EMDA

Release 1.1.3

Rangana Warshamanage

Aug 02, 2023

TABLE OF CONTENTS:

EMDA is an importable Python library for Electron Microscopy map and model manipulations written in Python3. EMDA comes under MPL-2.0 license.

A selected set of EMDA's core functionalities include:

- Map input/output and format conversion
- Resolution related functionalities
- Map statistics calculation in Image and Fourier space
- Likelihood based map fitting and Bayesian map averaging
- Likelihood based magnification refinement
- Local correlation map calculation for map and model validation
- FSC based map-model validation

CHAPTER

ONE

INSTALLATION

To install EMDA, just type following line in the terminal

```
$ pip install emda
```

All necessary files will be installed under pythonx.x/site-packages/

e.g.: /Users/ranganaw/anaconda3/lib/python3.6/site-packages

To update EMDA, just type

```
$ pip install -U emda
```

To test the success of installation, please run the following command in the terminal

```
$ emda_test
```

All tests will pass, if the installation is successful.

CHAPTER

TWO

DEPENDENCIES

EMDA has several dependencies in addition to general python modules (e.g. Numpy, Scipy). They are Pandas, Gemmi, Mrcfile and Matplotlib. All these dependencies will be automatically checked and installed during EMDA installation.

CHAPTER

THREE

HOW TO USE EMDA

There are two ways to access EMDA's underlying functionalities. EMDA is available as a standalone command line tool to ensure its easy usage. Each functionality is callable with a keyword followed by a set of arguments. Also, EMDA has a simple to use, yet powerful, API for advanced usage documented under emda_methods.

3.1 emda methods module

Author: "Rangana Warshamanage, Garib N. Murshudov" MRC Laboratory of Molecular Biology

This software is released under the Mozilla Public License, version 2.0; see LICENSE.

emda_methods.**apply_bfactor_to_map**(*mapname*, *bf_arr*, *mapout*)

Applies an array of B-factors on the map.

Arguments:

Inputs:

mapname: string
Map file name.

bf_arr: float, 1D array
An array/list of B-factors.

mapout: bool
If True, map for each B-factor will be output.

Outputs:

all_mapout: complex, ndarray
4D array containing Fourier coefficients of all maps. e.g. all_mapout[:,:,:,i], where i represents map number corresponding to the B-factor in bf_arr.

emda_methods.**applymask**(*mapname*, *maskname*, *outname*)

Apply a masl on map

emda_methods.**average_maps**(*maplist*, *rot=0.0*, *ncy=5*, *res=6*, *interp='linear'*, *fit=True*, *tra=None*, *axr=None*, *fobj=None*, *masklist=None*)

Calculates the best average maps using Bayesian principles.

Calculates the best average map using Bayesian principles. This is done in two steps; 1. Parameter estimation using a likelihood function, 2. Best map calculation. Parameter estimation is similar to map overlay where each map is brought onto static map by maximizing the overlap. The best maps are calculated using superimposed maps.

Arguments:

Inputs:

maplist: list
List of half maps to average.

masklist: list, optional
List of masks to apply on maps. len(masklist) == len(maplist) // 2

rot: float, optional
Initial rotation in degrees. Default is 0.0.

axr: list, optional
Rotation axis. Default is [1, 0, 0].

tra: list, optional
Translation vector in fractional units. Default is [0.0, 0.0, 0.0]

res: float, optional
Fit start resolution in Angstrom units. Default is 6.0 Angstrom.

ncy: integer, optional
Number of fitting cycles. Default is 5.

interp: string, optional
Interpolation type either "linear" or "cubic". Default is linear.

fobj: string
File object for logging. If None given, EMDA_average.txt will be output.

fit: bool, optional
If True, map fitting will be carried out before average map calculation. Default is True.

Outputs:
Outputs a series of overlaid maps (**fitted_map_**?.mrc). Also, outputs a series of average maps (**avgmap_**?.mrc)

emda_methods.**b_from_correlation**(*half1map*, *half2map*, *resol*, *kernel_size=5*, *mask_map=None*)

B from local correlation

emda_methods.**balbes_data**(*map1name*, *map2name*, *fsccutoff=0.5*, *mode='half'*)

Returns data required for Balbes pipeline.

Required data is output with their references in EMDA.xml.

Arguments:

Inputs:

map1name: string
Name of the map 1.

map2name: string
Name of the map 2.

fsccutoff: float, optional
FSC of desired resolution. Defualt is 0.5

mode: string
Mode can be either 'half' or 'any'. If the input maps are the half maps, mode should be 'half'. Otherwise, mode should be 'any'. Default mode is half.

Outputs:
Outputs EMDA.xml containing data and references to other data. No return variables.

emda_methods.**bestmap**(*hf1name*, *hf2name*, *outfile*, *mode=1*, *knl=5*, *mask=None*, *B=None*)

Calculates EMDA bestmap

emda_methods.**center_of_mass_density**(*arr*)

Returns the center of mass of 3D density array.

This function accepts density as 3D numpy array and caclulates the center-of-mass.

Arguments:

Inputs:
arr: density as 3D numpy array

Outputs:
com: tuple, center-of-mass (x, y, z)

emda_methods.**compositemap**(*maps*, *masks*)

Computes composite map

emda_methods.**difference_map**(*maplist*, *diffmapres=3.0*, *ncy=5*, *mode='norm'*, *fit=False*, *usehalfmaps=False*, *usecom=False*, *fitres=None*, *masklist=None*)

Calculates difference map.

Arguments:

Inputs:

maplist: string
List of map names to calculate difference maps. If combined with fit parameter, firstmap in the list will be taken as static/reference map. If this list contains coordinate file (PDB/CIF), give it in the second place. Always give MRC/MAP file at the beginning of the list. e.g:

[test1.mrc, test2.mrc] or [test1.mrc, model1.pdb/cif]

If combined with usehalfmaps argument, then halfmaps of the firstmap should be given first and those for second next. e.g:

[map1-halfmap1.mrc, map1-halfmap2.mrc,
map2-halfmap1.mrc, map2-halfmap2.mrc]

masklist: string, optional
List of masks to apply on maps. All masks should be in MRC/MAP format. e.g:

[mask1.mrc, mask2.mrc]

diffmapres: float
Resolution to which difference map be calculated. If an atomic model involved, this resolution will be used for map calculation from coordinates

mode: string, optional
Different modes to scale maps. Two difference modes are supported. 'ampli' - scale between maps is based on amplitudes . 'norm' [Default] - normalized maps are used to calculate difference map. If fit is enabled, only norm mode used.

usehalfmaps: boolean
If employed, halfmaps are used for fitting and difference map calculation. Default is False.

fit: boolean
If employed, maps and superimposed before calculating difference map. Default is False.

