Bioinformatics III

Prof. Dr. Volkhard Helms
Markus Hollander, Thorsten Will, Nicolas Künzel,
Andreas Denger, Dr. Pratiti Bhadra
Winter Semester 2019/2020

Saarland University
Chair for Computational Biology

Exercise Sheet 1

Due: October 25, 2019 13:15

Introduction to Python and Network Properties

The first part of the lecture is about networks and their properties. To keep complexity down for this assignment, we focus on random networks and provide several hints and code stubs to ease you into building and analysing networks.

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

Submission Process and Rules

(a) **Groups:** 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.

(b) **Deadline:** The deadline for submission is given on each assignment sheet. Late submissions will **not** be considered.

(c) **Submission:** Submit your solutions on paper, hand–written or printed at the **beginning** of the lecture in the lecture hall (room 007) or in room 309, both in the bioinformatics building (E2.1). Alternatively, you may send an email with a **single PDF** attachment to markus-hollander@web.de. Submissions as Word– or simple text files will **not** be considered.

Always attach your source code as a **single .zip file** to that email.

(d) **Format:** Your submissions should be readable and contain your names and matriculation numbers. The submitted document should include source code listings **with comments**. We will **not** merge and layout your source code for you.

You can use the LaTeX–template provided in the supplementary material to format your submission. If you are new to LaTeX, you can get started with overleaf.com. It provides free tutorials, online storage, a PDF autocompiler, and simultaneous editing by multiple people.

(e) **Tutors:** Please note that later sheets need to be sent to different teaching assistants. The respective email address will be given at the top of the exercise sheet below the due date. If you have questions, you can also use that email address to ask the current tutor.

(f) **Plagiarism:** Solutions and code should be written by you in your own words. Parts that were copied from other groups, the internet or submissions of previous years will be marked with 0 points.

(g) **First tutorial:** Discussion of this exercise will take place on Monday, October 28 at 12:15 in the lecture hall (building E2.1 room 007).

Python

The programming language Python (version 3.x) is used for the programming tasks of the first couple of assignments. The goal of this assignment is to learn basic Python concepts and to get familiar with network properties. Current Python 3 versions can be obtained from python.org or be installed via macOS and Linux package management systems.

Note that line indentation is crucial in Python! All code templates you are given use 4 spaces as tabs, so adjust your editor or IDE accordingly to avoid problems. An example of a good, free Python IDE is PyCharm.

Furthermore, you need to install the popular plotting library "matplotlib". Linux users are advised to install the library using their package management system. macOS users may find `http://fonnesbeck.github.io/ScipySuperpack/` useful and Windows users should take a look at `http://www.lfd.uci.edu/~gohlke/pythonlibs/`.

Exercise 1.1: Network Construction (50 Points)

We start by building a simple data structure that represents a general network. Therefore we first implement a **Node**–class that represents a single node in the network and stores edges to its neighbouring nodes. Then, we define an abstract super–class that provides basic network functionality and which can later be used to derive our different network types. Finally, we implement a sub–class 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 to suit your needs.

(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` → `node` fashion within a class variable called `self.nodes`. Implement all functions of the **AbstractNetwork**–class in **abstract_network.py**.

Hint: An example of using Python dictionaries:

```
# initialise a dictionary called nodes
nodes = {}

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

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

(c) Next, we want to build random networks. Our **RandomNetwork**–class inherits basic network functionality from the **AbstractNetwork**–class. All that is left to do is implement the initialisation method of **RandomNetwork** in **random_network.py**.

Exercise 1.2: Degree Distribution of Random Networks (50 Points)

(a) Implement the **DegreeDistribution**–class in **degree_distribution.py** that determines the normalised degree distribution of a given network.

(1) First, determine the largest degree occurring in the network and initialise a list of that size with zeros, which represents the degree histogram. An example of doing this for 10 entries is:

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

This array then holds the degree distribution.

(2) Next, count how often each degree occurs in the network by looping over the nodes in the network and increment the histogram cells indexed with the corresponding node degree.

(3) Finally, normalise the histogram with the number of nodes in the network to obtain a valid probability distribution.

(b) To visually assess if 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**:

(1) Implement the method **poisson(**k**,** λ**)**, which returns $P(k)$ for given λ:

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

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

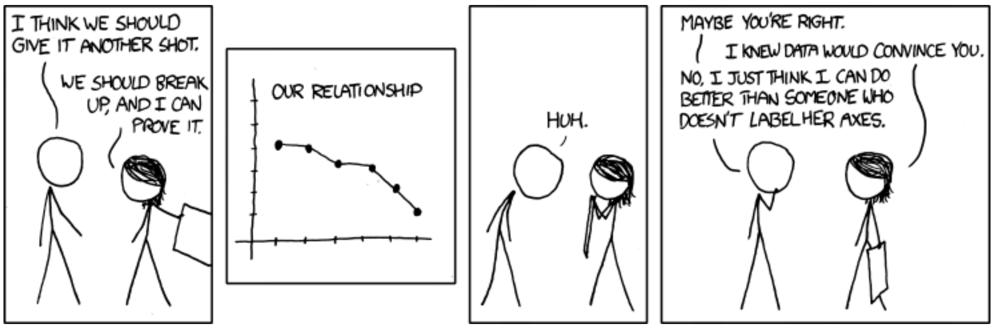
$$P(0) = e^{-\lambda} \text{ and } P(k) = \frac{\lambda}{k} \cdot P(k-1).$$

(2) In **poisson_histogram(**n**,** m**,** max**)** you first determine λ from the numbers of nodes n and number of edges m:

$$\lambda = \frac{2m}{n},$$

and then compute the Poisson distribution for all $k \in \{0, \ldots, max\}$. The structure of the output should match the output of (a).

(3) The file also contains the simple function **plot_distribution_comparison** that plots several distributions at once 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. Furthermore, two important annotations are still missing there, fill the two empty strings correctly.


I THINK WE SHOULD GIVE IT ANOTHER SHOT.
WE SHOULD BREAK UP, AND I CAN PROVE IT.
OUR RELATIONSHIP
HUH.
MAYBE YOU'RE RIGHT.
I KNEW DATA WOULD CONVINCE YOU.
NO, I JUST THINK I CAN DO BETTER THAN SOMEONE WHO DOESN'T LABEL HER AXES.

Question: Why are some distributions shorter than others and how do you need to "fill" the shorter distributions?

(c) Run the provided script **main.py**, which sets up the following networks and plots the Poisson distributions together with the degree distributions of the random networks. Save the plots and attach them to your solution.

	number of nodes / edges			
Plot 1:	50/100	500/1,000	5,000/10,000	50,000/100,000
Plot 2:	20,000/5,000	20,000/17,000	20,000/40,000	20,000/70,000

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

Have fun!