A&A 573, A42 (2015)
DOI: 10.1051/0004-6361/201323152

Astronomy & Astrophysics

PyNeb: a new tool for analyzing emission lines

I. Code description and validation of results

V. Luridiana[1,2], C. Morisset[3], and R. A. Shaw[4]

[1] Instituto de Astrofísica de Canarias, c/ Vía Láctea s/n, 38205 La Laguna, Tenerife, Spain
e-mail: vale@iac.es
[2] Departamento de Astrofísica, Universidad de La Laguna, 38206 La Laguna, Tenerife, Spain
[3] Instituto de Astronomía, Universidad Nacional Autónoma de México, Apdo. Postal 70264, 04510 México D.F., Mexico
[4] National Optical Astronomy Observatory, Tucson, AZ 85719, USA

Received 29 November 2013 / Accepted 29 September 2014

ABSTRACT

Analysis of emission lines in gaseous nebulae yields direct measures of physical conditions and chemical abundances and is the cornerstone of nebular astrophysics. Although the physical problem is conceptually simple, its practical complexity can be overwhelming since the amount of data to be analyzed steadily increases; furthermore, results depend crucially on the input atomic data, whose determination also improves each year. To address these challenges we created PyNeb, an innovative code for analyzing emission lines. PyNeb computes physical conditions and ionic and elemental abundances and produces both theoretical and observational diagnostic plots. It is designed to be portable, modular, and largely customizable in aspects such as the atomic data used, the format of the observational data to be analyzed, and the graphical output. It gives full access to the intermediate quantities of the calculation, making it possible to write scripts tailored to the specific type of analysis one wants to carry out. In the case of collisionally excited lines, PyNeb works by solving the equilibrium equations for an n-level atom; in the case of recombination lines, it works by interpolation in emissivity tables. The code offers a choice of extinction laws and ionization correction factors, which can be complemented by user-provided recipes. It is entirely written in the python programming language and uses standard python libraries. It is fully vectorized, making it apt for analyzing huge amounts of data. The code is stable and has been benchmarked against IRAF/NEBULAR. It is public, fully documented, and has already been satisfactorily used in a number of published papers.

Key words. methods: numerical – atomic data – H II regions – planetary nebulae: general – ISM: abundances

1. Introduction

The intensity of emission lines in the spectra of ionized nebulae carries valuable information on the physical conditions and chemical abundances of these objects. The physics of line formation in the relevant regime has been described in several seminal papers published throughout the past century (Menzel 1937; Baker & Menzel 1938; Baker et al. 1938; Burgess 1958; Osterbrock 1989) and can be summarized in a handful of equations. They are conceptually simple, but cannot be solved analytically, so a numerical code is necessary to do the task. Furthermore, they depend on a number of parameters – the atomic data – which are only approximately known and whose determination improves steadily. A suitable tool to carry out this task must therefore be prepared to include such advances.

This paper and a forthcoming one (Morisset et al., in prep., hereinafter Paper II) describe PyNeb, a new python package for the analysis of emission line spectra in ionized nebulae. PyNeb[1] is the latest in a lineage of tools dedicated to addressing this problem. The first of these codes, FIVEL (De Robertis et al. 1987), is a FORTRAN program that solves the equilibrium equations for a five-level atom model to determine level populations and line emissivities, which are then used to apply simple temperature and density diagnostics. The heir of FIVEL, NEBULAR (Shaw & Dufour 1995; Shaw et al. 1998), is an IRAF/STSDAS package that extends FIVEL's basic functionality with a small suite of tasks, such as determining ionic abundances and plotting diagnostic diagrams. NEBULAR also includes a simple tool to compute abundances from observed emission line intensities for over three dozen supported ions in a three-zone nebula (low, medium, and high-ionization) and a dereddening tool with a choice of interstellar extinction laws. One major asset of NEBULAR over FIVEL is that atomic data are no longer hardwired into the code; instead, they are read at run time from external files, which can therefore be conveniently updated without recompiling the source code. Both atomic data and observed emission line intensities are stored in FITS format to provide a standardized container for structured data with associated metadata. Another asset of NEBULAR over FIVEL is that, for a given ion, users may employ any recognized line ratio by creating an expression for the desired transitions. Finally, a simple web interface was created for two of the simpler tasks, and it has proven to be one of its most popular uses.

In its almost twenty years of life, NEBULAR has enjoyed wide use in the research community for analyzing physical conditions and chemical abundances in a variety of astrophysical contexts, including active galaxies, H II regions, and planetary nebulae (PNe). It has also been widely used as a starting point by people who model gaseous nebulae with photoionization codes. The decision to recast NEBULAR in a modern and very popular programming language, python, was motivated by the desire to expose the internal functionality and data model to users through application interfaces (APIs), to provide a richer set of visualization and analysis tools, to provide easier implementation options

[1] PyNeb is available at http://www.iac.es/proyecto/PyNeb

for GUI and web-based user interfaces, and to make it simple for developers or advanced users to extend the original functionality of the code. The result of this effort is PyNeb, a code that, in addition to the functionality of NEBULAR, provides several new analysis features, a vast range of visualization tools, and full access to the intermediate quantities of the calculation. The latter feature makes it possible to write scripts tailored to the specific type of analysis one wants to carry out.

In this paper, we give an overview of the current version of PyNeb (v. 1.0.1; an earlier version of the code was described in Luridiana et al. 2012). Specifically, Sect. 2 briefly reviews the physics underlying PyNeb; Sect. 3 provides an overview of the package functionality; Sect. 4 summarizes the validation of the code, consisting of a thorough comparison with NEBULAR; Sect. 5 addresses practical aspects of the code, such as software requirements and documentation; finally, Sect. 6 recapitulates the present status of PyNeb and describes possible directions for expanding the code in the future. Two sample scripts illustrating how PyNeb can be used to solve real physical problems are included in the Appendix. Paper II will be entirely devoted to describing the new atomic database.

2. Underlying physics

In photoionized objects in low-density environments, atoms are far away from LTE, and the only level that is appreciably populated is the ground state (with the only exception of excited metastable levels, such as the 2^3S of He I, or excited levels of extremely low energy). The spectra emitted in nebular conditions are dominated by permitted lines of H and He ions and forbidden lines of ions of abundant metals, such as N, O, and Ne. Although in principle all line-formation processes are possible for all chemical species, in practice only some of them are effective for each given ion; thus, H and He lines mainly form in the downward cascade following recombination of H^+, He^+, and He^{++}, while the strongest metal lines mainly form in the radiative de-excitation following collisional excitation from the ground state. In both cases, the emissivity of a line is proportional to the product of the population of the upper level n_u and the line transition probability A_{ul}. In the following, we summarize the main features of both types of lines.

2.1. Hydrogen and helium: recombination lines

The largest contribution to H and He lines in ionized nebulae comes from recombination of H^+, He^+, and He^{++}. The recombination cascade begins when a free electron recombines to an excited level and starts to fall through the atom's excited levels. The downward path is determined by the branching ratios from each level to the lower lying ones, possibly modulated by horizontal redistribution among sublevels of the same main level (l-mixing). Hummer & Storey (1987) and Storey & Hummer (1995) performed detailed computations of the recombination cascade of H-like ions and made them available in tabular form for a wide array of temperature and density values.

Although recombination is the most effective line-formation process for the ions of these two elements, the excited levels of H and He can also be populated through upward collisional excitations and continuum pumping. As anticipated above, both processes are mostly negligible in nebular conditions: the former is negligible because the energy gap to be crossed is very large compared to typical free electron energies (the only notable exceptions being collisional excitation from the H I ground level in very hot objects (Davidson & Kinman 1985; Luridiana et al. 2003) and of the He I 2^3S metastable level in medium- to high-density objects); and the latter is negligible because of the weakness of the stellar ionizing spectra at the relevant energy values (Ferland 1999; Luridiana et al. 2009).

2.2. Heavy elements: collisional lines

In contrast to H and He, several astrophysically common metal ions have levels lying just a few eV above the ground level that be easily populated through collisions with free electrons. Such a mechanism is progressively less efficient for higher lying levels, so that only the lowest levels are appreciably populated through this mechanism. As a consequence, only the first few levels need to be considered, and the atom can be represented as a simple n-level system, with the equilibrium level populations obeying the following set of equations:

$$\sum_{j\neq i} N_e n_j q_{ji} + \sum_{j>i} n_j A_{ji} = \sum_{j\neq i} N_e n_i q_{ij} + \sum_{j<i} n_i A_{ij}, \qquad (i = 1, \ldots n_{max}) \tag{1}$$

and:

$$\sum_i n_i = n_{tot}, \tag{2}$$

where N_e is the electron density, n_{tot} the total density of the ion, n_i the density of ions with an electron on level i, $N_e n_j q_{ji}$ is the rate of collisional excitation (if $j < i$) or de-excitation (if $j > i$), and A_{ji} ($j > i$) is the transition probability from level j to i. Densities are measured in cm^{-3} and transition probabilities in s^{-1}. The number of levels considered n is five or six for most ions, but the current database contains atoms with as few as two or as many as 34 levels. The rates of collisional de-excitation and excitation are derived from the effective collision strengths Υ through the following equations:

$$q_{ji} = \frac{8.629 \times 10^{-6}}{g_j} \frac{\Upsilon(T_e)_{jk}}{T_e^{1/2}},$$

and

$$q_{ij} = \frac{g_j}{g_i} q_{ji}\, e^{-\Delta E_{ij}/k_b T_e},$$

where $j > i$, g_i and g_j are the statistical weights of levels i and j, respectively, and $\exp(-\Delta E_{ij}/k_b T_e)$ is the Boltzmann factor (see Osterbrock & Ferland 2006, and references therein). The Υs are the result of atomic physics computations and are generally available in tabular form as a function of electron temperature.

