AN ABSTRACT OF THE THESIS OF

Kevin L.Vergin for the degree of Master of Science in Microbiology presented on January 6, 2015.

Title: Disturbances in a Marine Bacterioplankton System Drive Community Assembly Towards Niche-based Processes.

Abstract approved:

_____

Stephen J. Giovannoni

Over 100 monthly bacterioplankton DNA samples, from each of the surface and 200 m depths at the Bermuda Atlantic Time-series Study (BATS) site, were analyzed for community assembly processes. Correlation networks, filtered for potential autocorrelation artifacts, were constructed for each depth. Network characteristics for the two depths were remarkably similar and the nodal taxonomic units (NTUs) with the greatest number of connections for both networks were dominated by the SAR86 clade in the Gammaproteobacteria class. Categories of NTUs within each network constructed using a weighted, phylogenetically-based similarity of connections measure suggested ecological drift processes. A clustering approach based on Tarjan's algorithm used directed graph theory to link taxa by similarity in connections to other taxa, revealing apparent examples of niche-based processes among taxa that have similar connections. An algorithm that used hierarchical Dirichlet Processes to model neutral communities based on learned parameters indicated that community assembly processes were neutral at the local 200 m level but not at the 200 m metacommunity level nor at either level of surface samples. However, surface samples restricted to SAR11 NTUs supported the hypothesis of neutral assembly processes, suggesting that neutral processes may apply to lower taxonomic groupings within the whole community. The greater annual disturbance from the deep mixing event at the surface of BATS may alter dispersal rates and drive community assembly away from neutral processes towards niche-based processes.

Disturbances in a Marine Bacterioplankton System Drive Community Assembly Towards Niche-based Processes

by
Kevin L. Vergin

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented January 6, 2015
Commencement June 2015

Master of Science thesis of Kevin L. Vergin presented on January 6, 2015

APPROVED:

Major Professor, representing Microbiology

Chair of the Department of Microbiology

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Kevin L. Vergin, Author

ACKNOWLEDGEMENTS

I wish to express sincere appreciation to Dr. Stephen J. Giovannoni for over twenty years of support, encouragement, and training, to all of the members of the Giovannoni Laboratory past and present for friendship and stimulating scientific collaborations, and especially to my wife, Carol, for unconditional love and support.

Appendices A1 – A5 and E were written by Nick Jhirad. Appendix C was written by Charlotte Wickham. Jon Dodge helped implement the C^{++} script for the Dirichlet Process algorithm. Funding was provided by NSF Microbial Observatory grants and an investigator award from the Gordon and Betty Moore Foundation to Dr. Stephen Giovannoni.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

Chapter 1 - Introduction

Bacterial communities are ubiquitous in nature (Mouchka et al. 2010, Giovannoni & Vergin 2012, Bottos et al. 2014, Shafquat et al. 2014). However, community assembly processes, or constraints on species coexistence (Gotzenberger et al. 2012) are not easily resolved for a number of reasons (Ofiteru et al. 2010, Stegen et al. 2013). Defining phylogenetic units of interest, such as species, is an elusive goal (Vos 2011), so it is difficult to measure basic properties such as richness, the number of species in a population (Pedros-Alio 2006). Interactions between members of a community are also difficult to assess since it is nearly impossible to directly observe exchanges, especially in a complex, pelagic communities. Finally, long term studies are required to provide adequate statistical power for potential correlations and relationships (Rivera-Hutinel et al. 2012).

Samples from the Bermuda Atlantic Time-series Study (BATS) site in the Sargasso Sea provide an ideal opportunity to study bacterial community assembly processes. Monthly microbial samples have been collected for over a decade, yielding a rich dataset covering the upper 300 m of the water column (Steinberg et al. 2001, Lomas et al. 2013). The Sargasso Sea is extremely oligotrophic and is characterized by annual deep mixing events in late winter and early spring that extend frequently to 200 m. Summer waters are stably stratified as the surface layers warm. Sampling effort was most complete for surface and 200 m depths, making comparisons of major environmental drivers such as temperature, photic exposure, and annual mixing possible.

High throughput pyrosequencing was used to deeply probe 16S rRNA genes from over 384 samples from a nine year time series collected at the BATS site (Vergin et al. 2013b). These sequences were aligned and binned with a custom pipeline, PhyloAssigner, that uses pplacer (Matsen et al. 2010) to assign sequences to nodes of a reference phylogenetic tree (Vergin et al. 2013b). These assignments, which we call nodal taxonomic units (NTUs) accomplish two goals: they uniquely identify binned sequences by the reference sequence and they are embedded in a framework that allows quantification of the phylogenetic relationship between units. A previous study showed that phylogenetically grouped sequences in the SAR11 clade had characteristic ecological distributions and could be defined as ecotypes (Vergin et al. 2013b).

Networking analyses of this BATS time series dataset could reveal ecological processes. Other researchers have pioneered the use of association and correlation networks to discover potential relationships within bacterial communities (Steele et al. 2011, Xia et al. 2011, Faust et al. 2012). This approach has revealed and confirmed many ecological processes such as top-down control of bacterial communities (Chow et al. 2014), nitrogen fixation in corals (Rodriguez-Lanetty et al. 2013), and community succession in a salt marsh (Dini-Andreote et al. 2014). Cluster analyses of microbial networks, as has been applied in yeast gene metabolic networks (Said et al. 2004), may reveal additional insights.

The ultimate goal of this project was to address community assembly processes in the 1991-2004 BATS dataset. Hubbell's Unified Neutral Theory of Biodiversity and Biogeography (Hubbell 2001) provides a useful model for neutral processes. According to this theory, individuals are equal with respect to demographic rates of birth and death. At the opposite end of the conceptual spectrum is niche assembly theory, in which the community is determined by niches that filter species adapted to current sets of environmental parameters. More recent studies suggest that community assembly is not limited to the dichotomy of these two models, but more likely falls within a continuum between these extremes (Gravel et al. 2006). A recent study developed a machine learning algorithm to estimate parameters for the neutral model and compare the likelihood of the observed species abundance distributions with modeled communities at both the local and metacommunity level (Harris et al. 2014).

In this study, we used a time series dataset to test the hypothesis of neutral pelagic bacterial community assembly processes at BATS. We compared characteristics from networks constructed for both the surface and 200 m samples and probed for deeper relationships within the networks by applying clustering analyses. We also applied new machine learning algorithms to the abundance data to directly assess the degree of neutral community assembly at each depth. The comparison of surface and 200 m samples enables us to contrast annually disturbed surface samples with more stable 200 m samples, providing a unique opportunity to study factors that drive community assembly processes for bacterioplankton.

Chapter 2 - Literature review

Introduction

A major goal in the study of bacterial communities is to infer relationships between species. One method for finding relationships is network construction, where correlated species (nodes in the network) are connected (edges in the network). A network of correlations can be further studied for evidence of clustering and higher level relationships that may reveal underlying structuring of the community. Since individual bacterial cells and their interactions with other cells are difficult to observe directly, there are a number of challenges to re-constructing network interactions and detecting clustering within these networks. The study of bacterial communities by network analysis is still in early stages of method development, so there is not a consensus in the field about optimal methods. Several analytical approaches that have been suggested are reviewed here and the application of these methods to bacterial communities is discussed.

Most bacterial communities consist of hundreds, if not thousands, of different taxonomic groups. Taxonomic groups are difficult to define and delineate in the Division Bacteria but for the purposes of this review, I will refer to the lowest grouping of organisms with similar phenotype and distribution as “species”. It is likely that at the species level, organisms are sufficiently different that they can interact meaningfully with other species through resource competition and other mechanisms that inhibit reproductive success. Furthermore, it is generally accepted that species of bacteria in the same environment interact with other species at some level and are not wholly independent. These interactions can be mutualistic (both benefit), antagonistic (one benefits and other harmed), commensal (one benefits and other unaffected), and amensal (one harmed and other unaffected) (Faust & Raes 2012). These interactions may be detected by positive and negative correlations in abundance data used to construct networks.

One important goal of studying bacterial communities is to put observed relationships into an ecological context. One important question is: how are communities assembled? Neutral models assume that species are equivalent in terms of their likelihoods of birth and death, so they can be used as null models in comparisons to niche-based models, where species are

adapted to specific conditions and the community is assembled according to the niches present. Methods for learning community assembly pattern parameters and fitting neutral models are also presented and discussed in this review.

Background: Food web networks developed for macroorganisms

Using networks to study communities has a long history in the ecology of Eukaryotes. Commonly, networks were used to describe food webs. These relatively simple networks were developed by observing a few trophic levels such as what plants grew in an area, which animals ate those plants (herbivores), and what animals preyed on those herbivores (carnivores). With careful observation, it was possible to develop a comprehensive catalog of interacting species and accurately describe the nature of the interaction. These food webs were typically unidirectional although some considered cannibalism and other types of interactions such as pollination (Williams & Martinez 2000, Bascompte et al. 2003). Relatively few webs considered links such as parasitic and pathogenic interactions (Memmott et al. 2000).

Food web networks have been quantified in various ways, providing numbers that were used to compare food webs to find commonalities (Proulx et al. 2005). The total number of species is highly variable, depending on the extent of the study, but the number of links (interactions) divided by the total number of possible links (known as connectance) provides a useful metric for comparison across studies (Dunne et al. 2002). Other measures included path lengths (average number of connections between any pair of species) and the cumulative distribution of the number of connections for each species (Amaral et al. 2000, Dunne et al. 2004). Such measures have been used to characterize food webs, or networks. For example, networks have been identified as small world, where average path lengths are short, and scale-free, where the cumulative distribution of connections has a power law tail (Amaral et al. 2000). These measurements allowed for the generation of models, such as the niche model, to help explain network dynamics (Williams & Martinez 2000, Williams et al. 2002). A major goal was to anticipate the effects of extinction events by predicting the effect of disturbances on species in the network based on their connectedness. These potential disturbances are hypothesized to be increasing due to anthropogenic influences (Duffy 2002).

Networks have been generated for many other systems such as power grids, actors in films, and the world wide web (Amaral et al. 2000). These networks can be far more complex

than ecological food webs but can be described with similar statistics. The wide interest in network topology has resulted in advances in methods for analyzing patterns in network data (Hamill & Gilbert 2010, Decelle et al. 2011).

Bacterial community network construction methodologies

The development of network methods and models has coincided with the explosion of data available for various microbial environments and long term studies. Inexpensive next generation sequencing can interrogate species compositions in samples to a very high degree and with great accuracy. There is an obvious need for in-depth network analyses of microbial communities, but there are two large stumbling blocks slowing down progress in this field. First, sampling these environments is still a major challenge. Targeting of the 16S rRNA gene can identify members of the community, but it is not always clear if the presence of the gene correlates with active cells. In addition, biases in DNA extraction, amplification efficiency, rRNA copy number, and other biases make gene sequence abundance difficult to translate to cell numbers in the environment. Secondly, interactions tend to be indirect and nearly impossible to observe. Relationships common in the Eukaryotic realm, where organisms consume other organisms are only a part of the picture. Bacteria use products released by other organisms although likely there are still similar relationships such as mutualism, antagonism, etc.

A straightforward method for discerning relationships in the microbial environment is to calculate correlation coefficients such as Spearman (non-parametric, rank based comparison) and Pearson (parametric, direct linear comparison), especially with a time series experimental design. This method is attractive because intuitively, two interacting species should be correlated across time. For instance, in a mutualistic interaction where both species benefit, samples with one species in high abundance should create a favorable environment for the other species, resulting in a higher abundance and, thus, a correlation between the two species. If the correlation is linear, then it should be detected using Pearson or Spearman correlation coefficients. However, if the correlation does not follow a normal distribution and contains outliers, it is more likely to be detected by the Spearman correlation coefficient alone. These methods are the basis for a popular algorithm, LSA (Local Similarity Analysis) (Ruan et al. 2006a, Xia et al. 2011). It should be noted that a correlation is not evidence for an actual interaction between the two species. One method that might be able to determine causality, Convergent

Cross Mapping, requires a few hundred data points to reach a statistically valid conclusion (Sugihara et al. 2012). A drawback of LSA is that it does not appear to take into account potentially spurious correlations due to a common seasonal cycle.

Other methods have been developed that may be more suitable for other experimental designs. CoNet was developed to find co-occurrences of bacteria in different samples (Faust et al. 2012). This very conservative method uses an ensemble of four similarity methods, including Spearman and Pearson correlations, as well as distance measures such as Bray-Curtis and Kullback-Leibler divergence, to correct for possible biases due to mutual exclusions. The top few potential interactions from at least two of the methods are further subjected to significance testing. Separately, the data are analyzed using a machine learning generalized boosted linear modeling algorithm to determine the top few species that best predict the occurrence of a given species. These separate results are then merged and filtered to arrive at a final network of fewer, but highly supported, connections.

Another theory that has been applied to microarray data, but not high throughput sequencing data, is Random Matrix Theory (Luo et al. 2007), a method that was originally developed for modeling phase transitions in materials sciences. This co-occurrence method detects system-specific, nonrandom signals in complex assemblages. The method uses symmetric matrices and determines the distribution of nearest neighbor spacing of eigenvalues where Gaussian distributions are associated with random noise while Poisson distributions are associated with correlations. This method has been successfully applied to functional gene networks from soil bacteria (Zhou et al. 2010).

Cluster detection in network data

Once a network has been constructed, the next task is to discern patterns in the distribution of connections. Typically, this involves finding clusters (also called modules, subnetworks, or compartments) where connections within the nodes of the cluster are denser than connections to nodes outside the cluster. This is sometimes referred to as a K-mer problem since the number of clusters, K, is seldom known *a priori*. Again, these methods have been applied to food webs where compartments in a generalized niche model emerged because of the diet contiguity of component species (Williams & Martinez 2000). In addition, compartments can be related to the phylogeny, biomass, and habitat structure of the

component species where, for instance, closely related predators are in different clusters, presumably to reduce direct competition for prey (Rezende et al. 2009). However, many other disciplines are interested in network clustering. For example, clustering can be detected by removing edges with the highest betweenness values (presumably, these edges connect clusters), recalculating the network, and repeating the removal process until no connecting edges remain, although this is computationally difficult for larger networks (Girvan & Newman 2002). An algorithm that uses greedy optimization and exploits shortcuts in the optimization problem can run substantially faster, making larger networks (millions of nodes) more tractable (Clauset et al. 2004). More recently, machine learning techniques are being used to learn parameters and infer modules based on belief propagation algorithms (Decelle et al. 2011).

Another concept from ecology similar to clustering is hierarchy or nestedness, where generalist species are highly interconnected, but specialist species are more peripheral (Bascompte et al. 2003). A nestedness calculator was used to describe communities as consisting of highly stable core species and asymmetric specialist species. Greater hierarchy may lead to more controllability so a measure, Global Reaching Centrality, was developed to compare networks (Mones et al. 2012). With larger networks, it was shown that hierarchies exist in clusters and clusters are linked together to form networks (Olesen et al. 2007). Clustering methods have been applied to a yeast metabolic gene network to describe essential and toxicity-modulating clusters (Said et al. 2004).

Most microbial co-occurrence and correlation networks reported in the literature are descriptive and have not utilized clustering algorithms to reveal deeper insights. This level of network analysis may be due to limitations of the process for generating species-like units. The most popular method of grouping 16S rRNA sequence data is by Operational Taxonomic Units (OTUs). This method randomly selects a sequence and determines if the next randomly selected sequence is within a user-defined range, typically 97% similarity. If so, these sequences are put in the same unit and the next sequence is selected. If not, a new unit is created and the next sequence is compared against the two units using the same cutoff similarity. This process tends to collect sequences into units which do not necessarily represent coherent species, but may represent cohesive family units (Vergin et al. 2013b). In addition, the *de novo* creation of OTUs

precludes comparison to other studies, since the relationship between OTU designations is random and unknown.

Network characteristics for bacterial communities

Despite these limitations, many microbial environments, such as the human biome, soils, freshwater lakes, corals, and marine systems, have been the subjects of network analyses. Generally, these networks tend to be small world with short average path lengths, scale free with power law distributions in the cumulative distribution of connections, and modular with modularity indices greater than 0.4 (references in the following paragraphs). Examples of microbial networks from each of the environments listed above are provided in the following paragraphs.

Niche specialization was apparent in a very large human biome study that surveyed 18 body sites in each of 239 individuals (Faust et al. 2012). Even in physically close sites, such as sub- and supragingival plaques, clustering was visually apparent. In general, phylogenetically similar OTUs tended to inhabit similar niches and phylogenetically distant but functionally similar OTUs tended to exclude one another through competition. This study used CoNet to infer relationships, an appropriate method for this experimental design. Since samples were not time dependent, but were presumably stable, co-occurrences of bacterial groups between sites would be of interest. The large number of subjects for this study provided ample replication for statistical significance testing of potential associations.

Soil networks revealed two interesting patterns. In a study of 151 different samples from a variety of environments, generalist and specialist OTUs could be classified based on their appearance in a majority or minority, respectively, of samples (Barberan et al. 2012). While the generalists included representatives from well known bacterial groups, such as Acidobacteria, others, such as Crenarchaeota, were rare but perhaps provided an important biochemical function. Another study examined temporal niche partitioning in a natural succession of soil types in a salt marsh and found that early stages of succession were characterized by high complexity networks, while complexity decreased with later stages of succession (Dini-Andreote et al. 2014). This surprising result was opposite to macrobial ecosystems where succession results in a more varied environment with an abundance of potential niches. It is possible that in this system, the final successional stage was richer in organic material, but also had a high salt

content, which may have acted as a selecting force to favor the fewer organisms capable of adapting to these more extreme conditions.

Abundant Alphaproteobacteria in lake environments were less connected to other species and environmental variables than the less abundant Betaproteobacteria which were highly connected in a dimictic freshwater lake in Sweden (Eiler et al. 2012). Redundancy of ecological roles was also hypothesized due to the clustering of more closely related taxa (tribes in this manuscript). The highly connected taxa are hypothesized to be keystones in the system and might have added importance in propagating disturbances.

In a similar manner, the abundant bacterial group *Oceanospirillum* also was not well connected and outside the main cluster in a network obtained from Caribbean corals (Rodriguez-Lanetty et al. 2013). However, one member of *Oceanospirillum* in the main cluster was associated with a *Rhodobacterales*, a group that is dominant in the early developmental phase of the coral, 4-5 days after larval release. Interestingly, a link to Rhizobiales was also demonstrated, suggesting a coral-diazotroph association that may be important for fixing nitrogen in support of the symbiotic dinoflagellate, *Symbiodinium*. Another study of coral networks included viromes in the analysis and found that viruses were associated with a small number of bacteria, but bacteria were associated with many viruses, indicating a complex relationship between the two entities (Soffer et al. 2014).

Other marine environments have also been sampled and subjected to network analysis. In the Gulf of Triesta in the Mediterranean Sea, a coordinated response to freshwater inputs from the Isonzo River was apparent (Tinta et al. 2014). In the Western English Channel, bacteria were found to interact more strongly with each other than with either Eukaryotes or environmental variables (Gilbert et al. 2012). In perhaps the system most well-studied by network analysis, the San Pedro Ocean Time Series study site has provided several new insights. Similar to the Western English Channel, bacteria formed the most connections compared to Eukaryotes and Archaea (Steele et al. 2011). In a shorter, but more frequently sampled time series, networks of non-time delayed interactions were dominated by bacteria to bacteria and virus to virus interactions, while bacteria to virus interactions tended to be delayed by two days (Needham et al. 2013). In a longer, monthly time series, separate networks for surface samples and deep chlorophyll maximum samples shared few interactions (Chow et al. 2013). Surface

networks including protists and viruses revealed specific interactions, suggesting that both top-down and bottom-up processes are important for regulating bacterial communities (Chow et al. 2014). In a recent analysis of viromes in the Pacific Ocean, networks had few connections between samples from surface, photic zones and deep, aphotic zones (Hurwitz et al. 2014). Also, some samples could be distinguished by their proximity to shore or other environmental variables, suggesting that gradient drivers for viruses are similar to the gradient drivers of bacterial communities.

