The Cxnet Complex Network Analyser Software

Árpád Horváth

Óbuda University, Alba Regia University Centre, Székesfehérvár, Hungary
horvath.arpad@arek.uni-obuda.hu

Abstract: The study of complex networks has become important in several fields of science such as biology, sociology and physics. The collection of network data and the storage, analysis and visualisation of these data have become important contributors to the knowledge of programmers working in these fields. Our cxnet software connects several software packages of the Python language to make these tasks easier. One of the main goals of this development is to provide a comfortable application programming interface for students to develop their own programs. The cxnet software package is able to create the software package network of the Ubuntu Linux distribution. This network is a directed network with several types of vertices and connections. It changes quite fast and can be created easily. These properties make it an ideal object of investigation. The present paper describes some useful measures of the properties of the complex networks, the usage of the cxnet package with some examples, and our experiences in the education.

Keywords: complex network; graph theory; education; software

1 Introduction

There are complex systems that can be described as many objects with connections among them. These systems are called *complex networks*, the objects are called *vertices* or *nodes*, and the connections are called *edges*. Most of the networks are growing: many new vertices appear and some vertices disappear, and it is also true to the edges. These networks can be described mathematically as a series of graphs, so the literature of complex networks uses almost the same terminology as graph theory. Networks can be *undirected*, if the vertices in both ends of the edges have the same role, like in the Internet, or *directed*, like the Worldwide Web, where the pages are the vertices and the edges are the links between the web pages. The *degree* of a vertex refers to how many edges join to it. One of the most important properties of networks is the degrees of its nodes, which can be described and plotted as a degree distribution.

The computers and the Internet made it possible at the end of the 20^{th} Century to collect data on large networks, and it turned out that most networks have a power-law function as their degree distributions; these types of networks are called *scale-*

free networks [1]. The average degree is not a typical degree in these networks. In scale-free networks there are a lot of vertices with a small number of neighbours and there are some nodes, the *hubs*, with a number of neighbours a magnitude bigger than the average degree (the arithmetic mean of the degrees). The degree distribution is just one of the many useful measures that can be used to describe a complex network.

Our aim was to develop software that matches the demands of teaching complex networks in higher education, software that is easy to use and extend, open source, and possibly free of charge. There are many tools that can be used to investigate networks. There are network analyser programs with graphical user interface, such as Network Workbench [2] and Pajek [3], which can be used for graph visualization and analysis. They are easy to use, but developing new algorithms for them is quite difficult. Network Workbench runs on most operating systems that support Java. Pajek runs on Windows and with Wine on Linux. The Boost Graph Library implements many graph algorithms in C++ [4], but programming in C++ needs quite a lot of skills. As we stated, Python language is object oriented, but it has a very clear syntax that can be learned with middle-level programming skills, and we decided to use that language. In the Python language, there are two notable network analyzer packages: the igraph and NetworkX. These packages have implemented the most important algorithms of network analysis, but – as their developers do not want them to depend on many other packages – they do not have functions for plotting degree distribution and other functions that are useful in visualizing network properties and for acquisition of the data of special networks.

We developed the cxnet package of the Python language to make the investigations of networks easier. The Python language is ideal for education because it has a straightforward syntax, an excellent interactive shell and many useful standard and third party packages, and it is a free software. The cxnet package has many features beyond the features of the packages it is based on. It can create the network of the software packages of the Debian and Ubuntu Linux in tens of seconds and store these networks into files for further investigation. It can create and plot the degree distributions of any network, and plot it with several binning methods or as cumulative distribution. In the event the distribution is scale-free, it can determine its exponent with an equation that can be derived by the maximum likelihood method and plot a power-law function with this exponent easily. It is able to plot the neighbourhood of a package with edges and vertices coloured according to type, to list the names of the packages with the highest degree, indegree or outdegree, and to create statistics about the types of the vertices and edges. All but plotting can run on a server computer without graphical user interface, so it can be used to automate data acquisition and in dynamic web pages as well.

The four Python packages cxnet is based on to achieve these goals are:

- *igraph* for investigating, plotting, storing and loading networks,
- *matplotlib* for plotting diagrams, and
- *python-apt* for obtaining the properties of the Linux packages and the connections among them.

