`psntools` user guide

Protein Structure Networks (PSNs)

Protein Structure Networks (or PSNs) are graph-theoretical representations of a single protein structure or an ensemble of structures.

In the graph, nodes represent protein *residues* , while edges correspond to *pairwise interactions* between such residues. These interactions can have a direct physical meaning (i.e., salt bridges, hydrogen bonds) or incorporate other concepts of "contacts" between residues.

Within the scope of `psntools` , PSNs are intended to be:

- Weighted networks, namely networks where each edge has a weight associated with it, usually representing the strength of the interaction between the nodes connected by the edge itself.
- Undirected networks, meaning that the edges have no direction (i.e., an edge connecting node A to node B is the same as an edge connecting node B to node A).

Functionalities

Default settings in `psntools`

Hard-coded default settings for `psntools` utilities live in the `psntools.defaults` module. So far, the following hard-coded settings can be found:

- `CONFIG_PLOT_DIR` , the directory where `psntools` will look for configuration files used for plotting (see the "Plotting with `psntools` " section below for a more detailed description of these configuration files).

Core objects and PSN analysis in `psntools`

The `psntools.core` module contains classes defining the core objects used by `psntools` .

`PSN`

The `PSN` class implements single PSNs. The PSN is implemented as a `networkx.Graph` object whose nodes are `MDAnalysis.core.groups.Residue` instances representing the protein residues. Please go here and here for more information about `Graph` and `Residue` objects, respectively.

The `PSN` class also defines some attributes that all `PSN` instances will inherit:

- The `NODE_STR_FMT` attribute determines the format of the string representation of the nodes of the PSN. The default format is `{segid}-{resnum}{resname}` , where each keyword represents a valid `Residue` attribute in `MDAnalysis` .
- The `DEFAULT_SEGID` attribute sets the segment ID assigned to protein chains not having an ID. The default is `SYSTEM` .

A `PSN` is built from:

- An adjacency `matrix` M, namely a $N \times N$ matrix (with N number of residues in the protein system represented by the PSN) where each cell M_{ij} contains the weight associated to the edge between nodes i and j.
- A `universe` U, namely an `MDAnalysis` `Universe` instance representing the protein system (please go here for more information about Universe objects in `MDAnalysis`).

Before initializing a new `PSN`, a Universe object must be created from a topology file (ideally a PDB file) describing the protein system of interest.

```
import MDAnalysis as mda
u = mda.Universe("topology.pdb")
```

Then, a new `PSN` instance can be initialized, passing the `universe` to the constructor, together with either a `numpy` matrix or a string pointing to a file containing a matrix in a format readable by `numpy`:

```
import psntools.core as core
psn = core.PSN("matrix.dat", u)
```

The newly created `PSN` has two attributes, populated at initialization:

- The `matrix` attribute contains the matrix from which the PSN was created.
- The `graph` attribute represents the PSN as a `NetworkX` `Graph` object whose nodes are `MDAnalysis.core.groups.Residue` instances representing the protein residues.

General `psntools.core.PSN` methods

`get_nodes_residues2strings`

Returns a dictionary mapping the `MDAnalysis.core.groups.Residue` instances to their string representation, formatted as per the `NODE_STR_FMT` attribute.

`get_nodes_strings2residues`

Returns a dictionary mapping the string representations of the nodes, formatted as per the `NODE_STR_FMT` attribute, to the `MDAnalysis.core.groups.Residue` instances representing the nodes.

Methods in `psntools.core.PSN` for PSN analyses

`get_metric`

The method computes a given metric for the entire PSN, each node or edge. Available metrics so far are:

- Node metrics:
  - `degree`, computed calling the `get_degree` method.
  - `betweenness_centrality`, computed calling the `get_betweenness_centrality` method.
  - `closeness_centrality`, computed calling the `get_closeness_centrality` method.
- Edge metrics: none.
- Graph metrics: none.

Parameters:

- `metric`, the name of the metric to be computed.
- `kind`, whether the metric is a `node`, `edge`, or `graph` metric (some metrics have identical names even if they refer to different entities).
- `metric_kws`, a dictionary of keyword arguments to be passed to the method calculating the metric of interest.

`get_edges`

It returns a dictionary mapping each edge of the PSN to its weight.

