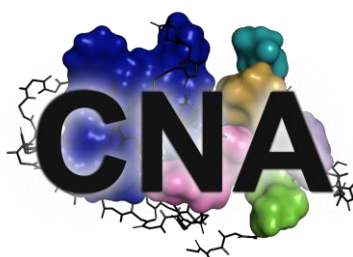


User Guide

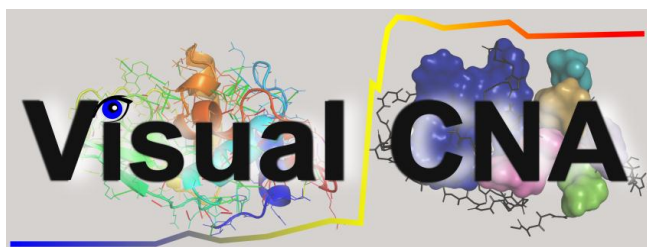
# **Constraint Network Analysis (CNA)**

**v2.0**



and

**VisualCNA v1.0**



Christopher Pfleger, Prakash Chandra Rathi, Daniel Mulnaes,  
Sebastian Radestock, Doris L. Klein, and Holger Gohlke

March 2014

## Table of contents

1	Installation .....	4
1.1	CNA Installation .....	4
1.2	VisualCNA Installation .....	4
2	Structure preparation .....	5
3	Running CNA.....	6
3.1	General program call .....	6
3.2	Description of options.....	7
4	Program calls for dilution runs .....	12
4.1	Calculating dilutions from a single input structure.....	12
4.2	Calculating dilutions from fuzzy-networks or ensembles .....	12
5	Indices to characterize flexibility and rigidity.....	14
5.1	Global flexibility indices .....	14
5.2	Local flexibility indices .....	14
6	Input files.....	16
6.1	PDB file list .....	16
6.2	Network information .....	16
7	Results files .....	17
7.1	Local indices file (*_local_indices.dat) .....	17
7.2	Global indices file (*_global_indices.dat).....	18
7.3	Transitions file (*_transitions.dat).....	19
7.4	Calculation status (*_status.dat).....	20
7.5	Parameter settings (parameters.out).....	20
7.6	Unfolding nuclei (*_unfolding_nuclei.dat) .....	20
7.7	PDB files ( *_unfolding_nuclei_type*.pdb).....	21
8	Using VisualCNA.....	22
8.1	Input formats.....	22
8.1.1	PDB files .....	22
8.1.2	Network input files .....	22
8.1.3	Result files from CNA runs.....	22
8.2	Starting VisualCNA (from within PyMOL) .....	22
8.3	Typical work-flows within VisualCNA.....	22
8.3.1	Setting up and running a CNA simulation ( <i>Setup</i> panel; Figure 1).....	23
8.3.2	Analyzing CNA data with VisualCNA ( <i>Analyze</i> panel, Figure 2).....	25

8.3.3	Modifying CNA constraint network with VisualCNA ( <i>Modify</i> panel, Figure 3)	.27
8.3.4	Modifying protein structure with VisualCNA (Mutate Panel, Figure 4)	.....29
9	Citing CNA.....	31
10	Third party licensing .....	32
10.1	Python .....	32
10.2	BioPython.....	33
10.3	SciPy .....	33
10.4	NumPy .....	34
10.5	SWIG .....	35
10.6	Open Babel.....	36
10.7	Matplotlib.....	36
10.8	Setuptools.....	38
10.9	PyMol.....	38
10.10	Tkintertable .....	39
11	References .....	40

## 1 Installation

### 1.1 CNA Installation

CNA requires the FIRST program, SWIG, Python® and four Python extensions (BioPython, Scipy, NumPy, and Open Babel). Except the FIRST program, all external programs are provided with the package. The FIRST source (not binaries!) can be obtained from Arizona State University at <http://flexweb.asu.edu/software/first/>.

To install the Constraint Network Analysis (CNA), copy the FIRST source tar ball (version 6.2/development version) in the \$BASE/CNA/ext/src/tgz directory (where \$BASE is the location of the CNA software package) and then run ‘make install’ from the CNA directory. Note that a “GNU make” must be available on the system you want to install CNA on. CNA is installed locally and, hence, no root privilege is required for installation.

CNA requires DSSP (available from <http://swift.cmbi.ru.nl/gv/dssp/>) for calculating dilutions from fuzzy-networks (see section 4.2). Therefore, you need to set the variable DSSP\_EXE in \$BASE/CNA/CNA.sh file to the right location of your DSSP executable.

After installing the program and setting up the variable in the CNA.sh file, go into the test directory and run ‘make test\_all’. This will run the tests and compare their output with the one in the saved output files. For the ent\_fnc (see section 3.2 for a description) test case, a fixed seed for generating random numbers is used to ensure that the results are identical to the saved results. However, in case of an actual ent\_fnc run, the system time is used as a seed.

CNA is also available as a web service at <http://cpclab.uni-duesseldorf.de/cna>.<sup>1</sup>

### 1.2 VisualCNA Installation

To install VisualCNA run install.sh script (make sure that it is executable) and follow the on-screen instructions. You are not required to be root for installing VisualCNA. Just provide an installation directory (here named \$BASE) where you have writing permissions. If you have already CNA installed with the CNA executable being located in \$BASE/CNA/CNA.sh, provide \$BASE as the path to the installation directory. VisualCNA can then setup and run CNA calculations. If you do not have CNA installed, Visual CNA can only be used to analyze results generated externally with CNA. For further details (including software dependencies) please read the README file in the VisualCNA installation package.

## **2 Structure preparation**

Make sure that hydrogen atoms have been added to the PDB structure before using CNA. Each subunit in the structure (e.g., different chains, ligands) has to be separated by a TER record. Residue IDs should be consecutive, and a residue should be named either by Amber or by PDB standard residue names. At present, the maximum number of chains allowed in the input protein structure is 26. Constraints for ligands are correctly identified automatically, provided they are hydrogenated appropriately. Water molecules, if present in the structure, are considered automatically by the program.

## 3 Running CNA

### 3.1 General program call

CNA<sup>2,3</sup> works as a front- and back-end to the FIRST program (<http://flexweb.asu.edu/>).<sup>4</sup> The FIRST program identifies rigid and flexible regions in protein structures using graph theory and employing the pebble game algorithm.