In our centre, the students may participate in a course called "Investigation of complex networks". This gives students the possibility to learn the theoretical basics of network science: the most important measures characterizing the structure of the networks, the models of growing networks, and methods to generate networks with given properties. They learn not only the theory, but also how to analyse networks and write programs using the software packages of the Python language, including the *cxnet* software package. In addition to the network of the Linux software packages, the cxnet package can fetch a lot of archived networks to study from its webpage.

2 Requirements and Installation of the Cxnet Software Package

The cxnet software package is written in the Python language, a general purpose object oriented language [5]. This package is based on several other packages. The most important dependency is the *igraph* network analyser package [6]. It is written mainly in C language to reach its goal, to be able to analyse huge networks fast. It can be used not only in C and C++ programs; it also has extensions for the Python and R languages. The igraph package has implemented most of the algorithms of the complex network science as well as an advanced network visualisation. Earlier version of cxnet based on another network analyzer package was named NetworkX [7]. To plot functions such as the degree distribution we need the *pylab* Python module of the *matplotlib* plotting package [8]. Pylab uses the *array* data type of the *numpy* package to store row vectors of float numbers and the *matplotlib* module to plot functions. Pylab combines these two modules to get functionalities nearly equivalent to that of MATLAB. The aim of our package is not to hide the possibilities of igraph, pylab and numpy, but to help to use them together. The last package is the python-apt [9], which is needed to provide cxnet with information about the software packages of Linux.

These packages can be installed on Windows, Linux and Mac OS X as well, except for the python-apt package. We can create the software package network just on Linux distributions using the APT package management tool, such as Ubuntu and Debian. APT is the standard installation tool of Ubuntu and its detailed description can be found in the next section. It is recommended to install cxnet on Ubuntu Linux distribution version 12.04 or later. On the Ubuntu 12.04,

the numpy, python-apt, ipython, igraph, networkx and matplotlib software packages can be installed using the APT package management tool. The stable and development version of the cxnet package can be downloaded by the git version control system. The up-to-date description of the installation process can be found in the documentation of cxnet [10].

The cxnet package is a rather complex program. Large part of the package can be tested automatically with the unit tests included in the package. Running of these unit tests (and the unit tests of the igraph package) is advisable to test that the package works on that platform as expected.

The cxnet and igraph packages use the unittest module of the standard library of the Python program language for unit tests. This makes possible to check, that the functions returns with the expected value for the given parameters, and raises the expected exceptions for the parameters, that have no sense.

3 The Software Package Network of Linux

Most of the Linux distributions have precompiled binary files for one or more processor architectures. Some distributions, like Ubuntu and Debian, use the same package management format, the *deb* format, developed by the Debian developers. The most convenient way to install these packages is to use the APT package management tool in command line or graphical user interface. APT can handle package repositories stored on the Internet. APT needs to know the dependencies of the packages to install, update or remove them. These dependencies and other connections are stored in compressed text files that can be downloaded from the repositories. In these files the dependencies (the packages a package depends on) of all packages are listed. There are, however, dependencies that are not *real packages*, and they cannot be downloaded; these packages are called *virtual packages*. In Figure 1, the editor package is a virtual package. More packages can provide the same virtual packages, so if one package has the editor as a dependency, one of the packages that provides editor is enough to install. These provisions are stored in the same file as the dependencies, along with other types of connections between packages, such as recommendations and suggestions [11].

In the software package network generated by cxnet, the packages are the vertices and these connections are the edges. It stores two types of packages (real, virtual) and three types of connections (dependencies, recommendation, provision). This is a directed network:

1 if A package *depends* on B package, there is a first type edge from A to B (A→B),

2 if A package *recommends* B package, there is a second type edge from A to B,

3 if A package *provides* the virtual package B, there is a third type edge from B to A.

The direction of the arrow in the third case has been chosen to be retrograde in the sense that in this case the connection information is listed at the package at the arrow head (at the target of the edge). The cxnet package uses colours on the plots to distinguish between the types of the vertices and edges, as can be seen in Fig. 1.

The software package network of the Debian distributions and its evolution are studied in the referenced articles [12, 13]. These analyses include only the real packages and the dependencies. The cxnet package can create and store a more detailed network of the connections between packages. It is a good exercise for the students to study how the properties of the network changes with the presence and absence of the virtual packages.

