Molecular Abundances Bar Plotter
(abundsbarplot.py)
User Guide

Andrés Megías Toledano
Centre for Astrobiology (CAB), Madrid

Version 1.6
February 2023

Index

1. Introduction

The Molecular Abundances Bar Plotter is a Python 3 script, `abundsbarplot.py`, that can create a bar plot for representing the molecular abundances of a list of species, for different sources, and for up to two positions per source. Additionally, it can compute the mean abundance and the mean molecular mass for the observed species, and the similarity between the different sources, if there is more than one.

The script requires the libraries PyYAML, NumPy, Pandas, Matplotlib, and RichValues.[1] In order to run the code, it is necessary to specify some options in a configuration file. For running the code we have to write the following command line in the terminal, being in the folder of the file `abundsbarplot.py`:

```
python3 abundsbarplot.py <path>
```

The argument `<path>` is the path of the configuration file. If we do not specify it, the script will search in the current folder (`./`) for the configuration file specified inside the script in the variable `config_file`, at the beginning of the code; we can use this variable to run the script in a programming environment like Spyder. If the script `abundsbarplot.py` is not in the current folder, but in a folder with path `<folder>`, we should write `<folder>/abundsbarplot.py`

1. https://github.com/andresmegias/richvalues

instead of abundsbarplot.py. Moreover, if the script is included in a folder of executable files, the term python3 can be removed from the command line, and the script will be run from any path.

2. Input files

We need a plain text document in .csv format containing the molecular abundances for each source and position observed.

The structure of the abundances files should be as follows. There should be two columns, one named molecule and another called abundance. In the first one, you should put the name of each species. It can be written without any complex format like LaTeX one, as the numbers will be automatically set as subscripts, and the numbers preceded by a hat symbol (^) will be automatically set as superscripts. In the second column, you should put the value of the abundance of the corresponding molecule. It must be set in a specific format. If the value has no uncertainty, just put the number. If it has uncertainty, you can join the central value and the uncertainty with +/-; for example: `5.2 +/- 0.4`. It it has a lower and an upper uncertainty, you should write the lower uncertainty just after the central value preceded by -, and then write the upper uncertainty preceded by +; for example: `5.2-0.3+0.4`. If you want to use scientific notation (exponential notation with decimal base), for the central value and the uncertainty (or uncertainties), you just have to write an e, separated by a space from the numbers and followed by the exponent argument; for example: `5.2-0.3+0.4 e-3`. Lastly, if you want to specify an upper limit, just write a < before the value, separated by a space (and without uncertainty); for example: `< 5.2 e2`.

3. Output files

The resulting file will be an image in PDF format of the bar plot of the abundances of the given molecules.

4. Configuration file

The configuration file is a plain text document with the YAML formatting style. This way, we can specify the options of the script abundsbarplot.py by defining different variables that can be nested.

In order to define the values of the variables, we have to write, in a single line, the name of the variable followed by a colon (:), and its value, separated with a space. For the logical variables, the possible values are yes/no. There are variables whose value can be a list of elements, in which case they can be written between brackets and separated by commas or in single lines preceded by a hyphen (–). Similarly, in the case of nested variables, which work like Python dictionaries, next level variables must be written in a different line and with an indent. Text variables can be written between simple quotation marks ('), but this is optional.

If the definition of a variable is not written,[2] its default value will be used (see Section 3).

Below is a list of all the variables with their meaning and possible values:

- **input files**. Nested variable for specifying the names of the input files. It should be a list containing a next level variable whose names are the file names of the abundances of the molecules for each source, so that a bar will be plotted for each source, with de different bars horizontally-aligned. Then, each of these variables should contain two next level variables:
  - **name**. Name of the source to be displayed on the legend of the plot.
  - **color**. Color used to plot the bar, from Matplotlib.