The program call in general is:

```
CNA.sh          [--help, -h||
                  --snt ||
                  --ent ||
                  --ent_fnc]

-E <E_stop> <E_start> [<E_step>]
-H <function_type>
--TUS <TUS_type>
-c <c_start> [<c_stop>]
--input, -i <PDB_file or infile or auto>

-F <F_steps>
--aic
--unfolding_nuclei <un_type>
    --transitions_source <source_type>

--stbmap
--netout
--netin <netin_file>
--res_dir <results_directory_name>
--all_results
--write_ligand_mol2
--verbose, -v <verbosity_level>
```

### 3.2 Description of options

#### **--help**

Description Shows the options and exit

#### **--snt**

Description Thermal unfolding simulation on a single network topology

#### **--ent**

Description Thermal unfolding simulation on an ensemble of network topologies created from an ensemble of input structures

#### **--ent\_fnc**

Description Thermal unfolding simulation on an ensemble of network topologies created from a single input structure

#### **-E**

Type Real number as  $\langle E_{\text{stop}} \rangle \langle E_{\text{start}} \rangle [\langle E_{\text{step}} \rangle]$

Default -7.5 -0.25 None

Description Sets the energy cut-off ( $E_{\text{cut,hb}}$ ) for including hydrogen bonds and salt bridges in the network. **-E** defines the range and step size determining how hydrogen bonds and salt bridges will be diluted from the network. The network state is built at  $E_{\text{cut,hb}} = E_{\text{start}}$  and then  $E_{\text{step}}$  is subtracted from the current  $E_{\text{cut,hb}}$  to create a new network state until the  $E_{\text{stop}}$  is reached. If  $E_{\text{step}}$  is not provided then each hydrogen bond/salt bridge will be removed sequentially in the ascending order of their strength.

#### **-H**

Type Integral number as  $\langle \text{method\_type} \rangle$

Default 1

Description Sets the method [0, 1, 2, 3] according to which hydrophobic tethers are placed between carbon and sulfur atoms in the network. For each method, all conditions (listed below) less than or equal to the method value are to be satisfied for placing a hydrophobic tether. A value of 1 places the most hydrophobic tethers whereas a value of 3 places the least. No hydrophobic tether are considered while building the network if a value of zero is used.

1. Carbon-carbon, carbon-sulfur, or sulfur-sulfur atom pairs satisfy the cutoff for identifying hydrophobic tethers (see below)
2. Each carbon and/or sulfur in the tether is only bonded to carbon, sulfur or hydrogen.
3. A given atom is only allowed to have one tether to another residue.

**-c**

Type	Real number as <c_start> [<c_stop>]
Defaults	0.25 when TUS_type is 1 0.25 0.35 when TUS_type is 2
Description	Sets the cut-off for identifying hydrophobic tethers ( $D_{cut, hp}$ ). -c defines the minimum distance between vdW radii of two carbon and/or sulfur atoms for which a hydrophobic tether is formed.

**--TUS**

Type	Integral number as <TUS_type>
Default	1
Description	Sets the TUS_type [1, 2] according to which hydrophobic tethers are treated during the Thermal Unfolding Simulation (TUS). The hydrophobic constraints are strengthened with increasing temperature if TUS_type is 2 (otherwise, the strength of the hydrophobic constraints is kept constant). To do this, the $D_{cut, hp}$ is increased linearly from c_start to c_stop for $E_{cut, hb} = 0.0$ to $E_{cut, hb} = -6.0$ according to the equation below <sup>5</sup> .

$$D_{cut, hp} = \begin{cases} c_{start} - \left( E_{cut, hb} * \frac{c_{start} - c_{stop}}{-6.0} \right), & \text{if } -6.0 \leq E_{cut, hb} \leq 0.0 \\ c_{start}, & \text{if } E_{cut, hb} > 0.0 \\ c_{stop}, & \text{if } E_{cut, hb} < -6.0 \end{cases}$$

**-i**

Type	String as <PDB_file or infile or auto>
Description	A single pdb file is needed for snt and ent_fcn runs. For an ent run, a file (infile) containing the names of pdb_files (one per line) or simply 'auto' (without



quotes) can be provided. The latter considers all pdb-files in the directory for analysis.

**-F**

Type	Integral number as <F_steps>
Default	100
Description	Sets the number of fuzzy network topologies that are new network topologies created by adding white noise to hydrogen bond and hydrophobic tethers in the initial network. This way CNA generates an ensemble of network topologies from a single input structure.

**--aic**

Description	Akaike information criteria based selection between single and double sigmoid fit for finding transition on CCE type 2.
-------------	---

**--unfolding\_nuclei**

Type	Integral number as <un_type>
Default	1
Description	Sets the unfolding nuclei type [1, 2, 3, 4] which defines the method of unfolding nuclei identification. More than one type can be entered separated by space. The following sets of residues are predicted as weak spots in case of each of these four different types. <ol style="list-style-type: none"><li>1. Residues in the largest rigid cluster becoming flexible after the phase transition temperature.<sup>5-7</sup></li><li>2. Residues in any rigid cluster with size <math>\geq 30</math> atoms becoming flexible after the phase transition temperature.</li><li>3. Residues of the largest rigid cluster making hydrogen bonds (critical hydrogen bonds) which are lost after the phase transition temperature.</li><li>4. Residues around 5 Å of the critical hydrogen bonds (see type 3) .</li></ol>

**--transition\_source**

Type	String as <source_type>
Default	cce2_sigmoid

**Description** Sets `source_type` [`cce1_spline`, `cce2_spline`, `cce2_sigmoid`, `rop1_spline`, `rop2_spline`, `rop3_spline`] for computing the phase transition temperature used for unfolding nuclei identification. The source type indicates the global index (first word) and method (second word) used for the identification of phase transition temperature. Either a spline interpolation or a double sigmoid curve fitting to the global index values is used to identify the phase transition temperature. For the description of the indices, refer to the section 5.1 and reference <sup>8</sup>.

#### **--stbmap**

**Description** Stability maps are calculated when this flag is set.<sup>7</sup>

#### **--netout**

**Description** Network constructed by FIRST is written to a file for each pdb-file analyzed (except in the case of `ent_fnc` run) which can be used as input to the CNA program. This option can be particularly useful for incorporating user-defined constraints in the network. A user might want to define or modify constraints identified by the program for non-standard residues, ions and ligands.

#### **--netin**

**Type** String as `<netin_file>`

**Description** Sets the `netin_file` containing network information. The type of constraints provided in this file will not be identified by the program, rather they will be read from this file. However, types of constraints which are not provided in this file will be identified by the program.

#### **--res\_dir**

**Type** String as `<results_directory_name>`

**Default** 'results'

**Description** Sets the name of a sub-directory created in the directory from where the CNA is run. All the output files from the program are written to this sub-directory.

**--all\_results**

Description Local and global indices for each snapshot from an ensemble are written out if this option is set.

**--write\_ligand\_mol2**

Description Writes out each ligand as a mol2 file. The bond orders for the ligand are automatically detected by the CNA.

**--verbose, -v**

Type Integral number as <verbosity\_level>

Default 1

Description Sets the level [0, 1, 2, 3] of printing messages to the terminal from the CNA program. A higher number will produce more output to the terminal; a value of 0 will run the program in quiet mode without any messages to terminal.

