Practical Analysis of Ecological Networks

Timothée Poisot

May 20, 2020

Contents

List of Figures

Chapter 1

Introduction

When we first attempted to synthetize the literature on *how to best analyse ecological networks?* (Delmas et al. 2018), two things became clear. First, an increasing number of fields in ecology were using networks as a formalism to represent and analyse their data. Second, methods were multiplying at an alarming rate, and there was a great degree of methodological confusion about what to use, how to interpret it, and when not to use it. In short, the field of ecological network analysis was long due for a textbook.

1.1 Why networks?

1.2 What this book is about

1.2.1 The structure of ecological networks

1.2.2 Best practices in ecological networks research

1.2.3 Programming

1.3 What this book is not about

1.3.1 The ecology of ecological networks

1.3.2 The foundation of network theory

There are a number of fantastic textbooks on network science, where the mathematical foundations of this field are laid out in a clear and comprehensive way. One of the best ways to share the excitement about why the mathematics of networks theory are worth looking into is to read a short, and very accessible, perspective by Strogatz (2001). Should this whet your appetite, there are multiple resources that are suitable to get a deeper education in graph theory. Two outstanding references are the now classic Newman (2010), and the more recent Barabási (2016). For readers seeking more introductory material, we recommed the short and very accessible volume by Chartrand (1985), as well as West (2001).

But fear not – this book is not about mathematics. The measures of ecological network structure we will discuss have are grounded in mathematics, but the point is *not* to go into much details into it. There will be a *fair* amount of mathematics, probably as much as during an advanced class in statistics (which is to say just enough to fill the average ecologist with apprehension, but not quite with dread), but it will not be the point of the material. This is merely something that comes with the analysis of ecological networks. You will realize through the different chapters that there is only a very small quantity of graph theory we need to know to do good ecology; this will have the highly desirable side effect to keep any mathematical posturing to a minimum.

1.3.3 Programming

1.4 How to read this book

1.4.1 The importance of primary literature

1.4.2 The importance of experimenting with code

This books involves *a lot* of computer code; in fact, it is mostly about writing computer code to analyse ecological networks. Although we rely extensively on a software package (**poisot_ecologicalnetworks.jl_2019**) to do this, there will be many applications where writing our own functions, or writing our own scripts, will be necessary. Copying and pasting this code, and running it exactly as presented here, is a terrible idea. Instead, we encourage readers to *adapt* the code to their own interests, uses, and questions.

How does one adapts code? **tk**

1.4.3 The importance of consistent notation

Throughout this book, we strive to use notation that remains as constant as possible. Networks are represented by capital letters, with the exact letter used representing some additional information about the type of network. Unipartite networks are U, bipartite networks are B, probabilistic networks are P, random networks are R, quantitative networks are G (only because Q is used for modularity already), and a network of an unspecified type is N. When there is a collection of multiple networks, we use the same letter in bold face – for example, we can note that a function generates random networks from a bipartite network using $f(B) = \mathbf{R}$.

To refer to a specific interaction, we will use the notation N_{ij}, which represents an interaction *from* species i *to* species j. In some situations, we will also use $N(i,j)$ to represent the same information. The notation $N_{i\cdot}$ and $N_{\cdot j}$ indicate, respectively, the set of species with which i interacts, and the set of species that interact with j. For example, the number of predators of species 4 in a unipartite food web is $\|U_{\cdot 4}\|$.

Chapter 2

Counting species and interactions

How many things are in a network? The goal of this chapter is to highlight the importance of thinking about the number of interactions in a network, and about the number of interactions established by a species within a network. We will use various examples to introduce the notions of connectance, degree, and linkage density. This chapter will also (mostly!) serve as an introduction to `EcologicalNetworks` and `Mangal`, which will provide valuable information for the other chapters.

2.1 Link-species relationships

We will begin this chapter by looking at the simplest information we can get on a single network: *how many things are there?*. Focusing on the basics is useful for a few reasons. First, it offers a powerful intuition on the structure of ecological networks, which is going to underpin most of the topics of this course. Second, this allows exploring some core concepts about how ecological networks are represented, and how we can look at their components. Finally, this provides an introduction to the manipulation of network data through programming. In particular, this section will serve as a short introduction to the `Mangal` package to download ecological networks.