2.3. Plasma diagnostics

Equations (1) and (2) represent a set of $n_{tot} + 1$ coupled equations in the unknowns n_j, which, for any given N_e, T_e combination, can be solved for the relative level populations n_j/n_{tot}. Once the populations are known, the line emissivities are computed as

$$\epsilon_{ji} = n_j A_{ji} h\nu_{ji}. \tag{3}$$

A ratio of two lines emitted by the same ion depends on N_e and T_e, but it does not depend on the total ion abundance n_{tot}, which cancels out. Since for cospatial ions the emissivity ratio is equal to the intensity ratio, an observed intensity ratio can be used to determine either of the two quantities given the other.

2.4. Ionic and elemental abundances

Once the physical conditions are known, Eq. (3) can be manipulated to compute the ionic abundances of observed ions relative to H^+:

$$\frac{n(X^i)}{n(H^+)} = \frac{I(\lambda)}{I(H\beta)} \frac{\epsilon(H\beta)}{\epsilon(\lambda)}, \tag{4}$$

where X^i is an ion emitting a line of wavelength λ. If all the expected ions of a given element are observed, the elemental abundance is trivially given by the sum of the ionic abundances:

$$\frac{n(X)}{n(H)} = \sum_i \frac{n(X^i)}{n(H^+)}. \tag{5}$$

In most cases, however, not all relevant ions are observed and the incomplete summation must be corrected for unseen ions, using an ionization correction factor, or ICF:

$$\frac{n(X)}{n(H)} = \sum_{\bar{\imath}} \frac{n(X^{\bar{\imath}})}{n(H^+)} \times \mathrm{ICF}, \tag{6}$$

where the index $\bar{\imath}$ now extends only over a subset of ions.

The ICFs incorporate some form of knowledge or expectation on the ionization structure of the nebula, based either on the results of photoionization models or on comparisons between the ionization potentials of different ions. In the literature one can find many such expressions, most of which have been devised for a specific kind of object (e.g., PNe, H II regions, etc.) and should therefore not be applied to objects of a different kind.

Table 1. Supported ions in PyNeb.

Atom	Ion	Ground state	Atom	Ion	Ground state
Al	II	s^2	N	I	p^3
Ar	II	p^5		II	p^2
	III	p^4		III	p^1
	IV	p^3		IV	s^2
	V	p^2	Na	IV	p^4
Ba	II	s^1		VI	p^2
	IV	p^5	Ne	II	p^5
Br	III	p^3		III	p^4
	IV	p^2		IV	p^3
C	I	p^2		V	p^2
	II	p^1		VI	p^1
	III	s^2	O	I	p^4
	IV	s^1		II	p^3
Ca	V	p^4		III	p^2
Cl	II	p^4		IV	p^1
	III	p^3		V	s^2
	IV	p^2	Rb	IV	p^4
Fe	III	s^2		V	p^3
K	IV	p^4	S	II	p^3
	V	p^3		III	p^2
Kr	III	p^4		IV	p^1
	IV	p^3	Se	III	p^2
	V	p^2		IV	p^1
Mg	V	p^4	Si	II	p^1
	VII	p^2		III	s^2
Xe	III	p^4			
	IV	p^3			

3. Overview of the code

Fundamentally, PyNeb is a toolbox meant to enable nebular analysis using direct methods and to visualize the results. Its key functionality is to solve for line emissivities, determine electron temperature and density given observed diagnostic line ratios, and compute (within some simplifying assumptions) ionic and total abundances for a large number of elements that are typically observed in gaseous nebulae. PyNeb offers a variety of methods, visualization tools, and programming abstractions to make this easy for a user. It is also designed to be easy for end-users with modest knowledge of scientific programming and the python language to update, extend, and customize PyNeb for particular science objectives by, for example, building scripts to determine abundances from observed emission lines, using the core PyNeb classes to build new software, or developing a new user interface. Users can also add new or alternate atomic data. The list of currently supported ions is given in Table 1.

The key concepts when using PyNeb are the internal model of the ion, which is embodied in the Atom and the RecAtom classes; the representation of a set of observed UV/Optical/IR emission lines, which is embodied in the Observation class; the representation of the plasma diagnostic ratios, embodied in the Diagnostics class; and the representation of ionization correction factors, embodied in the ICF class. The available methods for these classes, plus some ancillary classes, are the means by which the data can be visualized and the analysis carried out. The methods are intended to support a variety of workflows, from simple computations of diagnostics to visualizing the supporting atomic data, to complete abundance analyses. Beyond that there is a Stellar class for the special case of computing the effective temperature of a PN central star (via the Zanstra technique), as well as its luminosity.

New users may find studying the example scripts (available from the web page or on request through the forum; see Sect. 5.2) very helpful for worked examples of PyNeb capabilities. Each script may be executed from the host command line or imported into an active python session. The following sections provide a brief tour and highlight important scientific functionality, as well as details of key methods and their output.

3.1. PyNeb core functionality

The main PyNeb's classes and methods are represented in Fig. 1, which also schematically describes how they interact with each other and the user. In the following sections, we describe the scope of the main classes. The syntax will not be described in detail, since it is fully treated in the code documentation (see Sect. 5.2).

PyNeb distinguishes between atoms emitting collisionally-excited lines (called here "collisional atoms") and atoms emitting recombination lines ("recombination atoms"). Both are represented as n-level systems and share the same basic syntax, but they are encoded in different classes and their internal representation is different in either case, as detailed in the following sections.

3.2. The model collisional atom: the Atom class

The critical features of the physical atom relevant for our scope are encapsulated, for the collisional case, in the `Atom` data

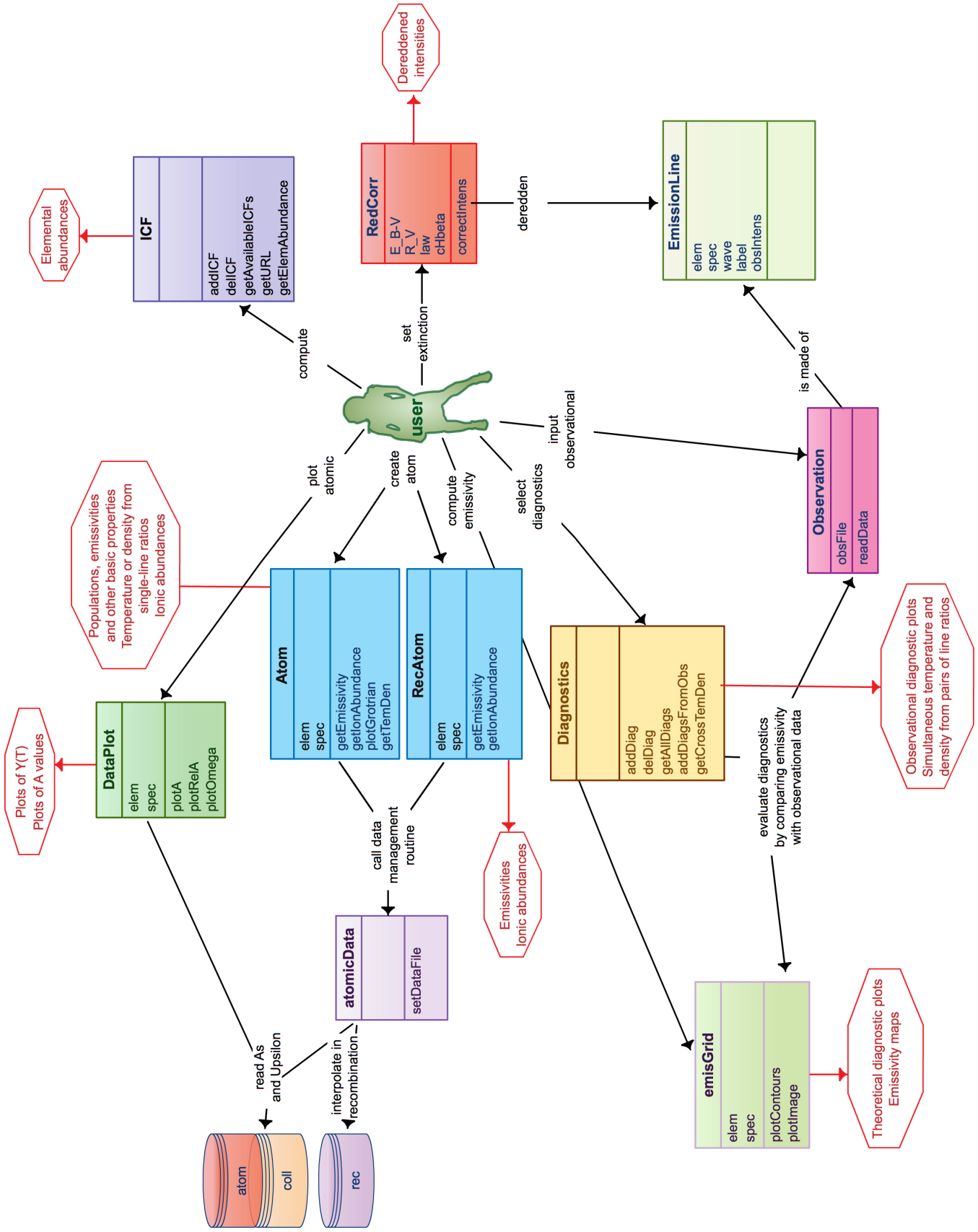


Fig. 1. Schematic view of PyNeb's structure in terms of its classes (rectangular boxes) and output quantities (red octagons). The first slot of each box contains the class name, the second the class arguments, and the third the class methods. The description is abridged and only meant for illustrative purposes.

structure. For example, the following command, defining an [O III] atom:

```
o3 = pn.Atom('O', 3)
```

creates a data structure capable of holding the A and Υ arrays and fills it with the actual [O III] data read from the database. The resulting object is also equipped with the calculus machinery necessary to: solve the system of Eqs. (1) and (2) to compute the level populations; apply Eq. (3) to calculate the emissivities; derive the ion abundance from Eq. (4); and compute or output several other standard quantities.

