Manual for iNNterfaceDesign

##### First time setup

The following software and python packages have to be installed in order to run iNNterfaceDesign:

1. python v3.7;
2. PyRosetta-4 2019;
3. Tensorflow v2.1.0;
4. h5py v.2.10.0.
5. NumPy v.1.19.1.

Setup consists in a mere downloading of a folder “iNNterfaceDesign\_scripts” and unzipping files in iNNterfaceDesign\_scripts /modules/ frag\_database /. Besides, neural network models are stored in “models” folder, they should be placed into “iNNterfaceDesign\_scripts /modules/models/” directory.

PepBB model is split into zip files; the files should be unzipped first; the result should be a single PepBB.hdf5.

You can change paths to directories with models in 2.binders.py, line 7 and 4.amn\_sampling.py, line 8:  
dir\_models = 'modules/models/'.

The same applies to line

frag\_dir = 'modules/frag\_database/' in modules/transform\_coords.py, line 9

##### PepBB input

PepBB input consists of a series of lines in an ASCII text file. The lines specify names of job files, desired features of the binders and other options.

Here is an example of such a file:

pdb: 3ztj\_id

anchor\_res: 363-365,367

The file specifies PDB file with a protein receptor and a list of anchor residues for which the binders are to be designed. This simplest input will generate single most probable backbone pose for each anchor residue with a single amino acid sequence. Input file “example\_input.txt” is attached. Pdb keyword value should not contain name of folder. User can use the following input as an example for generating more outputs:

pdb: 3ztj\_id

anchor\_res: 363-365,367

max\_pos: 36

max\_sst: 9

amn\_design: 6

Full list of keywords is presented in Table 1.

