '7332A', '2854A', '7170A', '7237A', '77.4m', '56.2m']
```

```
Ar5 ['6133A', '6435A', '7005A', '2637A', '2692A', '2786A', '4626A', '1218A', '1229A', '1249A', '1520A',
Ba2 ['4935A', '6497A', '6854A', '4555A', '5854A', '6142A', '2361A', '2668A', '2726A', '4524A', '4900A',
Ba4 ['5697A']
Br3 ['6646A', '6133A', '3714A', '8420A', '9419A', '3498A', '7385A', '8142A', '7.94m', '6.0m']
C1 ['9808A', '9824A', '9850A', '4618A', '4621A', '4627A', '8728A', '2963A', '2965A', '2967A', '4246A',
C2 ['2325A', '2328A', '2323A', '2327A', '2322A', '2325A', '1335A', '1336A', '3131A', '3133A', '3136A',
C3 ['1910A', '1909A', '1907A', '977A', '2000A', '2001A', '2003A', '422.0m', '124.9m', '177.4m']
C4 ['1551A', '1548A', '92.8m']
Ca2 ['7292A', '7324A']
Ca5 ['5309A', '6087A', '6428A', '2280A', '2413A', '2464A', '3998A', '4.16m', '3.05m', '11.5m']
Cl2 ['8579A', '9124A', '9381A', '3586A', '3678A', '3719A', '6162A', '14.4m', '10.0m', '33.3m']
Cl3 ['5538A', '5518A', '3353A', '8500A', '8548A', '3343A', '8434A', '8481A', '151.5m', '108.0m']
Cl4 ['7261A', '7531A', '8046A', '3071A', '3119A', '3204A', '5323A', '1463A', '1474A', '1493A', '1833A',
Fe3 ['4009A', '4659A', '4668A', '4701A', '4734A', '4755A', '5011A', '5085A', '5270A', '4881A', '4925A',
Fe4 ['4491A', '5685A', '5735A', '6740A']
Fe5 ['3783A', '3795A', '3822A', '3891A', '3895A', '3911A', '4071A', '4181A', '4227A']
Fe6 ['3556A', '3929A', '5146A', '5176A', '5278A', '5335A', '5370A', '5424A', '5427A', '5485A', '5631A',
Fe7 ['5159A', '5276A', '5721A', '6087A']
K4 ['6102A', '6796A', '7109A', '2594A', '2711A', '2760A', '4511A', '6.0m', '4.3m', '15.4m']
K5 ['4163A', '4123A', '2514A', '6349A', '6446A', '2495A', '6222A', '6316A', '42.2m', '31.0m']
K6 ['5602A', '6229A']
Kr3 ['6827A', '9902A', '3022A', '3504A', '3600A', '5423A', '2.2m', '1.88m', '13.1m', '1.07m']
Kr4 ['5868A', '5346A', '3219A', '7131A', '8091A', '2993A', '6108A', '6798A', '6.0m', '4.26m']
Kr5 ['5069A', '6256A', '8243A', '2550A', '2819A', '3163A', '5132A', '2.67m', '1.32m', '2.6m']
Mg4 ['4.5m']
Mg5 ['2783A', '2929A', '2992A', '1294A', '1325A', '1338A', '2418A', '5.6m', '3.96m', '13.5m']
Mg7 ['2441A', '2509A', '2629A', '1174A', '1190A', '1216A', '2261A', '943A', '953A', '970A', '1537A', '4
N1 ['5200A', '5198A', '3467A', '3466A']
N2 ['6527A', '6548A', '6584A', '3058A', '3063A', '3071A', '5755A', '2137A', '2139A', '2143A', '3177A',
N3 ['1749A', '1754A', '1747A', '1752A', '1744A', '1750A', '990A', '992A', '2280A', '2284A', '2288A', '2
N4 ['1488A', '1487A', '1483A', '765A', '1575A', '1576A', '1580A', '158.4m', '48.3m', '69.4m']
Na3 ['7.3m']
Na4 ['3242A', '3362A', '3416A', '1504A', '1529A', '1540A', '2804A', '9.0m', '6.34m', '21.3m']
Na6 ['2816A', '2872A', '2972A', '1343A', '1356A', '1378A', '2569A', '14.39m', '5.4m', '8.6m']
Ne2 ['12.8m']
Ne3 ['3869A', '3968A', '4012A', '1794A', '1815A', '1824A', '3343A', '15.6m', '10.9m', '36.0m']
Ne4 ['2425A', '2422A', '1602A', '4716A', '4726A', '1601A', '4714A', '4724A', '224.9m', '1579.3m']
Ne5 ['3300A', '3346A', '3426A', '1565A', '1575A', '1592A', '2973A', '1132A', '1137A', '1146A', '1721A',
Ne6 ['997A', '1010A', '993A', '1006A', '986A', '999A', '559A', '563A', '1271A', '1278A', '1289A', '559A
Ni3 ['7890A', '8500A', '6000A', '6401A', '6534A', '6682A', '6797A', '7125A', '6946A']
O1 ['6300A', '6364A', '6392A', '2959A', '2973A', '2979A', '5577A', '63.2m', '44.1m', '145.5m']
O2 ['3729A', '3726A', '2470A', '7319A', '7320A', '7330A', '7331A', '2470A', '834A', '1075A', '1260A', '8
O3 ['4931A', '4959A', '5007A', '2315A', '2321A', '2331A', '4363A', '1658A', '1661A', '1666A', '2497A',
O4 ['1400A', '1407A', '1397A', '1405A', '1394A', '1401A', '788A', '1801A', '1806A', '1812A', '608A', '6
O5 ['1220A', '1218A', '1214A', '630A', '1301A', '1303A', '1309A', '73.5m', '22.6m', '32.6m']
Rb4 ['5760A', '9009A', '9604A', '2603A', '3110A', '3178A', '4750A', '1.6m', '1.44m', '14.5m']
Rb5 ['5364A', '4742A', '2873A', '6188A', '7290A', '2609A', '5080A', '5800A', '4.1m', '2.84m']
Rb6 ['4210A', '5373A', '7220A', '2212A', '2495A', '2832A', '4660A', '1.95m', '1.01m', '2.1m']
S2 ['6731A', '6716A', '4076A', '4069A', '1260A', '1549A', '1550A', '1823A', '1824A', '1254A', '1541A',
S3 ['8829A', '9069A', '9531A', '3681A', '3722A', '3798A', '6312A', '33.5m', '12.0m', '18.7m']
S4 ['1405A', '1424A', '1398A', '1417A', '1387A', '1406A', '10.5m', '29.0m', '11.2m', '18.3m']
Se3 ['7671A', '8854A', '3516A', '3746A', '4082A', '6493A', '5.74m', '2.54m', '4.55m', '1.1m']
Se4 ['2.28m']
Si2 ['2335A', '2351A', '2329A', '2345A', '2320A', '1808A', '1817A', '8007A', '8077A', '8193A', '7997A',
```

