SenseNet Manual

Version 1.1.1

Contents

1 Introduction

SenseNet ("Structure ENSEmble NETworks") maps structure ensembles of biomolecules to atom interaction networks and provides functions for their analysis and visualization. It is available as a plugin for the free network visualization software *Cytoscape* [1].

Protein structures are frequently analysed to gain insights into the effects of ligand binding, residue mutations or conformational changes. In contrast to isolated structures generated by crystallography or other experimental sources, ensembles obtained from Molecular Dynamics (MD) provide additional information such as interaction lifetimes or correlation between conformations, allowing to investigate dynamic properties of biomolecules. In a structure ensemble network, each node represents one or a group of atoms while edges correspond to the interactions between these atoms (e.g. hydrophobic contacts or hydrogen bonds). Each edge is associated with a 'timeline' which indicates the presence of an interaction for each structure in the ensemble. Analysis functions are provided to extract information from these timelines and map results to network nodes and edges. Finally, SenseNet offers comprehensive visualization functions for side-by-side analyses of networks and 3D structures.

2 Installation

The recommended method for obtaining SenseNet is the Cytoscape App Store or alternatively from our website at `https://bioinformatics.wzw.tum.de`. The latter option requires you to install the plugin manually. To do this, place the 'SenseNet' .jar file into the 'CytoscapeConfiguration/3/apps/installed' folder. Make sure to remove any old version of SenseNet from this folder before starting Cytoscape.

3 Data model

SenseNet maps protein or other macromolecular structures to a network of nodes, which correspond to individual atoms or groups of atoms, and edges, representing interactions between atoms. Atoms may be grouped into a 'metanode'; the individual atom nodes that are contained in this group are hence called its 'subnodes'. In a network, either a metanode or its subnodes may be 'active' (i.e. present) at a given time. This is achieved by expanding (replacing a metanode by its subnodes) or collapsing (replacing all subnodes by their metanode). The metanodes of a network can be expanded or collapsed by a double click, and all analysis methods which act on the 'active' set of nodes and edges take the current state of the network into account. For example, an analysis can be performed while some selected residues are expanded into individual atom nodes, while other residues are represented by a single node. When collapsing subnodes, all edges connecting to these nodes are replaced by metaedges which represent the cumulative interactions of the replaced edges. Separate metaedges are created for each interaction type (e.g. contacts or hydrogen bonds). A detailed description of edge sets and how they change can be found

in section 5.3.

In order to model a structure ensemble, each edge is associated with a timeline represented as either a vector of integer values (e.g. presence or absence of contacts in each time frame) or a vector of real numbers (e.g. interaction energies). The timeline of a metaedge is called a 'metatimeline' and is calculated from its subedge timelines. The 'Sum' and 'Occurrence' frame weight methods yield two different metatimeline variants

$$\begin{aligned} X_{ijk,sum} &= \sum_{\alpha \in i} \sum_{\beta \in j} X_{\alpha\beta k} \\ X_{ijk,occ} &= \min(1, X_{ijk,sum}) \end{aligned} \tag{1}$$

in which X corresponds to a timeline, i, j are metanodes, k is an interaction type and α, β are subnodes of i, j.

The weight of an edge describes the strength of an interaction. It is determined by the 'Timeline weight method', which is by default the average over all time frames. Alternatively, users can set the network to any single time frame or use averages of time blocks, e.g. as obtained from clustering.

Difference networks can be used to compare networks of two similar, but different ensembles (e.g. structures with one or more point mutations). A difference network is created by mapping interactions between equivalent atoms of two networks onto each other. Two atoms are considered equivalent if they have the same chain name, residue index, residue inset, residue alternative location and atom name (see PDB specification [2]). Notably, the residue name is not compared in order to allow comparisons for residue point mutations. All interactions between two equivalent atoms which have the same interaction type are considered equivalent. Interactions for which no equivalent can be found are compared to an empty timeline of all zeroes. Once all equivalent interactions are mapped, the timeline vectors are subtracted element-wise

$$X_{\alpha\beta k,diff} = X_{\alpha\beta k} - X_{\alpha\beta k,ref} \tag{2}$$

where $X_{\alpha\beta k}$ is the timeline of the compared network, and $X_{\alpha\beta k,ref}$ is the timeline of the reference network. Metatimelines are calculated analogously to eq. 1.

4 User Guide

All plugin functions can be accessed via the Cytoscape GUI. The controlling elements can be found either in the 'SenseNet' tab on in the control panel or in the top menu within 'Apps - SenseNet'.