Outputs:
Outputs diffence maps and initial maps after scaling in MRC format. Differece maps are labelled as

emda_diffmap_m1.mrc, emda_diffmap_m2.mrc

Scaled maps are labelled as
emda_map1.mrc, emda_map2.mrc

emda_methods.**estimate_map_resol**(*hfmap1name*, *hfmap2name*)

Estimates map resolution.

Arguments:

Inputs:

hfmap1name: string
Halfmap 1 name.

hfmap2name: string
Halfmap 2 name.

Outputs:

map_resol: float
Map resolution determined by the halfmap FSC.

emda_methods.**fetch_data**(*emdbidlist*, *alldata=False*)

Download data and model

emda_methods.**fouriersp_correlation**(*half1_map*, *half2_map*, *kernel_size=5*, *mask=None*)

Calculates Fourier space local correlation using half maps.

Arguments:

Inputs:

half1_map: string
Name of half map 1.

half2_map: string
Name of half map 2.

kernel_size: integer, optional
Radius of integration kernal. Default is 5.

Outputs:
Following maps are written out: fouriercorr3d_halfmaps.mrc - local correlation in half maps. fouriercorr3d_fullmap.mrc - local correlation in full map

using the formula 2 x FSC(half) / (1 + FSC(half)).

fouriercorr3d_truemap.mrc - local correlation in true map.
Useful for validation purpose.

emda_methods.**get_biso_from_map**(*halfmap1*, *halfmap2*)

Calculates the isotropic B-value of the map.

Arguments:

Inputs:

halfmap1: string
Halfmap 1 file name.

halfmap2: string
Halfmap 2 file name.

Outputs:

biso: float
Map B-iso value.

emda_methods.**get_biso_from_model**(*mmcif_file*)

Calculates the isotropic B-value of the model.

Arguments:

Inputs:

mmcif_file: string
mmCIF file name.

Outputs:

biso: float
Model B-iso value.

emda_methods.**get_data**(*struct*, *resol=5.0*, *uc=None*, *dim=None*, *maporigin=None*)

Returns data of a map or a model into an ndarray.

Reads map data into an ndarray, or if the structure input is an atomic model, it calculates the map from the model and returns as an ndarray.

Arguments:

Inputs:

struct: string
CCP4/MRC map file name or PDB/ENT/CIF file resol: float, optional

resolution to calculates map from model. Default is 5.0 A.

uc: float, 1D array
Parameter for modelmap generation. If absent, this will be determined by dim parameter.

dim: sequence (integers), optional
Parameter for modelmap generation. If absent, this will be determined from the size of the molecule.

maporigin: sequence (integers), optional
Parameter for modelmap generation. If present, the calculated map will be shifted according to this information. If absent, this parameter is taken as [0, 0, 0].

Outputs:

uc: float, 1D array
Unit cell

arr: float, 3D array
Map values as Numpy array

origin: list
Map origin list

emda_methods.**get_dim**(*model*, *shiftmodel='new1.cif'*)

Returns the box dimension to put the modelmap in.

Determines the dimension of the box for the model based map.

Arguments:

Inputs:
model: atomic model as .pdb/.cif shiftmodel: name for COM shifted model, optional.

Default name - new1.cif.

Outputs:
dim: integer, dimension of the box.

emda_methods.**get_fsc**(*arr1*, *arr2*, *uc*)

Returns FSC as a function of resolution

Arguments:

Inputs:

arr1: float, ndarray
Density array 1.

arr2: float, ndarray
Density array 2.

uc: float, 1D array
Unit cell

Outputs:

res_arr: float, 1D array
Linear array of resolution in Angstrom units.

bin_fsc: float, 1D array
Linear array of FSC in each resolution bin.

emda_methods.**get_map_pointgroup**(*maplist*, *reslist*, *use_peakheight=True*, *peak_cutoff=0.8*, *use_fsc=False*, *fsc_cutoff=0.7*, *ang_tol=5.0*, *emdlist=None*, *fobj=None*)

Returns the point group of map.

This function determines the point group of an EM map using ProSHADE and EMDA.

Arguments:

Inputs:
maplist: list of strings List of map names in mrc/map format.

reslist: float List of map resolutions

use_peakheight: bool ProSHADE peaklist is used to decide the point group. Default option.

peak_cutoff: float Cutoff for peak heights in the peaklist. Default is 0.8. However, if the highest peak is lower than this threshold then the cutoff is chosen such that highest - 0.1.

use_fsc: bool If true, FSC is used in place of Proshade peak list to decide the point group.

fsc_cutoff: float Cutoff for FSC. Default is 0.7.

ang_tol: float Tolerence for angle between two axes in the proshade axes list. Default is 5.0 degrees.

emdlist: list of strings EMDB-id list to keep the correspondence in results.

Outputs:
pglist: list of strings Point group list decided by EMDA

ppglist: list of strings Point group kist decided by ProSHADE

emda_methods.**get_map_power**(*mapname*)

Calculates the map power spectrum.

Arguments:

Inputs:

mapname: string
Map name.

Outputs:

res_arr: float
Resolution array.

power_spectrum: float
Map power spectrum.

emda_methods.**get_variance**(*half1name*, *half2name*, *filename=None*, *maskname=None*)

Returns noise and signal variances of half maps.

Returns noise and signal variances of half maps. Return values are not corrected for full map.

Arguments:

Inputs:

half1name: string
Name of the half map 1.

half2name: string
Name of the half map 2.

filename: string
If present, statistics will be printed into this file.

maskname: String
If present, input maps will be masked before computing variances.

Outputs:

res_arr: float, 1D array
Linear array of resolution in Angstrom units.

noisevar: float, 1D array
Linear array of noise variance in each resolution bin.

signalvar: float, 1D array
Linear array of signal variance in each resolution bin.

emda_methods.**half2full**(*half1name*, *half2name*, *outfile='fullmap.mrc'*)

Combines half maps to generate full map.

Arguments:

Inputs:

half1name: string
Name of half map 1.

half2name: string
name of half map 2.

outfile: string, optional
Name of the output file. Default is fullmap.mrc

Outputs:

fullmap: float, 3D array
3D Numpy array of floats.

emda_methods.**halfmap_fsc**(*half1name*, *half2name*, *filename=None*, *maskname=None*)

Computes Fourier Shell Correlation (FSC) using half maps.

Computes Fourier Shell Correlation (FSC) using half maps. FSC is not corrected for mask effect in this implementation.

Arguments:

Inputs:

half1name: string
Name of the half map 1.

half2name: string
Name of the half map 2.

filename: string
If present, statistics will be printed into this file.

maskname: String
If present, input maps will be masked before computing FSC.

Outputs:

res_arr: float, 1D array
Linear array of resolution in Angstrom units.

bin_fsc: float, 1D array
Linear array of FSC in each resolution bin.

emda_methods.**halfmap_fsc_ph**(*half1name*, *half2name*, *filename='halffsc.txt'*, *maskname=None*)

Computes Fourier Shell Correlation (FSC) using half maps.

Computes Fourier Shell Correlation (FSC) using half maps. FSC is not corrected for mask effect in this implementation.

