FOAMS User Guide

Fitting **O**bserved **A**bundances to **M**etals from **S**upernovae

The FOAMS program takes a user input of abundance observations from celestial objects and fits those abundances using simulated nucleosynthesis models from supernovae and AGB stars. The fitted parameters are the amount of star formation and ratio of Type-Ia to core-collapse supernovae.The fraction of stars between 1 and 8 solar masses that have become Type-Ia supernovae is also returned. Fits to the input data are performed for all possible combinations of user-selected sets of models for Type-Ia supernovae, core-collapse supernovae, and AGB stars. This allows for comparisons between different simulations of metal production.

An example fit for solar abundances is provided in "Example_File.ipynb". The following is a walkthrough of this example:

Step 1:

Import all needed packages see example below.

```
import numpy as np
from matplotlib import pyplot as plt
import make_metal_fits as mmf
```

Step 2:

Create two arrays. Your measured abundances and your measured errors. Both of these arrays must be the same size. See the example below

```
measured_abundances= np.array([-2.20193105, -2.78113511, -3.21033912, -3.15755259, -3.47007123, -4.0345397,
 -4.22310703, -4.77006481, -4.8961619,  -2.9290349 , -4.14739664])
measured_error= np.array([0.07, 0.1 , 0.06, 0.01 ,0.02, 0.1,  0.02, 0.02, 0.01 ,0.08, 0.04])
```

Step 3:

Create an array of elements you are using for fitting like the example below. This should be the same size as the arrays in Step 2. Number correspond to the atomic number of each element.

```
elements_to_use = np.array([8,10,12,14,16,18,20,24,25,26,28])
```

Step 4:

Create 3 arrays choosing which models you would like to use.

```
CC_models_to_use = np.array(['W18_SN_total_yield_elements','W18_SN_wind_total_yield_elements', 'N20_SN_total_yield_elem
Ia_models_to_use = np.array(['shen_total_yield_elements','iwamoto_total_yield_elements', 'wdd2_total_yield_elements', '
AGB_models_to_use = np.array(['karakas_a2_0200_total_yield_elements','karakas_a3_0080_total_yield_elements','karakas_a4
```

The list of models to choose from is

Step 5:

Call the make model fits function and define your outputs

The items to the left of the equal sign are the items you name yourself this is a general form so you can see what the outputs are. Below is a line of code and a picture of it

Code:

solar_normalization,solar_Ia_ratio,solar_Ia_CC_ratio,solar_normalization_error,solar_Ia_error, solar_Ia_CC_ratio_error= mmf.make_metal_fits(elements_to_use, measured_abundances,measured_error,CC_models_to_use, Ia_models_to_use, AGB_models_to_use)

```
solar_normalization,solar_Ia_ratio,solar_Ia_CC_ratio,solar_normalization_error,solar_Ia_error,solar_Ia_CC_ratio_error
```

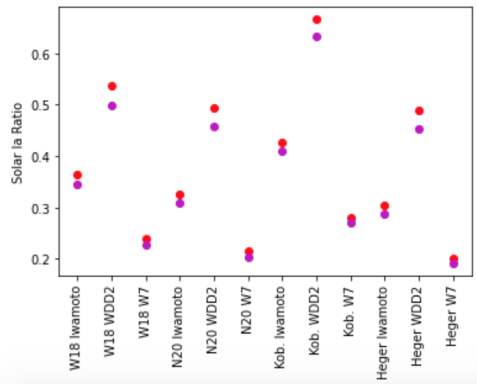
```
= mmf.make_metal_fits(elements_to_use,measured_abundances,measured_error,
                      CC_models_to_use, Ia_models_to_use, AGB_models_to_use)
```

Step 6:

Choose either the normalization or Ia_ratio and plot it like the example below using matplotlib.pyplot. Also set the NPM which is the Number of Plot Models. This should be the number of all your models from step 4. In this case it's 16. In the plot there is Ia_ratio[2:,0,:]. The first of the index is the core collapse models you want to use for the plot. This is from your CC models to use, the array from Step 4. In this case I used all of them starting from the second one "2:" The second index is the AGB model in this case I used 0, so the first model in the AGB_models_to_use array from Step 4. The third index is the Ia models being used from the Ia_models_to_use array in step 4. I used all of them in the example

```
NPM=12 #number plot models


# The first index is the CC model, second is AGB, third is Ia

plt.plot(np.reshape(solar_Ia_CC_ratio[:,0,:],(NPM)),'ro') #this is plotting the models using high solar metallicity agb

plt.plot(np.reshape(solar_Ia_CC_ratio[:,3,:],(NPM)),'mo') # this is plotting without using an agb star

plt.xticks([0, 1, 2,3,4,5,6,7,8,9,10,11], ['W18 Iwamoto', 'W18 WDD2', 'W18 W7','N20 Iwamoto', 'N20 WDD2', 'N20 W7','Kob
       rotation=90)

plt.ylabel('Solar Ia Ratio')
plt.show()
#This is plotting the four core collapse models and the three Ia models for two different agb models.
```

