A Large-Scale Statistical Survey of Environmental Metagenomes

Naneh Apkarian[*] Michelle Creek[†] Eric Guan[‡]
Mayra Hernandez[§] Kate Isaacs[¶] Chris Peterson[*]
Todd Regh[‖]

August 14, 2009

Abstract

A metagenome is a sampling of genetic sequences from the entire microbial community within an environment. Examining the functional diversity represented by these sequences gives insight into the biological processes within the environment as well as the biological differences between environments. Most research has focused on single specific environments and the few comparative analyses have been based only on small fractions of the metagenomic data which is currently available. Dinsdale et al used canonical discriminant analysis to investigate 45 microbial metagenomes in Functional metagenomic profiling of nine biomes, Nature, 452, (2008), 629-632. We expand on this work by applying a wider variety of multivariate statistic and machine learning techniques to study over 200 metagenomes from various human, marine, terrestrial, and extreme environments. Our findings demonstrate the ability to differentiate or predict environments with only a subset of key functional hierarchies.

[*]Pomona College
[†]Chapman University
[‡]Torrey Pines HS
[§]San Diego State University
[¶]San José State University
[‖]Southern Oregon University

Contents

1 Introduction

Vast communities of microbes occupy every environment, processing and producing substances at the molecular level which shape the local geochemistry. Study of the functional families of microbial proteins reveal these processes. Recent studies have shown the viability and reliability of functional metagenomic analysis in describing and identifying environmental microbial and viral samples [9, 32, 27, 28, 33]. Metagenomic analysis has been shown to far surpass 16s marker methods in explaining variance among and within environmental samples [9].

Technological advances over the last decade have allowed for quick genomic sequencing of microbes as well as larger organisms. However, it is estimated that less than 1% of microbes can be individually sequenced due to difficulties of culturing them in a controlled laboratory. The field of Metagenomics instead sequences samples of a community of microbes or virii directly from the environment.

Most of the focus in Metagenomics has been on single environments such as coral atolls[10, 32], cow intestines[4], ocean water[2], and human excrement[16, 29](Add more categories and citations). Recently, Dinsdale et al.[9] demonstrated that analysis of functional diversity in metagenomes can differentiate between multiple environments. We offer an overview of statistical techniques that aid in the analysis of multiple metagenomic environments.

2 Data

A protein or DNA chain is recorded as a sequence of letters representing the component amino acids or nucleotides. The matching of two such sequences against each other, letter by letter with possible gaps, is known as an alignment. The goal of a sequence alignment is maximizing the quality of the matching. Good matches imply homology between two sequences. The quality of the matching is often measured by a scoring function. A simple scoring function might add one every time the letters in the same positions match and subtracting every time they do not. More complex scoring schemes include varying weights depending on the particular letter combinations and linear gap penalties.

A common alignment task would be searching for significant matches against an entire database of known sequences. Exact algorithms for sequence alignment such as Smith-Waterman[24] are too computationally intensive to use on this scale. Heuristic algorithms such as Basic Local Alignment Search Tool (BLAST)[1] are used instead. BLAST searches the database for likely matches based on index sequence snippets. It then grows an alignment for each candidate. The statistical significance of a match is determined based on the length of the alignment, the size of the existing database, and the scoring function used.

Because metagenome sequence sets include tens of thousands to hunreds of thousands of sequences, search and alignment is still relatively slow even with existing heuristic algorithms. As a result, updating metagenomic sequence alignments after database changes is costly. K-mer alignment performs the matching based on only index sequence snippets, forgoing the calculation of match significance. The full set of available metagenomes can then be re-aligned after database changes. This removes differences between metagenomes due to differences of their underlying alignment databases.

Publicly available metagenomic sequence sets were acquired from the SEED database[19]. Sequences were aligned and categorized into functional hierarchies using K-mer alignment[11] with a word length of 10 and a word cutoff of 2.

Once alignment is completed, annotation is added, providing details about the sequence's function and structure. Anotation was originally done by hand, but there are now methods which perform this task computationally. With the wealth of genome data being generated and made available, the ability to automatically annotate genomic data has become increasingly im-

portant.

Metagenome sequences uploaded to the SEED database are automatically annotated with functional information. Curators of the SEED have grouped existing similar sequences into functional families. When a sequence in a metagenomic sequence set matches the existing database, it is automatically categorized with the same functional information as the database sequences it matched. Even though metagenomes include sequences from previously unsequenced and possibly unknown organisms, the homology implied by the alignment enables us to develop a functional profile of the environment. The functional information is what we use throughout this manuscript.

We have removed the functional hierarchies *clustering-based subsystems* and *experimental subsystems* from our data, leaving 27 first level functional hierarchies. Hierarchy counts were normalized by the total number of nonexcluded hits to yield percent composition by function.

3 Principal Component Analysis

Principal component analysis (PCA) is a statistical technique that uses principal components as a dimensional reduction tool to preserve variance. Principal components (PC) are orthonormal linear combinations of the variables of the data that maximize variance over a specific set of variables. The number of principal components of a dataset is equal to the number of variables, with the first principal component having maximal variance and successive principal components absorbing as much of the remaining variance as possible.

Given an $n \times q$ (with $n \geq q$) data matrix $\mathbf{Y}$ with $q \times q$ covariance matrix $\mathbf{S}$, the $q \times 1$ principal components $\vec{v}_1, \vec{v}_2, \ldots \vec{v}_q$ are vectors such that

$$\max_{\substack{\langle \vec{v}_i, \vec{v}_j \rangle = 0, 1 \leq j < i \\ \|\vec{v}_i\| = 1}} \|\vec{v}_i^T \mathbf{S} \vec{v}_i\| \tag{1}$$

Using the spectral decomposition theorem, which states that symmetric matrix $\mathbf{S}$ can be decomposed into $\mathbf{GBG}^T$, with orthogonal matrix $\mathbf{G}$ and diagonal matrix $\mathbf{B}$, it follows that $\vec{v}_1, \vec{v}_2, \ldots \vec{v}_q$ equal the eigenvectors of $\mathbf{S}$. The variance of the ith principal component is equal to its corresponding eigenvalue λ_i, with $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_q$.[15]

All of the variance in the q variables will be transformed into the q principal components, but an effective PCA will concentrate a large proportion of the variance in the first few principal components (as much as 95% of the variance in 2 components is not rare). This result minimizes the number of dimensions needed to sufficiently account for the variance in the entire dataset.

For our metagenomic research, PCA was used to identify clustering within the data, to test how various sample groups tended to cluster or spread, to identify outliers, and to note relations between certain variables with samples or other variables.

3.1 Visualization

A PCA plot (or biplot), as seen in Figure 1, graphs each metagenome as a point in a 2-dimensional graph transformed over the first two principal components. This can also be done in 3 dimensions over 3 components, or

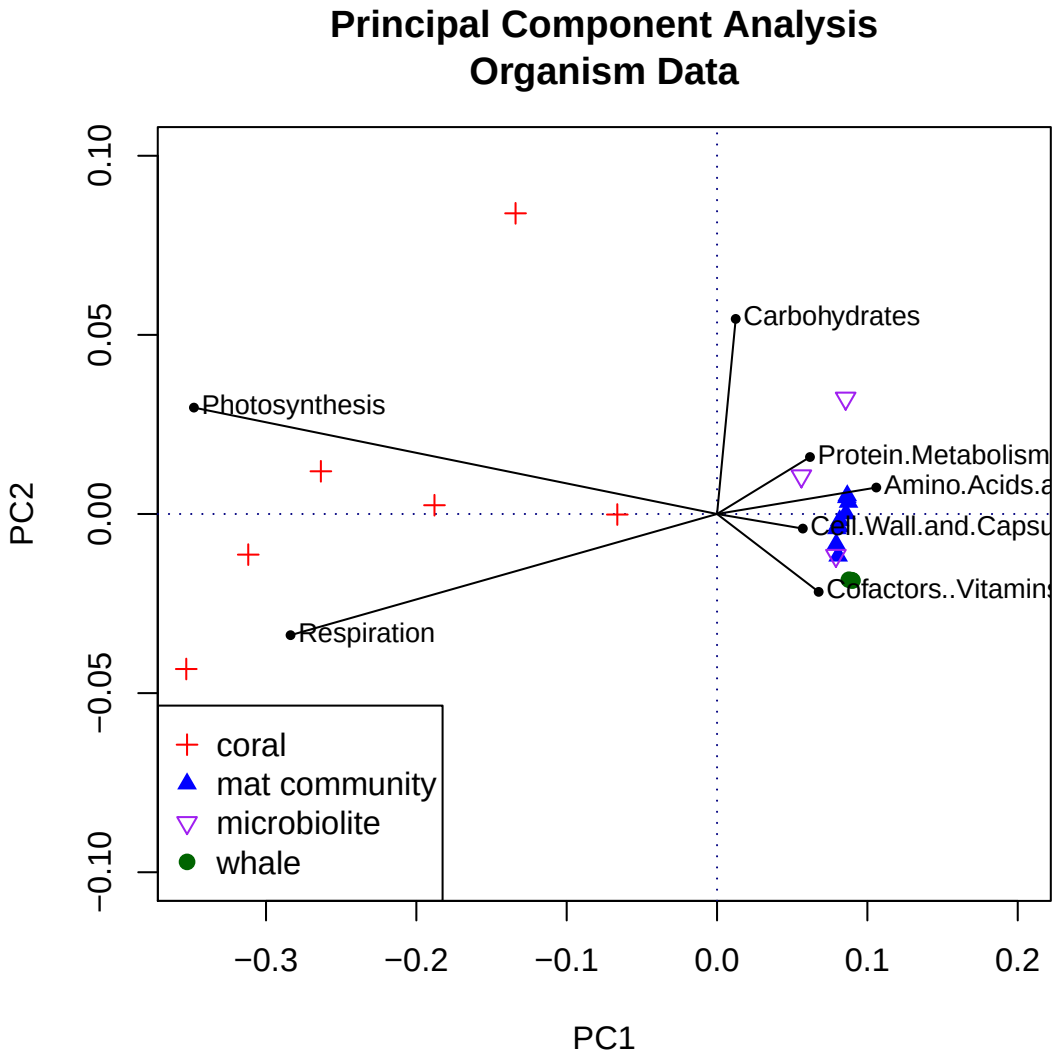


Figure 1: This figure shows a PCA plot of the organism-associated data. The variables are represented as vectors, and each sample is plotted as a point. Distinct clusters emerge naturally in this PCA, as seen in the corals widely dispersed on the left with high photosynthesis and respiration. The mat community and whale fall samples form their own tight clusters on the right, and the microbiolites are loosely grouped in that area. This graph displays over 90% of the variance from the original 27-variable dataset.

in higher dimensions with multiple pairwise plots. Data points are usually colored based on their metagenomic classification. Vectors representing variables are drawn, with coordinates based on their respective weights in the principal components. Due to the nature of the analysis, the amount of variance explained by the first 2 dimensions will depend on the effectiveness of the orthonormal combination. A good PCA graph will have the majority of the variance contained in the first two principal components.

3.2 Samples

The PCA process is unsupervised, meaning that no classifications of samples are given to the analysis. This makes PCA less vulnerable to misclassified data or forming false divisions in classified data. Sample selection is crucial in PCA. If the number of samples is too large and diverse, the graph is cluttered with data. If a particular class only has 2 or 3 samples (a common occurrence in metagenomic data), the PCA will not be able to make definitive conclusions about those classes. Ideally, the samples are fairly similar and are not trivially seperated, but with distinct groups that the PCA can differentiate between.

Due to the nature of PCA, outliers or outlier classes (most notably coral) skew the maximal-variance principal components and thus the graph as a whole. Conversely, PCA is adept at identifying outliers, partly due to being unsupervised. These outlier data points are more easily found with an unscaled PCA (see Section 3.3), and removing them usually improves the PCA (see Figure 2). Identifying and removing outlier data points has played a crucial role in our metagenomic research, as throughout the process several samples were found to be misclassified or contaminated.

3.3 Scaling and Centering

Scaling[1] is an important technique when performing PCA. Normalizing the variance for each variable prevents a variable with a high count (e.g. Carbohydrates) from being considered more important than a variable with a low count (e.g. Dormancy and Sporulation), even though their variances are proportionally the same. Conversely, a variable whose count is high may

[1]Note that this refers to scaling variables to normalize their variance, not the general scaling applied to all metagenome samples prior to statistical analysis that normalized the sum of each sample's variables to 1.