Arguments:

Inputs:

half1name: string
Name of the half map 1.

half2name: string
Name of the half map 2.

filename: string
If present, statistics will be printed into this file.

maskname: String
If present, input maps will be masked before computing FSC.

Outputs:

res_arr: float, 1D array
Linear array of resolution in Angstrom units.

bin_fsc: float, 1D array
Linear array of FSC in each resolution bin.

emda_methods.**lowpass_map**(*uc*, *arr1*, *resol*, *filter='ideal'*, *order=4*)

Lowpass filters a map to a specified resolution.

This function applies a lowpass filter on a map to a specified resolution. This operations is carried out in the Fourier space. Note that lowpass map will have the same shape as input data.

Arguments:

Inputs:

uc: float, 1D array
Unit cell params.

arr1: float, 3D array
3D Numpy array containing map values.

resol: float
Resolution cutoff for lowpass filtering in Angstrom units.

filter: string, optional
Fiter type to use in truncating Fourier coefficients. Currently, only 'ideal' or 'butterworth' filters can be employed. Default type is ideal.

order: integer, optional
Order of the Butterworth filter. Default is 4.

Outputs:

fmap1: complex, 3D array
Lowpass filtered Fourier coefficeints.

map1: float, 3D array
Lowpass filtered map in image/real space

emda_methods.**map2mtz**(*mapname*, *mtzname='map2mtz.mtz'*, *resol=None*)

Converts a map into MTZ format.

Arguments:

Inputs:

mapname: string
Map file name.

mtzname: string
Output MTZ file name. Default is map2mtz.mtz

Outputs:
Outputs MTZ file.

emda_methods.**map2mtzfull**(*uc*, *arr1*, *arr2*, *mtzname='halfnfull.mtz'*)

Writes several 3D Numpy arrays into an MTZ file.

This function accepts densities of two half maps as 3D numpy arrays and outputs an MTZ file containing amplitudes of half1, half2 and full map. The outfile data labels are H, K, L, Fout1, Fout2, Foutf, Poutf. The last four labels correspond to amplitude of halfmap1, amplitudes of halfmap2, amplitudes of fullmap and the phase values of fullmap, respectively.

Arguments:

Inputs:

uc: float, 1D array
Unit cell params.

arr1: float, 3D array
Half1 map values.

arr2: float, 3D array
Half2 map values.

mtzname: string, optional
Output MTZ file name. Default is halfnfull.mtz

Outputs:
Outputs an MTZ file containing amplitudes of half maps and full map.

emda_methods.**map_model_validate**(*half1map*, *half2map*, *modelfpdb*, *bfac=0.0*, *lig=True*, *model1pdb=None*, *mask=None*, *modelresol=None*, *lgf=None*)

Calculates various FSCs for maps and model validation.

Arguments:

Inputs:

half1map: string
Name of half map 1.

half2map: string
Name of half map 2.

modelfpdb: string
Name of the model refined against full map in cif/pdb/ent formats.

model1pdb: string, optional
Name of the model refined against one half map in cif/pdb/ent formats. If included, FSC between that and half maps will be calculated.

mask: string, optional
Name of the mask file. It will apply on half maps before computing FSC. If not included, a correlation based masked will employed.

modelresol: float, optional
An argument for model based map calculation using REFMAC. Resolution to calculate model based map. If not specified, an FSC based cutoff will be used.

bfac: float, optional
An overall B-factor for model map. Default is 0.0

lig: bool, optional
An argument for model based map calculation using REFMAC. Set True, if there is a ligand in the model, but no description. Default is True.

lgf: string, optional
An argument for model based map calculation using REFMAC. Ligand description file (cif).

Outputs:

fsc_list: list
List of FSCs is returned. If len(fsc_list) is 4, FSC lables are as follows: 0 - half maps FSC 1 - half1map - model1 FSC 2 - half2map - model1 FSC 3 - fullmap-fullmodel FSC If len(fsc_list) is 2, only 0 and 3 contains.

Outputs FSCs in allmap_fsc_modelvsmap.eps

emda_methods.**map_transform**(*mapname*, *tra*, *rot*, *axr*, *outname='transformed.mrc'*)

Imposes a transformation on a map.

Imposes a transformation (i.e. translation and rotation) on a map and returns the transformed map.

Arguments:

Inputs:

mapname: string
Name of the input map.

tra: list of three floats values
Translation vector as a list in Angstrom units.

rot: float
Rotation to apply in degrees.

axr: list of three integers
Axis to rotation. e.g [1, 0, 0]

outname: string, optional
Name of the transformed map. Default is transformed.mrc.

Outputs:

transformed_map: float, 3D array
3D Numpy array of floats.

emda_methods.**mapmagnification**(*maplist*, *rmap*)

Refine magnification

emda_methods.**mapmodel_fsc**(*map1*, *model*, *fobj*, *bfac=0.0*, *modelresol=5.0*, *lig=True*, *phaserand=False*, *mask=None*, *lgf=None*)

Map-model FSC

emda_methods.**mask_from_atomic_model**(*mapname*, *modelname*, *atmrad=5*)

Generates a mask from atomic coordinates.

Generates a mask from coordinates. First, atomic positions are mapped onto a 3D grid. Second, each atomic position is convluted with a sphere whose radius is defined by the atmrad paramter. Next, one pixel layer dialtion followed by the smoothening of edges.

Arguments:

Inputs:

mapname: string
Name of the map file. This is needed to get the sampling, unit cell and origin for the new mask. Allowed formats are - MRC/MAP

modelname: string
Atomic model name. Allowed formats are - PDB/CIF

atmrad: float
Radius of the sphere to be placed on atomic positions in Angstroms. Default is 5 A.

Outputs:

mask: float, 3D array
3D Numpy array of the mask.

Outputs emda_model_mask.mrc.

emda_methods.**mask_from_halfmaps**(*uc*, *half1*, *half2*, *radius=4*, *iter=1*, *dthresh=None*)

Generates a mask from half maps.

Generates a mask from half maps based on real space local correlation.

Arguments:

Inputs:

uc: float, 1D array
Unit cell parameters.

half1: float, 3D array
Half map 1 data.

half2: float, 3D array
Half map 2 data.

radius: integer, optional
Radius of integrating kernel in voxels. Default is 4.

iter: integer,optional
Number of dilation cycles. Default is 1 cycle.

dthresh: float, optional
The halfmap densities will be thresholded at this value prior calculating local correlation. If the value is not given, EMDA takes this values as the value at which the cumulative density probability is 0.99. This is the default. However, it is recommomded that this value should be supplied by the user and proven to be useful for cases those have micellular densities around protein.

Outputs:

mask: float, 3D array
3D Numpy array of correlation mask.

emda_methods.**mask_from_map**(*uc*, *arr*, *kern=5*, *resol=15*, *filter='butterworth'*, *order=1*, *prob=0.99*, *itr=3*, *orig=None*)

Generates a mask from a map.