```
Si3 ['1897A', '1892A', '1883A', '1206A', '3315A', '3329A', '3359A', '77.7m', '25.7m', '38.2m']
Xe3 ['5847A', '2769A', '3574A', '3800A', '5261A', '1.23m', '1.02m', '6.0m', '1.11m', '1.37m']
Xe4 ['7535A', '5709A', '3566A', '6769A', '9498A', '2804A', '4467A', '5511A', '2.36m', '1.31m']
Xe6 ['6409A']
```

The presence of a trailing "e" at the end of the label points to the error associated to the line. The error is considered to be relative to the intensity (i.e., 0.05 means 5% of the intensity), unless the parameter errIsRelative is set to False. A common value for all the errors can be defined by the parameter **err_default** (0.10 is the default value).

0.2.2 Extinction correction in Observation class

Once the data have been read, they have to be corrected from extinction. An instantiation of **RedCorr()** is available inside the **Observation** object as **obs.extinction**.

If the data file contains **cHbeta** or **E(B-V)** alongside of line labels, the corresponding information on extinction is transmitted to the extinction correction object. Otherwise, the extinction parameters must be set manually; for example:

```
In [33]: obs = pn.Observation('observations1.dat', fileFormat='lines_in_rows', corrected=True)
         obs.extinction.cHbeta = 1.2
         obs.extinction.E_BV = 0.34
```

An extinction law has to be specified in either case:

```
In [34]: obs.extinction.law = 'F99'
```

To correct all the lines at once:

```
In [36]: obs.correctData()
```

```
In [37]: obs.printIntens(returnObs=True)
```

```
S4_10.5m [ 7.]
Ne2_12.8m [ 8.30000019]
Ne3_15.6m [ 34.09999847]
S3_18.7m [ 10.]
O2_3726A [ 39.70000076]
O2_3729A [ 18.60000038]
Ne3_3869A [ 18.89999962]
Ne3_3968A [ 6.4000001]
S2_4069A [ 0.85000002]
S2_4076A [ 0.44999999]
O3_4363A [ 4.36000013]
H1r_4861A [ 100.]
O3_5007A [ 435.08999634]
N2_5755A [ 0.50999999]
S3_6312A [ 0.75999999]
O1_6300A [ 1.69000006]
O1_6364A [ 0.54000002]
N2_6548A [ 6.84000015]
H1r_6563A [ 3.45000005]
N2_6584A [ 19.]
S2_6716A [ 1.22000003]
S2_6731A [ 2.18000007]
Ar3_7136A [ 4.90999985]
O2_7319A+ [ 6.53999996]
O2_7330A+ [ 5.17000008]
```

