iCFN Manual

Mostafa Karimi, mostafa_karimi@tamu.edu

October 21, 2018

Contents

1 iCFN Download link and its Prerequisites **3**

2 Data format **4**

2.1 Input data for each protein 4

2.2 Multistate protein design with ensemble of substates 6

2.3 Toy example 6

3 Replication of TCR design **10**

4 iCFN for new design **11**

4.1 Running protein design based on OSPREY energy calculation 12

1 iCFN Download link and its Prerequisites

Binary executable version of iCFN is available in `https://shen-lab.github.io/software/iCFN/`. iCFN has been developed and tested in:

- C programming
- GCC 4.4.7 compiler
- Centos 6.7 Linux

2 Data format

2.1 Input data for each protein

For any protein, iCFN needs four files:

- Rot_positions: There should be a text file with 4 columns which explains about the design of this protein with each correspond to a mutable or flexible residue in the design:
  - First column: chain of the residue
  - Second column: residue number
  - Third column: Wild type amino acid of residue with three letter codes given in the below table.
  - Fourth column: number of rotamers for the residue
- Constant.dat: There should be a binary file with the constant energy (energy between fixed residue and the mutable/flexible with the fixed ones). (It should be in the format of double)
- Amino.txt: There should be a text file with each row correspond to one rotamer with their amino acid code based on the following table. They should follow the same order with the Rot_positions file.

Amino acid	Code
ALA	0
ARG	1
ASN	2
AS1	3
AS2	4
ASP	5
CYS	6
GLN	7
GL1	8
GL2	9
GLU	10
GLY	11
HIS	12
HSE	13
HSD	14
HSP	15
ILE	16
LEU	17
MET	18
PHE	19
PRO	20
SER	21
THR	22
TRP	23
TYR	24
VAL	25
LYS	26

- Energies.dat: A binary file contains both singleton and pairwise energy terms for the protein. (All the values should be in double format). All the singleton energy terms should be provided at first and then the all the pairwise energy terms should be provided. Singleton energy terms should be written in the energy.dat binary file in the same order with Rot_positions file. Then, just the half rectangle of pairwise energy terms should be provided in the same order.

2.2 Multistate protein design with ensemble of substates

Suppose for the positive state there are P proteins and for negative state there are Q proteins. At first, for each of the protein, the 4 mentioned files should be provided. Moreover, following files should be provided:

- Path of all amino.txt: Two files should be created separately for the positive and negative state. Each of which should contain the absolute path to the P (Q) Amino.txt files for positive (negative) state.
- Path of all rot_positions: Two files should be created separately for the positive and negative state. Each of which should contain the absolute path to the P (Q) rot_positions files for positive (negative) state.
- Path of all energy.dat: Two files should be created separately for the positive and negative state. Each of which should contain the absolute path to the P (Q) energy.dat files for positive (negative) state.
- Path of all constant.dat: Two files should be created separately for the positive and negative state. Each of which should contain the absolute path to the P (Q) constant.dat files for positive (negative) state.
- Rot_input: There should be a file for the general design problem which contains all of the mutable or flexible residues with wild-type and mutant amino acids. (All proteins should have same mutable/flexible residues with the same wild-type amino acid). It should contains at least 3 columns with each corresponds to a mutable or flexible residues:
  - First column: chain of the each residue
  - Second column: residue number
  - Third column: Wild type amino acid of residue with three letter codes given in the above table.
  - Fourth and so on: For any residue which is mutable, you should list all the amino acid it should be mutate to with a space separated.

2.3 Toy example

I will explain about the detail of providing data for iCFN with a very simple toy example. Suppose there are two state in which positive state has two sub-states and the negative state has just one sub-state. (Therefore there are 3 proteins) Suppose there are 3 mutable or flexible residues in this toy

example.
Rot_input format: The toy example, contains 3 residues with 2 mutable residues (26, 54) and one flexible (32). The first residue is ASP in wild type which can be mutated to ARG/ASN. Second residue is GLY which is just flexible. Third residue is VAL in wild type which can be mutate to GLU.