Community Assembly

Network analyses of bacterial communities, in many of these cases, seem to reveal that environmental gradients are drivers of community assembly. There are two main theories for community assembly, niche-based and neutral. Intuitively, niche-based assembly processes are attractive because it seems likely that species that are well adapted to the conditions found in a region will out-compete less well adapted species and will be more prevalent (Keddy 1992). Alternatively, neutral community assembly postulates that individuals are equal in terms of their birth and death likelihoods and that migration maintains diversity in a constant population size (Hubbell 2001). Hubbell's Unified Neutral Theory of Biodiversity and Biogeography links local communities, where migration is the main force introducing variation, to the overall metacommunity, where speciation introduces variation. Although this theory is simple, it seems to capture many features of natural communities. This theory predicts ecological drift of species as they fluctuate in abundance in a random walk towards extinction. The neutral model can be seen as a null model of community assembly, so many groups have developed methods to estimate the critical parameters for the model. The fundamental biodiversity constant, theta, is a dimensionless number that measures speciation-drift in the metacommunity (Hubbell 2001). The fundamental dispersal number, I (Etienne 2005, Alonso et al. 2006), measures the migration-drift in the local community and is related to the migration rate, m (Hubbell 2001, Rosindell et al. 2011). Parameters can be estimated from single, large samples, but improved estimates came from two stage calculations from several small samples (Munoz et al. 2007). Population genetics inspired further parameter estimation refinement (Munoz et al. 2008). Phylogenetics can be used to distinguish between alternate values of theta and m using Approximate Bayesian Computation, a technique also from population genetics (Jabot & Chave 2009).

Early attempts to apply community assembly theories to natural systems returned mixed results, suggesting that a simple dichotomy of processes was insufficient to explain natural systems. It is now recognized that a continuum between neutrality and niche-based systems better explains most environments (Gravel et al. 2006). For instance, phytoplankton sizes in the Western English Channel were best explained using a mixed emergent neutrality model (Scheffer & van Nes 2006, Vergnon et al. 2009). Phylogeny, implying niche conservatism in more closely related taxa, was used to estimate the effect of habitat filtering, or niche-based processes, on immigration rates in several tropical tree communities (Munoz et al. 2014). Testing for neutral assembly at the local and metacommunity levels has recently been accomplished using the hierarchical Dirichlet Process machine learning algorithm which is equivalent to the neutral model for large population sizes (Harris et al. 2014). This algorithm showed that human gut microbiomes were niche constrained (Harris et al. 2014).

Conclusion

In conclusion, microbial communities are well suited to network analyses and broad trends can be discerned. More detailed and rigorous clustering analyses are needed to address broader ecological theories, such as community assembly processes. However, method refinement may be necessary first to allow for better, consistent designation of fine scale phylogenetic taxa resolution. Without better phylogenetic resolution, it is difficult to conclude with confidence that two species-like entities are interacting, since an OTU may represent several different species. A consistent method of labeling species-like groups would also facilitate cross-comparisons between studies. It is currently nearly impossible to compare specific interactions and infer how those relationships might change between different environments. Many marine microbial groups have cosmopolitan distributions so it would be of great interest to know if species in one area have similar relationships to the same species in a different area. Network analysis has the potential to elucidate global phenomena, but methods must converge to be able to compare results and resolve interesting relationships.

Chapter 3 - Methods

Sample collection, nucleic acid isolation, PCR amplification, pyrosequencing procedures, and pyrosequencing data processing

Details for sample collection and processing were described previously (Vergin et al. 2013a). Briefly, in this study, 454 pyrosequencing data amplified from the V1 and V2 region of the 16S ribosomal RNA gene from 209 monthly samples at the BATS site (approximately nine years of samples) were used to construct surface (106 samples) and 200 m networks (103 samples; Table 1). Additional depth profile samples from the entire dataset (384 total samples) were used for some analyses (Vergin et al. 2013a). For the entire dataset, means of 6684 sequences with a mean of 257 bp length were generated. PhyloAssigner, a custom designed PERL pipeline (Vergin et al. 2013b) which incorporates the phylogenetic placement algorithm, pplacer (Matsen et al. 2010), grouped sequences into well defined phylogenetic groups, and a comprehensive backbone phylogenetic tree (http://aforge.awi.de/gf/project/phyloassigner), was used to sort the sequence data into NTUs. NTUs are therefore similar to more familiar OTUs (operational taxonomic units), except that NTUs are defined phylogenetically by reference sequences and clade structure rather than by cutoff thresholds. Although OTU binning using cutoff thresholds is computationally expedient, it has been demonstrated that phylogenetic placement using this method results in a loss of fine-scale phylogenetic information that can be important for resolving ecotypes (Koeppel & Wu 2013, Vergin et al. 2013b). NTUs were classified by phylum (or class in the case of Proteobacteria) and clade (SAR11, SAR86, SAR116, SAR202, SAR276, SAR324, SAR406, Plastids, and Roseobacter) based on their position in the reference phylogenetic tree.

Table 1. Time-series samples amplified for 16S rRNA pyrosequencing as indicated by an X.

Year	Month	BATS#	Surface	200	MLD(m)
1991	8	35	X	X	18
	9	36	X	X	38
	10	37	X	X	24
	11	38	X	X	56
	12	39	X	X	86
1992	1	40	X	X	132
	2	41	X	X	**238**
	3	42	X	X	32
	4	43	X	X	40
	5	44	X	X	20
	6	45	X	X	16
	7	46	X	X	22
	8	47	X	X	24
	9	48	X	X	16
	10	49	X	X	60
	11	50	X	X	64
	12	51	X		84
1993	1	52	X		104
	2	53	X	X	132
	3	54	X	X	**210**
	4	55	X	X	106
	5	56	X	X	12
	6	57	X	X	28
	7	58	X	X	10
	8	59	X	X	32
	9	60			30
	10	61	X	X	34
	11	62	X	X	56
	12	63	X	X	58
1994	1	64		X	114
	2	65	X	X	132
1997	9	108	X	X	38
	10	109	X	X	32
	11	110	X	X	40
	12	111	X	X	102
1998	1	112	X	X	76
	2	113	X	X	144
	3	114	X	X	**212**
	3	114 A	X	X	92
	4	115 A	X	X	116
	5	116	X	X	38
	6	117	X	X	16
	7	118	X	X	16

Table 1 (Continued)

	8	119		X	22
	9	120	X	X	40
	12	123	X	X	76
1999	1	124	X	X	108
	2	125	X	X	128
	3	126	X		122
	4	127			**208**
	5	128			
	6	129			26
	7	130			16
	8	131	X	X	30
	9	132	X	X	36
	10	133	X	X	48
	11	134	X	X	70
	12	135	X		
2000	1	136	X	X	171
	2	137	X	X	138
	2	137 A	X	X	247
	3	138	X	X	**248**
	4	139	X	X	46
	5	140	X	X	19
	6	141	X	X	23
	7	142	X		26
	8	143	X	X	9
	9	144	X	X	32
	10	145	X	X	47
	11	146	X	X	76
	12	147	X	X	96
2001	1	148	X	X	146
	2	149	X	X	91
	2	149 A	X	X	55
	3	150	X	X	127
	3	150 A	X		**158**
	4	151	X	X	52
	5	152	X	X	45
	6	153	X	X	8
	7	154	X	X	28
	8	155	X	X	22
	9	156	X	X	39
	10	157	X	X	38
	11	158	X	X	64
	12	159	X	X	88
2002	1	160	X	X	
	1	160 A	X	X	
	2	161	X	X	**158**

Table 1 (Continued)

	2	161 A			128
	3	162	X	X	152
	3	162 A	X	X	
	4	163	X	X	49
	5	164	X	X	27
	6	165	X	X	25
	7	166	X		19
	8	167	X	X	13
	9	168	X	X	48
	10	169	X	X	35
	11	170	X	X	65
	12	171	X	X	75
2003	1	172	X	X	112
	2	173	X	X	**215**
	2	173 A	X	X	
	3	174	X	X	90
	3	174 A	X	X	104
	4	175	X	X	31
	5	176	X	X	47
	7	177	X	X	19
	7	178	X	X	17
	8	179	X	X	25
	9	180	X	X	26
	10	181	X	X	53
	12	183	X	X	99
2004	1	184 A	X	X	**193**

Depths (Surface and 200) are in meters below the surface. MLD = Mixed Layer Depth, calculated as the depth where sigma-*t* was equal to sea surface sigma-*t* plus an increment in sigma-*t* equivalent to a 0.2°C temperature decrease (Sprintall & Tomczak 1992). The month of deepest mixing is indicated in bold font. Red X's indicate samples used for the analysis of residuals for non-normal distribution after seasonal differencing (Appendix B).

Calculating networks from Spearman and Pearson correlations

Amplicon sequence abundance was normalized for each sample. These percentages were then applied to the total cell count for each sample to get a relative abundance with reduced compositional bias. Samples with missing cell counts were estimated from the means of all samples from the same depth and month (relative to the month of deepest mixing). Time series of relative abundance for pairs of NTUs were compared using Spearman and Pearson correlations, as implemented in LSA (Ruan et al. 2006b) with the significance level adjusted for

multiple testing using the Benjamini-Hochberg correction (Appendix A.1-5; all algorithms were written for R (R core team 2013)). Both positive and negative correlations were detected, but time lags were not considered due to the high number of gaps in the time series data. The cumulative distribution of NTU connections was also calculated (Appendix A.6).

Network pruning using diagnostic filtering to remove spurious correlations

Potentially significant correlations were filtered in three steps (Appendix B). First, linear regression analysis was used to test the linearity of the relationship between the two NTUs (F statistic p value < 0.05). Second, since seasonality was a strong component in the data, samples were seasonally differenced, linear regressions were re-modeled, and residuals were examined using autocorrelation functions to detect non-normal distributions. Since seasonal differencing is sensitive to gaps in the time series, the >100 samples were trimmed to six complete years and partial years were removed (Table 1). Data for missing months were extrapolated by averaging through the one month prior and one month after the missing sample(s). Simulations demonstrate that non-normality in the residuals of only one NTU does not affect the type I error rate (appendix C) so the full time series was used for subsequent analyses. Finally, assumptions about the residuals in the linear regression analysis (independence and normal distribution) were examined using the autocorrelation function to detect non-random patterns. Initially, the residuals for potential linear correlations were tested and scored for non-normality. The total number of correlations with non-normal residuals for each NTU was recorded and the NTUs in the top 5% were identified. A Boolean matrix was then constructed where potential correlations between two NTUs both from the top 5% were scored as one and all others were scored 0. This matrix was then subtracted from the potential correlation matrix to yield the final screened matrix of correlations, hereafter referred to as “connections”.

Calculating the Phylogenetically Weighted Connectivity

A new similarity coefficient, called Phylogenetically Weighted Connectivity (∂), was developed to calculate similarities for all pairwise comparisons of NTUs (e.g. NTU A and NTU B) based on their shared connections to other NTUs. Since PhyloAssigner provides evolutionary distances between NTUs, a weighting system was developed that considered near phylogenetic

neighbors in the calculation of this similarity metric. For example, in the comparison of NTU A to NTU B, sliding windows considered the five nearest phylogenetic neighbors of all NTUs connected to A and B, in both directions within the tree. To facilitate this, profiles consisting of all connected NTUs + or – five nodes from the NTU being profiled were constructed for all NTUs. Each connected node in the profile was then scored as plus or minus one, depending on the sign of the correlation, and then neighboring NTUs were weighted inversely to phylogenetic distance from the connected NTU. Thus, we created a metric of the similarity between the connections of any two nodes, taking into account the network relationships of nodes closely related to the nodes being compared. Primer 6.0 software (Clarke & Gorley 2006) was then used to construct a non-metric multidimensional scaling plot and to conduct PERMANOVA analyses after categorizing by phylogeny.

To undertand this strategy, consider the following example. Suppose a network of university professors was made based on who knows whom. All professors have a list of professors that they know (also known as a profile). Suppose we wanted to calculate similarities between lists. A faculty member in microbiology who studies marine microorganisms works with oceanographers, and a particular oceanographer ("X") is on their list. Now consider a faculty member in physics who studies ocean wave dynamics and works with another oceanographer ("Y"). When we calculate a similarity between the microbiologist and the physicist (who are not necessarily connected), the oceanographers ("X" and "Y") are not a match but we can assign a weight to the relationship because they are both in the same department. Thus, the similarity between the microbiologist and physicist is higher because both have connections to oceanographers. By analogy, we infer that they play more similar roles in the university ecosystem because they both study oceanographic topics, although the overall similarities may not be high because they may not work with the same people. Subsequent analyses may use department affiliations as categories in the same way that phylogentic classifications are used for the BATS network.

Clustering using Tarjan's algorithm

Tarjan's algorithm (Tarjan 1972), a method for finding strongly connected components of a graph, was used to find clusters (also called "cores") based on connections within each network. Similarities between NTUs based on direct connections to other NTUs were again

calculated, except that phylogenetic relatedness was not considered, since a more direct similarity measure was desired (Appendix E). For a given network, obtained as described above (Network pruning using diagnostic filtering to remove spurious correlations), Tarjan's algorithm starts with one NTU (e.g. NTU A) and links it to the NTU with the most similar connections (e.g. NTU B). The algorithm then again identifies the NTU with the most similar connections to NTU B, and links it to form a chain. This process continues until the chain is linked back to the first NTU (A) to form a cluster. Clusters are formed by iteration such that subsequent iterations of the algorithm consider all of the correlations in a given cluster as a unit. Cluster composition was assessed by calculating mean pairwise phylogenetic distances using custom software (Appendix F) and comparing that mean to the distribution of means from 1000 permuted clusters, each with the same number of NTUs as the original clusters. Distributions of sixteen randomly selected NTUs from four clusters (third Tarjan iteration), at the depth for which the network was constructed, were represented using Ocean Data View software. The similarity of cluster composition (second Tarjan iteration) was calculated between surface and 200 m matrices and compared to permuted clusters from the 200 m matrix (Appendix G). Random clusters were generated from the list of 200 m NTU's and were apportioned as with the original clusters.

To understand this approach, and to contrast it with the previous weighted approach, again consider the hypothetical university professor network. In this case, only the greatest similarity of direct connections is considered. A chain may be formed from the microbiologist to oceanographer "X", to a zoologist who studies coral disease, to a fisheries biologist who studies marine fish ecology, to an ecologist who studies fungal interactions, and back to the microbiologist to form a cluster. This cluster would be described as a life science community, whereas the physicist might be in a separate physical science community. Both communities might include members from many different departments, and the communities would function in parallel to each other.

The Unified Neutral Theory of Biodiversity ecological parameters

The Unified Neutral Theory of Biodiversity (Hubbell 2001) uses three parameters to characterize sample population distributions: theta, a dimensionless fundamental biodiversity constant that measures speciation-drift in the metacommunity; I, a fundamental immigration number that measures the migration-drift in the local community; and m, the migration rate.

These parameters were estimated using the untb package in R (Hankin 2007). Cell counts at BATS vary within an order of magnitude, but there is a systematic bias towards greater numbers at the surface. Therefore, theta was not assumed to be uniform and, consequently, the G_{ST} method in the untb package was used to estimate m and I values. Relationships between these estimates, as well as temperature and cell counts, for each sample were graphed in contour plots using Ocean Data View software (Schlitzer 2014).

Individual samples (local communities) from both surface and 200 m and combined surface and 200 m samples (metacommunity) were assessed for fit to a neutral model using published software (Harris et al. 2014). The parameters for the neutral assembly model were estimated from the data by modeling the data as a hierarchical Dirichlet process (Harris et al. 2014). A Gibbs sampler, a type of Bayesian Markov chain Monte Carlo (MCMC) algorithm, then generated samples from the posterior distribution of the parameters given the data. Thus, the likelihood of the data was compared against a distribution of likelihoods from an artificial data matrix generated using the learned parameters. Data were trimmed to exclude NTU's with zero abundance at the depth under consideration (1031 and 960 NTUs for surface and 200 m, respectively). Surface samples pared to include only the SAR11 clade (63 NTUs) were also analyzed.

Chapter 4 - Results

Correlation networks were generated for two depths, surface and 200 m, from monthly samples collected at the BATS site over a discontinuous nine-year time span. An algorithm based on LSA (Ruan et al. 2006a) detected 45,980 and 43,153 putative significant correlations for the surface and 200 m networks, respectively (Table 2). Three diagnostic algorithms were subsequently applied to the networks to reduce the likelihood of type I errors (reject the null hypothesis of no significant correlation when the null hypothesis is true). Two of the diagnostic algorithms assessed the residuals from linear regressions calculated from all significant correlations between NTUs. The second diagnostic algorithm applied seasonal differencing to the time series and eliminated correlations with underlying non-normality in the linear regression residuals. The final diagnostic algorithm eliminated potentially spurious correlations due to underlying non-normality of linear regression residuals in non-differenced data. After these diagnostics were used to remove correlations that were identified as potential type I errors, the networks had 35,571 and 38,150 correlations for the surface and 200 m networks, respectively. Correlations that passed these diagnostic tests are referred to as “connections” throughout this document.

Table 2. Number of correlations at each step of diagnostic testing and final network characteristics.

		Surface	200 m
Network correlations	Spearman and Pearson	45980	43153
	Confirmed by linear regression	40435	39762
	Seasonal differencing check	35749	38150
	Dual ACF check	35571	38150
Network characteristics	Connectance	0.029	0.031
	Average path length	2.55	2.55
	Average edge betweeness	47.2	43.2
	Clustering coefficient	0.51	0.50

Network characteristics were remarkably similar for the two networks and comparable to many other reported microbial and food web networks ((Dunne et al. 2002, Steele et al. 2011, Barberan et al. 2012); Table 2). The small connectance values and short average path lengths indicate small world characteristics for these networks. A cumulative distribution of node connectivity had a complex shape but could be approximated by a truncated power law function. As was the case for the other characteristics, coefficients for the power law function were similar for the two networks (a=0.087 and 0.063, b=0.026 and 0.026 for surface and 200 m, respectively; Figure 1).

