Mass Sampling Documentation

Functions

Single Redshifts

Main function:

A. `def mass_sampling(mass_range, redshift = 0.0, mdef = '200c', model = 'bocquet16', sample_num = 100000):`

```
 the function to give back a sample of single-redshift cluster mass
distribution based on the halo mass function

 Parameters:
 -----------
 input:
 mass_range: a tuple of cluster masses, lower limit, and upper limit for
sampling
 redshift: a float, 0.0 by default
 sample_num: an integer of the number of samples, 100000 by default
 mdef: The mass definition in which the halo mass M is given
 model: the halo mass function model used by colossus
 output:
 mass_chain: a NumPy array of length = sample_num.
 test_func: the likelihood function

 Library:
 -----------
   ●  Colossus

 step:
 -----------
   ●  Initiate a NumPy array as cluster masses
   ●  Use the halo mass function given by colossus, give back an array of
      number density
   ●  Extract power from the mass array using helper 3, from 10^14 M☉ to
      14
   ●  Use helper 2 to calculate the final mass sampling
```

Helpers:

1. lnpo(mass, min, max, test_fun):

likelihood function used by MCMC

parameters:

----

mass: a float or a 1d NumPy array of cluster mass in 10^n M☉ unit

2. interpolate_MCMC(mass_array_p, mfunc_n, mass_range, sample_num, redshift):

interpolate and normalize mfunc_n, use the result as a likelihood function and perform MCMC method to get the sample.

parameters:

----

input:

mass_arr_p: 1d NumPy array of cluster mass power (for example, 10^14 M☉ represented as 14 in arr)

mfunc_n: 1d NumPy array of halo number density * 10^5

mass_range: a tuple of cluster masses, lower limit, and upper limit for sampling

sample_num: an integer of the number of samples

output:

sample_chain.flatten(): an 1d NumPy array of mass sampling, same unit as mass_arr_p

Library:

- **scipy(interpolate & integrate)**
- **Emcee**

step:

- **Interpolate mass_arr_p & mfunc_n, return a function object**
- **Calculate the integration of function between the mass range, normalize the function using integration, result in test_func**
- **Use emcee package to do MCMC simulation, Helpers 1 (lnpo) used as the likelihood function, return test_func and mass_chain.flatten()**

3. extract_power(mass_arr):

```
 Function to extract the power of galaxy cluster mass array, switch from
10^n to n

 parameters:
 -----------
 Input:
 mass_arr: 1d NumPy array of cluster mass in 10^n M☉ unit

 output:
 mass_arr_p: 1d NumPy array of mass of power = n

 step:
 -----------
 mass_arr_p = np.log10(mass_arr)
```

Multiple Redshifts

Main function:

```
      B. def mul_redshift_mass_sampling(rs_dist = "skewnorm", rs_range =
(0.0, 1.5), mass_range = (14.0, 16.0), mdef = '200c', model = 'bocquet16',
sample_num = 100000, store = True):

   the function to give back a sample of multi-redshift cluster mass
distribution based on halo mass function

   Parameters:
   -----------
   input:
   rs_dist: a string, representing the distribution of cluster redshift,
"skewnorm" by default
   rs_range: a tuple of redshift range, (0.0, 1.5) by default
   mass_range: a tuple of cluster masses, lower limit and upper limit for
sampling, [min, max] in 10^min M☉ unit
   mdef: The mass definition in which the halo mass M is given; see
colossus doc for more info
(https://bdiemer.bitbucket.io/colossus/lss_mass_function.html#lss.mass_fun
ction.massFunction)
   model: the halo mass function model used by colossus; see colossus doc
for more info
   sample_num: an integer of the number of samples, 100000 by default
```

store: a boolean, if True store mass array and redshift into a csv file and return a string of path to file if False returns None

output:
fin_cluster: a Pandas dataframe with 2 columns of ["mass_arr", "redshift"], NumPy array of cluster mass corresponding to redshift stored in each row
filepath: str of file path if store=True, else None

Library:

- **pandas**

Step:

- **Call helper 2 to obtain the redshift interval (chop) and corresponding cluster number.**
- **Loop through every redshift in chop, call function A(mass_sampling) to do single redshift mass sampling, store mass and redshifts in two pandas Series**

Helpers:

1. skew_sample(size = 10000):
"""
the function to give back a sample of redshift based on skew gaussian distribution imitating SPT cluster data: https://pole.uchicago.edu/public/data/sptsz-clusters/

Parameters:
----
input:
size: integer, sample number

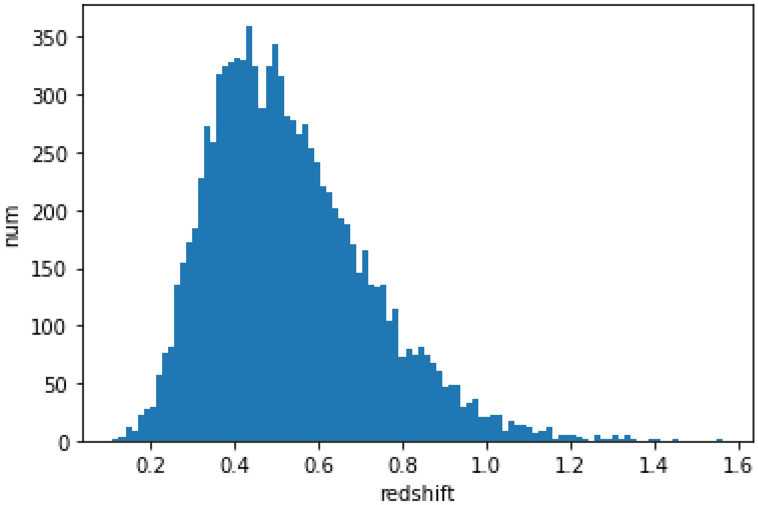
output:
mass_chain: a NumPy array of length = sample_num
rs_sample: a 1d NumPy array of clusters' redshift sample with length = size