Parameters:

- `min_weight`, the minimum weight for an edge to be reported. The default is no minimum weight set (`None`).
- `max_weight`, the maximum weight for an edge to be reported. The default is no maximum weight set (`None`).
- `mode`, whether to report all edges (`"all"`) or only intra-chain (`"intrachain"`) or inter-chains (`"interchain"`) edges. The default is `"all"`.
- `node_fmt`, whether to represent nodes as `Residue` instances (`"residues"`) or use their string representations (`"strings"`) in the output dictionary. The default is `"residues"`.

`get_degree`

It returns a dictionary mapping each node in the PSN to its degree. The degree of a node corresponds to the number of edges having the node as an extreme.

Parameters:

- `node_fmt`, whether to represent nodes as `Residue` instances (`"residues"`) or use their string representations (`"strings"`) in the output dictionary. The default is `"residues"`.

`get_hubs`

If the node is a hub, it returns a dictionary mapping each node in the PSN to its degree. A hub is a node having a degree greater than or equal to a pre-defined value.

Parameters:

- `min_degree`, the minimum degree for a node to be considered a hub. The default is `1`.
- `max_degree`, the maximum degree for a node to be reported in the output dictionary. The default is no maximum degree set (`None`).
- `node_fmt`, whether to represent nodes as `Residue` instances (`"residues"`) or use their string representations (`"strings"`) in the output dictionary. The default is `"residues"`.

`get_betweenness_centrality`

It returns a dictionary mapping each node of the PSN to its betweenness centrality value. The definition of betweenness centrality and how it is calculated can be found here. This method relies on the `networkx.algorithms.centrality.betweenness_centrality` function.

Parameters:

- `node_fmt`, whether to represent nodes as `Residue` instances (`"residues"`) or use their string representations (`"strings"`) in the output dictionary. The default is `"residues"`.
- `**kwargs`, the keyword arguments that will be passed to the

`networkx.algorithms.centrality.betweenness_centrality` function. See here for a complete list of possible arguments.

`get_closeness_centrality`

It returns a dictionary mapping each node of the PSN to its closeness centrality value. The definition of betweenness centrality and how it is calculated can be found here. This method relies on the `networkx.algorithms.centrality.closeness_centrality` function.

Parameters:

- `node_fmt`, whether to represent nodes as `Residue` instances (`"residues"`) or use their string representations (`"strings"`) in the output dictionary. The default is `"residues"`.
- `**kwargs`, the keyword arguments that will be passed to the `networkx.algorithms.centrality.betweenness_centrality` function. See here for a complete list of possible arguments.

`get_connected_components`

It returns a list of sets, with each set representing a connected component of the PSN. A connected component is a subset of nodes of the PSN, such as all nodes inside the component are connected through any number of edges but disconnected from any node outside the component. This method relies on the `networkx.algorithms.components.connected_components` function.

Parameters:

- `min_size`, the minimum size (expressed in number of nodes) for a connected component to be reported. The default is no minimum size set (`None`).
- `max_size`, the maximum size (expressed in number of nodes) for a connected component to be reported. The default is no maximum set (`None`).
- `ascending`, whether to report the connected components by increasing size (smallest first). The default is `False`.
- `node_fmt`, whether to represent nodes as `Residue` instances (`"residues"`) or use their string representations (`"strings"`) in the output dictionary. The default is `"residues"`.

`get_shortest_paths`

It returns a dictionary mapping a tuple representing each pair of nodes between which shortest paths have been calculated to a dictionary mapping each path found for that pair to the path's weight. A path between two nodes is composed of the nodes (and edges) that need to be traversed to go from one node to the other. The shortest paths between two nodes feature the minimum number of nodes (and edges) traversed. This method relies on the `networkx.algorithms.shortest_paths.generic.all_shortest_paths` function.

Parameters:

- `pairs`, an iterable of pairs of strings formatted as per `NODE_STR_FMT` representing nodes between which paths need to be computed.
- `node_fmt`, whether to represent nodes as `Residue` instances (`"residues"`) or use their string representations (`"strings"`) in the output dictionary. The default is `"residues"`.

Selections

`select_nodes`

It selects a subset of nodes in the PSN. The `matrix` and `graph` attributes of the `PSN` will be modified in place.