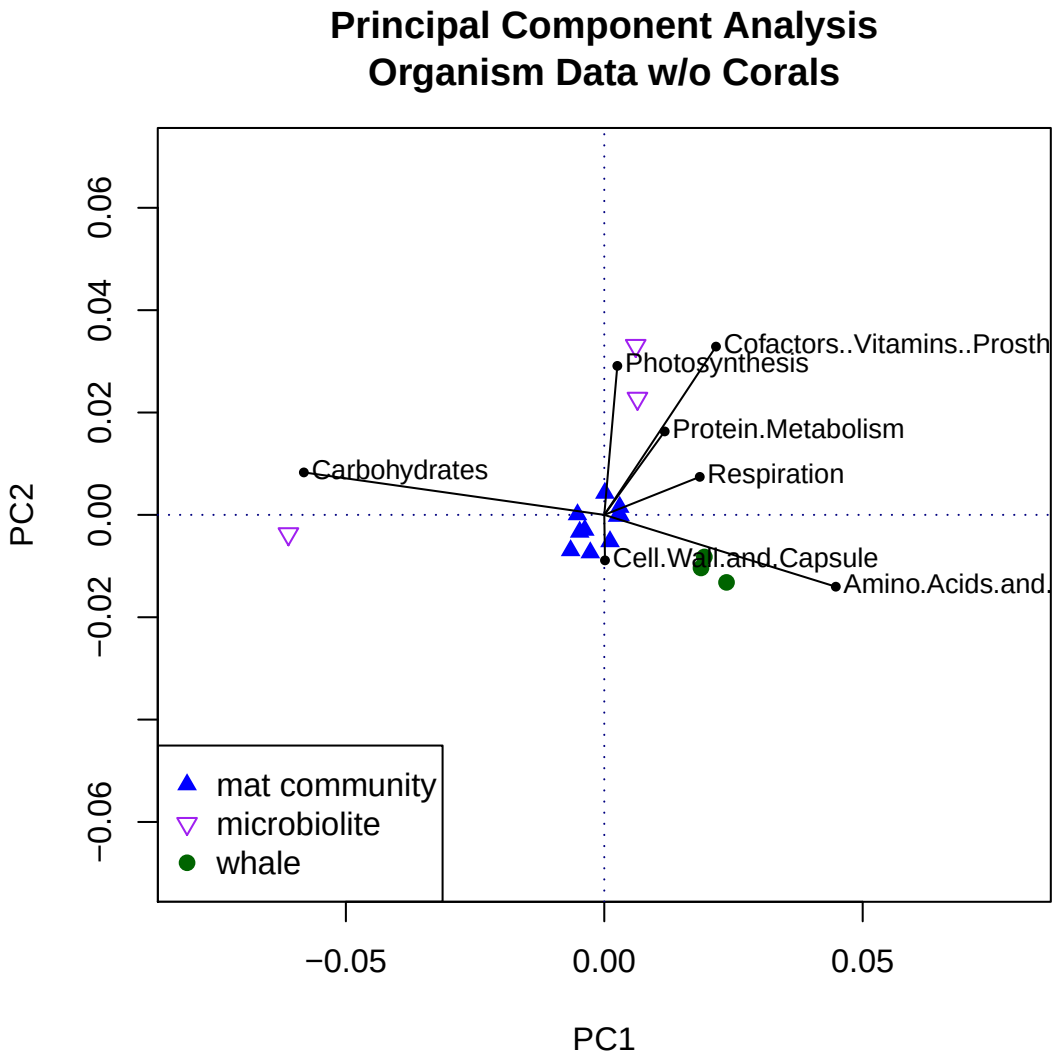
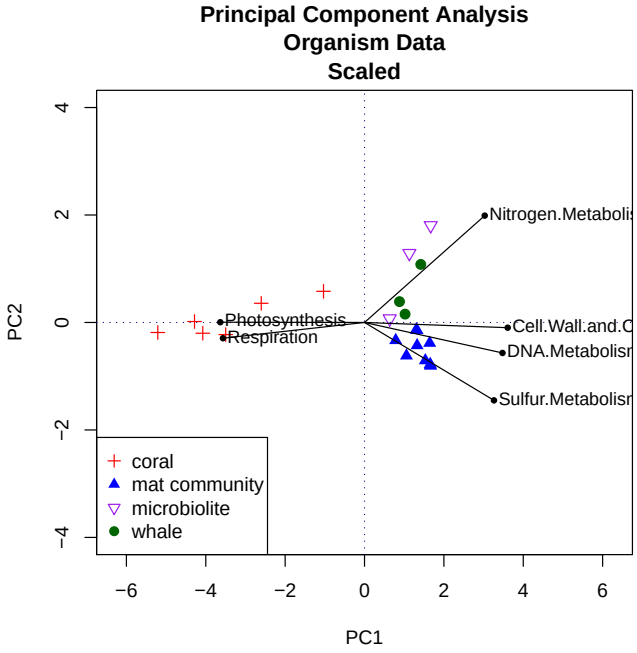
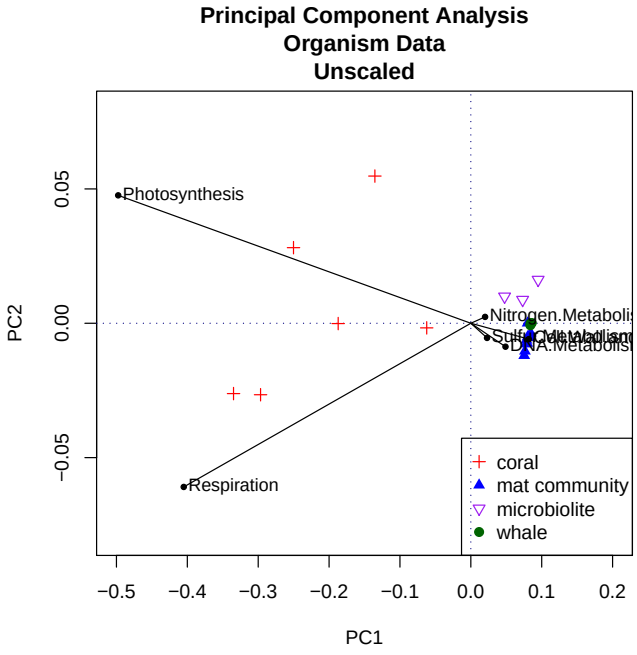


Figure 2: This plot uses the same data and variables as that in Figure 1, except that all of the coral samples have been removed. The new graph presents a much closer look at the non-coral samples than the previous graph that were previously obscured by the corals shifting the original graph.



(a) PCA Scaled



(b) PCA Unscaled

Figure 3: These two PCAs are of the same data over the same variables, but (a) has all of its variables scaled to equal variance and (b) does not. The samples spread well in the scaled graph, but this spread is only maintained by the coral samples in the unscaled graph. The scaled graph treats each variable equally, while the unscaled graph reveals Photosynthesis and Respiration to be the dominating variables. The scaled graph explains 79% of the variance, while the unscaled explains 84%.

actually be much more significant than a variable whose count is low, and scaling may not demonstrate this. Scaling also decreases the concentration of the variance in the first couple principal components, which hinders the ability of 2 or 3 dimensions to explain the entire dataset. This hindering effect is amplified when there are more variables.

The following plots of organism metagenomes demonstrate this scaling issue. Figure 3(a) is the scaled PCA, with an even spread between all of the groupings. Figure 3(b) is the unscaled PCA, with a much larger spread within the corals, and the rest of the samples clumped together. In this case, the scaled plot does not demonstrate the extremely high variance internal to the coral group, but the unscaled plot does not display any sort of spread between the mat community, whale fall, and microbiolite groups.

Centering the metagenomic data such that each variable's mean is 0 is necessary, because otherwise every value is positive and this uncentered PCA only occupies 2 quadrants of the graph. Note that the statistics program R centers the PCA by default.

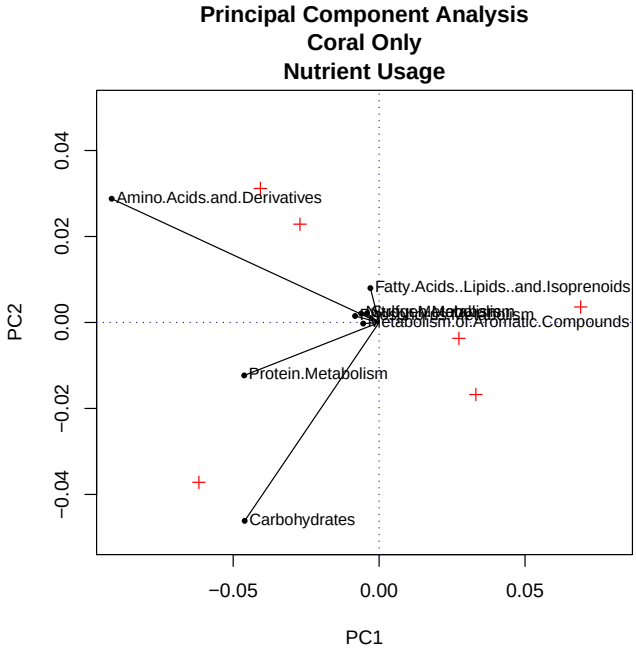
3.4 Variables

Variable choice is very important for PCA. Our metagenomic data has 27 different functional classifications as variables, yet very few samples due to the novel nature of metagenomic research. The number of variables selected must be less than the number of samples; otherwise PCA is not possible due to dimensional limitations. Generally 5 to 10 well-picked variables were sufficient capture the majority of the variance in a dataset.

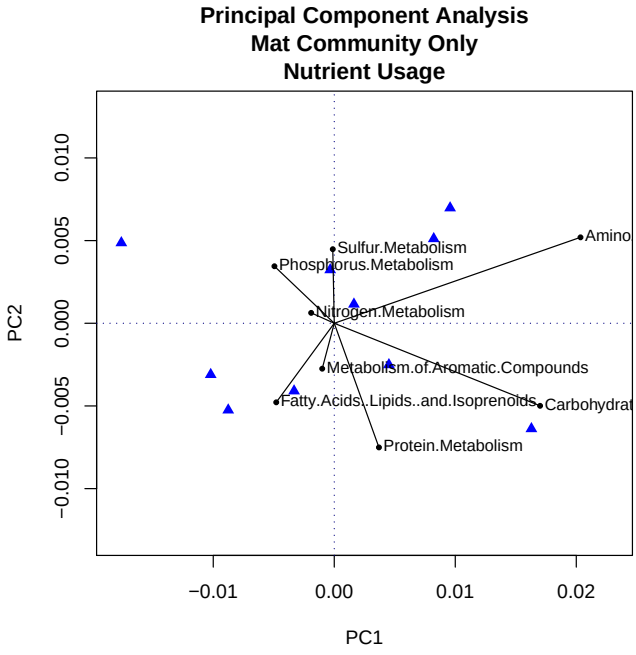
Selecting a handful of variables with highest variance is the best way to retain a high proportion of the variance in the dimensional reduction, although other methods are viable as well. Variables with the highest variance between class means aids in separating groups in the PCA. Using variables from supervised Random Forest variable importance plot (see Section 7.1) may be more effective in differentiating between classes. Selecting biologically significant groups of variables to see groupings based on specific traits may also prove useful (see Figure 4).

In addition to finding clusters, PCA with specific variables over one well-established group of samples yields properties of the variables in those samples (see Figure 4). The magnitude of the vectors is proportional to the variance within that particular variable, and the angle between a pair of vectors is linked to the correlation between those two variables. If two vectors

are pointing in the same direction, they are positively correlated; if they are pointing in opposite directions, they are negatively correlated; if they are orthogonal, they are uncorrelated. This technique decreases in effectiveness as the number of variables and thus the number of dimensions projected into a single plane increases.



(a) Coral Nutrient Usage



(b) Mat Community Nutrient Usage

Figure 4: These PCAs over variables associated with nutrient usage are of the coral (a) and mat community (b) metagenomes. In the coral samples, Amino Acids, Carbohydrates, and Protein Metabolism are by far the dominant nutrient-associated variables, as seen in the vector magnitudes. The mat community samples are much more diverse in terms of nutrient usage, and their vectors are much more even in length and distributed in all directions.

4 K-Means Clustering

K-means clustering is an unsupervised method which aims to classify observations into K groups, for a choice of K. This approach seeks to select K means and partition observations into clusters in order to minimize the sum of squared distances from each observation to the mean of its assigned group. The function that is minimized is called the objective function:

$$obj(\mu_1, \ldots, \mu_K) = \sum_{i=1}^{n} min_{\mu_1,\ldots,\mu_K} \left\| x^{(i)} - \mu_k \right\|^2 \tag{2}$$

where $x^{(i)}$ is an observation and $\mu_1, \ldots, \mu_K$ are the means. The result is K clusters where each observation belongs to the cluster with the closest mean.

The K-means algorithm starts by randomly selecting $\mu_1, \ldots, \mu_K$ and placing all observations into groups based on minimizing the objective function using Euclidean distance. As shown in Figure 5, group means are then calculated using the observations in each cluster and replace the previous means, $\mu_1, \ldots, \mu_K$. The algorithm is repeated until additional runs no longer modify the group means or the partitioning of observations. Each initialization of the algorithm with multiple runs will converge to a minimum objective function, though not necessarily a global minimum. It is therefore a good idea that the algorithm is initialized numerous times before selecting the means and clusters which produce the lowest sum of squares.

4.1 Methods for K Selection

4.1.1 Sum of Squares Plot

Different choices of K will alter the output of the K-means algorithm, so selecting the optimum K is important. A plot of the sum of squares versus values of K is one technique that can be used for selection. This method is useful for minimizing both the sum of squares within each cluster and the value of K. The sum of squares decreases as K increases since the larger the number of groups, the fewer observations are in each group, and thus, the smaller the within-group sums. For this reason, selecting the K with the smallest sum of squares overfits the data. It is desirable to select a K from a plot that has an 'elbow', where there is a steep drop at a specific K from which the plot more gradually approaches zero as shown Figure 8(a). With a distinct elbow, a best K can be selected and useful classification may result.

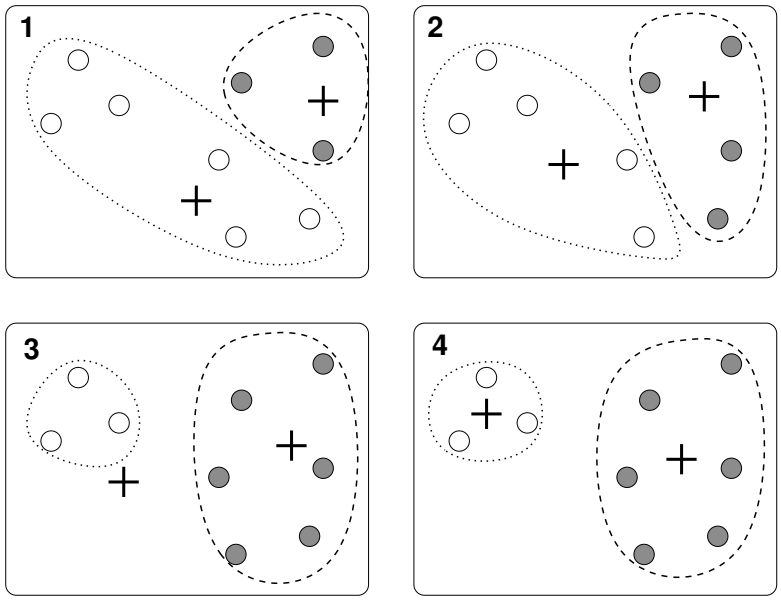
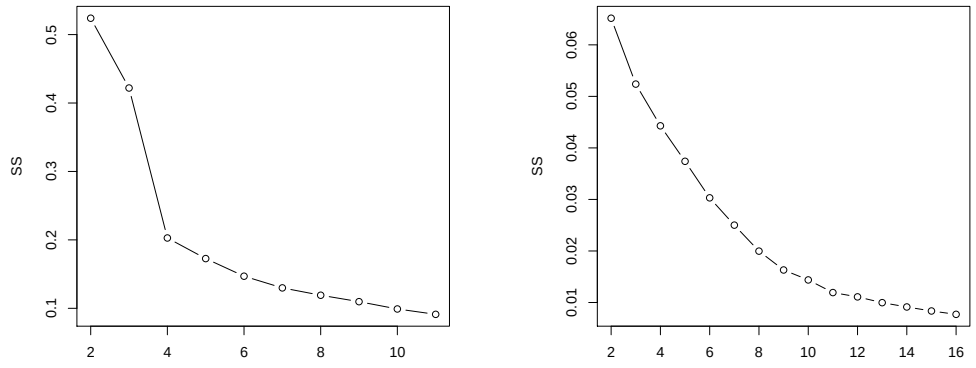


Figure 5: K-Means Algorithm. Observations are denoted by circles, means by crosses, and clusters by dashed lines. Step one: For $K = 2$, means are randomly selected and observations are partitioned into clusters. Step two, three, and four: Group means replace previous means and observations are reclassified. Algorithm results in clusters and means which minimize the objective function.



(a) Desirable Sum of Squares Plot. Compares the sum of squared distances versus values of K. An elbow aids in the minimization of both the sum of squares and K. $K = 4$ has a sharp elbow which indicates a best K of 3, using the one-up rule, or 4.

(b) Undesirable Sum of Squares Plot. A rounded plot does not suggest a good K to input into the K-means algorithm. In this case, alternative methods of K selection should be used.

Figure 6: K-Means Sum of Squares Plots

Depending on the data, the sum of squares plot does not always have a clear elbow (see Figure 6(b)). For metagenomic data, it was not unusual for the plot to appear rounded or have multiple elbows. In this case, alternative approaches of selecting K were tested.