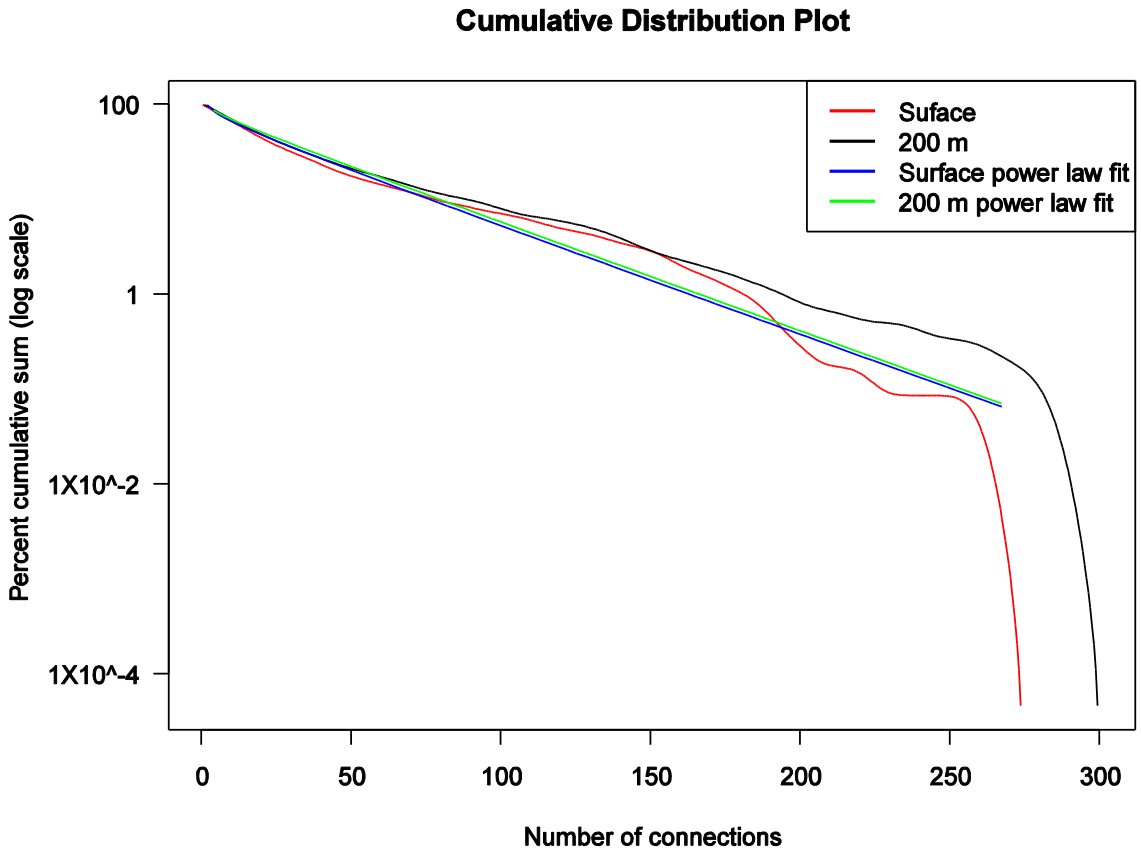

Figure 1. Cumulative distribution plot for the number of connections for each NTU (X axis). Cumulative percent is shown on the Y axis. Surface network NTUs are shown in red and 200 m network NTUs are shown in black. Truncated power law fits for surface (blue) and 200 m (green) are included. Power law fits were calculated from $F(x)= x^{-a}e^{-bx} + c$.

The NTUs with the greatest number of connections for each network were identified (Table 3). Interestingly, nearly all of these NTUs, for both networks, are in the class Gammaproteobacteria, many of which belong to the SAR86 clade, a ubiquitous and diverse phylogenetic group of uncultivated marine bacteria. These NTUs are simultaneously present throughout the time series, and were classified as Frequent Abundant in a previous analysis (Vergin et al. 2013a) (Figure 2).

Table 3. NTUs with the greatest number of connections.

Network	NTU	Number of Connections	Phylogenetic group
Surface	519	243	Gammaproteobacteria
	521	215	Gammaproteobacteria
	933	203	Alphaproteobacteria
	791	200	Gammaproteobacteria
	524	191	Gammaproteobacteria
	497	188	Gammaproteobacteria/SAR86
	420	184	Gammaproteobacteria/SAR86
	455	184	Gammaproteobacteria/SAR86
200 m	455	268	Gammaproteobacteria/SAR86
	457	268	Gammaproteobacteria/SAR86
	467	255	Gammaproteobacteria/SAR86
	456	249	Gammaproteobacteria/SAR86
	420	235	Gammaproteobacteria/SAR86
	418	223	Gammaproteobacteria/SAR86
	792	209	Gammaproteobacteria
	415	208	Gammaproteobacteria/SAR86

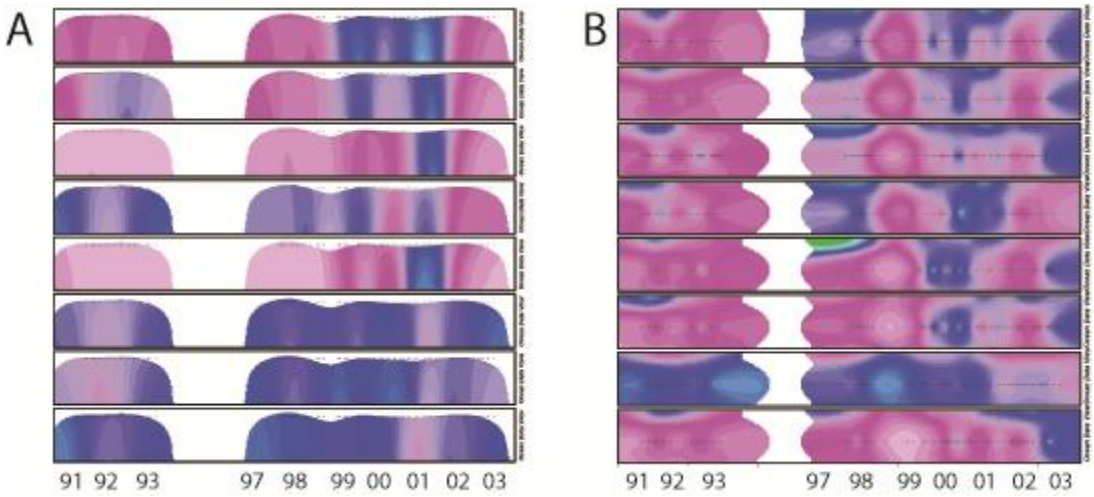

Figure 2. The most connected NTUs are nearly always present in the system. These contour plots for the eight NTUs with the greatest number of connections at the surface (A) and 200 m (B). Relative sequence abundances are represented as a heat map scaled from light pink (low abundance) to light blue (high abundance). The time series data are scaled by year on the X axes. Depths on the Y axes are from 0 to 30 (A) and 180 to 220 m below the surface (B).

Phylogenetically weighted connectivity coefficients (ð), described in the methods, revealed that phylogenetically similar NTUs tend to have similar connections to other NTUs. ð matrices were constructed for the networks, NTUs were sorted into phylogenetic categories (phylum/class or family), and PERMANOVA analyses were used to test the hypothesis that variance within a phylogenetic category was significantly different from a random expectation. Within the ð matrix, pairwise comparisons of phylogenetic categories were significantly different than random in most cases (Table 4).

Table 4. PERMANOVA results for phylogenetic groups within the ð matrix

	Phylum/Class		Clade	
Network	Pseudo-F[a]	Pairwise[b]	Pseudo-F	Pairwise
Surface	5.74	129/231	10.4	34/36
200 m	1.04	121/210	1.07	36/36

a – Pseudo-F statistic from the main test comparison. All pseudo-p values are < 0.001.
b – Number of significant pairwise tests compared to the total number of tests.

It was reasoned that two NTUs from the same community should have more similar connections than two NTUs from different communities. Tarjan's algorithm (Tarjan 1972) was used to find these directed similarity clusters (Tarjan clusters) in an iterative fashion by finding the NTU (or cluster in subsequent iterations) with the greatest similarity based on the NTUs to which it is connected. In the first iteration, hundreds of clusters consisted of two NTUs that were most similar to each other. In the second and third iterations, the number of clusters condensed to about 50 and 10, respectively, for each network (Table 5) and ranged in size from 3 to 110 NTUs in the second iteration, and 9 to 344 NTUs in the third iteration. The mean pairwise phylogenetic distance for NTUs in each Tarjan cluster was calculated and compared to 1000 randomly generated clusters of the same size. Between 20 and 40% of the Tarjan clusters had mean pairwise phylogenetic distances in, or below, the lower 2.5% of the random distribution and several clusters were above, or in, the upper 2.5% of the distribution (Table 5).

Table 5. Directed similarity cluster characteristics from Tarjan's algorithm analysis after 2 and 3 iterations.

Network	Iteration	Clusters	Min	Max	Low MPPD	High MPPD
Surface	2	49	4	86	10	0
Surface	3	11	13	270	4	1
200 m	2	52	3	110	15	3
200 m	3	10	9	344	4	2

Min – Minimum cluster size
Max – Maximum cluster size
MPPD – Mean pairwise phylogenetic distance; Low- pseudo-$p < 0.025$; High- pseudo-$p > 0.975$

Tarjan clusters were compared between the surface and 200 m networks for the second iteration clusters. It was hypothesized that two NTUs that are connected at the surface would also be connected at 200 m. For a given NTU, membership in its surface cluster was compared to membership in its 200 m cluster by measuring the similarity of composition of the respective Tarjan clusters. These similarities were compared to 1000 randomly permuted clusters for the 200 m network. Although cluster composition similarities were low in general, there was convincing evidence for conserved connections between depths for the observed data compared to randomly permuted clusters (Figure 3).

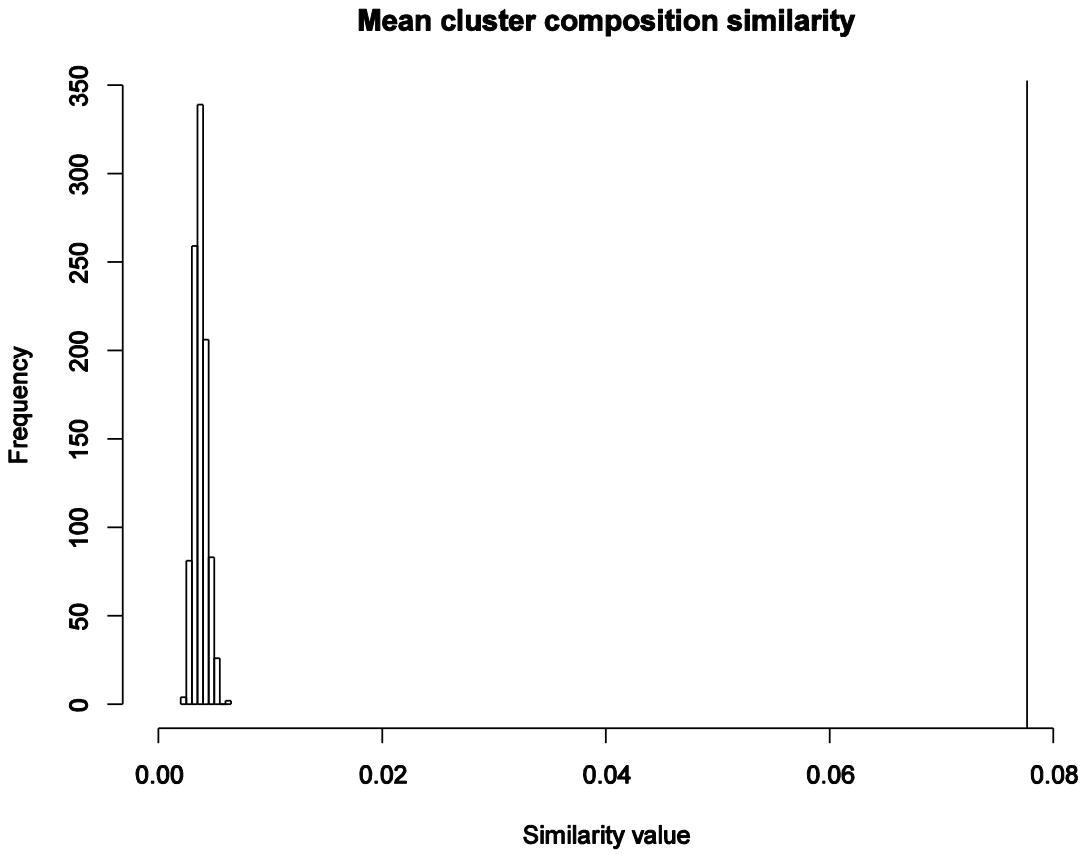

Figure 3. Histogram of mean Tarjan cluster composition pairwise similarities between the surface and 1000 simulations of 200 m network clusters. The observed pairwise similarity value from the data for Tarjan clusters after two iterations of Tarjan's algorithm is shown as a line.

Striking pattens emerged when the third iteration Tarjan clusters were plotted in time series (Figure 4). For example, in some cases (e.g. Fig. 4A) the members of a Tarjan cluster occur episodically, and rarely "bloom" contemporaneously. Most clusters are composed of a wide variety of phyla (Table 6), but many clusters are significantly less diverse than sets of randomly generated clusters. It should be noted that the NTUs with the greatest number of correlations were in the same Tarjan's cluster respectively for the surface and 200 m (Figure 2).

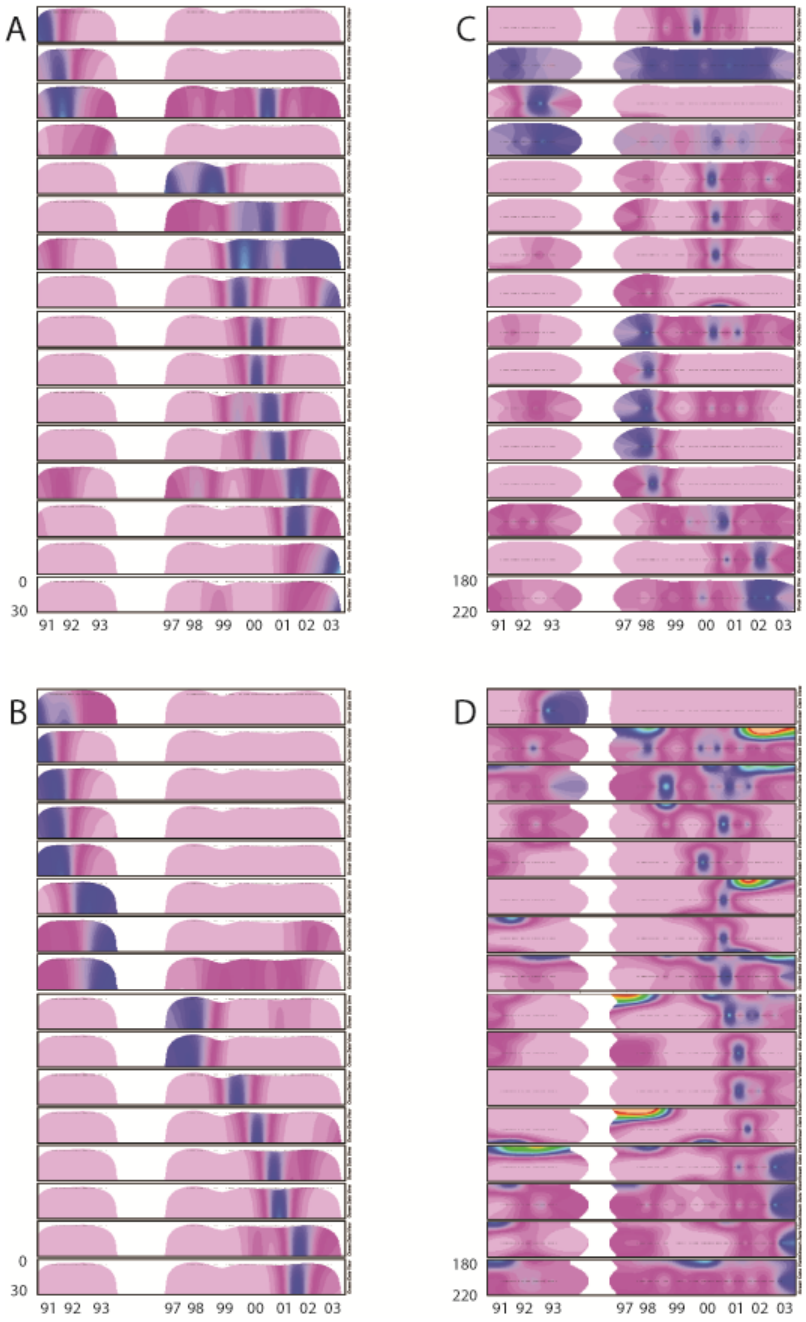

Figure 4. Contour plots for four selected third iteration directed similarity clusters from Tarjan's algorithm for the surface network (A and B) and 200 m network (C and D). Heat map scaling and axes are as described for Figure 2. NTUs representing each cluster were randomly selected. Plot details are provided in Table 6.

Table 6. NTU identity and phylogeny for data in Figure 4.

NTU[a]	Network	Directed Similarity Cluster[b]	Phylum/Class
859	surface	A	Alphaproteobacteria
1567	surface	A	Flavobacteria
1960	surface	A	Cyanobacteria
836	surface	A	Alphaproteobacteria
755	surface	A	Gammaproteobacteria
767	surface	A	Gammaproteobacteria
782	surface	A	Gammaproteobacteria
754	surface	A	Gammaproteobacteria
825	surface	A	Alphaproteobacteria
189	surface	A	Gammaproteobacteria
1196	surface	A	Alphaproteobacteria
624	surface	A	Betaproteobacteria
482	surface	A	Gammaproteobacteria
160	surface	A	Gammaproteobacteria
914	surface	A	Alphaproteobacteria
342	surface	A	Gammaproteobacteria
1796	surface	B	Flavobacteria
895	surface	B	Alphaproteobacteria
1584	surface	B	Flavobacteria
1025	surface	B	Alphaproteobacteria
829	surface	B	Alphaproteobacteria
1337	surface	B	Epsilonproteobacteria
83	surface	B	Gammaproteobacteria
937	surface	B	Alphaproteobacteria
525	surface	B	Gammaproteobacteria
183	surface	B	Gammaproteobacteria
2093	surface	B	Firmicutes
867	surface	B	Alphaproteobacteria
480	surface	B	Gammaproteobacteria
2006	surface	B	Cyanobacteria
2114	surface	B	Firmicutes
481	surface	B	Gammaproteobacteria
753	200 m	C	Gammaproteobacteria
1852	200 m	C	Bacteria
886	200 m	C	Alphaproteobacteria
1938	200 m	C	Verrucomicrobia
2060	200 m	C	Cyanobacteria
1952	200 m	C	Cyanobacteria
1645	200 m	C	Flavobacteria
909	200 m	C	Alphaproteobacteria

Table 6 (Continued)

1668	200 m	C	Flavobacteria
145	200 m	C	Gammaproteobacteria
2057	200 m	C	Cyanobacteria
1865	200 m	C	Acidobacteria
1184	200 m	C	Alphaproteobacteria
956	200 m	C	Alphaproteobacteria
563	200 m	C	Betaproteobacteria
1454	200 m	C	Deltaproteobacteria
2699	200 m	D	Chloroflexi
1957	200 m	D	Cyanobacteria
1960	200 m	D	Cyanobacteria
1973	200 m	D	Cyanobacteria
1812	200 m	D	Flavobacteria
1684	200 m	D	Flavobacteria
1539	200 m	D	Flavobacteria
2038	200 m	D	Cyanobacteria
505	200 m	D	Gammaproteobacteria
2108	200 m	D	Firmicutes
2369	200 m	D	Firmicutes
314	200 m	D	Gammaproteobacteria
1472	200 m	D	Deltaproteobacteria
1278	200 m	D	Alphaproteobacteria
1230	200 m	D	Alphaproteobacteria
1956	200 m	D	Cyanobacteria

[a] – NTUs are ordered top to bottom in the corresponding parts of Figure 4.
[b] – Letters in this column are from parts of Figure 4.

