PREDICTION OF ECOLOGICAL FUNCTION IN THE MICROBIOME USING MACHINE LEARNING ON THE GRAPH SPECTRA OF COEVOLVING SUBNETWORKS

Russell Y. Neches
Matthew D. McGee
Peter C. Wainwright
Jonathan A. Eisen

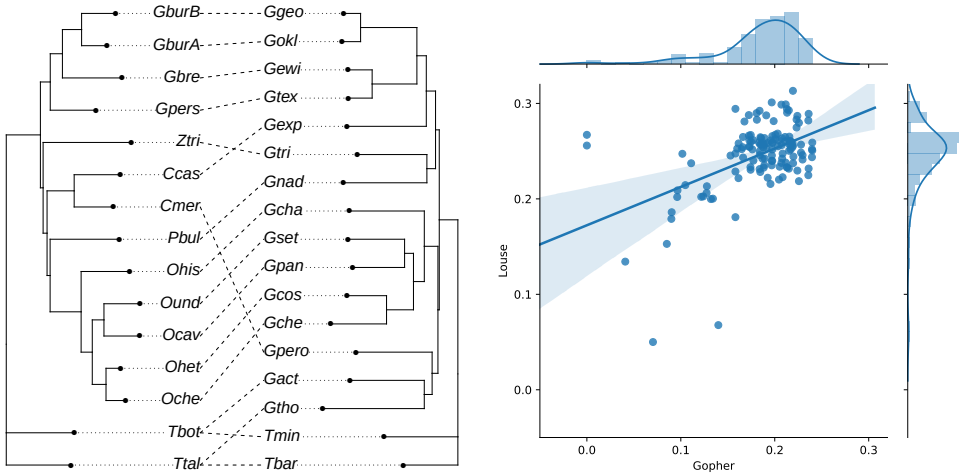
March 21, 2018

CONTENTS

1 INTRODUCTION

Biology has a major problem with charisma. Humans relate to some organisms more easily than others. Fuzzy creatures get more attention than scaly or slimy ones. Creatures with faces get more attention than creatures without. Animals get more attention than plans. By abundance, diversity, biomass, age or metabolic wattage, eukaryotes make up a tiny fraction of life on Earth, but they occupy almost all of our attention. The same phenomenon happens in ecology.

Ecology is the study of how organisms interact with one another and with their environment. Some relationships are more charismatic than others, and those relationships dominate our attention. There are trophic strategies in myriad diversity, but predator-prey interactions make the best television. Parasites give us the creeps



(a) Cophylogeny of gophers and lice

(b) Paired patristic distances

Figure 1: The relationship between pocket gophers and their chewing lice parasites has served as a benchmark case in the literature on coevolution since its appearance in Hafner *et al.* [1]. However, despite the strong case for coevolution from multiple lines of evidence, the agreement between the two trees (as measured by the correlation of pairwise patristic distances through the two trees [2]) is modest, with a Pierson's r of 0.49. If one were to exclude the relationships between the outgroups as outliers, the correlation would collapse. Without other forms of evidence, the detection of such relationships is challenging.

as much as they fascinate us. Mutualisms speak powerfully to aspirations and anxieties regarding our own societies. Disease has shaped and reshaped nations, and drives a large part of the moral imperative behind biological research. Nevertheless, the great majority of ecological relationships are none of the above, or cannot neatly fit into any one category.

Charisma is one of many heuristics that help us identify things that are likely to be relevant to our own experience. Heuristics are useful, but usefulness should not be mistaken for accuracy. The fact that charisma cannot be separated from the observer means that it generalizes poorly. The fact that charisma is a heuristic measure of importance means that it can still be wrong more often than right. It is a form of bias, and distorts the model of reality we use to understand how things work.

Biologists address charisma bias by applying other metrics for importance, and often look to ecology for parameters to include in these metrics. Importance is contextual, after all, and ecology is the study of the relationships that comprise the context in which organisms live. How, then, can ecology correct for its own charisma bias? Relationships can be categorized by their effects or their dynamics, but unless they exhibit a charismatic property – often a symmetry in structure, a simplicity of concept or an analogy to human experience – they can defy categorization or escape notice altogether. The symmetry of the Red Queen's Race, the straightforwardness of a predator's relationship with its prey, or the (projected) virtue of cooperation are conspicuous because they are unusual. Most relationships are too complicated or ambiguous to clearly exhibit them. They are not charismatic.