4.1.2 Silhouettes

With our metagenomic data, a more effective method of choosing K used silhouettes which test how well an observation fits into the cluster it has been partitioned into rather than the next nearest cluster. Silhouettes give a good indication of how spread out groups are from each other. Let $a(i) = \left\|x^{(i)} - \mu_k\right\|^2$ and $b(i) = \left\|x^{(i)} - \mu_l\right\|^2$, where $x^{(i)}$ is an observation in group k and l is the group with the next closest mean [15]. A silhouette is then defined as

$$\text{silhouette}(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}. \tag{3}$$

Ideally, each observation is much closer to the mean of its group than to the mean of any other group. In this case, the silhouette would be close

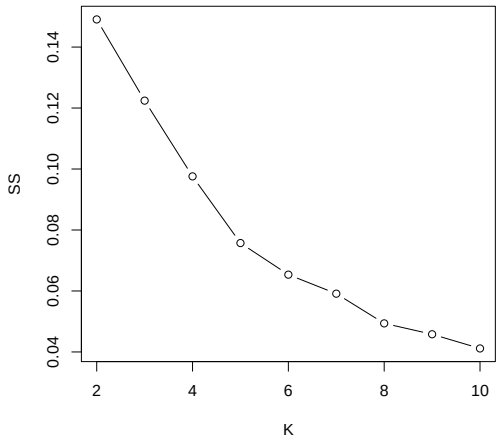


Figure 7: K-Means Sum of Squares Plot. Organism Associated and Coastal Marine metagenomic data. Slight elbow at $K = 5$ indicates an optimal K of 4 or 5.

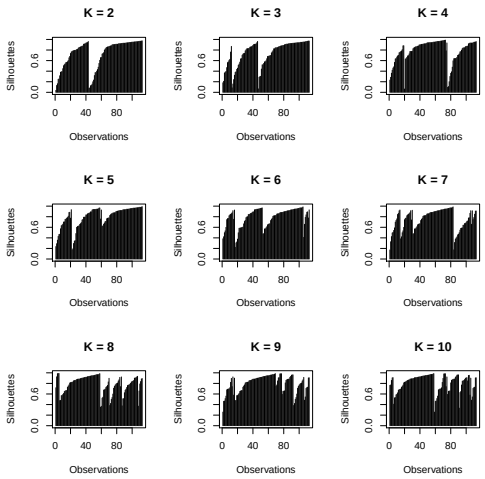
to 1. Figure 8(a) shows silhouette graphs for multiple values of K. Similar to the sum of squares plot, one must be careful about choosing a minimal K which has a large average silhouette width (see Figure 8(b)). Though silhouette graphs frequently suggest a clear K to select, for some data there is no indication of an optimal K. When neither the sum of squares plot nor the silhouette graphs provide a best K, the algorithm then depends on a semi-arbitrary choice of K.

4.2 Applications to Metagenomic Data

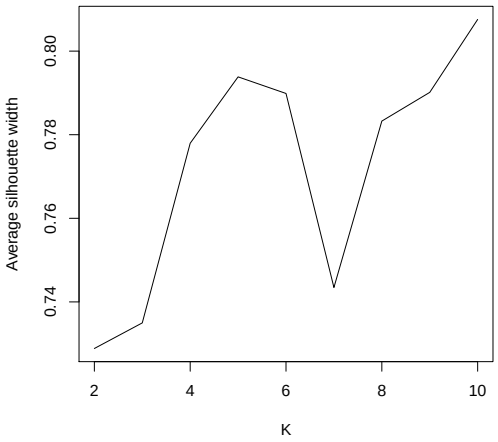
K-means is a mathematically supported method of selecting classifications without regard to predetermined groupings. It therefore can be used as a tool to check assumed classifications or to develop more meaningful ones. With some metagenomic data, K-means identified biologically supported groups which could then be used with visualization methods such as PCA (see Figure 9) and supervised techniques that require initial classifications. Here is an example with Organism Associated and Coastal Marine metagenomes[2].

The sum of squares plot given by the data, shown in Figure 7, has a slight elbow at $K = 5$. As a general rule, the K one up from the elbow is chosen to

[2]The Organism Associated data included the following number of samples: 2 fish gut, 2 fish slime, 9 mat communities, 3 whale fall, 31 human gut, 4 cow gut, 2 chicken gut, and 2 mouse gut. There were 57 samples of Coastal Marine metagenomes.



(a) Silhouette Graphs. Organism Associated and Coastal Marine metagenomic data. Silhouettes calculate how well each observation fits into the cluster it is partitioned into versus the group with the next closest mean. (In these graphs, silhouette width corresponds to height.)



(b) Average Silhouette Width Plot. Organism Associated and Coastal Marine metagenomic data. Minimizing K while maximizing the average silhouette width suggests 5 or 6 are optimal values of K. Demonstrates the convergence of average silhouette width to one as K increases.

Figure 8: K-Means Silhouettes

be the number of clusters. If it happens that the K at the elbow results in better classification it can be used instead. According to our sum of squares plot $K = 4$ and $K = 5$ are potential best K's. To verify this, silhouettes were used. Silhouette graphs for this data are shown in Figure 8(a). The average silhouette width plot in Figure 8(b) shows a high average width for both $K = 5$ and $K = 6$. Because these two values for K are similarly effective and the sum of squares plot did not have a sharp elbow, the best strategy would be to disregard $K = 4$ and run the K-means algorithm for both $K = 5$ and $K = 6$ to determine which value results in desirable classifications.

Tables 1 and 2 show the resulting clusters for $K = 5$ and $K = 6$. The clusters given by both values of K demonstrate two interesting qualities of K-means. First, when partitioning observations with an increased number of groups, clusters with high in-group variance are divided up first. When K changed from 5 to 6, the new sixth group was composed of all available fish associated samples along with two human samples, therefore splitting

		Cluster Number				
Environmental Groupings	Environment Types	1	2	3	4	5
Water Organism Associated	Fish Gut	2	0	0	0	0
	Fish Slime	2	0	0	0	0
	Mat Community	9	0	0	0	0
	Whale Fall	3	0	0	0	0
Land Organism Associated	Human Gut	1	30	0	0	0
	Mouse Gut	0	2	0	0	0
	Chicken Gut	0	2	0	0	0
	Cow Gut	0	3	1	0	0
Coastal Marine	Coastal Marine	1	0	52	2	2

Table 1: K-means Clusters. Organism Associated and Coastal Marine metagenomic data with $K = 5$. Clusters appear to agree with environmental groupings. Cluster 1 corresponds with Water Organism Associated data, Cluster 2 with Land Organism Associated data, and Cluster 3, 4, and 5 with Coastal Marine data.

Cluster 1 into two groups. Second, the ability of K-means to detect outliers was demonstrated. For both values of K, three of the four coastal marine metagenomes from Botany Bay did not group with the rest of the coastal marine samples (two of which fall in Cluster 4, one in Cluster 1). This indicates variance between these samples and the rest of the coastal marine metagenomes. In addition, one human sample and one cow sample did not group with metagenomes from similar environments for either value of K. It would be useful to study these outlying metagenomic samples to understand the biological reasons they are partitioned into different clusters.

For this example, the K-means algorithm output for $K = 5$ coincides with environmental groupings. Cluster 1 corresponds with Water Organism Associated data, Cluster 2 with Land Organism Associated data, and Cluster 3, 4, and 5 with Coastal Marine data. In this case, the differences in clusters were biologically identifiable in terms of environment. For $K = 6$, Cluster 1 is divided into two separate groups and another human sample was misclassified. Therefore this value of K did not give classifications that were as desirable. One way to visualize the given results for $K = 5$ is to graph a PCA with coloring given by K-means clusterings. Despite three misclassified samples, the resulting PCA in Figure 9 shows clear clustering based on environmental groups, hence demonstrating the ability of K-means to identify

		Cluster Number					
Environmental Groupings	Environment Types	1	2	3	4	5	6
Water Organism Associated	Fish Gut	0	0	0	0	0	2
	Fish Slime	0	0	0	0	0	2
	Mat Community	9	0	0	0	0	0
	Whale Fall	3	0	0	0	0	0
Land Organism Associated	Human Gut	0	29	0	0	0	2
	Mouse Gut	0	2	0	0	0	0
	Chicken Gut	0	2	0	0	0	0
	Cow Gut	0	3	1	0	0	0
Coastal Marine	Coastal Marine	1	0	52	2	2	0

Table 2: K-means Clusters. Organism Associated and Coastal Marine metagenomic data with $K = 6$. Clusters do not agree with environmental groupings as well as for $K = 5$. Cluster 1 corresponds with half of the Water Organism Associated data, Cluster 2 with Land Organism Associated data, and Cluster 3, 4, and 5 with Coastal Marine, and Cluster 6 with the remaining Water Organism Associated data.

groups which are biologically significant. Although K-means produced nice classifications in this example, even with a clear optimal K, clusters which are identifiably meaningful in a biological way are not guaranteed.

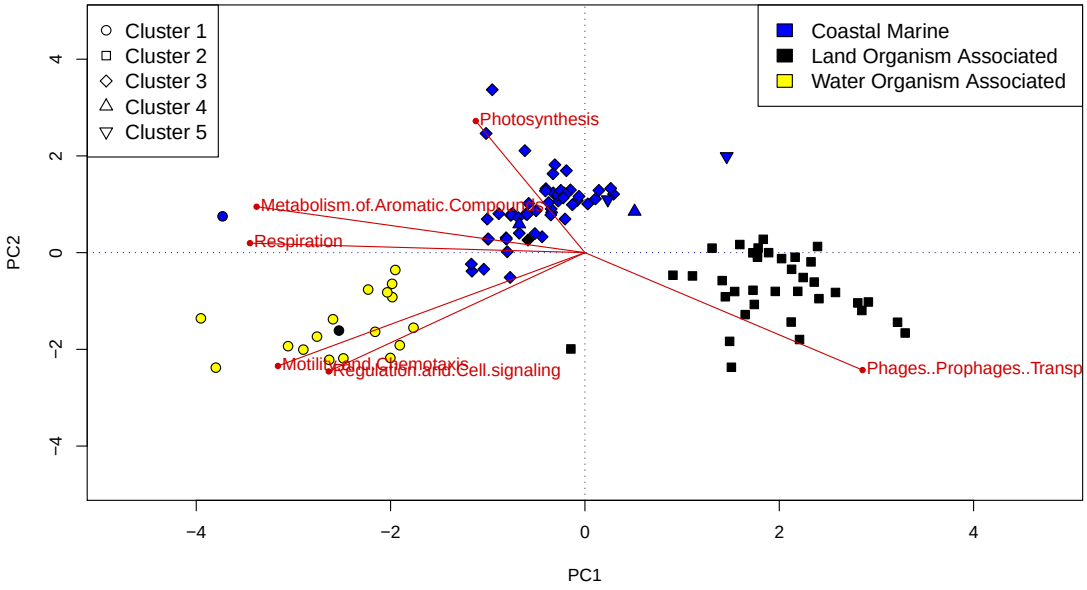


Figure 9: PCA using K-Means Clustering. Organism Associated and Coastal Marine metagenomic data. Variables selected using Random Forest variable importance plot (see Section 7.1). Despite a few misclassified samples, this PCA shows clear clustering based on environmental groups, demonstrating the ability of K-means to produce biologically significant classifications.

5 Linear Discriminant Analysis

For a data set with predetermined groups, linear discriminant analysis(LDA) constructs a classification criterion which can be used for predicting group membership of new data. That is, linear discriminant analysis is a *supervised* statistical technique. LDA finds linear combinations of the data's variables that best separate the groups, and using these linear combinations, we define linear discriminant functions. The linear discriminant functions are hyperplanes cutting through the data space trying to separate the groups.

Let $\mathbf{X}$ be a data set with defined groups 1, ... , n. For each group j, there exists a corresponding conditional distribution

$$\mathbf{X}^{(i)}|G^{(i)} = j \sim f_j. \tag{4}$$

Furthermore, let π_j represent the proportion of $\mathbf{X}$ that is contained in group j. To perform a linear discriminant analysis on $\mathbf{X}$, we must assume that each f_j is normally distributed with an equal covariance matrix, but with possibly different means [3]. With this assumption, our conditional distributions for our groups become

$$\mathbf{X}^{(i)}|G^{(i)} = j \sim N(\mu_j, \Sigma_j). \tag{5}$$

Consequently our maximum likelihood estimation becomes [15]

$$f_j(x; \mu_j, \Sigma_j) = \frac{C}{|\Sigma_j|^{1/2}} e^{-\frac{1}{2}(x-\mu_j)\Sigma_j^{-1}(x-\mu_j)'}, \tag{6}$$

where C is the constant that insures f_j is a pdf. Expanding (6) yields

$$\begin{aligned} f_j(x; \mu_j, \Sigma_j) &= \frac{C}{|\Sigma_j|^{1/2}} e^{(-\frac{1}{2}x\Sigma_j^{-1} - \frac{1}{2}\mu_j\Sigma_j^{-1})(x-\mu_j)'} \\ &= \frac{C}{|\Sigma_j|^{1/2}} e^{-\frac{1}{2}x\Sigma_j^{-1}x'} e^{\mu_j\Sigma_j^{-1}x'} e^{-\frac{1}{2}\mu_j\Sigma_j^{-1}\mu_j'}. \end{aligned}$$

To classify a given data value $\mathbf{x}$, we seek to maximize

$$\pi_j f_j(x; \mu_j, \Sigma_j) = \pi_j \frac{C}{|\Sigma_j|^{1/2}} e^{-\frac{1}{2}x\Sigma_j^{-1}x'} e^{\mu_j\Sigma_j^{-1}x'} e^{-\frac{1}{2}\mu_j\Sigma_j^{-1}\mu_j'} \tag{7}$$

[3] Logistic regression is a popular alternative to linear discriminant analysis, and does not require these two assumptions about the group conditional distributions. However, statisticians such as Jia Li and Jerome Friedman assert that LDA on a data set that has group distributions not normally distributed and/or lacking equal covariance matrices will have similar results to a logistic regression.

by choosing the appropriate j$\in\{1,\ldots,n\}$. Note that we assumed each group's conditional distribution has equal covariance, and consequently $\Sigma_j = \Sigma_k$ for all $j \in \{1,\ldots,n\}$. So let $\Sigma = \Sigma_1 = \cdots = \Sigma_n$. To find which j maximizes $\pi_j f_j(x;\mu_j,\Sigma_j)$ for a given $\mathbf{x}$, it suffices to compare $\pi_j f_j(x;\mu_j,\Sigma)$ to $\pi_k f_k(x;\mu_k,\Sigma)$ for all j. Therefore, it is unnecessary to keep the terms shared by each $f_j(x;\mu_j,\Sigma)$, and consequently we omit them. Hence we seek to maximize the following instead of (7)

$$\pi_j e^{\mu_j \Sigma^{-1} x'} e^{-\frac{1}{2}\mu_j \Sigma^{-1} \mu_j'}. \tag{8}$$

We take the natural logarithm of the above equation to obtain our *linear discriminant functions*

$$g_j(x) \equiv log(\pi_j) + \mu_j \Sigma^{-1} x' - \frac{1}{2}\mu_j \Sigma^{-1} \mu_j'. \tag{9}$$

Note that π_j, μ_j, and Σ are unknown parameters for our groups' conditional distributions, so we estimate them using our sample data $\mathbf{X}$ in an intuitive manner. Suppose $\mathbf{X}$ has N data points and group j has n_j points contained in it. Then we estimate π_j by $\hat{\pi}_j = \frac{n_j}{N}$, and μ by $\hat{\mu}_j = \sum_{i=1}^{n_j} \frac{x_i}{n_j}$. Let $\mathbf{S}_j$ be the sample covariance matrix for group j calculated from $\mathbf{X}$. Also, $\hat{\Sigma}_j$ is taken to be 1/nth of the pooled covariance matrix of $\mathbf{X}$. Consequently, $\hat{\Sigma}_j = \hat{\Sigma}_k$ for all $k \in \{1,\ldots,n\}$. Therefore, let $\hat{\Sigma}_j = \hat{\Sigma}$ for all $k \in \{1,\ldots,n\}$. With our population parameters estimated from our sample data $\mathbf{X}$, the linear discriminant functions from (9) become

$$g_j(x) \equiv log(\hat{\pi}_j) + \hat{\mu}_j \hat{\Sigma}^{-1} x' - \frac{1}{2}\hat{\mu}_j \hat{\Sigma}^{-1} \hat{\mu}_j'. \tag{10}$$

Note that (10) is a linear equation since $log(\hat{\pi}_j) - \frac{1}{2}\hat{\mu}_j \hat{\Sigma}^{-1} \hat{\mu}_j'$ is a constant.