## 4 Program calls for dilution runs

### 4.1 Calculating dilutions from a single input structure

Performs a thermal unfolding simulation on a single protein structure

USAGE:

CNA.sh

```
--snt
-E <E_start> <E_stop> [<E_step>]
-H <function_type>
[--TUS <TUS_type>]
-c <c_start> [<c_stop>]
-i <pdb_file>
[--stbmap]
[--unfolding_nuclei] <un_type>
      [--transitions_source] <source_type>
[--netout]
[--res_dir <results_directory>]
[--write_ligand_mol2]
[--netin] <netin_file>
--verbose, -v <verbosity_level>
```

### 4.2 Calculating dilutions from fuzzy-networks or ensembles

Performs thermal unfolding simulations either on an ensemble of network topologies derived from one input structure by the use of fuzzy constraint definitions<sup>9</sup> or on a structural ensemble derived, e.g., from an MD trajectory.<sup>5</sup>

USAGE:

CNA.sh

```
--ent_fnc || --ent
-E <E_start> <E_stop> [<E_step>]
-H <function_type>
[--TUS <TUS_type>]
-c <c_start> [<c_stop>]
-i <pdb_file> for ent_fnc, <infile or 'auto'> for ent
-F <F_steps> for ent_fcd
[--stbmap]
```

---

```
[--unfolding_nuclei] <un_type>
      [--transitions_source] <source_type>
[--netout]
[--res_dir <results_directory>]
[--all_results]
[--write_ligand_mol2]
[--netin] <netin_file>
--verbose, -v <verbosity_level>
```

## 5 Indices to characterize flexibility and rigidity

CNA calculates indices to characterize global and local flexibility/rigidity of biomolecules. These indices link purely mechanical information from a constraint network to biologically relevant characteristics. All global and local indices are derived by analyzing the constraint network along a thermal unfolding simulation.<sup>8</sup> In the case of an ensemble analysis, local indices are averaged over the entire ensemble.

All indices are calculated by default except for the stability maps and are given in the output files \*\_local\_indices.dat and \*\_global\_indices.dat. See section 6 for details.

### 5.1 Global flexibility indices

Rigidity order parameter ( $P_\infty$ , ROP): Monitors the fraction of the network belonging to the giant percolating cluster (type 1) or the actual largest rigid cluster (type 2).

Cluster configuration entropy ( $H$ , CCE): Monitors the degree of disorder in the realization of a given state. In contrast to the original definition (type 1), the quadratic cluster size definition is used in the type 2 entropy.

Mean rigid cluster size ( $S$ , MCS): Monitors the size distribution of rigid clusters.

Floppy mode density ( $\Phi$ , FMD): The number of internal independent degrees of freedom, normalized by the number of overall internal degrees of freedom associated with the number of atoms in the network.

### 5.2 Local flexibility indices

Percolation index ( $p$ ): Monitors the hydrogen bond cutoff energy  $E_{cut,hb}$  at which a bond segregates from the *giant* cluster (type 1) or the largest rigid cluster at each dilution step (type 2). For a  $C_\alpha$  atom-based representation of a protein structure, the lower of the  $p_i$  values of the two backbone bonds is taken.

Rigidity index ( $r$ ): Monitors the hydrogen bond cutoff energy  $E_{cut,hb}$  at which a bond segregates from *any* rigid cluster. For a  $C_\alpha$  atom-based representation of a protein structure, the average of the  $r_i$  values of the two backbone bonds is taken.

Side chain rigidity index (experimental): Monitors the hydrogen bond cutoff energy  $E_{cut,hb}$  when the  $C_\alpha$  and  $C_\beta$  atom of a residue are not in the same rigid cluster. In case of glycine and proline residues the value will be 'NaN'.

Rigid contact index (experimental): Monitors the hydrogen bond cutoff energy  $E_{cut,hb}$  when no rigid contact exists to any other residue. A rigid contact exists if the two residues are in the same rigid cluster.

Stability maps ( $rc_{ij}$ ): A stability map is a 2-dimensional itemization of the rigidity index  $r$  and is derived by identifying “rigid contacts” between two residues along a thermal unfolding simulation. A rigid contact exists if two residues belong to the same rigid cluster. Input and output files.

## 6 Input files

### 6.1 PDB file list

```

XXX_0001.pdb
XXX_0002.pdb
XXX_0003.pdb
XXX_0004.pdb
XXX_0005.pdb
...

```

### 6.2 Network information

#TYPE	ATOM 1	ATOM 2	BARS	RECORD1	RECORD2
COV	1	2	5	ATOM	ATOM
COV	1	3	5	ATOM	ATOM
...					

#TYPE	HYDROGEN	ACCEPTOR	BARS	ENERGY	RECORD1	RECORD2	HBTYPE
HBOND	2	19	5	-2.01	ATOM	ATOM	HB
HBOND	4	11	5	-2.30	ATOM	ATOM	HB
...							

#TYPE	ATOM 1	ATOM 2	BARS	DISTANCE	RECORD 1	RECORD 2
HPHOBES	2270	2279	2	3.206	ATOM	ATOM
HPHOBES	1277	1374	2	3.234	ATOM	ATOM
...						



## 7 Results files

### 7.1 Local indices file (\*\_local\_indices.dat)

In case of a dilution run a similar file without the standard error fields is written out.

```

No.  Resi Chain Perc.Ind(1)      Std  Perc.Ind(2)      Std  ...
1      1          -0.084  0.206          -0.084  0.206
2      2          -0.212  0.350          -0.212  0.350
3      3          -0.240  0.372          -0.258  0.467

...    Rig.Ind      Std  SC_Rig.Ind  Std  Ric.Ind      Std
      -0.068    0.189    -0.244      0.486  -0.076      0.207
      -0.154    0.315     NaN        NaN    -0.200      0.355
      -0.429    0.865    -0.128      0.183  -0.393      0.766

```

Field	Description
1	Serial number
2	Residue number
3	Chain Id
4-5	Average percolation index 1 and standard deviation
6-7	Average percolation index 2 and standard deviation
8-9	Average rigidity index 1 and standard deviation
10-11	Average side chain rigidity index and standard deviation
12-13	Average rigidity contact index and standard deviation

## 7.2 Global indices file (\*\_global\_indices.dat)

```
# No.      Egy      Temp      CCE1      CCE2...
```

1	-0.000	300.00	1.021	0.008
2	-0.100	302.00	1.268	0.015
3	-0.200	304.00	1.270	0.015

```
...ROP1      ROP2      ROP3      MCS      FMD
```

0.701	0.701	0.726	3.796	0.013
0.623	0.624	0.677	4.478	0.015
0.621	0.621	0.679	4.510	0.016

Field	Description
1	Serial number
2	$E_{cut,hb}$ [kcal mol <sup>-1</sup> ]
3	Temperature [K]
4	Cluster configuration entropy type 1
5	Cluster configuration entropy type 2
6	Rigidity order parameter type 1
7	Rigidity order parameter type 2
8	Rigidity order parameter type 3
9	Mean cluster size
10	Floppy mode density