We will load the `Mangal` package, to get access to a variety of functions allowing us to get network data from the `mangal.io` (`http://mangal.io`) website. Note that we will not load it with `using`, but with `import` - this means that we will need to preface the name of the functions by `Mangal`, to avoid having a lot of functions with non-specific names in our namespace:

```
import Mangal
```

All of these packages have documentations, and it is always a good idea to see the different options available for the various functions. One way to get a sense of the functions included in a package, or to get help on these functions, it to type `?[PackageName].` in the Julia REPL, and to press Tab to see the possible functions. This is also a great way to discover functionalities in packages.

2.1.1 Overview of the data querying

And with this, we are all set to get our first network from the `mangal.io` database. To illustrate the basic notions, we will work on a pollination network, which we access by its unique identifier (`935`). We will see in later sections how we can get more control over what networks we get.

```
first_network = Mangal.network(935)
print(first_network.description)
```

```
Pollination networks of the Azorean Flores island
```

This object is of the type Mangal.MangalNetwork, and has information about a network. To see what information is available in any type, we can call `fieldnames` on the type. In this case, this would return

```
fieldnames(typeof(first_network))
```

```
(:id, :public, :name, :date, :position, :created, :updated, :user, :descrip
tion, :complete, :dataset)
```

Most of these information are metadata, *e.g.* the date, spatial position, date of upload, and a reference. What we do *not* have is a list of the interactions. Luckily, we do not yet need a list of interactions, since we are only concerned with counting them (in due time, we will get the entire information on interactions). The `Mangal` package has a method called `count`, which will return the number of objects matching a series of queries. For example, we can get the number of nodes (of type `MangalNode`) in our network with:

```
count(Mangal.MangalNode, first_network)
```

```
22
```

We can do the same with the number of interactions:

```
count(Mangal.MangalInteraction, first_network)
```

```
30
```

This method is preferable to more complex ones requiring to download all interactions, because it is a lot faster. As we will see in the following section, the representation of network data in `mangal.io` is very rich, and if all we care about is counting things, then we do not need the overhead associated with retrieving all the data.

Equipped with the ability to count objects, we can measure the number of species and interactions in any network. We will now generalize this process a little bit, and explore the relationship that exists between the number of species and the number of links in a collection of networks.

2.1.2 Getting the data for multiple networks

In `mangal.io`, networks are organized in datasets - we will work on a classical food web analysis dataset, representing interactions in small lakes in the Adirondack:

```
small_lakes_dataset = first(Mangal.datasets("q" => "Adirondack"))
```

```
Mangal.MangalDataset(15, true, "havens_1992", 2019-02-23T01:47:06, 2019-02-
23T01:47:06, Mangal.MangalReference(15, 1992, "10.1126/science.257.5073.110
7", missing, missing, "@article{Havens_1992, doi = {10.1126/science.257.507
3.1107}, url = {https://doi.org/10.1126%2Fscience.257.5073.1107}, year = 19
92, month = {aug}, publisher = {American Association for the Advancement of
 Science ({AAAS})}, volume = {257}, number = {5073}, pages = {1107--1109},
author = {K. Havens}, title = {Scale and Structure in Natural Food Webs}, j
ournal = {Science}}", "https://doi.org/10.1126%2Fscience.257.5073.1107", "U
RL of the attached data"), 3, "Pelagic communities of small lakes and ponds
 of the Adirondack")
```

To get the networks associated to this dataset, we can use the `networks` function, which will give us an array of networks objects. But first, we may want to know how many networks we will get this way:

```
count(Mangal.MangalNetwork, small_lakes_dataset)
```

```
50
```

The following line will get all of these networks for us:

```
small_lakes_networks = Mangal.networks(small_lakes_dataset)
println("small_lakes_networks contains $(length(small_lakes_networks)) networks")
```

```
small_lakes_networks contains 50 networks
```

We will therefore be able to use our collection of 50 networks to see how the number of species and links are related.

2.1.3 Counting species and interactions

From our collection of networks (`small_lakes_networks`), we want to extract the number of nodes, and the number of interactions. There are a few ways to do this, as is always the case with programming tasks. In this case, we will summarize every network by a `Tuple` which will contain `(S, L)`, the number of species and links. A `Tuple` is not modifiable, so this is a safe way to collect data. In addition, tuples of coordinates are usable directly in plots, and this will simplify the syntax immensely.