Table 1. PepBB keywords

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Keyword** | **Description** | **Value** | **Default** | **Example** | **Remarks** |
| pdb | Protein receptor for modeling of binders. The structure must be prepared as described below. | Name of the file without “.pdb” extension |  | pdb: 3ztj\_id |  |
| prefix | Prefix for names of output and scratch files. | Single word string | “pdb” keyword | prefix: 3ztj |  |
| anchor\_res | Set the anchor residue, one or a few. | Integer corresponding to the number of the residue.  If there are multiple anchor residues, they should be separated by comma, ranges of the residues can be represented using dashes. | All surface residues of the protein receptor. | anchor\_res: 363-365,367 |  |
| chain | Set chain for selecting of anchor residues. All surface residues of the chain will be utilized. | Letter corresponding to a label of the chain. If there are multiple chains, the labels should be separated by comma. |  | chain: A,B | Optional.  The keyword is overlooked if “anchor\_res” keyword is provided. |
| get\_surf\_res | Set method for determining of surface residues of the protein receptor. We have written a script for the task providing the most relevant results during the design. However, due to its slowness, we implemented an option to perform the calculations using pyrosetta functions as well. | “pyrosetta” or  “PepBB” | PepBB | get\_surf\_res: pyrosetta |  |
| interf\_type | Set type of protein-protein interface to design: hetero-oligomeric (0) or homo-oligomeric (1) | 0 or 1 | 0 | interf\_type: 1 |  |
| interf\_res | Set a list of interface residues of the protein receptor. | Same as in a case of “anchor\_res” keyword |  | interf\_res: 360-370,375 | Optional |
| swap\_pose | Swapping of generated 6-residue backbone with RMSD-wise similar native 6-residue backbone | False/True | False | swap\_pose: True | Time consuming, especially in case of loop fragments. If RMSD between generated and native fragments equal or less than 0.5 Å, the replacement will not happen. |
| pos\_res1 | Set a residue of the protein receptor which is desired to be located near N-terminus of the designed binders. | Integer corresponding to the number of the residue. |  | pos\_res1: 361 | Optional |
| pos\_res2 | Set a residue of the protein receptor which is desired to be located near C-terminus of the designed binders. | Integer corresponding to the number of the residue. |  | pos\_res2: 381 | Optional |
| pos\_res1d | Set a residue of the protein receptor direction to which is undesired for N-terminus of the designed binders. | Integer corresponding to the number of the residue. |  | pos\_res1: 361 | Optional |
| pos\_res2d | Set a residue of the protein receptor direction to which is undesired for C-terminus of the designed binders. | Integer corresponding to the number of the residue. |  | pos\_res2: 381 | Optional |
| max\_pos | Set number of positions to utilize. Positions are ranked according to probability distribution. Most probable positions are selected | Integer between 1 and 36. | 1 | max\_pos: 3 |  |
| max\_sst | Set number of secondary structure sequences (SSS) for which the binders are designed. SSS are ranked according to probability distribution. Most probable of them are selected. | Integer between 1 and 9. | 1 | max\_pos: 3 |  |
| pos\_thr | Set threshold probability for positions selection. | Float between 0 and 1 | 0.01 | pos\_thr: 0.05 | Operation is applied before selecting positions according to keywords “max\_pos”, “pos\_res1” and “pos\_res2”. If the threshold is too high, there are possibility that none positions are selected and the binders are not generated. |
| sst\_thr | Set threshold probability for secondary structure sequences (SSS) selection. | Float between 0 and 1 | 0.01 | sst\_thr: 0.05 | Operation is applied before selecting SSS according to keyword “sst\_type”. If the threshold is too high, there are possibility that none SSS are selected and the binders are not generated. |
| sst\_type | Set secondary structure types which must be presented in secondary structure sequences. | L (loop)  E (b-sheet)  H (a-helix)  If there are multiple types, they should be separated by comma |  | sst\_type: H,L | Optional |
| add\_residues | Align the generated binder with another specified binder to make the latter longer by 3 residues at each (default) or only one terminus. | False/True | False | add\_residues: True | The names of binders to prolongate are read from “prefix +\_names.json” file. Otherwise, the binders can be listed in a text file specified through “binders\_list” command. |
| num\_pepbbe\_m | This keyword determines the number of trained PepBBE models to use for elongation of the binder. | 1/2/3 | 1 | num\_pepbbe\_m: 3 | We have kept three trained PepBBE models with similar accuracy, but slightly different output poses after experiments. The best models were set as default, however there is opportunity to extend backbones using all these models getting 3 times more poses: in most cases they are not exactly same and corresponding amino acid sequence designs for them can differ |
| binders\_list | Set a name of a text file providing a list of binders to prolongate if command “add\_residues” is active. | Name of a file |  | binders\_list: binders.txt | Binder names in the file should be separated by comma |
| binder\_end | Set a terminus of binders to be prolongated if command “add\_residues” is active. | N/C | Both termini | binder\_end: C | Optional |
| amn\_design | Set method for design of amino acid sequences for the designed binders | 1: PepSep1  6: PepSep6 | 1 | amn\_design: 6 |  |
| amn\_prob\_distr | Print outputs of PepSep1 after applying of “Softmax” activation function to get categorical probability distribution over amino acid types for each position. | False/True | False | amn\_prob\_distr: True |  |

##### Preparing of input PDB files for PepBB

The files must be cleaned from all information except the ATOM records. Besides, the files have to be renumbered such that the first residue gets number 1 and all subsequent residues are numbered consecutively. We recommend using clean\_pdb.py script which can be found in the Rosetta tools repository under or in

https://github.com/bestlab/GREMLIN\_RF/tree/master/preprocessing/rosetta\_scripts.

##### Running PepBB

The generation of the binders takes four steps each of which is carried out by different scripts and the same input file.

# 1. Construction of binding sites centered at anchor residues and extracting features of the binding sites.

*Command:* python 1.preprocessing.py input

The command creates a folder named according to “prefix” keyword and folders “binders” and “pockets” in it. Besides, scratch files “prefix + \_b.json” and “prefix + \_2b.json” , containing extracted features of the binding sites, have to be created within the folder. Patches of protein receptor surface selected as binding sites are stored in the “pockets” folder.

