Summary of the Galaxy comparison code

Francisco Machado

May 31, 2014

1 Introduction

The goal of this document is to provide an understanding of the code written to do the comparisons between galaxies. I will cover each file and explain how the input is taken and then the workings of the program. For each file, by running the program in the following manner:

```
python3 script_name.py -h
```

or:

```
python3 script_name.py --help
```

Gives a small description of the file as well as the list of arguments it requires.

2 Format_galaxies.py

The purpose of this file is to preprocess the Illustris simulation galaxy's SED, transforming the data from Luminosity per wavelength into AB magnitude at a distance of 10 pc.

2.1 Input

The input for this program is:

```
python3 Format_galaxies.py Input_Folder Output_Folder
```

The Input_Folder is the origin place of the files and the Output_Folder is the directory where the modified files will be stored.

The program runs through every .txt file and runs the appropriate transformations. Right now it assumes that the file consists **only** on two columns, the first one containing the wavelength (λ) and the other the Luminosity per wavelength (L_λ) at a distance of 50 Mpc in SI units (Wm^{-1}).

Moreover the code also assumes that when a row is split by the spaces (' ') using the string.split(' ') method from python, the second column corresponds to the FILE_SECOND_COLUMN position in the resulting array. Changes in the input file format must be taken into consideration by changing this value.

2.2 Data processing

To turn the data into AB magnitudes the program follows the following steps.

Firstly the value of Luminosity per wavelength is transformed to luminosity per frequency (L_ν), using the equation:

$$L_\nu = L_\lambda \frac{\lambda^2}{c} \tag{1}$$

This value will have units of WHz^{-1}

Then the luminosity is transformed into flux (F_ν) at a radius of $R = 10\,\mathrm{pc}$, by dividing it by the area of a sphere of such radius.

$$F_\nu = L_\nu \frac{1}{4\pi R^2} \tag{2}$$

F_ν will now have units of $\mathrm{WHz}^{-1}\mathrm{m}^{-2}$

Then we make the convertion into flux in microJanksys (S_ν):

$$S_\nu = F_\nu \times 10^{32} \tag{3}$$

The units of S_ν are $\mu\,\mathrm{Jy}$.

Then we use the formula to calulate the AB magnitude from the flux in $\mu\,\mathrm{Jy}$:

$$m_{AB} = 23.9 - \log_{10}(S_\nu) \tag{4}$$

m_{AB} will not have units but determines the apparent AB magnitude of the galaxy.

Since we calculate the flux at a distance of 10 pc this apparent magnitude matches the absolute magnitude (M_{AB}).

$$M_{AB} = m_{AB} \tag{5}$$

For the remainder of the article let's define the absolute magnitude of a simulation galaxy as M_S.

2.3 Saving the Data

The code then saves each file into a new file of the same name of the original on the Output_Folder. Each file consists on two columns separated by a space (' ') where the first column is the wavelength (λ) and the second will be the absolute AB magnitude for that wavelength (M_{AB}).

3 FastComp.py

This program has compares an input galaxy with a set of galaxies, by their absolute AB magnitude, using the χ^2 minimization method. It returns a file with the value of χ^2 for every galaxy in the comparison set, as well as drawing a graph and histogram with the same information.

3.1 Input

The input for this program is:

```
python3 FastComp.py Data_Folder Input_File_1 Input_File_2 ... Input_File_N
```

The Input_File_i is a file containing the information about the galaxy we wish to compare with our set. The file must be formated in the following way. The first line must contain a single number which will correspond to the redshift of the galaxy. The remainder of the file must contain 3 columns, which will be the wavelength (λ), the flux in μ Jy (S_ν), and the error in the flux (δS_ν) also in units of μ Jy. Each of the columns is separated by a space (' ').

The Data_Folder is a folder containing the galaxy files from the simulation. These are the galaxies that have been preprocessed by the Format_galaxies.py script.

3.2 Data Pre-Processing

The following description is the procedure followed by the program for all the files.