```
In [38]: obs.printIntens()
```

```
S4_10.5m [ 7.11975803]
Ne2_12.8m [ 8.41496764]
Ne3_15.6m [ 34.48348489]
S3_18.7m [ 10.09307839]
O2_3726A [ 171.24206763]
O2_3729A [ 80.15612017]
Ne3_3869A [ 78.13399959]
Ne3_3968A [ 25.71732799]
S2_4069A [ 3.32014218]
S2_4076A [ 1.75430783]
O3_4363A [ 15.7036518]
H1r_4861A [ 310.25435475]
O3_5007A [ 1289.85131043]
N2_5755A [ 1.24390974]
S3_6312A [ 1.66198348]
O1_6300A [ 3.70383903]
O1_6364A [ 1.16980719]
N2_6548A [ 14.34619111]
H1r_6563A [ 7.21747442]
N2_6584A [ 39.60657366]
S2_6716A [ 2.48780908]
S2_6731A [ 4.43455095]
Ar3_7136A [ 9.38391466]
O2_7319A+ [ 12.17965989]
O2_7330A+ [ 9.61370718]
```

If you want the corrected line intensities to be normalized to a given wavelength, use the following:

```
In [39]: obs.correctData(normWave=4861.)
```

The extinction correction can be determined by comparing the observed values to a theoretical ratio, as in the following:

```
In [41]: obs.printIntens()
```

```
S4_10.5m [ 2.29481325]
Ne2_12.8m [ 2.71228027]
Ne3_15.6m [ 11.11458529]
S3_18.7m [ 3.25316252]
O2_3726A [ 55.19408995]
O2_3729A [ 25.83561486]
Ne3_3869A [ 25.18385266]
Ne3_3968A [ 8.28911105]
S2_4069A [ 1.07013556]
S2_4076A [ 0.56544181]
O3_4363A [ 5.06154114]
H1r_4861A [ 100.]
O3_5007A [ 415.73995358]
N2_5755A [ 0.40093224]
S3_6312A [ 0.53568417]
O1_6300A [ 1.19380727]
O1_6364A [ 0.37704779]
N2_6548A [ 4.62400959]
```

```
H1r_6563A [ 2.32630882]
N2_6584A [ 12.76583972]
S2_6716A [ 0.80186113]
S2_6731A [ 1.42932755]
Ar3_7136A [ 3.02458757]
O2_7319A+ [ 3.92570151]
O2_7330A+ [ 3.09865342]
```

```
In [42]: obs.def_EBV(label1="H1r_6563A", label2="H1r_4861A", r_theo=2.85)
         print(obs.extinction.E_BV)
         obs.correctData(normWave=4861.)
```

```
[-3.80825098]
```

```
In [43]: obs.printIntens()
```

```
S4_10.5m [ 1862873.3321508]
Ne2_12.8m [ 2289624.63614709]
Ne3_15.6m [ 9681842.82513927]
S3_18.7m [ 2900918.21157661]
O2_3726A [ 0.99069915]
O2_3729A [ 0.46892362]
Ne3_3869A [ 0.75888083]
Ne3_3968A [ 0.35320246]
S2_4069A [ 0.06443513]
S2_4076A [ 0.03486379]
O3_4363A [ 0.81982369]
H1r_4861A [ 100.]
O3_5007A [ 724.22886843]
N2_5755A [ 7.55158652]
S3_6312A [ 38.21722558]
O1_6300A [ 82.92131172]
O1_6364A [ 30.17724087]
N2_6548A [ 549.03539807]
H1r_6563A [ 285.]
N2_6584A [ 1633.68483422]
S2_6716A [ 134.21899293]
S2_6731A [ 246.50683125]
Ar3_7136A [ 1116.57610378]
O2_7319A+ [ 1987.64418295]
O2_7330A+ [ 1598.12655872]
```

By default, this method prints out the corrected intensities. To print the observed intensities, use the **returnObs=True** parameter.

The method **getSortedLines** returns the lines sorted in alphabetical order according to either the emitting atoms (default) or the wavelength (using the **crit='wave'** parameter):