Parameters:

- `nodes`, an iterable of strings formatted as per `NODE_STR_FMT` representing the nodes to be selected.

`select_edges`

It selects a subset of edges in the PSN. The `matrix` and `graph` attributes of the `PSN` will be modified in place.

Parameters:

- `min_weight`, the minimum weight for an edge to be reported. The default is no minimum weight set (`None`).
- `max_weight`, the maximum weight for an edge to be reported. The default is no maximum weight set (`None`).
- `mode`, whether to report all edges (`all`) or only intra-chain (`intrachain`) or inter-chains (`interchain`) edges. The default is `"all"`.

`PSNGroup`

The `PSNGroup` class implements groups of PSNs, each identified by a custom label. A `PSNgroup` is built from:

- `psns`, an iterable of `PSN` instances representing the PSNs in the group.
- `labels`, an iterable of custom labels to identify the single PSNs in the group (i.e., if the PSNs represent different protein systems, a wild-type protein, and mutants, etc.). If no labels are passed, integer labels (starting from 0) will be used.
- `mappings`, a CSV file containing the one-to-one "mappings" of the nodes of the PSNs, where columns are labeled with the PSNs' labels and cells in each column have the string representations of each PSN's nodes. This is needed to ensure that nodes across the different PSNs are handled the way you expect them to. For example, you may have a PSN representing a wild-type protein and a series of PSNs expressing single mutants of such protein at the same position. Usually, in a comparison between such PSNs in a `PSNGroup`, the wild-type residue, and mutant residues, despite being at the same place in the protein, would be treated as different nodes since different `Residue` instances represent them. However, you may want this position treated as it was the same node across the PSN (for example, to compare interactions made by the wild-type protein and the mutants at this position). This is when the mappings come in handy since you can put nodes that you want to be treated as they were the same on the same row.

Instead of passing the `psns` argument, you can also build the single `PSN` instances on the fly by giving:

- `universes`, an iterable of `Universe` instances representing the protein systems.
- `matrices`, an iterable of either `numpy` matrices or strings pointing to files containing matrices in a format readable by `numpy`.

The `PSNGroup` class also defines a set of attributes that all `PSNGroup` instances will have:

- `psns`, a dictionary mapping the PSN labels to the corresponding `PSN` instances.
- `mappings`, a dictionary containing the PSN-to-PSN mappings parsed from the mappings' CSV file.

Analyses

`get_common_hubs`

It computes the hubs common to each possible of PSNs in the `PSNGroup`. It returns a dictionary mapping each combination of PSNs (identified by their labels) to a dictionary having as keys the hubs common to all PSNs in the combination and as values dictionaries where the degree of these hubs in each of the PSNs is reported.

Parameters:

See `PSN.get_hubs`. However, `node_fmt` defaults to `"strings"`.

Furthermore, a new parameter `inters_mode` can be passed. `inters_mode` specifies the intersection mode to be used when finding intersections between the sets of hubs. `"distinct"`, the default, means that, if hub "1" belongs to the intersection between sets A and B, but it also belongs to the intersection between A, B, and C, it will not be reported as a member of the intersection between A and B, but only as a member of the "stricter" intersection between A, B, and C. `"intersect"` means that hub "1" will be reported as a member of both intersections.

`get_common_edges`

It computes the edges common to each possible of PSNs in the `PSNGroup`. It returns a dictionary mapping each combination of PSNs (identified by their labels) to a dictionary having as keys the edges common to all PSNs in the combination and as values dictionaries where the weight of these edges in each of the PSNs is reported.

Parameters:

See `PSN.get_edges`. However, `node_fmt` defaults to `"strings"`.

Furthermore, a new parameter `inters_mode` can be passed. `inters_mode` specifies the intersection mode to be used when finding intersections between the sets of edges. `"distinct"`, the default, means that, if edge "1" belongs to the intersection between sets A and B, but it also belongs to the intersection between A, B, and C, it will not be reported as a member of the intersection between A and B, but only as a member of the "stricter" intersection between A, B, and C. `"intersect"` means that edge "1" will be reported as a member of both intersections.

Data frames in `psntools`

`psntools` can produce data frames containing data collected from the analyses of a `PSN` or of a `PSNGroup`. Utility functions to create such data frames are included in `psntools.dataframes`.