4.1 General

Import network *Parameters*

- **Import networks** Choose one or more input sources to import. Use the plus and minus buttons to add/remove fields. See also section 5.4 for more details on some of the file formats.
  - **Input type**
    - **AIF file** Import network from an AIF (Atom Interaction Format) .aif/.zaif file.
      - **.aif/.zaif file** Input file: Either in AIF or compressed ZAIF format.
      - **Frame sieve** Read only every nth frame. Useful to save memory.
      - **Skip timelines** Skip timeline depending on a threshold. For example the 'Skip timelines $<$ 0.05 avg.' option does not import a timeline if its average is below 0.05.
    - **CPPTRAJ H-bonds** Import network from CPPTRAJ [3] hbond command output.
      - **H-bond file** Generated by 'avgout'.
      - **Timeline file** Generated by 'uuseries'.
      - **Interaction type** Interaction type name.
      - **Ignore backbone** Choose to ignore contacts involving backbone atoms (atom names C,O,N,CA).
      - **Frame sieve** Same as for 'AIF file'.
      - **Skip timelines** Same as for 'AIF file'.
    - **CPPTRAJ nativecontacts** Import network from CPPTRAJ nativecontacts output.
      - **Contacts file** Generated by 'writecontacts'.
      - **Native timeline file** Generated by 'seriesout'.
      - **Nonnative timeline file** Generated by 'seriesnnout'.
      - **Contacts .pdb file** Generated by 'contactpdb'.
      - **Interaction type** Interaction type name.

        - **Ignore backbone** Choose to ignore contacts involving backbone atoms (atom names C,O,N,CA).
        - **Ignore intra-residue** Choose to ignore contacts within the same residue.
        - **Frame sieve** Same as for 'AIF file'.
        - **Skip timelines** Same as for 'AIF file'.
    * **PDB structure H-bonds** Import network from H-bonds found in a PDB file. Hydrogens are required to find H-bonds. All atoms in the PDB file whose names start with 'H' are considered hydrogens.
        - **PDB file** Select .pdb file to load.
        - **Distance cut-off** Maximum donor-acceptor distance.
        - **Angle cut-off** Minimum donor-hydrogen-acceptor angle.
        - **Donor mask** Atom mask for donor atoms (see mask reference in section 5.2).
        - **Acceptor mask** Atom mask for acceptor atoms (see mask reference in section 5.2).
        - **Interaction type** Interaction type name.
    * **PDB structure contacts** Import network from contacts found in a PDB file.
        - **PDB file** Select .pdb file to load.
        - **Distance cut-off** Maximum contact distance.
        - **Atom mask** Atom name mask to calculate contacts for (see mask reference in section 5.2).
        - **Interaction type** Interaction type name.
        - **Ignore backbone** Choose to ignore contacts involving backbone atoms (atom names C,O,N,CA).
        - **Ignore intra-residue** Choose to ignore contacts within the same residue.
    * **DSSP secondary structure** Import network of secondary structure elements. All residues belonging to the same secondary structure element (helix/sheet) are connected sequentially.
    * **DSSP file** Select .dssp file to load.
    * **Interaction type** Interaction type name.
- **Difference network** Check box to select input files for creating a difference network. When checked, the 'Import reference networks' panel will appear. The differences are calculated by subtracting the timelines of equivalent interactions of the reference network from the network loaded in the top import panel (**Import networks**).

- **Metanode definition** Choose how atom nodes are grouped together.
  - **Group definition** Grouping settings.
    - **Amino acids** Group atoms by their respective amino acids
    - **Backbone**/**Sidechain** Group atoms by their respective amino acids, but create seperate nodes for the backbone/sidechain portions. The **Backbone atom names** text field contains all atom names that will be categorized as backbone, separated by commas.
- **Network options**
  - **Create visual style** Check to automatically create a new visual style for the network.
  - **Network name** Displayed name of the network. Automatically filled when imported files are chosen. Can also be set manually.
  - **Remove edges** Setting to remove (deactivate) edges below a certain threshold. Equivalent to the setting with the same name in the 'Interaction weights' tab. Can be changed at any time.

Show log Show task logs.

Parameters

- **Log category** Select log category to display.
  - **Global** Continuous list of task logs since session start.
  - **Task** Last log of task type selected in the **Log type** box.

Export network Export current network in AIF format. The exported file can e.g. be used for importing a multisource network as a single file containing the combined information of all original import files.

4.2 Shown interactions

Only interaction types with checked boxes are shown in the network. The interaction type of each edge is read from the **shared interaction** column.

4.3 Interaction weights

This panel controls how individual atom timelines are combined into metatimelines. The radio buttons on the left chooses how each frame of the metatimeline is determined ('Frame weight method'; see eq. 1). The buttons on the right select how to determine the total weight of the metatimeline ('Timeline weight method'):

- **Average weight** Average over all frames in the timeline
- **Single frame** Select and show the network state at an individual time frame

- **Clusters** Show average of frames within a cluster. Requires previous clustering analysis

Weighting is performed on all imported edges (see section 5.3 for an explanation of edge sets). For metaedges, all subedges of the same interaction type are considered. Whenever weighting is performed, the results are written into the 'weight' and 'standard deviation' edge columns.