A	26	ASP	ARG	ASN
A	32	GLY		
A	54	VAL	GLU	

Rot_positions format: It should be in format similar to the following. Suppose first sub-state in positive state: It shows that the first residue in total has 5 rotamers (considering rotamers of wild type and mutates). Second residue has just one rotamer. Third residue has in total 5 rotamers.

A	26	ASP	5
A	32	GLY	1
A	54	VAL	5

Second sub-state in positive state:

A	26	ASP	4
A	32	GLY	1
A	54	VAL	3

First sub-state in negative state:

A	26	ASP	3
A	32	GLY	1
A	54	VAL	4

The number of rotamers for a given amino acid in different sub-state can vary as the toy example shows.
Amino files: I will give an example for the first sub-state in the positive state and the other two protein are very similar. Suppose for the first residue, there 2 rotamers for ASP (with code 5), 2 for ARG (with code 1) and 1 for ASN (with code 2). For the second residue, there are 1 rotamer for the GLY (with code 11). There are 2 rotamers for VAL (with code 25) and 3 for GLU (with code 10). Therefore, the amino.txt for this protein will be:

```
5
5
1
1
2
11
25
25
10
10
10
```

Energy file: Suppose for the above example (just one of the proteins), singleton energy terms are as follow in which r_j^i corresponds to the ith rotamer in jth residue:

$$E(r_1^1) = -1 \quad E(r_1^2) = -2 \quad E(r_1^3) = 0 \quad E(r_1^4) = -10 \quad E(r_1^5) = -1.5$$
$$E(r_2^1) = -2.3$$
$$E(r_3^1) = -1 \quad E(r_3^2) = -6 \quad E(r_3^3) = -1.7 \quad E(r_3^4) = -9 \quad E(r_3^5) = -3$$

Pairwise energy terms for the first and second rotamer of the first residue with rotamer of all the other residue are given as following:

$$E(r_1^1, r_2^1) = -0.6 \quad E(r_1^1, r_3^1) = -1 \quad E(r_1^1, r_3^2) = 0 \quad E(r_1^1, r_3^3) = 10000$$
$$E(r_1^1, r_3^4) = -1.5 \quad E(r_1^1, r_3^5) = 3 \quad E(r_1^2, r_2^1) = 10 \quad E(r_1^2, r_3^1) = -9$$
$$E(r_1^2, r_3^2) = 5 \quad E(r_1^2, r_3^3) = 10000 \quad E(r_1^2, r_3^4) = 10000 \quad E(r_1^2, r_3^5) = 2.4$$

Therefore, the energy file should be in the following order but it should be converted to binary file (double format):

```
-1
-2
0
-10
-1.5
-2.3
-1
-6
-1.7
-9
-3
-0.6
-1
0
10000
-1.5
3
10
-9
5
10000
10000
2.4
```

At first all the singleton energy term should be provided with same order as amino.txt. First 5 values correspond to the singleton energy terms of the first residue, the next one value to the second residue and the next 5 values correspond to third residue. After all the singleton energy terms are provided, the pairwise energy terms between first rotamer in first residue with rotamer of the succeeding residues should be provided. (For example the next 6 values correspond to pairwise energy terms of the first rotamer of the first residue with one rotamer of the second residue and the 5 rotamer of the third residue). Then, all the pairwise energy terms between the second rotamer of the first residue with rotamer of the succeeding residues should be provided. We goes on until we have the pairwise energy terms between all rotamers of the first residue with all the rotamers of succeeding residues. Then, pairwise energy terms of first rotamer of second residue with all the rotamer of succeeding residues (third, fourth,$\cdots$) should be provided.