This is especially evident in microbial interactions, where high-throughput sequencing has made it possible to generate spectacular quantities of complex, nuanced and mostly inconclusive data. This is frustrating, but the scale of the problem represents a unique opportunity to address charisma bias in ecology.

Correcting a bias always begins with the same task : find a way to collect data in a way this isn't subject to the bias. Microbiome surveys are vulnerable to a variety of technical biases (there is *always* another bias), but sequencing machines are not

impressed by charisma. Based on this data, it is possible to construct and test new metrics of importance, and to see which relationships merit further attention.

Unfortunately, we do not have very many theoretical tools that address the question of how to categorize ecological relationship data gathered using an non-targeted, unsupervised process.

Fortunately, ecology is not the only field that cares about the problem of detecting and assigning categories to complex relationships from non-targeted data. Vast resources have been devoted to this problem when it appears in the form of social networks of human beings. Technology companies want to know how to detect and identify categories within social networks so that they can sell more advertisements, and pursuit of this capability steers R&D budgets in the billions of dollars. If meaningful representations of microbiome data can be constructed in the same mathematical framework, we can divert some of those resources into our own projects. We can parasitize the tech companies.

A successful parasite must have a strategy for invading its host. If we want to abscond with their shiny computational tools, we need a way to inject our data into them. In the next post, I will lay out my strategy for injecting microbiome data into off-the-shelf machine learning frameworks, and in the subsequent posts, I will try to give you a taste of why this is awesome.

But, before we get into that, it needs to be said that there are downsides to these tools. Particularly, the intermediate steps in machine learning can be difficult to interpret intuitively, and machine learning stacks tend to both reflect and obfuscate the biases of their designers. Nevertheless, if the goal is to address charisma bias, placing *all* of the available data into the same context is a good start. Machine learning does not eliminate bias; it formalizes it. This can cut two ways. If used uncritically, machine learning can atomize and dissolve the designer's bias into the mathematics in ways that can be very harmful. Or, it can provide a framework within which one can select, quantify, explore and hopefully understand one's own biases.

Always compute responsibly.

2 REPRESENTING A CO-PHYLOGENY AS A GRAPH

Microbiome data yields two types of information. The sequences themselves shed light on relationships among organisms in through time, and the distributions and frequencies of those sequences on relationships among organisms across space. Evolutionary relationships tells us about what happened over deep time, and ecological relationships tell us about what was happening at the moment the samples were collected. There are many ways structure and interpret this information, and each carries an implicit opinion about what properties of these relationships are important.

One way to structure this information to build phylogenetic trees for each group of organisms involved in the interaction, and note which organisms at the tips of these trees were observed to interact with which other organisms. This model is called a co-phylogeny, and pops up in the literature whenever the evolution of two things is somehow linked together.[1] For example, phylogenomics addresses the implications of the fact that genes within the same genome have unique, individual evolutionary histories. Trees representing the evolution of homologous genes can be linked by the genomes in which the homologs co-occur, or they can be linked to trees representing the evolution of the species. Alternatively, the two trees can rep-

1 It is important to note that a co-phylogeny contains an implicit opinion that relationships that are "important" now were "important" in the past, and as such, may have influenced the evolution of the organisms in involved. Exactly what is meant by "important" depends on what the trees represent and on the calculations are performed on the co-phylogeny, but it should be acknowledged that simply arranging the data this way is not a neutral choice. There are other valid ways to think about this information.

resent two very different groups of species that interact ecologically, such a group of host organisms and their parasites.

Co-phylogenies exist on a continuum of intimacy. At one extreme, different regions of a single gene could evolve under somewhat different selective pressures. For example, the evolution of an catalytic domain and a signaling domain of a protein could be examined by building trees for the two domains and linking them through each allele. At the other extreme, the evolution of two groups of species that do not interact may fall under a common influence. For example, penguins and flowering plants probably did not strongly influence each other's evolution, but parallels in their co-phylogeny indicate that the evolution of both was influenced by the breakup the Gondwanan super-continent.