From the above, it follows that the number of levels of a particular ion is not intrinsic to the ion but instead depends on the data used. For most ions, different data sets are available, corresponding to different atomic data calculations, and changing the data set may possibly imply a change in the number of levels and, almost inevitably, in the result of calculations too.

One asset of PyNeb over similar programs is that all the atomic properties computed by PyNeb can be retrieved through manipulations of this Atom object; for example:

```
o3.getPopulations(tem, den)
o3.getCritDensity(tem, den)
```

return the populations and critical densities, respectively, at the specified physical conditions $T_e = tem$ and $N_e = den$. Properties made accessible by PyNeb include the name, the spectrum, the level energies, the transition probabilities, the set of Υs at a specified temperature, the Grotrian diagram, and many others. The complete inventory can be displayed and explored using standard python commands.

Atom objects are also equipped with basic functionality for determining physical conditions, i.e. the `Atom.getTemDen` method:

```
o3.getTemDen(int_ratio=0.01, den=1000,
             wave1=4363, wave2=5007),
```

which returns the electronic temperature such that [O III] λ4363/λ5007 = 0.01 at N_e = 1000 cm^{-3}.

3.3. Atoms emitting recombination lines: the RecAtom class

In the case of recombination lines, PyNeb computes the emissivity of a given line by either interpolating in tables or using a fitting function. This is accomplished by the `RecAtom class`: for example, the H I atom (or, to be more precise, the recombination spectrum of H I) is instantiated through the command

```
h1=pn.RecAtom(elem='H', spec=1).
```

Recombination atoms share many of the features of collisional atoms and follow the same syntax: for example, the `h1` object defined above can be explored with commands such as

```
h1.name
h1.getEmissivity(tem=10000, den=1000,
                 label='4_2'),
```

which are the recombination counterparts of the analogous methods of the Atom class.

If both the recombination and the collisional spectrum of the same atom are needed, two different objects must be built; thus, `pn.Atom('O', 2)` would describe the behavior of collisionally excited lines of O^+ – the [O II] spectrum, while `pn.RecAtom('O', 2)` would describe the behavior of recombination lines of the same ion – the O II spectrum.

Table 2. ICFs included in version 1.0.1. of PyNeb.

Source	Elements	Type
Delgado-Inglada et al. (2014)	He, O, N, Ne S, Cl, Ar, C	PNe
Girard et al. (2007)	Cl	PNe
Izotov et al. (2006)	N, O, Ne, S, Cl, Ar, Fe	H II
Kingsburgh & Barlow (1994)	C, N, O, Ne, S, Ar	PNe
Kwitter & Henry (2001)	He, N, O, Ne, Cl, Ar	PNe
Peimbert & Costero (1969)	Ne	H II
Peimbert et al. (1992)	He	H II
Pérez-Montero et al. (2007)	Ne, Ar	H II
Rodríguez & Rubin (2005)	Fe	H II
Stasińska (1978)	Ne	H II
Torres-Peimbert & Peimbert (1977)	N, O, Ne	PNe

3.4. Determining ionic and total abundances with PyNeb

If the physical conditions are known, it is straightforward to use the observed line intensities to determine the ionic abundance of any given X^i ion relative to H^+ by using Eq. (4), which, in PyNeb, is implemented as the `Atom` method `getIonAbundance`:

```
o3.getIonAbundance(int_ratio=100, tem=1.5e4, \
                   den=100, wave=5007).
```

Total abundances are computed by PyNeb following the ICF formalism described in Sect. 2.4. The code includes a large inventory of ICFs and a series of methods to explore them and their source papers. Table 2 lists the published ICFs included in version 1.0.1 of PyNeb. More expressions are being added constantly. PyNeb also features a method (`addICF`) to add customized expressions to the collection.

3.5. Physical condition diagnostics

The simple diagnostic calculation described in Sect. 3.2, which is a method of the `Atom` class, only involves a line ratio. The analysis of real objects often requires several diagnostic ratios to be considered simultaneously; doing this requires the use of a dedicated class, `Diagnostics`. A `Diagnostics` object, which is created by the command

```
diags=pn.Diagnostics(),
```

is a container for all the diagnostic ratios used during a calculation. Each diagnostic is associated to a label, the symbol and spectrum of the emitting ion, the defining expression in term of individual lines, and the rms error as a function of the rms error of the included lines.

A large number of published diagnostic ratios are stored in PyNeb by default and other ones are continuously added. Those included in the code at the time of writing this paper are shown in Table 3. User-defined diagnostics can also be added to PyNeb's built-in repertoire by adding any combination of lines of the

Table 3. Default line ratios included in version 1.0.1 of PyNeb.

[Ar III] λ5192/λ7136	Fe III] λ4735/λ4009	[Fe III] λ4987/λ4703	[N I] λ5198/λ5200	[O III] λ1666/λ4363
[Ar III] λ5192/λ7300+	Fe III] λ4882/λ4009	[Fe III] λ4987/λ4735	[N II] λ121μ/λ20.5μ	[O III] λ1666/λ5007
[Ar III] λ7136/λ9μ	Fe III] λ4882/λ4659	[Fe III] λ4987/λ4882	[N II] λ5755/λ6548	[O III] λ1666/λ5007+
[Ar III] $\lambda\lambda$7751, 7136/λ9μ	Fe III] λ4882/λ4703	[Fe III] λ4987/λ4926	[N II] λ5755/λ6584	[O III] λ4363/λ5007
[Ar III] λ9.0μ/λ21.8μ	[Fe III] λ4882/λ4932	[Fe III] λ4987/λ4932	[N II] λ5755/λ6584+	[O III] λ4363/λ5007+
[Ar IV] λ2860+/λ4720+	[Fe III] λ4882/λ5013	[Fe III] λ4987/λ5013	[Ne III] λ15.6μ/λ36.0μ	[O III] λ5007/λ88μ
[Ar IV] λ4740/λ4711	[Fe III] λ4926/λ4009	[Fe III] λ5013/λ4009	[N III] λ1750/λ57.4μ	[O III] λ51μ/λ88μ
[Ar IV] λ7230+/λ4720+	[Fe III] λ4926/λ4659	[Fe III] λ5013/λ4659	[Ne III] λ3343/λ3930+	[O IV] λ1401/λ1405
[Ar V] λ4626/λ6600+	[Fe III] λ4926/λ4703	[Fe III] λ5013/λ4735	[Ne III] λ3344/λ3930+	[S III] λ18.7μ/λ33.6μ
[C III] λ1909/λ1907	[Fe III] λ4926/λ4735	[Fe III] λ5013/λ4932	[Ne III] λ3869/λ15.6μ	[S III] λ6312λ/18.7μ
[Cl III] λ5538/λ5518	[Fe III] λ4926/λ4882	[Fe III] λ5272/λ4009	[Ne III] λ3930+/λ15.6μ	[S III] λ6312/λ9069
[Cl IV] λ5323/λ7531	[Fe III] λ4926/λ4932	[Fe III] λ5272/λ4659	[Ne V] λ14.3μ/λ24.2μ	[S III] λ6312/λ9200+
[Cl IV] λ5323/λ7700+	[Fe III] λ4926/λ5013	[Fe III] λ5272/λ4703	[Ne V] λ2975/λ3370+	[S III] λ9069/λ18.7μ
[Fe III] λ4659/λ4009	[Fe III] λ4932/λ4009	[Fe III] λ5272/λ4735	[O I] λ5579/λ6300	[S II] λ4069/λ4076
[Fe III] λ4659/λ4703	[Fe III] λ4932/λ4659	[Fe III] λ5272/λ4882	[O I] λ5579/λ6300+	[S II] λ4072+/λ6720+
[Fe III] λ4659/λ4735	[Fe III] λ4932/λ4703	[Fe III] λ5272/λ4926	[O I] λ5579/λ6302	[S II] λ6731/λ6716
[Fe III] λ4659+/λ4987+	[Fe III] λ4932/λ4735	[Fe III] λ5272/λ4932	[O I] λ63μ/λ147μ	
[Fe III] λ4703/λ4009	[Fe III] λ4987/λ4009	[Fe III] λ5272/λ4987	[O II] λ3726/λ3729	
[Fe III] λ4703/λ4735	[Fe III] λ4987/λ4659	[Fe III] λ5272/λ5013	[O II] λ3727+/λ7325+	

Notes. The suffix "+" indicates that the line is a blend.

same ion, but it is the user's responsibility to ensure that the diagnostics added are meaningful and that the correct expression for the rms is given.

The most important feature of the `Diagnostics` class is the `getCrossTemDen method`, which allows simultaneously determining the temperature and density by fitting two line ratios:

```
tem, den = diags.getCrossTemDen(
                        diag_tem='[NII] 5755/6548',
                        diag_den='[SII] 6731/6716',
                        value_tem=50,value_den=1.0).
```

In this command, the first two parameters are the labels of two line ratios and the following two are the corresponding values. The command stores the resulting T_e and N_e values in the variables `tem` and `den`, which can be used in subsequent calculations. This method works by making an initial guess of the temperature, after which it applies a simple multisection procedure that iteratively calls `getTemDen`, until a solution is found within a given precision.

3.6. Emissivities as a function of temperature and density: EmisGrid

Many of PyNeb's tasks are executed by manipulating precalculated grids of line emissivities as a function of temperature and density, obtained by instantiating the class `EmisGrid`, which produces a stack of emission grids for the selected atom. A stack has as many layers as lines in the atom (see the example of the [O III] `EmisGrid` object in Fig. 2). The class is equipped with methods to display the information in various ways, such as color-coded emissivity maps and contour plots (Fig. 3).

