Table of Contents

1. Code

1.1 AIzymes

1.1.1: AIzymes_MAIN

Main class for managing AIzymes workflow, including setup, initialization, control, and plotting functions.

1.1.2: __init__

Initializes an instance of the AIzymes_MAIN class.

1.1.3: setup

Sets up the AIzymes project environment with specified parameters.

Parameters:
FOLDER_HOME (str): Path to the main folder.
FOLDER_PARENT (str): Path to the parent folder.
CST_NAME (str): Constraint name.
WT (str): Wild type information.
LIGAND (str): Ligand data.
DESIGN (str): Design specifications.
MAX_JOBS (int): Maximum number of jobs to run concurrently.
N_PARENT_JOBS (int): Number of parent jobs.
MAX_DESIGNS (int): Maximum number of designs.
KBT_BOLTZMANN (list): Boltzmann constant values.
CST_WEIGHT (float): Constraint weight.
ProteinMPNN_PROB (float): Probability parameter for ProteinMPNN.
ProteinMPNN_BIAS (float): Bias parameter for ProteinMPNN.
LMPNN_PROB (float): Probability parameter for LMPNN.
FOLDER_MATCH (str): Path to match folder.
ProteinMPNN_T (str): Temperature for ProteinMPNN.
LMPNN_T (str): Temperature for LMPNN.
LMPNN_BIAS (float): Bias parameter for LMPNN.
SUBMIT_PREFIX (str): Submission prefix.
SYSTEM (str): System information.
MATCH (str): Match specifications.
ROSETTA_PATH (str): Path to Rosetta source.
EXPLORE (bool): Whether to explore parameter space.
FIELD_TOOLS (str): Path to FieldTools script.
LOG (str): Logging level.
PARENT_DES_MED (str): Parent design method.

1.1.4: initialize

Initializes AIzymes with given parameters.

Parameters:
FOLDER_HOME (str): Path to the main folder.
UNBLOCK_ALL (bool): Flag to unblock all processes.
PRINT_VAR (bool): Flag to print variables.
PLOT_DATA (bool): Flag to plot data.
LOG (str): Logging level.

1.1.5: controller

Controls the AIzymes project based on scoring and normalization parameters.

Parameters:
HIGHSCORE (float): High score threshold for evaluation.
NORM (dict): Normalization values for different scores.

1.1.6: plot

Generates plots based on AIzymes data, including main, tree, and landscape plots.

Parameters:
main_plots (bool): Flag to generate main plots.
tree_plot (bool): Flag to generate tree plot.
landscape_plot (bool): Flag to generate landscape plot.
print_vals (bool): Flag to print values on plots.
NORM (dict): Normalization values for different scores.
HIGHSCORE_NEGBEST (dict): High score and negative best score for different metrics.

1.2 design_ESMfold

1.2.1: prepare_ESMfold

Predicts structure of sequence in {index} using ESMfold.

Parameters:
index (str): The index of the protein variant to be predicted.
cmd (str): Growing list of commands to be exected by run_design using submit_job.

Returns:
cmd (str): Command to be exected by run_design using submit_job.

1.3 design_LigandMPNN

1.3.1: prepare_LigandMPNN

Executes the LigandMPNN pipeline for a given protein-ligand structure and generatesnew protein sequences with potentially higher functional scores considering the ligand context.

Parameters:
- **parent_index (str)**: The index of the parent protein variant.
- **new_index (str)**: The index assigned to the new protein variant.
- **all_scores_df (DataFrame)**: A DataFrame containing information for protein variants.

1.4 design_match

1.5 design_ProteinMPNN

1.5.1: prepare_ProteinMPNN

Executes the ProteinMPNN pipeline for a given protein structure and generatesnew protein sequences with potentially higher functional scores.

Parameters:
- **new_index (str)**: The index of the designed variant.
- **cmd (str)**: Growing list of commands to be exected by run_design using submit_job.

Returns:
- **cmd (str)**: Command to be exected by run_design using submit_job

Note:This function assumes the ProteinMPNN toolkit is available and properly set up in the specified location.It involves multiple subprocess calls to Python scripts for processing protein structures and generating new sequences.

1.6 design_RosettaDesign

1.6.1: prepare_RosettaDesign

Designs protein structure in {new_index} based on {parent_index} using RosettaDesign.