Neutral community parameters were estimated from relative abundance data for the entire dataset (384 samples). In addition, surface and 200 m depth strata were each tested for neutral community assembly processes using a published algorithm (Harris et al. 2014). The migration rate parameter, m, increased in surface waters during winter months corresponding to the deep mixing period and was much lower in the summer months when the water column was stratified (Figure 5A). At 200 m, the migration rate parameter, m, varied less throughout the year. The fundamental dispersal number, I, followed a similar pattern (Figure 5B) with higher values at the surface during deep mixing and lower values in surface stratified waters. There was strong evidence for non-neutral assembly processes for both surface and 200 m metacommunities (pseudo-p=0, test to reject neutral assembly) and marginal evidence for non-

neutral assembly processes for surface local samples (pseudo-p=0.038). However, 200 m local samples were consistent with a model of neutral assembly processes (pseudo-p=0.996). When surface samples consisted of only SAR11 clade NTUs, neutral assembly processes were supported at both the metacommunity and local levels (pseudo-p > 0.70).

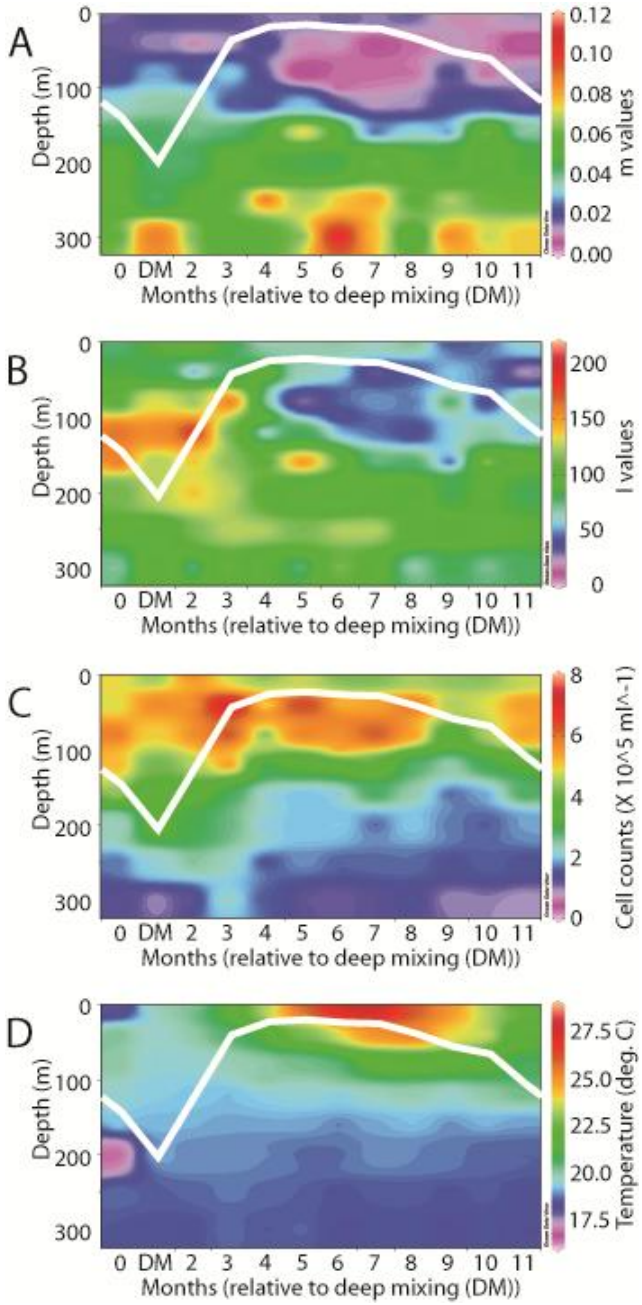

Figure 5. Contour plots for (A) migration rates, m, (B) fundamental dispersal number, I, (C) cell counts, and (D) temperature for all time series samples plotted as an average over a one year time span. The plots are indexed to the month of deepest mixing (DM) for each year, designated as month 1 (X axis). Sample depth is indicated on the Y axis. The white line represents the average mixed layer depth. Heat map scales are indicated to the right of the plot.

Chapter 5 - Discussion

We chose to modify an existing algorithm, LSA (Ruan et al. 2006b), to study time series data from BATS. The modified approach we took retained the Spearman and Pearson correlations from LSA, but filtered out potentially spurious correlations by linear modeling and examination of residuals from linear regressions for non-normal behavior. Since time series data are not independent (abundance tends to be related to the previous and subsequent time points), we examined the autocorrelation functions of the residuals to detect non-normal patterns. 17.3% of all correlations were eliminated as a result of the diagnostic steps. Non-normality in the residuals of linear regression analyses are most likely due to seasonality in the time series data, so it is interesting to note that fewer correlations were eliminated from the 200 m network (5,003) than the surface network (10,409), where seasonality drives community turnover (Gilbert et al. 2012, Giovannoni & Vergin 2012). The networks that emerged had small world characteristics very similar to those observed previously by other investigators using different construction methods (Dunne et al. 2004, Steele et al. 2011).

Phylogenetic community ecology attempts to model evolutionary processes resulting from the selection pressures of ecological processes. This idea has been applied to other networks and has demonstrated that phylogenetically similar species interact with the same set of species and/or occupy the similar positions in the network (Mouquet et al. 2014, and references therein). We sought to apply similar analyses to this bacterial network, so we implemented a new metric, ð, to examine the relationship between phylogenetic relatedness – which was readily available from PhyloAssigner – and the connections of NTUs. ð measures the similarity of two taxa in terms of shared connections within the network, but it takes into account not just the connected taxa being compared, but also their nearest phylogenetic neighbors. By comparing the ð values for NTU pairs, we reasoned that we could find sets of NTUs with similar roles in the community.

This analysis supported the role of ecological drift in community assembly by showing that phyogenetically related taxa tend to form similar connections. The similarity of connections increased (ð values were higher) with decreasing phylogenetic distance. We interpreted this evidence as support for the ecological drift hypothesis because it shows that related taxa tend to have similar network relationships, implying that they are interchangeable in communities

(Table 4). Interestingly, some of the most highly connected taxa were closely related phylogenetically, suggesting the possibility of drift among keystone taxa (Steele et al. 2011, Eiler et al. 2012).

Tarjan's algorithm has not been used widely in community ecology, but is well known among graph theorists as a tool for finding strongly connected nodes within a network. Because Tarjan's algorithm extends a cluster by finding the strongest directly linked node, forming a chain in an iterative process, we reasoned it could reveal clusters with strong interactions. In this analysis, phylogenetic relatedness was not considered. Two iterations of the Tarjan algorithm resulted in about 50 clusters at both the surface and 200 m. These smaller, more tightly linked clusters were compared to determine if NTUs maintained similar connections at the surface and at 200 m. There was not a high degree of overlap, but there was convincing evidence that many NTUs maintained similar relationships at both depths (Figure 3). This result is surprising, suggesting that relationships between some NTUs may persist over a broad range of environmental conditions.

Tarjan clusters formed after the third iteration of the algorithm revealed that NTUs in the clusters had overlapping temporal distributions and the clusters tended to encompass transitions over the entire time series (Figure 4). Many of these clusters contained a large proportion of phylogenetically related NTUs, possibly indicating divergence into ecotypes, while other clusters covered a broader phylogenetic range, suggesting NTUs with more general roles. These results suggest that NTUs appear to occupy temporally overlapping niches. This process of niche turnover may be occurring simultaneously for different Tarjan clusters at both depths.

We modeled community assembly processes at BATS using methods developed to test Hubbell's Unified Neutral Theory of Biodiversity (Hubbell 2001), which seeks to understand whether neutral or niche-based processes dominate assembly. The ð analyses reported above provided support for the ecological drift model, a prediction from Hubbell's neutral theory, by showing that phylogenetically related taxa share similar connections. Hubbell theorized that local communities form stochastically from a metacommunity. In his model, population size remains constant as all resources are used by the individuals in the local community. As individuals die, there is an equal chance that any individual will reproduce or a variable chance that a member of the metacommunity will migrate to replace the expired individual, thus

maintaining a constant population. The bacterioplankton community at BATS, in principle, can provide a good test of the neutral model because bacterial populations fluctuate over a small range, especially within depth strata, thus meeting the model's assumptions (Figure 5C). In this analysis, the entire time series at each depth is interpreted as a metacommunity with temporal and spatial dimensions, while the individual sample points are interpreted as the local communities. Comparing the data to neutral models, we found no support for the neutral assembly hypothesis at the metacommunity level, for either the surface or 200 m datasets. But, for local communities, at the surface, neutral assembly was rejected at the p=0.05 level, but not the p=0.01 level, indicating marginal support for non-neutral assembly. However, the evidence was strong for neutral assembly processes in local communities at 200 m.

The difference in community assembly processes between the surface and 200 m can be explained by the annual deep mixing event in the BATS system. The migration rate parameter, m (Figure 5A), and fundamental dispersal number, I (Figure 5B), for the full dataset, including depth profile samples (Vergin et al. 2013b), shows a change in dispersal during the deep mixing period in the late winter/early spring. Barriers to dispersal are probably reduced during deep mixing due to evenly distributed water temperatures (Figure 5D) and physical mixing from storm systems. By summer, water temperatures at the surface increase, resulting in a stable, stratified water column that likely increases barriers to dispersal, resulting in a lower diversity community at the surface. This observation is consistent with an analysis we reported previously that suggested that transport resulting from disturbances created opportunities for growth and immigration of rare taxa (Vergin et al. 2014a). The seasonal fluctuation between cold, nutrient rich, mixed waters and warm, nutrient depleted, stratified waters likely drives habitat filtering that limits the dispersal of individuals from the surface metacommunity. An analogous example of dispersal limitation was described in an unconfined aquifer system (Stegen et al. 2013). BATS deep mixing sometimes extends to 200 m, where there is a demonstrable, but weak, seasonality (Morris et al. 2005, Vergin et al. 2013a). Thus, the prevalence of neutral processes at 200 m is likely related to weaker forces driving changes in community structure (Hellweger et al. 2014).

Neutral community assembly was apparent within broad phylogenetic catagories, such as the SAR11 clade. When the surface data was trimmed to include only SAR11 NTUs, there was strong evidence for neutral community assembly, despite the influence of deep mixing. This

result is similar to human gut microbiome community assembly for intra-taxon groups (Harris et al. 2014) and ammonia oxidizing bacteria in wastewater treatment microbial communities (Ofiteru et al. 2010). The results presented here are consistent with other studies of natural systems and suggests that ecological drift may be a common feature for lower taxonomic groups. Niche conservatism is more likely preserved between more closely related taxonomic groups so community assembly may depend less on the selection of a particular species. However, particular species, once established, may alter their environments to favor other species giving the impression of temporally transitioning clusters. SAR11 has previously been reported to consist of ecotypes with distinct spatial and temporal distributions (Vergin et al. 2014b), but the neutral patterns reported here are likely due to the numerical domination of one or two subclades.

Chapter 6 - Conclusion

The framework provided by BATS and our application of PhyloAssigner provided us with opportunities to explore plankton community networks from a new perspective. Overall, plankton community networks at BATS have small world characteristics that are surprisingly like those that have been described in other plankton systems. Phylogenetic distances from PhyloAssigner allowed us to show that, within the networks, phylogenetically related taxa share similar connections. This suggested that related taxa can sometimes subsititute for each other, providing support for the hypothesis that neutral processes can play a role in plankton community assembly. The potential importance of neutral processes in community assembly was illustrated by the recent demonstration (Hellweger et al. 2014) that bacterioplankton biogeography can be impacted by neutral drift.

Hubbell's Unified Neutral Theory of Biodiversity (Hubbell 2001) provides a context for evaluating the relative contributions of habitat filtering and neutral processes to community assembly. Applying tools that were developed to assess the fit of data to the neutral theory, we found that metacommunities at both depths follow niche assembly processes, while local mesopelagic communities appear to follow neutral assembly processes. A major difference between the surface and mesopelgic is the impact and duration of mixing – a physical process that transports cells and brings nutrients to the surface. The alternating transition from a disturbed, mixed state to a stable, stratified state may create divergent niches that filter potential migrants and limit the establishment of conditions necessary for competitive exclusion. Even in the mesopelagic, where mixing is less pronounced, disrupted conditions may allow habitat filtering to influence metacommunity composition, but not enough to inhibit neutral assembly at local community levels. The data also suggested that, within diverse clades, such as SAR11, habitat filtering applies to phylogenetic branches, but within these branches neutral processes may prevail; in other words, the high resolution of PhyloAssigner can reveal neutral processes within clades that conform to the ecotype model.

It is striking that two related pelagic plankton environments – the surface and upper mesopelagic, can harbor networks that are very similar in many network characteristics, but be very different in how neutral processes impact community assembly. We attribute this difference to the effects of seasonal forcing, which impose a continuous pattern of change on

the surface system. In the mesopelagic, neutrality becomes important. These findings will be important to oceanographers who are interested in biogeochemistry because the findings focus attention on aspects of diversity that are correlated with significant functional differences between taxa, thereby leading to simpler models of community structure for the purpose of understanding elemental cycles.

References

Alonso D, Etienne RS, McKane AJ (2006) The merits of neutral theory. Trends Ecol Evol 21:451-457

Amaral LAN, Scala A, Barthelemy M, Stanley HE (2000) Classes of small-world networks. Proc Natl Acad Sci U S A 97:11149-11152

Barberan A, Bates ST, Casamayor EO, Fierer N (2012) Using network analysis to explore co-occurrence patterns in soil microbial communities. ISME J 6:343-351

Bascompte J, Jordano P, Melian CJ, Olesen JM (2003) The nested assembly of plant-animal mutualistic networks. Proc Natl Acad Sci U S A 100:9383-9387

Bottos EM, Scarrow JW, Archer SDJ, McDonald IR, Cary CS (2014) Bacterial community structures of Antarctic soils. In: Cowan DA (ed) Antarctic Terrestrial Microbiology, Book 1. Springer-Verlag, Berlin, Germany

Chow CE, Kim DY, Sachdeva R, Caron DA, Fuhrman JA (2014) Top-down controls on bacterial community structure: microbial network analysis of bacteria, T4-like viruses and protists. ISME J 8:816-829

Chow CE, Sachdeva R, Cram JA, Steele JA, Needham DM, Patel A, Parada AE, Fuhrman JA (2013) Temporal variability and coherence of euphotic zone bacterial communities over a decade in the Southern California Bight. ISME J 7:2259-2273

Clarke KR, Gorley RN (2006) PRIMER v6: User manual/tutorial. In. PRIMER-E Ltd., Plymouth, United Kingdom

Clauset A, Newman MEJ, Moore C (2004) Finding community structure in very large networks. Phys Rev E 70

Decelle A, Krzakala F, Moore C, Zdeborova L (2011) Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. Phys Rev E 84

Dini-Andreote F, de Cassia Pereira ESM, Triado-Margarit X, Casamayor EO, van Elsas JD, Salles JF (2014) Dynamics of bacterial community succession in a salt marsh chronosequence: evidences for temporal niche partitioning. ISME J 8:1989-2001

Duffy JE (2002) Biodiversity and ecosystem function: the consumer connection. Oikos 99:201-219

Dunne JA, Williams RJ, Martinez ND (2002) Network structure and biodiversity loss in food webs: robustness increases with connectance. Ecol Lett 5:558-567

Dunne JA, Williams RJ, Martinez ND (2004) Network structure and robustness of marine food webs. Mar Ecol Prog Ser 273:291-302

Eiler A, Heinrich F, Bertilsson S (2012) Coherent dynamics and association networks among lake bacterioplankton taxa. ISME J 6:330-342

Etienne RS (2005) A new sampling formula for neutral biodiversity. Ecol Lett 8:253-260

Faust K, Raes J (2012) Microbial interactions: from networks to models. Nat Rev Micro 10:538-550

Faust K, Sathirapongsasuti JF, Izard J, Segata N, Gevers D, Raes J, Huttenhower C (2012) Microbial co-occurrence relationships in the human microbiome. PLoS Comp Biol 8:e1002606

Gilbert JA, Steele JA, Caporaso JG, Steinbruck L, Reeder J, Temperton B, Huse S, McHardy AC, Knight R, Joint I, Somerfield P, Fuhrman JA, Field D (2012) Defining seasonal marine microbial community dynamics. ISME J 6:298-308

Giovannoni SJ, Vergin KL (2012) Seasonality in ocean microbial communities. Science 335:671-676

Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proc Natl Acad Sci USA 99:7821-7826

Gotzenberger L, de Bello F, Brathen KA, Davison J, Dubuis A, Guisan A, Leps J, Lindborg R, Moora M, Partel M, Pellissier L, Pottier J, Vittoz P, Zobel K, Zobel M (2011) Ecological assembly rules in plant communities - approaches, patterns, and prospects. Biol Rev 87:111-127

Gravel D, Canham CD, Beaudet M, Messier C (2006) Reconciling niche and neutrality: the continuum hypothesis. Ecol Lett 9:399-409

Hamill L, Gilbert N (2010) Simulating large social networks in agent-based models: A social circle model. Emergence: Complex and Org 12:78-94

Hankin RKS (2007) Introducing untb, an R package for simulating ecological drift under the unified neutral theory of Biodiversity. J Stat Softw 22:1-15

Harris K, Parsons TL, Ijaz UZ, Lahti L, Holmes I, Quince C (2014) Linking statistical and ecological theory: Hubbell's unified neutral theory of biodiversity as a hierarchical Dirichlet process. arXiv:1410.4038 [q-bio.PE], Cornell University

Hellweger FL, van Sebille E, Fredrick, ND (2014) Biogeographic patterns in ocean microbes emerge in a neutral agent-based model. Science 345:1346-1349

Hubbell SP (2001) The unified neutral theory of biodiversity and biogeography, Princeton University Press, Princeton

Hurwitz BL, Westveld AH, Brum JR, Sullivan MB (2014) Modeling ecological drivers in marine viral communities using comparative metagenomics and network analyses. Proc Natl Acad Sci U S A 111:10714-10719

Jabot F, Chave J (2009) Inferring the parameters of the neutral theory of biodiversity using phylogenetic information and implications for tropical forests. Ecol Lett 12:239-248

Keddy PA (1992) Assembly and Response Rules - 2 Goals for Predictive Community Ecology. J Veg Sci 3:157-164

Koeppel AF, Wu M (2013) Surprisingly extensive mixed phylogenetic and ecological signals among bacterial Operational Taxonomic Units. Nucleic Acids Res 41:5175-5188

Lomas MW, Bates NR, Johnson RJ, Knap AH, Steinberg DK, Carlson CA (2013) Two decades and counting: 24-years of sustained open ocean biogeochemical measurements in the Sargasso Sea. Deep-Sea Res Pt II 93:16-32

Luo F, Yang Y, Zhong J, Gao H, Khan L, Thompson DK, Zhou J (2007) Constructing gene co-expression networks and predicting functions of unknown genes by random matrix theory. BMC Bioinform 8:299

Matsen FA, Kodner RB, Armbrust EV (2010) pplacer: linear time maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed reference tree. BMC Bioinform 11:538

Memmott J, Martinez ND, Cohen JE (2000) Predators, parasitoids and pathogens: species richness, trophic generality and body sizes in a natural food web. J Anim Ecol 69:1-15

Mones E, Vicsek L, Vicsek T (2012) Hierarchy Measure for Complex Networks. PLoS One 7

Morris RM, Vergin KL, Cho J-C, Rappe MS, Carlson CA, Giovannoni SJ (2005) Temporal and spatial response of bacterial lineages to annual convective overturn at the Bermuda Atlantic time-series study site. Limnol Oceanog 50:1687-1696