# 2. Generating of backbones of the binders.

*Command:* python 2.binders.py input

The command generates backbones of the binders into the “binders” folder. A file “prefix +\_names.json” with names of the generated PDB files have to be created. Names of the files consists of four parts separated by underline: prefix, anchor residue and subsequent ranks of used position and secondary structure sequence according to the positioning and the secondary structure predicting models.

We recommend inspecting the generated binders at this point: the binders can miss one or few heavy atoms in some rare cases and such atoms must be recovered by side software on that occasion, otherwise the incomplete backbones will be ignored at step 3.

# 3. Idealization of the binders and extracting features for amino acid sequences design.

*Command:* python 3.preprocessing\_seq.py input

Folders “binders\_id” and “complexes” are created in the main folder of the job. The binders are idealized by means of IdealizeMover of pyrosetta package and stored in the “binders\_id” folder. Complexes are generated by attaching of the binders to the binding sites, the resulted structures are stored in the “complexes” folder. A scratch file named “prefix + \_for\_seqb.json” with the extracted features has to be generated as well.

# 4. Prediction of amino acid sequences.

*Command:* python 4.amn\_sampling.py input

A file “prefix + \_seqb.json” containing generated amino acid sequences for each binder is created. If “amn\_prob\_distr” keyword is set to True, the probability distribution for each position is presented additionally.

Example with probability distributions, which are represented as 6 dictionaries, containing probability for each type of amino acid:

[["3ztj\_id\_B363\_0\_0.pdb", "LIYENI", [{"C": 0.008, "H": 0.01, "W": 0.011, "M": 0.019, "I": 0.019, "V": 0.026, "K": 0.028, "R": 0.029, "T": 0.03, "G": 0.033, "N": 0.035, "P": 0.041, "D": 0.044, "A": 0.051, "Q": 0.056, "Y": 0.059, "F": 0.063, "E": 0.069, "S": 0.083, "L": 0.095}, {"P": 0.0, "C": 0.006, "G": 0.007, "M": 0.011, "D": 0.012, "N": 0.014, "W": 0.015, "H": 0.015, "A": 0.019, "F": 0.03, "S": 0.035, "Q": 0.037, "T": 0.039, "Y": 0.042, "K": 0.051, "L": 0.062, "R": 0.072, "V": 0.073, "E": 0.086, "I": 0.089}, {"P": 0.003, "C": 0.004, "G": 0.009, "E": 0.01, "S": 0.017, "H": 0.02, "Q": 0.021, "T": 0.023, "D": 0.023, "N": 0.028, "A": 0.031, "W": 0.037, "F": 0.042, "M": 0.045, "V": 0.054, "I": 0.061, "K": 0.077, "R": 0.096, "L": 0.105, "Y": 0.106}, {"P": 0.0, "W": 0.001, "V": 0.001, "I": 0.002, "Y": 0.003, "F": 0.003, "T": 0.011, "H": 0.013, "C": 0.018, "G": 0.018, "D": 0.027, "S": 0.03, "N": 0.033, "A": 0.041, "R": 0.06, "M": 0.075, "L": 0.095, "K": 0.114, "Q": 0.118, "E": 0.206}, {"P": 0.0, "G": 0.006, "M": 0.008, "Q": 0.014, "T": 0.015, "I": 0.022, "E": 0.022, "V": 0.024, "W": 0.026, "A": 0.028, "C": 0.028, "R": 0.032, "S": 0.033, "Y": 0.034, "K": 0.035, "L": 0.039, "D": 0.066, "F": 0.073, "H": 0.107, "N": 0.198}, {"G": 0.001, "E": 0.001, "D": 0.002, "A": 0.004, "P": 0.004, "Q": 0.007, "S": 0.008, "C": 0.008, "W": 0.009, "M": 0.012, "H": 0.012, "K": 0.02, "T": 0.049, "F": 0.051, "R": 0.066, "V": 0.079, "N": 0.097, "Y": 0.108, "L": 0.116, "I": 0.256}]]

