User Manual

December 6, 2021

Abstract

User Guide contains instruction for installing and running the software.

Introduction

Requirements

The software requires Python 3.8, pip, conda, git and following updated packages:

- scipy
- ProDy
- numpy
- scipy
- fuzzy-c-means
- GEM
- cdlib
- pymol
- python-dateutil
- pytz

User may install them in a conda environment or it is possible to run install_libraries.bat (on Windows) or setup_macOsX.sh on setupLinux.sh

Installation on Windows Platform

Windows OS: Software Installation
1) Check the installation of conda, python, and git.
2) Download and Unpack the Files.
3) Open a terminal window and go into the directory containing the software.
4) Type following commands to create a new conda environment and install all the libraries.

```
RUN setup.batinstall_libraries.bat
```
(1)

Installation on Linux Platform

Mac OS: Software Installation
1) Check the installation of conda, python, and git.
2) Download and Unpack the Files.
3) Open a terminal window and go into the directory containing the software.
4) Type following commands to create a new conda environment and install all the libraries.

```
./setupLinux.sh
```
(2)

Installation on MacOS Platform

Mac OS: Software Installation
1) Check the installation of conda, python, and git.
2) Download and Unpack the Files.
3) Open a terminal window and go into the directory containing the software.
4) Type following commands to create a new conda environment and install all the libraries.

```
./setupmacosX.sh
```
(3)

Installing Conda

Conda is a third party software for managing Python distribution. For installing conda on windows please refer to: https://docs.conda.io/projects/conda/en/latest/user-guide/install/windows.html

Installing git.

For git installation please refer to: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Workflow of Analysis

After installing the software, in order to run it you have to open a terminal window. Then you may go into the folder in which the software is installed and type:

Launch of the Software

```
conda activate PCN
```
(4)

On Conda Prompt:

```
python pcn_main.py
```
(5)

After launching the software, has to choice the type of the input file. User is able to process protein structures encoded as Protein Data Bank (pdb) files or previously determined contact networks. Thus, software will show the following message:

```
As First Step you must choose Format File Input:  PDB structures or Preprocessed PCN Digit
pdb' to use .pdb files or 'adj' to load existing PCN:
```

Then user has to choice the directory in which the software will store the output (please check writing permissions on that directory), and the directory containing input files. After choosing the directory, user may insert one or more pdb identifiers that will be analysed.

```
Input the Directory in which you want to store the outputs
The software will create three subdirectories
Insert Root Path of the Outputs:  ./
Input the Directory containing Input Files Insert Proteins filepath:

Please Insert Protein PDB Identifiers, separated by comma, without .pdb, e.g.  7nxc for
7nxc.pdb Enter proteins list items (splitted by ','):
```

At this point user has to choose one of the three possible analysis approach.

Spectral will apply spectral clustering on PCN
Embedding will apply embedding followed by clustering
Community will apply community discovery
Choice between 'spectral' (Spectral Clustering), 'embeddings' (Embeddings+Spectral Clustering) and 'community' (Community Extraction):
`Please select the analysis approaches:  SPECTRAL|EMBEDDINGS|COMMUNITY`

```
Spectral will apply spectral clustering on PCN
Embedding will apply embedding followed by clustering
Community will apply community discovery
Choice between 'spectral' (Spectral Clustering), 'embeddings' (Embeddings+Spectral
Clustering) and 'community' (Community Extraction):
```

Spectral Clustering

When prompting community, the user may choose one of the available algorithms for community analysis (this may vary on the basis of the operating system and software version). The current version of the software implements following algorithms wrapping the Scikit library[1]:

Unnorm_SSC : The unnormalised soft spectral clustering

Norm_SSC : it computes a soft clustering using a normalised lapacian

Norm _HSC : it calculates a hard clustering on a normalised laplacian;

Unnorm_HSC : it calculates a hard clustering on a unnormalised laplacian;

HSC_ShiMalik it calculates a hard clustering on a normalised laplacian by Sh [2]

SSC_ShiMalik [2]

