Bioinformatics III

Prof. Dr. Volkhard Helms
Maryam Nazarieh, Duy Nguyen, Thorsten Will
Winter Semester 2015/2016

Saarland University
Chair for Computational Biology

Exercise Sheet 1

Due: October 30, 2013 13:15

1 Introduction to Python and some Network Properties

To keep the complexity down for this assignment, we focus on random networks and provide several hints and code stubs to ease your the work in respect of the upcoming exercises. Therefore this assignment is split into separate modules which can (and have to) be reused in later assignments.

General remarks

Since the first assignments are based on each other, it is recommended you visit the tutorials if there are any problems early on. Otherwise you are likely to encounter difficulties when working on the next assignment sheets.

Python

For the programming tasks of those assignments the programming language Python (version 2.x) is used. The aim of this assignment is to learn basic Python concepts and to get familiar with network properties.
Line indentation is **crucial** in Python. All code templates you are given use 4 spaces as tabs, adjust your editor/IDE accordingly to avoid problems.
Furthermore, since it is the most common package for that, you should use the 'matplotlib' library for all plotting tasks. Linux users are advised to install the library using their package management system, OSX users may find this link useful `http://fonnesbeck.github.io/ScipySuperpack/` and Windows users should take a look here: `http://www.lfd.uci.edu/~gohlke/pythonlibs/`.

Submission process

(a) You are advised to work in groups of *two* people. If necessary, we will suggest teammates after the first assignment has been submitted. You may submit your solutions in english or german language.

(b) Submit your solutions on paper, hand-written or printed at the *beginning* of the lecture in the lecture hall or in Room 302, both E2.1. Alternatively, you may send an email with a **single** PDF attachment to thorsten.will@bioinformatik.uni-saarland.de. Also attach your source code, if appropriate. Late submissions will not be considered. Please note that later sheets need to be handed in to different teaching assistants.

(c) Include source code listings into the submitted document, we will **not** merge and layout your source code. If relevant sources are missing in the exercise sheet, they will not be graded.

(d) Feel free to use the LaTeX-template provided in the supplementary material.

(e) Do not forget to mention your names/matriculation numbers. :)

Discussion of this exercise will be Monday, Nov 2nd at 12:00 c.t. in the lecture room (E2.1 007).

Exercise 1.1: Constructing a network (50 Points)

We start by building a simple data structure that represents a network in general. Therefore we first (a) implement a **Node**-class that represents a single node in the network and stores its links to other nodes, we then (b) define an abstract superclass that stores all nodes of a network, which can later be used to derive our different network types from, and (c) implement a subclass that actually builds a random network.

We provide templates for **all** classes that you need to implement in the supplementary material. Most method implementations are very short. Feel free to extend your interface as needed.

(a) Implement the missing methods of the **Node**-class in **Node.py**.

(b) We need a network class that stores all nodes of a network. Use a Python-dictionary to store all nodes in a `key` $\rightarrow$ `node` fashion within a class variable called `self.nodes`. Implement it in **AbstractNetwork.py**.

Hint:

- An example how to use python dictionaries in this context:

```
# init dictionary
nodes = {}

# create node with id 0
node = Node(0)

# add entry to dictionary
nodes[node.id] = node
```

(c) Implement an algorithm to set up a random graph in the initialization method of a **RandomNetwork** object in **RandomNetwork.py**. As you will see in the code template, **RandomNetwork** extends **AbstractNetwork**. Building a random network is simple: first create a given amount of nodes, then set a given amount of links between them. To set each link, choose two nodes by random that are not yet connected and establish one.
In a network of n nodes, what is the minimal amount of links that are needed to connect the network? What is the maximal amount of links that can be placed uniquely? The constructor of the class should throw a `ValueError` if an invalid number of edges should be placed.

Hint:

- When adding a link between two nodes i and k, do not forget the entry for $k \rightarrow i$.

Exercise 1.2: Degree distribution of random networks (50 Points)

(a) Write a class that determines and prints the degree distribution for a network created above. Again, a stub is given in the supplement. Why are normalized measures generally advantageous in practice?

Hints:

- First determine the biggest number of links that occurs in your network and initialize a list of that size with zero values. An example to do this with 10 entries is:

```
histogram = [0] * 10
```

  This array then holds the degree distribution.
- Now loop over the network list and increment the cells indexed with the corresponding degree.
- Normalize the histogram to obtain a valid probability distribution which can be retrieved by the **getNormalizedDistribution()** method.

(b) To visually assess that the degree distribution of a random network obeys the Poisson distribution $P(k)$ with a mean value of λ, you need to implement a few methods in **Tools.py**:

- First implement the method **poisson(k, lambda)** which returns $P(k)$ for given λ according to

$$P(k) = \frac{\lambda^k}{k!} e^{-\lambda}.$$

  Then, in **getPoissonDistributionHistogram(num_nodes, num_links, k)** you determine λ from the numbers of nodes and links and compute the Poisson distribution for a sufficiently large range of k. The structure of the output should match the output of (a).

  Tip:
  In case you encounter numerical problems calculating the factorial, consider an iterative solution, e.g.

$$P(0) = e^{-\lambda}, P(1) = \frac{\lambda}{1} \cdot P(0), \ldots, P(n) = \frac{\lambda}{n} \cdot P(n-1)$$

- The file also contains a simple function that plots several distributions for comparison. To get visually pleasing plots ensure that all distributions that are plotted together have the same length. This can be done by appropriately extending the shorter ones. Why does this happen and how do you need to "fill" the shorter distributions? Are the ranges of the discrete distributions we obtain in (c) deterministic in our case?
  Furthermore, two important annotations are still missing there, fill the two empty strings correctly.

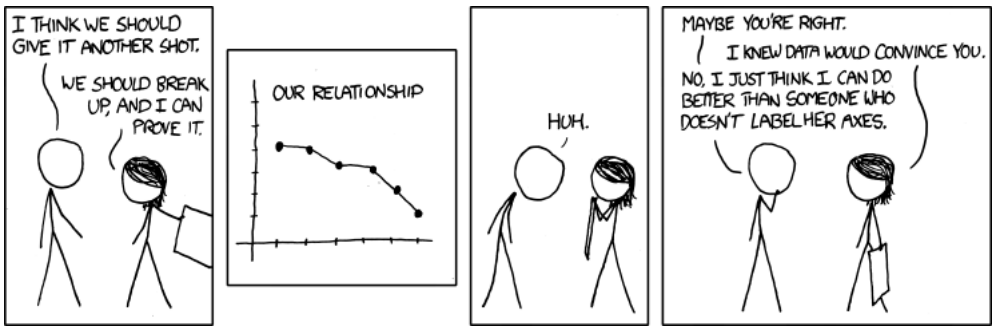
(c) Use the provided script **createAndPlotNetworks.py** that sets up the following networks (each given in the form nodes/links) and plots $P(k)$ together with the degree distributions of the random networks.. Save the plots and attach them to your solution.

Plot 1:	50/100	500/1000	5000/10000	50000/100000
Plot 2:	20000/5000	20000/17000	20000/40000	20000/70000

Describe both plots and **explain** the difference (or the trend) between the different parameter sets.

General hint:

- Never forget to label the axes! (This hint applies to the entire lecture.)



Have fun!