`get_psn_df`

It returns a `pandas.DataFrame` representation of a PSN, where rows and columns represent the nodes of the PSN, and cells store the weights of the edges connecting them.

Parameters:

- `psn`, the `PSN` instance.

`get_nodes_df`

It computes selected metrics for all nodes of a `PSN` and reports the results in a `pandas.DataFrame`. The metrics will either be calculated on the fly or their values will be taken from a dictionary of dictionaries keyed on the metrics' names (outer dictionary) and the `Residue` instances representing the nodes (inner dictionaries) if calculated elsewhere. Rows of the data frame represent nodes, while columns represent the different metrics.

Parameters:

- `psn`, the `PSN` instance.
- `metrics`, a dictionary where keys are node metrics to be computed, and values are dictionaries of keyword arguments to be passed to the methods calculating those metrics.

`get_edges_df`

It creates a `pandas.DataFrame` containing the edges found in a `PSN`.

Parameters:

- `data`, a pre-computed dictionary of edges (as the one outputted by the `psntools.core.PSN.get_edges` method).
- `psn`, a `PSN` instance. It should not be passed if `data` is given.
- `sort_by`, whether to sort the edges by node name (`"node"`) or edge weight (`"weight"`). The default is `"node"`.
- `ascending`, whether the sorting procedure should be ascending. The default is `True`.
- `**kwargs`, the keyword arguments that will be passed to the `psntools.core.PSN.get_edges` method, if edges need to be extracted from the PSN.

`get_hubs_df`

It creates a `pandas.DataFrame` containing the hubs found in a `PSN`.

Parameters:

- `psn`, the `PSN` instance.
- `sort_by`, whether to sort the hubs by node name (`"node"`) or node degree (`"degree"`). The default is `"degree"`.
- `ascending`, whether the sorting procedure should be ascending. The default is `False`.
- `**kwargs`, the keyword arguments that will be passed to the `psntools.core.PSN.get_hubs` method, if hubs need to be computed from the PSN.

`get_connected_components_df`

It creates a `pandas.DataFrame` containing the connected components found in a `PSN`.

Parameters:

- `data`, a list of sets representing the connected components (as the one returned by the `psntools.core.PSN.get_connected_components` method).
- `psn`, a `PSN` instance. It should not be passed if `data` is given.
- `cc_prefix`, a string to add to each connected component's default name (its number). The default is `"CC_"`. If the string passed is empty, integer labels (starting from 1) will be used for labeling the connected components.

- `node_sep`, the separator for nodes in each connected component. The default is `"."`.

`get_nodes_df_psngroup`

It computes a selected metric for all nodes of all PSNs of a `PSNGroup` and reports the results in a `pandas.DataFrame`. Rows of the data frame represent nodes, while each column represents a single PSN (and is named after the PSN's label).

Parameters:

- `psngroup`, the `PSNGroup` instance.
- `metric`, the label of the node metric to be computed, mapping to a dictionary with the keyword arguments to be passed to the function computing such metric.

`get_largest_connected_components_df_psngroup`

It computes the first n-th most populated connected components for each PSN in a `PSNGroup` and reports their sizes in a `pandas.DataFrame`. Rows of the data frame represent the connected component's names, while columns represent the single PSNs.

Parameters:

- `psngroup`, the `PSNGroup` instance.
- `n_ccs`, the number of connected components to report. The default is `5`.
- `cc_prefix`, a string to add to each connected component's default name (its number). The default is `"CC_"`. If the string passed is empty, integer labels (starting from 1) will be used for labeling the connected components.

`get_common_hubs_dfs`

It returns a dictionary of `pandas.DataFrame` instances containing the common hubs for each possible combination of PSNs in a `PSNGroup`.

Parameters:

- `data`, a pre-computed dictionary of common hubs as obtained with `psntools.core.PSNGroup.get_common_hubs`.
- `psngroup`, a `PSNGroup` instance. It should not be passed if `data` is given.
- `psn_sep`, the separator for PSNs in the combination. The default is `"_"`.
- `**kwargs`, the arguments to be passed to `psntools.core.PSNGroup.get_common_hubs` for calculating common hubs if `data` has not been given.

`get_common_edges_dfs`

It returns a dictionary of `pandas.DataFrame` instances containing the common edges for each possible combination of PSNs in a `PSNGroup`.