3.7. Managing observational data: the Observation class

PyNeb's `Observation` class creates the data structure in charge of holding observational data. An `Observation` object includes at least a set of line intensities for one or more objects and may include other related information, such as observational errors, the $c(H\beta)$ coefficient, the dereddened line intensities, and an indication of the dereddening law used (which may be either built-in or user-defined; see Table 4 and Fig. 4). The data are generally read from a file:

```
obs = pn.Observation()
obs.readData(obsFile='ic418.dat').
```

3.7.1. Observational diagnostics plots

The joint use of Observation, Diagnostic, and EmisGrid objects makes it possible to build observational diagnostic plots (Fig. 5). We note that, when more than two lines ratios are available for a given object, a unique solution generally does not exist. The analysis of these plots requires taking observational errors into account and full consideration of all possible causes for the occurrence of "discordant" diagnostics in such plot, such as observational ones (e.g., biased or otherwise low-quality observational data), theoretical ones (e.g., emission grids computed with biased atomic data, unaccounted-for processes such as fluorescence, etc.) or physical ones (e.g., a complex ionization structure, which hinders direct comparison of line ratios of different ions). A full discussion of the specific case of fluorescent emission in IC 418 is given in Escalante et al. (2012) (see also Sect. 4.2).

3.8. Atomic data

As already stated in Sect. 3.2, the results of the calculations carried out by PyNeb depend on the assumed transition probabilities and collision strengths. These quantities are the result of complex atomic physics calculations and are subject to uncertainties. While the uncertainty is relatively low for some atoms, other atoms are more complex to model and the resulting uncertainty is correspondingly larger; in such cases, the result of nebular analysis may depend strongly on the assumed atomic data set. To address this issue, PyNeb provides a large data base with as many determinations as possible for each ion and a number of tools to assess the differences between them. The atomic

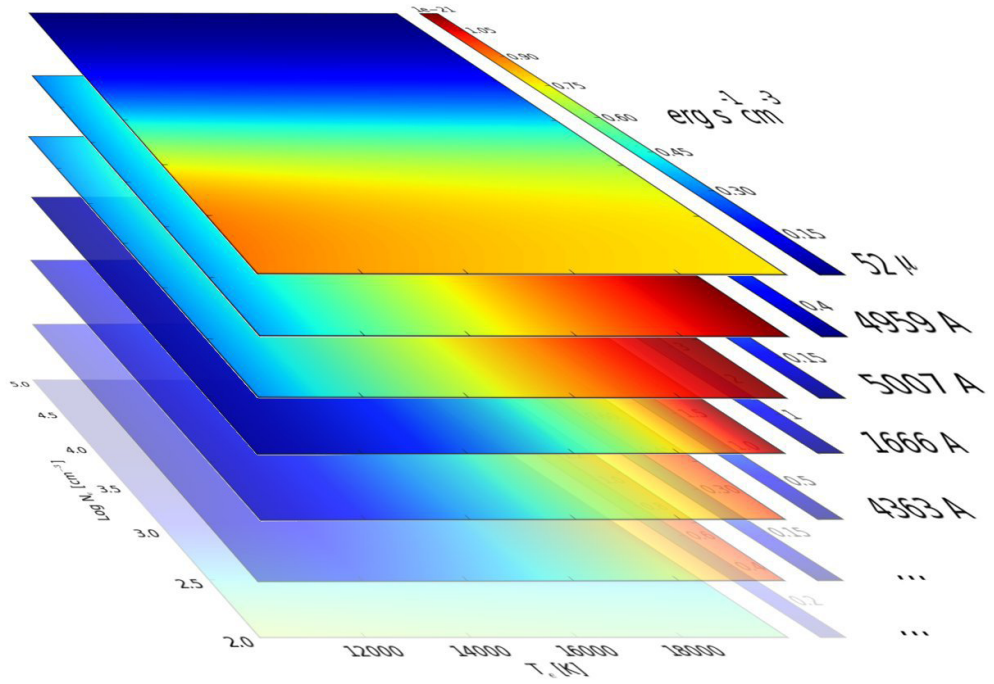
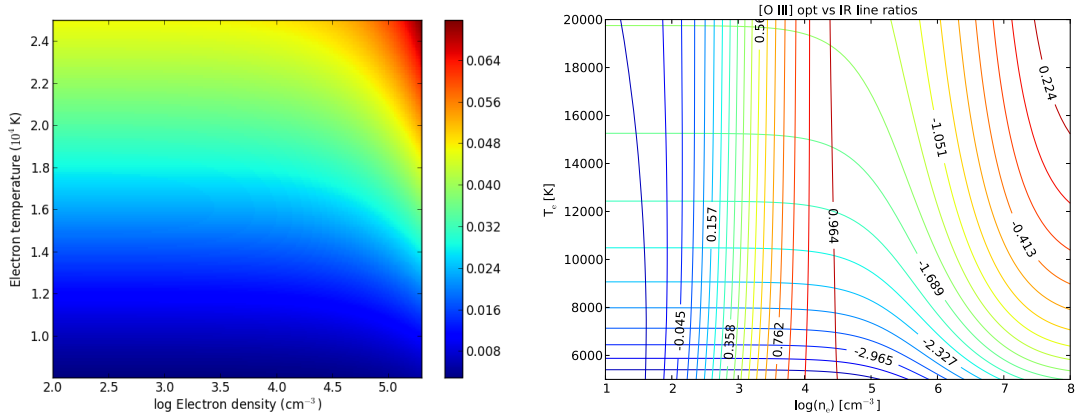


Fig. 2. EmisGrid object for O III. Each layer in the stack represents the emissivity of an O III line in the chosen T_e, N_e range.



(a) (b)

Fig. 3. *Left*: map of the [O III] λ4363/λ5007 intensity ratio as a function of electron density and temperature, obtained with the plotImage() command. *Right*: theoretical contour plot for the simultaneous determination of T_e and N_e with the [O III] $\lambda\lambda$4959, 5007/λ4363 (mainly horizontal curves) and [O III] $\lambda 52\mu/\lambda 88\mu$ (vertical curves) intensity ratios as a function of electronic density and temperature, obtained with the plotContours() command. The levels are numbered with the log of the ratio.

data base and the related tools are an integral part of PyNeb's philosophy and will be described in Paper II of this series.

4. Code validation

As mentioned above, PyNeb shares much of NEBULAR functionality; in particular, both are able to compute critical densities, level populations, line emissivities, electron temperatures and densities (given the other quantity and a line ratio), and ionic abundances. Table 5 lists the NEBULAR tasks involved and the corresponding PyNeb methods. The tests conducted show that PyNeb predictions coincide with those by NEBULAR (if, of course, the same atomic data are adopted). For other quantities computed by PyNeb, a direct comparison is not possible, either because NEBULAR does not compute them (as for elemental abundances) or because it does not output them (as for collisional rates); but, even in the latter case, the consistency of the results clearly indicates that PyNeb is working as expected.

The next section compares the output of selected NEBULAR tasks with the corresponding PyNeb's methods for specific cases. The NEBULAR version used to conduct the tests is the one contained in IRAF 2.16.

4.1. Populations, emissivities, and physical conditions

Figure 6 compares the output of `ionic` and `printIonic`, both of which run with the same [O III] data set. The output includes the level populations, the critical densities, and the emissivities. The results from the two codes agree in general but also exhibit slight differences. Even if these are negligible from a practical point of view, it is important to understand them to ensure that

Table 4. Extinction laws included in version 1.0.1 of PyNeb.

Source	Label	Type
Cardelli et al. (1989)	CCM 89	Galactic
Blagrave et al. (2007), Cardelli et al. (1989)	CCM89 Bal07	Galactic
O'Donnell (1994), Cardelli et al. (1989)	CCM89 oD94	Galactic
Fitzpatrick & Massa (1988), Fitzpatrick (1999)	F88 F99 LMC	LMC
Fitzpatrick (1999)	F99	Galactic
Fitzpatrick (1999)	F99-like	
Gordon et al. (2003)	G03 LMC	LMC
Kaler (1976)	K76	Galactic
Seaton (1979), Howarth (1983), Cardelli et al. (1989)	S79 H83 CCM89	Galactic
Savage & Mathis (1979)	SM79 Gal	Galactic

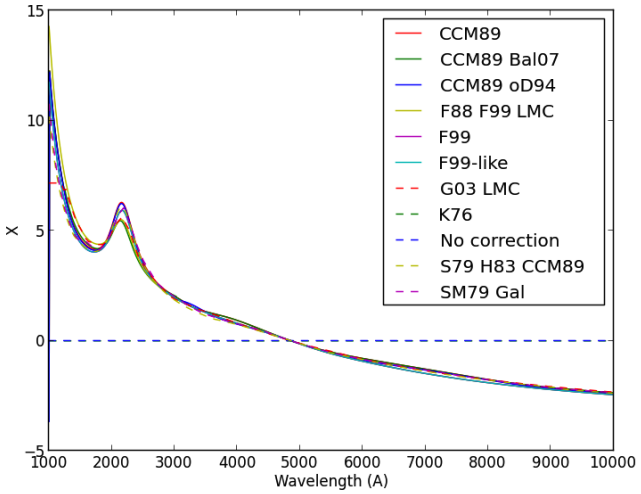


Fig. 4. Built-in extinction laws included in PyNeb. Details on the laws can be displayed by typing `pn.RedCorr().printLaws()`. Additional laws can be provided by the user, either in tabular form or as an analytical formula. The plot has been obtained with the `RC.plot(laws='all')` command, where `RC` is an instance of `pn.RedCorr()`.

