ProtPred Pipeline

03-713 Bioinformatics Data Practicum

2024-04-26

Compares protein structures using ESM-Fold

Anagha Anil, Manjot Nagyal, Andrew Lutsky, and Jonathan Zhu

https://github.com/jonazhu/protpredpipeline

Protein Prediction Pipeline is a workflow to model the effects of mutations on protein structure. This pipeline can has other functionality such as homologous protein structure comparison.

Table of contents

Part I.
About

This document describes a computational pipeline, ProtPred Pipeline for analyzing and comparing protein structures, with a focus on studying mutations in the ribosomal protein subunit 19 (RPS19) associated with Diamond Blackfan Anemia as the primary example. The pipeline integrates various tools and packages to process input data, predict protein structures, perform structural alignments, calculate properties, and visualize the results.

The pipeline is designed to be flexible and modular, allowing users to input either FASTA files containing known protein sequences or a list of UniProt protein IDs. It also includes scripts for handling point mutations and generating FASTA files from UniProt IDs.

Part II.
Set up

The installation of the following packages is required:

`python 3.X:` The scripts are written in Python and require any Python of version 3.0 or above to be installed.

`BioPython`: A Python utilities library for computational molecular biology. In this pipelinem it is used for handling biological data, such as reading in and processing protein FASTA files.

`pyfaidx:` A Python library for random access for FASTA files.

`requests`: A Python library for making HTTP requests. It is used in teh pipeline for retrieving protein sequences from UniProt.

`matplotlib:` A Python library for creating visualizations.

`Seaborn`: A Python library used in this pipeline for statistical data visualization, specifically to create a publication quality heatmap. This is a `matplotlib` wrapper.

`numpy:` A Python library for handling numerical data in martix and array structures.

`PyMOL`: A molecular visualization system that is used to visualzing proteins, aligning protein structures, and generating images of overalapped structural alignments with labeled variant proteins.

`ESM-Fold:` A machine learning-based protein structure pridction tool by facebook research that is used for predicitng protein structures.

`PDB2PQR`: Prepares predicted structures to produce a PQR file that can be used for electrostatics calculations.

`APBS`: The Adaptive Poisson-Boltzmann Solver. It is used in the pipeline for calculating hydrophobicity and electrostatic potentials of the predicted protein structures.

`LightDock`: A protein-protein docking tool. It is used in the pipeline for simulating docking between protein of interest and other protein(s).

II.1. conda environment setup

First, install the necessary files with

```
git clone https://github.com/jonazhu/protpredpipeline.git
```

Do this both on Bridges and on a local machine with `pymol` installed, and go into this directory with `cd protpredpipeline`.

Activate module AI in a GPU node using `module load AI`.

Run `install_esm.sh` bash script to install ESMFold and its dependencies.

```
bash install_esm.sh
```

——————————————— **end of setup for Bridges** ———————————————

Run `local_setup.sh` bash script to set up local environment. **Note: This will only work on MacOS architecture.**

```
source local_setup.sh
```

———————————————— **end of setup for local** ————————————————

Part III.
Usage

Run PROTPRED_BRIDGES.SH bash script to start the pipeline. This script calls `step1_final.py` which takes in a UniProt ID or a FASTA sequence file with a list of mutations to be inserted. This script then calls `step2_esm.sh` which puts protein FASTA files into ESMFold and outputs .pdb files of results.

Move ESMFold output folder onto local computer and setup local environment as specified above. Run PROTPRED_LOCAL.SH bash script to start using the second part of the pipeline. This script uses APBS for biochemical property calculation, PyMOL for structural alignment and distance matrix calculations, and LightDock for docking analysis. All analysis outputs are in a folder called `local_analysis`.

III.1. Available commands

III.1.1. Pipeline Command Format

III.1.1.1. Command formats for protpred_bridges.sh pipeline:

a.

```
bash protpred_bridges.sh ids list.txt . output
```

b.

```
bash protpred_bridges.sh onechange muts.txt fasta output
```

c.

```
bash protpred_bridges.sh multichange muts.txt fasta output
```

III.1.1.2. Command format for protpred_local.sh pipeline:

```
bash protpred_local.sh -f output_dba -d ligand.pdb
```

III.1.2. Describing pipeline arguments and values

III.1.2.1. Arguments and values for protpred_bridges.sh :

The protpred_bridges.sh script takes four arguments:

```
protpred_bridges.sh <mode> <input_file> <fasta_file> <output_directory>
```

`⟨mode⟩` (`"ids"`|`"onechange"`|`"multichange"`)

Specifies the mode of operation. See below for more information

`⟨input_file⟩`

Path to the input file, which can be either a list of UniProt IDs or a list of mutations, depending on the selected mode.