### 7.3 Transitions file (\*\_transitions.dat)

A similar file without the summary of transitions (mean, median, and standard error) at the end of the file is produced in case of a dilution run.

```

Name          CCE1_spline          CCE2_spline          ROP1_spline ...
              Egy  Temp              Egy  Temp              Egy  Temp
~~~~~
0003_F0001   -1.55 330.92         -1.56 331.28         -1.55 330.98
0003_F0002   -1.15 323.01         -1.61 332.11         -1.47 329.33
0003_F0003   -1.22 324.43         -1.85 336.95         -1.25 324.90

...          ROP2_spline          ROP3_spline          CCE2_sigmoid
              Egy  Temp              Egy  Temp              Egy  Temp
~~~~~
              -1.55 331.04         -1.56 331.28         -1.59 331.74
              -1.15 323.01         -1.27 325.49         -1.58 331.55
              -1.25 324.90         -1.58 331.63         -1.84 336.79

```

Field	Description
1	Pdb name/ent_fnc step
2-3	Phase transition identified on cce1-energy curve (using spline fitting) in $E_{cut, hb}$ [kcal mol <sup>-1</sup> ] and in temperature [K]
4-5	Phase transition identified on cce2-energy curve (using spline fitting) in $E_{cut, hb}$ [kcal mol <sup>-1</sup> ] and in temperature [K] <sup>6, 7</sup>
6-7	Phase transition identified on rop1-energy curve (using spline fitting) in $E_{cut, hb}$ [kcal mol <sup>-1</sup> ] and in temperature [K]
8-9	Phase transition identified on rop2-energy curve (using spline fitting) in $E_{cut, hb}$ [kcal mol <sup>-1</sup> ] and in temperature [K]
10-11	Phase transition identified on rop3-energy curve (using spline fitting) in $E_{cut, hb}$ [kcal mol <sup>-1</sup> ] and in temperature [K]
12-13	Phase transition identified on cce2-energy curve (using sigmoid fitting) in $E_{cut, hb}$ [kcal mol <sup>-1</sup> ] and in temperature [K] <sup>5</sup>

## 7.4 Calculation status (\*\_status.dat)

Gives information about the status and the runtime of each dilution run in the case of ent and ent\_fnc runs.

## 7.5 Parameter settings (parameters.out)

Writes out the call to the CNA program and the values of all the options chosen.

## 7.6 Unfolding nuclei (\*\_unfolding\_nuclei.dat)

In the case of a dilution run, all types of unfolding nuclei are written one after the other in a text file.

No	RES_NAME	RES_ID	CHAIN	Type1	Type2	Type3	Type4
1	ASP	1	A	0	0	0	1
2	THR	2	A	0	0	1	1
3	VAL	3	A	0	0	0	0

Field	Description
1	Serial number
2	Residue name
3	Residue Id
4	Chain Id
5	Unfolding nuclei type 1 with the percentage of the occurrence of the residue in the whole ensemble of networks
6	Unfolding nuclei type 2 with the percentage of the occurrence of the residue in the whole ensemble of networks
7	Unfolding nuclei type 3 with the percentage of the occurrence of the residue in the whole ensemble of networks
8	Unfolding nuclei type 4 with the percentage of the occurrence of the residue in the whole ensemble of networks

## 7.7 PDB files ( \*\_unfolding\_nuclei\_type\*.pdb)

The percentage of occurrence of a residue as a particular unfolding nuclei type is added as the B-factor value of its C $\alpha$  atom in the pdb file (the first pdb file is used if an ensemble run is carried out). For a dilution run, all the residues that are identified as an unfolding nucleus get a value of one in the b-factor column for their C $\alpha$  atom (all other residues get a value of two). By visualizing the molecule with PyMOL, one can color the molecule by their unfolding nuclei propensity, while choosing color according to B-factors.

## 8 Using VisualCNA

VisualCNA is a graphical user interface to the CNA program (working as a PyMOL plug-in) that allows setup of CNA runs and analysis of CNA results linking data plots with molecular graphics representations. VisualCNA also facilitates interactive protein engineering aimed at improving thermostability.

### 8.1 Input formats

#### 8.1.1 PDB files

... for PyMOL display and running CNA simulations. These must have alternate side-chains and atoms removed and all hydrogen's added; in the case of multiple models only the first model will be used. A detailed list of requirements for a PDB file can be found [here](#).

#### 8.1.2 Network input files

... are all generated in the correct format by CNA, and a detailed description of the file formats is found [here](#).

#### 8.1.3 Result files from CNA runs

[Result files from CNA runs](#) can be specified by naming a folder containing these files using the *Data Directory* button in the *Analyze* panel, which will automatically detect the files of each type in that directory. A detailed description of the result files is given [here](#).

### 8.2 Starting VisualCNA (from within PyMOL)

Click the *Plugin* tab in PyMOL and select *VisualCNA*. Alternately press Alt+P,Alt+V.

### 8.3 Typical work-flows within VisualCNA

Below four typical work-flows within VisualCNA are described. Each is linked to one of the panels *Setup*, *Analyze*, *Modify*, or *Mutate* in VisualCNA. In each panel, a detailed description of the functionality of each button will appear as a tool-tip when the mouse is hovered over that button for 2 seconds or more. If these tool-tips do not provide a sufficient answer more information can be found in previous sections in this user guide. Mandatory steps to complete a specific task are shown in bold. Numbers in square brackets refer to the red numbers in the corresponding Figure. Note that video tutorials for all four work-flows are available here:

<https://www.youtube.com/user/cpclabduesseldorf>

### 8.3.1 Setting up and running a CNA simulation (*Setup* panel; **Figure 1**)

(See video tutorial for this section here: <https://www.youtube.com/watch?v=3T2GhRUonu8>)

#### ❖ General

- **Select the type of CNA run to make (SNT, FNC or ENT) [1].**
  - SNT: Simulation on a **S**ingle **N**etwork **T**opology derived from a single input structure.
  - FNC: Simulation of an ensemble of network topologies derived from **F**uzzy **N**on-covalent **C**onstraints definitions as described in detail [here](#).
  - ENT: Simulation of an **E**nsemble of **N**etwork **T**opologies provided as PDB files in a specified input folder. These could originate from Molecular Dynamics simulations or other ensemble generation methods.

#### ❖ Input Settings

- Select the structure input type (molecule in PyMOL or PDB file) in the right side drop-down [2].
- **Select the structure input file/folder [3].**
- Select the network input type (molecule in PyMOL or Network file) in the right side drop-down [4].
- Select the network input file. Allowed file formats are CNA Network files as described [here](#) [5].
- Select if decision to use double sigmoid fit should be based on Akaike Information Criteria [6] as described [here](#).