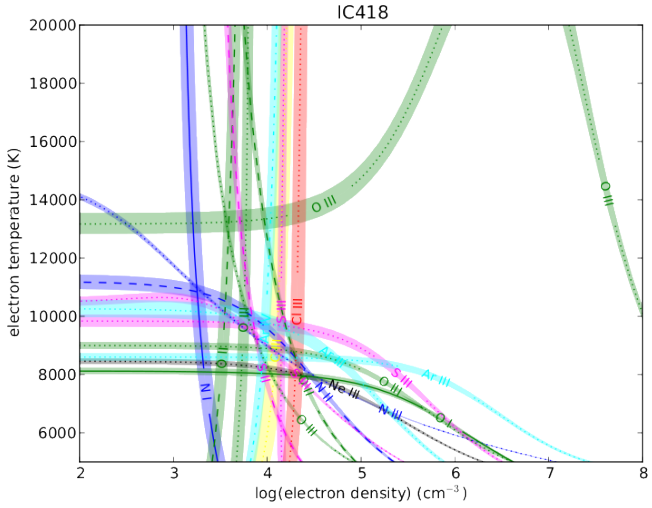


Fig. 5. Emission-line diagnostic plot of the planetary nebula IC 418. Each band represents the N_e, T_e locus consistent with the observed ratio and its uncertainty. The observational data have been taken from the compilation in Morisset & Georgiev (2009) and an analysis of the object is given in Escalante et al. (2012). A sample script that produces this kind of diagnostic plots is given in the Appendix.

no systematic errors affect the output of either code. It should be noticed that all differences can ultimately be ascribed to differences between the computed level populations. While we could not track back the differences to any specific factor, the experiments conducted suggest that the differences are not systematic and are most probably related to the greater precision of both constants and variables enforced in PyNeb. This implies that the differences are negligible and can be safely ignored; furthermore, on the scale of the tiny differences found, PyNeb is more precise than NEBULAR. As an example, we show in Fig. 7 the error in the total [S II] population output by both NEBULAR and PyNeb.

Table 5. Correspondence between NEBULAR's tasks and PyNeb's methods.

NEBULAR	PyNeb	Quantity computed
ionic	getIonAbundance	Ionic abundances
ionic	printIonic	Critical densities
ionic	printIonic	Emissivities
ionic	Atom.printIonic, RecAtom.getEmissivity	Hβ emissivity
temden	getTemDen	T_e or N_e
temden	getCrossTemDen	T_e and N_e
ntcontour	plotLineRatio, plotContour	Diagnostic plot
redcorr	correctData	Dereddened fluxes
ionic	getIonAbundance	Ionic abundances

Figure 8 compares the emissivity of Hβ returned by NEBULAR with the one computed by PyNeb with the same interpolation formula (Aller 1984). The default emissivity used by PyNeb (Storey & Hummer 1995) is also displayed for reference, normalized to the Aller (1984) formula. The wiggles are an artifact of the linear interpolation in the table; they indicate that a smooth interpolation would probably work better. Focusing on the data points underlying the wiggles, it can be seen that the Hβ emissivity by Storey & Hummer (1995) is roughly 2% less than the one predicted by Aller (1984).

Figure 9 compares the [O III]$\lambda\lambda$5007, 4959/λ4363 ratio emissivity of Hβ returned by NEBULAR's task `temden` with the one computed by PyNeb's method `getTemDen`, for two different density values. On the scale of the figure, NEBULAR's results are indistinguishable from PyNeb's results. By zooming in, differences on the order of 10 K can be seen. These differences, which are negligible in all practical applications, can be ascribed to the differences in the level populations mentioned above.

Tables 6 and 7 compare the results of NEBULAR and PyNeb for all the density and temperature diagnostics of Tables 2 and 3 of Shaw & Dufour (1995) at selected temperature and density points. Again, the differences are negligible in all cases.

Finally, diagnostic plots can be used in essentially the same way in both NEBULAR and PyNeb, with the latter allowing

IRAF	PyNeb
# Volume Emissivities for: O^2+ T_e: 10000.0; N_e: 1.000E3 # Level Populations - Critical Densities Level 1: 3.1E-1 Level 2: 4.877E-1 5.025E2 Level 3: 2.032E-1 3.436E3 Level 4: 4.411E-5 6.878E5 Level 5: 3.196E-9 2.372E7 Level 6: 2.61E-12 3.686E10 884013.85 # Wavelength (2-->1) # Upper->Lower Level 2.920E-22 # Volume Emissivity 325494.34 515186.14 (3-->1) (3-->2) 3.838E-28 7.598E-22 4930.99 4958.65 5006.84 (4-->1) (4-->2) (4-->3) 3.003E-25 1.236E-21 3.572E-21 2314.88 2320.96 2331.46 4363.21 (5-->1) (5-->2) (5-->3) (5-->4) INDEF 6.204E-24 1.659E-26 2.271E-23 1657.66 1660.77 1666.14 2497.12 5838.63 (6-->1) (6-->2) (6-->3) (6-->4) (6-->5) INDEF 6.857E-24 1.703E-23 9.826E-29 INDEF # H-beta Volume Emissivity: 1.258E-25 N(H+) * N(e-) (erg/s)	elem = O spec = 3 temperature = 10000.0 K density = 1000.0 cm-3 Level Populations Critical densities Level 1: 3.096E-01 0.000E+00 Level 2: 4.876E-01 5.044E+02 Level 3: 2.028E-01 3.449E+03 Level 4: 4.408E-05 6.884E+05 Level 5: 3.205E-09 2.396E+07 Level 6: 2.584E-12 3.718E+10 88.40m (2-->1) 2.919E-22 32.55m 51.52m (3-->1) (3-->2) 3.829E-28 7.579E-22 4930.99A 4958.65A 5006.84A (4-->1) (4-->2) (4-->3) 3.001E-25 1.235E-21 3.570E-21 2314.88A 2320.96A 2331.46A 4363.21A (5-->1) (5-->2) (5-->3) (5-->4) 0.000E+00 6.222E-24 1.663E-26 2.278E-23 1657.66A 1660.77A 1666.14A 2497.12A 5838.63A (6-->1) (6-->2) (6-->3) (6-->4) (6-->5) 0.000E+00 6.800E-24 1.688E-23 9.744E-29 0.000E+00 # H-beta volume emissivity: 1.237E-25 N(H+) * N(e-) (erg/s)

Fig. 6. Comparison between the output of the NEBULAR task `ionic` and the PyNeb method `printIonic`, both run with the [O III] data set adopted by NEBULAR in 2009.

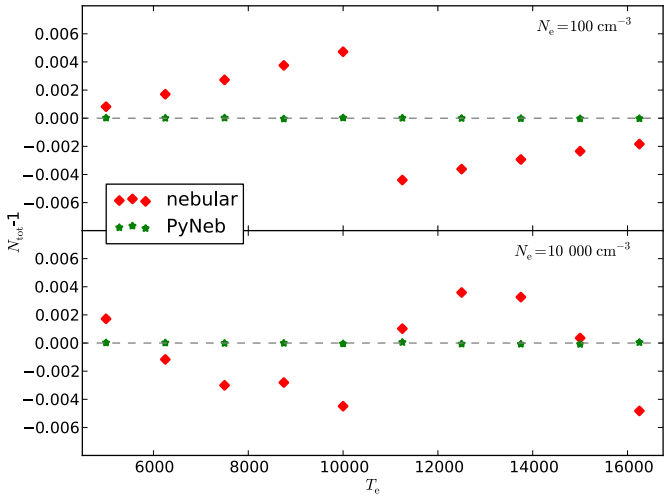


Fig. 7. Normalization error in the total [S II] population output by NEBULAR and PyNeb at two representative density values.

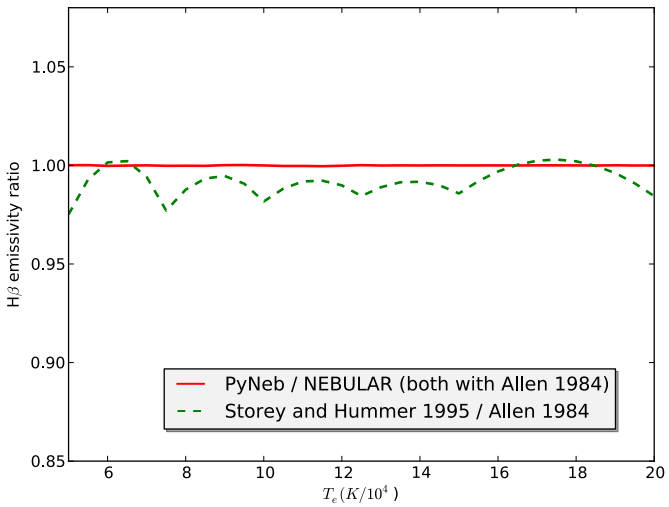


Fig. 8. Comparison between the Hβ emissivities computed by NEBULAR and PyNeb with the Aller (1984) formula. Also shown is the ratio of PyNeb's default Hβ emissivity, computed by linear interpolating in Storey & Hummer (1995) tables, relative to the emissivity by Aller (1984).

a greater flexibility in terms of the plot's graphical features (Fig. 10).

4.2. Limitations and caveats

Powerful as it is, PyNeb incorporates a fairly simple physical model of line emission in nebular objects, and it is important to ensure that it is not stretched outside its range of validity. This section enumerates a few physical problems that cannot be solved by PyNeb.

An important process that cannot be accounted for by PyNeb is fluorescence, which may affect the emitted intensity of both collisional and recombination lines (Escalante & Morisset 2005; Luridiana et al. 2009). Fluorescence is a line emission mechanism triggered by the absorption of a photon previously emitted either from a stellar source (continuum pumping) or from another ion (resonant fluorescence). While the intensity of collisional and recombination lines merely depends on the local conditions and the abundance of the ion involved, the intensity emitted through fluorescence depends on the intensity, spectrum, position, and relative velocity of the stellar source (in the case of continuum pumping) or on the simultaneous emission of other ions (fluorescence of resonant lines). The intensity of the line produced through this process adds to the intensity produced by collisional or recombination processes, a fact to take into account when observational data are analyzed. Since PyNeb considers the ions in isolation, it cannot account for this

Table 6. Comparison between density diagnostics in NEBULAR and PyNeb.