4.4 Analysis

4.4.1 Network interactions

The analysis functions in this panel act on the whole network. Results are usually presented as tables/plots in the result panel on the right.

Timeline Shows the timelines for all imported edges. For metaedges, the metatimeline is calculated according to the selected weight method.

Parameters

- **Frame weight** Method for determining metatimelines. See section 4.3

Degree Calculates weighted degree for active edges. Results are written into the 'degree' node table column.

Parameters

- **Degree weight** Method for calculating weights for adjacent edges.
  - **Edge weight sum** Sum values of edge columns.
- **Weight column** Source column for edge weights.
- **Negative weights** Method to treat negative edge weights.
  - **abs(x)** Use absolute value of x.

Centrality Calculate weighted centrality measures for active nodes, based on shortest paths. The algorithms are implemented as described in refs. [4, 5]. If two nodes are connected by multiple edges, they are treated as one, using one of several merging methods. Results are written into the 'centrality' node table column.

Parameters

- **Centrality type** Centrality measures to compute. A transformation function maps edge weights to determine the distance between nodes (see algorithm 10 in ref. [4])
- **Multiple edges weights** Method for merging parallel edges between two nodes.
  - **Sum/Min/Max** Use the weight sum/average/min/max of parallel edges as total weight.
  - **Edge count** Use the number of edges between two node pairs as total weight.

  - **Uniform** All edges are assigned an uniform weight. Parallel edges are ignored. This option effectively gives the centralities of an unweighted network.
- **Weight column** Column of edge weights.
- **Distance transformation** Function to transform edge weights to distances. The function is commonly chosen such that a high weight corresponds to a low distance.
- **Negative weights** Method to treat negative edge weights.
- **Normalization** Type of normalization to apply to centrality values.
  - **None** Do not perform normalization.
  - **Min-max range** Subtract the minimum centrality of the network from each value and divide by the range of values. The normalized value is limited between 0 (lowest centrality) and 1 (highest centrality).
  - **Max node pairs** Divide each node's centrality by the theoretical maximum number of node pairs, excluding that node, in an undirected network: $\frac{(N-1)(N-2)}{2}$, where N is the total number of active nodes.

Correlation Determine correlation between **active** edges. The edge neighbour correlation factor ECF is calculated as

$$\mathrm{ECF(i)} = \sum_{j \in A} |c(i,j)| \tag{3}$$

where i, j are network edges, $c(i,j)$ is a correlation function of edges i and j, and A is the set of edges that i is compared to. Correlation factor types may use different edge sets for A.

Parameters

- **Correlation factor type**
  - **Neighbour** Edge neighbour correlation factor. Here, A is the set of edges adjacent to i (i.e. separated by at most one node)
- **Correlation method** Correlation measure to use (see text below).
- **Frame weight** Method for determining metatimelines. See section 4.3.
- **Reference network** For methods calculating correlation differences.
- **Edge mapping** Determine how edges are matched between active and reference networks.
  - **Shared name** Match edges with identical 'shared name' entries in the edge table.
  - **Match location** Require that the 'shared name' entries of source and target nodes approximately match between networks. The approximation is that only the 'residue name' is allowed to differ. The

'altloc' and 'residue insert' identifiers are not considered part of the residue name and thus still have to be identical. This option is intended for comparing two networks differing e.g. in point mutations of single residues in order to match edges representing conserved interactions.

The 'Mutual information' correlation method determines correlation as

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \left(\frac{p(x,y)}{p(x)p(y)} \right) \tag{4}$$

in which, X and Y are **integer** timelines, $p(x,y)$ is the joint probability function of values x, y, and $p(x), p(y)$ are marginal probability functions of x, y. This method should be used when the timeline contains discrete values, such as the count of interactions. The unit for mutual information results is 'bits'.

'Mutual information difference' calculates the sum of absolute changes in expected pointwise mutual information for each event

$$\begin{aligned} I(X;Y) = \sum_{x \in (X \bigcup \hat{X})} \sum_{y \in (Y \bigcup \hat{Y})} \Big| & \, p(x,y) \log_2 \left(\frac{p(x,y)}{p(x)p(y)} \right) \\ & - \hat{p}(x,y) \log_2 \left(\frac{\hat{p}(x,y)}{\hat{p}(x)\hat{p}(y)} \right) \Big| \end{aligned} \tag{5}$$

where the $\hat{X}, \hat{Y}$ denote timelines in the reference network corresponding to X, Y. Edges are considered equivalent if their 'shared name' columns match. If no match can be found for an edge, the reference timeline is replaced by a vector of zeroes. This measure is useful for determining differences in dynamic behaviours between simulations, f.e. a protein with and without a ligand.