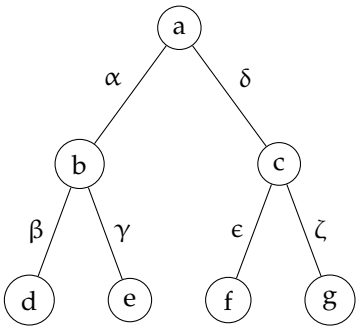
- domains within the same gene (domain-tree:domain-tree)
- domains of a protein verses the whole protein
- alleles of different genes (phylogenomics)
- alleles of a gene verses the species (speciation)
- nuclear verses mitochondrial or chloroplast genes (endosymbiosis)
- host verses parasite (antagonistic coevolution)
- host verses symbiont (mutualistic coevolution)
- interacting species (comensal coevolution)
- co-distributed species (biogeography, parallel evolution)
- propinquity (association without coevolution)

Co-phylogenies built from microbiome data represent a mixture of the last six categories. This is a substantial theoretical challenge. Most models for coevolution are designed to explore carefully framed interactions. Just as human beings in large social networks are often only vaugely aware of the emergent cliques to which they belong, microbiome data does not come with neatly curated host-microbe interactions. The salient features embedded in the graph must somehow be described, detected and extracted.

So, practically speaking, how does one go about building this representation? The first step is to build an adjacency matrix. The adjacency matrix of a graph of n nodes is an $n \times n$ matrix. Connections between nodes ("edges," in graph jargon) are represented as 1's in the row and column for the two nodes. Because there is a row and a column for each node, there are two ways places to mark a connection. If you imagine connections as going *from* the column node *to* the row node, you can have connections with a direction. Directed graphs are useful when the nodes represent things that can be connected asymmetrically. Social networks are like this; Alice might say that she is friends with Bob, but Bob might not reciprocate because he forgot his password (we all know that Alice and Bob are friends in real life).

Our nodes represent leafs and junctions of phylogenetic trees, and our edges represent either phylogenetic distance between tree junctions or ecological connections between leafs. It's not obvious what "direction" would mean for that kind of object, and so we're not going to use a directed graph. Undirected graphs can be represented either by a symmetric adjacency matrix ($A_{i,j} \equiv A_{j,i}$), or by a triangular half-matrix. I hate looking at plots of triangular matrixes, so I'm always going to draw them with top and bottom triangles. This is redundant information that we might want to discard later for faster performance, but for now we'll use it to make the plots easier to understand.

Adjacency matrixes can have connections from a node back to itself. These go along the diagonal ($A_{i,i}$). In a directed graph, it doesn't matter which direction



$$A = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \end{array} \begin{array}{c} \begin{array}{ccccccc} a & b & c & d & e & f & g \end{array} \\ \begin{pmatrix} 0 & \alpha & \delta & 0 & 0 & 0 & 0 \\ \alpha & 0 & 0 & \beta & \gamma & 0 & 0 \\ \delta & 0 & 0 & 0 & 0 & \epsilon & \zeta \\ 0 & \beta & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \epsilon & 0 & 0 & 0 & 0 \\ 0 & 0 & \zeta & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Figure 2: Construction of the graph adjacency matrix for a phylogenetic tree.

a self-connected edge is oriented, so it makes sense that they don't have a doppelganger in the other triangle of the adjacency matrix. Or, they are their own doppelganger, if you prefer to think of it that way. For our purposes, self-connected nodes don't make any sense. We will disallow them by requiring that $A_{i,i} \equiv 0$.

So far, we've only included information about *whether or not* two nodes are connected, and so edges are represented as either 0 or 1. Instead of representing a social network with a binary value (Alice is friends with Bob, but not with Mallory), we might represent it with a continuous quantity (Alice has sent Bob 14.3 gigabytes of cat videos). When we put those quantities into the adjacency matrix, the graph jargon for this is a *weighted* adjacency matrix. In Figure 5, you can see how this works for a small tree.

