Manual for iNNterfaceDesign

Read iNNterfaceDesign papers first for understanding of a workflow and terms, please. Manual does not contain explanation of methods.

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 or higher;
3. Tensorflow v2.1.0;
4. h5py v.2.10.0.
5. NumPy v.1.19.1.

Installation instructions

1. Install all required python packages. Pay attention to the versions of the packages. The requirements are **not the latest** releases.

   Approximate commands to install python packages:

   ```
   wget https://repo.anaconda.com/miniconda/Miniconda3-py37_4.11.0-Linux-x86_64.sh
   bash Miniconda3-py37_4.11.0-Linux-x86_64.sh
   ```

   source your .bashrc or better log out/ log in to computer

   ```
   conda install -c conda-forge tensorflow=2.1
   conda install numpy=1.19.1
   ```

   maybe pandas!!

   You can consider installation of cuda packages additionally if your computer has GPU.

2. Installation of PyRosetta

   Generally follow the PyRosetta installation instruction. Here is how we did it:

   You can get a precompiled pyrosetta version for 3.7. for instance, we used: PyRosetta4.Release.python37.ubuntu.release-315.tar.bz2

   Go into the setup folder
   Then execute:

   ```
   python setup.py install
   ```

3. Download "iNNterfaceDesign " project from GitHub.

   ```
   conda install -c anaconda git
   ```

```
git clone https://github.com/strauchlab/iNNterfaceDesign.git
```
Move content of “iNNterfaceDesign_scripts“ folder into main “iNNterfaceDesign” folder.

4. Unzip archives in iNNterfaceDesign/modules/frag_database/.
5. Create folder “iNNterfaceDesign/modules/models” and download content of “models“ folder of GitHub into there.
6. PepBB model is split into 3 files, which are PepBB.aa, PepBB.ab and PepBB.ac. The files must be combined first by command:

   ```
   cat PepBB.a* >> PepBB.hdf5
   ```

   The result should be a single PepBB.hdf5. The same is applied to PepBBE models (PepBB_N1, PepBB_N2, PepBB_N3, PepBB_C1, PepBB_C2, PepBB_C3):

   ```
   for i in {1..3} ; do cat PepBB_N$i.a* >>  PepBB_N$i.hdf5 ; done
   for i in {1..3} ; do cat PepBB_C$i.a* >>  PepBB_C$i.hdf5 ; done
   ```

7. Add “iNNterfaceDesign” directory to $PATH. Like so add this to the .bashrc:

   ```
   PATH="/home/user/bin:/home/user/iNNterfaceDesign:$PATH"
   ```

8. **if you arranged the folders differently**: The location of files must be modified in files “iNNterfaceDesign/ 2.binders.py” and “iNNterfaceDesign/4.amn_sampling.py”, function “get_path()”,

iNNterfaceDesign inputs

iNNterfaceDesign requires a few input files usually: a file to generate core 6-residue peptide ligand backbone using PepBB method and one or a few inputs elongating the resulted backbones using PepBBE. PepSeP methods does not require separate inputs. Inputs consist of a series of lines in an ASCII text file specifying names of job files, desired features of the binders and other options in a format *option: value*. Lines with comments start with the hash character, #. Empty lines are allowed but not wanted (they can cause errors in some cases).

Input 1

1st file contains options for generating core 6-residue peptide ligands using PepBB. It is sufficient if you do not want to make backbones longer. Application of PepSeP methods generating amino acid sequences for the results of PepBB can be carried out using that single file.

Here is an example of a such file:

```
pdb: 3ztj_ABCDEF
anchor_res: 363-365,367
```

The file specifies PDB file with a protein receptor (3ztj_ABCDEF.pdb) and a list of anchor residues (363-365,367) for which binders are to be designed. This simplest input generates four 6-residue binders: a top backbone pose for each anchor residue with a single amino acid sequence. The input file and 3ztj_ABCDEF.pdb can be found in test_inputs/iNNterfaceDesign/.

User can use the input below as an example for maximizing number of designed poses per anchor residue (probability thresholds for selection of positions and secondary structures can be reduced further but it is not recommended):

```
pdb: 3ztj_ABCDEF
anchor_res: 363-365,367
max_pos: 36
max_sst: 9
amn_design: 6
```

Full list of keywords is presented in Table 1.

