Systems Biology

Protein Network Analysis Software

9087412[1],

[1]Department of Bioinformatics and Systems Biology, Address: The University of Manchester.

Received on 010218;

Abstract

Motivation: The main idea behind this project is automatic protein-protein network analysis. A list of protein connections is often composed based on multiple papers, experiments and databases, which makes the analysis error prone if it is performed manually. Here, a user-friendly software is designed in an attempt to increase the efficiency of protein network analysis and subsequent studies.

Results: This software is able to analyse a text file containing pair-wise protein-protein interactions. The output is customisable and can include calculation of an average degree of the network, pulling up a degree of a particular node, creating a file with a degree distribution, showing hubs and adding a connection to the existing network.

Availability: Directory "ProteinNetwork" contains the source code and example files.

1 Introduction

One of the approaches to understand a disease is through modeling of dynamics in a biological system [1]. The specifics of these dynamics including protein-protein interactions give insight into disease mechanisms as well as aids drug-, target and biomarker discovery [2-4]. These interactions are derived by manual curation of scientific literature, high-throughput databases, computational predictions and literature text-mining [5, 6]. Examples of existing experimental methods to determine these interactions include yeast-two hybrid screening and affinity purification coupled to mass spectrometry [7-9].

From the molecular perspective any biological system is a complex of binary relations between different units. Representation of these interactions is based on the Graph theory to simplify the complexity of a real-life network (interactome) [1]. The units in a network are referred as nodes and their interactions or relations as edges. In a protein-protein network edges are undirected to simulate a physical connection between proteins. The importance of minute details and units in a network can play an important role to understand the emergent properties of a network. Some topological properties (Node degree, shortest path, scale-free networks, transivity, centralities etc.) can aid in revealing substructures within a network that further lead to discoveries in disease progression [3]. Manual extraction of these substructures is labour intensive and an automated tool is preferred to reduce error. In this report Network Analyser software, a tool for protein-protein interaction network analysis, is presented.

2 Methods and Implementation

The software is designed in Java 9.0.1. Its main purpose is to analyse a text file containing pairwise protein interactions composing a network. The software offers a range of manipulations to extract information from the file. The software contains five classes: Class Node defines the name of a node, Class Edge defines an edge created by two nodes, Class Network contains collections of Nodes and Edges composing a network as well as the methods for the analysis available for the end user. Class myGUI creates a graphical interface for the software. Class mainClass has the main method.

2.1.1 Class Node

Node class represents a protein that interacts with other proteins in the network. This class has a default constructor and a constructor accepting a string that allows for a Node with a specific name to be created. This class also has getters and setters and it overrides equals() and hashCode() native Java methods in order to compare Node names that share the same name, yet different references.

2.1.2 Class Edge

Edge class represents a physical connection between the proteins (Nodes). The constructor in this class takes two Nodes as arguments and creates an Edge (string), which consists of (Node1 Node2) names. Alike in Node class both equals() and hashCode are overridden.

2.1.3 Class Network

Network class represents a full network as a hashMap <Node, Integer> 'nodes' and LinkedList <Edge> 'edges', where all the Nodes with corre-

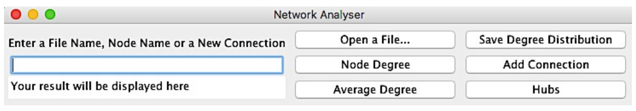
sponding degrees and all the connection are stored, respectively. This class also contains a string ‘answer’, which is used as an output in MyGUI (2.1.4). There are two file-handling methods, where one reads the text file and fills in ‘nodes’ and ‘edges’ (kept in MyGUI), and the other outputs the degree distribution stored in ‘nodes’. Both of these methods throw IOException.
There are also methods to find hubs (proteins with the highest degree), average degree of the network, degree of a particular protein and add a new connection to the network (only non-existent and non-orphan connections will be added). Adding a connection will update ‘nodes’ and ‘edges’ and a new distribution file, hub search, average calculations can be intitialised/ created.

2.1.4 Class MyGUI & Class mainClass

MyGUI class contains the script to build user-friendly interface, where all methods described in 2.1.3 are allocated a button with an ActionListener, which executes the method upon a click. Browsing option to open and save files is available. This class also creates a textArea and textField, where information and results are displayed and data can be entered respectively. This class creates an instance of the Network class. Class mainClass has the main method, which executes the program. In the main method an instance of the MyGUI class is created.

2.2 Utilising Network Analyser

The software can be compiled and run from the command line. A separate window with a graphic interface is initialized once the program is started (fig.1). The textArea will display: “Your result will be displayed here”, and textField will be empty. ‘Open a File…’ button will open a separate browsing window to search for a file. Only a text file of a required format will be read and analysed, otherwise an error will be displayed. Pressing ‘Hubs’ or ‘Average Degree’ will display the result in the TextArea. In order to add a new connection two proteins (Node) names should be typed with a space delimiter into the textFiled and ‘Add Connection’ be pressed. This will display whether a connection has been added. Typing in the name of a Node into the textArea and pressing ‘Node Degree’ will get a degree of a particular protein. ‘Save Degree Distribution’ will open a separate browsing window and any name and location can be chosen for the text file that will contain a degree distribution.


Network Analyser

Enter a File Name, Node Name or a New Connection

Your result will be displayed here

Fig 1. GUI of Network Analyser upon initialising.