Ion	Spectrum	Line Ratio	Value	Assumed T_e	N_e (NEBULAR)	N_e (PyNeb)
N^0	[N I]	$I(5198)/I(5200)$	2.00	10 000 K	3420	3420
S^+	[S II]	$I(6716)/I(6731)$	0.75	10 000 K	1740	1740
C^+	C II]	$I(2326)/I(2328)$	1.60	10 000 K	100	100
O^+	[O II]	$I(3726)/I(3729)$	1.50	10 000 K	1100	1090
Si^{+2}	Si III]	$I(1883)/I(1892)$	0.10	10 000 K	772 100	771 300
Cl^{+2}	[Cl III]	$I(5517)/I(5538)$	0.75	10 000 K	6480	6470
N^{+2}	N III]	$I(1749)/I(1752)$	1.20	10 000 K	510	510
Ar^{+3}	[Ar IV]	$I(4711)/I(4740)$	0.50	10 000 K	23 250	23 240
C^{+2}	C III]	$I(1907)/I(1909)$	0.40	15 000 K	115 800	114 400
O^{+3}	O IV]	$I(1401)/I(1405)$	1.25	10 000 K	4640	4650
Ne^{+3}	[Ne IV]	$I(2423)/I(2425)$	2.00	10 000 K	19 400	19 400

Notes. Densities have been rounded off to the tens for values below 10^5 cm^{-3} and to the hundreds elsewhere.

Table 7. Comparison between temperature diagnostics in NEBULAR and PyNeb.

Ion	Spectrum	Line ratio	Value	Assumed *Ne*	T_e (NEBULAR)	T_e (PyNeb)
O^0	[O I]	$I(5577)/I(6300+6363)$	0.01	1000 cm^{-3}	8400	8400
S^+	[S II]	$I(4068+4076)/I(6716+6731)$	0.10	100 cm^{-3}	11 510	11 510
O^+	[O II]	$I(3726+3729)/I(7320+7330)$	100	100 cm^{-3}	7480	7480
N^+	[N II]	$I(5755)/I(6548+6583)$	0.1	100 cm^{-3}	9970	9970
Si^{+2}	Si III]	$I(1206)/I(1883+1892)$	0.025	1000 cm^{-3}	15 790	15 790
S^{+2}	[S III]	$I(6312)/I(9069+9532)$	0.02	1000 cm^{-3}	11 040	11 090
Ar^{+2}	[Ar III]	$I(5192)/I(7136+7751)$	0.01	1000 cm^{-3}	12 410	12 420
O^{+2}	[O III]	$I(4363)/I(4959+5007)$	0.01	1000 cm^{-3}	12 940	12 920
Cl^{+3}	[Cl IV]	$I(5323)/I(7530+8045)$	0.01	1000 cm^{-3}	8180	8180
Ar^{+3}	[Ar IV]	$I(2854+2868)/I(4711+4740)$	0.01	100 cm^{-3}	8270	8270
Ne^{+2}	[Ne III]	$I(3342)/I(3869+3969)$	0.001	1000 cm^{-3}	10 050	10 060
Ar^{+4}	[Ar V]	$I(4626)/I(6435+7006)$	0.01	1000 cm^{-3}	12 520	12 500
Ne^{+4}	[Ne V]	$I(2975)/I(3426+3346)$	0.001	1000 cm^{-3}	12 020	12 040

Notes. Temperatures have been rounded off to the tens.

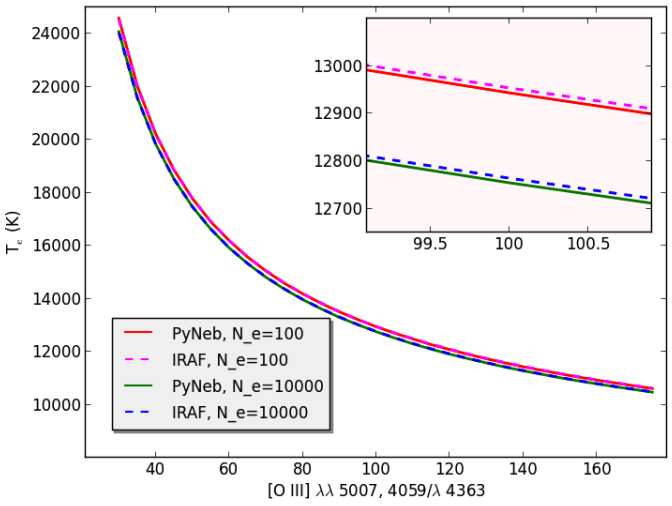
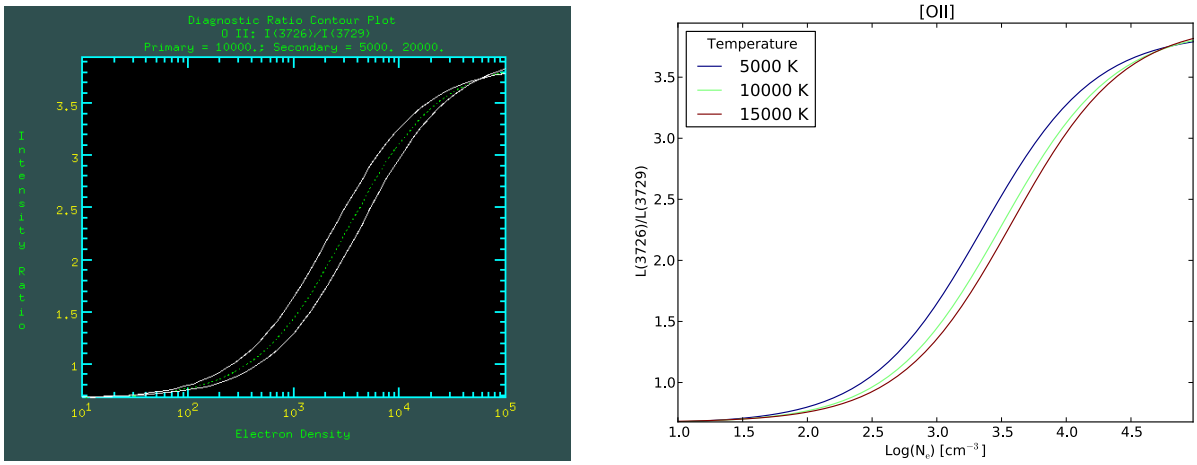


Fig. 9. Comparison between the output of the NEBULAR task `temden` and the PyNeb method `getTemDen`, both run with the [O III] data set adopted by IRAF in 2009.

line-enhancing process, which must be corrected for explicitly before interpreting observed intensities in terms of physical conditions and chemical abundances.

In the context of fluorescence, it is useful to discuss the recent results by Proxauf et al. (2014), who claim that the usual relation between the electron temperature and the [O III] line ratio $R = \lambda\lambda(5007+4959)/\lambda 4363$ (as computed, for instance, by PyNeb) is biased and should be revised. The new relation they propose would lead to a decrease in the electron temperature computed from a given ratio R, compared to what would be obtained with the standard relation, with the discrepancy between both relations increasing with the electron temperature (see their Fig. 1). For several reasons, we consider these results to be erroneous. First of all, the authors claim that the effect is due to fluorescence, but they fail to notice that, in their Cloudy photoionization models (Ferland et al. 1998), none of the three lines involved in the ratio R is affected by this mechanism. Second, since their results are based on a solar-metallicity grid, the authors are forced to increase the effective temperature of the ionizing SED up to unrealistic values of a few 10^5 K in order to reach high electron temperatures; furthermore, the models are density-bounded and have a very small geometrical thickness. As a consequence, the models with the highest electron temperature also have a very high ionization degree, which leads to unrealistically high values of O^{3+}/O^{2+} of ~100. This, in turn, causes a huge increase in the contribution of O^{3+} recombination to the emissivity of the [O III] λ4363 line (see Eq. (3) of Liu et al. 2000) by the same factor and, consequently, an artificial decrease in the diagnostic line ratio at a given electron temperature. In real nebulae, the contribution of recombination to this line is invariably much smaller than the contribution determined by Proxauf et al. (2014) (classical planetary nebulae only marginally reach $O^{3+}/O^{2+} = 1$ and H II regions have even lower values), ruling out the validity of their Eq. (1). If one ever needs to take into account the contribution of recombination to the intensity of the [O III] λ4363 line, we suggest using the recipe by Liu et al. (2000).



(a) (b)

Fig. 10. *Left*: density diagnostic plot of the [O III] λ3726/λ3729 intensity ratio as a function of electron density and temperature, obtained with the NEBULAR command `ntcontour`. *Right*: density diagnostic plot of the [O III] λ3726/λ3729 intensity ratio as a function of electron density and temperature, obtained with the PyNeb command `plotLineRatio`.

Similarly, PyNeb is a *local* code, so it is not designed to solve radiation transfer. This does not hinder interpretation of observed lines since, when optical depths are low – which is true for most of these lines – photons freely escape, and there is no need of a detailed treatment of radiation transfer. Dust is only included as the source of foreground extinction, but PyNeb is not designed to compute the effect of dust grains on the thermal or ionization equilibrium.

PyNeb cannot account for energy balance issues either (thus, it cannot give clues to the temperature fluctuations problem), nor can it compute a finite-depth, complex physical model. In short, PyNeb is not a photoionization code but rather a code for analyzing emission lines.

Finally, it must be remembered that both the *A*s and the Υs are subject to uncertainties. The discussion of these and the description of PyNeb's atomic data base will be the subject of Paper II.

5. Practical details

5.1. Software requirements

As of mid 2014, the code requires the following packages and libraries to function:

1. *python*[2] v. 2.7x (*not* v. 3.x, which is a different and incompatible branch of python)
2. *numpy*[3] v. 1.6 or later
3. *matplotlib*[4] v. 1.2 or later
4. *scipy*[5] v. 0.10 or later.

