PyInKnife2 user guide

Reference publication:

PyInteraph2 and PyInKnife2 to analyze networks in protein structural ensembles

Valentina Sora[1,2], Matteo Tiberti[1], Deniz Dogan[1], Shahriyar Mahdi Robbani[1], Joshua Rubin[1], Elena Papaleo[1,2]

[1] Computational Biology Laboratory, Danish Cancer Society Research Center, Copenhagen, Strandboulevarden 49, 2100, Copenhagen, Denmark
[2] Cancer Systems Biology, Section of Bioinformatics, Department of Health and Technology, Technical University of Denmark, 2800, Lyngby, Denmark

A early version of the pre-print available on biorxiv:

doi: https://doi.org/10.1101/2020.11.22.381616

PyInKnife2 is an accompanying tool for PyInteraph2 so we encourage to read the introduction part of the PyInteraph2 user guide first (https://github.com/ELELAB/pyinteraph2).

PyInKnife2 implements the approach originally proposed in Salamanca Viloria et al., 2017. The goal of PyInKnife2 is to provide a tool to assess the robustness of the PSNs generated by PyInteraph2 from protein conformational ensembles generated by molecular dynamics (MD) simulations. For a detailed description of the theoretical framework and the methodological choices underlying PyInKnife, please refer to the original publication (Salamanca Viloria et al., 2017). Furthermore, for a more in-depth explanation of the options used to generate the networks with PyInteraph2, please refer to the original PyInteraph publication (Tiberti et al. 2014) and PyInteraph2 pre-print (Sora and Tiberti et al., 2020) and the repository (https://github.com/ELELAB/pyinteraph2)).

The PyInKnife2 suite consists of three executables:

- *pyinknife_run*, which is responsible for running the PyInKnife pipeline (described below).
- *pyinknife_aggregate*, which takes care of aggregating the raw data generated by the pipeline.
- *pyinknife_plot*, which provides utilities to visualize the aggregated data.

For information on the installation please consult:

https://github.com/ELELAB/PyInKnife2#installation

PyInKnife2 relies on an ecosystem of YAML configuration files defining the parameters of the pipeline (for example, which networks should be generated), and options for the aggregation of the raw data and for plotting. Examples of such configuration files can be found inside the *config_plot* directory within the package.

pyinknife_run

The *pyinknife_run* executable takes the following arguments in input:

-f, --trj	Input trajectory (ensemble).
-s, --top	Input topology file.
-r, --ref	Input reference structure.
-c, --configfile	Configuration file containing the parameters for the run. Examples of configuration files can be found inside the 'config' directory inside the package.
-d, --rundir	Directory where to run the pipeline.
-n, --nproc	Number of processes to be started in parallel. The default is 1 process (no parallelization).
-ncaa, --noncanonical-residues	Non-canonical residues present in your system.

The PyInKnife2 pipeline starts by performing *N* resamplings on the ensemble using a jackknife strategy, meaning that *100/N* (*n*) percent of structures in the ensemble will not be considered in each resampling. For example, if 10 resamplings are performed, 10% of the structures will be excluded in each of them. Assuming the structures are ordered inside the ensemble (for instance, if the ensemble comes from a molecular dynamics simulation and each structure corresponds to a frame of the simulation), the structures removed in each resampling will be contiguous.

Therefore, PyInKnife2 can compute, for each resampled ensemble, networks of salt bridges, hydrophobic contacts, and hydrogen bonds as calculated by PyInteraph2. Different networks can be built for different distance and occurrence cut-offs for all interaction types and for different modalities for salt bridges (i.e., interactions between charged groups with the same charge, between charged groups with opposite charge, or both) and hydrogen bonds (i.e., main chain - main chain, main chain - side chain, and side chain - side chain hydrogen bonds). PyInKnife2 then can calculate connected components and hubs for each network produced. The comparison between the values of these two metrics in the networks built from each resampling of the ensemble is used to assess the robustness of the networks.
However, before a meaningful comparison can be produced, raw data need to be aggregated via *pyinknife_aggregate*.

pyinknife_aggregate

pyinknife_aggregate can be run with the following options:

-c, --configfile	Configuration file containing the parameters for the run. Examples of configuration files can be found inside the 'config' directory inside the package.
-ca, --configfile-aggregate	Configuration file containing the parameters for the aggregation. Examples of configuration files can be found inside the 'config' directory inside the package.
-d, --rundir	Directory where the pipeline was run.
-od, --outdir	Directory where to store the files generated by the aggregation process.
--firstccs	First # most populated connected components to be considered. The default is 5.

pyinknife_aggregate parse the output files generated by PyInteraph2 to extract information regarding the number of hubs and size of the most populated connected components found in the networks. For each network type, such information for the networks built from all resampled ensembles is aggregated in a CSV file. Data contained in these CSV files can then be plotted using *pyinknife_plot*.

pyinknife_plot

pyinknife_plot accepts the following options:

-c, --configfile	Configuration file containing the parameters for the run. Examples of configuration files can be found inside the 'config' directory inside the package.
-ca, --configfile-aggregate	Configuration file containing the parameters for the aggregation. Examples of configuration files can be found inside the 'config' directory inside the package.
-p, --plot	Whether to plot hubs ('hubs') or connected components ('ccs').
-cp, --configfile-plot	Configuration file containing the parameters for plotting. Examples of configuration files can be found inside the 'config' directory inside the package.
-d, --rundir	Directory where the aggregate outputs are saved.
-od, --outdir	Directory where to store the files generated by the plotting.

pyinknife_plot takes in input the output files generated by *pyinknife_aggregate* and produces bar plots summarizing the variation observed in either the size of the most populated connected component in each resampled network (if "ccs" was specified in the -p, --plot option) or the number of hubs found for each hub degree (if "hubs" was specified).

Some considerations before starting to allocate PyInKnife2 runs

PyInKnife2 could be time consuming depending on the trajectory length or the number of PSNs and cutoffs that the user want to test using the same config run.yalm file.
To give some guidance for new users, we have estimated the time required with different combinations of input parameters. The tests have been carried out using the CYPA MD trajectory (164 residues, 2471 atoms).

	# frames	cores	# cutoffs to test	# resampling	time (minutes)
cmPSN	10000	4	4	10	8.3
cmPSN	50000	4	4	10	37
cmPSN	250000	4	4	10	184.15
sbIIN	10000	4	4	10	1.94
sbIIN	250000	4	4	10	29
hbIIN (sc-sc)	1000	4	4	10	19

A case study of CYPA

This tutorial provides a step-by-step guide to the usage of the different PyInKnife2 executable We used as a case study one microsecond MD trajectory of the Cyclophilin A wild-type enzyme, previously published (Salamanca Viloria et al., 2017). In particular, we here show examples of analyses on a skipped version of the trajectory including 10000 frames (traj_prot_dt100.xtc).
This tutorial is based on a trajectory that has been resolved for periodic boundary conditions (PBC) and keeping the protein atoms only in our skipped trajectory. The pre-processing for PBC is always a step required before using Pyinteraph2 and PyInKnife2 an MD ensemble.
The same trajectory is used in the case study for the Pyinteraph2 user guide for sake of clarity.

PyInKnife applied to center-of-mass PSN (cmPSN)

This is done using `pyinknife_run` and a `run.yaml` configuration file. In the `run.yaml` configuration file only the hydrophobic contact option needs to be set to *True* and the other IINs set to *False*.

We tested with four different distance cutoffs, in the range between 4.5-6.0 Å and separated by 0.5 Å each.
We used 10 resamplings so that 10% of the structures were excluded in each of them as suggested in the original approach (Salamanca Viloria et al. 2017).
As a reference structure, we used the first structure from the simulation (pdbmovie_1.pdb in the example).

It is important that in the configuration file the flags will reflect the current available flags of PyInteraph2. If the job submission exits with a error that is unclear, there will be log files within each subfolder that can guide the identification of where the problems might be.
PyInKnife2 supports analyses using the outputs of PyInteraph2 that refers to all the intramolecular interactions and does not discriminate between intra- and intermolecular interactions (in a complex with at least two chains) in the current version.