#### ❖ Constraint Settings

- Select the type of hydrophobic modeling (TUS\_TYPE) to use [7], as described [here](#).
- Select the function (H) to identify hydrophobic constraints [8], as described [here](#).
- Select the initial and terminal energies and step-size for hydrogen bonds [9].
- Select the constant (TUS 1) or initial and terminal (TUS 2) distance cut-offs for hydrophobic tethers [10].

#### ❖ Unfolding Nuclei Settings

- Select the unfolding nuclei types to be calculated, as described [here](#) [11]
- Select the function fit used and global index used for detection of transition points [12]

#### ❖ Output Settings

- Select the types of output produced by VisualCNA [13].
- **Select the output data directory [14].**

#### ❖ Click *RUN CNA* [15]

VisualCNA v. 0.0

Setup Analyze Modify Mutate

**Visual CNA**

Thermal unfolding simulation on a single network created from a single input structure

SNT Analysis 1

**Input Settings:**

/home/mulnaes/src/VisualCNA/ PDB File 3 File 2

/home/mulnaes/src/VisualCNA/ Network 5 None 4

☐ Single or double sigmoid fit for CCE2 transition is based on Akaike Information Criteria 6

**Constraint Settings:**

The number of hydrophobic constraints remain constant during simulation

Place hydrophobic tethers between any atom-pair within a 0.25 Å cut-off if both atoms are C or S

TUS Type 1 7

H1 8

E-cutoff for H-Bonds: 9 StepSize 0.1 Initial -0.1 Terminal -6.0

Distance cutoff for placing Hydrophobic Tethers: 10 Constant 0.25

**Unfolding Nuclei Settings:**

Unfolding Nuclei Types: 11 ☒ Type 1 ☒ Type 2 ☒ Type 3 ☒ Type 4

Determine transition point from a CCE2 Sigmoid fit

CCE2 Sigmoid 12

**Output Settings:**

☒ Write stability maps 13 ☐ Write ligand

☐ Write indices for each structure ☐ Write FIRST Network

/home/mulnaes/src/VisualCNA/ Output 14

RUN CNA 15

Quit VisualCNA

**Figure 1. Setup panel**



### 8.3.2 Analyzing CNA data with VisualCNA (*Analyze* panel, Figure 2)

(See video tutorial for this section here: <https://www.youtube.com/watch?v=ztan2CU-Ioo> )

If a CNA run is finished, you can click *Yes* when prompted if you want to visualize the results.

❖ Load data from directory

- **Select the directory where CNA results are located by clicking *Data Directory* [1].**
- Select if data should be appended or should overwrite existing graphs using *Add to Plots* [9].
- **Click *Visualize Data* [10].**

❖ Load data from individual files

- Select the structure input type (from PDB file or PyMOL) with the *File* drop-down next to *PDB File* [2].
- **Select the structure input file by clicking *PDB File* [3].**
- Select the network input type (from PDB or Network File) with the *File* drop-down next to *Network* [4].
- **Select the network input file by clicking *Network* [5].**
- Select and append the results file with the respective file browser button “...” [6].
- Select if data should be appended or should overwrite existing graphs using *Add to Plots* [9].
- **In the case of new structure and/or network, click *Visualize Data* [10].**

❖ Save or change graph views

- Select or deselect files to be shown using the respective *Files* drop-down [7].
- Select or deselect the types of data shown using the respective *Types* drop-down [8]. A detailed description of each index type can be found [here](#).
- Change the state of the unfolding trajectory using the slider button [11].
- Save a graph to an image file using the respective save graph button [12].
- Pan or zoom on the graph using the respective pan [13] and zoom [14] buttons.
- Return the graph view to the original using the home button [15]

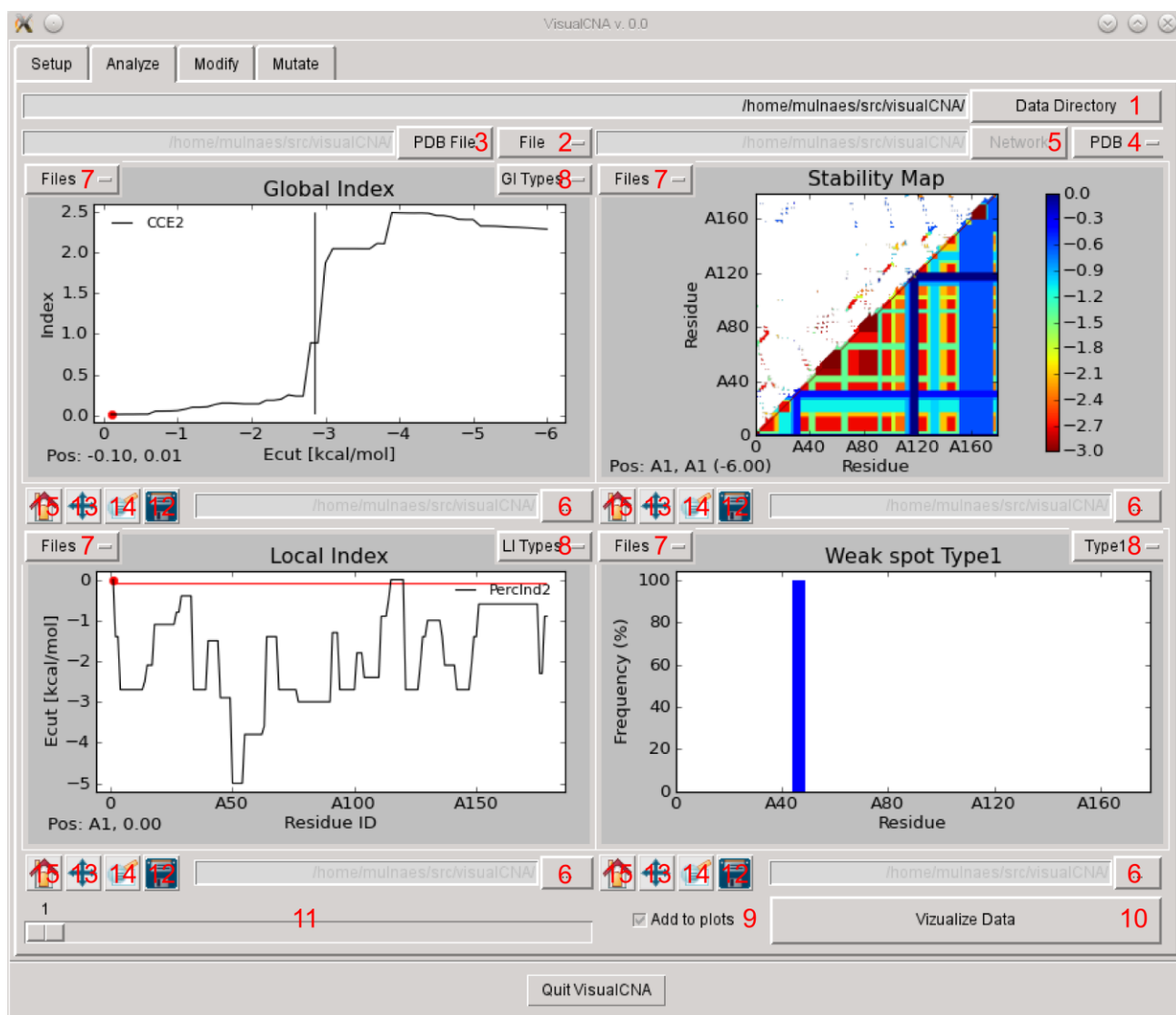


Figure 2. Analyze panel

### 8.3.3 Modifying CNA constraint network with VisualCNA (*Modify* panel, Figure 3)

(See video tutorial for this section here: <https://www.youtube.com/watch?v=BHkLXCZoVNU> )