```
In [45]: for line in obs.getSortedLines():
             print(line.label, line.corrIntens[0])
```

```
Ar3_7136A 1116.57610378
H1r_4861A 100.0
H1r_6563A 285.0
N2_5755A 7.55158651669
N2_6548A 549.03539807
N2_6584A 1633.68483422
```

```
Ne2_12.8m 2289624.63615
Ne3_15.6m 9681842.82514
Ne3_3869A 0.758880826821
Ne3_3968A 0.35320245771
O1_6300A 82.921311722
O1_6364A 30.1772408716
O2_3726A 0.990699145083
O2_3729A 0.468923618506
O2_7319A+ 1987.64418295
O2_7330A+ 1598.12655872
O3_4363A 0.819823693016
O3_5007A 724.228868428
S2_4069A 0.0644351280132
S2_4076A 0.0348637946044
S2_6716A 134.218992935
S2_6731A 246.506831252
S3_18.7m 2900918.21158
S3_6312A 38.2172255813
S4_10.5m 1862873.33215
```

The following method, which gives the list of all the atoms implied in the observed emission lines, will be useful later:

```
In [46]: atomList = obs.getUniqueAtoms()
```

```
In [47]: atomList
```

```
Out[47]: array(['Ar3', 'H1r', 'N2', 'Ne2', 'Ne3', 'O1', 'O2', 'O3', 'S2', 'S3', 'S4'],
               dtype='<U3')
```

0.2.3 Adding observations and lines

Once an **Observation** object is instantiated, you can add a new observation (corresponding, e.g., to a new object or a new fiber) by using:

```
In [48]: obs.addObs('test', np.random.rand(25))
```

where 'test' is the name of the new observation. The new observation must have the same size of **obs**, that is, it must contain **obs.n_lines** lines.

```
In [50]: obs.printIntens()
```

```
S4_10.5m [  1.86287333e+06   8.17609704e-01]
Ne2_12.8m [  2.28962464e+06   4.84209585e-01]
Ne3_15.6m [  9.68184283e+06   5.69878150e-01]
S3_18.7m [  2.90091821e+06   1.69287051e-01]
O2_3726A [ 0.99069915  0.27622725]
O2_3729A [ 0.46892362  0.93353075]
Ne3_3869A [ 0.75888083  0.76410908]
Ne3_3968A [ 0.35320246  0.75877006]
S2_4069A [ 0.06443513  0.43461895]
S2_4076A [ 0.03486379  0.01242744]
O3_4363A [ 0.81982369  0.87784109]
H1r_4861A [ 100.          0.10581813]
O3_5007A [  7.24228868e+02   2.66732250e-02]
N2_5755A [ 7.55158652  0.26535309]
```

```
S3_6312A [ 38.21722558   0.85286266]
O1_6300A [ 82.92131172   0.16316097]
O1_6364A [ 30.17724087   0.30763595]
N2_6548A [  5.49035398e+02   3.34564023e-01]
H1r_6563A [ 285.           0.60962985]
N2_6584A [  1.63368483e+03   7.55564203e-01]
S2_6716A [  1.34218993e+02   1.14665408e-02]
S2_6731A [ 246.50683125    0.55873937]
Ar3_7136A [  1.11657610e+03   5.42686282e-01]
O2_7319A+ [  1.98764418e+03   8.20687911e-01]
O2_7330A+ [  1.59812656e+03   9.84102059e-01]
```

```
In [51]: line = pn.EmissionLine(label='Cl3_5518A', obsIntens=[3.5, 2.5])
         obs.addLine(line)
```

```
In [52]: obs.printIntens()
```

```
S4_10.5m [  1.86287333e+06   8.17609704e-01]
Ne2_12.8m [  2.28962464e+06   4.84209585e-01]
Ne3_15.6m [  9.68184283e+06   5.69878150e-01]
S3_18.7m [  2.90091821e+06   1.69287051e-01]
O2_3726A [ 0.99069915  0.27622725]
O2_3729A [ 0.46892362  0.93353075]
Ne3_3869A [ 0.75888083  0.76410908]
Ne3_3968A [ 0.35320246  0.75877006]
S2_4069A [ 0.06443513  0.43461895]
S2_4076A [ 0.03486379  0.01242744]
O3_4363A [ 0.81982369  0.87784109]
H1r_4861A [ 100.           0.10581813]
O3_5007A [  7.24228868e+02   2.66732250e-02]
N2_5755A [ 7.55158652  0.26535309]
S3_6312A [ 38.21722558   0.85286266]
O1_6300A [ 82.92131172   0.16316097]
O1_6364A [ 30.17724087   0.30763595]
N2_6548A [  5.49035398e+02   3.34564023e-01]
H1r_6563A [ 285.           0.60962985]
N2_6584A [  1.63368483e+03   7.55564203e-01]
S2_6716A [  1.34218993e+02   1.14665408e-02]
S2_6731A [ 246.50683125    0.55873937]
Ar3_7136A [  1.11657610e+03   5.42686282e-01]
O2_7319A+ [  1.98764418e+03   8.20687911e-01]
O2_7330A+ [  1.59812656e+03   9.84102059e-01]
Cl3_5518A [  8.85845356e-05   6.32746683e-05]
```

0.2.4 Getting line intensities

You can extract the line intensities from an **Observation** object by, for example:

```
In [53]: obs.names
```

```
Out[53]: ['SMC_24', 'test']
```

```
In [54]: obs.getIntens(obsName='SMC_24')
```

```
Out[54]: {'Ar3_7136A': 1116.5761037772124,
          'Cl3_5518A': 8.8584535640522582e-05,
```

```
 'H1r_4861A': 100.0,
 'H1r_6563A': 285.0000000000004,
 'N2_5755A': 7.5515865166910849,
 'N2_6548A': 549.03539806966057,
 'N2_6584A': 1633.6848342208623,
 'Ne2_12.8m': 2289624.6361470865,
 'Ne3_15.6m': 9681842.8251392711,
 'Ne3_3869A': 0.75888082682114033,
 'Ne3_3968A': 0.35320245770990177,
 'O1_6300A': 82.921311721997114,
 'O1_6364A': 30.177240871552701,
 'O2_3726A': 0.99069914508263235,
 'O2_3729A': 0.46892361850581255,
 'O2_7319A+': 1987.6441829484479,
 'O2_7330A+': 1598.1265587226821,
 'O3_4363A': 0.81982369301604263,
 'O3_5007A': 724.22886842764478,
 'S2_4069A': 0.064435128013150128,
 'S2_4076A': 0.034863794604422474,
 'S2_6716A': 134.21899293495827,
 'S2_6731A': 246.50683125182974,
 'S3_18.7m': 2900918.2115766057,
 'S3_6312A': 38.217225581304128,
 'S4_10.5m': 1862873.3321507953}
```

```
In [55]: obs.getIntens()['O2_7330A+']
```

```
Out[55]: array([  1.59812656e+03,   9.84102059e-01])
```

0.3 Using Observation to determine ionic abundances

Once the electron temperature and density are determined, it is easy to obtain the ionic abundances from a set of emission lines included in an **Observation** object:

```
In [56]: obs = pn.Observation()
         obs.readData('observations1.dat', fileFormat='lines_in_rows', err_default=0.05) # fill obs wit
         obs.def_EBV(label1="H1r_6563A", label2="H1r_4861A", r_theo=2.85)
         obs.correctData(normWave=4861.)
         Te = [10000.]
         Ne = [1e3]
         # Define a dictionary to hold all the Atom objects needed
         all_atoms = pn.getAtomDict(atom_list=obs.getUniqueAtoms())
         # define a dictionary to store the abundances
         ab_dict = {}
         # we  use the following lines to determine the ionic abundances
         ab_labels = ['N2_6584A', 'O2_3726A', 'O3_5007A', 'S2_6716A',
                      'S3_6312A', 'Ar3_7136A', 'Ne3_3869A']
         for line in obs.getSortedLines():
             if line.label in ab_labels:
                 ab = all_atoms[line.atom].getIonAbundance(line.corrIntens, Te, Ne,
                                                           to_eval=line.to_eval, Hbeta=100)
                 ab_dict[line.atom] = ab
```

```
warng _ManageAtomicData: rec data not available for Ar3
warng _ManageAtomicData: atom data not available for H1
```

```
warng _ManageAtomicData: coll data not available for H1
warng _ManageAtomicData: rec data not available for Ne2
warng _ManageAtomicData: rec data not available for Ne3
warng _ManageAtomicData: rec data not available for S2
warng _ManageAtomicData: rec data not available for S3
warng _ManageAtomicData: rec data not available for S4
```

```
In [57]: ab_dict
```

```
Out[57]: {'Ar3': array([  4.06352863e-07]),
          'N2': array([  3.91252886e-06]),
          'Ne3': array([  2.05860787e-05]),
          'O2': array([  3.10454298e-05]),
          'O3': array([ 0.00015164]),
          'S2': array([  8.23174570e-08]),
          'S3': array([  1.69309386e-06])}
```