  Moreover, each element of this list of input files can have two elements instead of one, that is, it can have two sources that will be treated as a group. The abundances for these two sources will be plotted as different bars, but vertically-aligned, plotting the one with a lower value in front of the other one, for each molecule.
- **output file**. Name of the PDF file containing the created plot.
- **bar spacing**. Horizontal spacing between the bars from different molecules. This will determine the width of each bar, being this last one shorter as the spacing increases. Its value must be between 0 and 1, with 0 meaning no distance between bars from different molecules and 1 corresponding the limit case of null bar width.
- **point spacing**. In case there are two sources for each group of sources (that is, that there are two bars vertically-aligned, this is the horizontal separation between the points that indicate the value of the abundances, at the top of each bar. This is used to differentiate the uncertainties from both sources when the central values are too close. The units are such that 1 is the width of the bars, so the value of this variable must be between 0 and 1.
- **figure size**. Dimensions of the window containing the plot (width $\times$ height), in inches. It should be a list containing the width and the height, in this order.
- **font size**. Size of the font used in the plot, in points.
- **legend position**. Coordinates of the position of the centre of the legend, relatives to the size of the plot box (so that 0 is the start and 1 is the end of each axis). It should be a list containing the horizontal and vertical coordinates, in this order.
- **legend font size**. Size of the font used in the legend of the plot, in points. By default it will be the 90 % of the value of font size.
- **inferior limit**. Inferior limit in the vertical axis, that is, the abundance, in the same units as the data in the input files.
- **similarity type**. Text variable that specifies the function used to compute the similarity between the different sources, if the variable compute similarity is set to yes. The similarity will always range between 0 (both sources are not similar at all) and 1 (both sources are identical). To compute it, independently of the kind of similarity, the script

2. Here, 'not written' means that neither the variable name nor its value are written in the configuration file. If the variable name is written (and the colon) but its value is not, the corresponding Python variable will get the value None.

builds a vector for each source that contains the abundance of each molecule for each position (for every common position between the sources), so that the total number of entries is the number of molecules times the number of common positions. The three possible values of this variables are:

  - **fractional**. Fractional similarity, defined as the number of molecules for each position that have the same state of detection (detected or non-detected) in both compared sources divided by the product of the total number of molecules and the number of positions.
  - **cosine**. Cosine similarity, which is the scalar product of the normalized abundance vector for all the species and positions for both sources.
  - **binarized cosine**. Cosine similarity but applied to the abundance vectors with binary entries representing the state of detection (1 if detected, 0 if not).
  - **angular**. Angular similarity, which is 1 minus the normalized arccosine of the cosine similarity, so that the arccosine ranges between 0 and 1.
  - **binarized angular**. Angular similarity but applied to the abundance vectors with binary entries representing the state of detection (1 if detected, 0 if not).

  That is, for the fractional, binarized cosine and binarized angular similarities, the vectors used are constructed from the corresponding abundance vectors, and have binary entries which are 1 if the molecule is detected (in the corresponding position) and 0 if not.
- **compute similarity**. This logical variable is only useful when there is more than one source. If it is set to yes, the script will calculate the kind of similarity specified in the variable similarity type between the molecular abundances for every pair of sources. This value is 0 if both sources are totally different and 1 if they are equal.
- **compute mean abundance**. If this logical variable is set to yes, the script will compute the geometric mean of the abundances of all the species weighted with their molecular masses, for each source and position.
- **compute mean molecular mass**. If this logical variable is set to yes, the script will compute the mean molecular mass for all the species weighted with their abundances in logarithmic scale, foe each source and position.
- **use upper limits for calculations**. Logical variable that determines if the observations with upper limits are used for calculating the similarity and for the mean abundance and molecular mass for the observed species. If so, for the calculations upper limits will be represented as uniform distributions between 0 and the value of the upper limit.

Default variable values

Below are the default values of the variables of abundsbarplot.py, that is, the values that they will take if they are not declared in the configuration file.

```
input files: []
output file: 'barplot.pdf'
figure size: [12, 8]
font size: 12
legend font size: null
```

```
bar spacing: 0.25
point spacing: 0.5
legend positions: []
inferior limit: null
similarity type: 'fractional'
compute similarity: no
compute mean abundance: no
compute mean molecular mass: no
use upper limits for calculations: yes
```

Note that this default values will only be used if neither the variable name nor its value are written in the configuration file. If the variable name is written (and the colon) but its value is not, the corresponding Python variable will get the value `None`.

5. Example case

Here we show a brief example to see how to use the script `abundsbarplot.py`. In this example we will plot the abundances of three different sources (L1544, L1498, and L1517B). At the same time, for each source there are two different positions, so there will be three groups of bars, horizontally-aligned, and each of those groups will have two bars, vertically-aligned.

5.1. Input files

As we have three sources and two positions for each source, we will need 6 files containing the abundances of each species.