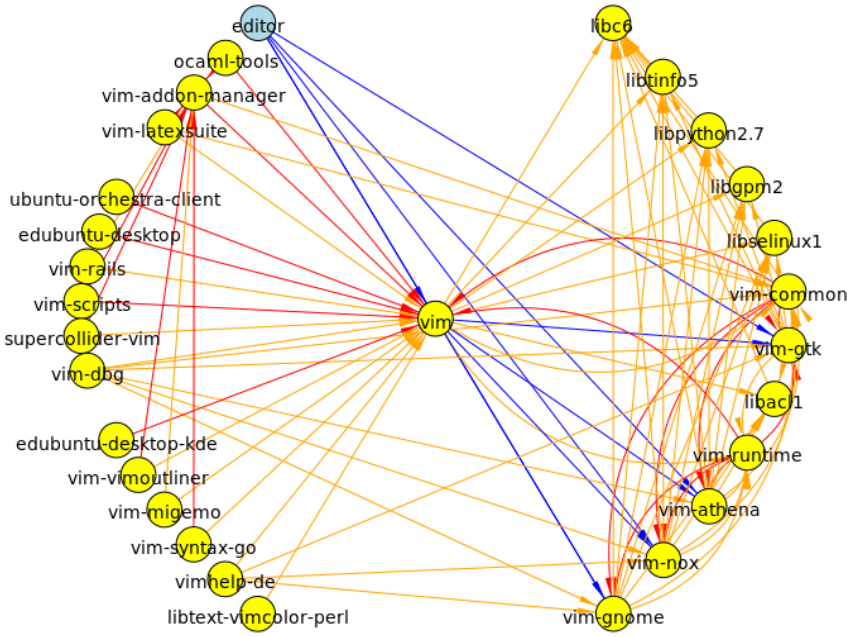

Figure 1
The neighbourhood of the package of the vim text editor

4 Investigating the Structure and the Evolution of Complex Networks

The structure of a diverse set of complex networks can be studied with cxnet. These networks include the software package network detailed earlier, the co-authorship networks of several fields of physics, the Internet at the autonomous system level, part of the Worldwide Web, the neural network of a worm, and the network of protein-protein interaction in yeast. However, cxnet is able to download network files from its webpage. The recent version of igraph (0.6) has its own network repository [14] as an alternative method to fetch networks to study. These networks include directed and undirected networks, bigger or smaller networks, and artificial and natural networks. Most of them are evolving networks, which means that new vertices and edges appear in this networks and some vertices can disappear as well, so the graph that describes the networks changes as the size of these networks grows. The evolution of complex networks can be studied using the thousands of software package networks of several versions of Ubuntu and Debian that have been stored.

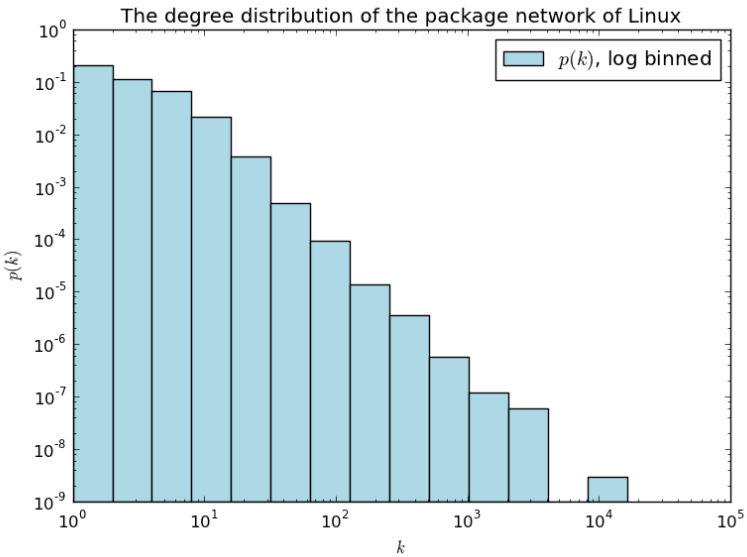

Figure 2

The degree distribution of the software package network. The degree is on the horizontal axis and the probability on the vertical one. Logaritmic binning (detailed in the text) has been used.