Input 2

2nd and further files contain options for elongating the core 6-residue peptide ligands using PepBBE. It must repeat “pdb”, “prefix” (if provided) lines of the first file and an additional option “add_residues”:

```
add_residues: True
pdb: 3ztj_ABCDEF
anchor_res: 363-365,367
max_pos: 2
max_sst: 2
amn_design: 6
```

Other lines, like numbers of positions and secondary structures to use for extra-residues, can differ. PepBBE is the weakest link in our pipeline and often results in twisted angles in joint places. Therefore, we

have kept three trained PepBBE models with similar accuracy, but slightly different output poses. The best model is set as default, however there is an opportunity to extend backbones using all these models getting 3 times more poses some of which can be better than others. Usage of all three models require option:

num_pepbbe_m: 3

Input 3 and further
They are required for second and further rounds of elongation if options are not the same as in input 2. For example, if max_pos is 2 in input 2 and the user wants max_pos of 1 in second (or further) elongation stages. If you do not want to make binders longer after first elongation or want to keep options of input 2 in other rounds of elongation, input3 and etc. are not necessary.

Table 1. **iNNterfaceDesign** 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 "innf_core"	pyrosetta	get_surf_res: innf_core	

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 (β-sheet) H (α-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
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 iNNterfaceDesign

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.

iNNterfaceDesign scripts

1.preprocessing.py

The script constructs binding sites centered at anchor residues and extracts features of the binding sites. The operations are carried out using pyrosetta methods.
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 "pockets" folder.

2.binders.py

The script generates backbones of the binders using PepBB or PepBBE neural networks.
Command:

```
python 2.binders.py input
```

The command generates backbones of the binders in "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.
The generated backbones can be inspected at this point: the binders can miss one or a 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 further.

3.preprocessing_seq.py

The script idealizes backbones of the binders generated by PepBB/PepBBE using IdealizeMover and extracts features for amino acid sequences design. The operations are carried out using pyrosetta methods.
Command:

```
python 3.preprocessing_seq.py input
```

Folders "binders_id", "binders_rel" 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 "binders_id" folder. Names of backbones get "_id.pdb" ending. Complexes are generated by attaching of binders to the binding sites, the resulted structures are stored in "complexes" folder. A scratch file named "prefix + _data_aas.json" with extracted features for PepSeP has to be generated as well.

4.amn_sampling.py

The script predicts of amino acid sequences by means of PepSeP1 or PepSeP6 neural networks.
Command:

```
python 4.amn_sampling.py input
```

A file "prefix + _aas.json" containing generated amino acid sequences for each binder has to be created.

5.pdb_ref.py

Command: python 5.pdb_ref.py list_of_backbones_from_binders_id_folder
Command to get list_of_backbones_from_binders_id_folder:

find *folder/binders_id/*pdb* | sed 's|.pdb||' > list,

All generated binders after each stage of elongation are stored together in */binders_id/* with different names (names become longer after each elongation), so user has to specify a fragment of names.

The script mutates glycine residues of backbones according to amino acid sequences predicted by PepSeP methods and refines them using FastRelax mover of pyrosetta. The resulted complexes are stored in the "binders_rel" folder. Names of backbones get "_cpx.pdb" ending. Besides, binding affinities and results of alanine scanning can be found in */scores/* folder.

Running of iNNterfaceDesign

Commands can be executed one by one according to order and commands in "iNNterfaceDesign scripts" section (above) or all steps can be executed through a single bash script, such as:

```
#!/bin/bash

dir=path_to_iNNterfaceDesign_folder
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}"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. The script elongates binders two times, the result would be 18-residue motifs. The results have to be refined using 5.pdb_ref.py after.

Number of 12-residue motifs which is possible to generate for 1 anchor residue

PepBB
Maximum number of positions: 36 *However, some part of binders would be oriented towards a protein surface causing clashes inevitably if too many positions are requested due to features of the method.*
Maximum number of secondary structure sequences: 9

PepBBE
Maximum number of positions per end: 6
Maximum number of secondary structure sequences per end: 9
Maximum number of models per end: 3

Number of backbones: 36*9*2*6*9*3 = 104 976

PepSep6
Maximum number of amino acid sequences: 6
Number of motifs: 104 976*6

Application of PepSeP6 more times on relaxed backbones:
Number of motifs: 104 976*6*6* …

The number of motifs grows accordingly if more than 1 anchor residue are used or more rounds of elongations are performed. Generation and subsequent relaxation of such number of binders can be a very long process. So, choose options with consideration.

It should be mentioned that part of solutions would be rejected by iNNterfaceDesign scripts and most motifs would be rejected by pyrosetta/Rosetta filters after relaxation.

Tips for users with programming skills

1. 5.prd_ref.py (and 5.pdb_ref_sp.py for PepSep) is based on common pyrosetta refining movers. Users can adjust options of these movers (types of movers, number of optimization cycles, constraints, etc.) for their purposes.
2. Secondary prediction of amino acid sequences for refined binders based on results of 5.pdb_ref.py can help to obtain better and more amino acid sequences (as described in "SARS-CoV-2_design_examples" folder). 1.preprocessing_sp.py script introduced for usage of standalone PepSeP processes a single complex per run only, which is not convenient in case of many generated binders. We usually modify 3.preprocessing_seq.py to make it process refined binders instead of binders generated by PepBB. However, we do not have universal script for the purpose and programmers can make such script themselves for their concrete case.
3. Replacement of generated helical binders with superimposed ideal helices before applying PepSeP can be a good idea in case of long backbones.
4. 5.pdb_ref.py is supposed to be applied on complexes which consists of relatively small binding sites and designed motifs for acceleration of jobs. If user choose a motif for grafting for practical applications after, the relaxation should be repeated by replacing binding site with bigger surfaces or initial crystal structures of targets. We usually transfer coordinates of side chains of

relaxed binding sites to corresponding residues of targets first using pyrosetta tools before application FastRelax or other optimization movers.

Implementation of these tips is entirely on users. We do not have opportunity to consult about them due to small number of contributors.

Manual for PepSep methods

PepSeP is a method predicting amino acid sequences for fragments of protein interfaces. Predictions are based on location of backbone atoms of the fragment under question and features of a binding site where the fragment is attached. It was developed as part of iNNterfaceDesign to make amino acid designs for backbones produced by PepBB and PepBBE, as demonstrated in iNNterfaceDesign section. This part of manual demonstrate usage of PepSeP on protein-protein interfaces beyond iNNterfaceDesign pipeline. The method exists in two versions: PepSeP1 and PepSeP6 producing 1 and 6 amino acid sequences correspondingly.

Input crystal structures must be protein-peptide or protein-protein complexes. PDB-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.

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_ABCDEFHG
res1: 1570-1574
```