Below is the content of one of the input abundances list files for the previous configuration file (`L1517B-methanol-all.csv`), that is, for one of the positions of one of the sources. It was generated by the SLIM Table Generator (`slimtables.py`), which is part of the MADCUBA-SLIM Python Scripts.[3]

molecule	abundance
CH3OH	1.160-0.107+0.131 e-9
CH3O	1.91-0.62+0.68 e-11
CH3OCHO	< 1.33 e-10
CH3OCH3	< 6.3 e-11
CH3CHO	< 2.15-0.44+0.49 e-11
t-HCOOH	< 2.6 e-10
c-C3H2O	< 6.6 e-12
H2CCO	7.55-2.08+2.23 e-11
CCCO	< 1.59 e-10
HCCCHO	< 2.6 e-11
HCCNC	1.89-0.81+0.93 e-11
CH2CHCN	< 4.2 e-12
CH3NC	< 3.0 e-12
CH3CN	< 1.89 e-11

3. https://github.com/andresmegias/madcuba-slim-scripts

HCCCN	3.88-0.34+0.43 e-10

5.2. Configuration file

Below is an example of configuration file

```
input files:
- L1544-methanol-all.dat:
    name: 'methanol peak'
    color: 'tab:red'
  L1544-center-all.dat:
    name: 'center'
    color: 'yellow'
- L1498-methanol-all.dat:
    name: 'methanol peak'
    color: 'dodgerblue'
  L1498-center-all.dat:
    name: 'center'
    color: 'cyan'
- L1517B-methanol-all.dat:
    name: 'methanol peak'
    color: 'darkorchid'
  L1517B-center-all.dat:
    name: 'center'
    color: 'orchid'
bar spacing: 0.25
point spacing: 0.06
figure size: [10, 7]
legend position: [0.5, 1.2]
inferior limit: 1e-13
similarity type: 'fractional'
compute similarity: yes
compute mean abundance: yes
compute mean molecular mass: yes
use upper limits for calculations: yes
```

5.3. Bar plot

Figure 1 shows the resulting bar plot for this example. For each molecule, there are three horizontally-aligned bars, one for each of the sources, but at the same time there are actually two bars, vertically-aligned, for each source, corresponding to two different positions in the source.

Upper limits are indicated with arrows at the top of the bar as well as with stripes on it, for the sake of readability. For the same reason, darker colors were used to represent one of the positions in the sources, and lighter colors were used for the other position.

Note that the horizontal spacing between the groups of bars for each molecule (`bar spacing`) should not be null, as the horizontal axis does not correspond to a continuous variable (like in a histogram).

5.4. Additional calculations

Below are the output of the calculations of the fractional similarity, the mean abundance and