We have two types of information we want to include in our graph; phylogenetic branch length and ecological associations. Figure 5 only shows branch lengths for a single tree. How do we get all of that into an adjacency matrix? Hold onto your hats, folks, because this is the one truly original idea from my Ph.D. Ready? OK. Here it goes.

All the nodes belong to either one tree or the other. So you take the adjacency matrixes for the two trees and just... stick them together. Yep. That's it.

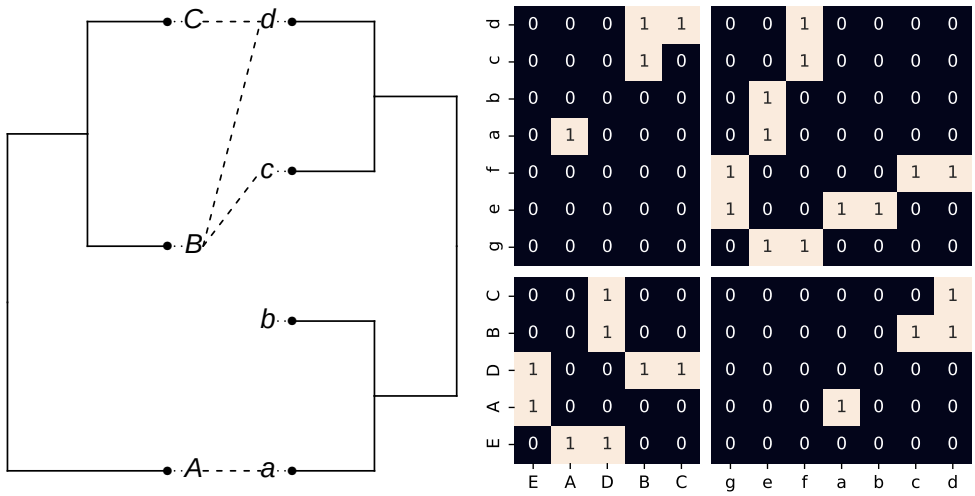
Graphs are absurdly flexible objects. They have the kind of omnivorous generality of a relational database, minus the need to actually *understand* your data. They are the perfect receptacle for mysterious, complicated stuff. So, you can represent a tree as a graph, as we've already done. Or, you can represent two trees. Graphs don't have to be fully connected, so we haven't broken any rules by pasting two graphs together. You just stick the adjacency matrixes together along their diagonals.

In the co-phylogeny, the leafs are connected by ecological data. For a host-parasite interaction, the leaf nodes are connected by the host range of the parasites. For a phylogenomics problem, the leaf nodes represent alleles that are linked by the genomes where they are found together. For microbiome data, the leaf nodes represent the sampled hosts and observed OTUs that are linked by the read count table. To express those links, you just find the row and column that correspond to the two leafs, and you put the quantity in that element.

Because the two trees are not connected by phylogenetic links, there are two empty blocks in the combined adjacency matrix above and below the diagonal. That is where the ecological links go. Because adjacency matrixes are all-by-all, the blocks representing each of the trees will be square, and so will the combined adjacency matrix. If the two trees aren't the same size, though, the blocks where the ecological links go will be rectangular. Figure 3 illustrates what this looks like.

There is one final consideration – normalization. Unless you are lucky enough to have time calibrated trees, the branch lengths of the trees are probably in different scales. The ecological observations are a totally different category of information. So, what to do?

As we'll see in following posts, we are building this graph so that we can compare it to other graphs representing co-phylogenies of other events. There are myriad rea-



(a) Cophylogeny graph **(b)** Adjacency matrix

Figure 3: The construction of the graph adjacency matrix of a cophylogeny.

sons why events with similar ecological and evolutionary dynamics might happen over vastly different scales – the generation time of the organisms involved, or the substitution rates of the marker genes used, or the population sizes... Suffice it to say that it makes the most sense to treat units of phylogenetic branch length in relative terms, and normalize the root-to-deepest-tip for all trees to 1.

To keep the ecological observations in scope, I have made the somewhat arbitrary choice to normalize them to be bounded by the longest and shortest branches found in the two trees. If they are presence-absence observations, then they receive edge weights equal to the average branch length found in the normalized trees.