Alternatively, choosing the 'Pearson' correlation method will calculate the Pearson correlation coefficient between interaction timelines

$$r = \frac{\sum_i (X(i) - \mu_x)(Y(i) - \mu_y)}{\sigma_x \sigma_y} \tag{6}$$

where i is a discrete time frame, $X(i), Y(i)$ are functions yielding the corresponding timeline value at position i, with associated sample means μ and standard deviations σ. This method is recommended when the timeline contains continuous values like interaction energies.

The resulting correlation factors for each edge are written into the 'correlation factor' column.

Lifetime Calculates estimates for the interaction lifetimes. These are calculated from the intermittent autocorrelation function

$$C_L(k) = \frac{1}{N}\sum_{i=0}^{N-k}\frac{X_{occ}(i)X_{occ}(i+k)}{X_{occ}(i)^2} \tag{7}$$

in which k is the discrete lag step, N is the total number of time frames and $X_{occ}(i)$ is the occurrence weighted metatimeline function. The calculated lifetime is intermittent, i.e. the interaction may break and reform between the compared time frames. The average lifetime is estimated from the autocorrelations following the same protocol as for the autocorrelation time during error estimation.

If the 'Replicas' setting is set to $n > 1$, the timeline is divided into n equal sized blocks, which are analyzed separately. The average of block lifetimes is written to the 'lifetime' edge column.

Weight error Calculates weight error estimates for all imported edges. These functions aim at approximating the standard error of the timeline weight.

Parameters

- **Error method** Method for calculating the error. Currently, only 'Autocorrelation' is available.
- **Frame weight** Method for determining metatimelines. See section 4.3.
- **Replica weight** Method to merge errors from multiple replicas.
  - **Max/Avg/Min** Use maximum, average or minimum of replicas as final error value.

The autocorrelation method is based on the approach outlined in ref. [6]. The timeline autocorrelation for different discrete lag steps is calculated as

$$C(k) = \frac{1}{N\sigma^2}\sum_{i=0}^{N-k}(X(i)-\mu)(X(i+k)-\mu) \tag{8}$$

with k as the discrete lag step, N as the total number of time frames, $X(i)$ as the timeline function at each discrete time frame, μ as the mean of X, and σ^2 as the variance of X. A single exponential of the form $A * e^{Bx}$ is fitted by weighted linear regression in log space with weights $\frac{1}{k+1}$. To reduce the influence of noisy function tails, only autocorrelation values above 0.1 are considered for fitting. The integral of the fitted exponential is calculated analytically and serves as estimate for the autocorrelation time τ. The estimate for the independent sample size is then determined as

$$N_{ind} = \frac{N}{\tau} \tag{9}$$

which is used to estimate the standard error of independent samples

$$\sigma_{e,ind} = \frac{\sigma}{\sqrt{N_{ind}}} \tag{10}$$

If the 'Replicas' setting is set to $n > 1$, the timeline is divided into n equal sized blocks, which are analyzed separately. The final error is determined as by the 'Replica weight method'.

The results of this analysis are written into the 'error estimate', 'autocorrelation sample size', and 'autocorrelation time' edge columns.

Entropy For each network edge, determine Shannon's information entropy

$$H(X) = -\sum_{x \in X} p(x) \log_2(p(x)) \tag{11}$$

where X is the edge's integer interaction timeline. Results are written into the 'entropy' edge column. The unit of reported results is 'bits'.

Parameters

- **Frame weight** Method for determining metatimelines. See section 4.3.

Random Walk Perform a random walk through the network. At each step, the next visited node is selected randomly from the list of nodes connected to the current node via one or more edges (neighbors). Multiple edges between node pairs do not affect the selection probability.

Parameters

- **Walk mode**
  - **Default** Regular random walk starting from the specified node and finishing after the specified number of steps, tracking visited nodes along the way. Restarts do not reset the list of visited nodes.
  - **Targeted** Random walk starting and finishing at the specified nodes. If the target node is not reached after the specified number of steps, the run is restarted. Restarts always reset the list of visited nodes.
  - **Targeted-Symmetric** Like 'Targeted', but each run is performed an additional second time after exchanging start and target nodes. The total number of runs is doubled.
- **Weighting mode**
  - **Unweighted** The next node is selected from the list of neighbors using a uniform probability distribution.
  - **Weighted** The next node is selected using a weighted probability factor obtained from a node table value specified by 'Weight Column'. The probability of a node neighbor to be selected for the next step is then

    $$p(i) = \frac{w(i)}{\sum_{n \in N} w(n)} \tag{12}$$

    where i is the neighbor candidate from the node neighbor list N and $w(i)$ is the weight obtained from the 'Weight Column' for node i.