The first two are required for the code to function. The graphical library *matplotlib* is needed to use the graphic modules of PyNeb, but the code is prepared to skip them if the library is not installed. In such cases, a warning will be issued, but the code will not stop. Finally, *scipy* is currently only required to compute recombination emissivities with the `RecAtom` class (see Sect. 3.3) and to compute stellar temperatures with the `Stellar` class.

5.1.1. Running the code

PyNeb can be used in two ways: either interactively feeding individual commands to the command line or running a script. The first way to proceed is suitable for quick, simple computations; the second is more adequate if a large number of commands have to be executed in a sequence. Examples of both are given in the documentation, which is discussed in Sect. 5.2.

5.1.2. Running with multiple processors

PyNeb takes full advantage of *numpy*'s features to perform fast, vectorized computations, thus avoiding time-consuming loops in repetitive operations such as computing an emissivity grid, which requires computing an $n_{\rm tem} \times n_{\rm den}$ array of emissivities. In addition to this, the method Atom.getTemDen (which computes the temperature given the density and vice versa, and will be presented in Sect. 3.2) has been parallelized to be used with multiple processors, reducing the execution time further. The method `Diagnostics.getCrossTemDen`, which simultaneously computes the temperature and density and will be described in Sect. 3.5, works by iteratively calling `Atom.getTemDen` and is therefore also boosted by the use of multiple processors.

5.2. Documentation and support

Efficient and safe use of a public package is only possible if the package is well documented. PyNeb currently has several sources of documentation, also listed on the web page of the code[6]:

1. PyNeb's manual: a short, discursive introduction to PyNeb with plenty of examples, which is continuously updated as the code evolves.
2. The code docstrings, which can be read interactively through standard python or ipython commands: these are also used to

[2] http://www.python.org
[3] http://www.numpy.org
[4] http://www.matplotlib.org
[5] http://www.scipy.org
[6] http://www.iac.es/proyecto/PyNeb

automatically generate a reference document, which is available in both html and pdf formats.
3. A series of well-commented scripts, which illustrate real science cases and can be downloaded from the web page.
4. This paper and Paper II.

In addition, there is also a Google discussion group called PyNeb[7] to post requests, help other users, and share your problems. Direct support from the development team (e-mail: `pyneb@googlegroups.com`) is on a time-available basis.

6. Concluding remarks: the present and future of PyNeb

The following is a list that summarizes PyNeb's main features:

1. A large sample of atomic data is provided with the code. More are being added, and tools to add your own data are also provided. Changing from one atomic data set to another is foolproof. PyNeb also includes tools to explore and visualize the atomic data used.
2. Thanks to *matplotlib*, PyNeb has improved graphical capabilities to produce print quality plots. Additionally, because the user has full access to intermediate quantities of computations, built-in commands can be complemented by user-produced scripts to produce tailored plots.
3. Simultaneous determination of temperature and density from pairs of line ratios is possible without the need of external looping on `getTemDen`.
4. Elemental abundances can be computed from ionic abundances with a simple command. A selection of published ICFs is provided with complete information on their analytical form and bibliographic source. The collection of built-in ICFs can be complemented by user-provided formulae.
5. Recombination intensities for some important ions have been added.
6. Some ions have been added to the NEBULAR inventory of collisional atoms, including both common ions (such as [Cl II]) and several s-elements.
7. Insight into emissivity properties is possible thanks to emissivity grids.
8. PyNeb has a modular structure, which makes it easy to embed modules in scripts.

Although the core functionality of PyNeb is already in place and fairly stable, we continue to improve the code and to add new functionality. The following are some areas of future potential development:

1. We intend to add new ions to the inventory of collisional-excited lines; in particular, we are seeking to extend the inventory of *s*-process elements and other heavy ions. More details on this point are given in Paper II.
2. We also plan to extend the choice of recombination ions to include, e.g., O II and C II.
3. Currently, the treatment of blends is not full: while they can be included in diagnostics, they cannot be used yet as input to the `getIonAbundance` method. We intend to improve this in the near future.
4. A Monte Carlo procedure to estimate the error in abundances derived from errors in fluxes will be added to the code.

Ideally, it would also be desirable to be able to create VO-compatible web services to provide PyNeb over the internet, or, alternatively, to create a GUI to be installed locally to provide at least some of PyNeb's functionality. While we currently lack the labor-force to accomplish such goals, we do not exclude that they might be achieved in the future, possibly with the collaboration of users. We would be grateful for any feedback, query, or comment on the code and its accompanying documentation.

PyNeb has already been used in a number of papers (García-Rojas et al. 2012, 2013, 2014; Esteban et al. 2013, 2014; Kameswara Rao et al. 2013; Nicholls et al. 2013, 2014; Marino et al. 2013; Delgado-Inglada et al. 2014; Delgado-Inglada & Rodríguez 2014; Mesa-Delgado et al. 2014; Sankrit et al. 2014; Ho et al. 2014; Pérez-Montero 2014). The scientific community is invited to keep using it for their projects. Whenever you use PyNeb for calculations that lead to a published paper, you are kindly asked to cite both this paper and Paper II.

Acknowledgements. V.L. acknowledges support of the Spanish Ministry of Science and Innovation through grant AYA 2011-22614. C.M. acknowledges support from CONACyT project CB-2010/153985. Support for programs numbered HST-GO–11657 and HST-GO–112600 was provided by NASA through a grant from the Space Telescope Science Institute, which is operated by the Associated Universities for Research in Astronomy, Inc., under NASA contract NAS5-26555. Thanks are due to Amanda Mashburn and Nick Sterling for kindly providing data for several *s*-process ions. We also acknowledge useful suggestions from Jorge García-Rojas, Adal Mesa Delgado, Luis López-Martín and Rubén García-Benito.

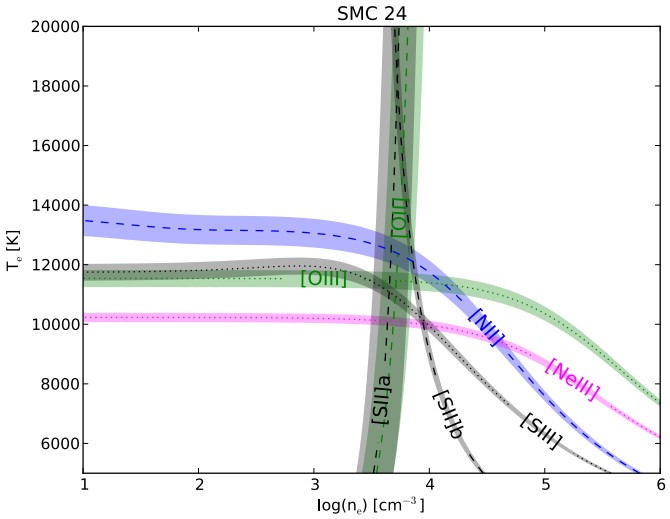


Fig. A.1. Diagnostic diagram of SMC24 produced with smc24_diagnostics.py.

Appendix A: Sample scripts

Here we include two sample scripts that illustrate how real physical problems can be analyzed with PyNeb. The first script produces a diagnostic plot similar to the one of Fig. 5 (Fig. A.1). The content of the data file is given at the end of the script and can also be downloaded from `http://www.iac.es/proyecto/PyNeb/smc24.dat`

```
# PyNeb script smc24_diagnostics.py

# Imports
import pyneb as pn
import matplotlib.pyplot as plt

# Adopt an extinction law
extinction_law = 'CCM 89'
```

[7] https://groups.google.com/forum/#!forum/pyneb

```
# Define the data file
obs_data = 'smc24.dat'

# Plot title
title = 'SMC 24'

### Read and deredden observational data
# define an Observation object and name it 'obs'
obs = pn.Observation()

# fill obs with data from obs_data
obs.readData(obs_data, fileFormat='lines_in_rows',
                     err_default=0.05)

# deredden data with Cardelli's law
obs.extinction.law = extinction_law
obs.correctData()

### Include the diagnostics of interest
# instantiate the Diagnostics class
diags = pn.Diagnostics()
# include in diags the relevant line ratios
diags.addDiag([
               '[NII] 5755/6584',
               '[OII] 3726/3729',
               '[OIII] 4363/5007',
               '[SII] 6731/6716',
               '[SII] 4072+/6720+',
               '[SIII] 6312/18.7m',
               '[NeIII] 3930+/15.6m',
              ])
diags.addClabel('[SII] 6731/6716', '[SII]a')
diags.addClabel('[SII] 4072+/6720+', '[SII]b')

# Create the emission maps to be compared to
# the observation data (some overkill here)
emisgrids = pn.getEmisGridDict(atom_list=\
    diags.getUniqueAtoms(), den_max=1e6)

### Plot
# Create the contour plot as the intersection of
# tem-den emission maps with dereddened line ratios
diags.plot(emisgrids, obs)

# Place the title
plt.title(title)

# Display the plot
plt.show()

# Data
# NAME SMC_24
# cHbeta       0.047
# S4_10.5m     7.00000
# Ne2_12.8m    8.3000
# Ne3_15.6m   34.10
# S3_18.7m    10.4
# O2_3726A    39.700
# O2_3729A    18.600
# Ne3_3869A   18.90
# Ne3_3967A    6.4
# S2_4069A     0.85
# S2_4076A     0.450
# O3_4363A     4.36
# O3_5007A     435.09
# N2_5755A     0.510000
# S3_6312A     0.76
# O1_6300A     1.69
# O1_6364A     0.54
# N2_6548A     6.840000
```