These $g_j's$ from (10) are our classifying functions. Since for a point $\mathbf{x}$ we sought to maximize $\pi_j f_j$, our *classification criterion* is

$$\text{assign } \mathbf{x} \text{ to group j if } g_j(x) > g_k(x) \text{ for all } k \neq j.$$

With the classification criterion, decision boundaries between groups can be found. The decision boundaries are where the discriminant functions intersect. That is, the *decision boundary between group j and k* is

$$\{\mathbf{x} : g_j(x) = g_k(x)\} \tag{11}$$

Therefore, the linear discriminant functions split the data space into regions. Each region corresponds to a specific group and the decision boundaries separate the regions.

5.1 Linear Discriminant Analysis on Metagenomes

Linear discriminant analyses were performed on a broad subset of the metagenomic data. For example, we looked at the metagenomes from the marine environment, hypersaline environment, gut environment, and microorganism environment separately. For an LDA on a subset of the metagenomic data, *leave one out* cross-validation was used to judge the quality of the LDA as a classifier. To visualize the separation between groups caused by the discriminant functions, 2-dimensional plots using linear discriminants[30].

5.1.1 Cross-Validation

To judge how well a given LDA acts as a classifier for new data, a function in the statistical program R that implemented *leave one out* cross-validation was written[31]. Let $\mathbf{X}$ be a data set with m data points, and with groups $1, \ldots, n$. For an LDA carried out on $\mathbf{X}$, leave one out cross-validation removes one observation, $x^{(i)}$, at a time from $\mathbf{X}$, performs an LDA on the reduced data set, and then uses this new LDA to classify $x^{(i)}$. Since the group membership of $x^{(i)}$ is already known, we can check if the quasi-LDA for $\mathbf{X}$ classifies $x^{(i)}$ correctly or not. For every observation in $\mathbf{X}$, the procedure of leaving one out, and classifying with a new LDA is performed. The number of misclassifications is found; say we had p misclassifications. Then the proportion $\frac{p}{m}$ is an estimate for the probability of the linear discriminant analysis carried out on $\mathbf{X}$ misclassifying a new observation.

The leave one out cross-validation uses the assumption that the quasi-LDA's of $\mathbf{X}$ as classifiers are representative of the LDA of $\mathbf{X}$. For the metagenomic data, the sizes of our defined groups varied greatly. In particular we had many groups with small sample sizes which raises the question if leave one out cross-validation technique is the most appropriate way to judge a given LDA as a classifier. To illustrate why, suppose we perform a linear discriminant analysis on the data set $\mathbf{X}$ with defined groups $1, \ldots, n$. For group j, suppose $j = \{x_{j_1}, x_{j_2}\}$. During the leave one out cross validation, eventually x_{j_1} will be left out and then classified. However, when x_{j_1} is left out, our quasi-group j contains only one point x_{j_2}. Then the discriminant

function for group j is built using only one point. Consequently if x_{j_1} is not similar to x_{j_2}, then it is not surprising that x_{j_1} may be misclassified.

To view the classification results of leave one out cross-validation, a confusion matrix was constructed (See Figure 10). For a confusion matrix, the row and column names of the matrix are the group names from the data set. If C is a confusion matrix, then the C_{ij} entry in the matrix represents how many data points from group i were classified into group j. Therefore the diagonal entries represent correct classifications, and the off-diagonal entries refer to misclassifications.

$$C = \begin{array}{c} \\ G_1 \\ G_2 \\ G_3 \\ G_4 \\ G_5 \\ G_6 \end{array} \begin{array}{c} \begin{array}{cccccc} G_1 & G_2 & G_3 & G_4 & G_5 & G_6 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 2 & 0 & 2 \\ 0 & 3 & 0 & 1 & 0 & 2 \\ 1 & 0 & 2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 & 0 & 2 \end{pmatrix} \end{array}$$

Figure 10: Example of a confusion matrix. The C_{ii} entry refers to the quantity of observations from group i classified into group i. The off diagonal entries refer to a misclassification. The C_{ij} entry represents how many observations from group i were misclassified into group j.

5.1.2 Subsampling

For a meaningful misclassification measure from a leave one out cross-validation, subsampling prior to performing an LDA may be needed. Subsampling should be implemented when group sizes have large variation, to reduce one or more groups' sizes. Subsampling refers to taking a random subset of a set with replacement. That is, subsampling k points from a group G is the process of randomly choosing k samples from group G with replacement. This gives k random samples from G, and these k samples are taken to be the new group G. The motivation for subsampling groups for linear discriminant analysis is to prevent groups with large sample sizes from dominating the leave one out cross-validation.

Consider the gut sample metagenomes(Table 3). The twin group consists of 18 samples, and the human group consists of 13 samples. These

two group sizes are significantly larger than the other groups from this data set, and therefore, dominate an LDA–in particular the human and twin groups monopolize the misclassification rate reported from the cross-validation. Two different linear discriminant analyses were run on the gut sample metagenomes; one with subsampling, and one without subsampling (Table 3). The leave one out cross-validation led to differing results depending on whether subsampling was used or not. For an LDA performed on the original groups, the leave one out cross-validation resulted in 27% misclassification rate. For the gut sample metagenomes with subsampled human and twin groups, the leave one out cross-validation resulted in a 35% misclassification. Differing conclusions can be drawn from the two linear discriminant analyses even though the misclassifications rates only differ by 8%.

The human and twin groups account for 31 out of 41 total data samples for the gut metagenomes, and 16% of the twin and human samples were misclassified. The remaining 10 data points, which represent 4 of the 6 groups, had a 60% misclassification rate. Since each data sample contributes equally to the misclassification measure, the human and twin groups are overrepresented in the misclassification measure.For the metagenomic data, near equal contribution to the misclassification measure by each group is desired. In this instance, it is wise to subsample the human and twin groups, in order for each group to contribute fairly to the misclassification measure.

For the LDA performed on the gut sample metagenomes that were subsampled, there was a 35% misclassification rate, as mentioned above. From the confusion matrix (Figure 11(b)), it is evident that the fish samples are classified correctly, while the mouse samples were both misclassified. Besides these comments, it is difficult to take away other conclusions from the confusion matrix.

$$\begin{array}{l} Chicken \\ Cow \\ Fish \\ Human \\ Mouse \\ Twin \end{array} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 3 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 9 & 2 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 17 \end{pmatrix}$$

(a) Original Gut Metagenomes

$$\begin{array}{l} Chicken \\ Cow \\ Fish \\ Human \\ Mouse \\ Twin \end{array} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

(b) Subsampled Gut Metagenomes

Figure 11: Confusion Matrices for LDA's Performed on Gut Metagenomes

Gut Sample Metagenomes		
Group Name	Original Sample Size	Sample Size After Subsampling
Chicken	2	2
Cow	4	4
Fish	2	2
Human	13	3
Mouse	2	2
Twin Study	18	4

Table 3: Group sample sizes before and after subsampling was performed

5.1.3 2-Dimensional Plots using Linear Discriminants

The linear discriminant functions are functions of 27 variables for the original metagenome data. Consequently we cannot visualize the discriminant functions directly in the data space, and how they separate the groups. To see the separation of the groups, a 2-dimensional plot using the *linear discriminants* as axes variables are made in *R*[31]. Linear discriminants, not to be confused with linear discriminant functions, are linear combinations of the variables that best represent the between-class variance[30]. There is a close relationship between linear discriminants and the linear discriminant functions, and therefore, no harm is done in using linear discriminants as our axes.

Consider the Organism Associated and Mat-forming Metagenomes (Table 4). The mat community group consists of samples from microbial mats, the microbiolite group consists of samples taken from sedimentary structures structures that are composed of microorganisms, and the slime group consists of slime samples taken from fish. Figure 12 shows a 2-D plot using the first two linear discriminants from the LDA performed on the organism associated and mat-forming metagenomes. The mat community and microbiolites are plotted next to each other, while there is separation between the other groups. Since the first two linear discriminants represent 91% of the between-group variance, this plot represents most of the group separation caused by the LDA. Furthermore, the leave one out cross-validation of the LDA resulted in 5 misclassifications. 3 of the 5 misclassifications were microbiolites misclassifying into the mat community, and 1 of the remaining 2 misclasifications was a mat community sample misclassifying into the micro-

Organism Associated and Mat-forming Metagenomes	
Group Name	Sample Size
Coral	6
Mat Community	10
Microbiolites	3
Slime	2

Table 4: Organism Associated and Mat-forming Metagenomes

biolites. These four misclassifications are reasonable from Figure 12, since the mat community and microbiolite groups were plotted next to each other.

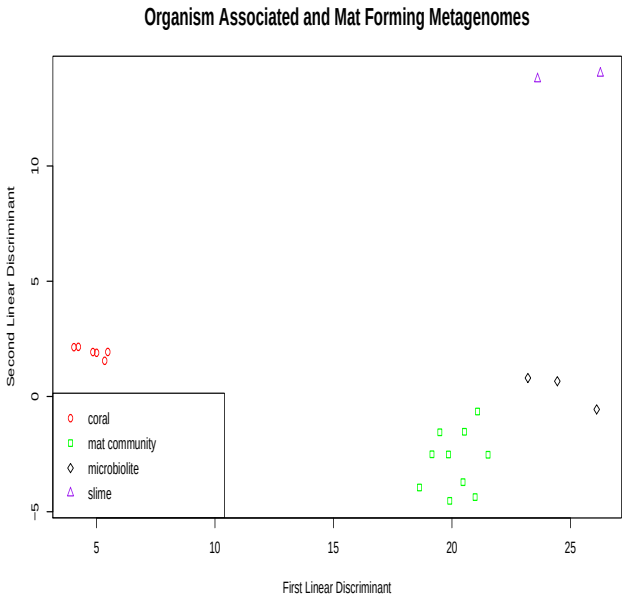


Figure 12: Together, the first two linear discriminants explain 91% of the between-class variance. Five misclassifications resulted from leave one out cross-validation: all three microbiolites were misclassified into the mat community, one mat community sample was misclassified as a microbiolite, and one coral sample was misclassified into the mat community.

6 Trees

6.1 An Introduction to Trees

A tree analysis generates a branching decision scheme, which graphically relates values of a response variable to values of one or more predictor variables. There are two types of trees, classification trees and regression trees. When the dataset is divided into predetermined classes–generally by a discrete-valued response variable–the tree algorithm constructs a *classification tree*, which uses the predictor variables to distinguish between classes. When classes are not provided, the process constructs a *regression tree*, which predicts average values of the response variable. At each branching point, trees of each type attempt to minimize the *node impurity*, or the mixing of different values of the response variable on a single side of a branch. However, the two types of trees rely on different measures of impurity. Our analysis will focus on classification trees, outlining the theory, construction, and application of these trees as they relate to metagenomic analysis.

Most trees are constructed using a greedy algorithm, which begins at the root node and proceeds down a series of binary branching decisions until terminating at the leaves. Although the set of predictor variables and splitting points chosen by this greedy algorithm may not be globally optimal, the procedure is computationally efficient. At each branch, the algorithm chooses a single predictor variable and a critical value of this variable which minimizes the impurity of the resulting node. Specifically, most tree algorithms use either the Gini criterion, the twoing criterion, the deviance, or the misclassification rate as methods of measuring and ultimately of limiting node impurity [3, 14]. Deviance is based on the likelihood of each split, and the Gini criterion is described in the section on Random Forests 7. Minimizing the deviance or Gini during tree construction and then minimizing the misclassification rate during tree cross-validation is a standard technique [7]. As a result of their reliance on binary partitions, trees are invariant under monotonic transformations of the predictor variables and thus most variable scaling is unnecessary. Good trees balance classification strength against model complexity to maximize prediction strength and minimize over-fitting. Most authors suggest growing a large tree and using a *pruning* technique to select the strongest model from a nested series of subtrees of that original tree [3].

Zone	Cell Wall	Photosynthesis
tropics	0.055789155	0.002531765
n.temp	0.054635574	0.00270913
n.temp	0.047888534	0.00272034
tropics	0.053397667	0.002728348
⋮	⋮	⋮

Table 5: An excerpt from the coastal marine dataset

6.2 Coastal Marine Samples by Geographic Zone

The tree in Figure 13 uses photosynthesis and cell wall hierarchies to classify various coastal marine metagenomes based on the geographic zone in which each sample was collected. First, an initial tree containing 7 different leaves was grown to fit the coastal marine data, an excerpt of which is given in Table. 5 [22]. Next, this tree was pruned to minimize the 10-fold cross-validation rate over a series of 100 random trials [4]. Limiting the data to metagenomes of a single type (marine) from a single environment (coastal), helps eliminate confounding variables, and focuses the analysis on differences between the two geographic zones. Coastal samples from the southern temperate zone were excluded because of a lack of sufficient data, and samples from between $23.4°N$ and $30°N$ were also excluded in order to better distinguish between samples from the two remaining zones, northern temperate and tropical. The resulting tree is effective as a classifier, with an error rate of only 12%, although its error rate as a predictor may be higher. Furthermore, the model indicates simple relationships between the predictor and the response variables: coastal samples from the tropical zone exhibit higher diversity in photosynthesis and cell wall hierarchies than those collected in northern temperate zone.