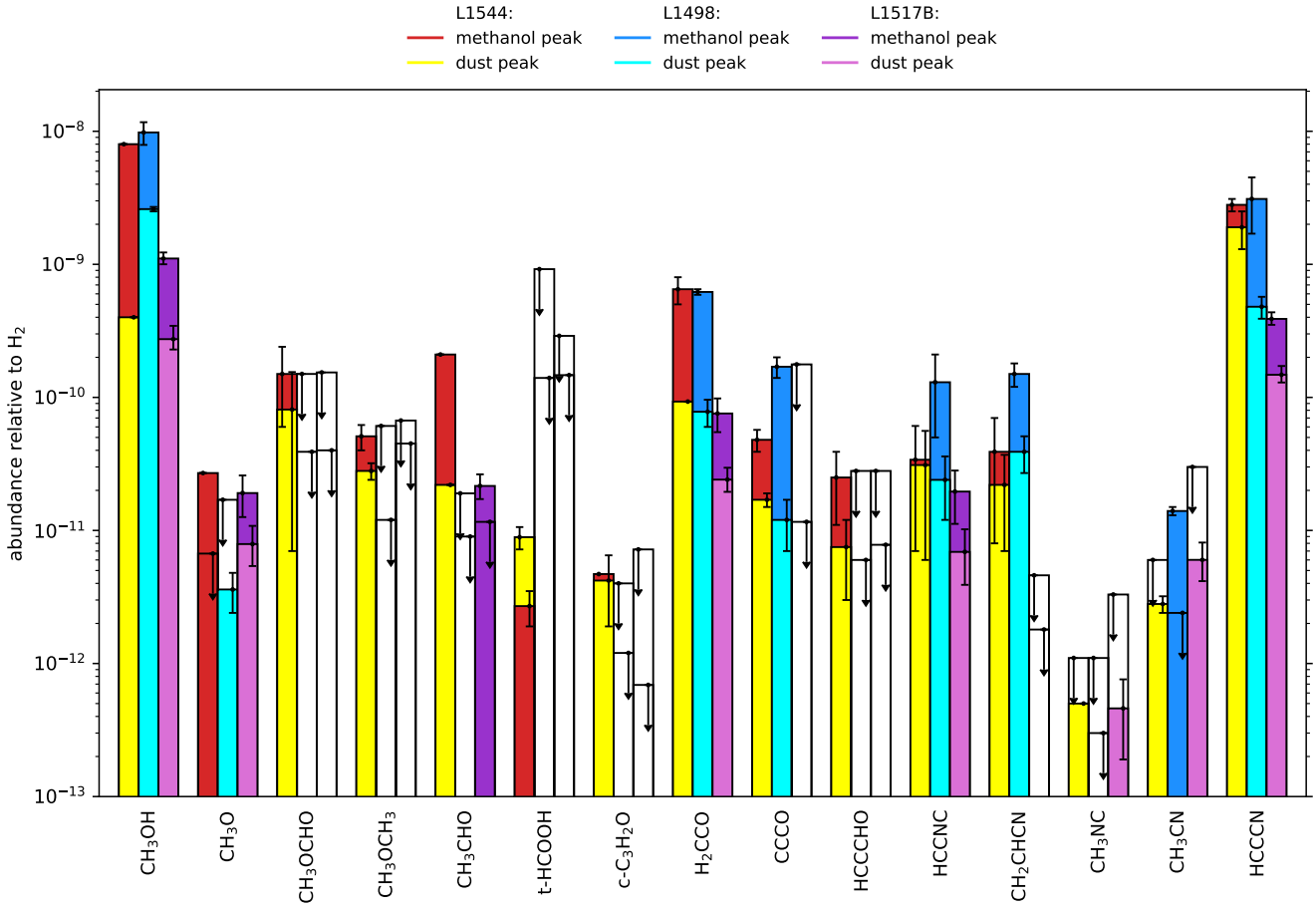

Figure 1. Abundance relative to H_2 of different molecules for the three pre-stellar cores L1544, L1498, and L1517B, towards the core's center and the methanol peak. Lower limits are represented with arrows at the top of the bar as well with strips on it.

the mean molecular mass as they are displayed on the terminal when the script is run.

```
Fractional similarity.
- L1544-L1498: 0.433+/-0.033
- L1544-L1517B: 0.467+/-0.033
- L1498-L1517B: 0.700+/-0.033
Mean abundance.
L1544 - methanol peak: 4.2+/-0.7 e-11
L1544 - dust peak: 1.9-0.4+0.3 e-11
L1544 - (mean): 3.0+/-0.4 e-11
L1498 - methanol peak: 5.2+/-1.0 e-11
L1498 - dust peak: 1.05+/-0.21 e-11
L1498 - (mean): 3.2+/-0.5 e-11
L1517B - methanol peak: 2.3+/-0.5 e-11
L1517B - dust peak: 6.6+/-1.4 e-12
L1517B - (mean): 1.46-0.24+0.25 e-11
Mean molecular mass. (g/mol)
L1544 - methanol peak: 46.13-0.19+0.16
L1544 - dust peak: 46.8-0.4+0.3
L1544 - (mean): 46.46-0.20+0.15
L1498 - methanol peak: 46.32-0.24+0.22
L1498 - dust peak: 46.18-0.24+0.18
L1498 - (mean): 46.25-0.17+0.14
L1517B - methanol peak: 45.7-0.4+0.3
L1517B - dust peak: 45.4-0.4+0.3
L1517B - (mean): 45.53-0.27+0.21
```

Useful links

The following links may be of interest:

- **MADCUBA.**
  https://cab.inta-csic.es/madcuba/
- **MADCUBA-SLIM Python Scripts.**
  https://github.com/andresmegias/madcuba-slim-scripts
- **Python.**
  https://www.python.org/
- **PyYAML.**
  https://pyyaml.org/wiki/PyYAMLDocumentation/
- **NumPy.**
  https://numpy.org/
- **Matplotlib.**
  https://matplotlib.org/
- **RichValues.**
  https://github.com/andresmegias/richvalues/

Credits

This software has been developed at the Centre for Astrobiology (*Centro de Astrobiología*, CAB), in Madrid (Spain), within the group of Chemical Complexity in the Interstellar Medium and Star Formation (Department of Astrophysics).

Coding and testing

Andrés Megías Toledano

Supervising

Izaskun Jiménez Serra

License