The program starts by reading the information of the file and storing the redshift on the variable called (appropriately) **redshift**. The remaining values are stored on an dictionary of arrays where the 'wl' corresponds to the wavelengths, 'fl', the fluxes and 'err' the error in the flux.

Then the distance in Mpc (D_{Mpc}) is calculated using the cosmocalc package, using the following command (included to specify the constants):

```
(cosmocalc.cosmocalc(redshift, H0=70.4, WM=0.2726, WV=0.7274))['DL_Mpc']
```

Then the distance is transformed into pc, (now D_{pc}) and used to calculate the distance modulus (μ) which will then be used for calculating the variation in the magnitude of the Input_File.

$$\mu = 5 \cdot (\log_{10}(D_{pc}) - 1) \tag{6}$$

Now for each data point in the Input_File_i we tranform the flux (S_ν) into an absolute AB magnitude (M_{AB}) (AB magnitude measured at a distance of 10 pc). To do this, the flux is firstly used to calculate the apparent AB magnitude (m_{AB}) and then we make use of the modulus distance (μ) to transform from apparent to absolute.

$$m_{AB} = 23.9 - 2.5 \cdot \log_{10}(S_\nu) \tag{7}$$

$$M_{AB} = m_{AB} - \mu \tag{8}$$

The error will also be propagated.

$$\delta m_{AB} = 2.5 \frac{\delta S_\nu}{S_\nu}$$

$$\delta M_{AB} = \delta m_{AB}$$

We are **not taking into account error in the redshift.** An improvement may be done on this part.

For the remainder of the articles let's define the absolute magnitude of an input galaxy as M_I, and its error as δM_I.

3.3 Comparison Method

To compare the Input_File_i with the set of galaxies we use the χ^2 minimization method by comparing to every galaxy in our set and select the galaxies which present the least value of χ^2.

We then open a simulation galaxy file and for every input file we calculate the χ^2 between the input file and that galaxy file. This way we do not open a simulation galaxy file more than once.

If the values of the wavelengths match with the ones we have for the set of galaxies, then we simply calculate the χ^2 term directly:

$$\chi^2 = \frac{1}{N}\sum_{k=1}^{N}\frac{(M_S - M_I)^2}{\delta M_I} \tag{9}$$

If the wavelengths do not match, then we must do an interpolation between the two neighboring points to gather the correct value we should use for M_S.

Let λ_S^+ and λ_S^- be the closest values of the wavelengths to λ_I such that $\lambda_S^+ > \lambda_I > \lambda_S^-$. Also let M_S^+ and M_S^- be the values of the absolute magnitude of the simulation galaxy for wavelengths λ_S^+ and λ_S^- respectively.

Then we can define an interpoled value for the absolute magnitude $M_S^{interpol}$ and then do the comparison with that term.

$$M_S^{interpol} = \frac{M_S^+ - M_S^-}{\lambda_S^+ - \lambda_S^-}(M_I - M_S^-) + M_S^- \tag{10}$$

$$\chi^2 = \frac{1}{N}\sum_{k=1}^{N}\frac{(M_S^{interpol} - M_I)^2}{\delta M_I} \tag{11}$$

Note, if there are values of wavelengths in the input file greater than the largest wavelength in our simulated galaxys' files or smaller than the smallest one in our simulated galaxys' files, then we have no way of doing a precise comparison with those and they are disregarded.

3.4 Saving the Data

The program then saves the resulting χ^2 of all the file comparisons to a different file in the Results folder for each input file. Each file is composed of two columns, one is corresponds to a simulated galaxy and the other one is the χ^2 of the comparison of the input file with that simulated galaxy. The values are ordered by the χ^2 for quick reference.

4 Fits_to_File.py

The goal of this program is to transfrom the .fit from the Gama Survey data set to text files with a formating that is accepted by the FastComp.py program.

4.1 Input

The input for this file is:

```
python3 Fits_to_file.py Input_Folder Output_Folder
```

The Input_Folder is the folder contaning the fit files we wish to process to text files. The Output_Folder is the folder where the resulting files are saved.