Generates a mask from a map.

Arguments:

Inputs:

uc: float, 1D array
Unit cell parameters.

arr: float, 3D array
Map data.

half2: float, 3D array
Half map 2 data.

kern: integer, optional
Radius of integrating kernel in voxels. Default is 5.

resol: float, optional
Resolution cutoff for lowpass filtering in Angstrom units. Default is 15 Angstrom.

filter: string,optional
Filter type to use with lowpass filtering. Default is butterworth.

order: integer, optional
Butterworth filter order. Default is 1.

prob: float, optional
Cumulative probability cutoff to decide the density threshold. Default value is 0.99.

itr: integer, optional
Number of dilation cycles. Default is 3 cycles.

orig: list of three integer values.
Map origin. e.g. [0, 0, 0]

Outputs:

mask: float, 3D array
3D Numpy array of the mask.

Outputs lowpass.mrc and mapmask.mrc files.

emda_methods.**mirror_map**(*mapname*)

Invert map

This method invert the map at its center-of-mass

Args:
mapname (string): Name of the map

emda_methods.**model2map**(*modelxyz*, *dim*, *resol*, *cell*, *bfac=None*, *maporigin=None*, *ligfile=None*, *outputpath=None*)

Calculates EM map from atomic coordinates using REFMAC5

Args:
modelxyz (string): Name of the coordinate file (.cif/.pdb) dim (list): Map dimensions [nx, ny, nz] as a list of integers resol (float): Requested resolution for density calculation in Angstroms. cell (list): Cell parameters a, b and c as floats maporigin (list, optional): Location of the first column (nxstart),

row (nystart), section (nzstart) of the unit cell. Defaults to [0, 0, 0].

ligfile (string, optional): Name of the ligand description file. Defaults to None. outputpath (string, optional): Path for auxilliary files. Defaults to current

working directory

bfac(float, optional): Parameter for refmac. Set all atomic B values to bfac
when it is positive. Default to None.

Returns:
float ndarray: calculated model-based density array

emda_methods.**model2map_gm**(*modelxyz*, *resol*, *dim*, *cell=None*, *maporigin=None*)

Calculates model from coordinates using GEMMI in Servalcat

Args:
modelxyz (string): Name of the coordinate file (.cif/.pdb) dim (list): Map dimensions [nx, ny, nz] as a list of integers resol (float): Requested resolution for density calculation in Angstroms. cell (list): Cell parameters a, b and c as floats maporigin (list, optional): Location of the first column (nxstart),

row (nystart), section (nzstart) of the unit cell. Defaults to [0, 0, 0].

Returns:
float ndarray: calculated model-based density array

emda_methods.**mtz2map**(*mtzname*, *map_size*)

Converts an MTZ file into MRC format.

This function converts data in an MTZ file into a 3D Numpy array. It combines amplitudes and phases with "Fout0" and "Pout0" labels to form Fourier coefficients. If the MTZ contains several aplitude columns, only the one corresponding to "Fout0" will be used.

Arguments:

Inputs:

mtzname: string
MTZ file name.

map_size: list
Shape of output 3D Numpy array as a list of three integers.

Outputs:
outarr: float 3D Numpy array of map values.

emda_methods.**overall_cc**(*map1name*, *map2name*, *space='real'*, *resol=5*, *maskname=None*)

Calculates overall correlation coefficient (CC) between two density maps

This method calculates overall CC in real or Fourier space. First two inputs can be both maps or one of them is map and the other is an atomic model (pdb/ent/cif). If one of them is an atomi model, the resolution (Angstroms) should be given. Default resolution is 5 A.

Args:
map1name (string): Name of map1 map2name (string): Name of map1 space (str, optional): CC calculation in real/Fourier space. Defaults to "real". resol (int, optional): [description]. Defaults to 5. maskname ([type], optional): [description]. Defaults to None.

Returns:
occ: Overall CC corrected to fullmap hocc: Overall CC between half maps

emda_methods.**overlay_maps**(*maplist*, *rot=0.0*, *ncy=5*, *res=6*, *interp='linear'*, *hfm=False*, *modelres=5.0*, *masklist=None*, *tra=None*, *axr=None*, *fobj=None*, *usemodel=False*, *fitres=None*, *usecom=False*)

Superimposes several maps.

Superimposes several maps using a likelihood function. All maps are overlaid on the first map.

Arguments:

Inputs:

maplist: list
List of maps to overlay.

masklist: list
List of masks to apply on maps.

rot: float, optional
Initial rotation in degrees. Default is 0.0.

axr: list, optional
Rotation axis. Default is [1, 0, 0].

tra: list, optional
Translation vector in fractional units. Default is [0.0, 0.0, 0.0]

res: float, optional
Fit start resolution in Angstrom units. Default is 6.0 Angstrom.

ncy: integer, optional
Number of fitting cycles. Default is 5.

interp: string, optional
Interpolation type either "linear" or "cubic". Default is linear.

hfm: bool, optional
If True, overlay will be carried out on half maps. In this case, maplist will contain half maps. e.g. [map1_half1.mrc, map1_half2.mrc, map2_half1.mrc, map2_half2.mrc, ...]. masklist will contain masks for each map. e.g. [map1_mask.mrc, map2_mask.mrc]. The length of masklist should be equal to half the length of maplist. If False, uses full maps for overlay. Default is False.

fobj: string
File object for logging. If None given, EMDA_overlay.txt will be output.

Outputs:
Outputs a series of overlaid maps (**fitted_map_**?.mrc).

emda_methods.**predict_fsc**(*hf1name*, *hf2name*, *nparticles=None*, *bfac=None*, *mask=None*)

Predicts FSC based on reference FSC curve and number of particles

emda_methods.**prepare_refmac_data**(*hf1name*, *hf2name*, *outfile='fullmap.mtz'*, *bfac=None*, *maskname=None*, *xmlobj=None*, *fsccutoff=None*)

Prepare variance data for refmac refinement

emda_methods.**read_atomsf**(*atom*, *fpath=None*)

Reads 'atomsf.lib' file

emda_methods.**read_map**(*mapname*)

Reads CCP4 type map (.map) or MRC type map.

Arguments:

Inputs:

mapname: string
CCP4/MRC map file name

Outputs:

uc: float, 1D array
Unit cell

arr: float, 3D array
Map values as Numpy array

origin: list
Map origin list

emda_methods.**read_mtz**(*mtzfile*)

Reads mtzfile and returns unit_cell and data in Pandas DataFrame.

Arguments:

Inputs:

mtzfile: string
MTZ file name

Outputs:

uc: float, 1D array
Unit cell

df: Pandas data frame
Map values in Pandas Dataframe

emda_methods.**realsp_correlation**(*half1map*, *half2map*, *kernel_size=5*, *norm=False*, *lig=True*, *model=None*, *model_resol=None*, *mask_map=None*, *lgf=None*)

Calculates local correlation in real/image space.