The first way to achieve this is to use a ”list comprehension”:

```
LS_1 = [
    (
        count(Mangal.MangalNode, network),
        count(Mangal.MangalInteraction, network)
    ) for network in small_lakes_networks
]
first(LS_1)
```

```
(26, 60)
```

This is a little bit unwieldy, as this is a long line, and requires to create a variable (`network`) only to use inside the loop. Instead, we might want to use the more readable expression below:

```
S = count.(Mangal.MangalNode, small_lakes_networks);
L = count.(Mangal.MangalInteraction, small_lakes_networks);
LS = collect(zip(S,L));
```

This alternative version is easier to parse, and throughout the book we will try to favor readability over ”clever” code. The `LS` variable contains tuples that look like:

```
first(LS)
```

```
(26, 60)
```

The first network in our dataset has 26 species, and 60 interactions.

TK PLOT

2.1.4 Theories for link-species scaling

2.2 Degree and degree distribution

In the previous section, we focused on the relationship between the number of species and the number of interactions *across* networks. It is also very informative to look at the distribution of interactions *across* species *within* networks

- we call the number of interactions that a species has its ”degree”, and the degree distribution is crucial information about a network.

To look at the degree and degree distribution, we will need a few packages, which you have installed during the introduction. Specifically, we will want to load functions to analyse networks themselves, so we will use `EcologicalNetworks`.

```julia
using EcologicalNetworks
```

2.2.1 Getting the network data

And with this, we are all set to get our first network (we also briefly encountered it during the previous section):

```julia
first_network = Mangal.network(935)
print(first_network.description)
```

```
Pollination networks of the Azorean Flores island
```

As opposed to the previous section, we now want to get a list of interactions contained in this network:

```julia
first_network_interactions = Mangal.interactions(first_network);
print("There are $(length(first_network_interactions)) interactions in this network")
```

```
There are 30 interactions in this network
```

We can have a deeper look at the first interaction:

```julia
first_interaction = first(first_network_interactions)
```

```
Mangal.MangalInteraction(55518, Mangal.MangalNetwork(935, true, "olesen_al_
2002_20000701_935", 2000-07-01T00:00:00, GeoInterface.Point([-31.193594, 39
.444798]), 2019-02-25T16:29:16, 2019-02-25T16:29:16, 2, "Pollination networ
ks of the Azorean Flores island", false, Mangal.MangalDataset(62, true, "ol
esen_al_2002", 2019-02-25T16:27:42, 2019-02-25T16:27:42, Mangal.MangalRefer
ence(61, 2002, "10.1046/j.1472-4642.2002.00148.x", missing, missing, "@arti
cle{Olesen_2002, doi = {10.1046/j.1472-4642.2002.00148.x}, url = {https://d
oi.org/10.1046%2Fj.1472-4642.2002.00148.x}, year = 2002, month = {may}, pub
lisher = {Wiley-Blackwell}, volume = {8}, number = {3}, pages = {181--192},
 author = {Jens M. Olesen and Louise I. Eskildsen and Shadila Venkatasamy},
 title = {Invasion of pollination networks on oceanic islands: importance o
f invader complexes and endemic super generalists}, journal = {Diversity an
d Distributions}}", "https://onlinelibrary.wiley.com/doi/abs/10.1046/j.1472
-4642.2002.00148.x", "https://www.nceas.ucsb.edu/interactionweb/html/olesen
_et_al_2002.html"), 2, "Pollination networks for two oceanic islands, the A
zorean Flores and the Mauritian Ile aux Aigrettes")), Mangal.MangalNode(668
4, "Halictus sp.", 2019-02-25T16:29:23, 2019-02-25T16:29:23, Mangal.MangalR
eferenceTaxon(2617, "Halictus", 2225, 154356, 88465, 2742846, missing, 2019
-02-22T00:24:21, 2019-06-14T15:22:22)), Mangal.MangalNode(6674, "Azorina vi
dalii", 2019-02-25T16:29:22, 2019-02-25T16:29:22, Mangal.MangalReferenceTax
on(2607, "Azorina vidalii", 420018, missing, 239395, 5135848, 3166266, 2019
-02-22T00:24:20, 2019-06-14T15:22:21)), 2000-07-01T00:00:00, missing, true,
 :mutualism, "field observations", 98, 2019-02-25T16:29:39, 2019-02-25T16:2
9:39, Mangal.MangalAttribute(11, "number of visit by a pollinator", "Number
 of individual of a species observed/caught on a flower", "number of indivi
dual"))
```