4.2 Library used

To access the information I am using the library pyfits which allows for easy acess to a fit file.

4.3 Data Processing

First let's list the assumptions on the fit file formating:

- There is only one part in the file, meaning only one header and only one set of data.
- The redshift label on the header is 'Z'
- The range of the wavelengths is given by the headers 'WMIN' and 'WMAX', in angstroms (A).
- **The wavelengths are uniformly distributed in this range**
- The first row or first array of data corresponds to the flux in units of $10^{-17}\,\mathrm{erg}\cdot\mathrm{s}^{-1}\cdot\mathrm{cm}^{-2}\cdot\mathrm{A}^{-1}$
- The second row or third array is the error associated with the flux in the same units.

The file starts by reading the value of the redshift and creating the possible values of the wavelengths, based on the limits in the header.

Then for each possible value of wavelength the program transforms it to SI units (meters) and reads and transforms the flux and error associated with that wave length.

The transformations of the values is done as follows (let R_λ be the value of the flux for a wavelength λ and δR_λ its error):

$$L_\lambda = R_\lambda \cdot 10^{-17} \cdot 10^{10}$$
$$\delta L_\lambda = \delta R_\lambda \cdot 10^{-17} \cdot 10^{10}$$

This transforms the units from $10^{-17}\,\mathrm{erg}\cdot\mathrm{s}^{-1}\cdot\mathrm{cm}^{-2}\cdot\mathrm{A}$ to $\mathrm{erg}\cdot\mathrm{s}^{-1}\cdot\mathrm{cm}^{-2}\cdot\mathrm{m}^{-1}$

$$L_\nu = L_\lambda \frac{\lambda^2}{c}$$
$$\delta L_\nu = \delta L_\lambda \frac{\lambda^2}{c}$$

This transforms flux per wavelength ($\mathrm{erg} \cdot \mathrm{s}^{-1} \cdot \mathrm{cm}^{-2} \cdot \mathrm{m}^{-1}$) to flux per frequency ($\mathrm{erg} \cdot \mathrm{s}^{-1} \cdot \mathrm{cm}^{-2} \cdot \mathrm{Hz}^{-1}$)

$$S_\nu = L_\nu \cdot 10^{29} \ \delta S_\nu = \delta L_\nu \cdot 10^{29}$$

This transforms the value from $\mathrm{erg} \cdot \mathrm{s}^{-1} \cdot \mathrm{cm}^{-2} \cdot \mathrm{Hz}^{-1}$ to $\mu\,\mathrm{Jy}$.

4.4 Saving data

The program then for each file, saves a file in the output folder with the same name but .txt extension containing:

- First line containing the redshift of the galaxy
- The remainder consists of lines with 3 values separated by a space containing the wavelength of light, the flux of the light in μJy and the error of the flux in the same units

5 Comp_vec.py

This script corresponds to an improvement of the FastComp.py file. Instead of using arrays and running through all of them for each calculation, the data is vectorized as much as possible. This way we obtain faster and cleaner code. The only part that was not vectorized was the interpolation which is still done one element at a time.

5.1 Input

Like the FastComp.py file, the input is

```
python3 Comp_vec.py Data_Folder Input_File_1 Input_File_2 ... Input_File_N
```

5.2 Notes

All of the assumptions and calculations done in FastComp.py are reproduced here, so in case of any questions reference back to it.

A difference is that now all of the input is stored in the same structure, a dictionary named **data** with entries **'red'**, **'wl'**, **'fl'** and **'err'** which store the redshift, and array of the wavelengths, and array of the corresponding fluxes, and an array of the corresponding errors, respectively.

6 Comp_vec_norm.py

The purpose of this script is to find the best fit galaxy up to a constant factor, meaning that each inpu galaxy is multiplied by the constant factor that minimizes the χ^2 and then the χ^2 is calculated. The Illustris galaxy with the least χ^2 is then chosen as the best fit.