The file specifies PDB file (3ztj_ABCDEFHG) with a complex and the first residue of 6-mer binder or a fragment (1570-1574) of protein ligand which amino acid sequences are to be recovered. Residues 1570-1574 are located on chain G. Chains ABCDEFH are to be used to construct a binding site. User can specify which chains to use to construct the binding site more specifically:

```
pdb: 3ztj_ABCDEFHG
res1: 1570-1574
p_chains: A,B,C,D,E,F
```

Default method for prediction of amino acid sequence is PepSeP1. PepSeP6 can be used through: amn_design: 6

res1 option creates 6-residue fragments for each of mentioned residues: 1570-1575, 1571-1576, 1572-1577, 1573-1578, 1574-1579, in this case. Amino acid sequences will be predicted for each fragment separately.

Output with PepSeP6:

[[“3ztj_ ABCDEFHG _G1570”, [“YFLLDR”, “FLLDRL”, “LLDRLD”, “LDRLDL”, “DRLDLL”, “RLDLLV”]], [“3ztj_ ABCDEFHG _G1571”, [“YILYWK”, “ILYWKL”, “LYWKLV”, “YWKLVL”, “WKLVLF”, “KLVLFW”]], [“3ztj_ ABCDEFHG _G1572”, [“ILAWKK”, “LAWKKL”, “AWKKLL”, “WKKLLA”, “KKLLAH”, “KLLAHR”]], [“3ztj_ ABCDEFHG _G1573”, [“LLYHPF”, “LYHPFL”, “YHPFLA”, “HPFLAW”, “PFLAWH”, “FLAWHP”]], [“3ztj_ ABCDEFHG _G1574”, [“YGTDYY”, “GTDYYL”, “TDYYLY”, “DYYLYN”, “YYLYND”, “YLYNDD”]]]

User can use “res_interval” instead of “res1” in case of producing amino acid sequence for a long fragment:

```
prefix: 3ztj_pepsep6_long_frag
pdb: 3ztj_ABCDEFHG
res_interval: 1570-1580
p_chains: A,B,C,D,E,F
amn_design: 6
```

Output:

[["3ztj_pepsep6_long_frag_G1570", ["IFILYWKPFYP", "FLLLWKPLYPL", "LLYWKPLYPYN", "LYWKPLLPYND", "DWKPLLPYNDD", "RKDLLPWNPDL"]]]

Full list of keywords is presented in Table 2.

Table 2. 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	Integers corresponding to first and last residues of the interval. If there are multiple binders/fragments to process, the		res_interval: 100-110,361-370	

	acid sequences are to be recovered	intervals should be separated by comma.			
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.

Running PepSep

The generation of the binders takes three 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 "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 Idealize-Mover 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.

5.pdb_ref_sp.py

Command: python 5.pdb_ref_sp.py list_of_fragments_from_structures_folder
Command to get list list_of_fragments_from_structures_folder:

ls *folder/structures/* *binder.* > list

The script mutates residues of backbones according to amino acid sequences predicted by PepSeP methods and refines them using FastRelax mover of pyrosetta. The resulted complexes are stored in the "binders_rel" folder. Names of backbones get "_cpx.pdb" ending.