# *pymfinder*: Tool Guide

Bernat Bramon Mora[1], Alyssa R. Cirtwill[2], Daniel B. Stouffer [1]

[1]Centre for Integrative Ecology, School of Biological Sciences, University of Canterbury, Christchurch, New Zealand

[2]Department of Physics, Chemistry, and Biology (IFM), Linköping University, Linköping, Sweden

# 1 General information

**Description:** *pymfinder* is a Python package designed to detect motifs in complex networks and define the roles of nodes and links using these motifs. Both weighted and binary networks can be analyzed. At its core, *pymfinder* is a combination of Python methods for network-motif analysis as well as a Python wrapper for the original *mfinder* version 1.2 written in C and available at http://www.weizmann.ac.il/mcb/UriAlon/. This code has been included and modified here with the explicit consent of Nadav Kashtan, the author of mfinder 1.2.
**License:** MIT License (2018)
**Version info:** v1.0
**Availability:** https://github.com/stoufferlab/pymfinder
**Platforms:** Windows, Linux, Mac OSX. *pymfinder* will require you to have the Python modules Numpy and Setuptools installed in your machine. Following recommendations for mfinder, large and dense networks (>10 000 nodes) require a computer with at least 512 Mbyte RAM in order to calculate motif frequencies. Calculating node or link roles will require greater resources. The analysis of motifs bigger then 8 nodes is not recommended.

# 2 How to use *pymfinder*

## 2.1 Download and installation

### 2.1.1 Download

*pymfinder* can be downloaded from https://github.com/stoufferlab/pymfinder. Please make sure you cite Bramon Mora et. al. (1) if you decide to use *pymfinder*.

### 2.1.2 Installation

Installation within a command-line terminal should be straightforward using the function 'setup.py' included in the *pymfinder* package. After navigating to the directory containing the package, run:

```
python setup.py install
```

If an error message of 'Permission denied' or similar is returned, run:

```
python setup.py install --user
```

This will install *pymfinder* locally rather than in the global Python site-packages or dist-packages directory.

If you are using Python 3, you should switch to the branch pymfinder-python3 of the Github repository.

### 2.1.3 Checking installation

After installation, running the included test suite is strongly encouraged. This may be accomplished by running:

```
python setup.py test
```

## 2.2 Basic usage

## 2.3 Input file format

Input network file format should be in simple '.txt' format. Species names may be given as text or integers but should **not** include spaces. Each edge should be represented by a line of the following format:
`<source node><target node>`

Example:

```
1 2
3 1
Salmo_trutta midge
Corvus_corax Salmo_trutta
```

If interaction strengths are known, they can be passed to *pymfinder* in the input file. In this case, each edge should be represented by a line with the format:
`<source node><target node><interaction strength>`

Example:

```
1 2 1
3 1 2.5
Salmo_trutta midge 0.005
Corvus_corax Salmo_trutta 3
```

### 2.3.1 Function call and arguments

All of the functions within *pymfinder* can be called using the same framework. Within a Python environment, first import the *pymfinder* package using, for example:

```
import pymfinder as py
```

The motif structure, motif participation, and motif roles for the network can then be calculated simultaneously using:

```
results = py.pymfinder(network,
                        links=False,
                        motifsize = 3,
                        stoufferIDs = None,
                        allmotifs = False,
                        nrandomizations = 0,
                        randomize = False,
                        usemetropolis = False,
                        networktype = "unipartite")
```

The *pymfinder* function call includes the following arguments:

- **network**: This can be a path to a network file, a list of interactions or a NetworkStats object. No default given (see the description in the article presenting the software).
- **links**: Determines whether or not to calculate statistics for links as well as nodes. If **links=True**, link participation and roles will be calculated. Defaults to **links=False**.
- **motifsize**: Size of motifs to be calculated. Defaults to **motifsize=3**. There are 13 possible three-species motifs for unipartite networks (Fig. 1). For bipartite networks, there are only four three-species motifs (Fig. 2) and a larger motif size may be necessary. If one needs to analyze the motif structure or the motif participation of motifs $> 3$ nodes in unipartite networks or $> 6$ nodes in bipartite networks, 'motif_structure' and 'motif_participation' need to be run independently.
- **stoufferIDs**: Determines whether to label motifs following Stouffer et al. (2) or based on the representation of the adjacency matrix of the motif as a binary integer, following the original *mfinder* (see list_motifs() and print_motifs() to identify motifs based on motif ID). If **stoufferIDs=True**, labels will be as in Stouffer et al. (2) when printing the pymfinder output. Defaults to **stoufferIDs=False**.
- **allmotifs**: If true, displays results for all possible motifs regardless of whether all have been observed. If false, displays only results for motifs observed in the network. Defaults to **allmotifs=False**.