Input

The input for this file is the same as for Comp_vec.py and FastComp.py.

```
python3 Comp_vec_norm.py Data_Folder Input_File_1 Input_File_2 ... Input_File_N
```

Data Processing

The data processing methods are the sames as in FastComp.py, but with vectorized structures. Please refer to the FastComp.py section to read the details.

Data Comparison

In this file the data comparison is different from the other files. Like normally we load the input files and then go through each Illustris galaxy file and compare the entirety of the input galaxies.

However now instead of comparing directly the SED's, first it is found the constant such that multiplied by all of the data in the input galaxy minimizes the value of χ^2.

To find the value of the constant A we minimize χ^2 in order to it:

$$\chi^2 = \sum \frac{(AM_I - M_S)^2}{A\delta M_I} \tag{12}$$

$$\frac{\partial \chi^2}{\partial A} = 0 = \sum \frac{2M_I(M_I - M_S)A\delta M_I - (AM_I - M_S)^2 \delta M_I}{A^2 \delta M_I^2} \tag{13}$$

$$0 = \sum \frac{2A^2M_I^2 - 2AM_IM_S - A^2M_I^2 + 2AM_IM_S - M_S^2}{\delta M_I} = A^2 \sum \frac{M_I^2}{\delta M_I} - \sum \frac{M_S^2}{\delta M_I} \tag{14}$$

$$A = \sqrt{\frac{\sum \frac{M_S^2}{\delta M_I}}{\sum \frac{M_I^2}{\delta M_I}}} \tag{15}$$

Both values are then multiplied by A and the χ^2 is calculated.

$$\chi^2 = \sum \frac{(AM_I - M_S)^2}{A\delta M_I} \tag{16}$$

The results are then ordered by χ^2, like in the other files and saved into a file for each input galaxy.

This file was not expanded very throughouly so the values of A are not save in the results file along with the χ^2. It should be implemented in the future.

7 Comp_Bay.py and Prof_Comp_Bay.py

In this script we shift the comparison measure from the χ^2 test to a baysean approach. We are no longer interested in fitting a galacy, but recovering the properties of the inputed galaxy from the Illustris galaxies.

Input

Besides the parameters required for the best fitting scripts, this script also requires a file which specifies the various properties of the Illustris galaxies. At this moment only metalicity and stellar mass are implemented, but more properties can be introduced without much complexity added to the code.

```
python3 Prof_Comp_Bay.py data_folder properties_file results_folder \
input_file_1 ... input_file_N
```

Data Processing

The process is very similar to the best fit scripts. The input galaxies are opened and stored into a dictionary. Then the properties file is opened and their are stored in a dictionary with 1 entries which specifies the number of the Illustris galaxy. To this key it is associated an array with the properties of the galaxy. In position 0 is the stellar mass and in position 1 is the metalicity.

The rest of the data processing to the galaxies is the same as the best fit files.

8 GAMA_Scraper.py

This script downloads the fit files from the galaxys from a GAMA query file. It downloads both the .fit file as well as obtaining the properties of the galaxies, which are then all saved into a single file.

8.1 Input

The input of this script is the output file from a query to the GAMA survey database.

```
python3 GAMA_Scraper.py input_file output_file
```

8.2 Inner workings

The script works by assuming all of the webpages of the galaxies are of the form:

```
http://www.gama-survey.org/dr2/tools/sov.php?cataid=(#id)
```

Where the parenteses part corresponds to the unique ID for each galacy. By accessing the html file from this we can obtain the download link for the .fit file.

This is done by finding the bit of html code before the download link and then from that get the download link aftwards. Then the script downloads the file to results_folder.

To obtain the properties of the galaxy, the script access a different page which contains all the properties of interest of the galaxy. From there, since the information is coded on the html file, we can look for the name of the property and from there obtain the value itself.

The url for this page is:

```
http://www.gama-survey.org/dr2/tools/querytab.php?tab=StellarMasses&cataid=(#id)
```

This is done for every property we are interested and saves them all into a file with the galaxy id, followed by the various properties.