The scatterplot in Figure 14, indicates the correspondence between the sample space and the tree diagram. Each coastal marine data point is labeled based on the geographic zone in which it was collected, and each axis

[4]See Section 6.3 for more information on cross-validation procedures. Figure 15 plots this misclassification rate for various tree sizes

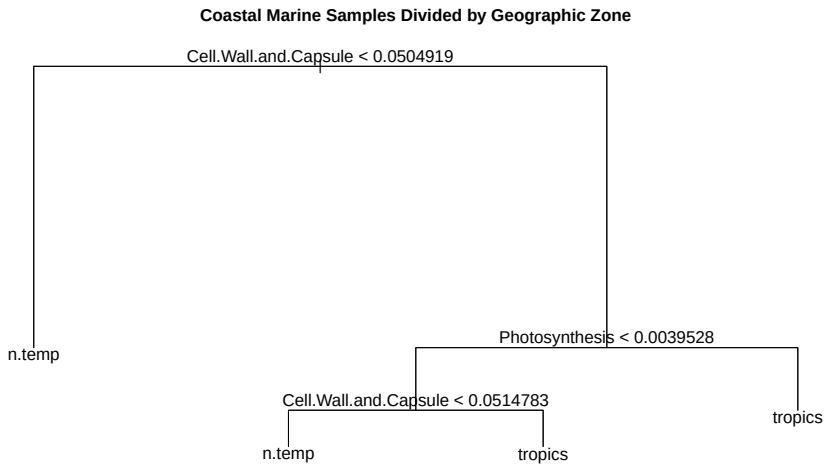


Figure 13: Coastal marine samples from two geographic zones: northern temperate and tropical. The tree plot groups samples using as predictor variables the relative diversity of the cell wall hierarchy and of the photosynthesis hierarchy. Labels on each branch indicate the splitting criteria, and labels on the leaves indicate classifications. For example, samples with less than 5.05% of identified genetic material in the cell wall hierarchy are grouped on the left side of the first branch and are classified as tropical. The plot is both a classification tool and an indicator of how these functional groups change across geographic zones.

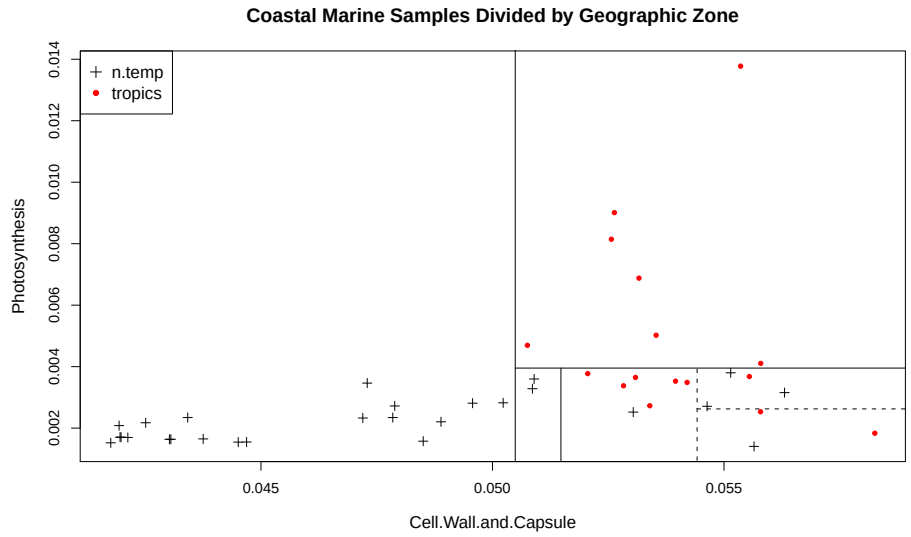


Figure 14: A scatter plot corresponding to the coastal marine data in Figure 13 The two axes correspond to the two functional hierarchies, and data points have been labeled based on geographic zone. Solid lines within the plot indicate branching in the tree diagram, whereas dashed lines indicate branchings in the over-fitted tree, which was originally grown. All divisions in the sample space are rectangular, later divisions (all those on the bottom right) divide between fewer data points, and the predictive strength of each split must be determined through cross-validation.

corresponds to one of the predictor variables, the x-axis to cell wall diversity and the y-axis to photosynthesis diversity. The solid lines drawn inside the plot indicate branchings in the tree diagram. The first branch corresponds to the vertical division separating low and high levels of cell wall diversity, and subsequent branches correspond to the solid lines in the lower right. It is evident from the scatterplot that all divisions in the sample space are rectangular, i.e. that branching only depends on whether a single predictor variable is greater than or less than a critical value.

The original 7-leaf tree was pruned back after cross-validation, and the scatterplot indicates that the tree was likely overfit. Dashed lines within the plot indicate branches grown in the original tree that were pruned back during cross-validation. The position of each of these divisions was determined by very small subsets of the original dataset: only 6 samples were used to fit the horizontal line on the bottom right. The final branches in a large tree

diagram are fit using small subsets of the original data and may not be statistically significant. As long as data points from separate classes are distinct, a tree can be grown until it perfectly fits the data set, although in extreme cases this may result in a tree with only one data point per leaf. Cross-validation and pruning techniques help avoid this type over-fitting, and many of the techniques are more straightforward than those available for other classification techniques.

6.3 Pruning and Cross-Validation

After a tree has been grown, model selection algorithms can be used to select a best-fitting subtree. The different model selection techniques balance tree size against classification strength, seeking to minimize the quantity:

$$\lambda(treesize) - impurity(Tree). \tag{12}$$

In Equation 12, λ is a *cost complexity parameter*, which determines the specific balance between model size and strength. Depending on the specific model selection tool, the term for impurity will be replaced by a specific measure such as the misclassification rate or the deviance of the entire tree[5]. Given a cost complexity parameter, one can select a unique subtree from a nested series inside the original tree that minimizes the quantity in Equation 12 [3]. This procedure yields a sequence of subtrees of decreasing size; the length of this sequence is at most the number of leaves in the original tree. The choice of this cost complexity parameter is not clear, but cross-validation can be used to select a best-fitting tree from this sequence of subtrees.

Many projects in metagenomics already suffer from a lack of sufficient data, and so cross-validation techniques should attempt to preserve as much data as possible in the training set. One of the most efficient techniques is *k-fold cross-validation* [3]. In this method, the data set is divided into k randomly selected groups of near equal size. A tree is built using the data points in only $(k-1)$ of these subsets, and a sequence of subtrees is constructed as described in the previous paragraph. Next, this tree and its subtrees are used to predict the classes of the remaining $1/k$ data points,

[5]Total deviance is computed by summing the deviance of each leaf, and this value should be distinguished from the single-node deviance used in tree construction.

Tree Size	Cross Validated Deviance
1	68.99766
2	67.43564
3	65.40249
4	63.74900
6	63.51468
7	70.38529

Table 6: A table comparing average deviance and tree size, for the coastal marine data set. Estimated total deviance values were computed using one hundred 10-fold cross-validation trials. Using this method of cross-validation one would select a tree with either 4 or 6 leaves. Note that this is a larger tree than selected using the method in Figure 15.

and these predictions are compared against the actual classes of these data points. The misclassification rate and the cross-validated deviance estimate are computed for each tree, and the process is repeated for each group of $k-1$ subsets. The misclassification and deviance values for each tree size are averaged over the k repetitions, and the best subtree can be selected. This k-fold cross-validation procedure can be repeated several times, so that different subsets are selected in each trial.

Several statistical packages, including R, have implementations of k-fold cross-validation. However, some freedom remains in selecting the specific method and style of cross-validation. One of the simplest methods is to average the cross-validated deviance over several trials, and to select the tree size which minimizes this deviance. Then, a full tree can be grown using all of the available data, and this tree can be pruned back to the selected size [15]. However, this method often over-estimates the number of branches that should be included in the final tree [7]. An alternative method that addresses this issue is as follows: Determine the misclassification rate over several cross-validation trials. Next, compute the standard error in the misclassification rate. Finally, select a larger tree only if its misclassification rate is one standard deviation less than that of a smaller tree [3]. We applied both methods to the coastal marine dataset with $k = 10$. The results of 100 cross-validation trials are given in Figure 15 and in Table 6. Although

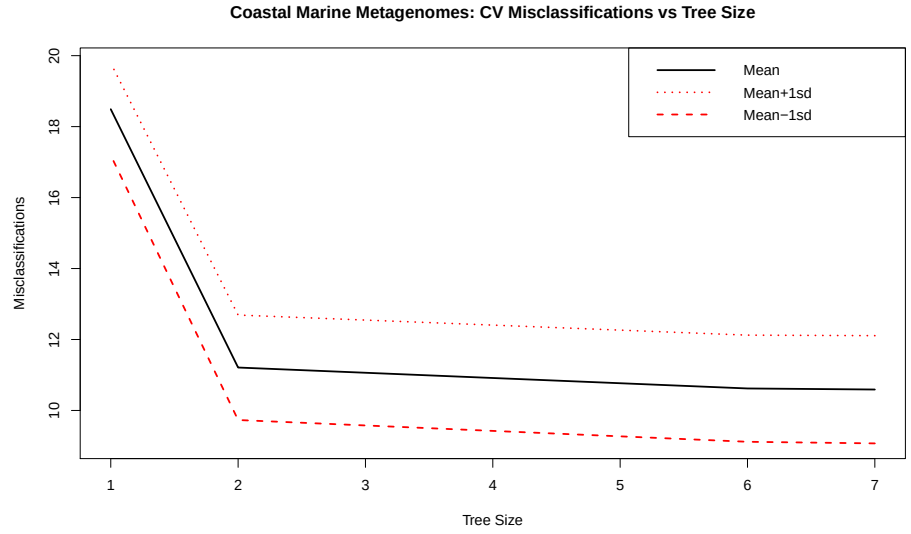


Figure 15: A plot of the average number of misclassifications versus sub-tree size, where the original 7-leaf tree was grown using the coastal marine dataset. The misclassification rate is based 100 trials of a 10-fold cross-validation experiment. In contrast to the method of minimizing cross-validated deviance, this plot suggests that a best-fitting tree only include one branch, the first branch in Figure 13.

there are many different options for and implementations of these model selection algorithms, a basic understanding of the procedures will result in reasonable trees well suited for data exploration. Trees constructed using cross-validation tools are less susceptible to overfitting than many other forms of classification.

6.4 Applications to Metagenomics

Since metagenomic data often includes a large number of variables but only a small number of samples, variable selection is an important consideration. Unlike other classification methods (such as linear discriminant analysis), trees often select small subsets of the original predictor variables for use in classification, and so trees can be used as variable selection tools. However, since the set of variables chosen by a tree is not necessarily optimal and since trees do not give information on any predictor variables not used in classification, trees are not ideally suited for variable selection. Other tools, such

as the related Random Forests 7, are preferable. If a subset of the predictor variables is of special biological or statistical interest, a tree analysis can be performed just on that subset of variables. Selecting variables before growing a tree will decreasing computation time and can also increase the significance of the final analysis. For example using the Random Forest technique, one can determine which predictor variables are strong classifiers, and a tree can be grown just using these variables. However, the significance of a tree grown using a preselected variable set can be challenging to calculate, and such calculation ultimately depends on the specifics of the variable selection procedure.

The standard tree construction scores errors between any pair of groups in the same way. The tree in Figure 17 includes samples from two different studies of human subjects, labeled as twin gut and as human gut. The tree attempts to separate these classes of human associated samples just as strongly as it attempts to separate the human samples from the fish associated samples. It might not be important for a tree to distinguish between these two very similar classes, whereas a good tree plot should probably distinguish between the human samples and metagenomes from deep water marine environments. In order to account for similarities between classes or to prioritize separations between certain classes, one can run the tree analysis using a loss matrix. This matrix includes a scoring system for penalizing misclassifications between different groups, and the resulting tree should emphasize separations between groups as prioritized in the loss matrix. However, the implementation of this technique in many statistical programs and the choice of reasonable scoring weights remain obstacles [6] .

Another consideration when using tree analysis is its frequent instability with respect to changes in the dataset. If even a small number of samples is discarded from the dataset and a new tree is constructed, this second tree may use a different set of predictor variables than the first, and the two trees may not be easily compared. This can complicate cross-validation, which assumes that the misclassification error rate of a tree grown with the entire dataset is approximately equal to that of a tree grown with $k - 1/k$ of the data set. Instability generally increases as both the size of the dataset decreases and the number of predictor variables increase, so finding a sufficient number of metagenomic samples for the construction of stable trees can be

[6]The R package *rpart* [26] allows the specification of a loss matrix whereas the package *tree* [22] allows the specification of observational weights.

challenging. Variable selection can help decrease instability issues, but at the price of decreasing the tree's simplicity and eliminating its use as a variable selection method. Random Forest techniques address problems of both variable selection and instability while eliminating the need for cross-validation. However, Random Forests often lack the the simplicity which makes trees an attractive analytical option.

6.5 A Tree Analysis of Various Microbial Metagenomes

This final subsection presents a brief overview of the tree construction procedure as applied to a simple dataset consisting of metagenomic samples of five different classes. An overlarge tree is first grown, and the results of 10-fold cross-validation experiments are presented in Table 7 and in Figure 16. Since a 7-leaf tree was the largest tree which could be grown using this dataset, the misclassification plot in Figure 16 terminates before sloping upwards for higher values of tree size. The lowest value of average cross-validated deviance is attained with a 6-leaf tree. However, the misclassification method suggests a choice between the 3-leaf tree and the 5-leaf tree, which has an error rate close to one standard deviation less than the 3-leaf tree. The smaller tree separates between human and twin samples, and groups all other samples on the other side of the tree. This grouping results from sample size issues–the human and twin studies were larger than the other studies–and could be altered using a loss matrix. The 5-leaf tree is given below in Figure 17. This tree groups twins and humans on the same side of the tree, it distinguishes between samples from each of the five different classes, and it indicates the importance of the respiration hierarchy in distinguishing between terrestrial associated human gut samples and the remaining sample types. A more detailed exploration will require a larger dataset and additional statistical tools.

Various Microbial Metagenomes: Tree Size	Tree Size vs. Deviance: Average CV Deviance
1	163.4791
2	172.7798
3	161.7896
4	150.6584
5	147.9482
6	146.6510
7	150.7419

Table 7: A table of average deviance versus tree size for an analysis of various microbial metagenomes. The minimum deviance is attained with a 6-leaf tree, although the improvement between a tree of size 5 and size 6 is minimal. The tree resulting from pruning to 5 leaves is plotted in Figure 17, which can be found below.

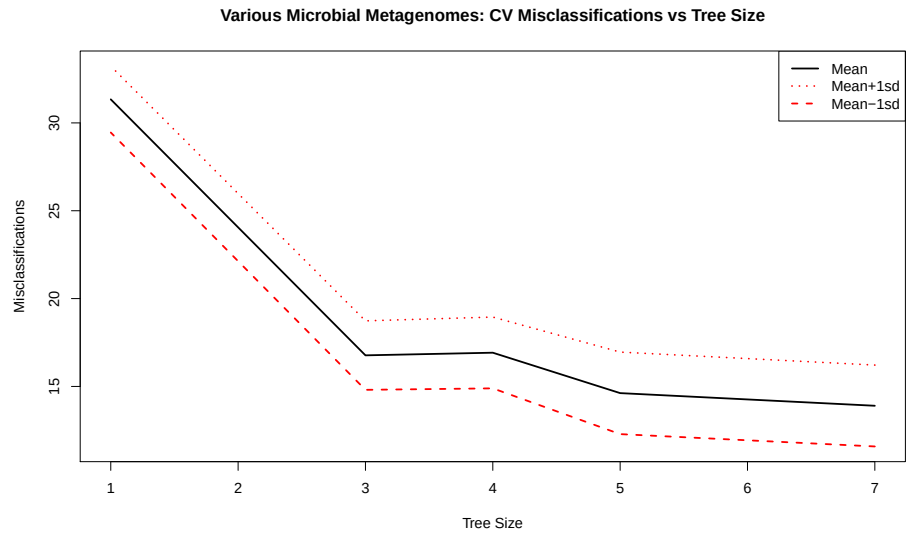


Figure 16: A plot of CV misclassifications for a tree analysis of the mixed microbial dataset. The mean and standard deviations were computed after performing one hundred 10-fold cross-validation trials.