Library:

- **scipy.stats**

Step:

- **Use scipy.stats to imitate redshift distribution**
- **Use skewnorm.rvs to draw a sample from it**



2. single_redshift_num(rs_range, sample_num, rs_dist_model):

```
   """
   the function to give back redshifts and sample_num per redshift for
multi-redshift sampling

   Parameters:
   -----------
   input:
   rs_dist_model: a string, representing the distribution of cluster
redshift
   rs_range: a tuple of redshift range, (0.0, 1.5) by default
   sample_num: an integer of the number of samples, 100000 by default

   output:
   chop: a NumPy array of redshifts
   num_per_redshift: a NumPy array of cluster num within the corresponding
redshift interval of the same index number in chop
   """
```

Step:

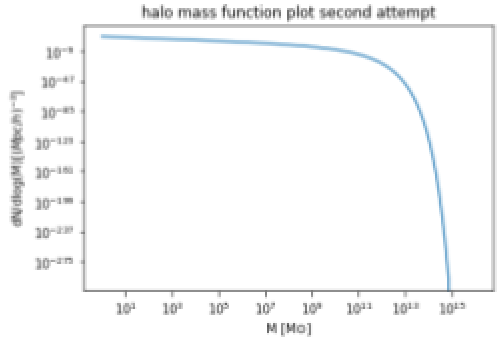
- **Call helper 1 to obtain redshift distribution array**
- **Based on redshift range, divide it to several chops of redshift, each chop used as limit**
- **Loop through the redshifts array given by helper 1, approximate redshifts between two chops to the lower limit of it.**
- **Use another NumPy array of the same size to keep track of cluster number in each interval**

Procedure

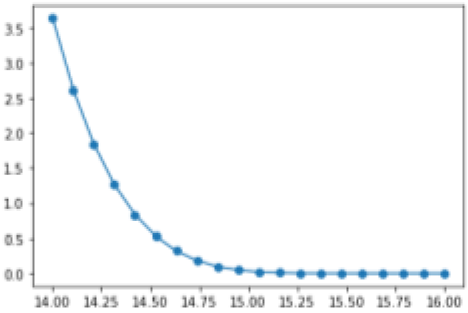
Single redshift

Function A
[mass_sampling]
Single redshift mass sampling

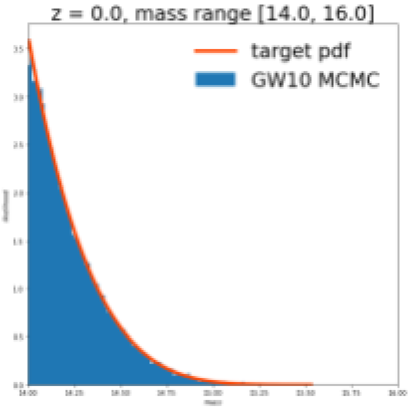
Input:
mass_range, redshift



Colossus output:
Mass array, number density array



Interpolate & integrate output:
Mass array, normalized likelihood function

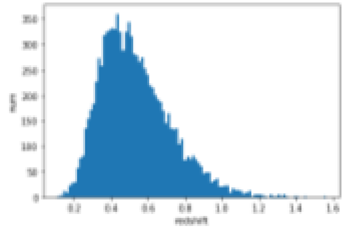


MCMC Output:
function object and mass sample

Multi-redshift:

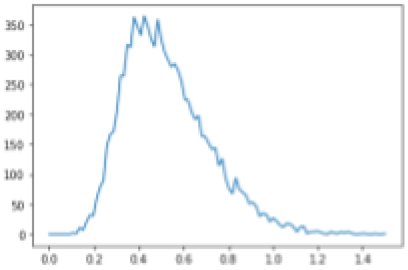
Function B
[mul_redshift_mass_sampling]
Multiple redshift

Input: mass_range, redshift range, sample size

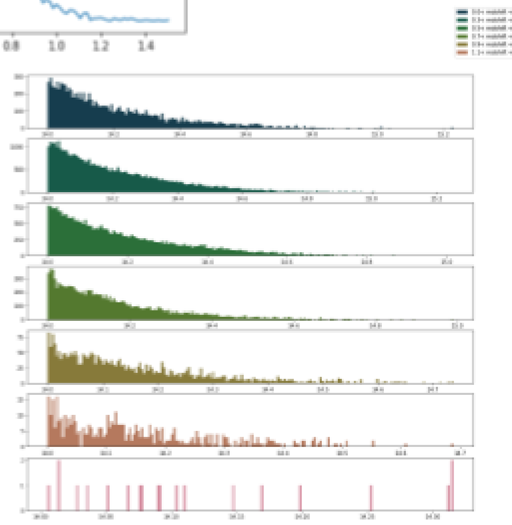


[Skew_sample] output: redshift array based on skewed gaussian distribution imitating redshift distribution

[single_redshift_num] output: a chop array of redshift interval, a num_per_redshift array of cluster number/redshift interval



Called [mass_sampling] to sample for each redshift in chop



Output: a pandas dataframe with mass and redshift as two columns