- **Weight column** Column used for obtaining weights and calculate weighted probabilities.
- **Max steps** Maximum number of walk steps in a single run.
- **Restart probability** Probability of restarting the run at each step. The list of visited nodes may be retained or cleared, depending on the 'Walk mode'.
- **Num runs** Number of independent runs to be performed. Nodes visited multipled times during one individual run are counted only once. Individual runs are added up to yield the final result.
- **Random seed** Seed value for random generator. Choosing the same seed guruantees reproduction of a specific analysis outcome, provided that the network and all other parameters are the same.

The number of random walk visits for each node are written into the 'visited' edge column.

4.4.2 Selected interactions

Show detailed analyses for one or more selected edges.

Timeline Equivalent to the corresponding function in section 4.4.1, but shows a plot of the timeline using the currently active frame weight method.

Correlation Calculates correlation measures between a single selected edge and all other imported edges (see section 4.4.1).

Autocorrelation Equivalent to the corresponding function in section 4.4.1. Uses the currently active frame weight method and plots autocorrelation functions for each selected edge.

Blocked error Plots the blocked standard errors following a procedure from ref. [6]. Uses the currently active frame weight method.

In essence, the standard error (see eq. 10) is calculated multiple times between blocks of time frames. For the minimum block size of 1, the result is the conventional standard error. For a block size of 2, each time frame is averaged with its successor yielding $\frac{n}{2}$ non-overlapping 'block averages', where n is the total number of time frames. The standard error is then calculated between these block averages as if each was an independent data point. This is repeated for increasing block sizes. The maximum block size is set as $\frac{n}{4r}$, where n is the total number of time frames and r is the number of replicas. The total standard error can be estimated from the plot as the value to which the blocked standard errors converge.

4.4.3 Network matrix

These functions allow plotting and exporting networks in matrix form.

Show Display matrix of active network edges as a dotplot.

Parameters

- **Weight column** Select column to use for weighting.
- **Node index column** Select column to use as indices for the X and Y axes of the plot. Only columns of integers are allowed.
- **Min/Max value** Set min and max value for weight color scale.

Export Export matrix of active network edges.

Parameters

- **Weight column** Select column to use for weighting.
- **Node name column** Select column to use as names for the X and Y axes of the plot.
- **Output file** File to write matrix to.

4.4.4 Paths

The following functions provide functions for the identification of pathways between two selected nodes.

Shortest paths Find all shortest paths by traversing active edges between two selected nodes, starting from the node that was selected first. Edges are considered to represent equal distances, and parallel edges are ignored. The paths are presented in a table in the result panel. In addition, two measures of interaction strength are shown: The 'timeline sum' is the sum of average interactions of edges contributing to the pathway. In contrast, 'timeline occurrence' gives the average occurrence weights along that path. If multiple edges are present between two nodes, they are treated as if their timelines were merged.

Suboptimal paths Find all paths of a fixed length range between two nodes. Only active edges are considered and all edges are assumed to represent the same distance. Minimum and maximum path lengths can be set as parameters. Otherwise, output is equivalent to the 'Shortest paths' function.

4.4.5 Clustering

From this panel, functions for clustering of time frames can be accessed.

Cluster Start a new clustering run, grouping time frames with similar network states until a limit is reached. For this purpose, each time frame in the network is represented by an interaction matrix of dimension $N \times N$, where N is the number of nodes. Each entry in the matrix corresponds to the selected weight. If multiple edges are present between two nodes, their weights are summed. The distance between two time frames is calculated as the Frobenius norm of their matrix differences. This set of distances is then used for clustering.

Parameters

- **Clustering method**
  - **Agglomerative** Hierarchical agglomerative clustering.
  - **Linkage** Select linkage mode for agglomerative clustering.
- **Target cluster count/Epsilon** Select to stop clustering either at N clusters or when the minimum intercluster distance drops below a certain limit.
- **Sieve** Select to use only every Nth frame for clustering.
- **Frame weight** Select weight mode for the frame interaction matrix. See section 4.3.

4.5 Structure visualization

This panel contains functions to connect Cytoscape to a 3D structure viewer session. Networks can be linked to structures or trajectories loaded in the viewer. Node and edge selections in a linked network are highlighted in the structure.

Connect viewer Start a structure viewer and connect to it.

Parameters

- **Viewer** Select one of the available viewers to start (VMD, PyMOL or UCSF Chimera). See section 2 for viewer installation requirements.
- **Load session** Select a session file for the viewer to load after starting.