Arguments:

Inputs:

half1map: string
Name of half map 1.

half1map: string
Name of half map 2.

kernel_size: integer, optional
Radius of integration kernal in pixels. Default is 5.

norm: bool, optional
If True, correlation will be carried out on normalized maps. Default is False.

model: string, optional
An argument for model based map calculation using REFMAC. Name of model file (cif/pdb). If present, map-model local correlation will be calculated.

model_resol: float, optional
An argument for model based map calculation using REFMAC. Resolution to calculate model based map. If absent, FSC based resolution cutoff will be employed.

mask_map: string, optional
Mask file to apply on correlation maps. If not given, correlation based mask will be employed.

lig: bool, optional
An argument for model based map calculation using REFMAC. Set True, if there is a ligand in the model, but no description. Default is True.

lgf: string, optional
An argument for model based map calculation using REFMAC. Ligand description file (cif).

Outputs:
Following maps are written out: rcc_halfmap_smax?.mrc - reals space half map local correlation. rcc_fullmap_smax?.mrc - correlation map corrected to full map

using the formula 2 x FSC(half) / (1 + FSC(half)).

If a model included, then rcc_mapmodel_smax?.mrc - local correlation map between model and

full map.

rcc_truemapmodel_smax?.mrc - truemap-model correaltion map for
validation purpose.

emda_methods.**realsp_correlation_mapmodel**(*fullmap*, *model*, *resol*, *kernel_size=5*, *norm=False*, *mask_map=None*, *lgf=None*)

Calculates real space local correlation between map and model.

Arguments:

Inputs:

fullmap: string
Name of the map.

model: string
An argument for model based map calculation using REFMAC. Name of model file (cif/pdb/ent/mtz/mrc).

resol: float
An argument for model based map calculation using REFMAC. Resolution to calculate model based map. All maps are lowpass filtered to this resolution before calculating local correlation.

kernel_size: integer, optional
Radius of integration kernal in pixels. Default is 5.

mask_map: string, optional
Mask file to apply on correlation maps.

norm: bool, optional
Defalt to False. If True, correlation will be carried out on normalized maps. Default is False.

lgf: string, optional
An argument for model based map calculation using REFMAC. Ligand description file (cif).

Outputs:
Following maps are written out: modelmap.mrc - model based map (only if atomic model is given). rcc_mapmodel.mrc - real space local correlation map.

emda_methods.**rebox_mapmodel**(*maplist*, *masklist*, *modellist=None*)

Reboxes a map by a mask.

This function reboxes a map by a mask into a smaller cube. Size of the box is determine by the size of te mask + 10 voxels in each dimetion. It can accept list arguments if there's more than one map and mask. Also, a model can be given and then that will also be reboxed.

Args:
maplist (string): List of map names to be reboxed masklist (string): List of mask names to be used in reboxing modellist (string, optional): List of model names (pdb/cif) to be reboxed.

Defaults to None.

Output:
It outputs reboxed maps, models (if supplied).

emda_methods.**resample_data**(*curnt_pix*, *targt_pix*, *targt_dim*, *arr*)

Resamples a 3D array.

Arguments:

Inputs:
curnt_pix: float list, Current pixel sizes along c, b, a. targt_pix: float list, Target pixel sizes along c, b a. targt_dim: int list, Target sampling along z, y, x. arr: float, 3D array of map values.

Outputs:

new_arr: float, 3D array
Resampled 3D array.

emda_methods.**rotate_density**(*arr*, *rotmat*, *interp='linear'*)

Returns a rotated array of density

Rotates the array of density using inperpolation.

Arguments:

Inputs:
arr: density as 3D numpy array rotmat: 3 x 3 rotation matrix as 2D numpy array. interp: string.

Type of interpolation to use: cubic or linear. Default is linear

Outputs:
rotated_arr: ndarray. Rotated array.

emda_methods.**scale_map2map**(*staticmap*, *map2scale*, *outfile*)

Scale one map to another map

emda_methods.**set_dim_equal**(*x*)

Sets all dimentions equal and even

This function accepts 3D numpy array and sets its all 3 dims even and equal

Arguments:

Inputs:
x: 3D numpy array

Outputs:
x: 3D numpy array with all dims are even and equal

emda_methods.**set_dim_even**(*x*)

Sets all dimentions even

This function accepts 3D numpy array and sets its all 3 dims even

Arguments:

Inputs:
x: 3D numpy array

Outputs:
x: 3D numpy array with all dims are even

emda_methods.**shift_density**(*arr*, *shift*)

Returns a shifted copy of the input array.

Shift the array using spline interpolation (order=3). Same as Scipy implementation.

Arguments:

Inputs:
arr: density as 3D numpy array shift: sequence. The shifts along the axes.

Outputs:
shifted_arr: ndarray. Shifted array

emda_methods.**singlemap_fsc**(*map1name*, *knl=3*)

Returns Fourier Shell Correlation (FSC) of a map.

Computes Fourier Shell Correlation (FSC) between a map and its reconstituted other half from neighbough Fourier coefficients. This method can be used to estimate FSC based resolution. However, results seem to be reliable when an unfiltered map is used.

Arguments:

Inputs:

map1name: string
Name of the map.

knl: integer, optional
Radius of the integrating kernel.

Outputs:

res_arr: float, 1D array
Linear array of resolution in Angstrom units.

bin_fsc: float, 1D array
Linear array of FSC in each resolution bin.

Outputs reconstituted map as 'fakehalf.mrc'

emda_methods.**sphere_kernel_softedge**(*radius=5*)

Generates a soft-edged spherical kernel.

Arguments:

Inputs:

radius: integer, optional
Radius of integrating kernel in voxels. Default is 5.

Outputs:

kernel: float, 3D array
3D Numpy array of spherical kernel.

emda_methods.**symmetry_average**(*maplist*, *reslist*, *use_peakheight=True*, *peak_cutoff=0.8*, *use_fsc=False*, *fsc_cutoff=0.7*, *ang_tol=5.0*, *pglist=None*, *emdlist=None*, *fobj=None*)

Returns symmetry averaged map.

This function does three difference things:

1. Determines the point group of a map using ProSHADE.
2. Identify group generators and refine them.
3. Generate the full finite point group operators and use them to average the map.

If point groups are supplied, then they are used for symmetry averaging in C, D and O groups. Averaging in T and I groups is done using operators from refined axes. This function can be used to average maps, whose point groups and symmetry operators are not known a priori. NOTE: Current axis refinement for T and I groups may include numerical inaccuracies and so the symmetry averaged map may not be the optimal.