#### ❖ Modify constraints from table

- **Select the constraint type in the table selector [1].**
- Search for a constraint by specifying a search term [9].
- Find the next [10] or previous [8] occurrence of the term.
- **Break or form constraint using check-boxes [11].**
- Double-click constraint to change PyMOL view to focus on that constraint [12].
- Change page of the table with the arrow buttons [13,14].

#### ❖ Add constraints from PyMOL

- **Select two atoms in PyMOL to add a constraint in between.**
- Select the constraint type in the constraint drop-down [2].
- Select the number of bars for the constraint in the bar drop-down [3].
- **Make the constraint by clicking *Make Constraint* [4].**

#### ❖ Delete constraints from PyMOL or browse through network changes

- **Select two atoms in PyMOL to remove a constraint in between.**
- **Delete the constraint by clicking *Delete Constraint* [6].**
- Undo or redo changes to the constraint network using *Undo Change* [6] or *Redo Change* [7].

#### ❖ Add constraints from text fields

- **Select atom ID pair to constrain [15,16].**
- Select the constraint type in the constraint drop-down [2].
- Select the number of bars for the constraint in the bar drop-down [17].
- Click *Add* to add the constraint to the Network [18].

#### ❖ Save Network to a file

- **Save a the current constraint network to a file using *Save Network* [19].**

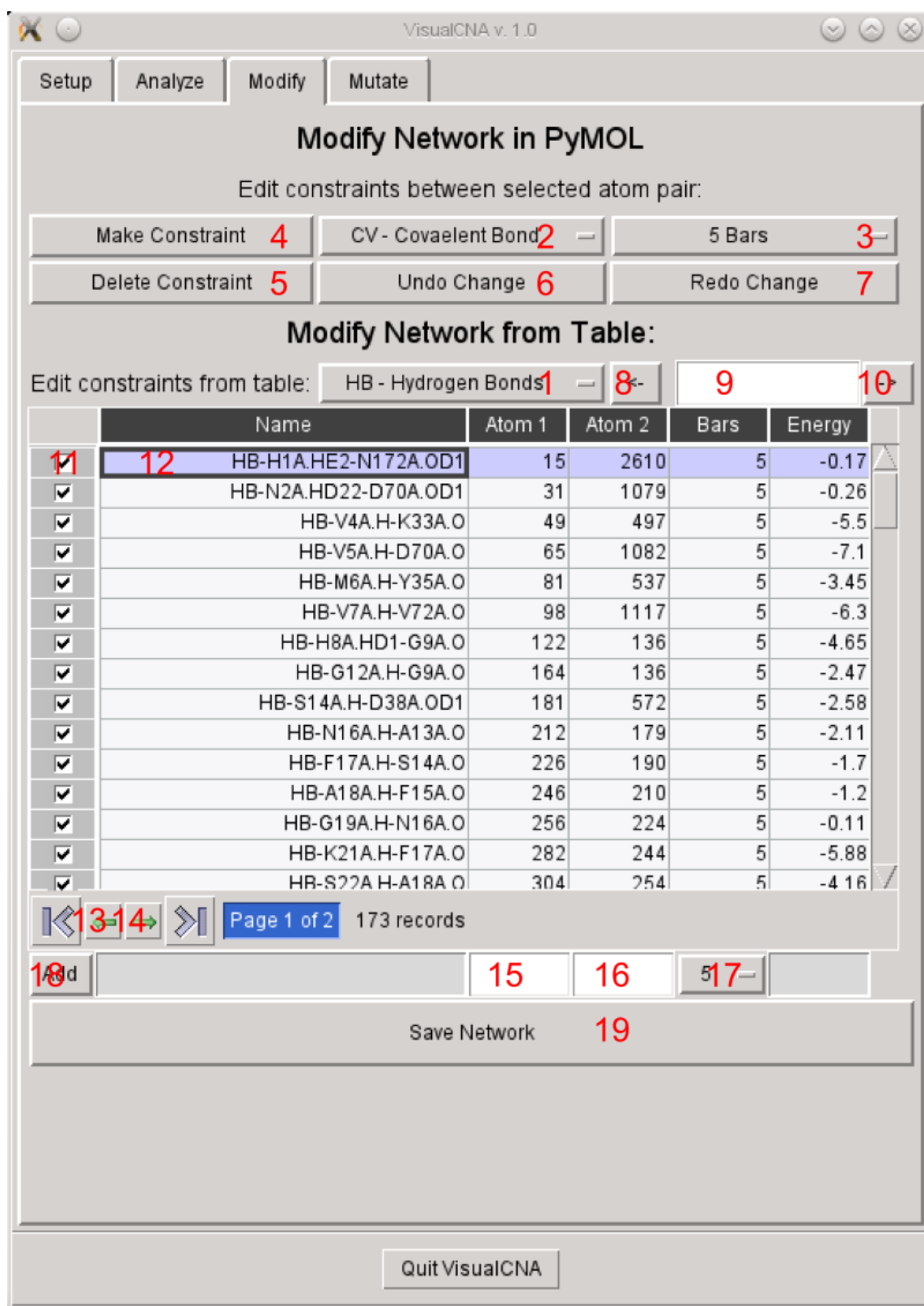
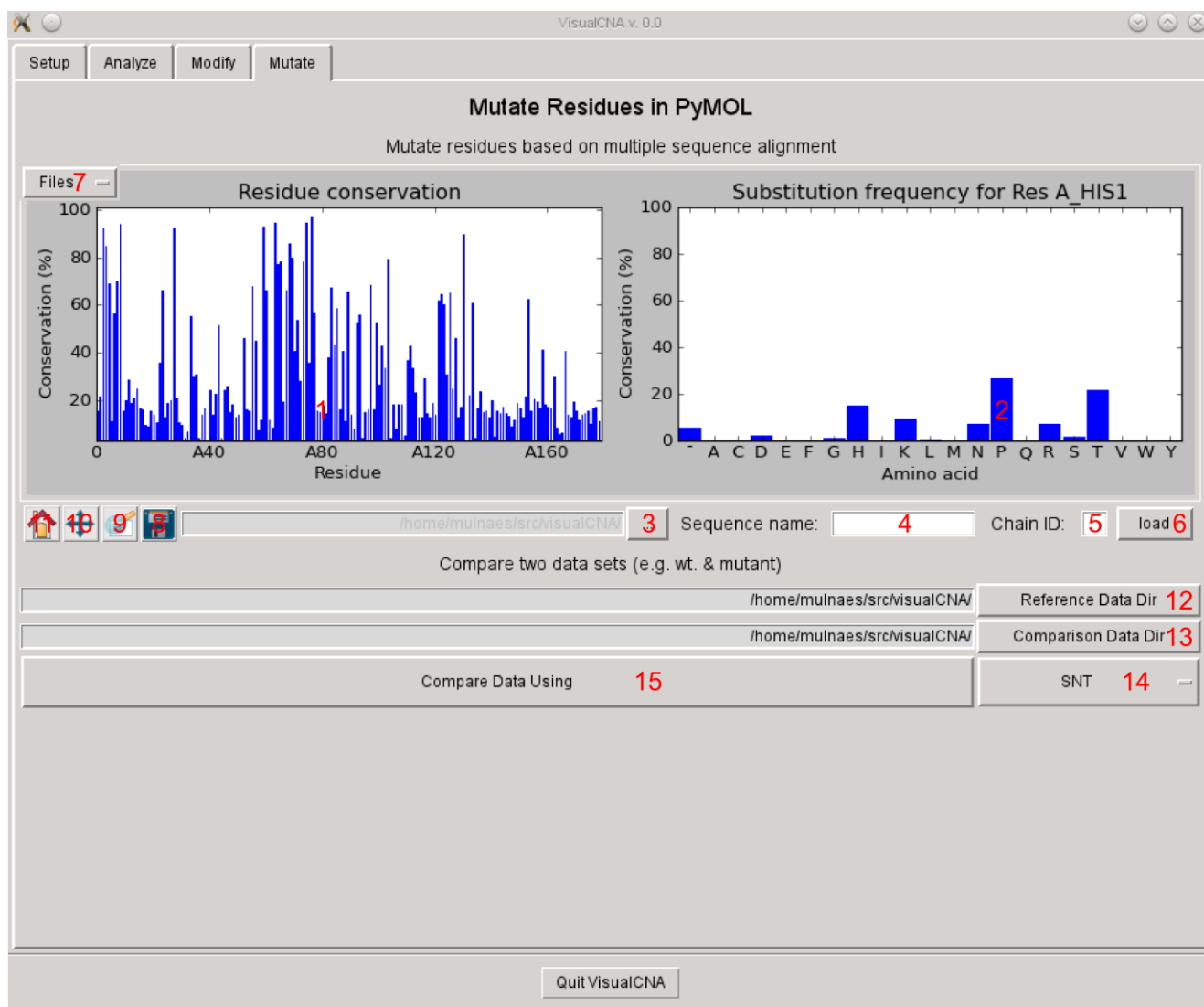


