SH3-Peptide Complex Structure Modeling - User Manual

Mor Vazana, Magal Gafni, Hilla Asida and Gabriella Levy

Data processing:

1. Run the python script **alignSH.py** -
    - The user must enter the directory of the PDB files as input of the align function.
    - Output: a txt file that contains three columns: the PDB name, the chain of SH3 and the transformation found when aligning this chain.
    - Description - this script identifies the SH3 domain in each structure and aligns it with the reference structure. It also filters out the unsuitable structures according to fixed thresholds that determine which alignments were considered somewhat successful.

2. Run the python script **transform.py** -
    - The user must change the value of RESULTS to the directory of the output of the alignSH3_new.py script.
    - Output: the PDB files with the given transformation from the input file.
    - Description - this script transforms the whole PDB according to the transformations found in the previous step.

3. Run the c++ script **extractClosestProtein.cpp** -
    - Input: a reference PDB with one chain, a txt file that has two columns- one with the pdb name and the second with the chain ID of the SH3 domain (separated by whitespace) and a string = “peptide”/”domain”.
    - Output: a csv file that contains 4 columns: pdb name, the chain ID of the protein that has been found to be the closest to the reference, the start residue ID of that protein and the end residue ID of that protein.
    - Description - this script calculates the distance of all chains of each pdb in the directory “transformedPDBs” to the reference, and finds the closest chain and its start and end residue ID.

4. Run **Create_pdb.py** -

    - Input: the output csv from extractClosestProtein.cpp for peptide, the output csv from extractClosestProtein.cpp for domain(SH3).
    - Output: a PDB file for each domain and peptide in directory “ready_pdbs”.
    - Description - this script creates a new PDB file for each domain and peptide that were found in the same original PDB according to the residues in the input files. Domain chain is ‘A’ and the peptide chain is ‘B’.

5. Run **SH3_MODELING_utils.ipynb** -
    - Input: a path to a directory with PDBs.
    - Output: a matrix for the network input- one hot encoding of the domain sequence and the peptide sequence (same matrix with padding between), a matrix for network labels - each three consecutive columns represent x,y,z coordinates of each atom [N, Cα, C, O, C] in that order.
    - Description - this script preprocess of the data to feed the network.

Training neural network:

- **SH3_MODELING_NET.ipynb** - a neural network that predicts the SH3-peptide complex structure. Its Input and labels are the output from the previous step.

Evaluating performance:

1. **RMSDbyChain.py** -
    - Input: a path to the directory with the test PDBs (containing two empty directory - testTransformed and test_sh3), the name of the predictor, a path to the directory of the predicted PDBs (containing two empty directories - SH3 and peptide), the chain of the SH3 domain and the chain of the peptide domain in the predicted PDBs.
    - Output: two text files containing the RMSD of the SH3 domain and the peptide for each PDB file in the test set.
    - Description - this script aligns the SH3 domain in the reference structure to the SH3 domain of the predicted structure. The output

transformations of the alignment is used to transform the total reference structure. Finally, this script calculates the RMSD between the transformed reference structure and the predicted structure, for both the SH3 domain and peptide.

2. **boxplot_hackathon.R** - the user must enter the results of RMSDbyChain.py to the variable df_alpha and df_model.

3. **scatterplot_hackathon.R** - the user must enter the results of RMSDbyChain.py to the variable df_alpha and df_model.