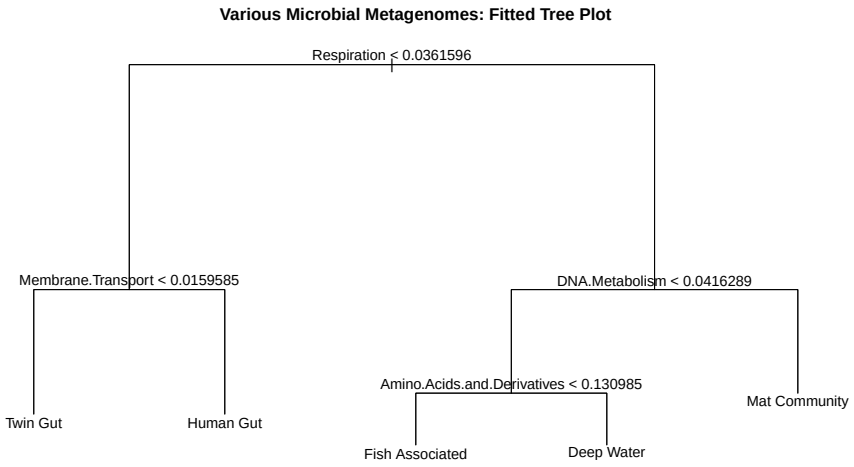


Figure 17: A classification tree dividing various microbial metagenomes into five groups corresponding to sample type. The tree plot distinguishes between data groups with misclassification error of approximately 19%, and was constructed by pruning a larger tree. The number of leaves was selected from the cross-validation experiments, which are depicted in Table 7 and Figure 16. The importance of the respiration hierarchy in distinguishing between marine and terrestrial associated samples is evident from the first branch.

7 Random Forests

While decision trees are useful classification instruments, the biggest drawback is their lack of robustness. Re-running the same data can yield different results. Instead, Random Forests[25] examine a large ensemble of trees. We generate the different trees by randomly choosing a subset of the data and variables at each step. We can then count how many times given results were observed and return the most plural result. This is analogous to a democracy where each member is given a vote and majority rules. Random Forests may be used alone or in conjunction with other clustering and graphical methods to ascertain a range of information.

Another benefit of Random Forests is that they are not computationally intensive. A Random Forest with one thousand trees trained on two hundred metagenome datasets takes only seconds to compute on a desktop computer.

7.1 Supervised Random Forest

In the supervised version of a Random Forest, class groupings are given to the classification algorithm. The trees are generated to classify around these given groups. A random sampling of available metagenomes is chosen with replacement to generate each tree as shown in Figure 18. Furthermore, at each node, a random subset of the available variables are used to determine node splitting, unlike the single classification tree which would examine them all. The number of variables to be considered at each node can be specified as an input variable to the algorithm.

New metagenomic data is analyzed by all the trees and classified into the group that the plurality of the trees indicate. The Random Forest can then use the generated trees to predict the environment to which a new unclassified metagenome belongs. Due to the plurality consensus of this method, the instability of a single tree is overcome, resulting in a more robust result, especially when using a large number of trees. However, because the trees in the Random Forest all differ from one another, unlike a single classification tree, the Random Forest does not produce branching rules. Instead, the Random Forest produces an out-of-bag (OOB) error, a confusion matrix, and variable importance measures.

To grow each tree, a new dataset, the same size as the initial training set, is generated by sampling with replacement. The term for this process is *bagging*, which stands for *b*ootstrap *ag*gregat*ing*. The metagenomes that are

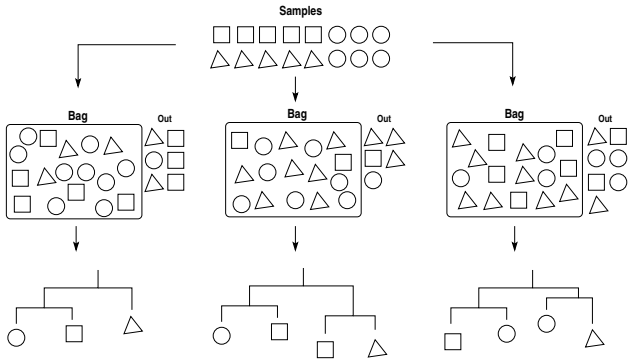


Figure 18: The squares, circles and triangles represent samples from three different metagenome environments. The metagenomes are used to create many permutations of in-bag and out-of-bag sets. The in-bag sets are generated by bootstrap sampling with replacement. The out-of-bag sets are composed of all metagenomes that were not selected at least once by the bootstrap sampling. Each in-bag set is used as training data to generate one tree. Each tree has a different structure for predicting the classes square, circle and triangle.

chosen at least once during the sampling process are considered *in-bag*. The remaining metagenomes are considered *out-of-bag* (*OOB*) for the resulting tree. These new datasets are used to generate a single tree each in the Random Forest, as depicted in Figure 19. Upon completion of the forest, each metagenome sample is out-of-bag for a subset of the trees. That subset is used to predict the class of the metagenome. If the predicted class does not match the original given class, *OOB* error is increased[25]. A low *OOB* error means the forest is a strong predictor.

Misclassifications contributing to the *OOB* error are displayed in a *confusion matrix* as shown in Figure 20. The confusion matrix shows how many of the metagenomes were classified in each class, on a class basis. If some classes are often misclassified as each other, that may indicate a biological closeness between those groups. Furthermore, the confusion matrix may clarify the *OOB* error. A Random Forest with a high *OOB* error may still be a strong predictor of particular environments. A Random Forest with a low OOB error may not actually be a good predictor – in the case where the classes have highly varying sizes, if all the small classes are misclassified as the larger ones, *OOB* error will still be small.

While the Random Forest does not have branching rules, biological insight

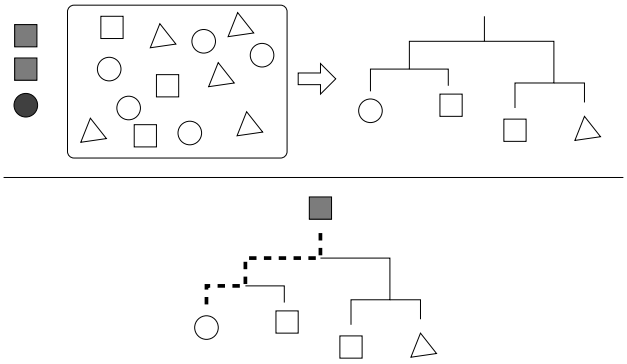


Figure 19: Graphical representation of samples sorted into *in-bag* and *out-of-bag*. The in-bag samples are used to generate a tree. The out-of-bag samples are compared to all trees for which they were out-of-bag. In this figure, a square was predicted to be a circle. This would increase the *OOB* error.

	human	hypersaline	mat community	spring	terrestrial	water	class.error
human	31	0	1	0	0	1	0.0606061
hypersaline	0	8	0	0	0	7	0.4666667
mat community	0	0	9	0	0	1	0.1000000
spring	0	2	0	3	0	1	0.5000000
terrestrial	5	0	0	0	3	1	0.6666667
water	0	0	1	0	0	129	0.0076923

Figure 20: Confusion matrix showing results from a Random Forest generated from 33 human, 15 hypersaline, 10 mat community, 6 spring, 9 terrestrial, and 130 water environments. The rows in the confusion matrix represent the given classes of the metagenomes. The row sums without class.error equal the total number of samples of each class. The columns represent the classes predicted by the subsets of the trees for which each metagenome was *OOB*. Each class error, weighted for class size, contributes to the single *OOB* error. The overall *OOB* error in this example is 9.85%, with the hypersaline and terrestrial classes being misclassified the most often.

may be ascertained by exploring variable importance measures such as *mean decrease accuracy* and *mean decrease Gini*. These values indicate which variables contributed the most to generating stronger trees. We then use the indicated important variables to generate single trees with branching rules or used in canonical discriminant analyses.

The mean decrease accuracy of a variable is determined during the *OOB* error calculation phase. One at a time, each variable is randomly permuted along the set of metagenomes. The classification is then computed by the subset of the Random Forest similarly to the computation of normal *OOB* error. The more the accuracy of the Random Forest decreases due to the permutation, the more important variable is deemed[25]. Variables with large mean decrease accuracy are more important in proper classification.

The mean decrease Gini is a measure of how a variable contributes to the homogeneity of nodes and leaves in the Random Forest. Let p_{mg_i} be the proportion of samples of group g_i in node m. Let g_c be the most plural group in node m. The Gini index of node m G_m is defined as:

$$G_m = 1 - \sum_{i \in g} p_{mg_i} * p_{mg_i}. \tag{13}$$

The Gini index is a measure of the purity of the node, with smaller values indicating a purer node and thus a lesser likelihood of misclassification[14]. Tree generating algorithms may use this index as their likelihood to pick which variable to split on. Each time a particular variable is used to split a node, the Gini index for the child nodes are calculated and compared to that of the original node. When node m is split into m_r and m_l, there is a probability p_{m_r} of samples going into the child node m_r and p_{m_l} of going into m_l. The decrease[3] in Gini is then:

$$D_m = G_m - p_{m_r} G_{m_r} - p_{m_l} G_{m_l}. \tag{14}$$

The calculated decrease is added to the mean decrease Gini for the splitting variable and normalized at the end. The greater the mean decrease Gini of a variable, the purer the nodes splitting.

In Figures 21 and 22, the variable importance plots for a Random Forest generated from the microbial data sets are shown. Looking at the natural breaks in the graph, we chose the top scoring 7-8 variables from both. Because many are overlapping, this generates a set of 11 unique variables used to generate trees or canonical discriminants.

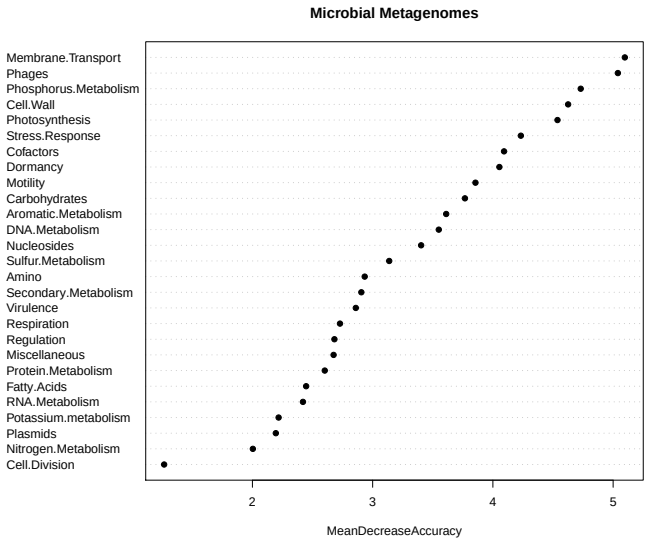


Figure 21: Mean decrease accuracy variable importance plot for Aquatic, Human, Hypersaline, Mat Community, Spring, and Terrestrial groupings. Variables Phages and Membrane Transport contribute greatly to the classification accuracy of the generated trees.

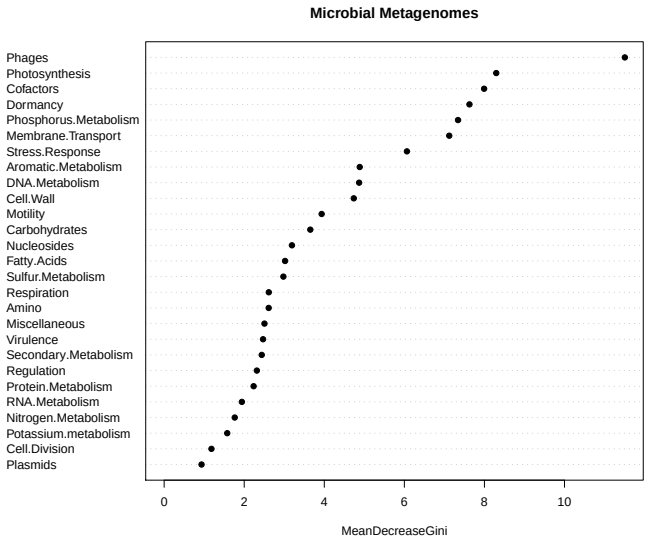


Figure 22: Mean decrease gini variable importance plot for Aquatic, Human, Hypersaline, Mat Community, Spring, and Terrestrial groupings. The variable Phages contributes greatly to the node purity of the generated trees.

Supervised Random Forests are a good tool for classification, prediction and variable selection. The variable importance measures generated offered insight into which hierarchies strongly differentiate metagenomes. The importance rankings generated can be used to select variables for other techniques. Out-of-bag error and confusion matrices can be used to assess the quality and differentiability of the metagenome groupings.

7.2 Unsupervised Random Forest

In an unsupervised Random Forest, the metagenome data is input without class specifications. Instead, synthetic classes are generated randomly and then the trees are fit. Clusters form because metagenomes will end up in the same leaves despite the synthetic classes. As Figure 23 shows, the samples that are similar may still end up in the same leaf despite different synthetic classes. Since the prediction and variable importance calculations in this method are based on the synthetic classes, these features are not useful for an unsupervised Random Forest.

Whenever two metagenomes end up in the same leaf of a tree in a Random Forest, their proximity measure is increased[23]. This proximity is normalized so that a metagenome has a proximity of one with itself. Then, $1-$proximity can be used as a dissimilarity measure for partitioning around medoids (PAM). The dissimilarity grouping can be visualized using a multidimensional scaling (MDS) plot which works similar to PCA but on dissimilarity data. An example using all three of these techniques will be given after the section on PAM.

7.3 Partitioning Around Medoids (PAM)

Partitioning around medoids (PAM)[15] is similar to K-means with a different measure. Like K-means, PAM attempts to create K clusters, where K is given. Also, original grouping size variances are not much of an issue, though within group-variances may cause problems. Unlike K-means, the measure used need not obey the triangle inequality. Dissimilarity measures should be symmetric between metagenomes and each metagenome should have a dissimiliarity of 0 with itself.