Figure 3. Modify panel

### 8.3.4 Modifying protein structure with VisualCNA (Mutate Panel, Figure 4)

(See video tutorial for this section here: <https://www.youtube.com/watch?v=cLmUCG-h6m4>)

- Mutate residues
  - Select the residue position in the *Residue Conservation* plot you wish to mutate [1].
  - Select the residue type in the *Substitution Frequency* plot to mutate that position to [2].
  - Select the side-chain conformation of the mutated residue by changing PyMOL state.
  - Click *OK* to mutate or *Cancel* to avoid mutating.
- Load a multiple sequence alignment
  - Select the multiple sequence alignment file with the file browser button “...” [3].
  - Specify the name [4] and chain ID [5] of the target sequence in the alignment file.
  - Load the multiple sequence alignment file using *load* [6].
  - Browse between multiple alignment files using the *Files* drop-down [7].
  - Save the plots to an image file using the save graph button [8].
  - Pan or zoom on the graph using the pan [10] and zoom [9] buttons.
  - Return the graph view to the original using the home button [11]
- Compare mutant and wild-type networks
  - Select data directories for reference [12] (wildtype) and comparison (mutant) [13] data sets.
  - Select the type of CNA analysis to be used for comparison.
  - Click *Compare Data Using* [15].
  - When prompted if you wish to see the comparison, select *Yes*.

**Figure 4. Mutate panel**

## 9 Citing CNA

When using the CNA software please reference:

- Pfleger, C., Rathi, P.C., Klein D.L., Radestock S., Gohlke H., J. Chem. Inf. Model. 2013, DOI: 10.1021/ci400044m.

For information on the global and local indices, please refer to / cite:

- Pfleger, C., Radestock, S., Schmidt, E., Gohlke, H., J. Comput. Chem. 2013, 34, 220-233.

For information on weak spots, please refer to / cite:

- Radestock, S., Gohlke, H. Eng. Life Sci. 2008, 8, 507-522.
- Rathi, P.C., Radestock, S., Gohlke, H., J. Biotechnology 2012, 159, 135-144.

For information on MD simulation derived ensemble-based CNA, please refer to / cite:

- Gohlke, H., Kuhn, L.A., Case, D.A., Proteins 2004, 56, 322-327.
- Rathi, P.C., Radestock, S., Gohlke, H., J. Biotechnology 2012, 159, 135-144.

For information on Fuzzy non-covalent constraint derived ensemble-based CNA, please refer to / cite:

- Pfleger, C., Gohlke, H., Structure 2013, DOI: 10.1016/j.str.2013.07.012.

When using VisualCNA, please refer to/cite:

- Rathi P.C., Mulnaes D., and Gohlke H. Submitted 2014.

## 10 Third party licensing

CNA owes its existence to several excellent open source packages. This section is in the documentation in order to comply with licensing requirements of these packages.

### 10.1 Python

#### PSF LICENSE AGREEMENT FOR PYTHON 2.7.2

1. This LICENSE AGREEMENT is between the Python Software Foundation (“PSF”), and the Individual or Organization (“Licensee”) accessing and otherwise using Python 2.7.2 software in source or binary form and its associated documentation.
2. Subject to the terms and conditions of this License Agreement, PSF hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 2.7.2 alone or in any derivative version, provided, however, that PSF’s License Agreement and PSF’s notice of copyright, i.e., “Copyright © 2001-2010 Python Software Foundation; All Rights Reserved” are retained in Python 2.7.2 alone or in any derivative version prepared by Licensee.
3. In the event Licensee prepares a derivative work that is based on or incorporates Python 2.7.2 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 2.7.2.
4. PSF is making Python 2.7.2 available to Licensee on an “AS IS” basis. PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 2.7.2 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 2.7.2 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 2.7.2, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.



7. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between PSF and Licensee. This License Agreement does not grant permission to use PSF trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.
8. By copying, installing or otherwise using Python 2.7.2, Licensee agrees to be bound by the terms and conditions of this License Agreement.

## 10.2 BioPython

CNA uses the PDB package from Biopython 1.58.

### Biopython License Agreement

Permission to use, copy, modify, and distribute this software and its documentation with or without modifications and for any purpose and without fee is hereby granted, provided that any copyright notices appear in all copies and that both those copyright notices and this permission notice appear in supporting documentation, and that the names of the contributors or copyright holders not be used in advertising or publicity pertaining to distribution of the software without specific prior permission.

THE CONTRIBUTORS AND COPYRIGHT HOLDERS OF THIS SOFTWARE DISCLAIM ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL THE CONTRIBUTORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

## 10.3 SciPy

Copyright © 2001, 2002 Enthought, Inc. All rights reserved.

Copyright © 2003-2009 SciPy Developers. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the Enthought nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## 10.4 NumPy

Copyright © 2005-2009, NumPy Developers. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the NumPy Developers nor the names of any contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## 10.5 SWIG

Simplified Wrapper and Interface Generator (SWIG)

Copyright © 1998 - 2002 University of Chicago. All rights reserved.

Copyright © 1995 - 1998 University of Utah and the Regents of the University of California.  
All rights reserved.

### LICENSE

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE UNIVERSITY OF CHICAGO AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF

MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE UNIVERSITY OF CHICAGO OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

IN NO EVENT SHALL THE AUTHOR, THE UNIVERSITY OF CALIFORNIA, THE UNIVERSITY OF UTAH OR DISTRIBUTORS OF THIS SOFTWARE BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE AUTHORS OR ANY OF THE ABOVE PARTIES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHOR, THE UNIVERSITY OF CALIFORNIA, AND THE UNIVERSITY OF UTAH SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

## **10.6 Open Babel**

The Open Babel is distributed under the GNU GENERAL PUBLIC LICENSE Version 2.

## **10.7 Matplotlib**

1. This LICENSE AGREEMENT is between John D. Hunter ("JDH"), and the Individual or Organization ("Licensee") accessing and otherwise using matplotlib software in source or binary form and its associated documentation.

2. Subject to the terms and conditions of this License Agreement, JDH hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use matplotlib 1.3.1 alone or in any derivative version, provided, however, that JDH's License Agreement and JDH's notice of copyright, i.e., "Copyright (c) 2002-2009 John D. Hunter; All Rights Reserved" are retained in matplotlib 1.3.1 alone or in any derivative version prepared by Licensee.

3. In the event Licensee prepares a derivative work that is based on or incorporates matplotlib 1.3.1 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to matplotlib 1.3.1.

4. JDH is making matplotlib 1.3.1 available to Licensee on an "AS IS" basis. JDH MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, JDH MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF MATPLOTLIB 1.3.1 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. JDH SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF MATPLOTLIB 1.3.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING MATPLOTLIB 1.3.1, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between JDH and Licensee. This License Agreement does not grant permission to use JDH trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.

8. By copying, installing or otherwise using matplotlib 1.3.1, Licensee agrees to be bound by the terms and conditions of this License Agreement.

## 10.8 Setuptools

The setuptools is delivered under Python Software Foundation (PSF) license that can be found here <http://opensource.org/licenses/Python-2.0>.

## 10.9 PyMol

### Open-Source PyMOL Copyright Notice

The Open-Source PyMOL source code is copyrighted, but you can freely use and copy it as long as you don't change or remove any of the Copyright notices. The Open-Source PyMOL product is made available under the following open-source license terms:

Open-Source PyMOL is Copyright (C) Schrodinger, LLC. All Rights Reserved

Permission to use, copy, modify, distribute, and distribute modified versions of this software and its built-in documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appears in all copies and that both the copyright notice and this permission notice appear in supporting documentation, and that the name of Schrodinger, LLC not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

SCHRODINGER, LLC DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL SCHRODINGER, LLC BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

### PyMOL Trademark Notice

PyMOL(TM) is a trademark of Schrodinger, LLC. Derivative software which contains PyMOL source code must be plainly distinguished from any and all PyMOL products distributed by

Schrodinger, LLC in all publicity, advertising, and documentation. The slogans, "Includes PyMOL(TM).", "Based on PyMOL(TM) technology.", "Contains PyMOL(TM) source code.", and "Built using PyMOL(TM).", may be used in advertising, publicity, and documentation of derivative software provided that the notice, "PyMOL is a trademark of Schrodinger, LLC.", is included in a footnote or at the end of the document.

All other endorsements employing the PyMOL trademark require specific, written prior permission.

### **10.10 Tkintertable**

Tkintertable is distributed under GNU General Purpose License and can be found <http://www.gnu.org/licenses/gpl.html>.

## 11 References

1. Kruger, D. M.; Rathi, P. C.; Pflieger, C.; Gohlke, H. CNA web server: rigidity theory-based thermal unfolding simulations of proteins for linking structure, (thermo-)stability, and function. *Nucleic Acids Res.* **2013**, 41, W340-W348.
2. Pflieger, C.; Rathi, P. C.; Klein, D. L.; Radestock, S.; Gohlke, H. Constraint Network Analysis (CNA): A Python software package for efficiently linking biomacromolecular structure, flexibility, (thermo-)stability, and function. *J. Chem. Inf. Model.* **2013**, 53, 1007-1015.
3. Rathi, P. C.; Pflieger, C.; Fulle, S.; Klein, D. L.; Gohlke, H. Statics of biomacromolecules. In *Molecular Modeling*, Comba, P., Ed. Wiley-VCH: Weinheim, 2011; pp 281-299.
4. Jacobs, D. J.; Rader, A. J.; Kuhn, L. A.; Thorpe, M. F. Protein flexibility predictions using graph theory. *Proteins* **2001**, 44, 150-165.
5. Rathi, P. C.; Radestock, S.; Gohlke, H. Thermostabilizing mutations preferentially occur at structural weak spots with a high mutation ratio. *J. Biotechnol.* **2012**, 159, 135-144.
6. Radestock, S.; Gohlke, H. Exploiting the link between protein rigidity and thermostability for data-driven protein engineering. *Eng. Life Sci.* **2008**, 8, 507-522.
7. Radestock, S.; Gohlke, H. Protein rigidity and thermophilic adaptation. *Proteins* **2011**, 79, 1089-1108.
8. Pflieger, C.; Radestock, S.; Schmidt, E.; Gohlke, H. Global and local indices for characterizing biomolecular flexibility and rigidity. *J. Comp. Chem.* **2013**, 34, 220-233.
9. Pflieger, C.; Gohlke, H. Efficient and robust analysis of biomacromolecular flexibility using ensembles of network topologies based on fuzzy noncovalent constraints. *Structure* **2013**, DOI: 10.1016/j.str.2013.07.012.