- **nrandomizations**: Number of random networks with which to compare the observed network. Defaults to 0 (no randomizations performed).

- **randomize**: Determines whether or not to randomize the network before analyzing the participation and roles of the nodes and links in the network. If false, no randomization will be applied to the network. Defaults to **randomize=False**.

- **usemetropolis**: If randomizations are to be performed, determines whether to use the Metropolis algorithm. If Metropolis is not used, *pymfinder* uses an MCMC algorithm to shuffle the original network while preserving in- and out-degrees of nodes. Defaults to **usemetropolis=False** (MCMC-based randomizations).

- **networktype**: Indicates whether the network is unipartite (all species may interact with all other species) or bipartite (species are divided into two groups and may interact between groups but not within a group). Defaults to **networktype="unipartite"**.

- **weighted**: If true, the motif analysis will account for the weight of the interactions. Defaults to **weighted=False**.

### 2.3.2 Functions

The *pymfinder* function call references three subordinate functions: *motif_structure*, *motif_participation*, and *motif_roles*. Each subordinate function may also be called independently if the full output from *pymfinder* is not required. When *pymfinder* is called, the subordinate functions are run in order (*motif_structure* then *motif_participation* then *motif_roles*). If a subordinate function is called directly, any preceding function will also be called. That is, calling *motif_structure* returns only the motif structure output but calling *motif_participation* returns the motif participation and motif structure output. Note that the three functions differ in the way in which they handle interaction weights.

*motif_structure* calculates the motif profile of the network. Arguments passable to *motif_structure* are the same as those for *pymfinder*, except that the **links** argument is not relevant. The same motif profile will be returned whether or not **links=True**. If **weighted=True**, the weight of all interactions forming each motif will also be considered (see the description in the article presenting the software). The characterization of the weight of a motif can be changed using the functional argument **fweight**. Such function **fweight** needs to take a Python list as input and return a value. The default **fweight** is the arithmetic mean.

*motif_participation* calculates the motif participation for each node (and each link, if **links=True**). All arguments passable to *motif_participation* are the same as those passed to *pymfinder*. If **weighted=True**, the weight of all interactions forming each motif will also be considered (see the description in the article presenting the software). The characterization of the weight of a motif can be changed using the functional argument

**fweight**. Such function **fweight** needs to take a Python list as input and return a value. The default **fweight** is the arithmetic mean.

*motif_roles* calculates the role of each node (and each link, if **links=True**). All arguments passable to *motif_roles* are the same as those passed to *pymfinder*. Only defined for 2≤motifsize≤3 for unipartite networks and 2≤motifsize≤6 for bipartite networks. If **weighted=True**, the weight of all interactions forming each motif will also be considered (see the description in the article presenting the software). The characterization of the weight of a motif can be changed using the functional argument **fweight**. Such function **fweight** needs to take a Python list as input and return a value. The default **fweight** is the arithmetic mean.

*list_motifs* returns all the motif IDs for a given motif size. The only arguments passable to *list_motifs* is the motif size **motifsize**.

*print_motifs* print out all network motifs and motif IDs for any given motif size. The first arguments passed to *list_motifs* is the motif size **motifsize**. Other arguments passable to the function are **motifID** to print a specific motif, **outFile** to save the output in a file, and **links** to print the links forming each motif. We do not recommended using this function for **motifsize**>6.

#### 2.3.3 Output

The object 'results' returned by *pymfinder* is a NetworkStats object containing dictionaries of motifs, nodes, and links. The value for each motif in the .motifs dictionary is an Motif object containing the motif profile for that motif. Similarly, the value for each node or link in the .nodes or .links dictionaries is a NodeLink object containing the motif participation or role of that node or link.