9 Trim_prop_file.py

This files removes all of the galaxies in the properties file with invalid properties, hence galaxies that are of no interest to us, saving the ones of interest to a new file.

9.1 Input

The input for this script is a file with the properties from the Illustris simulation.

```
python3 Trim_prop_file.py properties_file.py
```

9.2 Inner Workings

This is a simple script, it simples goes through each line of the file and check if any of the properties is invalid (either 0.0 or -inf).

If it is valid is adds to an array, if not, that value is disregarded. The entries on the array are then printed to a file with the name "trim_aux_properties.txt" on the same directory as the original file.

A possible improvement would be to include the result file as a parameter.

10 Get_N_Gal_from_trim.py

From the trimmed properties file, it gets obtains information from N valid galaxies and copies their information into a folder. Not only is the a new properties file created but also the galaxy files are copied.

10.1 Input

The script accepts the folder with the Illustris galaxy data, the number of galaxies we want (N), the trimmed properties file, and a target folder for the galaxy and the properties file.

```
python3 Get_N_Gal_from_trim.py data_folder N trimmed_file target_folder target_properties_folder
```

10.2 Inner Workings

Firstly the file reads the trimmed properties file, saving the galaxies properties into arrays within a dictionary.

Then the id array is transversed and the first N available galaxies have their galaxy file copied into the new directory and their properties written to the new file.

11 Bay_MPI.py

Implementation of the baysean comparison using the MPI protocol for multi threaded running. The theoretical part is the same as the Comp_Bay.py file, however the implementation is different. In this file, the transformation from raw Illustris galaxy files to formatted ones is implemented.

11.1 Input

The input of this file is very similar to the Comp_Bay.py file however instead of the formatted galaxy folder, it expects the raw data folder. Moreover it expects, not a properties file containing only the properties from the galaxies we are comparing, but the entire trimmed file. Notice, also, that we need to include the MPI initialization code.
The file run.sh contains the correct argument to run the code with a certain amount of threads.

11.2 Descriptions of each function

The code is seperated into the functions to make the code more modular for the paralelization.

11.2.1 LoadSimFile

This function loads the galaxy file given a galaxy file name (data_file) and a number (gal_num) which will correspond to the position of the galaxy's information in the array.

It opens the file, reads the information and transforms it using the same calculations from the Format_galaxies.py file. The information is then store it in the array at the position gal_num.

11.2.2 loadProperties

This function is responsible for opening, reading and storing the properties of the Illustris galaxies. Then, based on the galaxies that have been read, it constructs a properties array containing only the information from the Illustris galaxies used.

11.2.3 Comp

The function that does the comparison of a given galaxy (file_name) against the galaxy files loaded.

The function opens the galaxy file that was given as an argument and loads the data from there, ensuring that all the information is valid.

Then the fluxes and the errors are transformed using the Transform function which performs the same transformation of the data as in the other comparison scripts.

Then the function creates a interpolation base. This corresponds to precomputing the values which will be involved in the interpolation of the data, meaning that, since all of the Illustris galaxies have the same wavelengths, we can pre-calculate between which values of the Illustris wavelengths do the wavelengths of the input galaxy are.

Then we can use this base, in the Calculate_Inter, to calculate the interpolation of a Illustris galaxy on the values of the wavelengths of the input galaxy.

Then the χ^2 is calculated, using the interpolation, the input galaxy information and its error. From there a baysean weight is calculated and used to calculate the best estimation for the properties of the input galaxy.

11.2.4 MPI

The paralelized implemention of the code goes as follows. First, the MPI communication is set between all of the threads. Then the loading of the Illustris galaxy informations is divided between the various threads.

After that all of the information is shared between the threads, and a single thread is chosen to load the properties of those galaxies and share them with every galaxy.

From there, the galaxies are divided among the threads, and each thread calculates the properties for the galaxies that have been assigned to it.

Once that task is complete, all of the information is joined and then a single task is responsible to write the information onto a file.