All steps can be executed through a single bash script, such as:

#!/bin/bash

dir=Dir\_of\_method\_files

inp1=$1

python "${dir}"1.preprocessing.py $inp1

python "${dir}"2.binders.py $inp1

python "${dir}"3.preprocessing\_seq.py $inp1

inp2=$2

python "${dir}"1.preprocessing.py $inp2

python "${dir}"2.binders.py $inp2

python "${dir}"3.preprocessing\_seq.py $inp2

python "${dir}"4.amn\_sampling.py $inp2

where, inp1 and inp2 are input files for generation of 6-residue poses and for extension of them, correspondingly.

Manual for PepSep methods

##### First time setup

Setup is the same as for iNNterfaceDesign.

##### PepSep input

PepSep input consists of a series of lines in an ASCII text file. The lines specify names of job files, desired features of the binders and other options.

Here is an example of such a file:

pdb: 3ztj\_id

res1: 363

The file specifies PDB file with a complex and the first residue of 6-mer binder or fragment of protein ligand which amino acid sequences are to be recovered. Full list of keywords is presented in Table 1.

Table 1. PepSep keywords

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Keyword** | **Description** | **Value** | **Default** | **Example** | **Remarks** |
| pdb | Protein receptor for modeling of binders. The structure must be prepared as described below. | Name of the file without “.pdb” extension |  | pdb: 3ztj\_id |  |
| prefix | Prefix for names of output and scratch files. | Single word string | “pdb” keyword | prefix: 3ztj |  |
| res1 | Set first residue of 6-mer binder or a 6-mer fragment of protein ligand which amino acid sequences are to be recovered | Integer corresponding to the number of the residue. If there are multiple binders/fragments to process, the first residues should be separated by comma, ranges of the residues can be represented using dashes. |  | res1: 361,365-370 |  |
| res\_interval | Set interval of residues of a binder or a fragment of protein ligand which amino acid sequences are to be recovered | Integers corresponding to first and last residues of the interval. If there are multiple binders/fragments to process, the intervals should be separated by comma. |  | res\_interval: 100-110,361-370 |  |
| interf\_type | Set type of protein-protein interface to design: hetero-oligomeric (0) or homo-oligomeric (1) | 0 or 1 | 0 | interf\_type: 1 |  |
| p\_chains | Set chains corresponding to binding site. | Letter corresponding to a label of the chain. If there are multiple chains, the labels should be separated by comma. | All chains except a chain containing res1 | chain: A,B |  |
| amn\_design | Set method for design of amino acid sequences for the designed binders | 1: PepSep1  6: PepSep6 | 1 | amn\_design: 6 |  |
| amn\_prob\_distr | Print outputs of PepSep1 after applying of “Softmax” activation function to get categorical probability distribution over amino acid types for each position. | False/True | False | amn\_prob\_distr: True |  |
| use\_prot\_lig | Take into account residues of the protein ligand with known amino acid identities, which are close to target fragment, if provided. | False/True | False | use\_prot\_lig: True | Accuracy obtained with the option is not studied. Residues of protein receptor were considered as binding site residues in our experiments only. |
| idealize | Idealize all-glycine binder | False/True | True | idealize: False | We idealized all-glycine binders from our datasets by means of IdealizeMover of pyrosetta. We suppose that this step can be skipped in case of native complexes, but we do not know how it would affect the accuracy. |
| filter\_designs | Removing designs with redundant number of the same amino acid identity | False/True | False | filter\_designs: True | Models predict redundant number (3 or more per one 6-residue fragment) of the same amino acid identity, usually Tyr and Asp, on highly disturbed peptide ligands (RMSD ≈ 4 Å) in some cases. One might want to delete such designs right away. |

##### Preparing of input PDB files for PepSep

The files must be cleaned from all information except the ATOM records. Besides, the files have to be renumbered such that the first residue gets number 1 and all subsequent residues are numbered consecutively. We recommend using clean\_pdb.py script which can be found in the Rosetta tools repository under.

