density2graph

▶ View Source

def load_density_file(fname): ▶ View Source

load a .mrc file using the mrcfile package

Args

- **fname ([str]):** filename / filepath

Returns

[mrcfile object]: MRC data

def normalize_and_threshold_data(mrc, t, noise_stdev=0.0, norm_T=False): ▶ View Source

normalizes threshold value and densities then applies a cutoff threshold

Args

- **mrc ([mrcfile object]):** mrc data
- **t ([float]):** raw (unormalized) pixel intensity cutoff threshold
- **noise_stdev ([float]):** Standard deviation of Gaussian noise (mean=0) to add. Default is 0 (no noise added)
- **norm_T ([bool]):** threshold value is normalized (True) or not normalized (False). Default is False.

Returns

[numpy array]: array of x,y,z coordinates which are above the cutoff threshold. A[0] = [x0,y0,z0]

def cluster_data(xyz_data, DBSCAN_epsilon, DBSCAN_min_samples): ▶ View Source

Clusters data using DBSCAN from sklearn

Args

- **xyz_data ([numpy array]):** A[0] = [x0,y0,z0]
- **DBSCAN_epsilon ([float]):** DBSCAN epsilon value (in pixels)
- **DBSCAN_min_samples ([int]):** DBSCAN min_samples

Returns

[sklearn DBSCAN cluster object]: clustering results stored in an object

def get_cluster_centroids(xyz_data, model): ▶ View Source

Coarse grain density model using cluster centroids

Args

- **xyz_data ([numpy array]):** A[0] = [x0,y0,z0]
- **model ([sklearn DBSCAN cluster object]):** clustering results stored in an object

Returns

[numpy array]: array of cluster centroids, A[0] = [centroid_x0, centroid_y0, centroid_z0]

def plot_clustering_results(xyz_data, coarse_model, figsize=3): ▶ View Source

creates a 3D scatter plot containing both the xyz data and the cluster centroids.

Note: should rotate afterwards for better visualization.

Args

- **xyz_data ([numpy array]):** A[0] = [x0,y0,z0]
- **coarse_model ([numpy array]):** array of cluster centroids, A[0] = [centroid_x0, centroid_y0, centroid_z0]

Returns

[matplotlib figure object]: 3d scatter plot figure

def create_and_save_graph(coarse_model, proximity_px, out_fname, save=True): ▶ View Source

creates a Networkx graph from the coarse grained model (cluster centroids) and saves it as a graph XML file (.gexf)

Args

- **coarse_model ([numpy array]):** array of cluster centroids, A[0] = [centroid_x0, centroid_y0, centroid_z0]
- **proximity_px ([float]):** pairwise cutoff distance for assigning edges to nodes, in pixels.
- **out_fname ([string]):** filename for output
- **save ([boolean]):** flag to save file (True) or not (False)

Returns

[networkx graph object]: graph representation of coarse model (cluster centroids)

def add_Gaussian_noise(mrc, loc=0.0, scale=1.0): ▶ View Source

Adds Gaussian white noise to the data in an mrc file

ref: https://numpy.org/doc/stable/reference/random/generated/numpy.random.normal.html

Args

- **mrc (mrcfile object):** mrc object to add noise to
- **loc (float, optional):** mean of Gaussian distribution. Defaults to 0.
- **scale (float, optional):** standard deviation of Gaussian distribution. Defaults to 1.

Returns

[numpy array]: data with noise added

def main(args): ▶ View Source

Takes a 3D density (.mrc), applies threshold, coarse-grains data, and converts it into a graph network. Outputs a .png file of the coarse grained model, and a .gexf graph xml file.

Args

- **args ([argument parser object]):** - args.fname: .mrc filename (white density w/ black background)
    - args.t: unormalized pixel intensity threshold level
    - args.eps: DBSCAN epsilon (inter cluster distance)
    - args.ms: DBSCAN min samples (minimum number of samples in cluster)
    - args.d_cut: pairwise distance cutoff for assigning edges to graph, in pixels