Arguments:

Inputs:
maplist: list of strings List of map names in mrc/map format.

reslist: float List of map resolutions

use_peakheight: bool ProSHADE peaklist is used to decide the point group. Default option.

peak_cutoff: float Cutoff for peak heights in the peaklist. Default is 0.8. However, if the highest peak is lower than this threshold then the cutoff is chosen such that highest - 0.1.

use_fsc: bool If true, FSC is used in place of Proshade peak list to decide the point group.

fsc_cutoff: float Cutoff for FSC. Default is 0.7.

ang_tol: float Tolerence for angle between two axes in the proshade axes list. Default is 5.0 degrees.

emdlist: list of strings EMDB-id list to keep the correspondence in results.

pglist: list of strings List of point groups

Outputs:
symavgmaplist: list of maps List of symmetry averaged maps

emda_methods.**symmetry_average_using_ops**(*imap*, *ops*, *outmapname=None*)

Returns symmetry averaged map using given operators.

This function can be used to average a map by a given set of symmetry operators.

Arguments:

Inputs:
imap: 3D-EM map in mrc/map format. ops: List of symmetry operators to be applied on the map.

List should not contain the identity. An operator must be a 3x3 matrix.

Outputs:
Results in a list: [symmetry-averaged-density, unit-cell, map-origin]

emda_methods.**twomap_fsc**(*map1name*, *map2name*, *fobj=None*, *xmlobj=None*)

Returns Fourier Shell Correlation (FSC) between any two maps.

Computes Fourier Shell Correlation (FSC) using any two maps.

Arguments:

Inputs:

map1name: string
Name of the map 1.

map2name: string
Name of the map 2.

fobj: file object for logging
If present, statistics will be printed into this file.

xmlobj: xml object
If present, statistics will be printed into an XML file.

Outputs:

res_arr: float, 1D array
Linear array of resolution in Angstrom units.

bin_fsc: float, 1D array
Linear array of FSC in each resolution bin.

emda_methods.**write_mrc**(*mapdata*, *filename*, *unit_cell*, *map_origin=None*)

Writes 3D Numpy array into MRC file.

Arguments:

Inputs:

mapdata: float, 3D array
Map values to write

filename: string
Output file name

unit_cell: float, 1D array
Unit cell params

map_origin: list, optional
map origin. Default is [0.0, 0.0, 0.0]

axesorder: string, optional
Axes order can be specified for the data to be written. By defualt EMDA write data in ZXY convension. With this argument, the axes order can be changed.

Outputs:
Output MRC file

emda_methods.**write_mtz**(*uc*, *arr*, *outfile='map2mtz.mtz'*, *resol=None*)

Writes 3D Numpy array into MTZ file.

Arguments:

Inputs:

uc: float, 1D array
Unit cell params.

arr: complex, 3D array
Map values to write.

Outputs:
outfile: string Output file name. Default is map2mtz.mtz.

3.2 emda command line module

```
usage: emda [command] [arguments]
```

3.2.1 Positional Arguments

command Possible choices: info, halffsc, fsc, singlemapfsc, ccmask, mapmask, modelmask, lowpass, power, bfac, resol, half2full, map2mtz, map2mtzfull, transform, mtz2map, resample, resamplemap2map, rcc, bfromcc, mmcc, fcc, mapmodelvalidate, mapmodelfsc, overlay, average, diffmap, applymask, scalemap, bestmap, predfsc, refmac, occ, mirror, model2map, composite, magref, com, fetch, pointgroup, symmetrise, rebox

3.2.2 Sub-commands

3.2.2.1 info

Output basic information about the map.

```
emda info [-h] --map MAP
```

Named Arguments

--map input map

3.2.2.2 halffsc

Calculates FSC between half-maps.

```
emda halffsc [-h] --h1 H1 --h2 H2 [--msk MSK] [--out OUT] [--phaserand]
```

Named Arguments

--h1 input map1 map

--h2 input map2 map

--msk input mask (mrc/map)

--out output data table

Default: "table_variances.txt"

--phaserand use if phase randomized FSC is calculated

Default: False

3.2.2.3 fsc

Calculates FSC between two maps.

```
emda fsc [-h] --map1 MAP1 --map2 MAP2
```

Named Arguments

--map1 input map1 map

--map2 input map2 map

3.2.2.4 singlemapfsc

Calculates FSC using neighbour average.

```
emda singlemapfsc [-h] --h1 H1 [--knl KNL]
```

Named Arguments

--h1	input map1 map
--knl	kernel radius in voxels
	Default: 3

3.2.2.5 ccmask

Generates a mask based on halfmaps correlation.

```
emda ccmask [-h] --h1 H1 --h2 H2 [--knl KNL] [--itr ITR]
            [--dthreshold DTHRESHOLD]
```

Named Arguments

--h1	input halfmap1 map
--h2	input halfmap2 map
--knl	kernel radius in voxels
	Default: 4
--itr	number of dilation cycles
	Default: 1
--dthreshold	threshold for density

3.2.2.6 mapmask

Generate a mask from a map.

```
emda mapmask [-h] --map MAP [--knl KNL] [--prb PRB] [--itr ITR] [--res RES]
             [--fil FIL]
```

Named Arguments

--map	input map
--knl	kernel radius in voxels
	Default: 5
--prb	density cutoff probability in cumulative density function
	Default: 0.99
--itr	number of dilate iterations
	Default: 3
--res	lowpass resolution in Angstroms
	Default: 15.0

--fil filter type to use: ideal or butterworth
Default: "butterworth"

3.2.2.7 modelmask

Generate a mask from an atomic model.

```
emda modelmask [-h] --map MAP --mdl MDL [--atmrad ATMRAD]
```

Named Arguments

--map input map MRC/MAP
--mdl input atomic model PDB/CIF
--atmrad radius of the atomic sphere in Angstroms
Default: 3.0

3.2.2.8 lowpass

Lowpass filter to specified resolution.

```
emda lowpass [-h] --map MAP --res RES [--fil FIL]
```

Named Arguments

--map input map (mrc/map)
--res lowpass resolution in Angstrom
--fil filter type to use: ideal or butterworth
Default: "ideal"

3.2.2.9 power

Calculates power spectrum.

```
emda power [-h] --map MAP
```

Named Arguments

--map input map (mrc/map)

3.2.2.10 bfac

Apply a B-factor on the map.

```
emda bfac [-h] --map MAP --bfc BFC [BFC ...] [--out]
```

Named Arguments

--map	input map (mrc/map)
--bfc	bfactor(s) to apply
--out	if use, writes out map Default: False

3.2.2.11 resol

Estimates map resolution based on FSC.

```
emda resol [-h] --h1 H1 --h2 H2
```

Named Arguments

--h1	input halfmap1 map
--h2	input halfmap2 map

3.2.2.12 half2full

Combine two halfmaps to make the fullmap.

```
emda half2full [-h] --h1 H1 --h2 H2 [--out OUT]
```

Named Arguments

--h1	input halfmap1 map
--h2	input halfmap2 map
--out	output map (mrc/map) Default: “fullmap.mrc”

3.2.2.13 map2mtz

Convert MRC/MAP to MTZ.

```
emda map2mtz [-h] --map MAP [--res RES] [--out OUT]
```