Model link Shows whether the currently focused network is linked to a structure.

Link network Link current network to a structure in the connected viewer.

Parameters

- **Single structure** Load a single structure into the viewer and link the network to it.
  - **Structure file** Select file to load structure from.
  - **Format** Automatically determined by structure file extension. Can be set manually. Available options depend on the connected viewer.
  - **Model name** Model name to use for the structure. Must be unique.
- **Trajectory** Load a trajectory of structures into the viewer and link the network to it.
  - **Structure file** Select file to load structure or topology from.
  - **Format** same as in 'Single structure'. Depending on the connected viewer, topology formats are accepted as well.
  - **Model name** same as in 'Single structure'.

  - **Trajectory file** Select file to load trajectory from.
  - **Trajectory format** same as 'Format', but for trajectories.
- **Preloaded** Link network to a structure already present in the viewer
  - **Model name** Select model name in structure viewer to link network to.

Unlink network Remove structure link of the current network. An option is given to remove the structure from the viewer as well.

Pause link Temporarily disable structure link of the current network. Until the link is unpaused, selection changes are not updated between network and structure viewer.

Transfer colors Color linked structure according to node colors in the network. For VMD and UCSF Chimera, the visually closest color of the defined color set is used. For PyMOL, the colors are transferred exactly as shown in the network.

4.6 Style

4.6.1 Node style

Auto style Map continuous node attributes to visual style. Creates a copy of the current style with a '_auto' suffix. If a style with that name already exists, it will be overridden.

Parameters

- **Style property** Select visual property to map values to. Different style settings are available for each property.
  - **Color** Map node attribute to fill color.
    - **Min/Mid/Max value** Minimum/Middle/Maximum value to map. Default values are determined automatically according the range of values in the network for the selected column.
    - **Low/Mid/High color** Color gradient from low to high.
  - **Size** Map node attribute to node size.
  - **Min/Max value** see 'Color'.
  - **Min/Max size** Minimum/Maximum node size.
- **Column** Column to map values from. All columns containing 'Double' values can be chosen.

Label format Select node label style. See section 5.1 for an explanation of naming conventions.

Renumber Renumber residue indices in labels.

Parameters

- **Chain** Chain(s) to renumber. Select one character ('A','B', etc.) or '*' to select all chains.
- **First residue index** First residue index to renumber.
- **Last residue index** Last residue index to renumber or '-1' to select up until and including the last residue index of selected chain(s).
- **Offset** Offset to add to the selected residues. Can be positive or negative.

Note that renumbering always acts on the **original residue numbering**. Therefore, if you renumbered residue 1 to become residue 100, you would have to select residue index 1 again to renumber it a second time. In order to return all numberings to the original imported state, you can use the 'Reset numbering' button in the dialog.

Labeling and renumbering never changes the 'shared name' or 'residue index' columns. Instead, results are written into the 'label' and 'residue index label' columns in the node table. This is done to avoid accidental ambiguity and loss of data. See section 5.1 for an explanation of the underlying concepts.

4.6.2 Edge style

Auto style Analogous to the corresponding function in section 4.6.1.

4.7 Settings

Structure viewer Configuration for structure viewers

- **PyMOL/VMD/UCSF Chimera location** Location to search for respective executables. If no location is given, the plugin will attempt to run the displayed command from the operating system's PATH variable
- **Max shown residues** Maximum number of residues to show as sticks before an error is thrown
- **Max shown interactions** Maximum number of interactions highlighted before an error is thrown
- **Selected interaction color** Highlight color for selected interactions
- **Selected residue color** Highlight color for selected residues
- **Zoom to selection** Check to enable auto-zoom to selected residues

5 Concepts

5.1 Labels and identifiers

Within the plugin, certain naming conventions are used to map nodes and edges to their structural counterparts. The standard naming style for nodes is

<Chain>/<Altloc><Residue name><Residue insert>
-<Residue index>(-<Mutated residue name>)(:<Atom name>)
(#<Group tag>)

The standard naming style for edges is

<Node name 1>_<Bridge name>_<Node name 2>
_<Interaction type>

The data fields correspond to the RCSB PDB [2] standard[1]. 'Group tag' is used when a non-amino acid metanode definition is used (such as backbone / side-chain). 'Bridge name' is used for interactions that involve more than two atoms, for example the name of the hydrogen in a hydrogen bond interaction. For amino acids, three-letter codes are used for residue names, but longer names are possible. Empty fields are allowed. Field separators enclosed in parentheses will only appear if the corresponding field is filled. For edges with a bridge atom, node names 1 and 2 are the source and target nodes, respectively. If no bridge atom is present (e.g. in metaedges), the node names appear in alphabetical order. This is done to ensure that edge names are predictable for symmetric interactions. the An example for a standard node label is 'A/TYR-290', and for a standard edge label it is 'A/TYR-290__A/VAL-80_H-bond'.

