pdb2graph

▶ View Source

def get_hydrophobicity(name, warn=False):

▶ View Source

Gets hydophobicity based on amino acid name.

ref: https://www.cgl.ucsf.edu/chimera/docs/UsersGuide/midas/hydrophob.html

note: returns NaN if input name is invalid!

Args

- **name ([str]):** Amino acid name
- **warn ([bool]):** if True, print a warning when hydrophobicity can't be determined

Returns

[float]: hydrophobicity

def PDB_to_df(pdb_code, fname, pdbx, offset, CA_only=1):

▶ View Source

Loads a PDB (or PDBx) file and stores the atom coordinates and residue name and number into a dataframe.

Note: if the PDB file has more than one model, the first model is chosen.

Args

- **pdb_code ([str]):** PDB ID / label for the protein of interest
- **fname ([str]):** filename for the protein of interest. Can be PDB or PDBx format
- **pdbx ([int]):** Set=1 if using the newer PDBx file format.
- **offest ([int]):** index offset incase first residue ID in PDB file is not the first physical residue (e.g. PDB starts at 5th residue).
- **CA_only ([int]):** Set=1 [default] if using only alpha carbons, else all atoms are used.

Returns

[Pandas dataframe object]: dataframe containing every atom's x,y,z coord and serial number

def PDB_df_to_G(PDB_df, d_cut=8.0):

▶ View Source

Converts a dataframe containing alpha carbon / atom coordinates (in Angstroms) into a graph, G(V,E).

Each vertex, V, is an alpha carbon / atom. Two alpha carbons / atoms with a distance (in Angstroms) less than a cutoff, d_cut, are connected by an edge, E.

Args

- **PDB_df ([Pandas dataframe object]):** a dataframe containing alpha carbon / atom coordinate columns labeled: 'x', 'y', and 'z'
- **d_cut ([float]):** Threshold for two alpha carbons / atoms to be connected (in Angstroms) by an edge. Defaults to 8.0

Returns

G ([networkX graph object]): protein structure network graph, G(V,E)

def save_data(df, G, df_name, G_name): ▶ View Source

Convenience function that stores dataframe as .csv and graph as .gexf file

Args

- **df ([Pandas dataframe object]):** dataframe to save
- **G ([NetworkX graph object]):** graph to save
- **df_name ([str]):** output filename for dataframe .csv
- **G_name ([str]):** output filename for graph .gexf

def save_data_at_this_folder(data_path, df, G, df_name, G_name): ▶ View Source

Convenience function that stores dataframe as .csv and graph as .gexf file

Args

- **data_path ([str] or [Path]):** output directory path
- **df ([Pandas dataframe object]):** dataframe to save
- **G ([NetworkX graph object]):** graph to save
- **df_name ([str]):** output filename for dataframe .csv
- **G_name ([str]):** output filename for graph .gexf

def plot_coordinates(xyz_data, figsize=5): ▶ View Source

creates a 3D scatter plot containing the xyz data

Args

- **xyz_data ([numpy array]):** A[0] = [x0,y0,z0]
- **figsize (int, optional):** size of figure (figsize x figsize). Defaults to 5.

Returns

[matplotlib figure object]: 3d scatter plot figure

def plot_FA_and_CA_coordinates(FA_xyz, CA_xyz, figsize=5):

▶ View Source

creates a 3D scatter plot containing CA and FA atom coordinates

Args

- **FA_xyz ([numpy array]):** A[0] = [x0,y0,z0] for all atom coordinate data
- **CA_xyz ([numpy array]):** A[0] = [x0,y0,z0] for alpha carbon only coordinate data
- **figsize (int, optional):** size of figure (figsize x figsize). Defaults to 5.

Returns

[matplotlib figure object]: 3d scatter plot figure

def main(args):

▶ View Source

Takes a .pdb(x) file, converts it into a graph, and saves the atom coordinates to .csv and graph as .gexf

Args

- **args ([argument parser object]):** - args.pdb_code: PDB id / protein name
    - args.fname: PDB/PDBx filename
    - args.d_cut: Alpha Carbon / atom pairwise contact distance cutoff (in Angstroms)
    - args.o: PDB residue index offset integer. Default is 0.
    - args.pdbx: set=1 to use pdbx file parser
    - args.CA_only: set=1 to use only alpha carbons (0 for all atoms)