3 Replication of TCR design

To easily replicate our results in the paper for the T Cell Receptor (TCR) design, we provided a shell script with all needed files in correct directories with the path given in the shell script. With this shell scripts, one can run our single point mutation designs for TCR.
To run it:

```
$ ./run_protein_design_TCR_Multistate.sh 26 2 2 5 10 > log_output &
```

in which the arguments correspond to:

- First argument: the position we want to mutate. In our design there are four positions 26, 28, 98 and 100.
- Second argument: Energy cutoff for side chain packing and pruning by type dependent DEE. In the above example it is 2 Kcal/Mol.
- Third argument: Energy cutoff for across substate type dependent DEE. In the above example it is 2 Kcal/Mol.
- Fourth argument: Stability energy cutoff for positive state from wild type. In the above example it is 5 Kcal/Mol.
- Fifth argument: Specificity energy cutoff for ensemble of sequences. In the above example it is 10 Kcal/Mol.

To check if one have run the code correctly, we provided the correct results in the folder outputs/correct_output_MSD.

4 iCFN for new design

To run iCFN for a new design, we have provided a shell script. At first, one need to provide the files for each of the proteins in two states. When the files are ready, We create folder "new_data", in which has two empty folders of "positive" (for the protein correspond to positive state) and "negative" (for the protein correspond to negative state). One should do the following:

1. `$ cd new_data`

2. Put the rot_input, with exactly "rot_input" name inside of the "new_data" folder.

3. `$ cd positive`

4. Put all the proteins correspond to positive state and their files in "positive" directory. Provide the following as well:
    - Path of all rot_positions files with the name "rot_pos_positive_files.txt"
    - Path of all binary energy files with the name "energies_positive_files.txt"
    - Path of all amino files with the name "amino_positive_files.txt"
    - Path of all constant energy files with the name "const_positive_files.txt"

5. `$ cd negative`

6. Similarly, put all the proteins correspond to negative state and their files in "negative" directory. Provide the following as well:
    - Path of all rot_positions files with the name "rot_pos_negative_files.txt"
    - Path of all binary energy files with the name "energies_negative_files.txt"
    - Path of all amino files with the name "amino_negative_files.txt"
    - Path of all constant energy files with the name "const_negative_files.txt"

7. `$ cd ../../src`

8. Run

    ```
    ./run_protein_design_Multistate.sh
    ```

    with the following arguments:

    - First argument: Energy cutoff for side chain packing and pruning by type dependent DEE

- Second argument: Energy cutoff for across substate type dependent DEE.
- Third argument: Stability energy cutoff for positive state from wild type.
- Fourth argument: Specificity energy cutoff for ensemble of sequences.
- Fifth argument: Maximum number of conformation for each substate per amino acid assignment (Integer value)
- Sixth argument: Maximum number of discrepancy, sequences can have from wild type. (Integer value). If you don't want to constraint it, you can put arbitrary large number. For example at least equal the number of mutable residues.
- seventh argument: The output name

9. Then the outputs will be generated in the "outputs" directory.

4.1 Running protein design based on OSPREY energy calculation

If you have already calculated the energy based on OSPREY, we have provided a source code to create all the necessary files for a given pdb. This code, will create the float version of binary energies, therefore you need to use our **float** version binary code. To do that:

1.
   ```
   $ cd OSPREY-conversion
   ```

2. Run
   ```
   $ make_io.o
   ```

   with following arguments:

   - First argument: OSPREY energy file for a given pdb
   - Second argument: output name of Rot_position file
   - Third argument: output name of energy file
   - Fourth argument: output name of amino file
   - Fifth argument: output name of fleximer file (An internal file might not need it later)
   - Sixth argument: output name of constant energy term file

Then the files will be ready based on the names you gave in the arguments. There are two example of energy calculation from OSPREY in the "OSPREY-conversion/example" directory with the names 3K75_unbound (as negative state) and 3K75_bound (as positive state). Try to run our code for this example!!