executed in 566ms, finished 16:08:24 2022-04-07



Below is all possible Ia, CC, and AGB models

Ia models: shen_total_yield_elements, iwamoto_total_yield_elements,

seitenzahl_total_yield_elements, wdd2_total_yield_elements, w7_total_yield_elements

CC models: ‘W18_SN_total_yield_elements'(this is just supernova),

'W18_SN_wind_total_yield_elements'(this is pre-supernova wind and supernova),

'N20_SN_total_yield_elements'(this is just supernova), 'N20_SN_wind_total_yield_elements'(this is pre-supernova wind and supernova), 'kobayashi_02_total_yield_elements', 'heger_2010'

AGB models: 'karakas_a2_0200_total_yield_elements', 'karakas_a3_0080_total_yield_elements', 'karakas_a4_0040_total_yield_elements', 'karakas_a5_0001_total_yield_elements',' no_agb'

For Perseus, these are the elements that are available.

8,10,12,14,16,18,20,24,25,26,28

Make Metal Fits

The “make_metal_fits.py” file exists to define a function called “make_metal_fits”, this function works to take user input of metal abundances and return the normalization, Ia supernova ratio, ia to core collapse ratio and all three of their errors.

The function and it’s inputs is

make_metal_fits(elements_to_use, measured_abundance, measured_error, CC_models_to_use, Ia_models_to_use, AGB_models_to_use)

The inputs are:

Elements_to_use: An array of elements the user is choosing to plot using atomic numbers. (See the list of elements in the user guide.)

measured_abundance : This is the abundance of each element that the user chose in the “elements_to_use” array. This must be the same size as “elements_to_use” and the abundances should line up with each element.

Measured_error: This is an array of the errors on each abundance in the “measured_abundance” array.

CC_models_to_use: Array of core-collapse models the user is deciding to use.(see user guide for full list)

Ia_models_to_use: Array of Ia models the user is deciding to use.(see user guide for full list)

AGB_models_to_use: Array of AGB models the user is deciding to use. ‘0200’ is high agb and ‘no agb’ is no agb. The others have AGB models with different metallicities(see user guide for full list)

The outputs are:

normalization: The Fraction of gas that has been converted to stars (1 would mean all available hydrogen and helium was turned into stars). Anything more than 1 means more supernova are needed than could be created with the IMF used.

Ia_ratio: The fraction of stars that produce a type Ia supernova.

Ia_cc_ratio: Ratio of Ia to Core collapse supernova.

normalization_error: error on the normalization

Ia_ratio_error: The error on the Ia_ratio

Ia_CC_ratio_error: The error on the Ia_CC_ratio

N1_All_Model_Data

Located in the main directory, “N1_All_Model_Data.ipynb” is used to load all data from supernovae and AGB star models, and put this data into a unified set of python arrays. If you wanted to add new data from simulations, it should be loaded by modifying this file.

The first box imports all the packages that will be used, and the second and third boxes load in all the data files. For each nucleosynthesis data file in the third box, we account for the formatting of that data file to load the amount of mass produced for each isotope. For each model, an 'isotope' array, with the names of the isotopes, and a 'mass' array, with the amount of mass of each isotope, is created.

The lines of code in box 5 create a map of isotopes for each model to a common set of isotopes. Some models use more or less isotopes compared to others, so we need a superset of all isotopes used in any model. We use the isotope list from the W18 Core Collapse model as our isotope set. Every isotope from the other models is mapped to the isotopes in the W18 model. Currently, if a model had an isotope that the W18 Isotope map does not, then it is discarded. . If a model does not have an isotope that the W18 model does, the mass of that isotope is set to zero for that model.