In the next section, we will look at some relatively simple things we can do by representing co-phylogenies as graphs.

3 CO-PHYLOGENY GRAPHS WITH SUCHTREE

Without practical implementations, theory isn't much good to anyone but theorists. There are lots of excellent packages for working with phylogenetic trees. If you're not already using them, I would recommend `ape`, `phytools` and `geiger` if R is your thing, or `ete3`, `dendropy` and `scikit-bio` if you roll with python. However, these packages are really designed for working with trees that you might want to actually *look* at. As a figure in a manuscript, for example. They were not designed for trees with millions of taxa. Trees with millions of taxa are exactly what we get when we look at microbiomes.

For this, I have done what Ph.D. students often do, which is to write their own software package. I've done my best not to re-invent the wheel *too* much – there's no point in re-implementing what existing packages already do, and do very well. So, let me introduce you to `SuchTree`, which I sincerely hope will remain a one-trick pony among phylogenetics packages.

`SuchTree` aims to do one thing very well – speed. If you've read *Around the World in Eighty Days*, you might recall Phileas Fogg's crossing of the Atlantic aboard the steamship *Henrietta*. After running out of coal, Fogg buys the ship from the captain and has the crew burn every wooden part of the vessel :

> It was necessary to have dry wood to keep the steam up to the adequate pressure, and on that day the poop, cabins, bunks, and the spare deck were sacrificed. On the next day, the 19th of December, the masts, rafts,

> and spars were burned; the crew worked lustily, keeping up the fires. Passepartout hewed, cut, and sawed away with all his might. There was a perfect rage for demolition.
>
> The railings, fittings, the greater part of the deck, and top sides disappeared on the 20th, and the *Henrietta* was now only a flat hulk. But on this day they sighted the Irish coast and Fastnet Light. By ten in the evening they were passing Queenstown. Phileas Fogg had only twenty-four hours more in which to get to London; that length of time was necessary to reach Liverpool, with all steam on. And the steam was about to give out altogether!

This is basically how I designed `SuchTree`. If you want to tweak, edit, re-root, visualize, or even save your trees, you'll have to use some other tool. If you want to do traversals, like finding patristic distances, or most recent common ancestors, or tip-to-root depth, `SuchTree` does this with blazing speed. Everything else you might look for in a phylogenetics package went up the smokestack.

This might seem like a rather trivial sort of thing to optimize, but it is the only way to work with tress that capture microbiome-scale diversity. Consider, for example, the task of calculating patristic distances. Normally, if we were going to do something like the Hommola test [2], we would build distance matrixes for each of the trees and then look up corresponding distances (this is how it is implemented in 'scikit-bio'). Unfortunately, distance matrixes scale quadratically with the number of leafs in the tree. Fewer operations are required to look up the patristic distances in a matrix verses calculating it by traversing a tree, but as trees become very large, it becomes more important to contain the memory footprint than to have fewer operations.

Trees with millions nodes have distance matrixes with trillions of elements. Even if you have a machine with enough RAM (terabytes of it!), your CPUs will spend most of their time waiting for data. Fortunately, trees scale linearly with the number of leafs. A bifurcating tree with n leafs has a distance matrix of $n(n-1)$ elements, but the tree itself contains only $2n-1$ nodes. Trees with millions of nodes occupy mere megabytes of RAM. They can even fit into the on-chip cache of many modern CPUs, assuming you store them in a tidy data structure. `SuchTree` represents bifurcating trees as a single, immutable, one-dimensional block of memory, and preorders the nodes by how they are most likely to be accessed. This is pretty much the dumbest data structure that can store a tree. It would earn a solid B- on an Introduction Computer Science 101 exam. However, this design gives the CPU's speculative memory management hardware the best chance at guessing which memory pages it needs to pre-fetch. That means the CPUs stay nice and busy.

`SuchTree` is implemented in `Cython`, with the key components outside of Python's Global Interpreter Lock. Because the underlying data structure is immutable, tree-traversal operations on `SuchTree` objects are fully thread-safe. This turns out to be quite important when you do lots statistics on Very Big Trees.