Mouchka ME, Hewson I, Harvell CD (2010) Coral-Associated Bacterial Assemblages: Current Knowledge and the Potential for Climate-Driven Impacts. Integr Comp Biol 50:662-674

Mouquet N, Devictor V, Meynard CN, Munoz F, Bersier L-F, Chave J, Couteron P, Dalecky A, Fontaine C, Gravel D, Hardy OJ, Jabot F, Lavergne S, Leibold M, Mouillot D, Munkemuller

T, Pavoine S, Prinzing A, Rodrigues ASL, Rohr RP, Thebault E, Thuiller W (2014) Ecophylogenetics: advances and perspectives. Biol Rev 87: 769-785

Munoz F, Couteron P, Ramesh BR (2008) Beta diversity in spatially implicit neutral models: A new way to assess species migration. Am Nat 172:116-127

Munoz F, Couteron P, Ramesh BR, Etienne RS (2007) Estimating parameters of neutral communities: From one single large to several small samples. Ecology 88:2482-2488

Munoz F, Ramesh BR, Couteron P (2014) How do habitat filtering and niche conservatism affect community composition at different taxonomic resolutions? Ecology 95:2179-2191

Needham DM, Chow CET, Cram JA, Sachdeva R, Parada A, Fuhrman JA (2013) Short-term observations of marine bacterial and viral communities: patterns, connections and resilience. ISME J 7:1274-1285

Ofiteru ID, Lunn M, Curtis TP, Wells GF, Criddle CS, Francis CA, Sloan WT (2010) Combined niche and neutral effects in a microbial wastewater treatment community. Proc Natl Acad Sci U S A 107:15345-15350

Olesen JM, Bascompte J, Dupont YL, Jordano P (2007) The modularity of pollination networks. Proc Natl Acad Sci U S A 104:19891-19896

Pedros-Alio C (2006) Marine microbial diversity: can it be determined? Trends Microbiol 14:257-263

Proulx SR, Promislow DEL, Phillips PC (2005) Network thinking in ecology and evolution. Trends Ecol Evol 20:345-353

Rezende EL, Albert EM, Fortuna MA, Bascompte J (2009) Compartments in a marine food web associated with phylogeny, body mass, and habitat structure. Ecol Lett 12:779-788

Rivera-Hutinel A, Bustamante RO, Marin VH, Medel R (2012) Effects of sampling completeness on the structure of plant-pollinator networks. Ecology 93:1593-1603

Rodriguez-Lanetty M, Granados-Cifuentes C, Barberan A, Bellantuono AJ, Bastidas C (2013) Ecological Inferences from a deep screening of the Complex Bacterial Consortia associated with the coral, Porites astreoides. Mol Ecol 22:4349-4362

Rosindell J, Hubbell SP, Etienne RS (2011) The unified neutral theory of biodiversity and biogeography at age ten. Trends Ecol Evol 26:340-348

Ruan Q, Dutta D, Schwalbach MS, Steele JA, Fuhrman JA, Sun F (2006a) Local similarity analysis reveals unique associations among marine bacterioplankton species and environmental factors. Bioinform 22:2532-2538

Ruan QS, Dutta D, Schwalbach MS, Steele JA, Fuhrman JA, Sun FZ (2006b) Local similarity analysis reveals unique associations among marine bacterioplankton species and environmental factors. Bioinform 22:2532-2538

Said MR, Begley TJ, Oppenheim AV, Lauffenburger DA, Samson LD (2004) Global network analysis of phenotypic effects: Protein networks and toxicity modulation in Saccharomyces cerevisiae. Proc Natl Acad Sci USA 101:18006-18011

Scheffer M, van Nes EH (2006) Self-organized similarity, the evolutionary emergence of groups of similar species. Proc Natl Acad Sci U S A 103:6230-6235

Schlitzer R (2014) Ocean Data View. http://www.awi.de/

Shafquat A, Joice R, Simmons SL, Huttenhower C (2014) Functional and phylogenetic assembly of microbial communities in the human microbiome. Trends in Microbiology 22:261-266

Soffer N, Zaneveld J, Vega Thurber R (2014) Phage-bacteria network analysis and its implication for the understanding of coral disease. Environ Microbiol

Sprintall J, Tomczak M (1992) Evidence of the Barrier Layer in the Surface-Layer of the Tropics. Journal of Geophysical Research-Oceans 97:7305-7316

Steele JA, Countway PD, Xia L, Vigil PD, Beman JM, Kim DY, Chow CE, Sachdeva R, Jones AC, Schwalbach MS, Rose JM, Hewson I, Patel A, Sun F, Caron DA, Fuhrman JA (2011) Marine bacterial, archaeal and protistan association networks reveal ecological linkages. ISME J 5:1414-1425

Stegen JC, Lin X, Fredrickson JK, Chen X, Kennedy DW, Murray CJ, Rockhold ML, Konopka A (2013) Quantifying community assembly processes and identifying features that impose them. ISME J 7:2069-2079

Steinberg DK, Carlson CA, Bates NR, Johnson RJ, Michaels AF, Knap AH (2001) Overview of the US JGOFS Bermuda Atlantic Time-series Study (BATS): a decade-scale look at ocean biology and biogeochemistry. Deep Sea Research Part II: Topical Studies in Oceanography 48:1405-1447

Sugihara G, May R, Ye H, Hsieh CH, Deyle E, Fogarty M, Munch S (2012) Detecting causality in complex ecosystems. Science 338:496-500

Tarjan R (1972) Depth-First Search and Linear Graph Algorithms. SIAM J Comp 1:146-160

R core team (2013) R: A language and environment for statistical computing. In. R Foundation for Statistical Computing, Vienna, Austria

Tinta T, Vojvoda J, Mozetič P, Talaber I, Vodopivec M, Malfatti F, Turk V (2014) Bacterial community shift is induced by dynamic environmental parameters in a changing coastal ecosystem (northern Adriatic, northeastern Mediterranean Sea) – a 2-year time-series study. Environ Microbiol doi: 10.1111/1462-2920.12519

Vergin K, Done B, Carlson C, Giovannoni S (2013a) Spatiotemporal distributions of rare bacterioplankton populations indicate adaptive strategies in the oligotrophic ocean. Aquatic Microbial Ecology 71:1-13

Vergin KL, Beszteri B, Monier A, Cameron Thrash J, Temperton B, Treusch AH, Kilpert F, Worden AZ, Giovannoni SJ (2013b) High-resolution SAR11 ecotype dynamics at the Bermuda Atlantic Time-series Study site by phylogenetic placement of pyrosequences. ISME J 7:1322-1332

Vergnon R, Dulvy NK, Freckleton RP (2009) Niches versus neutrality: uncovering the drivers of diversity in a species-rich community. Ecol Lett 12:1079-1090

Vos M (2011) A species concept for bacteria based on adaptive divergence. Trends Microbiol 19:1-7

Williams RJ, Berlow EL, Dunne JA, Barabasi AL, Martinez ND (2002) Two degrees of separation in complex food webs. Proc Natl Acad Sci U S A 99:12913-12916

Williams RJ, Martinez ND (2000) Simple rules yield complex food webs. Nature 404:180-183

Xia LC, Steele JA, Cram JA, Cardon ZG, Simmons SL, Vallino JJ, Fuhrman JA, Sun FZ (2011) Extended local similarity analysis (eLSA) of microbial community and other time series data with replicates. Bmc Syst Biol 5

Zhou JZ, Deng Y, Luo F, He ZL, Tu QC, Zhi XY (2010) Functional Molecular Ecological Networks. mBio 1

Appendices

Appendix: A LSA scripts

Appendix A.1: LSA file formatting

```
#Starting from a csv file with samples in columns and NTUs in rows.
#Sample depths, months, and years are in separate rows after the data.

#For this data set, earlier samples were in columns 287:385 and later
#samples were in columns 2:286.



tenv = new.env()




Bermuda_Filename = "~\Desktop\R_files\BermudaDataSamp.csv"



SDS <- read.csv(Bermuda_Filename)



rownames(SDS) <- SDS[,1]



#OTUS <- 1:1386



#Want just the numeric values

data = 2:385

data1 = 287:385

data2 = 2:286
```

```
no_dup_data = which(SDS['repeats',] == 0)


#convert depth scale 1:9 to the actual measured depths

depthconv = c(0,40,80,100,120,160,200,250,300)


#finish depth conversions

unconv_depth = as.numeric(SDS['depth',data])

SDS['depth',data] = depthconv[unconv_depth]



#structure by date

#monthread = SDS['month',data]

#yearread = SDS['year',data]

#absdate = yearread*12 + monthread
```

Appendix A.2: LSA correlations

#LSA Code

#i: represents varying on unit i

#j: represents varying on unit j

#w: represents varying on timeslot w

#d: represents varying on delay

#Order of subscripts indicates order in array, row, column... etc..

#POPiw = population for unit i at timeslot k

#RNiw = rank normal score for unit i at timeslot k

#LSAij_d = Local similarity score for unit i, unit j, delay d

#CORNij_d = correlation using rank normal for unit i, unit j, delay d

#CORij_d = correlation for unit i, unit j, delay d

#CORNSij_d = Correlation (using rank normal) significance for unit i, unit j, delay d

#CORSij_d = Correlation significance for unit i, unit j, delay d

###Note: I could split this up into 2 timeslots to track the exact pair of times that produced a particular

###Combination, but as of yet, I can't use this, better to make the array more manageable and just track 1 slot

```
###All I need right now is the series for a given delay, not all the exact times


#Tw = vector of absolute times associated with timeslot k


#returns LSA score
LSA <- function(x) {
        yP = 0
        yN = 0
        yMaxP = 0
        yMaxN = 0
        tmp = x
        tmp[is.na(tmp)]=0

        for(i in tmp) {
                yP = max(yP+i,0)
                yN = min(yN+i,0)
                yMaxP = max(yMaxP,yP)
                yMaxN = min(yMaxN,yN)
        }
        if(yMaxP>abs(yMaxN)) {
        return(yMaxP)
        }
        else {
```

```
		return(yMaxN)

		}


}



normalTransform <- function(POPiw) t(apply(POPiw,1,function(x)
qnorm(rank(x,na.last="keep")/(1+sum(!is.na(x))))))


#widens Tk to account for gaps

expandTw <- function(Tw,res=1) seq(min(Tw),max(Tw),by=res)


#widens Matrix according to Tk

expandMatrixUsingTw <- function(timeMatrix,Tw,res=1) {

		TwE <- expandTw(Tw,res=res)

		timeMatrixNew <- matrix(NA,nrow=nrow(timeMatrix),ncol=length(TwE))

		timeMatrixNew[,is.element(TwE,Tw)] = timeMatrix

		return(timeMatrixNew)

}



#res should always divide maxdelay and the minimum time measurement

compute_LSA_COR <- function(POPiw,maxdelay = 1,res=1,Tw=c(),expand=!is.null(Tw)) {

		POPiw_ = POPiw
```

```
Tw_ = Tw


#expand will expand the matrix given the time.vector to ensure resolution is standard
if(expand) {
        POPiw_=expandMatrixUsingTw(POPiw,Tw,res)
        Tw_ = expandTw(Tw)
}


#number of OTUS
n = dim(POPiw_)[[1]]
#number of timeslots
m = dim(POPiw_)[[2]]


RNiw = normalTransform(POPiw_)


#Store the delay range
delrange = seq(0,maxdelay,by=res)
ndelay = length(delrange)


LSAij_d = array(NA,dim=c(n,n,ndelay))
dimnames(LSAij_d)[[1]] = paste("OTU",1:n,sep="")
dimnames(LSAij_d)[[2]] = paste("OTU",1:n,sep="")
dimnames(LSAij_d)[[3]] = paste("DELAY",delrange, sep="")
```

```
CORNij_d = array(NA,dim=c(n,n,ndelay))

dimnames(CORNij_d)[[1]] = paste("OTU",1:n,sep="")

dimnames(CORNij_d)[[2]] = paste("OTU",1:n,sep="")

dimnames(CORNij_d)[[3]] = paste("DELAY",delrange, sep="")


CORij_d = array(NA,dim=c(n,n,ndelay))

dimnames(CORij_d)[[1]] = paste("OTU",1:n,sep="")

dimnames(CORij_d)[[2]] = paste("OTU",1:n,sep="")

dimnames(CORij_d)[[3]] = paste("DELAY",delrange, sep="")


CORNSij_d = array(NA,dim=c(n,n,ndelay))

dimnames(CORNSij_d)[[1]] = paste("OTU",1:n,sep="")

dimnames(CORNSij_d)[[2]] = paste("OTU",1:n,sep="")

dimnames(CORNSij_d)[[3]] = paste("DELAY",delrange, sep="")


CORSij_d = array(NA,dim=c(n,n,ndelay))

dimnames(CORSij_d)[[1]] = paste("OTU",1:n,sep="")

dimnames(CORSij_d)[[2]] = paste("OTU",1:n,sep="")

dimnames(CORSij_d)[[3]] = paste("DELAY",delrange, sep="")


for(k in 1:ndelay) {
```

```
		#We will only be taking part of the time series range
		#to account for this, we need to remove data points when considering a delay
		#in this case, positive delay means that the second is offset from the beginning,
		#and the first is offset from the end
		d = delrange[k]
		wrange = 1:(m-d/res)
		vrange = (1+d/res):m

		#compute LSA and COR on every pair of rows and store by delay

		LSAij = t(apply(RNiw[,wrange],1,function(x) apply(RNiw[,vrange],1,function(y)
LSA(x*y))))


		CORNij = t(apply(RNiw[,wrange],1,function(x) apply(RNiw[,vrange],1,function(y)
{
			not_na = !is.na(x) & !is.na(y)
			if(sum(not_na)>10) return(cor.test(x[not_na], y[not_na])$estimate)
			else return(NA)
			})))


		CORij = t(apply(POPiw_[,wrange],1,function(x)
apply(POPiw_[,vrange],1,function(y) {
			not_na = !is.na(x) & !is.na(y)
```

```
                    if(sum(not_na)>10) return(cor.test(x[not_na], y[not_na])$estimate)

                    else return(NA)


                    }))))


                CORNSij = t(apply(RNiw[,wrange],1,function(x)
apply(RNiw[,vrange],1,function(y) {

                    not_na = !is.na(x) & !is.na(y)

                    if(sum(not_na)>10) return(cor.test(x[not_na], y[not_na])$p.value)

                    else return(NA)


                    }))))


                CORSij = t(apply(POPiw_[,wrange],1,function(x)
apply(POPiw_[,vrange],1,function(y) {

                    not_na = !is.na(x) & !is.na(y)

                    if(sum(not_na)>10) return(cor.test(x[not_na], y[not_na])$p.value)

                    else return(NA)

                    }))))


                dim(LSAij) = c(n,n)

                dim(CORNij) = c(n,n)

                dim(CORij) = c(n,n)

                dim(CORij) = c(n,n)
```

```
        dim(CORSij) = c(n,n)


        LSAij_d[,,k] = LSAij
        CORNij_d[,,k] = CORNij
        CORij_d[,,k] = CORij
        CORNSij_d[,,k] = CORNSij
        CORSij_d[,,k] = CORSij

}



#LSA_matrix = apply(LSAij_d,c(1,2),function(x) x[which.max(abs(x))])
#LSA_ID_matrix = apply(LSAij_d,c(1,2),function(x) which.max(abs(x)))
#LSA_DELAY_matrix = delrange[LSA_ID_matrix]
#LSA_PON_matrix = sign(LSA_matrix)
#dim(LSA_DELAY_matrix) = c(n,n)


#CORN_matrix = apply(CORNij_d,c(1,2),function(x) x[which.max(abs(x))])
#CORN_ID_matrix = apply(CORNij_d,c(1,2),function(x) which.max(abs(x)))
#CORN_DELAY_matrix = delrange[CORN_ID_matrix]
#dim(CORN_DELAY_matrix) = c(n,n)
#CORN_PON_matrix = sign(CORN_matrix)
```

```
        #COR_matrix = apply(CORij_d,c(1,2),function(x) x[which.max(abs(x))])

        #COR_ID_matrix = apply(CORij_d,c(1,2),function(x) which.max(abs(x)))

        #COR_DELAY_matrix = delrange[COR_ID_matrix]

        #dim(COR_DELAY_matrix) = c(n,n)

        #COR_PON_matrix = sign(COR_matrix)



        return(list(lsa_d = LSAij_d,

                                corn_d = CORNij_d,

                                cor_d = CORij_d,

                                corns_d = CORNSij_d,

                                cors_d = CORSij_d

                                ))

}


ConfidentPairs <- function(LSA_COR_obj,lsasighash_file="LSA98hashtable.csv",origdata =
m000.matrix[OTUS,],cinterv=0.95) {

        lsadata = LSA_COR_obj$lsa_d

        corsdata = LSA_COR_obj$cors_d

        cornsdata = LSA_COR_obj$corns_d

        hashtable = read.csv(lsasighash_file)

        datazeros = apply(origdata,1,function(x) sum(x==0,na.rm=TRUE))


        dim1 = dim(lsadata)
```

```
#significance level is determined by the number of ties in the original data

#we expect these ties to take place at 0, effecting the significance value of the score

#hence a full set of hashed permutations of the data is stored prior to the analysis

#this strongly cuts down on the number of duplicate permutation runs

#one way to change this would be to ensure that all vectors have rank score conserved

#but this may bias certain results


lsasiglevels = matrix(Inf,nrow=dim1,ncol=dim1)

for(i in 1:(dim1[[1]]-1)) for(j in (i+1):dim1[[2]]) lsasiglevels[i,j] =
hashtable[datazeros[i],datazeros[j]]


#compute COR sig for i j as the minimum over all delays

COR_strong = apply(corsdata,c(1,2),function(x) min(x,na.rm=TRUE))


#will contain numeric(0)

COR_delay = apply(corsdata,c(1,2), function(x) which.max(x)-1)


#compute CORN sig for i j as the minimum over all delays

CORN_strong = apply(cornsdata,c(1,2),function(x) min(x,na.rm=TRUE))


#will contain numeric(0)

CORN_delay = apply(cornsdata,c(1,2), function(x) which.max(x)-1)
```

```
#compute LSA for i j as the maximum over all delays

LSA_strong = apply(lsadata,c(1,2),function(x) max(abs(x),na.rm=TRUE))


#need to fix this so that it gives the proper delays, but currently, this should be alright

LSA_delay = apply(lsadata,c(1,2),function(x) which.max(abs(x))-1)



sigLSA = LSA_strong>lsasiglevels

sigCOR = COR_strong<(1-cinterv)

sigCORN = CORN_strong<(1-cinterv)


sigLSA[is.na(sigLSA)]=FALSE

sigCOR[is.na(sigCOR)]=FALSE

sigCORN[is.na(sigCORN)]=FALSE


#don't want the self-score or duplicate scores

selfscore = matrix(FALSE,dim1[[1]],dim1[[1]])

for(i in 1:dim1[[1]]) for(j in i:dim1[[1]]) selfscore[j,i]=TRUE



lsapairs = which(sigLSA & !selfscore,arr.ind=TRUE)

corpairs = which(sigCOR & !selfscore,arr.ind=TRUE)
```

```
	cornpairs = which(sigCORN & !selfscore,arr.ind=TRUE)

	allpairs = which((sigLSA | sigCOR | sigCORN) & !selfscore,arr.ind=TRUE)

	sharedpairs = which(sigLSA & sigCOR & sigCORN & !selfscore,arr.ind=TRUE)


	return(list(lsa_pairs=lsapairs,cor_pairs=corpairs,corn_pairs=cornpairs,all_pairs=allpairs,s
hared_pairs=sharedpairs))


}


pairlistToCSV <- function(pairlist,filename="LSACORpairs.csv") {

	pairs = pairlist$all_pairs

	totpairs = dim(pairs)[[1]]

	csvoutput = matrix(ncol=3,nrow=totpairs)

	csvoutput[,1] = pairs[,1]

	csvoutput[,2] = "pu"

	csvoutput[,3] = pairs[,2]

	colnames(csvoutput) = c("Source","Interaction","Target")

	write.csv(csvoutput,file = filename,quote=FALSE)

	return(filename)

}
```