To run the process using our tutorial folder, we recommend using at least 4 cores (one for each distance cutoff scrutinized). If you cannot do this, a suggestion is to change the config file so that you test the tutorial with only one or two distance cutoff values.

```
cd 5.psn/pyinknife2/cmPSN

pyinknife_run  -f ../../../3.filt_trjs/traj_prot_dt100.xtc -s pdbmovie_1.pdb -
r pdbmovie_1.pdb -c run.yaml -d . -n 4
```

This should take approximately 8-10 minutes

The output tree structure is for each resampling and fulltrj:

```
resampling0/
└── hc
    ├── 4.5
    │   └── 20.0
    │       ├── ccs
    │       └── hubs
    ├── 5
    │   └── 20.0
    │       ├── ccs
    │       └── hubs
    ├── 5.5
    │   └── 20.0
    │       ├── ccs
    │       └── hubs
    └── 6.0
        └── 20.0
            ├── ccs
            └── hubs
```

The outputs and log files from the `pyinteraph` step will be in the 'hc/distance_cutoff' folders (e.g., hc/4.5/). The outputs and log files of the `filter_graph` step in the hc/distance_cutoff/occurence_perc (e.g., 20.0 in this example) folder in this example. The outputs and log files of the graph_analysis step are in the subfolders `ccs` and `hubs` for connected components and hubs, respectively.

`pyinknife_run` runs, for each resampling trajectory and for the total trajectory, `pyinteraph`, `fliter_graph` and `graph_analysis` processes of Pyinteraph2.
The outputs for hubs and connected components are then used by `pyinknife_aggregate`.

```
pyinknife_aggregate -c run.yaml -ca aggregate.yaml -d . -od aggregate --firstccs
5
```

This step should write the .csv files in the folder aggregate that are used for the plotting by `pyinknife_plot`

```
pyinknife_plot -c run.yaml -ca aggregate.yaml -cp plot_ccs_barplot.yaml -p ccs
-d aggregate -od plots

pyinknife_plot -c run.yaml -ca aggregate.yaml -cp plot_hubs_barplot.yaml -p hubs
-d aggregate -od plots
```

The plots can be used to understand: 1) the stability of the hubs and connected components values during the simulation (i.e., referring to the standard deviation bar); 2) the distance cutoff which is a good compromise between avoiding a too connected or sparse network in term of connected components, along with have residues with hub behaviour and different hub degrees. For example, in this run a good distance cutoff could be 5 Å, as previously published (Salamanca Viloria et al., 2017). To be more accurate, the user could then run a new round of PyInKnife2 scrutinizing cutoffs in the range of 5-5.5 Å with smaller steps (e.g., 5, 5.10, 5.20, 5.30, 5.40 Å).

We are working on similar protocols to support the new PSN methods implemented in PyInteraph2 (ccmPSN and acPSN).

The analysis of IIN based on hydrophobic clusters (hc) is similar to the example provided here. The config file `run.yaml` needs to include only hydrophophobic (and aromatic) residues, depending on the focus of the analyses. For example:

```
--hc-residues: ALA,ILE,LEU,MET,PHE,PRO,TRP,TYR,VAL
```

Similarly, when the protein includes post-translationally modified residues (or modified residues in general), it is sufficient to introduce their residue name as it appears in the topology file in the config file in the list of ‘hc-residues’. For example, in a case of a phosphorylated serine labelled as ‘SEP’

```
 --hc-residues: ALA,ARG,ASN,ASP,CYS,GLN,GLU,HIS,ILE,LEU,LYS,MET,PHE,PRO,SER,THR,TRP,TYR,VAL
```

Moreover, we did not change the occurrence cutoff in this example or in the following ones, but the protocol can also be used to evaluate the influence of using different cutoffs for occurrence probability on the hubs and connected components of cmPSNs or IINs.

IIN based on salt bridges

The steps are the same as above. Before starting the `run.yaml` needs to be customized so that the mode to run `sb` is set to `True`, the distance cutoffs to scrutinize are listed.
In this example, we tested four different distance cutoffs that are generally used in the analyses of MD simulations to study electrostatic interactions (i.e., 4, 4.5, 5, 5.5 Å).