So, how fast is it?

One very simple practical question that can be explored using `SuchTree` is to compare the structural dissimilarities among trees built using different phylogenetic inference methods. If I take two trees built from the same set of 54,327 OTUs using neighbor-joining and approximate maximum likelihood (`using FastTree` [3]) I can randomly sample pairs of OTUs and calculate the patristic distances between them through the NJ tree and through the ML tree, and then examine how well these distances are correlated (Figure 4). For one million random pairs of OTUs, `SuchTree` takes about 10 seconds to calculate the distances through the two trees. That's an average of about five microseconds per sample.

In addition to `SuchTree`, the package provides a helper class called `SuchLinkedTrees` which handles a lot of the annoying bookkeeping that crops up when you are working with pairs of linked phylogenetic trees. `SuchLinkedTrees` does following :

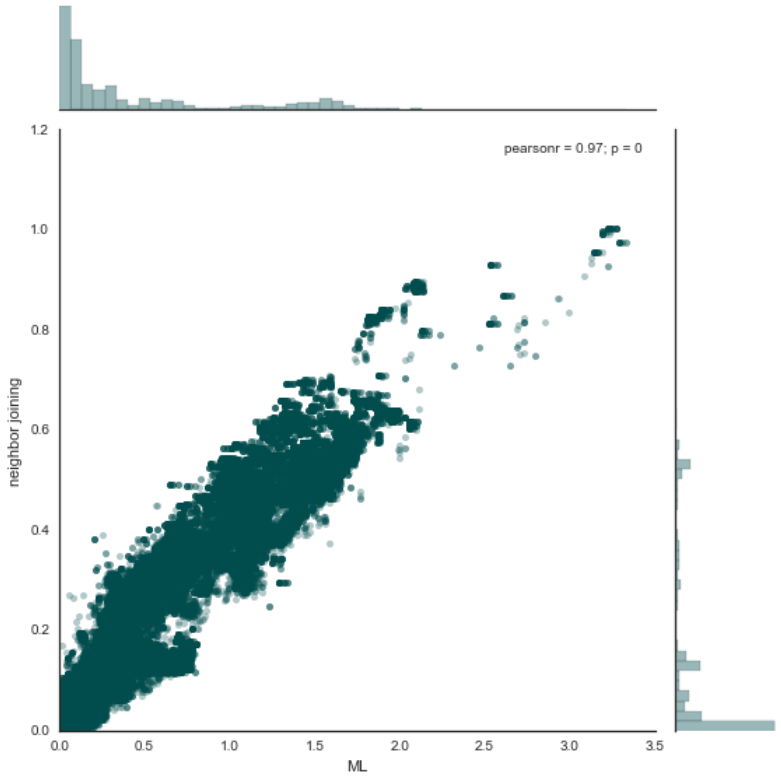


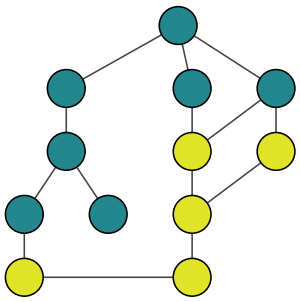
Figure 4: Trees were built from a microbiome dataset containing 54,327 taxa using approximate maximum likelihood and neighbor-joining. One million random pairs of taxa were sampled and patristic distances were computed for each pair using `SuchTree` objects representing the ML and NJ trees.

- It performs extensive input validation, and raises specific, meaningful error messages when the checks fail. Because linked trees are based on fairly complex pieces of data often are often assembled from different sources, useful error messages are a huge time-saver.
- It allows you to mask out pieces of one or both trees based on cladistic relationships. This makes it easy to examine statistical properties of embedded clades.
- It calculates linked patristic distances for clades with a relatively small number of taxa.
- It will sample linked patristic distances for clades with a large number of taxa.
- It calculates adjacency matrixes for any subsetted pair of clades.
- If the `igraph` package is installed, it will directly generate `Graph` objects.