Parameters:
- **parent_index (str)**: Index of the parent protein variant to be designed.
- **new_index (str)**: Index assigned to the resulting design.
- **input_suffix (str)**: Suffix of the input structure to be used for design.

Returns:
- **cmd (str)**: Command to be exected by run_design using submit_job.

1.7 design_RosettaRelax

1.7.1: prepare_RosettaRelax

Relaxes protein structure in {index} using RosettaRelax.

Parameters:
- **index (str)**: The index of the protein variant to be relaxed.
- **cmd (str)**: collection of commands to be run, this script wil append its commands to cmd

Optional parameters:- PreMatchRelax (bool): True if ESMfold to be run without ligand (prior to RosettaMatch).

1.8 helper

1.8.1: run_command

Wrapper to execute .py files in runtime with arguments, and print error messages if they occur.

Parameters:
- **`command`**: The command to run as a list of strings.
- **`cwd`**: Optional; The directory to execute the command in.
- **`capture_output`**: Optional; If True, capture stdout and stderr. Defaults to False (This is to conserve memory).

1.8.2: get_PDB_in

Based on index, find the input PDB files for the AIzymes modules
Paramters:- index: The indix of the current design
Output:- PDBfile_Design_in: Input file for RosettaDesign- PDBfile_Relax_in: Input file for RosettaRelax- PDBfile_Relax_ligand_in: Input file for Ligand to be used in RosettaRelax

1.8.3: save_cat_res_into_all_scores_df

Finds the indices and names of the catalytic residue from Saves indices and residues into in row as lists.To make sure these are saved and loaded as list, ";".join() and .split(";") should be usedIf information is read from an input structure for design do not save cat_resn

1.8.4: reset_to_after_index

This function resets the run back to a chosen index. It removes all later entries from the all_scores.csv and the home dir.index: The last index to keep, after which everything will be deleted.

1.8.5: wait_for_file

Wait for a file to exist and have a non-zero size.

1.9 main_design

1.10 main_running

1.10.1: start_controller

Runs the main loop to manage the design process until the maximum number of designs is reached.
Continues submitting jobs, monitoring running processes, and updating scores based on Boltzmann selectionuntil `MAX_DESIGNS` is achieved. The function pauses or starts new designs based on system resources.

Parameters:
`self`: An instance of the AIzymes_MAIN class with setup attributes and properties.

1.10.2: check_running_jobs

Checks the current number of running jobs based on the system type.
Depending on the value of `SYSTEM`, this function counts the active jobs in GRID, BLUEPEBBLE, BACKGROUND_JOB,or ABBIE_LOCAL systems.

Returns:
`int`: Number of running jobs for the specific system.

1.10.3: update_potential

Updates the potential file for a given score type at the specified variant index.
Creates or appends to a `_potential.dat` file in `FOLDER_HOME/`, calculating andupdating potentials for the parent variant if necessary.

Parameters:
`score_type (str)`: Type of score to update (e.g., total, interface, catalytic, efield).
`index (int)`: Variant index to update potential data.

1.10.4: update_potential

1.10.4: update_potential

Creates a _potential.dat file in FOLDER_HOME/
If latest score comes from Rosetta Relax - then the score for variant will be added to the _potential.dat of the parent indexand the _potential value of the dataframe for the parentwill be updated with the average of the parent and child scores.

Parameters:
- **score_type(str)**: Type of score to update, one of these options: total, interface, catalytic, efield
- **index (int)**: variant index to update

1.10.5: update_scores

Updates various scores, including total, interface, catalytic, and efield scores for each design variant.
This function iterates over design variants, updating scores based on files generated by different processes.It also updates sequence information, tracks mutations, and saves the updated DataFrame.

Parameters:
self: An instance of the AIzymes_MAIN class with setup attributes and properties.

1.10.6: boltzmann_selection

Selects a design variant based on a Boltzmann-weighted probability distribution.
Filters variants based on certain conditions (e.g., scores, block status), then computes probabilitiesusing Boltzmann factors with a temperature factor (`KBT_BOLTZMANN`) to select a variant for further design steps.

Returns:
int: Index of the selected design variant.

1.10.7: check_parent_done

Determines if parent designs are complete based on the number of generated designs and parent jobs.

Returns:
bool: True if parent designs are complete, otherwise False.

1.10.8: start_parent_design

Initiates a new design process for a parent structure by creating a new variant entry.
Sets up the required files and configuration for designing a parent structure, then calls the design methodspecified in `PARENT_DES_MED`.