This has *a lot* of information! We can focus on the most important properties: **TK table**

2.2.2 Creating the network object

There are a few ways to collect these interactions into a network, but the simplest by far is to transform (`convert`) our `MangalNetwork` object into a `UnipartiteQuantitativeNet` (which is used by `EcologicalNetworks`) - this is done as follows:

```
N = convert(UnipartiteQuantitativeNetwork, first_network)
```

```
22×22 unipartite quantitative ecological network (Float64, Mangal.MangalNod
e) (L: 30)
```

At this point, you are hopefuly familiar with what ”network” means, but may have some hesitations about the meaning of ”unipartite” and ”quantitative”. **TK**

As a sidenote, we will often adhere to the following convention when naming networks. `N` is a unipartite network, `B` is a bipartite network, and `R` is a random network **TK link to notation**. This is a useful way to get a sense of what we can expect the variable to contain.

Speaking of which, we know that this network contains information on pollination, and pollination networks are bipartite, because they have two groups of organisms (pollinators and polinized), which we assume do not establish interactions within the group. We can therefore transform (`convert`) our network into its bipartite representation:

```
B = convert(BipartiteQuantitativeNetwork, N)
```

```
12×10 bipartite quantitative ecological network (Float64, Mangal.MangalNode
) (L: 30)
```

If you are not sure that a network can be represented in a bipartite way, you can always use the check for bipartiteness, which will return `true` if the network can be converted.

```
EcologicalNetworks.check_bipartiteness(N)
```

```
true
```

Not to worry, the `convert` function will fail with an explanation if the network cannot be transformed into a bipartite one.

2.2.3 Visualizing the network

TK plots package

To get a sense of what the network ”looks like”, we may decide to plot it. In practice we will often refrain from plotting networks. Plotting networks is very rarely useful. Depending on the layout, it is easy to mis-communicate about the structure of the network. Past a certain number of species and interactions, the plots look like hairballs. Most importantly, we have a wide variety of quantitative measures that we can apply, and they will teach us a lot more about the network structure than any plot ever would. Plotting is marginally useful during explorations, and to illustrate certain concepts, but should largely be avoided for research purposes. The few exceptions to this rule will be during the chapters on network comparison, network motifs, and network modularity.

2.2.4 Counting species and interactions

```
richness(B)
```

```
22
```

```
richness(B; dims=1)
```

```
12
```

```
richness(B; dims=2)
```

```
10
```

```
links(B)
```

```
30
```

```
sum(B)
```

```
1139.0
```