A node or edge 'name' (as found in the 'shared name' or 'name' columns) always follows the standard naming style, is assigned at import and is never changed. 'Names' are meant to identify edges as uniquely as possible, but **there is no guarantee that a name is unique in the network**. Node label customization functions create 'labels', which are allowed to omit information (such as chain names). Edge labels are automatically updated accordingly to the node labels. Labeling functions always write into separate 'label' columns.

5.2 Atom masks

When importing networks using the plugin, sometimes a selection of atom names is needed. Whenever an 'atom mask' is required, a Java regular expression[2] needs to be provided. For most purposes, a tiny subset of the regular expression language is sufficient for a proper selection. Examples of often used patterns are

- **.*** All atom names (. = any character, * = zero or more repetitions of the preceding character)
- **C.*** Atom names starting with C
- **F.*|O.*|N.*** Atom names starting with F,O, or N (| = 'or')

5.3 Network operations

The number of nodes and edges currently present can change due to filtering or expanding/collapsing of metanodes. Hence, it is important for each function that deals with edges to define on which set it operates. The sets are defined as follows:

[1] http://www.wwpdb.org/documentation/file-format-content/format33/v3.3.html
[2] https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html

- **Active nodes/edges** All edges that are currently present in the network. This includes nodes that are present, but hidden from view (f.e. using Cytoscape's 'hide nodes' feature). A node or edge that becomes inactive is temporarily deleted from the network. When an inactive node/edge becomes active, it is restored.
- **Imported nodes/edges** All nodes/edges from the original import, regardless of whether they are active or not.

The following operations can render a node/edge active or inactive:

- **Expanding/Collapsing a metanode** The metanode and associated metaedges are deactivated. Subnodes and associated edges are activated.
- **Edge weight filtering** Edges which are filtered out due to low weights are deactivated.
- **Subnetwork creation** When subnetworks are created from selected nodes, the selected nodes become the active node set for that network.
- **Manual deletion** Manually deleting nodes/edges f.e. using the 'DEL' key are deactivated, but can be reactivated f.e. when weight filtering is updated or a metanode is expanded/collapsed.

Generally, **the set of active nodes and edges is equivalent to the currently visible network**. Nodes and edges that are only hidden due to visualization (such as using Cytoscape's 'hide nodes/edges' feature) are still considered 'active' and part of the network. Inactive nodes are temporarily deleted from the network and are hence invisible both in the presentation and for network analysis algorithms. Hence, tools like NetworkAnalyzer or other analysis plugins can be used normally.

5.4 AIF file format

The AIF (Atom Interaction Format) was created as a convenient way to define interaction networks based on timeline data. All lines start with a record indicator, followed by one or more comma-separated data fields. Interaction data may be given as TIMELINE and DIFFERENCE_TIMELINE records, which define the following fields

- interaction type (string)
- source atomname (string)
- target atomname (string)
- source residue index (int)
- target residue index (int)
- source residue name (string)
- target residue name (string)
- source residue insert (string)
- target residue insert (string)

- source altloc (string)
- target altloc (string)
- source chain (string)
- target chain (string)
- bridge names (whitespace delimited list of strings)
- timeline (whitespace delimited list of ints/floats)

An AIF file must have the following properties:

- Lines are separated by Unix style newline ('Linefeed') characters.
- A line consists of one or more fields, which are separated by a comma. Leading and trailing whitespace are ignored for each field. Fields may be empty.
- The first field of each line denotes the record type. The record type defines how many fields follow in the same line and what their field data types are. The record type is case insensitive.
- Lines starting with a '#' character indicate comment lines and should be ignored by parsers.
- Empty lines or lines that contain only whitespace should be ignored by parsers.
- Tabs count as regular characters (not whitespace) and should be avoided entirely.

In order to save disk space, AIF files may be zipped (.zaif).

6 External tools guide

The SenseNet plugin can interact with a number of programs. This section gives details on how to set up and use those programs together with the plugin.

6.1 Command line interface

In order to allow automatized workflows, SenseNet allows some of its functions to be called either via the Cytoscape automation console, script files or the CyREST interface. They fulfill the same purpose as their equally named GUI counterparts and are called by preceding them with the "sensenet" namespace tag (e.g. "senseset importAif"). In addition, functions provide documentation by using the "help" command (e.g. "help sensenet importAif"). Validity of input is checked after parsing the command and corresponding error messages will appear, spelling out problems and allowed input options.

