TIMOTHÉE POISOT

BIPY – DOCUMENTATION

Contents

Introduction

`bipy` is a set of functions written in Python, whose goal is to make it easy to analyze and visualize bipartite networks. This document describes the use of the `bipy`. This documentation is valid only for the versions before the 1.0 release (scheduled to occur sometime in the fall or winter of 2011). The aim of this documentation is to present only the "high-level" functions, i.e. these that the user will need to perform the analyses (although `bipy` was conceived to automate most of the analyses, so there are a very small number of functions that need to be known). In addition, some background about the measures used and the most important references will be given.

Installing `bipy`

`bipy` can be obtained on *GitHub* (http://github.com/tpoisot/bipy). There are a number of required additional software. Obviously, `bipy` requires a Python installation, preferentially 2.6 or 2.7. For those of you with a Mac, you need to download the version from python.org (http://www.python.org/getit/).

Once this is done,

Do I need to know Python ?

This is a good question! If you just want to have basic informations about your data, the short answer is **no**. `bipy` was designed from the beginning to automate most of the calculations, and use the most robust statistics available, so you don't have to play too much with the code. Most of what you will have to do is start python, load your data, and use the summary functions.

If you want to use `bipy` for more complicated analyses, then **yes**, learning python will definitely be a huge plus. But if you are willing to adopt a computational approach, chances are you already know a programming language, and you should be able to pick up python really quick.

Loading data

This chapter describes how to load data in a format that can be understood by `bipy`. There are two different ways to load data in `bipy`: *via* a text file, or *via* the web [1]. We will cover both in this chapter.

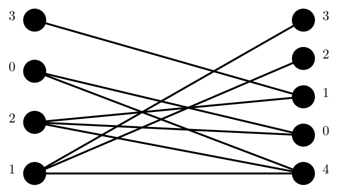
[1] Actually, there exists a third way to load data, namely the interaction with the EWDB – given that this database is still highly experimental, and that the server on which it is developed is not ready to support real usage, this will be introduced in future versions of `bipy`.

Reading data from a local file

Most of the time, this will be the default solution. `bipy` assumes that a bipartite network is nothing more than a matrix, in which each square represents the strength of the interaction between a species from the top trophic level, and a species from the bottom trophic level. Thus, a bipartite network is defined by a matrix $\mathbf{M}$, in which $\mathbf{M}_{ij}$ is the strength of the link between upper trophic level species i and lower trophic level species j. It is assumed that the rows of $\mathbf{M}$ are the upper trophic level species. This is a notational convention, but we will stick to it through all the documentation.

The advantage of this organization of the data is that it can easily be stored in a text file, with each interaction strength separated by a space, and the different organisms separated by a new line. Thus, the file

```
1 0 0 0 1
0 0 1 1 1
1 1 0 0 1
0 1 0 0 0
```



described the network illustrated in the margin. Of course, the values within the file can be anything (as long as they are numeric): zeroes and ones, number of visits, predation rate, percent of the diet, and so on.

Assuming that your data are in this format, and the file is called `mydata.web`, you can load them with

```
web = readweb('mydata.web')
```

This will only create a 2-dimensional array. The power of `bipy` lies in its ability to automate the calculations, so it is better to write

```
web = bipartite(readweb('mydata.web'))
```

Most of the functions will need an object returned by the `bipartite` function, so it is good practice to use it [2]. At this point, the most difficult part is done. Your data are loaded, and you will be able to see what story they tell using the metrics described in the other chapters.

[2] The actual reason is that most of the networks statistics will need simple informations, such as number of links per species, connectance, size of the network. The `bipartite` class calculate all of them once and for all, so this avoid re-calculating the same information several times.

If your data are not in the "correct" format, i.e. the top level organisms are in columns, do not panic. You just need to pass a supplementary argument `t` to the `bipartite` function:

```
web = bipartite(readweb('mydata.web'),t=True)
```

This will the the `bipy` that your data need to be transposed before going on with the analysis.

Reading data from the web

The alternative way is to load data stored on the internet. `bipy` is able to do so by using the `readRemoteWeb` function, which requires three arguments. The first one is the URL of a text file, which stores the interaction matrix. The second are two booleans, telling if the network should be transformed using `bipartite` (true by default), and if the network should be transposed (false by default).

```
web = readRemoteWeb('http://server/mydata.web'),
    True, False)
```

Adding species names

Adding bibliographical references

Species–level metrics

Network–level metrics

Null models

Graphics

Example scripts