2.2.5 Getting the degree of species

```
degree(B)
```

```
Dict{Mangal.MangalNode,Int64} with 22 entries:
  MangalNode(6678, "Daucus carota", 2019-02-25T16:29:23, 2019-02-25T16:29:…
 => 3
  MangalNode(6693, "Eristalix tenax", 2019-02-25T16:29:23, 2019-02-25T16:2…
 => 1
  MangalNode(6675, "Crithmum maritimum", 2019-02-25T16:29:22, 2019-02-25T1…
 => 4
  MangalNode(6684, "Halictus sp.", 2019-02-25T16:29:23, 2019-02-25T16:29:2…
 => 6
  MangalNode(6681, "Lotus corniculatus", 2019-02-25T16:29:23, 2019-02-25T1…
 => 3
  MangalNode(6683, "Reseda luteola", 2019-02-25T16:29:23, 2019-02-25T16:29…
 => 2
  MangalNode(6695, "Anothomyia pluvialis", 2019-02-25T16:29:23, 2019-02-25…
 => 1
  MangalNode(6692, "Lucilia sericata", 2019-02-25T16:29:23, 2019-02-25T16:…
 => 2
  MangalNode(6687, "Colias crocea", 2019-02-25T16:29:23, 2019-02-25T16:29:…
 => 3
  MangalNode(6674, "Azorina vidalii", 2019-02-25T16:29:22, 2019-02-25T16:2…
 => 8
  MangalNode(6682, "Freesia refracta", 2019-02-25T16:29:23, 2019-02-25T16:…
 => 2
  MangalNode(6691, "Calliphora vemitoria", 2019-02-25T16:29:23, 2019-02-25…
 => 1
  MangalNode(6686, "Bombus ruderatus", 2019-02-25T16:29:23, 2019-02-25T16:…
 => 3
  MangalNode(6685, "Sepsis thoracica", 2019-02-25T16:29:23, 2019-02-25T16:…
 => 4
  MangalNode(6688, "Musca domestica", 2019-02-25T16:29:23, 2019-02-25T16:2…
 => 4
  MangalNode(6677, "Beta vulgaris", 2019-02-25T16:29:23, 2019-02-25T16:29:…
 => 1
  MangalNode(6679, "Silene vulgaris", 2019-02-25T16:29:23, 2019-02-25T16:2…
 => 2
  MangalNode(6690, "Lasius niger", 2019-02-25T16:29:23, 2019-02-25T16:29:2…
 => 1
  MangalNode(6676, "Solidago sempervivens", 2019-02-25T16:29:22, 2019-02-2…
 => 2

 =>
```

```
degree(B; dims=1)
```

```
Dict{Mangal.MangalNode,Int64} with 12 entries:
  MangalNode(6693, "Eristalix tenax", 2019-02-25T16:29:23, 2019-02-25T16:2…
 => 1
  MangalNode(6684, "Halictus sp.", 2019-02-25T16:29:23, 2019-02-25T16:29:2…
 => 6
  MangalNode(6695, "Anothomyia pluvialis", 2019-02-25T16:29:23, 2019-02-25…
 => 1
  MangalNode(6692, "Lucilia sericata", 2019-02-25T16:29:23, 2019-02-25T16:…
 => 2
  MangalNode(6687, "Colias crocea", 2019-02-25T16:29:23, 2019-02-25T16:29:…
 => 3
  MangalNode(6691, "Calliphora vemitoria", 2019-02-25T16:29:23, 2019-02-25…
 => 1
  MangalNode(6686, "Bombus ruderatus", 2019-02-25T16:29:23, 2019-02-25T16:…
 => 3
  MangalNode(6685, "Sepsis thoracica", 2019-02-25T16:29:23, 2019-02-25T16:…
 => 4
  MangalNode(6688, "Musca domestica", 2019-02-25T16:29:23, 2019-02-25T16:2…
 => 4
  MangalNode(6690, "Lasius niger", 2019-02-25T16:29:23, 2019-02-25T16:29:2…
 => 1
  MangalNode(6689, "Apis mellifera", 2019-02-25T16:29:23, 2019-02-25T16:29…
 => 3
  MangalNode(6694, "Agrotis ipsilon", 2019-02-25T16:29:23, 2019-02-25T16:2…
 => 1
```

```
degree(B; dims=2)
```

```
Dict{Mangal.MangalNode,Int64} with 10 entries:
  MangalNode(6680, "Chamomilla suaveolens", 2019-02-25T16:29:23, 2019-02-2…
 => 3
  MangalNode(6676, "Solidago sempervivens", 2019-02-25T16:29:22, 2019-02-2…
 => 2
  MangalNode(6675, "Crithmum maritimum", 2019-02-25T16:29:22, 2019-02-25T1…
 => 4
  MangalNode(6674, "Azorina vidalii", 2019-02-25T16:29:22, 2019-02-25T16:2…
 => 8
  MangalNode(6681, "Lotus corniculatus", 2019-02-25T16:29:23, 2019-02-25T1…
 => 3
  MangalNode(6677, "Beta vulgaris", 2019-02-25T16:29:23, 2019-02-25T16:29:…
 => 1
  MangalNode(6682, "Freesia refracta", 2019-02-25T16:29:23, 2019-02-25T16:…
 => 2
  MangalNode(6683, "Reseda luteola", 2019-02-25T16:29:23, 2019-02-25T16:29…
=> 2
 MangalNode(6679, "Silene vulgaris", 2019-02-25T16:29:23, 2019-02-25T16:2…
=> 2
 MangalNode(6678, "Daucus carota", 2019-02-25T16:29:23, 2019-02-25T16:29:…
=> 3
```

2.2.6 Degree distribution

Chapter 3

Comparing ecological networks

Although this might seem like a more advanced topic, comparing network is relatively easy in that it mostly involves enumerating elements – how many species, and how many interactions, are either unique to one, or shared by two networks? As such, a good understanding of the core concepts in **??** will be useful for this chapter.