Parameters:

- `data`, a pre-computed dictionary of common hubs as obtained with `psntools.core.PSNGroup.get_common_hubs`.
- `psngroup`, a `PSNGroup` instance. It should not be passed if `data` is given.
- `psn_sep`, the separator for PSNs in the combination. The default is `"_"`.

- `node_sep`, the separator for nodes in each edge. The default is `"_"`.
- `**kwargs`, the arguments to be passed to `psntools.core.PSNGroup.get_common_hubs` for calculating common hubs if `data` has not been given.

Writing in `psntools`

Several utility functions to write out various kinds of outputs containing data from the analysis of either a `PSN` or a `PSNGroup` are available in the `psntools.writing` module.

`write_psn_csv`

It writes a CSV file containing the `pandas.DataFrame` generated by the `psntools.dataframes.get_psn_df` function.

Parameters:

- `psn`, the `PSN` instance.
- `outfile`, the output CSV file.
- `csv_sep`, the field separator to be used in the output CSV file. The default is `","`.
- `float_fmt`, the format for floating-point numbers in the output CSV file. The default is `"%2.3f"`.

`write_nodes_list`

It writes a plain text file listing all nodes in a `PSN` (string representation).

Parameters:

- `psn`, the `PSN` instance.
- `outfile`, the output text file.

`write_nodes_csv`

It writes a CSV file containing the `pandas.DataFrame` generated by the `psntools.dataframes.get_nodes_df` function.

Parameters:

- `outfile`, the output CSV file.
- `df`, a pre-computed data frame of edges (as the one outputted by the `psntools.dataframes.get_nodes_csv` function).
- `psn`, a `PSN` instance. It should not be passed if `df` is passed.
- `metrics`, a dictionary where keys are node metrics to be computed, and values are dictionaries of keyword arguments to be passed to the methods calculating those metrics. It should not be given if `df` is passed.
- `csv_sep`, the field separator to be used in the output CSV file. The default is `","`.
- `float_fmt`, the format for floating-point numbers in the output CSV file. The default is `"%2.3f"`.

`write_edges_csv`

It writes a CSV file containing the `pandas.DataFrame` generated by the `psntools.dataframes.get_edges_df` function.

Parameters:

- `outfile`, the output CSV file.
- `df`, a pre-computed data frame of edges (as the one outputted by the `psntools.dataframes.get_edges_csv` function).
- `data`, a pre-computed dictionary of edges (as the one outputted by the `psntools.core.PSN.get_edges` method). It should not be passed if `df` is given.
- `psn`, a `PSN` instance. It should not be passed if `df` or `data` are given.
- `sort_by`, whether to sort the edges by node name (`"node"`) or edge weight (`"weight"`). The default is `"node"`.
- `ascending`, whether the sorting procedure should be ascending. The default is `False`.
- `csv_sep`, the field separator to be used in the output CSV file. The default is `","`.
- `float_fmt`, the format for floating-point numbers in the output CSV file. The default is `"%2.3f"`.
- `**kwargs`, the keyword arguments that will be passed to the `psntools.core.PSN.get_edges` method, if edges need to be extracted from the PSN.

`write_hubs_csv`

It writes a CSV file containing the `pandas.DataFrame` generated by the `psntools.dataframes.get_hubs_df` function.

Parameters:

- `outfile`, the output CSV file.
- `df`, a pre-computed data frame of hubs (as the one outputted by the `psntools.dataframes.get_hubs_df` function).
- `psn`, a `PSN` instance. It should not be passed if `df` is given.
- `sort_by`, whether to sort the hubs by node name (`"node"`) or node degree (`"degree"`). The default is `"node"`.
- `ascending`, whether the sorting procedure should be ascending. The default is `False`.
- `csv_sep`, the field separator to be used in the output CSV file. The default is `","`.
- `**kwargs`, the keyword arguments that will be passed to the `psntools.core.PSN.get_hubs` method, if hubs need to be calculated from the PSN.

`write_connected_components_csv`

It writes a CSV file containing the `pandas.DataFrame` generated by the `psntools.dataframes.get_connected_components_df` function.