Named Arguments

--map	input map (mrc/map)
--res	resolution cutoff (A). default Nyquist
--out	output map (mtz) Default: "map2mtz.mtz"

3.2.2.14 map2mtzfull

Convert MRC/MAP to MTZ using half maps.

```
emda map2mtzfull [-h] --h1 H1 --h2 H2 [--out OUT]
```

Named Arguments

--h1	input hfmap1 (mrc/map)
--h2	input hfmap2 (mrc/map)
--out	output map (mtz) Default: "map2mtzfull.mtz"

3.2.2.15 transform

Apply a transformation on the map.

```
emda transform [-h] --map MAP [--tra TRA [TRA ...]] [--rot ROT]
               [--axr AXR [AXR ...]] [--out OUT]
```

Named Arguments

--map	input map (mrc/map)
--tra	translation vec. in Angstrom. eg 1.0 0.0 0.0 Default: [0.0, 0.0, 0.0]
--rot	rotation in degree Default: 0.0
--axr	rotation axis Default: [1, 0, 0]

--out output map (mrc/map)
Default: "transformed.mrc"

3.2.2.16 mtz2map

Convert MTZ to MRC/MAP.

```
emda mtz2map [-h] --mtz MTZ --map MAP --out OUT
```

Named Arguments

--mtz input map (mtz)

--map input map (mrc/map)

--out output map (mrc/map)

3.2.2.17 resample

Resample a map in Fourier space.

```
emda resample [-h] --map MAP --pix PIX [PIX ...] --dim DIM [DIM ...]
              [--cel CEL [CEL ...]] [--out OUT]
```

Named Arguments

--map input map (mrc/map)

--pix target pixel size (A)

--dim target map dimensions. e.g. 100 100 100

--cel target cell. e.g. a b c 90 90 90

--out output map name
Default: "resampled.mrc"

3.2.2.18 resamplemap2map

Resample map2 on map1.

```
emda resamplemap2map [-h] --map1 MAP1 --map2 MAP2 [--out OUT]
```

Named Arguments

--map1	static map (mrc/map)
--map2	map to resample (mrc/map)
--out	output map name
	Default: "resampled2staticmap.mrc"

3.2.2.19 rcc

real space correlation

```
emda rcc [-h] --h1 H1 --h2 H2 [--mdl MDL] [--res RES] [--msk MSK] [--nrm]
         [--knl KNL] [--lgf LGF]
```

Named Arguments

--h1	input halfmap1 map
--h2	input halfmap2 map
--mdl	Input model (cif/pdb)
--res	Resolution (A)
--msk	input mask (mrc/map)
--nrm	if True use normalized maps
	Default: False
--knl	Kernel size (pixels)
	Default: 5
--lgf	ligand description file

3.2.2.20 bfromcc

local b from real space correlation

```
emda bfromcc [-h] --h1 H1 --h2 H2 --res RES [--msk MSK] [--knl KNL]
```

Named Arguments

--h1	input halfmap1 map
--h2	input halfmap2 map
--res	Resolution (A)
--msk	input mask (mrc/map)
--knl	Kernel size (pixels)
	Default: 5

3.2.2.21 mmcc

real space correlation

```
emda mmcc [-h] --map MAP --mdl MDL --res RES [--nrm] [--msk MSK] [--knl KNL]
          [--lgf LGF]
```

Named Arguments

--map	input full/deposited map
--mdl	Input model (cif/pdb)
--res	Resolution (A)
--nrm	if use, normalized maps are used
	Default: False
--msk	input mask (mrc/map)
--knl	Kernel size (pixels)
	Default: 5
--lgf	ligand description file

3.2.2.22 fcc

Fourier space correlation

```
emda fcc [-h] --h1 H1 --h2 H2 [--knl KNL] [--msk MSK]
```

Named Arguments

--h1	input halfmap1 map
--h2	input halfmap2 map
--knl	Kernel size (pixels)
	Default: 5
--msk	input mask (mrc/map)

3.2.2.23 mapmodelvalidate

map-model validation using FSC

```
emda mapmodelvalidate [-h] --h1 H1 --h2 H2 --mdf MDF [--md1 MD1] [--msk MSK]
                      [--res RES] [--bfc BFC] [--lgf LGF]
```

Named Arguments

--h1	input halfmap1 map
--h2	input halfmap2 map
--mdf	input full atomic model
--md1	input halfmap1 atomic model
--msk	input mask (mrc/map)
--res	Resolution (A)
--bfc	Overall B factor for model. default=0.0
	Default: 0.0
--lgf	ligand description file

3.2.2.24 mapmodelfsc

map-model FSC

```
emda mapmodelfsc [-h] --map MAP --mdl MDL [--msk MSK] --res RES [--bfc BFC]
                 [--lgf LGF] [--phaserand]
```

Named Arguments

--map	input map
--mdl	input atomic model
--msk	input mask (mrc/map)
--res	Resolution (A)
--bfc	Overall B factor for model. default=0.0 ignored by REFMAC
	Default: 0.0
--lgf	ligand description file
--phaserand	use if phase randomized FSC is calculated
	Default: False

3.2.2.25 overlay

overlay maps

```
emda overlay [-h] --map MAP [MAP ...] [--msk MSK [MSK ...]]
             [--tra TRA [TRA ...]] [--rot ROT] [--axr AXR [AXR ...]]
             [--ncy NCY] [--res RES] [--fitres FITRES] [--int INT]
             [--modelres MODELRES] [--hfm] [--mod] [--usecom]
```

Named Arguments

--map	maplist for overlay
--msk	masklist for overlay
--tra	translation vector. default=[0.0, 0.0, 0.0]
	Default: [0.0, 0.0, 0.0]
--rot	rotation in deg. default=0.0
	Default: 0.0
--axr	rotation axis. default=[1,0,0]
	Default: [1, 0, 0]
--ncy	number of fitting cycles. default=5
	Default: 5
--res	starting fit resol. (A). default= 6 A
	Default: 6
--fitres	final fit resol. (A). default= 0.0 A
	Default: 0.0
--int	interpolation method (linear/cubic). default= linear
	Default: "linear"
--modelres	model resol. (A). default= 5 A
	Default: 5
--hfm	if use employ half maps
	Default: False
--mod	if use calls model overlay
	Default: False
--usecom	if used, center-of-mass is used to superimpose maps
	Default: False

3.2.2.26 average

weighted average of several maps

```
emda average [-h] --map MAP [MAP ...] [--msk MSK [MSK ...]]
             [--tra TRA [TRA ...]] [--rot ROT] [--axr AXR [AXR ...]]
             [--ncy NCY] [--res RES] [--int INT]
```

Named Arguments

--map	maplist to average
--msk	masklist for maps
--tra	translation vec. Default: [0.0, 0.0, 0.0]
--rot	rotation in deg Default: 0.0
--axr	rotation axis Default: [1, 0, 0]
--ncy	number of fitting cycles Default: 10
--res	starting fit resol. (A) Default: 6
--int	interpolation method ([linear]/cubic) Default: "linear"