Appendix A.3: Benjamini-Hochberg correction

```
extractmatrices <- function(LCO, delayrange=c(1), fdr=TRUE) {
        corn_d = LCO$corn_d
        corns_d = LCO$corns_d

        if(fdr) {
                #adjust p-value according to Benjamini-Hochberg to control FDR
                dim(corns_d) = c()
                corns_d = p.adjust(corns_d,method="BH",n=sum(!is.na(corns_d)))
                dim(corns_d) = dim(LCO$corns_d)
        }
        otu1 = dim(corns_d)[[1]]
        otu2 = dim(corns_d)[[2]]
        delay = dim(corns_d)[[3]]
        finalmatrix = matrix(0,nrow=otu1,ncol=otu2)
        delaymatrix = matrix(NA,nrow=otu1,ncol=otu2)
        cormatrix = matrix(0,nrow=otu1,ncol=otu2)
        for(i in 1:otu1) {
                for(j in 1:otu2) {
                        smallestp = which.min(corns_d[i,j,])
                        effectivp = which.min(corns_d[i,j,delayrange])
                        if(length(effectivp)==0) next
```

```
                        if(corns_d[i,j,effectivp]<=0.01) {
                                delaymatrix[i,j] = smallestp-1
                                cormatrix[i,j] = corn_d[i,j,effectivp]
                                }
                    }
            }
            signmatrix = sign(cormatrix)
            finalmatrix = abs(signmatrix)
            return(list(adjm=finalmatrix,delay =delaymatrix, cor = cormatrix,sign = signmatrix))
}

LSAextract2 = extractmatrices(LSA_COR_obj2)
#LSAextract3 = extractmatrices(LSA_COR_obj3)
#LSAextract4 = extractmatrices(LSA_COR_obj4)
#LSAextract5 = extractmatrices(LSA_COR_obj5)
#LSAextract6 = extractmatrices(LSA_COR_obj6)
#LSAextract7 = extractmatrices(LSA_COR_obj7)
LSAextract8 = extractmatrices(LSA_COR_obj8)
#LSAextract9 = extractmatrices(LSA_COR_obj9)
#LSAextract10 = extractmatrices(LSA_COR_obj10)
```

Appendix A.4: Matrix to pairlist

```
adjntimes <- function(x,n) {
        if(n>1) return(x%*%adjntimes(x,n-1))
        else return(x)
        }


#Takes matrix, converts it to csv pairlist


matrixToPairlist <- function(signmat,delaymat) {
        n = dim(signmat)[[1]]
        m = dim(signmat)[[2]]
        z = colnames(signmat)
        res = c()
        for(i in 1:n) {
                for(j in 1:m) {
                        if(i==j) next
                        if(signmat[i,j]==0) next
                        interaction = paste(ifelse(signmat[i,j]>0,"p","n"),
ifelse(delaymat[i,j]==0,"u","dr"),sep="")
                        res = rbind(res,c(z[i], interaction, z[j]))
                        print(c(i,j))
                }
        }
```

```
        return(res)
}

applyOTUnames <- function(LSAextract,otunames) {
        tmp = LSAextract
        rownames(tmp$adjm) = otunames
        colnames(tmp$adjm) = otunames
        rownames(tmp$delay) = otunames
        colnames(tmp$delay) = otunames
        rownames(tmp$cor) = otunames
        colnames(tmp$cor) = otunames
        rownames(tmp$sign) = otunames
        colnames(tmp$sign) = otunames
        return(tmp)
}
```

Appendix A.5: Compare multiple depths

```
#initialize the environment, comment this out after running the initial steps
#Bermuda = new.env(parent = .GlobalEnv)

directory_loc = "~\Desktop\R_files"

sys.source(paste(directory_loc,"/LSA0_rawdataload_simple.R",sep=""),envir=Bermuda)

sys.source(paste(directory_loc,"/LSA1_correlations.R",sep=""), envir=Bermuda)

Bermuda$otunames = rownames(Bermuda$SDS)

#NOTE: the following steps are viable because we are not considering delays
#if we consider delays, the data must be properly ordered before these routines are run

#Only run the next series of lines if you want to run the dataset, otherwise it should be loaded in
a workspace
#VERY INTENSIVE

Bermuda$m000 = which(Bermuda$SDS['depth',] == 0)
Bermuda$m040 = which(Bermuda$SDS['depth',] == 40)
Bermuda$m080 = which(Bermuda$SDS['depth',] == 80)
Bermuda$m100 = which(Bermuda$SDS['depth',] == 100)
Bermuda$m120 = which(Bermuda$SDS['depth',] == 120)
Bermuda$m160 = which(Bermuda$SDS['depth',] == 160)
Bermuda$m200 = which(Bermuda$SDS['depth',] == 200)
Bermuda$m250 = which(Bermuda$SDS['depth',] == 250)
Bermuda$m300 = which(Bermuda$SDS['depth',] == 300)


#In order to combine two depths, we need the elements from each that overlap
#Otherwise they are incomparable
combine_m000_200 <- function() {
        monthread = Bermuda$SDS['month',]
        yearread = Bermuda$SDS['year',]
        absdate = yearread*12 + monthread

        m000dates = absdate[intersect(Bermuda$data,Bermuda$m000)]
        m200dates = absdate[intersect(Bermuda$data,Bermuda$m200)]

        #Which elements now transition
        m000200 =
intersect(Bermuda$data,Bermuda$m000)[is.element(m000dates,m200dates)]
```

```
	m200000 =
intersect(Bermuda$data,Bermuda$m200)[is.element(m200dates,m000dates)]

	SDS2m000 = Bermuda$SDS[,m000200]
	SDS2m200 = Bermuda$SDS[,m200000]

	rownames(SDS2m000) = paste(rownames(SDS2m000),"m000",sep="")
	rownames(SDS2m200) = paste(rownames(SDS2m200),"m200",sep="")
	colnames(SDS2m200) = colnames(SDS2m000)
	return(rbind(SDS2m000,SDS2m200))
}
Bermuda$SDS2 = combine_m000_200()

combine_all_depths <- function() {
	x = names(Bermuda$SDS)
	y = strsplit(x,"\\.|\\X|\\_p7")
	y = as.data.frame(y)

	#m000n = y[2,intersect(Bermuda$data,Bermuda$m000)]
	#m040n = y[2,intersect(Bermuda$data,Bermuda$m040)]
	#m080n = y[2,intersect(Bermuda$data,Bermuda$m080)]
	m120n = y[2,intersect(Bermuda$data,Bermuda$m120)]
	m160n = y[2,intersect(Bermuda$data,Bermuda$m160)]
	#m200n = y[2,intersect(Bermuda$data,Bermuda$m200)]

	#names(m000n) = NULL
	#names(m040n) = NULL
	#names(m080n) = NULL
	names(m120n) = NULL
	names(m160n) = NULL
	#names(m200n) = NULL

	#m000n = as.character(unlist(m000n))
	#m040n = as.character(unlist(m040n))
	#m080n = as.character(unlist(m080n))
	m120n = as.character(unlist(m120n))
	m160n = as.character(unlist(m160n))
	#m200n = as.character(unlist(m200n))

	groupn = intersect(m120n,m160n)

	#Which elements now transition
	#m000g = intersect(Bermuda$data,Bermuda$m000)[is.element(m000n,groupn)]
	#m040g = intersect(Bermuda$data,Bermuda$m040)[is.element(m040n,groupn)]
	#m080g = intersect(Bermuda$data,Bermuda$m080)[is.element(m080n,groupn)]
	m120g = intersect(Bermuda$data,Bermuda$m120)[is.element(m120n,groupn)]
	m160g = intersect(Bermuda$data,Bermuda$m160)[is.element(m160n,groupn)]
```

```
        #m200g = intersect(Bermuda$data,Bermuda$m200)[is.element(m200n,groupn)]

        #SDS2m000 = Bermuda$SDS[,m000g]
        #SDS2m040 = Bermuda$SDS[,m040g]
        #SDS2m080 = Bermuda$SDS[,m080g]
        SDS2m120 = Bermuda$SDS[,m120g]
        SDS2m160 = Bermuda$SDS[,m160g]
        #SDS2m200 = Bermuda$SDS[,m200g]

        #rownames(SDS2m000) = paste(rownames(SDS2m000),"m000",sep="")
        #rownames(SDS2m040) = paste(rownames(SDS2m040),"m040",sep="")
        #rownames(SDS2m080) = paste(rownames(SDS2m080),"m080",sep="")
        rownames(SDS2m120) = paste(rownames(SDS2m120),"m120",sep="")
        rownames(SDS2m160) = paste(rownames(SDS2m160),"m160",sep="")
        #rownames(SDS2m200) = paste(rownames(SDS2m200),"m200",sep="")

        #colnames(SDS2m000) = groupn
        #colnames(SDS2m040) = groupn
        #colnames(SDS2m080) = groupn
        colnames(SDS2m120) = groupn
        colnames(SDS2m160) = groupn
        #colnames(SDS2m200) = groupn

        return(rbind(SDS2m120,SDS2m160))
}

Bermuda$SDS120.160 = combine_all_depths()


#Bermuda$LSA_COR_obj2 =
Bermuda$compute_LSA_COR(Bermuda$SDS[,intersect(Bermuda$data,Bermuda$m000)],maxdel
ay=0)
#Bermuda$LSA_COR_obj8 =
Bermuda$compute_LSA_COR(Bermuda$SDS[,intersect(Bermuda$data,Bermuda$m200)],maxdel
ay=0)
#Bermuda$LSA_COR_obj_comb = Bermuda$compute_LSA_COR(Bermuda$SDS2,maxdelay=0)
#Bermuda$LSA_COR_obj_23 =
Bermuda$compute_LSA_COR(Bermuda$SDS000.040,maxdelay=0)
Bermuda$LSA_COR_obj_56 = Bermuda$compute_LSA_COR(Bermuda$SDS120.160,maxdelay=0)


#Bermuda$SDS_0_200_combine = matrix(NA,nrow =
2*dim(Bermuda$SDS)[[1]],ncol=max(length(Bermuda$m000),length(Bermuda$m200)))

#sys.source(paste(directory_loc,"/LSA2_objectextraction.R",sep=""), envir=Bermuda)

#sys.source(paste(directory_loc,"/LSA3_fixobjects.R",sep=""),envir=Bermuda)
```

```
#Bermuda$LSAextract2 =
Bermuda$applyOTUnames(Bermuda$LSAextract2,Bermuda$otunames)
#Bermuda$LSAextract8 =
Bermuda$applyOTUnames(Bermuda$LSAextract8,Bermuda$otunames)


#sys.source(paste(directory_loc,"/LSA4_simmetric_treeanalysis.R",sep=""), envir = Bermuda)

#sys.source(paste(directory_loc,"/LSA5_SimilarityCores.R",sep=""), envir = Bermuda)
```

Appendix A.6: Cumulative distribution

```
setwd("z:/Kevin/zipped sequences/BATS amplicons/LSA_2/final_matrices/")
final4surf <- read.csv("final4_confirmation_000.csv")
final4surf <- final4surf[,-1]
final4deep <- read.csv("final4_confirmation_200.csv")
final4deep <- final4deep[,-1]
tmp <- rowSums(final4surf)
tmp2 <- density(tmp)

newy2 <- cumsum(tmp2$y[tmp2$x>0])

 newy3 <- 1-(newy2/max(newy2))
par(mar=c(5.1,6.1,4.1,0.1))
 plot(tmp2$x[tmp2$x>0], newy3, log="y", type="l", col="red" ,xlim=c(1,300),yaxt="n",
xlab="Number of connections", ylab="",main="Cumulative Distribution Plot")
axis(2, las=2, at=c(0.000001,0.0001,0.01,1), labels=c("1X10^-4","1X10^-2","1","100"))
mtext("Percent cumulative sum (log scale)",side=2, line=4.5)

par(new=T)
tmp3 <- rowSums(final4deep)
tmp4 <- density(tmp3)

newy4 <- cumsum(tmp4$y[tmp4$x>0])

 newy5 <- 1-(newy4/max(newy4))
plot(tmp4$x[tmp4$x>0], newy5, log="y", type="l", col="blue", axes=F, xlab="", ylab="")

surfcumsum <- cbind(tmp2$x[tmp2$x>0],newy3)
deepcumsum <- cbind(tmp4$x[tmp4$x>0],newy5)
write.csv(surfcumsum,file="surface cumulative sum.csv")
write.csv(deepcumsum,file="deep cumulative sum.csv")

y <- surfcumsum[,2]
x <- surfcumsum[,1]
dy <- deepcumsum[,2]
dx<- deepcumsum[,1]
#fit first degree polynomial equation:
fit  <- lm(y~x)
#second degree
fit2 <- lm(y~poly(x,2,raw=TRUE))
#third degree
fit3 <- lm(y~poly(x,3,raw=TRUE))
#fourth degree
fit4 <- lm(y~poly(x,4,raw=TRUE))
```

```
#fifth degree
fit5 <- lm(y~poly(x,5,raw=TRUE))
#sixth degree
fit6 <- lm(y~poly(x,6,raw=TRUE))
#seventh degree
fit7 <- lm(y~poly(x,7,raw=TRUE))


#generate range of 100 numbers starting from 1 and ending at 300
xx <- seq(1,300, length=100)
plot(x,y,type="l",ylim=c(0.000001,1))
lines(xx, predict(fit4, data.frame(x=xx)), col="red")
lines(xx, predict(fit5, data.frame(x=xx)), col="green")
lines(xx, predict(fit6, data.frame(x=xx)), col="blue")
lines(xx, predict(fit7, data.frame(x=xx)), col="purple")

anova(fit,fit2)
anova(fit2,fit3)
anova(fit3,fit4)
 summary(fit)
summary(fit2)
summary(fit3)
summary(fit4)

dfit  <- lm(dy~dx)
#second degree
dfit2 <- lm(dy~poly(dx,2,raw=TRUE))
#third degree
dfit3 <- lm(dy~poly(dx,3,raw=TRUE))
#fourth degree
dfit4 <- lm(dy~poly(dx,4,raw=TRUE))
dfit5  <- lm(dy~poly(dx,5,raw=TRUE))
dfit6 <- lm(dy~poly(dx,6,raw=TRUE))
dfit7 <- lm(dy~poly(dx,7,raw=TRUE))
dfit8 <- lm(dy~poly(dx,8,raw=TRUE))

#generate range of 100 numbers starting from 0 and ending at 300
dxx <- seq(0,300, length=100)
plot(dx,dy,type="l",ylim=c(0,1))
lines(dxx, predict(dfit4, data.frame(dx=dxx)), col="red")
lines(dxx, predict(dfit5, data.frame(dx=dxx)), col="green")
lines(dxx, predict(dfit6, data.frame(dx=dxx)), col="blue")
lines(dxx, predict(dfit7, data.frame(dx=dxx)), col="purple")

#Can start with other values of p1 and p2 but end up at the same fitted values  but with
different #iterations to convergence
```

```
p1 <- 0.05
p2 <- 0.1

> fite1 <- nls(y ~ (x^(-p1))*(exp(-(p2)*(x))),start=list(p1=p1,p2=p2))
> summary(fite1)

Formula: y ~ (x^(-p1)) * (exp(-(p2) * (x)))

Parameters:
   Estimate Std. Error t value Pr(>|t|)
p1 0.0867127  0.0028833   30.07   <2e-16 ***
p2 0.0262209  0.0003044   86.14   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.01596 on 484 degrees of freedom

Number of iterations to convergence: 8
Achieved convergence tolerance: 9.94e-06

> plot(x,y,type="l",ylim=c(0.000001,1),log="y")
> lines(xx, predict(fite1, data.frame(x=xx)), col="blue")
> fite2 <- nls(dy ~ (dx^(-p1))*(exp(-(p2)*(dx))),start=list(p1=p1,p2=p2))
> plot(dx,dy,type="l",ylim=c(0.000001,1),log="y")
> lines(dxx, predict(fite2, data.frame(dx=dxx)), col="blue")
> summary(fite2)

Formula: dy ~ (dx^(-p1)) * (exp(-(p2) * (dx)))

Parameters:
   Estimate Std. Error t value Pr(>|t|)
p1 0.0627748  0.0026595   23.60   <2e-16 ***
p2 0.0264308  0.0002916   90.63   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.01846 on 484 degrees of freedom

Number of iterations to convergence: 7
Achieved convergence tolerance: 9.428e-06
```

```
> confint(fite1)
Waiting for profiling to be done...
       2.5%      97.5%
p1 0.08062648 0.09276734
p2 0.02556533 0.02689100
> confint(fite2)
Waiting for profiling to be done...
       2.5%      97.5%
p1 0.05749339 0.06800879
p2 0.02582636 0.02704914

> pdf(file="cumulative_dist.pdf")

par(mar=c(5.1,6.1,4.1,0.1))
 plot(x, y, log="y", type="l", col="red" ,xlim=c(1,300),yaxt="n", xlab="Number of connections",
ylab="",main="Cumulative Distribution Plot")
axis(2, las=2, at=c(0.000001,0.0001,0.01,1), labels=c("1X10^-4","1X10^-2","1","100"))
mtext("Percent cumulative sum (log scale)",side=2, line=4.5)
par(new=T)
plot(dx, dy, log="y", type="l", col="black", axes=F, xlab="", ylab="")
lines(xx[0:87], predict(fite1, data.frame(x=xx[0:87])), col="blue")
lines(dxx[0:87], predict(fite2, data.frame(dx=dxx[0:87])), col="green")
legend("topright",c("Suface","200 m","Surface power law fit","200 m power law
fit"),lty=c(1,1,1,1),lwd=2,col=c("red","black","blue","green"))
```