Instead of clustering around calculated means, PAM creates its clusters around whichever K metagenomes (known as medoids) minimize the sum of the distances between other cluster members and the medoid. Silhouettes

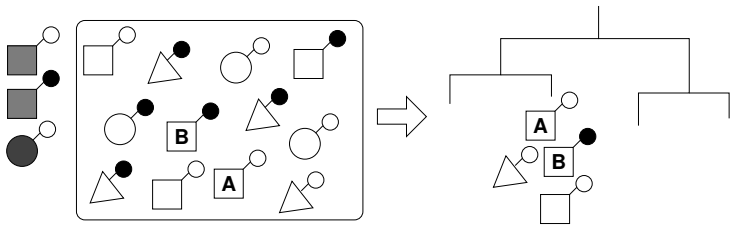


Figure 23: The squares, circles and triangles represent samples from three different metagenome environments. Samples are randomly flagged with synthetic classes *black* and *white*. Samples A and B have different synthetic classes, but are sorted into the same leaf during tree generation. Every time two samples are in the same node, their proximity measure is increased. In this example, the proximity between sample A and sample B would be increased.

can be calculated as they were for K-means to help determine a value for K. Again, visualization of the groups can be done with an MDS plot.

Unsupervised Random Forest with PAM classifying can be a strong argument for metagenome groupings – none of the original grouping information was used, yet groupings formed within a large number of randomly created trees. The method also has the benefit of not relying on any particular distribution of the data, which works well with metagenomic data which may not be normal.

7.4 Applications to Organism-Associated and Mat-Forming Metagenomes

Supervised and unsupervised Random Forest methods explain variations in data samples via different mechanisms which result in each method providing the opportunity to glean biological connections from different points of view. A good example of the differences and interplay between the results obtained from supervised Random Forest and unsupervised Random Forests with PAM is the case of 4 distinct microbial environments that were analyzed together.

The four environments are: coral ($n = 6$), slime ($n = 2$), mat community ($n = 10$) and microbiolites ($n = 3$). Microbial slime samples were collected from the sides of freshwater fish that were found to be high in Sulfur. Both healthy and morbid fish slime samples collected. The coral samples were collected from the community of metazoans, protists and mi-

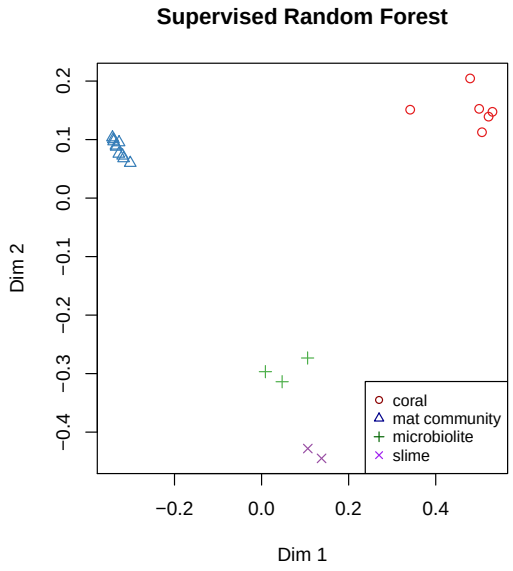


Figure 24: Four organism environments: Coral, Mat Community, Microbiolite, and Slime. The three Microbiolite microbial samples come from very different environments and naturally class with organisms that live in those environments.

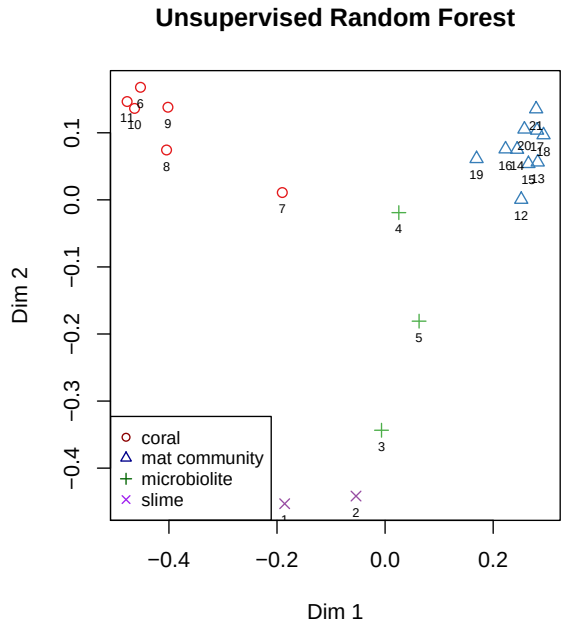


Figure 25: Four organism environments: Coral, Mat Community, Microbiolite, and Slime. Even though the microbiolite are highly different from one another, it was able to find subsystems that uniquely differentiate each community from the rest. Hence the four tight clusters we see here.

crobes associated with scleractinian corals, otherwise known as the coral holobiont. Metagenomes from both healthy and stressed coral (each subject to a different kind of stress) samples were collected [28]. The mat-forming metagenomes came from porous Teepee-like structures of evaporative salt deposits. Vast and diverse microbial life was found within the evaporates creating layered hypersaline communities that had developed highly adaptive methods for surviving in their extreme environmental conditions(cite/). Last the microbiolite came from stromatolites and thrombolites found in a marine location at Highborne Cay in the Bahamas and two freshwater locations: Pozas Azules II and Rio Mesquites in Mexico [8].

The different metagenomic samples of a given environment type, as described above, were found to exhibit high genomic and physiological diversity. While being phylogenetically related microbes, a genetic uniqueness could be ascribed to each of them. Further microbes with similar lifestyles were found to contain a core or minimal set of genes required for survival in similar conditions [28].

Such findings emanated from a newly formed school of thought that has found that functional subsystems uniquely define environment-types [9]. Detractors to this novel idea often posit the enormous complexity and variability within environment-types and the 'sharing' of given genomic parts by microbes that belong to different environments making it difficult to uniquely differentiate given environment-types. We were interested in describing these findings via different statistical methods.

To do so, we would first need to employ a classification algorithm that would create tight clusters for each environment-type, in spite of the genomic diversity among samples belonging to the same environment-type. And that at the same time would display clear separations among environment-types, in spite of the high similarities of environments-types (it is often very challenging to statistically differentiate mat communities and microbiolites). Secondly, we needed a classifier that was unbiased of environment-type membership which would find relevant associations of individual samples with other different environment-type microbes who shared similar lifestyles.

To carry out the first part of the analysis we employed a supervised Random Forest algorithm which generated the proximity measure for each of the $n = 21$ samples and then employed an MDS plot that found the linear combinations of the proximity measures that would minimize the intra-cluster distances and maximize the inter-cluster space when the 27 functional subsystem space was collapsed into the 2 dimensions that best described the

clusters. Figure 24 shows the MDS plot of the supervised Random Forest proximity measures of the four microbial environment-types.

Because each of the environments was placed in a single tight cluster with statistically significant separation between them, we know that the proximity values must be high within each group, leading to clear divisions in the MDS plot. The algorithm knows the desired groupings so it does its best to create branching patterns among the 27 subsystems that are best at differentiating the desired classes. It should be noted that such results are not the norm, especially when dealing with environmental metagenomic data. The confusion matrix showed an error of 19% which came primarily from microbiolite being misclassified as mat. One coral that was stressed by adding nitrogen (nutrient) to it is sometimes misclassified as a mat due to the high nitrate concentration in the hypersaline deposit.

The four unique clusters seen on Figure 24 support findings by Dinsdale et al to the effect that metagenomic functional subsystem analysis is able to clearly define environments [9], and explain variations among and within microbial communities.

For the second part of the analysis we employed an unsupervised Random Forest algorithm to generate proximity measure for each sample, unbiased environment-type membership. Because the unsupervised Random Forest starts by generating random synthetic classes ignorant of the sample's true group membership, an unbiased 'natural' clustering emerges. While the supervised RF was employed to find given classes (i.e. to find the core shared set of genes per members of a given environment), the unsupervised RF is used to glean the natural tendencies of given samples (i.e. will samples cluster with others who share not their group membership but their lifestyle?). The results were plotted using an MDS plot depicted in Figure 25. This was followed by generation of silhouettes and a PAM algorithm that would tell us both the 'ideal' k number of clusters for this data and the kth group membership of each of sample.

Figure 25 shows that while coral, mat community, and slime environments each cluster together nicely, the microbiolite do not form a tight cluster, and in fact different members of the microbiolite environment are seen to have higher proximity to other environment-types rather than to the microbiolite environment-type.

In trying to understand why the three microbiolite tend to naturally be pulled towards different environment-types and away from one another (low intra-group proximity), the metagenomic composition of each microbiolite

sample offers key information. The Highborn Cay (HC) microbiolite sample clustered near the slime cluster. Both the slime and HC microbiolite came from environments high in sulfur which seems to be the determining environmental factor that has driven their adaptation. The microbiolite has had millennia to adapt to the sulfur environment, and hence become specialized to it over time.

The slime samples came from a freshwater farmed hybrid bass tank that due to its containment and isolation forced the bass and its associated micribiota to quickly adapt to high sulfur conditions. Isolation, captivity and industrial type environments have been shown to bring rapid behavioral change in animals, it is then not a far stretch to suppose that microbiota living under similar conditions would also feel the pressure of their disrupted environment to evolve quickly to specialized adaptations to the environmental conditions. While different process have led the microbiolite and the slime to develop specialized adaptations to high sulfur conditions, the unsupervised RF shows them sharing functional subsystems that are best explained along lifestyle lines.

The Pozas Azules (PA) sample microbiolite came from a thrombolites which is a rock deposit that exhibits an internal clotted structure. It was found to be above a biofilm mat community [8]. This sample clusters close to the mat community environment which has a similar structure (layered saline porous deposit). The PA sample had marine traces which point date back to a time when the PA was connected to the ocean. Having marine traces in an isolated environment that adapted to living along mat communities makes this sample the most like the hypersaline mat community. Though the HC sample was also marine, it was not sharing habitat with a mat community, and though the Rio mesquite (RM) sample was also sharing space with bio film mat communities, it had much lower saline levels (freshwater).

Finally the RM microbiolite clusters somewhere between the PA and HC samples. It clusters closer to the PA, perhaps due to biogeographic functional adaptation. Yet in the MDS plot it is about equidistant from either of the other two microbiolite. This finding supports Desnues et al [8] findings regarding the fact that the 3 microbiolite samples were genetically unique and highly adapted to their environments. The fact that these are ancient organisms that have had a lot of time to evolve and adapt and that they are still around in minimally changed form makes them successful example of bio-adaptation and highlights the important role that environment plays in the survival of organisms. Each of the 3 microbiolites are seen to nat-

urally adapt along the lines of their environment (marine, freshwater) and along the lines of the other members of the environment (mat communities, marine life) so that environment or ecosystem composition are seen to be fundamental to the ways in which organisms adapt and evolve. Disruptions in either can be truly devastating.

PAM algorithm silhouettes of the unsupervised Random Forest showed $k = 3$ was the ideal number of clusters for the unsupervised RF (on this data). The slime and coral, despite their differences, were all classified in their own groups, while mat and microbiolites were both classified as single class. The unique clusters of slime and coral and the clustering of each microbiolite sample closer to organisms that share their environmental lifestyle points to the fact that there is a minimal set of core genes shared along lifestyle / environment lines. This set is minimal and while much interplay and similarities remain among different environments (as seen in the mat community / microbiolite samples). Nonetheless, these similarities can be partially detangled by a supervised RF which can find subsystems that uniquely define each environment.

In analyzing the clustering we see a two-pronged behavior. In the supervised RF each environment clustered together tightly, even though the each environment had high intra-environment diversity [27, 28]. On the other hand, the microbiolite environment, whose members had otherwise similar levels of stress and health and even relative peaks of key subsystems, clustered not with each other, but with the environment in which they had evolved. Thus we find good statistical support for the view that functional genomic analysis is effective due to environments exhibiting genetic uniqueness. Further subgroups of given environments would naturally cluster together due to microbes with similar lifestyles containing a core or minimal set of functional subsystems required for survival in similar conditions.

7.5 Summary

On the whole we find that both the supervised and unsupervised Random Forest are able to provide statistical support for biologically meaningful results. Supervised and unsupervised RF provides us with tools for different types of investigations, and are very good at yielding good results that speak of different underlying biological mechanisms. MDS plots of supervised RF derived proximity measures can tell us if the algorithm is acting as a good classifier if tight clustering results which discriminates well between distinct

groups. Such good classification means that the associated variable importance hierarchies are highly insightful. In such cases we can use a small subset of the functional subsystems in lieu of every subsystem when perfuming as the starting point while perfuming a PCA, LDA or Canonical Discriminate Analysis (CDA). Using a small subset is a direct result, as mentioned above, of the fact that a minimal functional subset is elemental to given environments.

8 Canonical Discriminant Analysis

Canonical Discriminant Analysis (CDA) is a dimension reduction statistical tool, similar to both principal component analysis (PCA) and linear discriminant analysis (LDA), used to separate data into categories. The most important aspect of the CDA is that it is used to separate data into g preassigned categories. Specifically, it finds *g-1* axes (uncorrelated linear functions) that best separate the data classes. The goal of the CDA is to understand which variables are responsible for differentiating between chosen groups.

8.1 How it Works

To perform a canonical discriminant analysis, we start with a data set containing n multivariate samples, along k variables, that has been divided into g groups. Canonical variables (or components), which are linear combinations of the explanatory variables, are then derived. These canonical components summarize the inter-class variance of the data. The theory is similar to that of constructing principal components in the PCA, except that the PCA is blind to classes. The PCA summarizes the total variance, across all samples, while the CDA looks only at variation *between* classes. When we have k variables and g groups, we can construct $\min(k,g-1)$ canonical components to explain the variance. These are constructed in much the same way as principal components, where the first canonical component is the linear combination of variables that has maximum multiple correlation with the classes[5]. The second canonical component is obtained by finding the linear combination *uncorrelated with the first canonical variable* that has the highest possible multiple correlation with the classes. This process is repeated until we have the maximum number of canonical components. In our research, we dealt with 27 variables (the functional groups) and a range of class numbers. Since we almost always had fewer classes than variables, we had *g-1* canonical components. In general, only two or three dimensions are needed to adequately separate distinct groupings, and so only the first few canonical components are pertinent.

The theory behind calculating the canonical components during CDA is identical to that of calculating principal components during PCA. The practical difference is in the covariance matrix. In PCA, the covariance matrix displays the variance between individual samples, while in CDA the covariance matrix contains variance between classes. The covariance matrix

must be full rank in order to complete the analysis, and as such requires at least as many individual samples as variables.

Ensuring the full rank of the covariance matrix can require removing some variables from the analysis. In our research, we started out with 27 variables and 248 distinct microbial metagenomes. This are good values when computing analyses over the entire data set, since there is room to remove outliers and highly unique samples. The problem comes when working with smaller subsets of the data. When looking at small enough subsets of data, it is necessary to reduce the number of variables used in the analysis, but care must be taken. While it will not compromise the analysis to remove unimportant data, how do we know which data is important before running the analysis? There are several tools at our disposal for deciding which are the most important.

One method is informed selection. For example, if a researcher was specifically interested in how nutrient metabolism varies across environments, she could choose to look only at variables pertinent to nutrient metabolism. Choosing variables like this introduces an inherent bias into any analysis that is to be performed, and in general it is to be avoided. An analysis done by openly selecting variables can only speak as to the relative importance of the chosen variables, and the divisions along those lines. This is because important mitigating factors may have been eliminated. In an overall analysis, the nutrient metabolism levels may have such small differences as to get lost in the noise. Then our researcher friend has not modeled the overall trends, but some of the noise.