As above, it is suggested to allocate one core for each distance cutoff under analysis (for a total of 4 cores in this example). If this is not possible, you can run the tutorial using only two or one distances in the config file.

```
cd 5.psn/pyinknife2/sb_dt100
```

```
pyinknife_run  -f ../../../3.filt_trjs/traj_prot_dt100.xtc -s pdbmovie_1.pdb -r pdbmovie_1.pdb -c run.yaml -d . -n 4
```

```
pyinknife_aggregate -c run.yaml -ca aggregate.yaml -d . -od aggregate --firstccs 5
```

```
pyinknife_plot -c run.yaml -ca aggregate.yaml -cp plot_ccs_barplot.yaml -p ccs -d aggregate -od plots
```

```
pyinknife_plot -c run.yaml -ca aggregate.yaml -cp plot_hubs_barplot.yaml -p hubs -d aggregate -od plots
```

Salt bridges are by definition more sparse networks, with the tendency to form small local networks of interactions with 2-4 nodes than using other sources of interactions since they involve only a small subset of residues and the nature of the salt bridge interaction itself. The PyInKnife2 protocol is not necessarily useful to assess the proper distance cutoff for salt bridge networks but it can still be useful to evaluate the standard deviations associated with the estimation of the hub and connected components. We would recommend to use distance cutoffs commonly used in structural biology when you are interested in understanding salt bridge interactions. The possibility to scrutinize cutoffs around the selected values that PyInKnife provides can also be handy to evaluate if interesting electrostatic interactions are lost because of the usage of a strict cutoff.

IIN based on hydrogen bonds

Hydrogen bonds can be analyzed by PyInteraph2 in different groups: all, mainchain-mainchain, sidechain-sidechain, mainchain-sidechian.
The protocol with PyInKnife2 is the same for all of them. The user should remember to change the information in the `run.yaml` depending on the class under analysis, as an example we report the analysis with the sidechain-sidechain hydrogen bonds
This is one of the most time consuming steps with PyInteraph2. For the sake of the tutorial we will run with a trajectory with 1000 frames only (traj_prot_dt1000.xtc)
If you received a warning like:
"UserWarning: Element information is absent or missing for a few atoms. Elements attributes will not be populated." is means that the corresponding 'element' column of the PDB is empty and it cannot be assigned directly. MDAnalysis will use a guess to infer the element from the name of the atom, where needed

```
cd 5.psn/pyinknife2/hb_dt1000
```

```
pyinknife_run  -f ../../../3.filt_trjs/traj_prot_dt1000.xtc -s pdbmovie_1.pdb -
r pdbmovie_1.pdb -c run.yaml -d . -n 4

pyinknife_aggregate -c run.yaml -ca aggregate.yaml -d . -od aggregate --firstccs
5

pyinknife_plot -c run.yaml -ca aggregate.yaml -cp plot_ccs_barplot.yaml -p ccs
-d aggregate -od plots

pyinknife_plot -c run.yaml -ca aggregate.yaml -cp plot_hubs_barplot.yaml -p hubs
-d aggregate -od plots
```

We tested two of the most common used distance cutoff (3.0 and 3.5 Å) for hydrogen bond analyses. From the analyses of the plots it seems that 3.5 Å could be a more suitable value to investigate networks based on this class of interaction, since with a lower distance almost no hubs are identified. To drive more clear conclusions although the user should also try to test different angle cutoffs and evaluate the influence of both the parameters

References

Salamanca Viloria, Juan, et al. "An optimal distance cutoff for contact-based Protein Structure Networks using side-chain centers of mass." Scientific reports 7.1 (2017): 1-11.

Sora, Valentina, Tiberti, Matteo et al. "PyInteraph2 and PyInKnife2 to analyze networks in protein structural ensembles." bioRxiv (2020).

Tiberti, Matteo, et al. "PyInteraph: a framework for the analysis of interaction networks in structural ensembles of proteins." Journal of chemical information and modeling 54.5 (2014): 1537-1551.