Appendix B: Linear model diagnostics

```
#Written for deep (200 m) samples. For surface samples, replace “deep”
 #with “surf” and grab appropriate files



combSPdeep <- read.csv("combinedSP_200.csv")

combSPdeep <- combSPdeep[,-1]

table(as.matrix(combSPdeep))

combSPdeep <- as.matrix(combSPdeep)

(sum(combSPdeep)-sum(diag(combSPdeep)))/2 #number of 1’s in the upper triangle




#use matrix=surface_popcounts, boolean=combSPdeep for function below.

# combined_SpearmanPearson_surface.csv is sum of adjacency matrix output from LSA for
#Spearman and Pearson correlations consisting of 0’s and 1’s.

#This script uses linear modeling to confirm the relationship between two NTUs that are judged
#related by the LSA analysis and creates a new upper triangle matrix with 1’s confirming a
#relationship and 0’s eliminating a relationship.



s_matrixlm <- function(matrix, boolean,fn="lmconfirmation_200.csv") {

 s_mat <- matrix(NA,nrow=nrow(boolean),ncol=ncol(boolean))

   for(i in 1:(dim(matrix)[1])) {

      for(j in 1:(dim(matrix)[1])) {

         if(i>j) next

#only considers upper triangle
```

```
        if(i == j) next
#ignores the diagonal


        if(boolean[i,j] < 0.5) next
#only looks at putative correlations
        print(paste(i, "and", j))
        xval <- unlist(matrix[i,])
        yval <- unlist(matrix[j,])
        tmp <- lm(xval~yval)
        tmp2 <- anova(tmp)
        tmp3 <- tmp2$"Pr(>F)"[1]
        s_mat[i,j] <- ifelse(tmp3<=0.05,1,0)
      }
   }
   write.csv(s_mat,file=fn)
  }
s_matrixlm(matrix=deep_popcounts, boolean=combSPdeep)
lmconfirmdeep <- read.csv("lmconfirmation_200.csv")
lmconfirmdeep <- lmconfirmdeep [,-1]  #removes 1st column of 1-1374
table(as.matrix(lmconfirmdeep))
lmconfirmdeep[is.na(lmconfirmdeep)] <- 0   #replace NA's with 0's


#Check assumptions of linear modeling that residuals are normally distributed by observing the
#autocorrelation function. Need a modified dataset that has full years of data (trim away partial
#years and impute data for 1-2 month gaps). Transposed the data but that is not strictly
```

```
#necessary. The autocorrelation function is sensitive to the order of the variables in the linear
#model (F test and p-value are insensitive to order) so generate data for both cases. Difference
#the data by season (lag=12) to pre-whiten the data.




deep_popcounts_diff <-
read.csv("BATS_200m_timeseries_cellcounts_transposed_differenced.csv", header=T)



#Use matrix=deep_popcounts_diff,boolean=lmconfirmdeep in next function

#At least 3 of 9 significant values in autocorrelations of residuals if there is structure in the
#residuals so use that as criteria for identifying relationships to discard (1 in matrix to discard, 0
#if OK, NA elsewhere).



s_matrixacf <- function(matrix, boolean,fn="acf_resid_diff2way_200.csv") {

   s_mat <- matrix(NA,nrow=nrow(boolean),ncol=ncol(boolean))

   for(i in 1:(dim(matrix)[2])) {

      for(j in 1:(dim(matrix)[2])) {

         if(i>j) next

         if(i == j) next

         if(boolean[i,j] <0.5) next

         xval <- unlist(matrix[,i])

         yval <- unlist(matrix[,j])

         tmp <- lm(xval~yval)

        tmp3 <- lm(yval~xval)

         tmp2 <- acf(tmp$resid, plot=F)

           tmp4 <- acf(tmp3$resid, plot=F)
```

```
        s_mat[i,j] <- ifelse((sum((abs(tmp2$acf[2:10]))>=0.20))>=3,1,(
ifelse((sum((abs(tmp4$acf[2:10]))>=0.20))>=3,1,0)))

      }

   }

   write.csv(s_mat,file=fn)

  }

s_matrixacf(matrix=deep_popcounts_diff,boolean=lmconfirmdeep)

acfd2waydeep <- read.csv("acf_resid_diff2way_200.csv")

acfd2waydeep <- acfd2waydeep [,-1]

table(as.matrix(acfd2waydeep))

acfd2waydeep [is.na(acfd2waydeep)] <- 0

lmconfirmdeep_2 <- lmconfirmdeep - acfd2waydeep  #subtract correlated NTUs with non-
normal linear #model residuals



table(as.matrix(lmconfirmdeep_2))



#Initial data screening suggested that most data did not need to be differenced seasonally first.
#Simulations show that structure in residuals for one NTU only does not affect type I error.
#However, structure in two NTU's can greatly affect type I error. Check autocorrelation
#functions for undifferenced modified data (as before). Look at row and column sums for each
#NTU. Most NTUs have only a few acf's with structure. Most of the acf's with structure are
#found in the top 5% of NTUs (sorted by the number of problem acfs). If both NTUs are in the
#upper 5%, their type I error rate is probably elevated so they should be eliminated. Others are
#more likely to be OK.



deep_popcounts_trans <- read.csv("BATS_200m_timeseries_cellcounts_transposed.csv")
```

```
# use matrix=deep_popcounts_trans, boolean=lmconfirmdeep_2 in next function


s_matrixacf2 <- function(matrix, boolean,fn="acf_resid_norm_200.csv") {
   s_mat <- matrix(NA,nrow=nrow(boolean),ncol=ncol(boolean))
   for(i in 1:(dim(matrix)[2])) {
      for(j in 1:(dim(matrix)[2])) {
        if(i>j) next
          if(i == j) next
          if(boolean[i,j] <0.5) next
          xval <- unlist(matrix[,i])
          yval <- unlist(matrix[,j])
          tmp <- lm(xval~yval)
            tmp3 <- lm(yval~xval)
          tmp2 <- acf(tmp$resid, plot=F)
            tmp4 <- acf(tmp3$resid, plot=F)


         s_mat[i,j] <- ifelse((sum((abs(tmp2$acf[2:10]))>=0.20))>=3,1,(
ifelse((sum((abs(tmp4$acf[2:10]))>=0.20))>=3,1,0)))
      }
   }
   write.csv(s_mat,file=fn)
  }
s_matrixacf2(matrix=deep_popcounts_trans, boolean=lmconfirmdeep_2)
```

```
acfnormdeep <- read.csv("acf_resid_norm_200.csv")

acfnormdeep <- acfnormdeep [,-1]

table(as.matrix(acfnormdeep))

acfnormdeep [is.na(acfnormdeep)] <- 0

rs <- rowSums(acfnormdeep)

cs <- colSums(acfnormdeep)

totaldeep <- rs + cs

totaldeep <- as.matrix(totaldeep)

total2 <- sum(totaldeep>0)

total3 <- total2*0.05

sum(totaldeep>X)  #find X by trial and error such that the sum is = total3. Fill in X in next script.

s_matrixdualacf <- function(matrix, boolean,fn="acf_resid_dual_200.csv") {
   s_mat <- matrix(NA,nrow=nrow(boolean),ncol=ncol(boolean))
   for(i in 1:(dim(boolean)[2])) {
      for(j in 1:(dim(boolean)[2])) {
        if(i>j) next
          if(i == j) next
          if(boolean[i,j] <0.5) next



         s_mat[i,j] <- ifelse(matrix[i,]>=X,(ifelse(matrix[j,]>=X,1,0)),0)
      }
   }
```

```
   write.csv(s_mat,file=fn)

  }

s_matrixdualacf(matrix=totaldeep, boolean=acfnormdeep)

acfdualdeep <- read.csv("acf_resid_dual_200.csv")

acfdualdeep <- acfdualdeep [,-1]

table(as.matrix(acfdualdeep))

acfdualdeep [is.na(acfdualdeep)] <- 0



lmconfirmdeep_3 <- lmconfirmdeep_2 - acfdualdeep
#subtract correlated NTUs with likely dual NTUs
#with structure in residuals

table(as.matrix(lmconfirmdeep_3))

write.csv(lmconfirmdeep_3, file="final4_confirmation_200.csv")
```

Appendix C: ARIMA structure simulation

```
#both white noise
one_sim <- function(){
  x <- arima.sim(list(), n = 100)
  y <- 1 + arima.sim(list(), n = 100)
  fit <- arima(y, order = c(0, 0, 0), xreg = x)
  abs( fit$coef["x"] /
     sqrt(diag(fit$var.coef)["x"]) ) > 1.96
}

reject <- replicate(1000, one_sim())
table(reject)
reject
FALSE  TRUE
  954    46

#one white noise, one structured
one_sim <- function(){
  x <- arima.sim(list(order = c(1,0,0), ar = 0.7), n = 100)
  y <- 1 + arima.sim(list(), n = 100)
  fit <- arima(y, order = c(0, 0, 0), xreg = x)

  abs( fit$coef["x"] /
     sqrt(diag(fit$var.coef)["x"]) ) > 1.96
}

reject <- replicate(1000, one_sim())
table(reject)
reject
FALSE  TRUE
  950    50

#both structured
one_sim <- function(){
  x <- arima.sim(list(order = c(1,0,0), ar = 0.7), n = 100)
  y <- 1 + arima.sim(list(order = c(1,0,0), ar = 0.9), n = 100)
  fit <- arima(y, order = c(0, 0, 0), xreg = x)

  abs( fit$coef["x"] /
     sqrt(diag(fit$var.coef)["x"]) ) > 1.96
}

reject <- replicate(1000, one_sim())
table(reject)
```

reject
FALSE TRUE
663 337

Appendix D: Weighted NTU correlation similarity

```
x <- read.delim("~/Desktop/BermudaCodeOrg/v4.output.txt")


#v4.output.txt is a file of tree distances for each NTU compared to the 5
#final nodes to either side of that NTU. Final nodes are nodes with
#sequences assigned by Phyloassigner so each distance is between two
#nodes containing some number of sequences. This is essentially a moving
#window for phylogenetic distance. The adjacency matrix comes from the
#linear modeling filtering script. Otunames (or Ntunames) is a list of
#NTU identifiers for each node. Run this script by providing the
#adjacency matrix to num_sim_metric_tree.

#extractOBJ <- c()

#extractOBJ$adjm <- (adjacency matrix)

#num_sim_metric_tree(extractOBJ)



#remove last element

x = x[-13411,]



x_comb = x



#y = x[order(x[,1]),]

#n = length(unique(x[,1]))

n = length(otunames)

n_comb = 2*n
```

```
distadj <- function(xdata = x, n = length(otunames))

{

eta = exp(-10)

ymat = matrix(nrow=n,ncol=n)

rownames(ymat) = otunames

colnames(ymat) = otunames


for(i in 1:length(x[,1])) {

        ymat[as.character(x[i,1]),as.character(x[i,2])] = x[i,3]

        ymat[as.character(x[i,1]),as.character(x[i,1])] = 0

}

ymat = exp(-5*ymat)

ymat[is.na(ymat)] = 0

return(ymat)

}


ymatrix = distadj()


#need a second version for combined


distadj_comb <- function(xdata = x_comb, n = 2*length(otunames))

{

eta = exp(-10)
```

```
ymat = matrix(nrow=n,ncol=n)
rownames(ymat) = c(paste(otunames,"_000",sep=""), paste(otunames,"_200",sep=""))
colnames(ymat) = c(paste(otunames,"_000",sep=""), paste(otunames,"_200",sep=""))

for(i in 1:length(x[,1])) {
        xa = paste(as.character(x[i,1]), "_000", sep="")
        xb = paste(as.character(x[i,2]), "_000", sep="")
        x2a = paste(as.character(x[i,1]), "_200", sep="")
        x2b = paste(as.character(x[i,2]), "_200", sep="")
        ymat[xa,xb] = x[i,3]
        ymat[xb,xa] = x[i,3]
        ymat[x2a,x2b] = x[i,3]
        ymat[x2b,x2a] = x[i,3]
        #ymat[xa,x2b] = x[i,3]
        #ymat[x2b,xa] = x[i,3]
        #ymat[x2a,xb] = x[i,3]
        #ymat[xb,x2a] = x[i,3]
        ymat[xa,xa] = 0
        #ymat[xa,x2a] = 0
        #ymat[x2a,xa] = 0
        ymat[x2a,x2a] = 0
}
ymat = exp(-5*ymat)
```

```
ymat[is.na(ymat)] = 0

return(ymat)

}


ymatrix_comb = distadj_comb()




num_sim_metric_tree <- function(extractOBJ,group = c(), ymat = ymatrix,nodenames =
otunames) {

        tmp_adjm = extractOBJ$adjm

        tmp_nodenames = nodenames

        tmp_ymatrix = ymat


        if(!is.null(group)) {

        #filter to just the group we're interested in

        tmp_adjm = tmp_adjm[group,group]

        tmp_ymatrix = tmp_ymatrix[group,group]

        tmp_nodenames = tmp_nodenames[group]

        }


        #coerce nodes to numbers

        adj_nodenames = as.character(tmp_nodenames)

        adj_nodenames = as.numeric(adj_nodenames)
```

```
#select those nodes which have numbered values

tmpset = which(!is.na(adj_nodenames))

group_nodenames = tmp_nodenames[tmpset]


#smaller adjacency matrix

tmp_adjm = tmp_adjm[tmpset,tmpset]

tmp_ymatrix = tmp_ymatrix[tmpset,tmpset]


rownames(tmp_adjm) = group_nodenames

colnames(tmp_adjm) = group_nodenames


tmp_mavem = matrix(nrow = length(tmpset),ncol = length(tmpset))

rownames(tmp_mavem) = group_nodenames

colnames(tmp_mavem) = group_nodenames


tmp_mavem = tmp_adjm%*%(tmp_ymatrix)


sim_matrix = matrix(nrow = length(tmpset),ncol = length(tmpset))


for(i in 1:length(tmpset)) {

        for(j in 1:length(tmpset)) {

                totpoints = sum((tmp_mavem[i,] + tmp_mavem[j,])**2)
```

```
                    diffpoints = sum((tmp_mavem[i,] - tmp_mavem[j,])**2)

                    ipoints = sum(tmp_mavem[i,])

                    jpoints = sum(tmp_mavem[j,])

                    if(ipoints == 0 | jpoints == 0) next

                    sim_matrix[i,j] = diffpoints/totpoints

                }
        }

        y = t(1/(sim_matrix+1))

        rownames(y) = group_nodenames

        colnames(y) = group_nodenames

        return(y)

        }

sim_metric_csv <- function(LSAextract,fname,groupA) {

        tmp = num_sim_metric_tree(LSAextract,group = groupA )
```

```
	print(paste("Writing ", fname))

	write.csv(tmp, file=fname)

}


sim_metric_csv_mangroup <- function(LSAextract,fname,nodenames=otunames) {

	tmp = num_sim_metric_tree(LSAextract)

	print(paste("Writing ", fname))


	adj_nodenames = as.character(nodenames)

	adj_nodenames = as.numeric(nodenames)


	#select those nodes which have numbered values

	tmpsetind = which(!is.na(adj_nodenames))

	tmpset = adj_nodenames[tmpsetind]


	write.csv(tmp[intersect(group1surf,tmpset),intersect(group1surf,tmpset)],
file=paste(fname,"_1.csv",sep=""))

	write.csv(tmp[intersect(group2surf,tmpset),intersect(group2surf,tmpset)],
file=paste(fname,"_2.csv",sep=""))

	write.csv(tmp[intersect(group3surf,tmpset),intersect(group3surf,tmpset)],
file=paste(fname,"_3.csv",sep=""))

}
```