Parameters:
self: An instance of the AIzymes_MAIN class with setup attributes and properties.

1.10.9: start_calculation

Decides the next calculation step for the specified design variant index.
Based on the current design state, this function decides to run ESMfold, RosettaRelax, or a design methodfor the given index.

Parameters:
parent_index (int): Index of the variant to evaluate for further calculations.

1.10.10: create_new_index

Creates a new design entry in `all_scores_df` with a unique index, inheriting attributes from a parent variant.
Updates the DataFrame with new index information, saves the updated file, and sets up the directory structurefor the new design variant.

Parameters:
parent_index (str): The index of the parent variant or "Parent" for initial designs.

Returns:
int: The newly created index for the variant.

1.11 main_scripts

1.12 main_startup

1.13 plotting

1.13.1: plot_interface_v_total_score_selection

Plots a scatter plot of total_scores vs interface_scores and highlights the pointscorresponding to the selected indices.

Parameters:
- **ax (matplotlib.axes.Axes)**: The Axes object to plot on.
- **total_scores (list or np.array)**: The total scores of the structures.
- **interface_scores (list or np.array)**: The interface scores of the structures.
- **selected_indices (list of int)**: Indices of the points to highlight.

1.13.2: plot_interface_v_total_score_generation

Plots a scatter plot of total_scores vs interface_scores and colors the pointsaccording to the generation for all data points, using categorical coloring.Adds a legend to represent each unique generation with its corresponding color.

Parameters:
- **ax (matplotlib.axes.Axes)**: The Axes object to plot on.
- **total_scores (list or np.array)**: The total scores of the structures.
- **interface_scores (list or np.array)**: The interface scores of the structures.
- **generation (pd.Series or np.array)**: Generation numbers for all data points.

1.13.3: plot_stacked_histogram_by_cat_resi

Plots a stacked bar plot of interface scores colored by cat_resi on the given Axes object,where each bar's segments represent counts of different cat_resi values in that bin.

Parameters:
- **ax (matplotlib.axes.Axes)**: The Axes object to plot on.
- **all_scores_df (pd.DataFrame)**: DataFrame containing 'cat_resi' and 'interface_score' columns.
- **color_map (dict)**: Optional; A dictionary mapping catalytic residue indices to colors.
- **show_legend (bool)**: Optional; Whether to show the legend. Defaults to False.

1.13.4: plot_stacked_histogram_by_cat_resn

Plots a stacked bar plot of interface scores colored by cat_resn on the given Axes object,where each bar's segments represent counts of different cat_resn values in that bin.

Parameters:
- **ax (matplotlib.axes.Axes)**: The Axes object to plot on.
- **all_scores_df (pd.DataFrame)**: DataFrame containing 'cat_resn' and 'interface_score' columns.

1.13.5: plot_stacked_histogram_by_generation

Plots a stacked bar plot of interface scores colored by generation on the given Axes object,where each bar's segments represent counts of different generation values in that bin.

Parameters:
- **ax (matplotlib.axes.Axes)**: The Axes object to plot on.
- **all_scores_df (pd.DataFrame)**: DataFrame containing 'generation' and 'interface_score' columns.

1.14 scoring_efields

1.14.1: generate_AMBER_files

Uses tleap to create a .parm7 and .rst7 file from a pdb. Requires ambertools.Also requires 5TS.prepi and 5TS.frcmod in the INPUT folderTODO: Add script to generate these if not present.

Parameters:

- **filename (str)**: The path to the pdb file to analyse without the file extension.

1.14.2: calc_efields_score

Executes the FieldTools.py script to calculate the electric field across the C-H bond of 5TS.Requires a field_target.dat in the Input folder. Currently hard-coded based on 5TSTODO: Make this function agnostic to contents of field_target

Parameters:
- **pdb_path (str)**: The path to the pdb structure to analyse - either from design or relax.

Returns:
- **bond_field (float)**: The total electric field across the 5TS@C9_:5TS@H04 bond in MV/cm. (Currently hard coded to these atoms)
- **all_fields (dict)**: The components of the electric field across the 5TS@C9_:5TS@H04 bond per residue.

1.14.3: update_efieldsdf

Adds a new row to "{FOLDER_HOME}/electric_fields.csv" containing the electric fieldsgenerated by FieldTools.py for all residues in the protein

1.15 setup_system