pyconsFold: Supplemental Notes

Lamb J., Elofsson A.

1 Installation

pyconsFold requires a working installation of CNS. This needs to be done manually due to license.

1.1 CNS

1. Request a download link from CNS.
2. Follow the emailed instructions to download
   ```
   cns_solve_1.3 _all_intel −mac_linux.tar.gz
   ```
3. Extract the files
   ```
   tar xzvf cns_solve_1.3 _all_intel −mac_linux.tar.gz
   ```
4. Change into the resulting directory
   ```
   cd cns_solve_1.3
   ```
5. Unhide the bash-specific file
   ```
   mv .cns_solve_env_sh cns_solve_env.sh
   ```
6. In this resulting file, replace _CNSsolve_location_ with the CNS installation folder. If you extracted the file in your home folder then the CNS installation would be:
   /home/<your username>/cns_solve_1.3
7. Source CNS, source cns_solve_env.sh[1], to make this permanent and to prevent you having to do this every time, add it to your .bashrc file.
8. Test CNS by going into the test folder cd test and run the tests
   ```
   ../bin/run_tests −tidy *.inp
   ```

1.2 pyconsFold

1. Run:
   ```
   pip3 install pyconsFold
   ```
   It is recommended to run a separate python environment for pyconsFold, ie virtualenv.

[1]If you get an error about csh interpreter, you need to install csh

2 Examples

2.1 Distance modelling

Distance modelling is done using a contacts file with predicted distances and errors.

```
import pyconsFold
pyconsFold.model_dist(fasta_file, contact_file,
                      output_directory)
```

2.2 Classical modelling

Classical model is done by treating the contacts file as binary contacts and using a static distance and error. This means a contacts file with predicted distances can be used as input but the program will treat the contacts as static contacts with the same distance and error.

```
import pyconsFold
pyconsFold.model(fasta_file, contact_file,
                 output_directory)
```

2.3 Docking modelling

Docking requires a contacts file with both inter and intra contacts between the two proteins. If no inter contacts are found, a warning is generated and an artificial contact is created between the centers of the two proteins to prevent the program from failing. Either predicted distances or classical contacts can be used, see the dist parameter.

```
import pyconsFold
pyconsFold.model_dock(first_fasta_file,
                      second_fasta_file,
                      contact_file, output_directory)
```

Further examples and use of advanced paramters can be found in in "examples/examples.py" in the github repository.

3 Arguments

Multiple optional arguments are available. The following are the most commonly used. For a full list, see the github repository.

```
rr_pthres  -- Threshold for the confidence we
              want in a prediction (default
              model(0.80), model_dist(0.45),
              model_dock(0.50))
```

```
rr_sep     -- Separation between contacts
              (default 0)
save_step  -- Save working steps
              (default False)
stage2     -- Run stage2, filter contacts vs
              generated structure and generate
              new structures with filtered
              contacts (default False)
debug      -- Write out debug information
              (default False)
selectrr   -- How many contacts to use?
              Can be "all", "#L", or #.
              (default "all")
mcount     -- How many models to generate?
              (default 20)
top_models -- How many of the generated
              models should be ranked and
              saved? (default 20)
use_angles -- If predicted angels should be
              used, only works with npz
              (default False)
omega      -- RR-formated file with omega
              angles (if npz are not used)
              (default '')
theta      -- RR-formated file with theta
              angles (if npz are not used)
              (default '')
```

4 Arguments

QA-function arguments to all above functions:

- pcons (default False) – If set to true, gives pcons scores for all models (using either pcons installed in the PATH or the builtin binary)
- tmscore_pdb_file – If a structure file is supplied, runs all models against this (presumed) native structure and reports the TMscore (using either TMscore in the PATH or builtin binary)

5 Extras

```
from pyconsFold.utils import npz_to_casp, pdb_to_npz

npz_to_casp("trRosetta.npz")  ## Converts trRosetta
                              ## distance and angle
```

```
                                   ## predictions to CASP
                                   ## format in separate files

pdb_to_npz("structure.pdb")        ## Converts a structure
                                   ## (pdb/mmCif) to trRosetta
                                   ## distances and angles,
                                   ## useful when
                                   ## investigating how well
                                   ## a model conforms to
                                   ## restraints
```

6 Adjustable parameters for CNS, advanced

```
rrtype      -- Between which atoms in a residue are the contacts?
               (default 'cb')
lbd         -- Lambda, 0.1-10 (default 0.4)
contwt      -- Contact restraint weights, 0.1-10000 (default 10)
sswt        -- Secondary structure weights, 0.1-100 (default 5)
bin_values  -- Dictionary of bin_values for conversion of npz to
               RR-format, see source code (default {})
```

7 Benchmark & Data

Three datasets has been tested: PconsC3 (see github repository and Figure S1) and CASP13 (see Table S1) on individual chains. The docking was benchmarked on all 222 heterodimeric pairs from Dockground 4.3 (see Figure 1 in the main text).