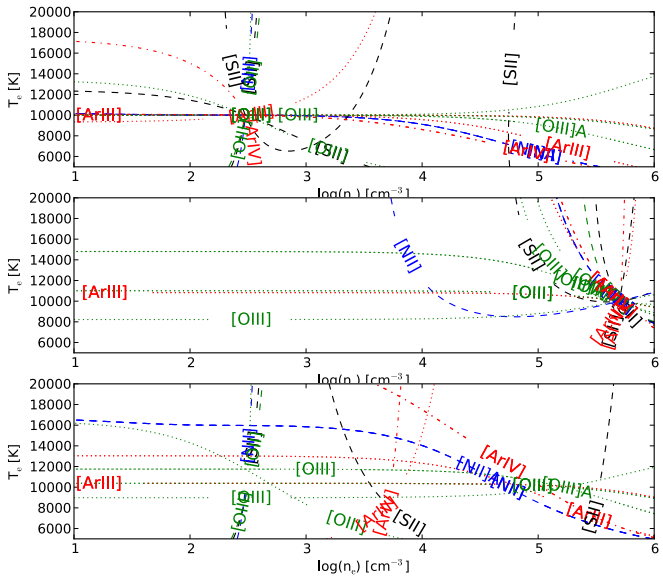


Fig. A.2. Plots produced with the script two_comp.py to illustrate the bias resulting from analyzing a two-density region as a homogeneous one.

```
# N2_6584A     19.00
# S2_6716A      1.220000
# S2_6731A      2.180000
# Ar3_7136A     4.91
# O2_7319A+     6.540000
# O2_7330A+     5.17
```

The second script illustrates the case of a density-inhomogenous region that is erroneously analyzed as a constant-density region (see also Fig. A.2).

```
# PyNeb script smc24_diagnostics.py

# imports
import pyneb as pn
import matplotlib.pyplot as plt

# Define the two subregions in terms
# of temperature, density and mass

tem1 = 1e4
tem2 = 1e4
dens1 = 3e2
dens2 = 5e5
mass1 = 1
mass2 = 5e-4

# Create list of diagnostics to analyze the region
diags = pn.Diagnostics()
for ion in ['O2', 'O3', 'S2', 'N2', 'Ar3', 'Ar4']:
    diags.addDiag(atom=ion)
    print 'Adding diagnostics of', ion

# Use alternate labels for some ratios
diags.addClabel('[NII] 5755/6584', '[NII]A')
diags.addClabel('[OIII] 4363/5007', '[OIII]A')

# Create all the ions involved in the diagnostics
adict = diags.atomDict
pn.log_.message('Atoms built')
```

```
# Create a virtual observation with 3 records:
# subregion1, subregion 2 and the sum of both
obs = pn.Observation(corrected = True)

# Compute the intensities of all the lines
# of all the ions considered
for atom in adict:
    for line in pn.LINE_LABEL_LIST[atom]:
        if line[-1] == 'm':
            wave = float(line[:-1])*1e4
        else:
            wave = float(line[:-1])

        intens1 = adict[atom].getEmissivity(tem1, \
                      dens1, wave=wave)*dens1*mass1
        intens2 = adict[atom].getEmissivity(tem2, \
                      dens2, wave=wave)*dens2*mass2
        elem = adict[atom].elem
        spec = adict[atom].spec
        obs.addLine(pn.EmissionLine(elem, spec, wave, \
              obsIntens= [intens1, intens2, \
              intens1+intens2], obsError=[0., 0., 0.]))
pn.log_.message('Virtual observations computed')

# Compute emission grids of the atoms considered
emisgrids = pn.getEmisGridDict(atomDict=adict)
pn.log_.message('EmisGrids available')

# Produce a diagnostic plot for each region and
# another one for the (misanalyzed) overall region
plt.figure()
plt.subplot(3, 1, 1)
diags.plot(emisgrids, obs, i_obs=0)
plt.subplot(3, 1, 2)
diags.plot(emisgrids, obs, i_obs=1)
plt.subplot(3, 1, 3)
diags.plot(emisgrids, obs, i_obs=2)
plt.show()
```

References

Aller, L. H., ed. 1984, Physics of thermal gaseous nebulae, Astrophys. Space Sci. Lib. (Dordrecht: D. Reidel Publishing), 112
Baker, J. G., & Menzel, D. H. 1938, ApJ, 88, 52
Baker, J. G., Menzel, D. H., & Aller, L. H. 1938, ApJ, 88, 422
Blagrave, K. P. M., Martin, P. G., Rubin, R. H., et al. 2007, ApJ, 655, 299
Burgess, A. 1958, MNRAS, 118, 477
Cardelli, J. A., Clayton, G. C., & Mathis, J. S. 1989, ApJ, 345, 245
Davidson, K., & Kinman, T. D. 1985, ApJS, 58, 321
De Robertis, M. M., Dufour, R. J., & Hunt, R. W. 1987, JRASC, 81, 195
Delgado-Inglada, G., & Rodríguez, M. 2014, ApJ, 784, 173
Delgado-Inglada, G., Morisset, C., & Stasińska, G. 2014, MNRAS, 440, 536
Escalante, V., & Morisset, C. 2005, MNRAS, 361, 813
Escalante, V., Morisset, C., & Georgiev, L. 2012, in IAU Symp., 283, 350
Esteban, C., Carigi, L., Copetti, M. V. F., et al. 2013, MNRAS, 433, 382
Esteban, C., García-Rojas, J., Carigi, L., et al. 2014, MNRAS, 443, 624
Ferland, G. J. 1999, PASP, 111, 1524
Ferland, G. J., Korista, K. T., Verner, D. A., et al. 1998, PASP, 110, 761
Fitzpatrick, E. L. 1999, PASP, 111, 63
Fitzpatrick, E. L., & Massa, D. 1988, ApJ, 328, 734
García-Rojas, J., Peña, M., Morisset, C., Mesa-Delgado, A., & Ruiz, M. T. 2012, A&A, 538, A54
García-Rojas, J., Peña, M., Morisset, C., et al. 2013, A&A, 558, A122
García-Rojas, J., Simón-Díaz, S., & Esteban, C. 2014, A&A, 571, A93
Girard, P., Köppen, J., & Acker, A. 2007, A&A, 463, 265
Gordon, K. D., Clayton, G. C., Misselt, K. A., Landolt, A. U., & Wolff, M. J. 2003, ApJ, 594, 279
Ho, I., Kewley, L. J., Dopita, M. A., et al. 2014, MNRAS, 444, 3894
Howarth, I. D. 1983, MNRAS, 203, 301
Hummer, D. G., & Storey, P. J. 1987, MNRAS, 224, 801
Izotov, Y. I., Stasińska, G., Meynet, G., Guseva, N. G., & Thuan, T. X. 2006, A&A, 448, 955
Kaler, J. B. 1976, ApJS, 31, 517
Kameswara Rao, N., Lambert, D. L., García-Hernández, D. A., & Manchado, A. 2013, MNRAS, 431, 159
Kingsburgh, R. L., & Barlow, M. J. 1994, MNRAS, 271, 257
Kwitter, K. B., & Henry, R. B. C. 2001, ApJ, 562, 804
Liu, X.-W., Storey, P. J., Barlow, M. J., et al. 2000, MNRAS, 312, 585
Luridiana, V., Morisset, C., & Shaw, R. A. 2012, in IAU Symp., 283, 422
Luridiana, V., Peimbert, A., Peimbert, M., & Cerviño, M. 2003, ApJ, 592, 846
Luridiana, V., Simón-Díaz, S., Cerviño, M., et al. 2009, ApJ, 691, 1712
Marino, R. A., Rosales-Ortega, F. F., Sánchez, S. F., et al. 2013, A&A, 559, A114
Menzel, D. H. 1937, ApJ, 85, 330
Mesa-Delgado, A., Esteban, C., García-Rojas, J., et al. 2014, ApJ, 785, 100
Morisset, C., & Georgiev, L. 2009, A&A, 507, 1517
Nicholls, D. C., Dopita, M. A., Sutherland, R. S., Kewley, L. J., & Palay, E. 2013, ApJS, 207, 21
Nicholls, D. C., Dopita, M. A., Sutherland, R. S., et al. 2014, ApJ, 786, 155
O'Donnell, J. E. 1994, ApJ, 422, 158
Osterbrock, D. E. 1989, Astrophysics of gaseous nebulae and active galactic nuclei (University Science Books)
Osterbrock, D. E., & Ferland, G. J. 2006, Astrophysics of gaseous nebulae and active galactic nuclei, 2nd edn. (University Science Books)
Peimbert, M., & Costero, R. 1969, Boletin de los Observatorios Tonantzintla y Tacubaya, 5, 3
Peimbert, M., Torres-Peimbert, S., & Ruiz, M. T. 1992, Rev. Mex. Astron. Astrofis., 24, 155
Pérez-Montero, E. 2014, MNRAS, 441, 2663
Pérez-Montero, E., Hägele, G. F., Contini, T., & Díaz, Á. I. 2007, MNRAS, 381, 125
Proxauf, B., Öttl, S., & Kimeswenger, S. 2014, A&A, 561, A10
Rodríguez, M., & Rubin, R. H. 2005, ApJ, 626, 900
Sankrit, R., Raymond, J. C., Bautista, M., et al. 2014, ApJ, 787, 3
Savage, B. D., & Mathis, J. S. 1979, ARA&A, 17, 73
Seaton, M. J. 1979, MNRAS, 187, 73P
Shaw, R. A., & Dufour, R. J. 1995, PASP, 107, 896
Shaw, R. A., de La Pena, M. D., Katsanis, R. M., & Williams, R. E. 1998, in Astronomical Data Analysis Software and Systems VII, eds. R. Albrecht, R. N. Hook, & H. A. Bushouse, ASP Conf. Ser., 145, 192
Stasińska, G. 1978, A&A, 66, 257
Storey, P. J., & Hummer, D. G. 1995, MNRAS, 272, 41
Torres-Peimbert, S., & Peimbert, M. 1977, Rev. Mex. Astron. Astrofis., 2, 181