`⟨fasta_file⟩`

Path to the FASTA file containing the protein sequence(s). This argument is required for the onechange and multichange modes, but should be specified as . (empty argument) for the ids mode.

`⟨output_directory⟩`

Name of the directory where the ESMFold output files will be stored. Note that the name of this folder MUST be one that does not exist at the current location.

The three modes arguments and values are listed below:

- UniProt IDs (ids mode)
  1. arg[1]: `string` `("ids")`
  2. arg[2]: a .txt file containing a list of UniProt IDs `("list.txt")`
  3. arg[3]: an empty argument `(".")`
  4. arg[4]: the directory name to store ESM output files `("output")`
- Single Mutations (onechange mode)
  1. arg[1]: `string` `("onechange")`
  2. arg[2]: a .txt file containing the mutations to implement in the format "X Y" on each line, where X is the position and Y is the new residue. Ex: 117 F changes position 117 to Phenylalanine `("muts.txt")`
  3. arg[3]: the FASTA file of the sequence to change `("example.fasta")`
  4. arg[4]: the directory name to store ESM output files `("output")`
- Multiple Mutations (multichange mode)
  1. arg[1]: `string` `("multichange")`
  2. arg[2]: a .txt file containing the mutations to implement in the format "X Y" on each line, where X is the position and Y is the new residue. Ex: 117 F changes position 117 to Phenylalanine `("muts.txt")`
  3. arg[3]: the FASTA file of the sequence to change `("example.fasta")`

4. arg[4]: the directory name to store ESM output files `("output")`

Below is an example of the `muts.txt` file format:

15 F
127 E

III.1.2.2. Arguments and values for protpred_local.sh :

The PROTPRED_LOCAL.SH takes two arguements, `-f` and `-d` flags must be provided:

```
bash protpred_local.sh -f <protein_structures_directory>  -d <docking_ligand_pdb>
```

`-f ⟨protein_structures_directory⟩`

Path to directory containing predicted file structures, such as `output_dba`

`-d ⟨docking_ligand_pdb⟩`

.pdb file of ligand for docking (optional argument)

III.1.3. Describing pipeline commands

III.1.3.1. Commands for protpred_bridges.sh pipeline:

Pipeline Overview

The script starts by activating the conda environment where ESMFold is installed. It then runs the `step1_final.py` script, which processes the input file based on the selected mode:

For `ids` mode, it retrieves the protein sequences corresponding to the provided UniProt IDs. For `onechange` and `multichange` modes, it introduces the specified mutations into the protein sequence(s) from the FASTA file.

The processed protein sequences are saved in the `homomer.fasta` file. The script creates the output directory specified by `<output_directory>`. It copies the input file (`<input_file>`) to the output directory. The script changes the current directory to the output directory. It runs the `step2_esm.sh` script, which executes ESMFold on the `homomer.fasta` file. After ESMFold finishes running, the script changes back to the original directory. The generated PDB files can be found in the specified output directory and can be visualized using PyMol on a local machine.

Note:
a. This command is used when the user has UniProt IDs of the protein sequences to be retrieved from the UniProt database. This writes the spcified protein sequences to a .FASTA file.

b. This command is used when the user has a .FASTA file for the protein of interest. This writes a separate file for each change in `list.txt`, a new sequence per mutation.

c. This command is used when the user has a .FASTA file for the protein of interest. This writes one file for all changes in `list.txt`, a new sequence of each mutation specified.

III.1.3.2. Command for protpred_local.sh :

This command takes in predicted protein files and performs RMSD analysis, APBS calculations, docking analysis, creates an RMSD heatmap, and colors the predicted structure by hydrophobicity. Note that before running you must have activated the conda environment protpredlocal set up by the `local_setup.sh` bash script.

III.1.4. Individual commands

Depending on your needs, you can also utilize the steps of the ProtPred Pipeline individually.