6.2 Structure viewers

The plugin can interact with several structure viewers (PyMOL, VMD and UCSF Chimera) in order to map the network onto a molecule structure. In order to use the plugin together with one of these viewers, it is only necessary that the plugin

can start the viewer. By default, it will attempt to look up the installation location from the operating system's PATH variable (equivalent to typing 'pymol','vmd' or 'chimera' on the command line). Alternatively, the installation location can be set manually in the 'Settings' menu. The viewer needs to be started using the 'Connect viewer' button in order to link a network.

The plugin was tested in combination with PyMOL 2.5.0, VMD 1.9.2 and UCSF Chimera 1.12.

6.3 CPPTRAJ

CPPTRAJ [3] is part of the AmberTools program suite and can be used to process and analyze molecular dynamics trajectories. The following sections describe how to use CPPTRAJ to write interaction timeline data, which can be used to create interaction networks. The scripts were tested with the CPPTRAJ version contained in AmberTools17.

6.3.1 nativecontacts

The following output files of the CPPTRAJ nativecontacts command are necessary for network import:

- **contacts.out** Contact table
- **contacts.series** Timeline series for native contacts
- **contacts.nonnative.series** Timeline series for non-native contacts
- **contacts.pdb** PDB file as output by nativecontacts

These files can be created using the following CPPTRAJ commands (adjust **bold** arguments as necessary)

```
parm md.prmtop
trajin md.nc
nativecontacts @C* distance 5.0 \
      writecontacts contacts.out contactpdb contacts.pdb \
      series seriesout contacts.series \
      savenonnative seriesnnout contacts.nonnative.series
run
```

6.3.2 hbond

The following output files of the CPPTRAJ hbond command are needed for importing a network:

- **hbonds.out** Interaction table
- **hbonds.series** Interaction timeline series

These files can be generated by the following CPPTRAJ commands (adjust **bold** arguments as necessary)

```
parm md.prmtop
trajin md.nc
hbond !(:WAT,Na+,Cl-) dist 3.5 angle 135 \
      avgout hbonds.out \
      series uuseries hbonds.series
run
```

7 Troubleshooting

7.1 Installation

- *Could not find plugin for installation* Make sure that your current working directory contains the .jar file found in the zip archive
- *Could not find Cytoscape app directory* Check whether Cytoscape is installed on your system. If you never started Cytoscape on your machine, try starting it once before installing the plugin. The CytoscapeConfiguration directory will be created on its first start.

7.2 Session files

Cytoscape offers to save the current working state in a .cys session file. The plugin is fully compatible with this function. However, reading session files generated with older versions of the plugin may fail. In general, you can expect session to work if the first two digits of the plugin version are identical. For example, a session created with version 1.2.0 is guaranteed to be readable by plugins of version 1.2.X, but not by version 1.1.0.

7.3 Slow analyses and 'out of memory' errors

Keeping the full timelines of long trajectories consumes a lot of memory, especially for large systems. Reducing the amount of analysed time frames, e.g. by using the 'sieve' option in CPPTRAJ or during import, can accelerate analyses substantially. For a workstation with 8 GB RAM and a protein of about 200 amino acids, we found a number of 5000 frames to work well.

7.4 Subnetworks and changes in node/edge tables

It is often useful to create one or more subnetworks to analyze specific regions in the protein. It is important to remember that the node and edge data tables are shared between the network and all subnetworks. This means that an analysis performed in a subnetwork also changes the data tables in all other networks belonging to the same group.

References

[1] P Shannon, A Markiel, O Ozier, N S Baliga, J T Wang, D Ramage, N Amin, B Schwikowski, and T Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res*, 13:2498–2504, 2003.

[2] Peter W. Rose, Bojan Beran, Chunxiao Bi, Wolfgang F. Bluhm, Dimitris Dimitropoulos, David S. Goodsell, Andreas Prlić, Martha Quesada, Gregory B. Quinn, John D. Westbrook, Jasmine Young, Benjamin Yukich, Christine Zardecki, Helen M. Berman, and Philip E. Bourne. The rcsb protein data bank: redesigned web site and web services. *Nucleic Acids Research*, 39(suppl_1):D392–D401, 2011.

[3] D. R. Roe and 3rd Cheatham, T. E. Ptraj and cpptraj: Software for processing and analysis of molecular dynamics trajectory data. *J Chem Theory Comput*, 9(7):3084–95, 2013.

[4] Ulrik Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2):136 – 145, 2008.

[5] Antonio del Sol, Hirotomo Fujihashi, Dolors Amoros, and Ruth Nussinov. Residues crucial for maintaining short paths in network communication mediate signaling in proteins. *Molecular Systems Biology*, 2(1):2006.0019, 2006.

[6] A. Grossfield and D. M. Zuckerman. Quantifying uncertainty and sampling quality in biomolecular simulations. *Annu Rep Comput Chem*, 5:23–48, 2009.