3 Results

3.1 Step-by-Step Example

Here, a text file ‘PPI_example.txt’ with hypothetical protein connections is analysed by Network Analyser (fig2).

A	B
A	A
B	C
D	A
D	B
D	E
B	F

Node	Edge(s)
A	4
B	4
C	1
D	3
E	1
F	2

Fig 2. Initial file (left) and degree distribution file (right).

3.1.1 Software Results & Biological Interpretation

The average degree of the network is two; ‘B’ is a hub with 4 edges. Addition of ‘A F’ connection is allowed only once, otherwise “connection already exists is displayed”. This brings ‘A’ degree to 4, which also becomes a hub. Calling ‘H’ for Node degree returns “H does not exist”. Pressing ‘Download Degree Distribution’ yields a text file in fig 2. The “Example” network contains two hubs ‘A’ and ‘B’, which suggests they hold the most connections and are required to sustain the network. Deletion of these proteins will have a deleterious effect on the network according to centrality-lethality concept [10]. Nodes ‘C’ and ‘D’ only have one connection, being peripheral nodes, which usually are the best drug targets [4].

4 Discussion and Conclusions

Overall, the software yields useful output, yet is quite limited in analysis options. Possible improvements would include an addition of degree connectivity, degree centrality options and other properties (e.g. Python NetworkX), which would provide a better understanding of a network [11]. Likewise, visualisation of a network is an additional feature aiding in analysis (Java JUNG interactive graphs) [12]. Alternatively, there are a few sources that offer network visualization and analysis like IntAct (EMBL-EBI) or Cytoscape as well as NAViGaTOR may appear more developed than NetworkAnalyser [13-15]. However, these are suitable only for analysis of known interactions. Advances in high-throughput technologies lead to accumulation of data that is yet to be arranged into networks [18]. On a bigger scale, machine learning as a relatively novel approach to studying protein networks can be considered as it will allow assembling underdeveloped network more efficiently, based on preliminary assessment of well annotated networks including not only proteins but also nucleic acids [16, 17].

References

1. Sevimoglu, T. and K.Y. Arga, *The role of protein interaction networks in systems biomedicine.* Comput Struct Biotechnol J, 2014. **11**(18): p. 22-7.
2. Goh, K.I. and I.G. Choi, *Exploring the human diseasome: the human disease network.* Brief Funct Genomics, 2012. **11**(6): p. 533-42.
3. Taylor, I.W., et al., *Dynamic modularity in protein interaction networks predicts breast cancer outcome.* Nat Biotechnol, 2009. **27**(2): p. 199-204.
4. Winter, C., et al., *Google goes cancer: improving outcome prediction for cancer patients by network-based ranking of marker genes.* PLoS Comput Biol, 2012. **8**(5): p. e1002511.
5. Papanikolaou, N., et al., *Protein-protein interaction predictions using text mining methods.* Methods, 2015. **74**: p. 47-53.
6. Chen, X., M. Chen, and K. Ning, *BNArray: an R package for constructing gene regulatory networks from microarray data by using Bayesian network.* Bioinformatics, 2006. **22**(23): p. 2952-4.
7. Mashaghi, A.R., A. Ramezanpour, and V. Karimipour *Investigation of a protein complex network*. 2004; 41: 113:[Available from: https://doi.org/10.1140/epjb/e2004-00301-0.
8. Fields, S. and O. Song, *A novel genetic system to detect protein-protein interactions.* Nature, 1989. **340**(6230): p. 245-6.
9. Terentiev, A.A., N.T. Moldogazieva, and K.V. Shaitan, *Dynamic proteomics in modeling of the living cell. Protein-protein interactions.* Biochemistry (Mosc), 2009. **74**(13): p. 1586-607.
10. He, X. and J. Zhang, *Why do hubs tend to be essential in protein networks?* PLoS Genet, 2006. **2**(6): p. e88.
11. Developers, N. *networkx 2.1*. 2018 [cited 2018 9 Feb 2018]; Python package for creating and manipulating graphs and networks]. Available from: https://pypi.python.org/pypi/networkxAvailable from: http://networkx.github.io/.
12. Team, T.J.D. *JUNG: The Java Universal Network/Graph Framework.* 2006; Available from: http://jung.sourceforge.net.
13. Orchard, S., et al., *The MIntAct project--IntAct as a common curation platform for 11 molecular interaction databases.* Nucleic Acids Res, 2014. **42**(Database issue): p. D358-63.
14. Li, M., et al., *CytoCluster: A Cytoscape Plugin for Cluster Analysis and Visualization of Biological Networks.* Int J Mol Sci, 2017. **18**(9).
15. Brown, K.R., et al., *NAViGaTOR: Network Analysis, Visualization and Graphing Toronto.* Bioinformatics, 2009. **25**(24): p. 3327-9.
16. Ganapathiraju, M.K., et al., *Schizophrenia interactome with 504 novel protein-protein interactions.* NPJ Schizophr, 2016. **2**: p. 16012.
17. Qi, Y., et al., *Systematic prediction of human membrane receptor interactions.* Proteomics, 2009. **9**(23): p. 5243-55.
18. Ward, R.M., et al., *Big data challenges and opportunities in high-throughput sequencing.* Systems Biomedicine, 2013. **1**(1): p. 29-34.