Parameters:

- `outfile`, the output CSV file.
- `df`, a pre-computed data frame of connected components (as the one outputted by the `psntools.dataframes.get_connected_components_df` function).
- `data`, a pre-computed dictionary of connected components (as the one outputted by

`psntools.core.PSN.get_connected_components`). It should not be passed if `df` is given.
- `psn`, a `PSN` instance. It should not be passed if `df` or `data` are given.
- `cc_prefix`, a string to add to each connected component's default name (its number). The default is `"CC_"`. If the string passed is empty, integer labels (starting from 1) will be used for labeling the connected components.
- `node_sep`, the separator for nodes in each connected component. The default is `"."`.
- `csv_sep`, the field separator to be used in the output CSV file. The default is `","`.
- `**kwargs`, the keyword arguments that will be passed to the `psntools.core.PSN.get_connected_components` method, if connected components need to be calculated from the PSN.

`write_shortest_paths_csvs`

It writes as many CSV files as the data frames (`pandas.DataFrame`) generated by the `psntools.dataframes.get_shortest_paths_dfs` function.

Parameters:

- `data`, a pre-computed dictionary of shortest paths (as the one outputted by `psntools.core.PSN.get_shortest_paths`).
- `psn`, a `PSN` instance. It should not be passed if `data` is given.
- `outfiles_prefix`, the prefix to be used for the output CSV files. The default is `"path_"`.
- `sort_by`, whether to sort the hubs primarily by path length and secondarily by path weight (`("length", "weight")`) or viceversa (`("weight", "length")`). The default is `("length", "weight")`.
- `ascending`, whether the sorting procedures should be ascending. The default is `(False, False)`.
- `pair_node_sep`, the separator for nodes in each pair. The default is `"_"`.
- `path_node_sep`, the separator for nodes in each path. The default is `"."`.
- `csv_sep`, the field separator to be used in the output CSV file. The default is `","`.
- `float_fmt`, the format for floating-point numbers in the output CSV file. The default is `"%2.3f"`.
- `**kwargs`, the keyword arguments that will be passed to the `psntools.core.PSN.get_shortest_paths` method, if shortest paths need to be calculated from the PSN.

`write_nodes_csv_psngroup`

It writes a CSV file containing the data frame generated by the `psntools.dataframes.get_nodes_df_psngroup` function.

Parameters:

- `df`, a pre-computed data frame of nodes (as the one outputted by the `psntools.dataframes.get_nodes_df_psngroup` function).
- `psngroup`, a `PSNGroup` instance. It should not be passed if `df` is passed.
- `metric`, the label of a node metric to be computed, mapping to a dictionary with the keyword arguments to be passed to the function computing such metric.
- `csv_sep`, the field separator to be used in the output CSV file. The default is `","`.
- `float_fmt`, the format for floating-point numbers in the output CSV file. The default is `"%2.3f"`.

`write_common_hubs_csvs`

It writes as many CSV files as possible combinations of PSNs in a `PSNGroup` having hubs in common, listing such common hubs and their degree in each PSN.

Parameters:

- `dfs`, pre-computed data frames of common hubs (as those outputted by the `psntools.dataframes.get_common_hubs_dfs` function).
- `data`, a pre-computed dictionary of common hubs (as the one outputted by the `psntools.core.PSNGroup.get_common_hubs` function). It should not be passed if `dfs` is passed.
- `psngroup`, a `PSNGroup` instance. It should not be passed if `dfs` or `data` are passed.
- `outfiles_prefix`, the prefix to be used for the output CSV files. The default is `"hubs_"`.
- `psn_sep`, the separator for PSNs in the combination. The default is `"_"`.
- `csv_sep`, the field separator to be used in the output CSV file. The default is `","`.
- `**kwargs`, the keyword arguments that will be passed to the `psntools.core.PSNGroup.get_common_hubs` method, if common hubs need to be calculated from the PSN group.

`write_common_edges_csvs`

It writes as many CSV files as possible PSN combinations of PSNs in a `PSNGroup` having edges in common, listing such common edges and their degree in each PSN.