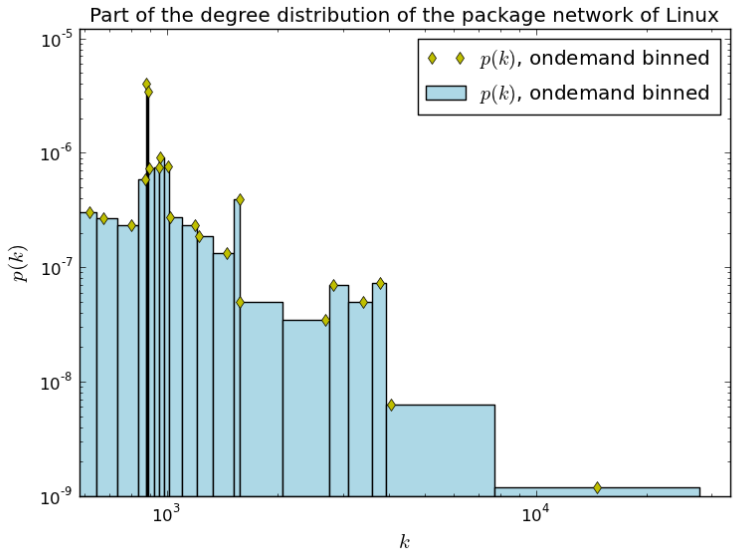

Figure 3
A part of the degree distribution of the software package network. Ondemand binning has been used. Detailed in the section 4.1.

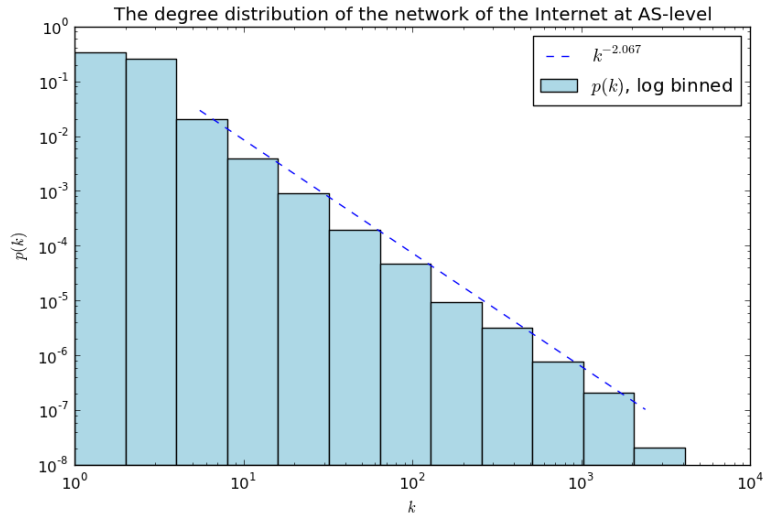

Figure 4
The degree distribution of the Internet. Logaritmic binning has been used.

4.1 Plotting the Degree Distribution

The degree distribution of a network is the p function that assigns to every natural number the probability that a randomly chosen vertex has this degree. The cumulative degree distributions of a network is the P function that assigns to every non-negative number the probability that a randomly chosen vertex has a degree equal or bigger than this value. In equations:

$$p(k) = Prob(\text{degree of the randomly chosen vertex} = k) \quad (1)$$

$$P(k) = Prob(\text{degree of the randomly chosen vertex} \geq k) \quad (2)$$

We can define indegree distribution or outdegree distribution and their cumulative analogously.

The plotting of the degree distribution (with or without the approximate power-law distribution) is easy with the cxnet package, as the examples in the appendix show. We can plot degree distribution or cumulative degree distribution. For the degree distribution we can choose three binning methods. In the first, the width of the bins is one degree. The second is logarithmic binning, which is widely used for plotting the degree distribution of the scale-free networks (Figure 2). The third is the method that is called the "ondemand" method by cxnet. In this method, the width of the bins depends on the degrees of the vertices. There is a division point between the degrees i and j if there is at least 1 vertex with degree i and at least 1 vertex with degree j, but there is no vertex with degrees between i and j. The division point is at the geometric mean of the two degrees ($\sqrt{ij}$). In Figure 3 a part of a degree distribution was plotted with ondemand binning using points and bars. The vertical edges of the bars are at the division points. The abscissas of the diamonds are the existing degrees in the network. In the cxnet program, we can set the binning method with the parameter values "all", "log" and "ondemand", respectively, as can be seen in the appendix. As we can see in Figure 4, the degree distribution of all networks (directed and undirected) can be plotted by the cxnet package.