These results can be collected into text-formatted tables and may be seen using:

```
print results
```

or written to a file using:

```
f=open('filename','w')
f.write(str(results))
f.close()
```

If **links=False**, the results include three tables. The first presents the motif profile of the network. Each row gives a motif ID (see list_motifs() and print_motifs() to identify motifs based on ID), the count of that motif in the observed network, the mean and standard deviation of the count of that motif in the randomized networks, and the $Z$-score comparing the real network to the randomized networks. If **randomizations=False**, the random mean and standard deviations will be reported as 0.000 and the $Z$-score will be given as 888888.000.

The second table presents the motif participation for each node in the network. Each row gives a node ID and the number of times the node appears in each motif. Motif ID's are given in the first row.

The third table presents the role for each node in the network. Each row gives a node ID and the number of times the node appears in each position in each motif. Node positions are labeled using the following notation: (motif ID, number of predators/out links, number of prey/in links).

If **links=True**, the results will also contain tables presenting links' motif participation and roles. Links' motif participation follows nodes' motif participation and link roles follow node roles. In both cases, the format of the output table echoes that of the node tables. The only notable difference is in the labeling of link positions. Rather than (motif ID, number of out links, number of in links), link positions are labeled (motif ID, (out links for node 1, in links for node 1), (out links for node 2, in links for node 2)) where nodes 1 and 2 are the two species connected by link (1,2).

# 3 Figures

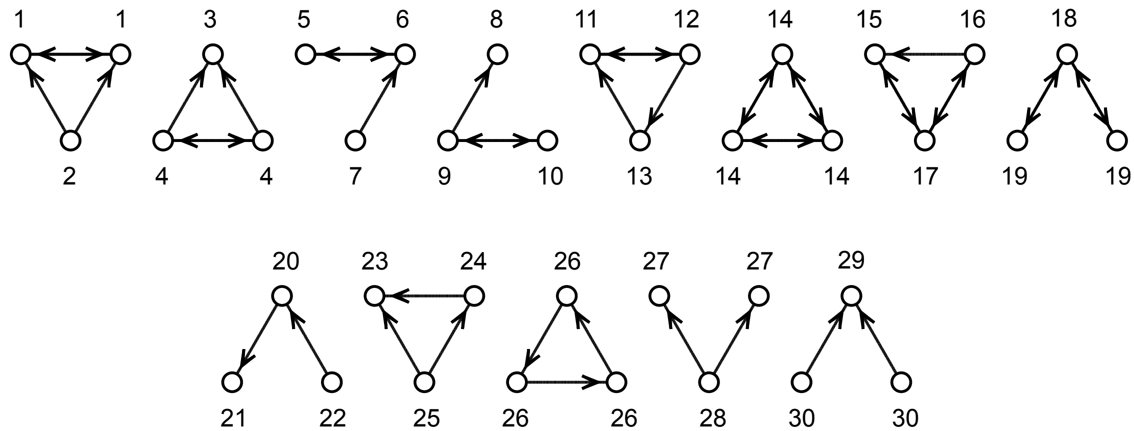


**Figure 1:** The thirteen three-species motifs in unipartite networks. The 30 unique positions are numbered. Note also that some motifs contain three unique positions (e.g., positions 5, 6, and 7) while other motifs contain only one or two unique positions.