Appendix E: Tarjan's algorithm

```
tmp_LSAextract = c()

tmp_LSAextract$adjm =deep



#To start, use directed.core.list.n script with
#tmp_LSAextract=LSA_extract, a list of nodenames, and the number of
#iterations.

# The nodenames have to be a character vector and the adjacency matrix
#has to be as.matrix



adjtocores <- function(adjm, nodenames=rownames(adjm)) {

        tmp_adjm <- adjm




        sharedconn <- tmp_adjm %*% t(tmp_adjm)

        connsbyntu <- rowSums(tmp_adjm)


        n = length(connsbyntu)
```

```
        totconn <- rep(connsbyntu,each=n) + rep(connsbyntu,n)
        dim(totconn) = c(n,n)

        #exclude
        perc_shared <- 2*sharedconn/totconn
        close_core <- apply(perc_shared-diag(n),1,function(x)
                                                ifelse(is.nan(sum(x)),NA,which.max(x)))

        return(list(percsh = perc_shared, cc = close_core))
}

cores <- function(cclist) {
        #ccmat holds the cycle matrix
        #it is an nxn matrix where n is the number of vertices

        n = length(cclist)
ccmat = matrix(NA,nrow = n,ncol = n)
x = cclist

for (i in 1:n){
```

```
for (j in 1:n){
  ccmat[i,j] <- ifelse(i+j<= n,x[i+j],x[i+j-n])
}


    }
        cyclend = c()

        for(i in 1:n) {
                cyclend = c(cyclend,min(which(ccmat[,i]==i)))
        }

        #Define each group by the element with the smallest id
        ccgroup <- matrix(NA,nrow=1,ncol=n)
        colnames(ccgroup) = colnames(ccmat)

        tmpind <- which(cyclend<Inf)

        for(i in tmpind) {
                ccgroup[i] = min(ccmat[1:cyclend[i],i])
        }

        groupids = unique(ccgroup)
        tmpind <- which(!(cyclend<Inf))
```

```
        for(i in tmpind) {
                ccgroup[i] = groupids[is.element(groupids,ccmat[,i])]
        }



        #cycmat is the full matrix, grouped by node in columns, and by step in rows
        #cycind is the length of the cycle beginning from each node (Inf if it does not cycle)
        #group has all of the nodes assigned a group identifier, (the smallest member id)

        return(list(cycmat = ccmat, cycind = cyclend, group = ccgroup, index = names(cclist)))

}

#takes a directed core list object (x) and produces the subgraph of the
#desired core at the desired iteration

core.adj.node <- function(x, node, niter =2) {

        corenum = x$ci[niter+1,as.character(node)]
        group_mem = which(is.element(x$ci[niter+1,],corenum))

        starts = x$ci[niter,]
```

```
group_starts = starts[group_mem]
group_u_starts = unique(group_starts)

u_starts = unique(starts)

nstarts = length(unique(starts))

adj_mat = matrix(0,nrow=nstarts,ncol=nstarts)
rownames(adj_mat) = u_starts
colnames(adj_mat) = u_starts

for(i in 1:length(u_starts)) {
        #only need one element with that start, all the others will be the same
        tmp_x = which(starts==(u_starts[i]))

        #find the index of the destination
        ind_destination = x$dgra[niter+1,tmp_x[1]]

        if(niter == 1) {
                destination = which(ind_destination == u_starts)
                adj_mat[i,destination] = 1
                }
```

```
                if(niter != 1) {
                        destination = u_starts[ind_destination]
                        adj_mat[i,as.character(destination)] = 1
                        }

        }

        adj_mat = adj_mat[as.character(group_u_starts),as.character(group_u_starts)]
        return(adj_mat)

}

core.graph.node <- function(x, node, niter=2) {
        tmp_adjm = core.adj.node(x,node,niter)
        coregraph = graph.adjacency(tmp_adjm,mode = "directed")
        if(niter == 1) V(coregraph)$label =
(as.character(otunames))[as.numeric(colnames(tmp_adjm))]
        if(niter != 1) V(coregraph)$label = colnames(tmp_adjm)
        V(coregraph)$size = 2
        E(coregraph)$arrow.size = 0.5
        E(coregraph)$color = 2
        return(coregraph)
}
```

```
core.adj <- function(x, corenum, niter =2) {
        group_mem = which(is.element(x$ci[niter+1,],corenum))

        starts = x$ci[niter,]

        group_starts = starts[group_mem]
        group_u_starts = unique(group_starts)

        u_starts = unique(starts)

        nstarts = length(unique(starts))

        adj_mat = matrix(0,nrow=nstarts,ncol=nstarts)
        rownames(adj_mat) = u_starts
        colnames(adj_mat) = u_starts

        for(i in 1:length(u_starts)) {
                #only need one element with that start, all the others will be the same
                tmp_x = which(starts==(u_starts[i]))

                #find the index of the destination
                ind_destination = x$dgra[niter+1,tmp_x[1]]
```

```
                if(niter == 1) {
                        destination = which(ind_destination == u_starts)
                        adj_mat[i,destination] = 1
                        }
                if(niter != 1) {
                        destination = u_starts[ind_destination]
                        adj_mat[i,as.character(destination)] = 1
                        }

        }

        adj_mat = adj_mat[as.character(group_u_starts),as.character(group_u_starts)]
        return(adj_mat)

}

core.graph <- function(x, corenum, niter=2) {
        tmp_adjm = core.adj(x,corenum,niter)
        coregraph = graph.adjacency(tmp_adjm,mode = "directed")
        if(niter == 1) V(coregraph)$label =
(as.character(otunames))[as.numeric(colnames(tmp_adjm))]
        if(niter != 1) V(coregraph)$label = colnames(tmp_adjm)
        V(coregraph)$size = 2
        E(coregraph)$arrow.size = 0.5
```

```
        E(coregraph)$color = 2
        return(coregraph)
}

directed.ls.graph <- function(cclist, nodenames = otunames) {
        tmpgraph = graph.edgelist(cbind((1:length(cclist))[!is.na(cclist)],cclist[!is.na(cclist)]))
        tmpgraph$labels = nodenames[(1:length(cclist))[!is.na(cclist)]]
        return(tmpgraph)
}

directed.core.list.n <- function(LSAextract, nodenames = otunames, niter = 1) {
    tmp_adjm <- LSAextract$adjm
    tmp_nodenames <- 1:length(nodenames)

    group_nodenames = tmp_nodenames
    rownames(tmp_adjm) = group_nodenames
    colnames(tmp_adjm) = group_nodenames

    #create the first core list
    z1 = adjtocores(tmp_adjm)
    corelist = cores(z1$cc)
```

```
#coreindex tracks the group list

coreindex = corelist$group


#number and value of cores

unique_cores = corelist$group

unique_cores = unique_cores[!is.na(unique_cores)]

unique_cores = unique(unique_cores)

ncores = length(unique_cores)


#replace core number with an absolute number, so we can know a core as "core 4 in
iteration 2"


coreindex = t(as.matrix((1:ncores)[match(coreindex,unique_cores)]))

nntu = dim(coreindex)[2]

        colnames(coreindex) = 1:nntu


#coreindex = rbind(as.numeric(colnames(coreindex)),coreindex)

#coreindex tracks group identity in each iteration


#core_dgra tracks the core connection directed graph

core_dgra = corelist$cycmat[1,]

#core_dgra = rbind(as.numeric(colnames(core_dgra)),core_dgra)
```

```
#statistics for similarity
avgconnsim = c()
maxconnsim = c()
minconnsim = c()

for(j in 1:niter) {

    #get a list of the number of cores
    unique_cores = corelist$group
    unique_cores = unique_cores[!is.na(unique_cores)]
    unique_cores = unique(unique_cores)

    #number of cores and shared connections between cores
    ncores = length(unique_cores)
    coreconnections = matrix(nrow = ncores, ncol= nntu)
    rownames(coreconnections) = unique_cores

                                #for each group we add up all the ntus in a core
#by column and normalize by the number of elements in a core
    for(i in 1:ncores) {
    k = sum(coreindex[j,] == i,na.rm=TRUE)
    coreconnections[i,] = colSums(tmp_adjm[which(coreindex[j,] == i),,drop=FALSE])/k
    }
```

```
#gives shared connections between 2 cores for each ntu as the smaller of the 2 values
coreconnA = coreconnections %*% t(sign(coreconnections))
coreconnB = sign(coreconnections) %*% t(coreconnections)
sharedcoreconns <-  ifelse(coreconnA<coreconnB,coreconnA,coreconnB)
connsbycore <- rowSums(coreconnections)

totconns <- rep(connsbycore,each=ncores) + rep(connsbycore,ncores)
dim(totconns) = c(ncores,ncores)

#same statistic for percentage shared
perc_shared <- 2*sharedcoreconns/totconns


close_core <- apply(perc_shared-diag(ncores),1,function(x)
                        ifelse(is.nan(sum(x)),NA,which.max(x)))



#now create new cores
tmp_cores_i <- cores(close_core)

                        #store the new unique cores, and number
                        tmp_unique_cores = tmp_cores_i$group
```

```
            tmp_unique_cores = tmp_unique_cores[!is.na(tmp_unique_cores)]
            tmp_unique_cores = unique(tmp_unique_cores)
                                    tmp_ncores = length(tmp_unique_cores)

                                    tmp_coreindex = t(as.matrix(tmp_cores_i$group[coreindex[j,]]))
            coreindex = rbind(coreindex,tmp_coreindex)

            tmp_core_dgra = t(as.matrix((tmp_cores_i$cycmat[1,])[coreindex[j,]]))
            core_dgra = rbind(core_dgra,tmp_core_dgra)
            corelist = tmp_cores_i
            }

      colnames(coreindex) = nodenames
      colnames(core_dgra) = nodenames
      return(list(ci = coreindex, dgra = core_dgra))

}


directed.graph.from.corelist <- function(x,niter=3) {
            dests = unique(x$ci[niter,])[x$dgra[niter+1,]]
            starts = x$ci[niter,]
```

```
	#unique_dests = unique(ndests)

	#ndests = length(unique_dests)


	u_starts = unique(starts)


	nstarts = length(unique(starts))


	adj_mat = matrix(0,nrow=nstarts,ncol=nstarts)

	rownames(adj_mat) = u_starts

	colnames(adj_mat) = u_starts


	for(i in 1:nstarts) {

		#only need one element with that start, all the others will be the same

		x = which(starts==(u_starts[i]))

		destination = dests[x[1]]

		adj_mat[i,as.character(destination)] = 1

	}


	return(list(adj = adj_mat))

}


directed.graph.from.corelist.1 <-function(x) {

	n = length(x$ci[2,])
```

```
        adj_mat = matrix(0,nrow=n,ncol=n)
        rownames(adj_mat) = colnames(x$ci)
        colnames(adj_mat) = colnames(x$ci)
        for(i in 1:n) {
                adj_mat[i,as.character(otunames[x$ci[2,i]])] = 1
        }
        return(list(adj = adj_mat))
}

directed.core.list.1 <- function(LSAextract, nodenames = otunames) {
        tmp_adjm <- LSAextract$adjm
        tmp_nodenames <- nodenames

        adj_nodenames = as.character(tmp_nodenames)
        adj_nodenames = as.numeric(adj_nodenames)

        #select those nodes which have numbered values
        tmpset = which(!is.na(adj_nodenames))
        group_nodenames = tmp_nodenames[tmpset]

        #smaller adjacency matrix
        tmp_adjm = tmp_adjm[tmpset,tmpset]
```

```
rownames(tmp_adjm) = group_nodenames

colnames(tmp_adjm) = group_nodenames



sharedconn <- tmp_adjm %*% t(tmp_adjm)

connsbyntu <- rowSums(tmp_adjm)

m = dim(tmp_adjm)[2]


#create the first core list

z1 = adjtocores(LSAextract)

corelist = cores(z1$cc)

corelist2 = unique(corelist$group)

corelist2 = corelist2[!is.na(corelist2)]


n = length(corelist2)

coreconnections = matrix(nrow = n, ncol= m)


for(i in 1:n) {

        k = sum(corelist$group == corelist2[i],na.rm=TRUE)

        coreconnections[i,] = colSums(tmp_adjm[which(corelist$group ==
corelist2[i]),])/k

}

coreconnA = coreconnections %*% t(sign(coreconnections))
```

```
        coreconnB = sign(coreconnections) %*% t(coreconnections)
        sharedcoreconns <-  ifelse(coreconnA<coreconnB,coreconnA,coreconnB)
        connsbycore <- rowSums(coreconnections)

        n = length(connsbycore)
        totconns <- rep(connsbycore,each=n) + rep(connsbycore,n)
        dim(totconns) = c(n,n)

        perc_shared <- 2*sharedcoreconns/totconns
        close_core <- apply(perc_shared-diag(n),1,function(x)
                                                ifelse(is.nan(sum(x)),NA,which.max(x)))

        return(list(percsh = perc_shared, cc = close_core))

}

#Tarjan's Algorithm
#vlistobj = list(index = <vec>, lowlink = <vec>, k = <sca>, cclist = <vec>, Sk = <sca>, S = <list>)
#strongconnect <- function(v,i) {
#       tmpv = v
```

```
#       if(is.na(tmpv$index[i])) {
#               tmpv$index[i] = tmpv$k
#               tmpv$lowlink[i] = tmpv$k
#               tmpv$k = tmpv$k + 1
#               tmpv$S[[tmpv$Sk]] = c(tmpv$S[[tmpv$Sk]],
#               }
#       j = tmpv$cclist[i]
#       if(is.na(tmpv$index[j])) {
#               tmpv = strongconnect(tmpv,j)
#               tmpv$lowlink[i] = min(tmpv$lowlink[i], tmpv$lowlink[j],na.rm=TRUE)
#       } else {
#               tmpv$lowlink[i] = min(tmpv$lowlink[i], tmpv$index[j],na.rm=TRUE)
#       }


#       if(tmpv$lowlink = v.index) {}


#}


# Takes the list of closest connections, produces core groups
# takes element i and recurs until it reaches i again
# our system has only 1 outgoing link per vertex, so if something is part of
# a core, it will repeat.
# if it reaches i, it will return the vector of core locations
```

Appendix F: Mean pairwise phylogenetic distance algorithm

```
dm <- read.csv("tree_dist_matrix_2.csv",header=T)

ntu <- as.character(dm[,1])

dm <- dm[,-1]

dm <- as.matrix(dm)

ntu <- as.numeric(ntu)

rownames(dm) <- ntu

colnames(dm) <- ntu

sc <- read.csv("surf_cores.csv", header=T)

sc <- sc[-1,]

sc <- sc[,-1]

sc <- sc[1:1344,]

sc <- cbind(ntu,sc)

# You will need a distance matrix from a phylogenetic tree file for each NTU to NTU pairwise
#comparison. You will also need output from the LSA5 script, a list of all NTUs and the cores to
#which they belong from each iteration. If the distance matrix has a first column of NTU names,
#that can be extracted and used as row and column names. The core list should have as many
#rows as the distance matrix and should have the same NTU names.

corestats <- function(core_matrix,dist_matrix,ntunames,niter){
```

```
diffcore(core_matrix,niter)

simcore(list=corels,ntunames)

coredist(list=simc,matrix=dist_matrix)

}

#Run corestats with the distance matrix, core list, NTU names (ntunames=ntu), and number of
#iterations (niter= ). Returns the number of simulated cores with a smaller mean pairwise
#phylogenetic distance for each tested core and the mean pairwise phylogenetic distance for
#the tested core.




diffcore <- function(matrix,niter){

 #input matrix is the core list from LSA5_SimilarityCores script (sc)

corels <<- list()

 for (g in 1:max(matrix[,niter+1],na.rm=T)){

   core <-  matrix[which(matrix[,niter+1]==g),1]

   corels[[g]] <<- core

   # generates lists of NTUs for each core in a given iteration



 }

 return(corels)

}




simcore <- function(list,ntunames){
```

```
#input for this function is output from diffcore, corels

  simc <<- list()

  for (h in 1:length(list)){

  sim <- replicate(1000,sample(ntunames,length(list[[h]]),replace = F))

#makes a 1000 column matrix with appropriate number of randomly chosen NTU's in each
column

simc[[h]] <<- cbind(sim,list[[h]])

#generates 1000 random simulations for each core and returns it in separate lists in simc

}

return(simc)

}



coredist <- function(list,matrix){

  stats <- list()

mns <- list()

  for (f in 1:length(list)){

    distm <- vector(length=ncol(list[[f]]))

    dm2 <- matrix(NA,nrow=nrow(list[[f]]),ncol=nrow(list[[f]]))

    for (k in 1:ncol(list[[f]])){

      for (i in 1:nrow(list[[f]])){

        for (j in 1:nrow(list[[f]])){

          if (i<=j) next

          dm2[i,j] <- dm[as.character(list[[f]][i,k]),as.character(list[[f]] [j,k])]
```

```
      }
     }
     distm[k] <- mean(dm2,na.rm=T)



   }
   hist(distm[1:1000])
   abline(v=distm[1001])
   stats[[f]] <- (sum(distm[]<distm[1001]))
mns[[f]] <- distm[1001]
 }
 return(list(stats,mns))
}
```

Appendix G: Surface vs. 200 m cluster comparison

```
#Initial data processing. Input files are from LSA5_SimilarityCores_5 script
ntu <- read.csv("otunames.csv")
ntu <- as.numeric(ntu)
ntu <- rbind(0,ntu)
sc <- read.csv("surf_cores.csv", header=T)
sc <- sc[-1,]
sc <- sc[,-1]
sc <- sc[1:1344,]
sc <- cbind(ntu,sc)
dc <- read.csv("deep_cores.csv", header=T)
dc <- dc[-1,]
dc <- dc[,-1]
dc <- dc[1:1344,]
dc <- cbind(ntu,dc)


# Main function
allcomparison <- function(corelist1,corelist2,ntu,niter){
# create two matrices for comparison; write to files to validate
fn1=paste("surface_",niter, "iter_matrix.csv",sep="")
surfmat <- core_sim_matrix(corelist1,niter=niter)
surfmat[is.na(surfmat)] <- 0
rownames(surfmat) <- ntu[,1]
colnames(surfmat) <- ntu[,1]
write.csv(surfmat,file=fn1)

fn2=paste("deep_",niter, "iter_matrix.csv",sep="")
deepmat <- core_sim_matrix(corelist2,niter=niter)
deepmat[is.na(deepmat)] <- 0
rownames(deepmat) <- ntu[,1]
colnames(deepmat) <- ntu[,1]
write.csv(deepmat,file =fn2)

adjmd <- read.csv(fn2,header=T)
adjms <- read.csv(fn1,header=T)
adjmd <- as.matrix(adjmd[,-1])  #(Get rid of index column)
adjms <- as.matrix(adjms[,-1])
rownames(adjmd) <- ntu[,1]  #(insert rownames to both)
rownames(adjms) <- ntu[,1]
```

```
# call adjtosims to get initial similarities
simbyrow1 <- adjtosims(adjm1=adjms,adjm2=adjmd)
# call simntu to get a list of ntus from one of the matrices
simntus1 <- simntu(matrix=corelist2,niter=niter)
# call diffcore to get a list of ntus in each core
corels1 <- diffcore(matrix=dc,niter=niter)
# call randtest to do actual comparison. This function has three
#subfunctions.
randtest1 <- randtest(simntus=simntus1,corels=corels1,matrix=dc,ntu=ntu,adjm1=adjms)
# create vectors with true value in first position and random test values in
# next 1000 positions
allmeans <- c((mean(simbyrow1,na.rm=T)),randtest1$randmeans)
allsds <- c((sd(simbyrow1,na.rm=T)),randtest1$randsds)
out2 <- data.frame(allmeans,allsds)
return(out2)
}



core_sim_matrix  <- function(corelist,niter){
mat <- matrix(ncol=nrow(corelist),nrow=nrow(corelist))
corelist[is.na(corelist)] <- 0
for(i in 1:nrow(corelist)){
for(j in 1:nrow(corelist)){
if(i>=j) next
if(corelist[i,niter+1]==0) next
if(corelist[i,niter+1]==corelist[j,niter+1])
mat[i,j] <- 1
else
mat[i,j] <- 0
}
}
return(mat)
}


#run this first for real data
adjtosims <- function(adjm1, adjm2) {

  tmp1 <- adjm1 + t(adjm1)
```

```
  tmp2 <- adjm2 + t(adjm2)

  n <- nrow(adjm1)


  simbyrow <- c()

  for(i in 1:n){
    shared <- crossprod(tmp1[i,2:n],tmp2[i,2:n])
    total <- sum(tmp1[i,2:n]+tmp2[i,2:n])
    simbyrow[i] <- (shared*2)/total
  }
  return(simbyrow)
}

simntu <- function(matrix,niter){
simntus <- c()
matrix[is.na(matrix)] <- 0
for(i in 1:nrow(matrix)){
if(matrix[i,niter+1]>0)
simntus[i] <- matrix[i,1]
}
return(simntus)
}

diffcore <- function(matrix,niter){

  #input matrix is the core list from LSA5_SimilarityCores_5 script

corels <- list()

  for (g in 1:max(matrix[,niter+1],na.rm=T)){

    core <-  matrix[which(matrix[,niter+1]==g),1]

    corels[[g]] <- core
# generates lists of NTUs for each core in a given iteration

  }

  return(corels)

}
```

```
randmat <- function(simntus, corels){
simntus[is.na(simntus)] <- 0
simntus <- simntus[simntus !=0]
simntusr <- sample(simntus,replace=F)
newcores <- list()
for(j in 1:length(corels)){
if(j>1)
newcores[[j]] <- simntusr [(1+(length(corels [[j-1]]))):length(corels [[j]])]
else
newcores[[j]] <- simntusr [1:length(corels [[j]])]

}

return(newcores)
}

newmat  <-  function(matrix, ntu,newcores){
#need to get ntu names from the first matrix for the new matrix
newmat  <-  matrix(nrow=nrow(matrix), ncol=nrow(matrix))
rownames(newmat) <- ntu[,1]
colnames(newmat) <- ntu[,1]
for(f in 1:length(newcores)){
for(h in 1:length(newcores[[f]])){
for(k in 1:length(newcores[[f]])){
if(h==k) next
newmat[ as.character(newcores[[f]][h]),as.character(newcores[[f]][k])] <- 1
}
}
}
return(newmat)
}

adjtorandmat <- function(newmat,adjm1){

newmat[is.na(newmat)] <- 0
tmp1 <- adjm1 + t(adjm1)


  n <- nrow(adjm1)
```

```
 simbyrowr <- c()

 for(i in 1:n){
  shared <- crossprod(tmp1[i,2:n],newmat[i,2:n])
  total <- sum(tmp1[i,2:n]+newmat[i,2:n])
  simbyrowr[i] <- (shared*2)/total
 }
 return(simbyrowr)
}


randtest <- function(simntus,corels,matrix,ntu,adjm1){
randmeans <- c()
randsds <- c()
# run 1000 tests
for(v in 1:1000){
# call randmat to randomize the ntus in the core list and put back into a new
#core list
randmat1 <- randmat(simntus=simntus1,corels=corels1)
# call newmat to generate new upper triangle adjacency matrices by inserting
# 1's for pairwise ntus in each core
newmat1 <- newmat(matrix=dc,ntu=ntu,newcores=randmat1)
#call adjtorandmat to calculate similarities between unaltered matrix and new
# random matrix
simbyrowr1 <- adjtorandmat(newmat=newmat1,adjm1=adjms)

# save means and sd's for each random matrix in vectors to reduce memory
randmeans[v] <- mean(simbyrowr1,na.rm=T)
randsds[v] <- sd(simbyrowr1,na.rm=T)
out <- data.frame(randmeans,randsds)
}
return(out)
}

# Commands to run script. May take over an hour to run.
allcomparison1 <- allcomparison(corelist1=sc,corelist2=dc,ntu=ntu,niter=2)
write.csv(allcomparison1,file="surfacevsdeepsim.csv")

# Graph results. xlim may need to be changed for other matrix comparisons
hist(data[2:1001,1],xlim=c(0,0.08),main="Mean cluster connection similarity",xlab="Similarity
value")
```

```
abline(v=data[1,1])

hist(data[2:1001,2],xlim=c(0,0.12),main="Standard deviation values for cluster connection
similarities",xlab="Standard Deviation")
abline(v=data[1,2])
```