4.2 The Clustering Coefficient Degree Diagram

The clustering coefficient is a number of the [0, 1] interval defined for the vertices of a network and for the whole network. For a vertex this coefficient is 1 if each of its neighbours is connected to the other neighbours and 0 if none of its neighbours are connected. The exact definition of the clustering coefficient of a vertex *i* is the quotient of the number E_i of existing edges between its neighbours and the maximal number of edges that could be between neighbours. In this definition, the network is handled as an undirected and simple one. The simple graph has no multiple edges between vertices and has no loop edge having the same vertex at its ends. The maximal edges between the k_i neighbours of vertex i is $(k_i \times (k_i - 1))/2$, so the clustering coefficient of the vertex *i* is

$$C_i = \frac{2E_i}{k_i \times (k_i - 1)} \tag{3}$$

The cxnet package is able to plot the clustering coefficient as a function of degree (Figures 5 and 6). The ordinate of the point in this plot is the arithmetic mean of the clustering coefficient of the vertices with the same degree. This function was investigated in the articles [15, 16] and it was concluded that the clustering coefficient in many real networks decreases roughly inversely proportional to the degree, and this is an indication of the hierarchical property of these networks.

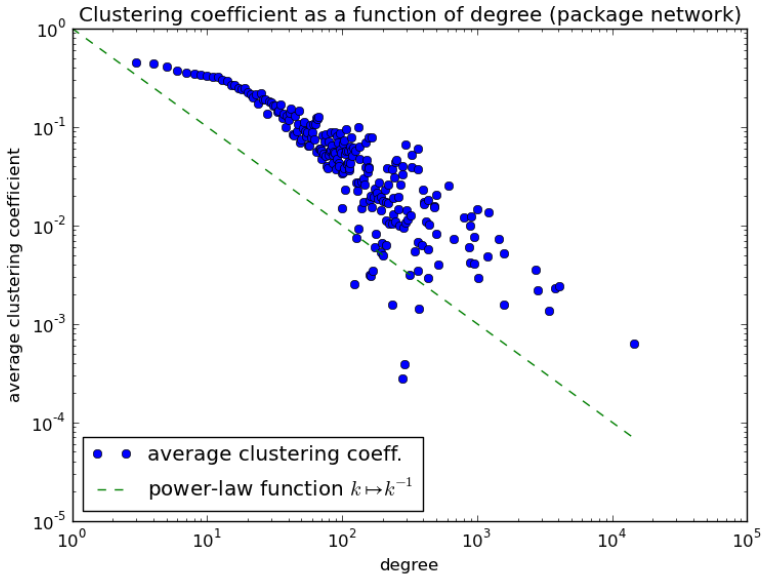

Figure 5

The average clustering coefficient of the software package network as the function of the degree

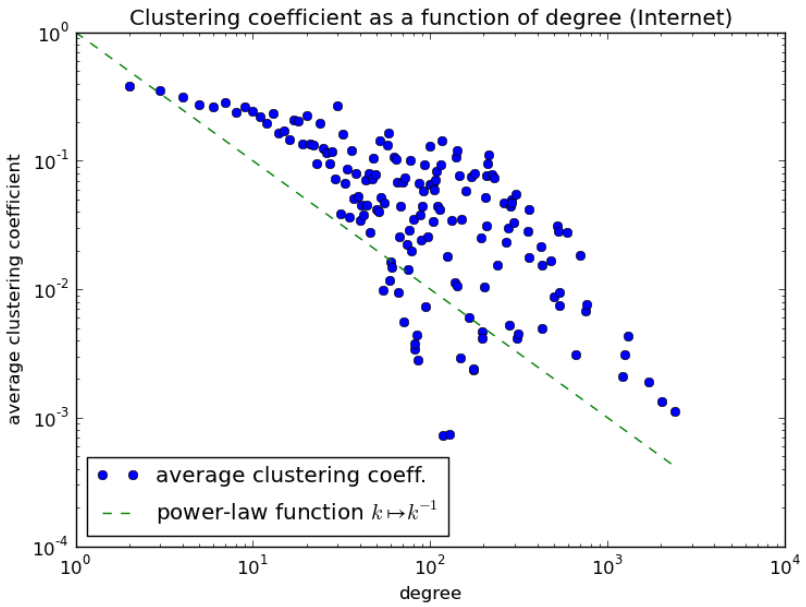

Figure 6

The average clustering coefficient of the Internet as the function of the degree

The clustering coefficient of the network is the arithmetic mean of the clustering coefficients of the vertices. In almost all networks the probability that there is an edge between two randomly chosen vertices is far less than the probability in the case that the two randomly chosen vertices would be chosen with the condition that the two vertices should have a common neighbour. In the example of a friendship network, the possibility that my two friends are friends of each other is more than the possibility that two members chosen from this network are friends. This property can be quantified with the clustering coefficient. The clustering coefficient of real networks is some order of magnitude higher than it would be if the edges would connect vertices randomly.