4 A FIRST LOOK AT CO-PHYLOGENY GRAPHS

Although they are feature-rich, comparing graphs to one another it is not straightforward. One might start by asking the obvious, *do these two graphs have the same topology?* We thus face-plant into the famous graph isomorphism problem. It is NP-complete. The only way we know how to solve it is with computational bruit-force. Dang it.

Fortunately, finding exact topological congruence is not particularly useful to us. We want to look for motifs within the graph that correspond to events that happen



moment	quantity
average degree	2.896
σ degree	2.690
mean betweenness	9.514
σ betweenness	4.999
density	0.212
squareness	0.75
occupancy	1.143

Figure 5: Moments of the co-phylogeny graph from Figure 3. Degree, betweenness centrality, and density are traditional graph properties which can be calculated using `igraph`. The degree of a node is the number of edges connected to a node. Betweenness is a measure of how 'central' a node is within the graph. The density is the ratio of the number of edges to the number of possible edges. Squareness and occupancy are moments specific to co-phylogenies. Squareness is the ratio of the number of taxa in the 'host' tree to the number of taxa in the 'guest' tree. Occupancy is the ratio of the number of links between taxa to the number of taxa.

in nature. If we were a morally bankrupt tech company, we might be interested in identifying different processes by which humans form social cliques. This will allow us to help advertisers exploit those relationships to manipulate people so that they will buy awful things and vote for awful people.

Cliques that form by different processes will tend to have different topologies. For example, if we extract all the subgraphs that represent families, they will probably be structured somewhat differently than the subgraphs representing workplaces. So, if we can find topological motifs in a curated collection of workplaces and families, we can use those motifs to find all the workplaces and families in the social network.

We would like to do something similar, but hopefully with less techbro sociopathy. A microbiome co-phylogeny graph will also have cliques, and the topology of those cliques may have different motifs depending on the underlying process that created them. For example, an endosymbiosis is likely to be structured differently than a comensalism. To describe and detect these motifs, we need a way to find *similar* graphs.

Broadly speaking, there are two ways to go about this. You can calculate moments of the graph, or you can build similarity or dissimilarity metrics on graphs. We'll look at dissimilarity metrics of graph spectra in Part 3 and dissimilarity metrics based om graph kernels in Part 5.

A moment is a value that represents some property of a graph. For example, what is the average number of connections per node? What is the standard deviation of the number of connections per node? What is the smallest number of connections that must be severed in order to divide the graph into two pieces? Individually, a moment only says one thing about a graph's topology, but with a collection of moments we can develop a general idea of how the graph is structured.

In the next section, we will look at some more sophisticated moments on graph topology, and we'll start using them to cluster graphs.

5 USING SPECTRAL THEORY FOR MOMENTS AND CLUSTERING

If you take an infinitely long random walk from node to node within a graph, the frequency that each node will be visited relative to the others is called the *spectrum*

of the graph, and is a powerful way to simplify the topology of the graph into one dimension (there is also a convenient shortcut to calculating the frequencies without actually performing the random walk). Graph spectra are not guaranteed to be unique; graphs with different topologies can have the same spectra, and are called *isospectral graphs*. Nevertheless, the bigger the graph, the less likely any pair of them will be isospectral. One can extract moments from the one-dimensional spectrum, such as its skewness, kurtosis

6 BUILDING A FEATURE SPACE

7 CLUSTERING WITH GRAPH KERNELS

8 A GRAPH CO-PHYLOGENY FEATURE SPACE, ALL GROWN UP

9 MACHINE LEARNING, FINALLY

10 A GRAPH OF MANY TREES

11 FUNDING

RYN was funded by a grant from the Alfred P. Sloan Foundation to Jonathan A. Eisen.

References

[1] Mark S Hafner et al. "Disparate rates of molecular evolution in cospeciating hosts and parasites". In: *Science* 265.5175 (1994), p. 1087.

[2] Kerstin Hommola et al. "A permutation test of host–parasite cospeciation". In: *Molecular biology and evolution* 26.7 (2009), pp. 1457–1468.

[3] Morgan N Price, Paramvir S Dehal, and Adam P Arkin. "FastTree 2–approximately maximum-likelihood trees for large alignments". In: *PloS one* 5.3 (2010), e9490.