To develop an intuition of why ecological networks differ, it is useful to think of them as sets. Not only does it allows defining a number of mathematical operations that will greatly simplify the notation, it also matches directly the formalism developped by Koleff, Gaston, and Lennon (2003) for the dissimilarity of presence-absence data, which we will follow throughout this chapter.

3.1 Pairwise network comparison

3.2 The metaweb

In the previous section, we have focused on the fact that different networks can share, or not, some of their species, and some of their interactions. This provided a powerful intuition to think about network dissimilarity in terms of α and β diversity - but what about γ? In this section, we will think about the ”metaweb”, defined by **Dunn06** as the aggregation of all regional interactions in a series of related networks. In other words, we can define a metaweb as the recursive union of a collection of networks.

3.2.1 Metaweb properties

3.2.2 Network distance to the metaweb

3.3 Multi-site network comparison

Chapter 4

Motifs in ecological networks

Chapter 5

Nestedness and interaction overlap

Chapter 6

The deeply flawed art of null hypothesis significance testing

```
using EcologicalNetworks
import Mangal
using Statistics
```

Ecologists, like sadly a great majority of scientists, decide on the worthiness of an observation by ignoring their best judgement, decades of training in biology, and expertise; instead, we rely on whether some arbitrary value (the p-value) is lower than some even more arbitrary threshold (0.05). It all seems very *ad hoc* (it is). And yet, a surprising amount of literature on ecological networks attempts to decide whether some observed value of a network measure is "significant", and so it is with great reluctance that we will dedicate a chapter to this practice.

6.1 Fundamentals of network NHST

Null Hypothesis Significance Testing (NHST) for ecological networks strives to answer the following question: "if I observed the value f_0 for the measure f on a network $\mathbf{N}$, is it in the range of values expected by chance?". Most often, we want to refine this question by asking if the value is larger or smaller than we expect, and this is why this process usually involves a one-tailed one-sample t-test.

6.1.1 Generating a sample

Given a function f and a network $\mathbf{N}$, NHST requires to measure the distribution of possible values of $f(\mathbf{N})$. For most measures (all of them, in fact), there is no direct expression of this distribution, and so we need to rely on an imperfect proxy: generating random networks. The exact models under which these models are generated is explained in section **TK**.

6.1.2 From the sample to the distribution

6.1.3 Estimating significance

6.2 Overview of the null models

6.2.1 Probabilistic models

6.2.2 Generalization of the probabilistic models

$$P_{ij} = \alpha_1 N_{ij} + \alpha_2 \frac{N_{\cdot j}}{S_1} + \alpha_3 \frac{N_{i\cdot}}{S_2} + \alpha_4 \frac{L}{S_1 S_2} \tag{6.1}$$

$$P = lf(N, \alpha = (0,0,0,1)) \tag{6.2}$$

6.2.3 Permutational models

6.3 Application – nestedness of oil-flowers pollination networks

In this section, we will use the NHST process outline in figure **??** to determine whether bipartite networks are *more* nested than expected by chance, under two assumptions as to what drives the structure of the model
6.2

6.3.1 Preparing the data and measuring nestedness

We will first work on a dataset of pollination interactions between insects and oil-flowers, collected by Bezerra, Machado, and Mello (2009).

```
oilflower_network = Mangal.network(929)
N = convert(UnipartiteNetwork, oilflower_network)
B = convert(BipartiteNetwork, N)
```

```
13×13 bipartite  ecological network (Bool, Mangal.MangalNode) (L: 71)
```

We will use NODF (**tk**) to measure the nestedness of this network. This is an important step, as it will provide our f_0, the reference value, measured on the empirical network:

```
f0 = nodf(B)
```

```
0.849310643060643
```

Recall that NODF takes values between 0 and 1, where 1 indicates perfect nestedness – this network has therefore a rather high nestedness, and it makes sense to ask whether this value is larger than expected by chance. Our definition of *chance* here will be represented by two null models: either the nestedness is larger than expected given the value of connectance, or the nestedness is larger than expected given the degree distribution of the network.

6.3.2 Generating the null sample

```
P = null2(B)
R = rand(P, 5000)
```

TK

```
f = nodf.(R)
```

We can start looking at the average value of the nestedness of the random networks:

```
mean(f)
```

```
NaN
```

Well, that didn't work.