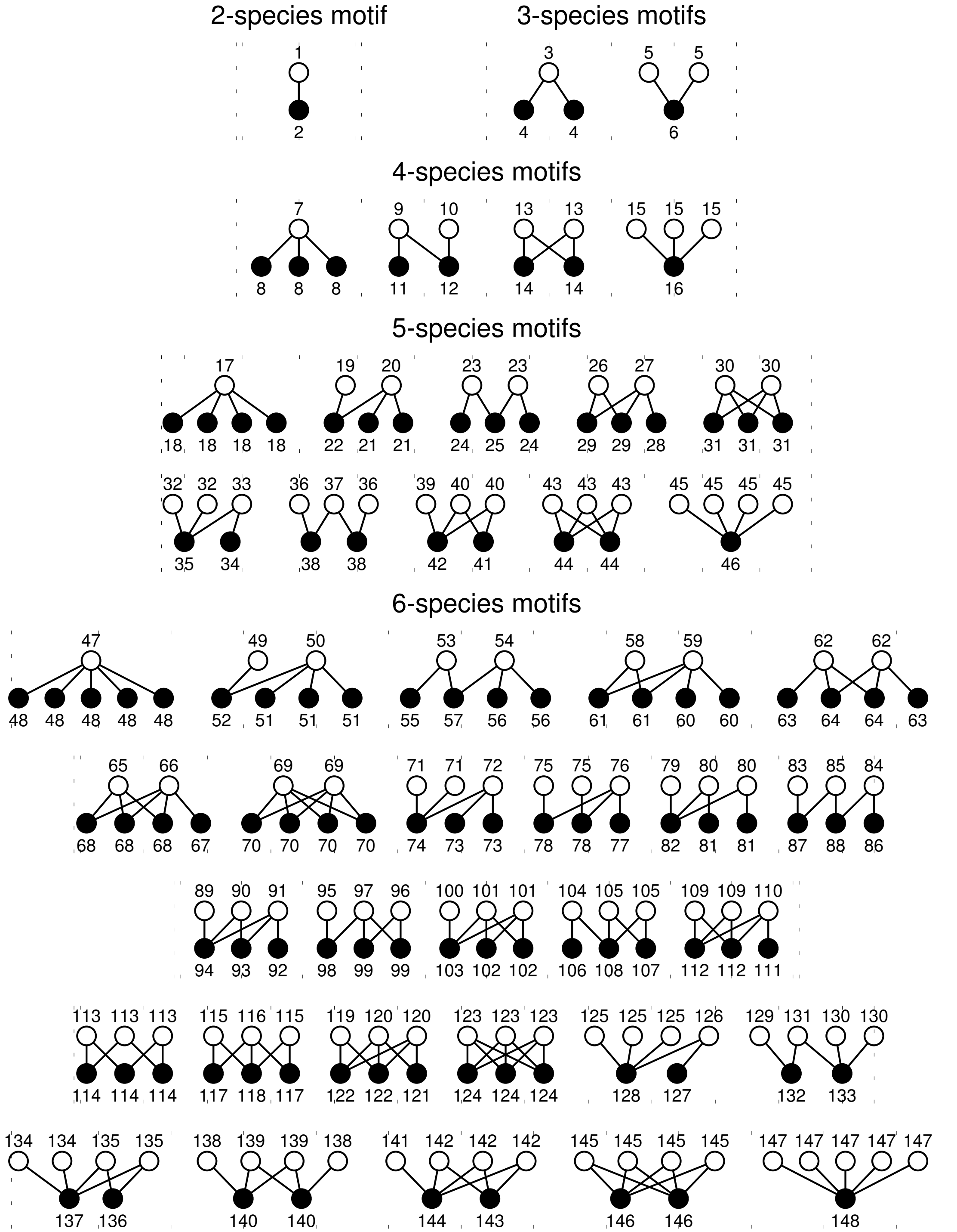


**Figure 2:** The singe two-species motif, two three-species motifs, four four-species motifs, 10 five-species motifs, and 27 six-species motifs that can be found in bipartite networks. Unique positions are numbered. If the identity of particular motifs and/or positions is important, be sure to note the motif number and position information provided in the output file.

# References


[1] Bernat Bramon Mora, Alyssa R Cirtwill, and Daniel B Stouffer. pymfinder: a tool for the motif analysis of binary and quantitative complex networks. *bioRxiv*, 2018. doi: 10.1101/364703. URL https://www.biorxiv.org/content/early/2018/07/07/364703.

[2] Daniel B Stouffer, Juan Camacho, Wenxin Jiang, and Luís A. Nunes Amaral. Evidence for the existence of a robust pattern of prey selection in food webs. *Proceedings of the Royal Society B: Biological Sciences*, 274(1621):1931–1940, August 2007. doi: 10.1098/rspb.2007.0571.