III.1.4.1. Step 1: Getting sequences

This step will retrieve the protein sequences you wish to analyze and output them in a single FASTA file, "homomer.fasta". To run this step, use

```
python step1_final.py ids list.txt
```

for cases where you have the list of UniProt IDs;

```
python step1_final.py onechange list.txt fasta
```

for cases where you wish to implement a individual, separate amino acid changes;

```
python step1_final.py multichange list.txt fasta
```

for cases where you wish to implement many changes onto a new mutant protein. The arguments taken for these commands are the exact same as those listed in section III.1.1; please refer to that section for more details.

In summary, `step1_final.py` serves as the main entry point for the protein sequence manipulation and analysis pipeline. It provides three options for obtaining protein sequences and introducing mutations.

Options:
- `ids`: Reads a list of protein IDs from a file, retrieves the corresponding protein sequences, and writes them to a FASTA file.
- `onechange`: Introduces a single mutation to a protein sequence provided in a FASTA file.
- `multichange`: Introduces multiple mutations to a protein sequence provided in a FASTA file.

III.1.4.2. Step 2: Folding

The `step2_esm.sh` bash script runs the ESM-fold model on a FASTA file containing protein sequences.

Folding with ESMFold is done with the command

```
bash step2_esm.sh homomer.fasta
```

where homomer.fasta is the FASTA file given from the previous step, but feel free to replace it with a protein FASTA file of your own depending on your needs.

Importantly, this script will create all output files in the current directory! In order to get a separate folder for these outputs, use the following commands in order:

```
mkdir <output_dir>
cd <output_dir>
bash ../step2_esm.sh ../homomer.fasta
```

where `<output_dir>` corresponds to the name of the directory you wish to specify.

Various later steps in the local portion of this pipeline also require a list of the mutations specified in Step 1 (mainly for visualization purposes). This file can be easily copied and moved to the new folder with a tool like FileZilla or with the `cp` command, such as in the form

```
cp <source_file> <destination>
```

where `<source_file>` is the full file path to muts.txt, and `<destination>` is the full file path to the new directory.

III.1.4.3. Step 3: Alignment

The pipeline from this point on is heavily dependent on directory structure. To ensure that these commands work in the right way, run the protpred_local.sh and not these python scripts individually. Alternatively, these scripts depend on a folder called local_analysis to be created in the same directory. Please ensure that this folder is present.

The `step_3_align.py` python script aligns multiple PDB files and calculates the RMSD (Root Mean Square Deviation) values between them.

Done on a local machine with `PyMOL` installed, this aligns the resulting sequences are aligned using the command

```
python3 step_3_align.py directory
```

This script takes in a directory of pdb files and writes a distance matrix and moves it into the local_analysis folder.

Functions:

`align(pdb_files)`: Aligns the given PDB files and returns a distance matrix of RMSD values.
`write_dist_mat(dist_mat, pdb_files)`: Writes the distance matrix and the order of PDB files to separate files.

III.1.4.4. Step 4: Docking

The `step_4_docking.py` python script performs protein-prtein docking using the LightDock tool.

```
python3 step_4_docking.py receptor ligand
```

The script sets up the docking environment, runs LightDock, generates conformations, clusters the conformations, and visualizes the best docking pose using PyMol and creates an image of the docking.

III.1.4.5. Step 5: Visualizing Heatmap

The `step_5_heatmap.py` python script generates a heatmap of the RMSD values between protein structures given a distance matrix generated from step 3.

```
python step_5_heatmap.py distance_matrix.txt
```

where distance_matrix.txt is a file containing the RMSD values from the structural alignment of the proteins.

The script reads a square matrix of RMSD values from a file, along with the corresponding PDB file names, and generates a heatmap using the seaborn library.

III.1.4.6. Step 6: Visualizing Hydrophobicity

The `step_6_props.py` python script colors proteins structures based on amino acid properties using PyMol.

```
python step_6_props.py pdb_file
```

The script loads PDB files, sets custom colors for each amino acid type and applies the colors to the structure. It then saves an image of the colored structures in the local_analysis folder..

3.1.4.6 Available commands

III.1.5. Limitations

The second part of this pipeline (PROTPRED_LOCAL.SH) does need to be run locally, and it requires that the user have PyMOL installed. A future improvement could be to use an alternative software that can also be run on Bridges.

Part IV. Index