Parameters:

- `dfs`, pre-computed data frames of common edges (as the ones outputted by the `psntools.dataframes.get_common_edges_dfs` function).
- `data`, a pre-computed dictionary of common edges (as the one outputted by the `psntools.core.PSNGroup.get_common_edges` function). It should not be passed if `dfs` is passed.
- `psngroup`, a `PSNGroup` instance. It should not be passed if `dfs` or `data` are passed.
- `outfiles_prefix`, the prefix to be used for the output CSV files. The default is `"edges_"`.
- `psn_sep`, the separator for PSNs in the combination. The default is `"_"`.
- `node_sep`, the separator for nodes in each edge. The default is `"_"`.
- `csv_sep`, the field separator to be used in the output CSV file. The default is `","`.
- `float_fmt`, the format for floating-point numbers in the output CSV file. The default is `"%2.3f"`.
- `**kwargs`, the keyword arguments that will be passed to the `psntools.core.PSNGroup.get_common_edges` method, if common hubs need to be calculated from the PSN group.

Plotting with `psntools`

`psntools` provides plotting utilities to visualize the results of the `PSN` and `PSNGroup` analyses in the `psntools.plotting` module.

An additional module, `psntools.upset`, contains a class to generate UpSet plots [^1] to visualize the common hubs or edges found in every possible combination of PSNs in a `PSNGroup`. This class is not meant to be used directly but through a wrapper function (described later) that lives in the `psntools.plotting` module.

The configuration file system

For tweaking the graphics of the plots (axes' titles, font type, and size of the text elements, color palette, etc.), `psntools` relies on a set of plot-specific YAML configuration files that the user can customize.

An example of the configuration file for each plot type is provided with the package in the `config_plot` directory. This is also the default directory where `psntools` will look for configuration files if only the name of the configuration file is provided (without the .yaml extension). You can always see where `psntools` looks for the configuration files by inspecting the `psntools.defaults.CONFIG_PLOT_DIR` variable.

Functions

`plot_heatmap_nodes`

It generates a heatmap with a set of nodes represented on the x-axis and the value of a specific node metric calculated for each node in different PSNs on the y-axis. It outputs a PDF file containing the plot.

Parameters:

- `df`, a data frame containing the node metric's values for the nodes of interest in different PSNs (as the one outputted by the `psntools.dataframes.get_nodes_df_psngroup` function).
- `outfile`, the output PDF file.
- `configfile`, the YAML configuration file containing graphical parameters to be used in the plot.
- `selected_nodes`, a list of nodes that should be selected from the data frame. Only values corresponding to these nodes will be plotted.
- `nodes_per_page`, how many nodes are plotted on each PDF file page. The default is `20`.
- `psn_labels`, a list of custom labels for the PSNs present in the data frame. Labels must be passed in the same order as the corresponding PSNs in the data frame.
- `node_labels`, a list of custom labels used for the nodes represented in the data frame. Labels must be passed in the same order as the corresponding nodes in the data frame.

`plot_barplot_connected_components`

It generates a bar plot with the distribution of nodes in the most populated connected components of a PSN.

Parameters:

- `df`, a data frame containing data about the distribution of nodes in the most populated connected components of a PSN (as the one outputted by the `psntools.dataframes.get_connected_components_df_psngroup` function).
- `outfile`, the output PDF file.
- `configfile`, the YAML configuration file containing graphical parameters in the plot.
- `n_ccs`, the number of most populated connected components to plot. The default is `5`.
- `cc_prefix`, a string to add to each connected component's default name (its number). The default is `"CC_"`.
- `psn_labels`, a list of custom labels for the PSNs present in the data frame. Labels must be passed in the same order as the corresponding PSNs in the data frame.

`plot_upsetplot`

It generates an UpSet plot to visualize the intersections between the sets of hubs or edges found in the PSNs of a `PSNGroup`.

Parameters:

- `psngroup`, a `PSNGroup` instance.
- `item_type`, whether the intersections calculated between the PSNs in the `psngroup` will be of `"hubs"` or `"edges"`. The default is `hubs`.
- `outfile`, the output PDF file.
- `configfile`, the YAML configuration file containing graphical parameters in the plot.
- `**kwargs`, the keyword arguments to be passed to `psntools.core.PSNGroup.get_common_hubs` or to `psntools.core.PSNGroup.get_common_edges` (depending on the `item_type`) to calculate common hubs/edges in the `PSNGroup`.

References

[1]: Lex, Alexander, et al. "UpSet: visualization of intersecting sets." IEEE transactions on visualization and computer graphics 20.12 (2014): 1983-1992.