The clustering coefficient has no meaning for vertices with a degree of zero or one, so there are two possibilities to calculate the clustering coefficient of the network. In the igraph and cxnet packages, this coefficient is calculated as the average for the vertices with a degree more than one. Another possibility is also common in the literature: defining a value – usually 1 – as the clustering coefficient of these vertices, and the clustering coefficient of the network includes all vertices.

5 Documentation and Educational Materials

The cxnet package has a fairly up-to-date documentation on the webpage of the cxnet package [10]. This includes a detailed tutorial (English and Hungarian) and package reference for this package (English).

The educational materials for the "Complex networks" course, which was developed in our centre, are mainly in Hungarian. The syllabus and some video tutorials can be found on the web. Students can access tests on the e-learning (Moodle) website of Óbuda University. These tests cover the theory of complex networks, the software development in Python and the usage of the igraph, cxnet and pylab modules. Our earlier article [17] details some of the properties of the software package network, which can be studied with the cxnet module.

A book by Mark Newman [18] is big help for teachers and ambitious students. There are also some papers available for free on the web that include a summary of the most important concepts of complex networks as well as the properties of many real networks [1, 19, 20]. There are some other articles about more specific fields of complex networks that are also useful in education [15, 21–25].

Conclusions

The Python language with the cxnet package can be used to study networks easily, so it is an ideal tool for courses in higher education. On deb-package-based Linux distributions, the cxnet package is able to create the software package network of Linux, a large and fast evolving directed network. The most important network properties and the laws of network evolution can be studied with the cxnet packages using this and many other archived networks. This package is unique in the sense that it unifies some of the benefits of other packages without their drawbacks. These advantages are:

- the fast computation of the network properties for networks with millions of vertices as well,
- the plotting of the degree distribution with several binning methods or as a cumulative distribution,
- the plotting of the average clustering coefficient against the degree, to decide whether the network is hierarchical or not,
- the easy interactive investigation of the networks with the Python shell,
- the possibility to create the software package network of Linux.

As it is written in Python, the cxnet package gives the possibility for the students to implement algorithms easily in object oriented or procedural style.

Acknowledgement

I would thank to Tamás Nepusz for his help in using igraph. I would thank to Marianna Machata for her linguistic help and advice.

Appendix: Examples on the usage of the cxnet package

This is just a short introduction of the possibilities of cxnet. More detailed documentation can be found on the webpage of the cxnet package [10]. We need to start one of the python shells. We recommend to use the igraph with the `--pylab` option.

As the Network class of the cxnet is derived from the Graph class of igraph package, we can use all of its functions. The member functions (methods) of Network added in the cxnet starts with "cw", so we can distinguish between cxnet and igraph functions. We need to import the cxnet package, and then we can create or load the software package network:

```
import cxnet
net = cxnet.debnetwork()
```

or

```
net = cxnet.load_netdata("ubuntu-12.04-packages-2012-07-18_21.06GMT.graphmlz")
```

Then we can count the number of vertices and edges:

```
net.vcount(), net.ecount()
```

We can get the maximal degree and maximal indegree in the network:

```
max(net.degree()), max(net.indegree())
```

We can plot the cumulative degree distribution of the package network (Figure 7).

```
dd.cumulative_plot(with_powerlaw=True)
legend()
```

We can create the degree distribution and plot with chosen binning methods and save the figure (Figure 8). As we want to create another figure we need to close the figure window, or close the figure:

```
clf()                                          # close the figure
dd = net.cxdegdist()
dd.set_binning("log")                          # logarithmic binning
dd.loglog()                          # plot using log scales on each axes
dd.set_binning("ondemand")
dd.loglog()
savefig("degdist.png")                         # save the figure into a file
dd.plot_powerlaw()       # power-law function with the calculated exponent
print(dd.gamma)                  # print the absolute value of the exponent
```