6.3.3 Fixing the null sample

This is probably because the NODF measure is failing on some networks, and returning a value of `NaN`. Let's see which networks are to blame:

```
X = R[findall(isnan.(f))]
```

We can look for all sorts of reasons to understand why the NODF measure is failing. The answer is as follows: as you will recall from section **??**, NODF involves dividing the number of overlapping interactions by the XXX number of interactions. So if one species has no interactions, we would end up dividing by 0, which would result in `NaN`. How could a species have no interaction? The null model we use is in fact a probabilistic network, and we know from **probanet** that there is a chance (which can be very small), that a species will end up having no interactions if we perform independent random draws.

This provides an interesting hypothesis: that the networks for which we do not have a NODF score have ”degenerate” matrices, after the **twosides** nomenclature.

```
all(isdegenerate.(X))
```

```
true
```

This can be solved in two ways. We can either decide to remove these networks, or we can decide to remove the species that are not connected. In section **??**, we will discuss the consequences of this choice, but for now, we will remove thes networks:

```
filter!(!isdegenerate, R)
```

We can now update the distribution of values:

```
f = nodf.(R)
```

Visualized in 6.1

6.3.4 Measuring significance

```
sum(f .>= f0)/length(f)
```

```
0.0
```

6.4 Application – looking for generalities

6.5 Overview of the issues

6.5.1 Degenerate matrices

6.5.2 Strongly constrained matrices

6.5.3 Representativity of the sample

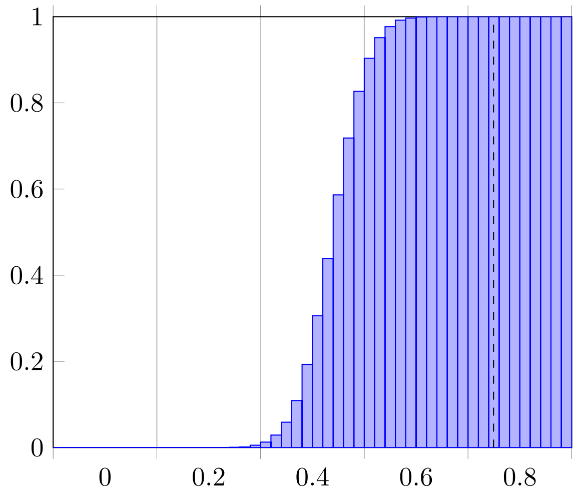

Figure 6.1: Cumulative distribution of the values of nestedness on the simulated networks based on TK. The value f_0 of the original network is marked as a vertical bar.

Bibliography

Barabási, Albert-Lászĺo (2016). *Network Science*. Cambridge University Press.

Bezerra, Elisângela L. S., Isabel C. Machado, and Marco A. R. Mello (2009). "Pollination Networks of Oil-Flowers: A Tiny World within the Smallest of All Worlds". In: *Journal of Animal Ecology* 78.5, pp. 1096–1101. ISSN: 1365-2656. DOI: 10.1111/j.1365-2656.2009.01567.x.

Chartrand, Gary (1985). *Introductory Graph Theory*. Unabridged and corr. New York: Dover. 294 pp. ISBN: 978-0-486-24775-5.

Delmas, Eva et al. (June 20, 2018). "Analysing Ecological Networks of Species Interactions". In: *Biological Reviews*, p. 112540. ISSN: 14647931. DOI: 10.1111/brv.12433.

Koleff, Patricia, Kevin J. Gaston, and Jack J. Lennon (May 1, 2003). "Measuring Beta Diversity for Presence–Absence Data". In: *Journal of Animal Ecology* 72.3, pp. 367–382. ISSN: 1365-2656. DOI: 10.1046/j.1365-2656.2003.00710.x.

Newman, Mark E. J. (2010). *Networks. An Introduction*. New York, NY: Oxford University Press. URL: http://dl.acm.org/citation.cfm?id=1809753.

Strogatz, Steven H. (Mar. 8, 2001). "Exploring Complex Networks". In: *Nature* 410.6825, pp. 268–276. ISSN: 0028-0836. DOI: 10.1038/35065725.

West, Douglas Brent (2001). *Introduction to Graph Theory*. 2nd ed. Upper Saddle River, N.J: Prentice Hall. 588 pp. ISBN: 978-0-13-014400-3.