3.2.2.27 diffmap

Calculate the difference map

```
emda diffmap [-h] --map MAP [MAP ...] [--msk MSK [MSK ...]] --res RES
             [--mod MOD] [--fit] [--ncy NCY] [--fitres FITRES] [--usecom]
             [--usehalfmaps]
```

Named Arguments

--map	maplist to diffmap
--msk	masklist for maps
--res	resolution for difference map in Angstroms.
--mod	scaling method. norm (default) - normalized FC, ampli - amplitudes in resolution bins Default: "norm"
--fit	if used, maps are superimposed before calculating difference map Default: False
--ncy	number of fitting cycles Default: 5
--fitres	final fit resol. (A). default= 0.0 A Default: 0.0

--usecom	if used, center-of-mass is used to superimpose maps Default: False
--usehalfmaps	if used, halfmaps are used to calculate difference map Default: False

3.2.2.28 applymask

apply mask on the map

```
emda applymask [-h] --map MAP --msk MSK [--out OUT]
```

Named Arguments

--map	map to be masked
--msk	mask to be applied
--out	output map name Default: "mapmasked.mrc"

3.2.2.29 scalemap

scale onemap to another

```
emda scalemap [-h] --m1 M1 --m2 M2 [--out OUT]
```

Named Arguments

--m1	input map
--m2	map to be scaled
--out	output map name Default: "scaledmap.mrc"

3.2.2.30 bestmap

calculate bestmap

```
emda bestmap [-h] --h1 H1 --h2 H2 [--msk MSK] [--B B] [--knl KNL] [--mod MOD]
             [--out OUT]
```

Named Arguments

--h1	input halfmap1 map
--h2	input halfmap2 map
--msk	mask to be applied
--B	relative B
--knl	kernel radius (pixels) Default: 5
--mod	fsc type (1-resol bins, 2-local) Default: 1
--out	output map name Default: "bestmap.mrc"

3.2.2.31 predfsc

predict FSC based on # particles

```
emda predfsc [-h] --h1 H1 --h2 H2 [--msk MSK] [--npa NPA [NPA ...]]
             [--bfc BFC [BFC ...]]
```

Named Arguments

--h1	input halfmap1 map
--h2	input halfmap2 map
--msk	mask map
--npa	n fold of particles
--bfc	list of B factors

3.2.2.32 refmac

prepare data for refmac refinement

```
emda refmac [-h] --h1 H1 --h2 H2 [--msk MSK] [--bfc BFC [BFC ...]] [--out OUT]
```

Named Arguments

--h1	input halfmap1 map
--h2	input halfmap2 map
--msk	mask map
--bfc	b-factor list
--out	output mtz file name Default: "output.mtz"

3.2.2.33 occ

overall correlation in real space

```
emda occ [-h] --m1 M1 --m2 M2 [--msk MSK] [--spc SPC]
```

Named Arguments

--m1	input map1 map
--m2	input map2 map
--msk	mask map
--spc	space (real/fourier) for CC calculation Default: "real"

3.2.2.34 mirror

mirror the map

```
emda mirror [-h] --map MAP
```

Named Arguments

--map	input map

3.2.2.35 model2map

calculate model based map

```
emda model2map [-h] --mdl MDL --res RES --dim DIM [DIM ...] --cel CEL
               [CEL ...] [--lgf LGF] [--org ORG [ORG ...]] [--gemmi]
```

Named Arguments

--mdl	input atomic model
--res	Resolution (A)
--dim	map dim
--cel	cell parameters
--lgf	ligand description file
--org	map origin
--gemmi	if used, GEMMI is used instead REFMAC for structure factor calculation Default: False

3.2.2.36 composite

make composite map

```
emda composite [-h] --map MAP [MAP ...] [--msk MSK [MSK ...]]
```

Named Arguments

--map	maplist to combine
--msk	masklist for maps

3.2.2.37 magref

magnification refinement

```
emda magref [-h] --map MAP [MAP ...] --ref REF
```

Named Arguments

--map	maplist to correct for magnification [.mrc/.map]
--ref	reference map [.mrc/.map]

3.2.2.38 com

center of mass

```
emda com [-h] --map MAP [--msk MSK]
```

Named Arguments

--map	input map (MRC/MAP)
--msk	mask to apply on the map

3.2.2.39 fetch

fetch EMmap and model

```
emda fetch [-h] --emd EMD [EMD ...] [--all]
```

Named Arguments

--emd	list of EMD entries. e.g. 3651
--all	Use to download all data (mask, map, halfdata, model) Default: False

3.2.2.40 pointgroup

detect point group from the map

```
emda pointgroup [-h] --map MAP [MAP ...] --res RES [RES ...]
                [--emd EMD [EMD ...]] [--peak_cutoff PEAK_CUTOFF] [--use_fsc]
                [--fsc_cutoff FSC_CUTOFF] [--ang_tol ANG_TOL]
```

Named Arguments

--map	list of maps to find point groups
--res	list of resolution of maps (A)
--emd	list of emdbid of maps
--peak_cutoff	cutoff for Proshade peak height. default= 0.8 Default: 0.8
--use_fsc	if used, FSC is used in place for proshade peakheight to decide point group Default: False
--fsc_cutoff	cutoff for Proshade peak height, default= 0.7 Default: 0.7
--ang_tol	angle tolerence between two axes for determining point group. default= 5 deg. Default: 5.0

3.2.2.41 symmetrise

symmetrize map using point group symmetry

```
emda symmetrise [-h] --map MAP [MAP ...] --res RES [RES ...]
                [--emd EMD [EMD ...]]
                [--pointgroup POINTGROUP [POINTGROUP ...]]
                [--peak_cutoff PEAK_CUTOFF] [--use_fsc]
                [--fsc_cutoff FSC_CUTOFF] [--ang_tol ANG_TOL]
```

Named Arguments

--map	list of maps to find point groups
--res	list of resolution of maps (A)
--emd	list of emdbid of maps
--pointgroup	list of point groups
--peak_cutoff	cutoff for Proshade peak height. default= 0.8
	Default: 0.8
--use_fsc	if used, FSC is used in place for proshade peakheight to decide point group
	Default: False
--fsc_cutoff	cutoff for Proshade peak height, default= 0.7
	Default: 0.7
--ang_tol	angle tolerence between two axes for determining point group. default= 5 deg.
	Default: 5.0

3.2.2.42 rebox

rebox map and model using a mask

```
emda rebox [-h] --map MAP [MAP ...] --msk MSK [MSK ...] [--mdl MDL [MDL ...]]
```

Named Arguments

--map	list of map names for reboxing
--msk	list of mask names for reboxing
--mdl	list of model names (pdb/cif) for reboxing

PYTHON MODULE INDEX

e

INDEX

S

T

W