Spectral Clustering algorithms have been implemented following the algorithm proposed in [3]. Hard clustering is obtained by using k-means, while soft clustering use fuzzy c-means. When selecting spectral clustering, the software will ask for the choice of the number of clusters (for both k-means and c-means). User may: (i) select a single value as number of cluster, (ii) a set of values; or he can leave to the software the choice of the best number of clusters through an optimisation scheme.

```
Entering k for clustering:  Enter an int, a list of ints (split with ',') or type 'best_k':
```

Embedding+Clustering

By choosing embedding, the software will perform first the embedding of the matrix, then spectral clustering is used to find clusters. The current version of the software implements: HOPE [4] and LaplacianEigenmaps as embedding softwares and all the clustering algorithms presented in the previous section. Embeddings algorithms are provided by the libraries discussed in [5]. Thus, user may select one of the following analysis approaches:`'Unnorm_SSC_HOPE', 'Norm_SSC_HOPE', 'Unnorm_HSC_HOPE', 'Norm_HSC_HOPE', 'HSC_ShiMalik_HOPE', 'SSC_ShiMalik_HOPE', 'Unnorm_SSC_Laplacian 'Norm_SSC_LaplacianEigenmaps', 'Unnorm_HSC_LaplacianEigenmaps', 'Norm_HSC_LaplacianEigenmaps', 'HSC_ShiMalik_Laplac 'SSC_ShiMalik_LaplacianEigenmaps'`.

If the user choose HOPE as embedding algorithms, the software will ask the input of the dimension of the embedding space.

```
Enter d parameter for d-dimensional embedding:
Enter beta parameter for d-dimensional HOPE embedding:
```

Community Extraction

When prompting community, the user may choose one of the available algorithms for community analysis (this may vary on the basis of the operating system and software version). The current version of the software implements following algorithms: Louvain [6], Leiden [7], Infomap [8], Spinglass [9] and Walktrap [10] from the CdLib [11]; asynchronous fluid communities algorithm (asyn_fluid) [12] and the Clauset-Newman-Moore greedy modularity maximization (greedy_modularity) [13], from the networkX package.

The software checks for the existence of pre-computed contact network. If there is no contact network for the analysed protein, the software will generate a novel PCN. User may choose two distance thresholds: **Non covalent bonds threshold distance for PCN costruction**, and **Entering only significant bonds threshold**.

```
Choice one Community Extraction algorithm from ['Louvain', 'Leiden', 'Walktrap',
'Asyn_fluidc', 'Greedy_modularity', 'Infomap', 'Spinglass']:
```

For instance, if the user chooses Louvain, the software will calculate communities by using the Louvain algorithm as implemented in the cdlib library [11]. After the computation of the communities the software will automatically call the Pymol software and it will highlight the communities on the protein structure. Results (community assignments and pymol visualised structure) are stored into the user defined directory.

References

[1] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[2] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[3] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[4] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114, 2016.

[5] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[6] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[7] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):1–12, 2019.

[8] Jianping Zeng and Hongfeng Yu. A distributed infomap algorithm for scalable and high-quality community detection. In *Proceedings of the 47th International Conference on Parallel Processing*, pages 1–11, 2018.

[9] Eric Eaton and Rachael Mansbach. A spin-glass model for semi-supervised community detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, 2012.

[10] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *International symposium on computer and information sciences*, pages 284–293. Springer, 2005.

[11] Giulio Rossetti, Letizia Milli, and Rémy Cazabet. Cdlib: a python library to extract, compare and evaluate communities from complex networks. *Applied Network Science*, 4(1):1–26, 2019.

[12] Ferran Parés, Dario Garcia Gasulla, Armand Vilalta, Jonatan Moreno, Eduard Ayguadé, Jesús Labarta, Ulises Cortés, and Toyotaro Suzumura. Fluid communities: A competitive, scalable and diverse community detection algorithm. In *International Conference on Complex Networks and their Applications*, pages 229–240. Springer, 2017.

[13] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.