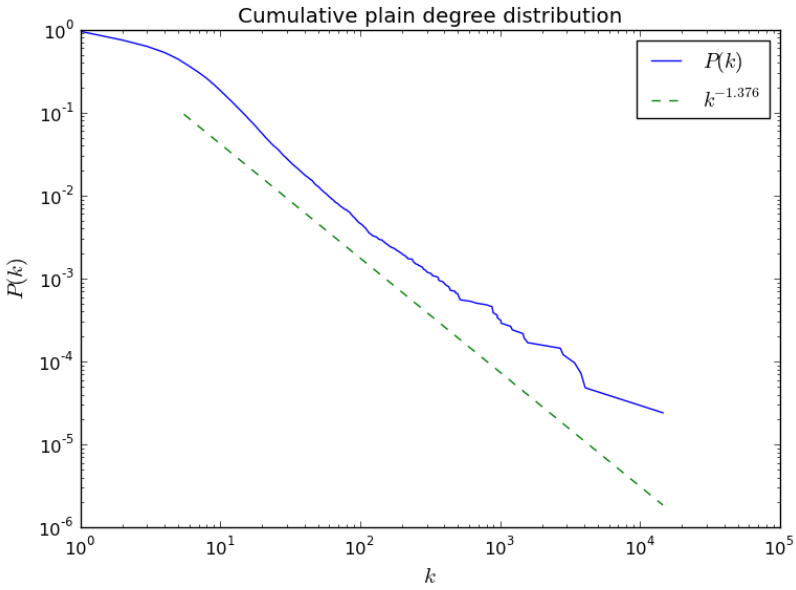

Figure 7
The cumulative degree distribution with the approximate power-law function

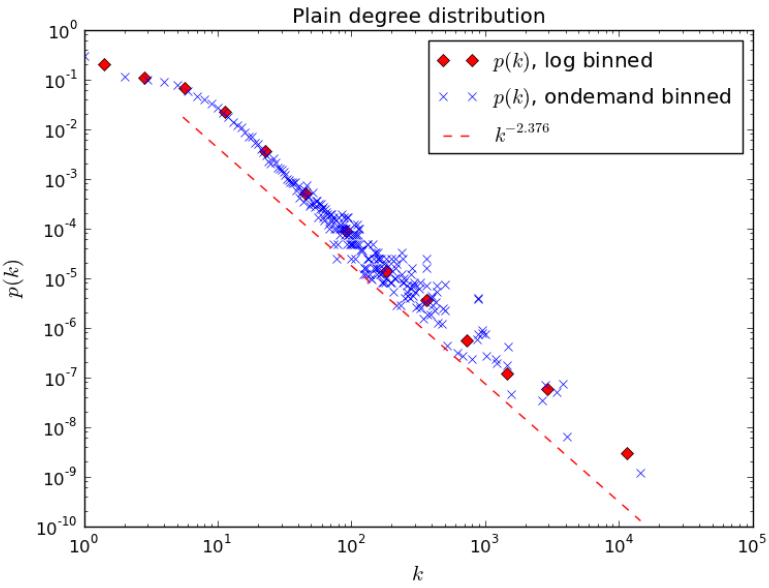

Figure 8
The degree distribution plotted using two binning methods and the approximate power-law function

If we typed dd = net.cxdegdist("in") in the previous example, the in-degree distribution would be plotted.

The bar plot allows to see the division points of the bins (Figures 2, 3 and 4). This plot can be plotted as follows:

```
clf()
dd.set_binning("log")
dd.bar_loglog()                          # logarithmic scales on both axes
legend()
```

We can list the packages with the largest degree (indegree, outdegree) as follows:

```
net.cxlargest_degree()
net.cxlargest_degree("in")
```

We can plot the average degree as a function of degree as follows (Figures 5 and 6):

```
net.cxclustering_degree_plot()
```

We can plot the neighbourhood of a package with one line (Figure 1):

```
net.cxneighborhood("vim")
```

The packages depending on the package called vim are at left, and the packages that vim depends on are at right. The vertices and edges are coloured according to its types. The blue vertex is virtual package, the yellow vertices are real packages. The gold edges are dependencies, the red ones recommendations and the blue ones provisions.