##### Running PepSep

The generation of the binders takes two steps each of which is carried out by different scripts and the same input file.

# 1. Construction of binding sites centered at anchor residues and extracting features of the binding sites.

*Command:* python 1.preprocessing\_sp.py input

The command creates a job folder named according to “prefix” keyword and folder “structures” in it. Patches of protein-protein interfaces containing the binder and the binding site are derived from the initial complex and stored in the “structures” folder. Names of the files consists of three parts separated by underline: prefix, the first residue and ending “\_cpx”. The features of the binding site and preprocessed binder backbone are stored in a file named “prefix + \_data\_aas.json” in the job folder. The preprocessing of the binder consists of mutation of all residues to glycine and subsequent idealization by means of IdealizeMover of pyrosetta (the last operation can be skipped using “idealize” keyword).

# 2. Prediction of amino acid sequences.

*Command:* python 4.amn\_sampling.py input

A file “prefix + \_aas.json” containing generated amino acid sequences for each binder is created. If “amn\_prob\_distr” keyword is set to True, the probability distribution for each position is presented additionally.

Example with probability distributions, which are represented as 6 dictionaries, containing probability for each type of amino acid:

[["3ztj\_id\_B363\_0\_0.pdb", "LIYENI", [{"C": 0.008, "H": 0.01, "W": 0.011, "M": 0.019, "I": 0.019, "V": 0.026, "K": 0.028, "R": 0.029, "T": 0.03, "G": 0.033, "N": 0.035, "P": 0.041, "D": 0.044, "A": 0.051, "Q": 0.056, "Y": 0.059, "F": 0.063, "E": 0.069, "S": 0.083, "L": 0.095}, {"P": 0.0, "C": 0.006, "G": 0.007, "M": 0.011, "D": 0.012, "N": 0.014, "W": 0.015, "H": 0.015, "A": 0.019, "F": 0.03, "S": 0.035, "Q": 0.037, "T": 0.039, "Y": 0.042, "K": 0.051, "L": 0.062, "R": 0.072, "V": 0.073, "E": 0.086, "I": 0.089}, {"P": 0.003, "C": 0.004, "G": 0.009, "E": 0.01, "S": 0.017, "H": 0.02, "Q": 0.021, "T": 0.023, "D": 0.023, "N": 0.028, "A": 0.031, "W": 0.037, "F": 0.042, "M": 0.045, "V": 0.054, "I": 0.061, "K": 0.077, "R": 0.096, "L": 0.105, "Y": 0.106}, {"P": 0.0, "W": 0.001, "V": 0.001, "I": 0.002, "Y": 0.003, "F": 0.003, "T": 0.011, "H": 0.013, "C": 0.018, "G": 0.018, "D": 0.027, "S": 0.03, "N": 0.033, "A": 0.041, "R": 0.06, "M": 0.075, "L": 0.095, "K": 0.114, "Q": 0.118, "E": 0.206}, {"P": 0.0, "G": 0.006, "M": 0.008, "Q": 0.014, "T": 0.015, "I": 0.022, "E": 0.022, "V": 0.024, "W": 0.026, "A": 0.028, "C": 0.028, "R": 0.032, "S": 0.033, "Y": 0.034, "K": 0.035, "L": 0.039, "D": 0.066, "F": 0.073, "H": 0.107, "N": 0.198}, {"G": 0.001, "E": 0.001, "D": 0.002, "A": 0.004, "P": 0.004, "Q": 0.007, "S": 0.008, "C": 0.008, "W": 0.009, "M": 0.012, "H": 0.012, "K": 0.02, "T": 0.049, "F": 0.051, "R": 0.066, "V": 0.079, "N": 0.097, "Y": 0.108, "L": 0.116, "I": 0.256}]]