In box 6, a list of isotopes names for the W18 isotope list is created. The procedure isotopres_to_elements defined in box 7 and in the python file titled “imf_procedures.py”, is used to combine masses of all isotopes of the same element for each model, this is used in the second file to create an “elements” array for each model, after the fitting where the index of the array corresponds to the atomic number (e.g. element_mass[1] = H mass, element_mass[2] = He mass, etc.).

In the last box (box 7), all data is saved into the file all_model_data.db to be loaded in the beginning of notebook file 2 titled, “N2_imf_data_default.ipynb”.

N2_imf_data_default

In the main directory, “imf_procedures.py” is used to create an IMF then weight the amount of each element produced by a given model by the number of stars in the appropriate mass range. Any changes to the IMF, or to the mass ranges over which different types of SNe or AGB stars occur should be made in this file.

Box 1 loads all the necessary packages, and loads the all_model_data.db file created by the first file “Research_N1_All_Model_Data”. A Salpeter IMF is created by calling make_IMF_powerlaw ,from “imf_procedures.py” ,with a slope of -2.35, minimum mass of 0.2 solar masses, and upper mass limit of 100 solar masses.

[? We need to move where Ia_num and CC_num are calculated to box 2, after the Ia and CC limits are set. ?]

In Box 2 , the mass bounds of each source of metals is defined. By default we have AGB stars between 1 and 8 solar masses, Type-Ia SNe possible for stars between 1 and 8 solar masses, and CCSNe for stars between 8 and 140 solar masses. For AGB stars and CCSNe, we assume all stars in the respective masses ranges become an AGB star or a CCSNe. Each set of models for AGB stars and CCSNe has models for several different initial stellar masses. The total amount of each isotope produced is weighted by the fraction of stars produced by the IMF at each mass, using the CCSN or AGB star model with the closest mass. For Type-Ia SN, only a single model is used regardless of initial stellar mass, with the fraction of stars in the set mass range that produce Type-Ia SNe a free parameter.

In box 3, we use the procedure isotope_to_elements to combine all isotopes of the same element together for each integrated model. This allows direct comparison to element abundances from observational data.

In box 4, a function is defined as “abundance_func” ; this is the function we used to measure the abundance of each element in all of the models.

In Boxes 5 and 6, observed abundance measurements for the Sun and Perseus cluster from Lodders (2009) and Simionescue et al. (2015) are loaded. The abundances and uncertainties are formatted the same way as the element and mass arrays from the supernova models.

In the last two boxes (7 and 8) all data is saved in the file imf_data_default.db. The integrated data for each set of models, along with solar and perseus abundances and necessary ancillary data, is stored in the file “imf_data_pick.pkl” to be loaded by “make_metal_fits.py”.

Imf_procedures

The file imf_procedures.py contains two procedures for making an IMF. The first is:

make_IMF_powerlaw(n, m_min, m_max, m_cut, imf_index)

The inputs are:

n : number of mass bins

m_min : minimum mass in solar masses

m_max : maximum mass in solar masses

m_cut : upper mass cutoff above which to have only empty mass bins

imf_index : slope of the powerlaw. For a Salpeter IMF, imf_index would be -2.35

The outputs are:

mass : an array of length n containing the the average mass of a stars in each bin

num_m : an array of length n containing the number stars in each mass bin for 1 solar mass of stars formed

mass_limits : An array of length n+1 containing the masses at the edges of each bin.

For this function, the number of stars per mass is dN(M)/dM ~ M^imf_index. The number of stars per bin is normalized such that sum of mass*num_m will equal to 1.

The second procedure to make an IMF is make_imf_kroupa. For this procedure, the inputs are:

n : number of mass bins

m_min : minimum mass in solar masses

m_max : maximum mass in solar masses

m_cut : upper mass cutoff above which to have only empty mass bins

imf_index : slope of the powerlaw above 0.5 solar masses. For a default Kroupa IMF, imf_index would be -2.3

The outputs are:

mass : an array of length n containing the the average mass of a stars in each bin

num_m : an array of length n containing the number stars in each mass bin for 1 solar mass of stars formed

mass_limits : An array of length n+1 containing the masses at the edges of each bin.

For this function, the number of stars per mass is dN(M)/dM ~ M^(-0.3) below 0.08 solar masses, dN(M)/dM ~ M^(-1.3) between 0.08 and 0.5 solar masses, and dN(M)/dM ~ M^imf_index above 0.5 solar masses. The number of stars per bin is normalized such that sum of mass*num_m will equal to 1.