References

[1] Albert, R., Barabási, A.: Statistical Mechanics of Complex Networks, Reviews of Modern Physics, Vol. 74, No. 1, 2002, pp. 47-97

[2] Batagelj, V., Andrej M.: Pajek—Analysis and Visualization of Large Networks. *Graph Drawing*. Springer Berlin/Heidelberg, 2002

[3] Börner, K., et al. Rete-Netzwerk-Red: Analyzing and Visualizing Scholarly Networks using the Network Workbench Tool. *Scientometrics*, 2010, 83: 863-876

[4] Siek, J. G., Lee, L. Q., Lumsdaine, A.: *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley Professional, 2001

[5] Mark Summerfield: Programming in Python 3: A Complete Introduction to the Python Language, Addison-Wesley Professional, 2010, see also http://python.org

[6] Csárdi, G., Nepusz, T.: The Igraph Software Package for Complex Network research, InterJournal Complex Systems, 2006, Manuscript Number. 1695

[7] Hagberg A. A., Schult D. A., Swart P. J.: Exploring Network Structure, Dynamics, and Function using NetworkX. , *Proceedings of the 7th Python*

in Science Conference (SciPy 2008), pp. 11-15, Pasadena, CA USA, August 2008

[8] Tosi, S.: Matplotlib for Python Developers, Packt Publishing, 2009, see also http://matplotlib.sourceforge.net

[9] The documentation of the python-apt program, http://apt.alioth.debian.org/python-apt-doc

[10] The homepage of the cxnet program, http://django.arek.uni-obuda.hu/cxnet

[11] Hertzog, R., Mas, R.: The Debian Administrator's Handbook, Freexian SARL, 2012, Chapter 5, see also, http://www.debian.org/doc/manuals/debian-faq/ch-pkg_basics.en.html

[12] Maillart, T, Sornette, D., Spaeth, S., von Krogh, G.: Empirical Tests of Zipf's Law Mechanism in Open Source Linux Distribution, Phys. Rev. Lett., Vol. 101, 2008, p. 218701, doi:10.1103/PhysRevLett.101.218701

[13] de Sousa, O. F., de Menezes, M. A., Penna, T. J. P. Analysis of the Package Dependency on Debian GNU/Linux. *Journal of Computational Interdisciplinary Sciences*, Vol. 1, 2009, pp. 127-133

[14] The Nexus homepage http://nexus.igraph.org

[15] Ravasz, E., Barabási, A.: Hierarchical Organization in Complex Networks, Physical Review E, Vol. 67, 2003, pp. 026112, doi:10.1103/PhysRevE.67.026112

[16] Vázquez, A., Pastor-Satorras, R., Vespignani, A.: Large-Scale Topological and Dynamical Properties of the Internet, Physical Review E, Vol. 65, No. 6, 2002, pp. 066130

[17] Horváth, A.: Studying Complex Networks with Cxnet, Acta Physica Debrecina, Vol. XLIV, 2010, pp. 37-47

[18] Newman, M. E. J: Networks, An Introduction, Oxford University Press, Oxford, 2010.

[19] M. E. J. Newman, The Structure and Function of Complex Networks, SIAM Review, Vol. 45, No. 2, 2003, pp. 167-256

[20] Dorogovtsev, S. N., Goltsev, A. V. and Mendes, J. F. F.: Critical Phenomena in Complex Networks, Rev. Mod. Phys., Vol. 80, No. 4, pp. 1275-1335

[21] Colizza, V., Barrat, A., Barthelemy, M., Vespignani, A.: Predictability and Epidemic Pathways in Global Outbreaks of Infectious Diseases: the SARS case study, BMC Medicine, Vol. 5, 2007

[22] Kitsak, M., Havlin, S., Paul, G., Riccaboni, M., Pammolli, F., Stanley, H. E.: Betweenness Centrality of Fractal and Nonfractal Scale-Free Model

Networks and Tests on Real Networks, Phys. Rev. E, Vol. 75, No. 5, Part 2, 2007

[23] Myers C. R.: Software Systems as Complex Networks: Structure, Function, and Evolvability of Software Collaboration Graphs, Phys. Rew. E, Vol. 68, 2003

[24] Zlatic V., Bozicevic, M., Stefancic, H., Domazet, M.: Wikipedias: Collaborative Web-based Encyclopedias as Complex Networks, Phys. Rew. E, Vol. 74, 2006

[25] Liu, Y.-Y., Slotine J.-J., Barabási, A.-L.: Controllability of Complex Networks, Nature, Vol. 473, No. 7346, 2011, pp. 167-173