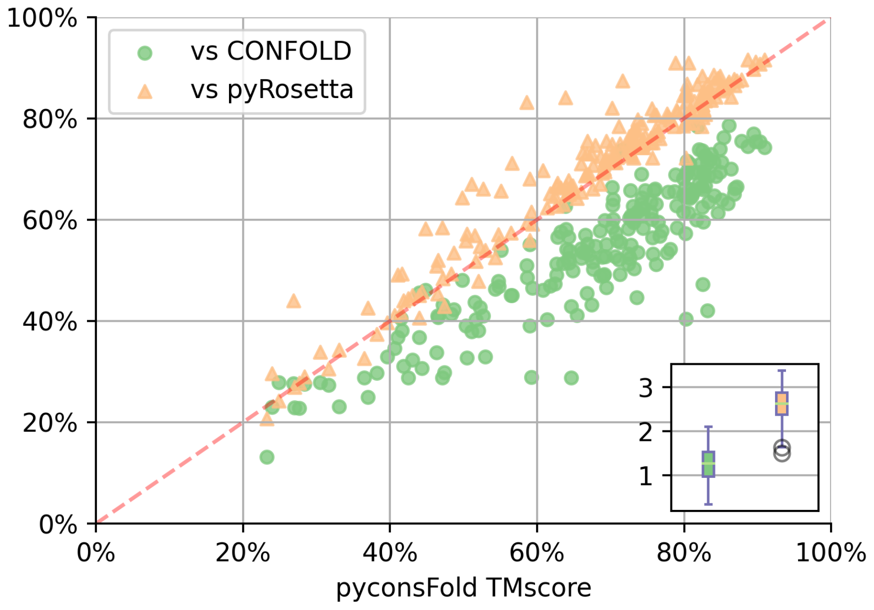

Figure S1: pyRosetta, pyconsFold and CONFOLD models on the PconsC3 dataset. pyconsFold against both contact based CONFOLD (green circles) and pyRosetta (orange triangles) models. pyRosetta and CONFOLD TMscores are on the y-axis while the pyconsFold distance based TMscores are on the x-axis. pyconsFold predictions outperforms CONFOLD in almost all cases. pyconsFolds models perform almost as well as pyRosetta but is around 20 times (more than 1 order of magnitude) faster per model as the inset shows(log10 scale of per model time in seconds). pyconsFold in green and pyRosetta in orange.

Target	trRosetta	pyconsFold	CONFOLD
T0950	0.6358	0.2994	0.1519
T0951	0.8598	0.8625	0.3305
T0952	0.5357	0.4673	0.1487
T0954	0.7830	0.7172	0.1554
T0955	0.5155	0.4978	0.2023
T0958	0.6107	0.6072	0.2113
T0960	0.2394	0.2209	0.1794
T0961	0.8398	0.7378	0.2139
T0963	0.1977	0.2124	0.1965
T0965	0.7989	0.7873	0.2583
T0966	0.3328	0.2307	0.1615
T0967	0.7570	0.7322	0.2396
T0969	0.7750	0.4528	0.2781
T0970	0.4651	0.4304	0.1486
T0971	0.8939	0.3275	0.2922
T0976	0.5043	0.3257	0.1585
T0982	0.4766	0.4574	0.1815
T0983	0.8713	0.7956	0.2940
T0984	0.2563	0.2050	0.0931
T0987	0.3546	0.3728	0.1763
T0990	0.5148	0.3018	0.1004
T0996	0.2398	0.2064	0.1444
T1000	0.7847	0.6588	0.2922
T1003	0.7998	0.5490	0.3410
T1005	0.6684	0.6396	0.3207
T1006	0.8824	0.8652	0.2824
T1008	0.3731	0.3866	0.1698
T1009	0.7825	0.7791	0.2012
T1010	0.2961	0.2908	0.1328
T1011	0.4929	0.2688	0.1303
T1016	0.8756	0.8633	0.3255
T1018	0.8569	0.8600	0.2320

Table S1: TMscore for models generated with the trRosetta, distance based pyconsFold and contact based classic CONFOLD. There is a clear difference between the binary contact based CONFOLD and the distance baed trRosetta and pyconsFold.