Another method of measuring variable importance is the correlation matrix, which can be computed over any number of variables. This is a rough estimate of what explanatory variables are linearly related to the response variables. The variables with the largest correlation values (positive or negative) are the ones that are most likely important toward determining class. It proves useful for estimation, but is not infallible, and has scaling problems. One of the major drawbacks is that it measures only *linear* correlation, and in real-world data other types of relationships abound. Using the correlation matrix is an acceptable method for eliminating a small number of variables from an analysis, but for further whittling it is not sound.

It is a good idea to combine statistical techniques, as they strengthen the analysis. Two of the stronger variable importance measures we worked with are the Random Forest variable importance measures described in Section 7.1. These determine the ability of each variable to classify data, us-

ing different measures of classification skill. Performing a CDA using only the most discretionary variables yields a more significant analysis, especially when the data has many more variables than samples.

8.2 Visualization

An advantage of the CDA is the visualization aspect. It is trivial to plot the analysis, since it centers around transforming the coordinate system. All we need to do plot the canonical scores of the data points along new axes (the canonical components). See Figure 27 for a plot of CDA performed on marine metagenomic data. In this CDA we used 46 samples, 26 function groups (removing the 'miscellaneous' group) as explanatory variables, and classified our data into four groups, based on the types of marine environment they came from. This subset of the data was collected in the northern hemisphere, which homogenizes the data and reduces the possibility of confounding variables.

The influence of each variable along the canonical axes is visualized through vectors. CDA centers on the construction of canonical components to explain the variance between classes. The amount of the inter-class variance that is explained by each component is indicated in parentheses along each axis. For a data set with variables $\{v_1, v_2, \ldots, v_k\}$, these canonical components have the form

$$Can1 = \hat{a}_1 v_1 + \hat{a}_2 v_2 + \cdots + \hat{a}_k v_k$$

$$Can2 = \hat{b}_1 v_1 + \hat{b}_2 v_2 + \cdots + \hat{b}_k v_k.$$

Figure 26 shows the projections of variable vectors v_1 and v_2, onto the canonical component axes Can1 and Can2. The projections of the variables maintain the relationship between their coefficient variables. That is,

$$\frac{a_i}{b_i} = \frac{\hat{a}_i}{\hat{b}_i} \qquad \text{and} \qquad \frac{a_i}{a_j} = \frac{\hat{a}_i}{\hat{a}_j}.$$

Graphically, the vectors can be rescaled to obtain the clearest visualization, but they must maintain the ratio of their lengths. Qualitatively, the greater the magnitude of a canonical variable vector, the greater its importance to separation of groups.

Looking at the plot, it is easy to see that removing variables will clarify the graphical interpretation of a CDA, and also that the choice of these

variables is important. We can see qualitatively that the 'Carbohydrates,' 'Cell.Wall,' 'Cofactor,' and 'Potassium' vectors are small, which indicates that those variables are explaining the least of the inter-class variance. This implies that the proportion of hits in these categories is similar across the classes, and so these are good candidates for removal in a second analysis. It is important to remember, however, that this graphical view shows only the first two canonical components, and they may have larger magnitude in higher dimensions.

Each sample is plotted according to its canonical scores. Let x be a sample, such that $x = (x_1, x_2, ..., x_k)$ from a data set whose first canonical components are C_1 and C_2, such that the coefficients of C_1 are $\{a_1, a_2, ..., a_k\}$ and those of C_2 are $\{b_1, b_2, ..., b_k\}$. Then we compute

$$xC = x \begin{pmatrix} | & | \\ C_1 & C_2 \\ | & | \end{pmatrix} = (x_1 x_2 \cdots x_k) \begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_k & b_k \end{pmatrix} = \begin{pmatrix} C_1(x) \\ C_2(x) \end{pmatrix}.$$

The canonical scores of a sample x are $C_1(x), C_2(x)$, which describe its position in the 2-dimensional space defined by the first two canonical components. We also plot the mean scores, represented by $+$, and confidence intervals of the means, represented with circles. Our research centered on the strength of clusters, so we focused on the spread of individual samples about the mean. For other analyses, it is valid to plot only the means and confidence intervals.

Using a variable importance plot, calculated using a Random Forest, 12 most important variables were selected to further investigate the metabolic processes driving class divisions (Figure 28). We also removed the confidence intervals, to simplify the representation. This plot is easier to comprehend, and more legible. Note that the environments are less clearly separated in this second image. This analysis, and the accompanying plot, are more significant. The canonical discriminant analysis is prone to overfitting, because it is very good at finding small differences. Our metagenomic data, with its small sample sizes, is particularly vulnerable to artificial separations. The smallest vectors are often statistically insignificant, and so removing them gives more robust results. These two plots (Figures 27 and 28) insinuate that estuary samples are very similar to open ocean and coastal water samples, which themselves are closer than initially apparent. Only variables with small impact on separation were removed, and the environments clustered

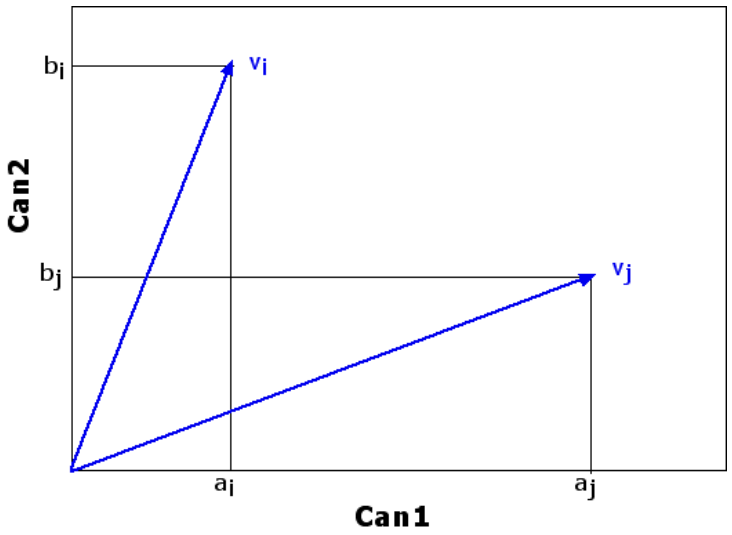


Figure 26: Projection of variable vectors onto the first two canonical components.

more tightly. This implies that there are only small differences between the two.

8.3 Prediction and Error Estimation

It is important to estimate the error of any statistical model. The CDA can separate preassigned classes easily when it knows what they are, but a misclassification error helps establish if separations exist between the classes themselves, or only the particular samples used. In general we estimate error by testing a model's skill at classifying a set of known samples. There were no available error estimation functions for canonical discriminant analyses in R, so we devised our own. Two proved especially useful[7].

The first, `mahalErrorEst.fun`, operates on the leave-one-out principal. For a data set with n samples, it computes n canonical discriminant analyses, each time leaving out a different sample. Using the canonical components of each analysis, the canonical scores are computed for the left out sample, and this score then determines how the sample is classified.

The choice of group was determined by the minimal Mahalanobis distance[6].

[7]The code for these can be found in A.7.

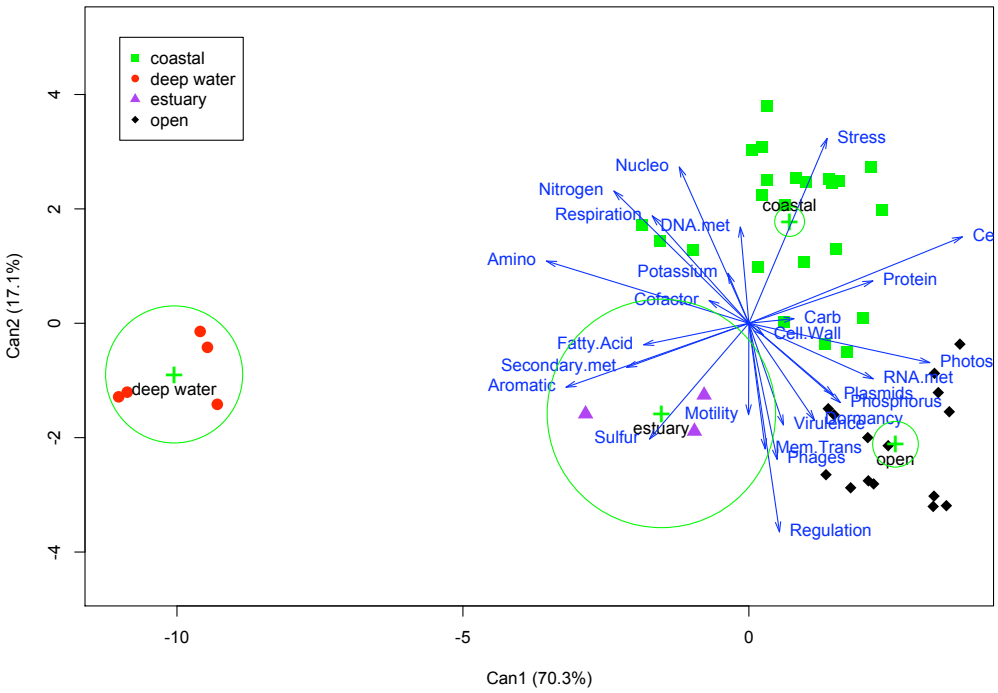


Figure 27: Plot of a CDA of marine microbial metagenomic samples collected in the northern hemisphere. This was calculated with 46 samples, 26 variables, and 4 environments.

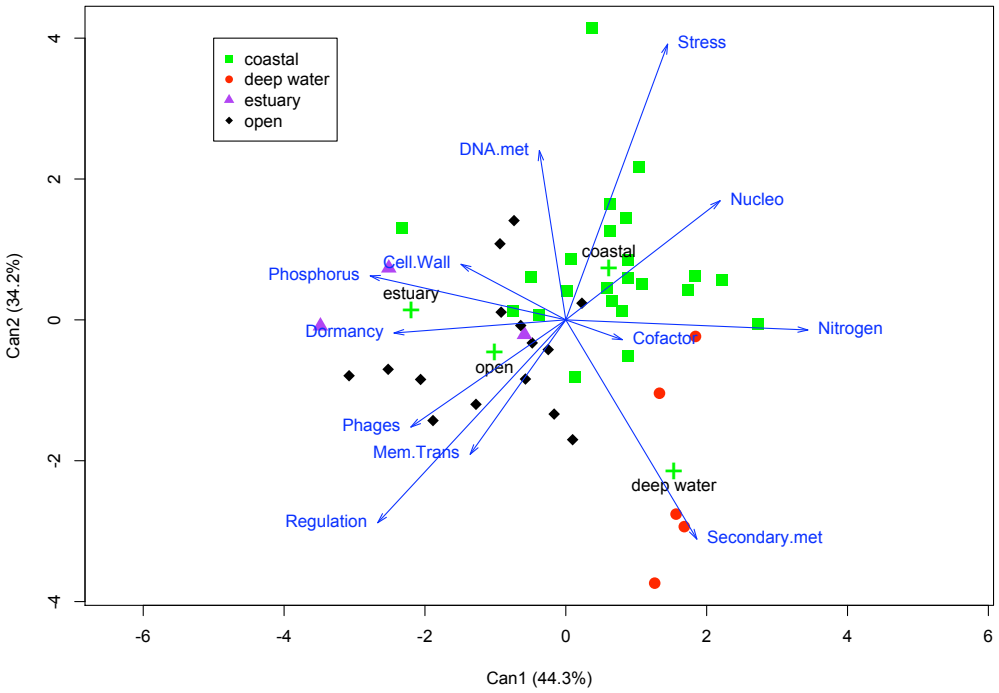


Figure 28: Plot of a CDA of the same samples as Figure 27. This CDA was calculated using only the 12 most important variables, as indicated by a Random Forest variable analysis.

The Mahalanobis measure is a scale-invariant distance measure based on correlation. The distance of a multivariate vector $x = (x_1, x_2, ..., x_n)$ from a group with mean $\mu = (\mu_1, \mu_2, ..., \mu_n)$ and covariance matrix S is defined as:

$$D_M(x) = \sqrt{(x-\mu)S^{-1}(x-\mu)^T}$$

More intuitively, consider the ellipsoid that best represents the group's probability density. The Mahalanobis distance is simply the distance of the sample point from the center of mass, divided by the spread (width of the ellipsoid) in the direction of the sample vector.

`mahalErrorEst.fun` goes through the data in this manner, computing a CDA for all but one of the data points, and predicting the class of the leftover sample. It returns both a misclassification rate and a confusion matrix[8], a record of how each sample was classified. From the confusion matrix we can see how the misclassifications occurred (was a human sample classified as a chicken or a mouse?), from which we can determine not only the percentage of samples which were misclassified (error estimate), but how egregious the errors were. For example, it would be more egregious to mistake a human metagenome for one from deep water, rather than to mistake a deep water metagenome for one from the open ocean.

The second test function, `cdaErrorEst.fun`, combines canonical and linear discriminant analyses. The function first randomly divides the data into *in-bag* and *out-of-bag (OOB)* data sets. By default, `cdaErrorEst.fun` chooses 20% of the data in each class as OOB. It then performs a CDA on the bagged data, looks at the canonical scores of each sample, and performs an LDA on the canonical scores. Canonical scores are then calculated for the OOB samples, and LDA is used to predict their classes. Like `mahalErrorEst.fun`, `cdaErrorEst.fun` returns a confusion matrix along with a misclassification rate.

8.4 Considerations

Canonical discriminant analysis is far from perfect. There are several things to take into consideration when using this tool. The most obvious of these is the presence of bias. We group our data before CDA, which means that the analysis is not blind. The canonical components are the linear combinations that best separate predetermined groups, not blind data, so CDA is

[8]For a detailed explanation, see Figure 10 in Section 5.1.2.

very prone to overfitting, or finding artificial separations in data (often from modeling noise). The larger a data set, and the more similar class sizes are within that data set, the smaller the risk of overfitting.

A major constraint on canonical discriminant analysis is sample size. To use CDA, a data set must have at least as many samples as explanatory variables to ensure the full rank of the matrices involved in the computations. Having bigger sample sizes is always better, and will improve the validity of the analysis. The size of the class groupings must also be taken into consideration. Given a data set with many more samples than variables, a class with only few samples will reduce the significance of the analysis. Classification is done by comparing a sample to existing class means. Small samples have large confidence intervals, which means that their means are not significant. That is not to say that CDA can say nothing about small samples, but the results must be taken with a grain of salt.

The sample size considerations are particularly relevant to metagenomic analysis, due to the nature of the data. Metagenomic data is divided into subsystems, currently 27 at the broadest level. The data available to us was the public metagenome database of about 300 samples. The data set is sparse, with samples from a limited range geographically, and of vastly ranging sample sizes. For example, there are dozens of marine samples, ranging around the world but still not globally representative. There is only a single termite gut sample. Trying to significantly differentiate between these two classes with the CDA would be futile.

As with any statistical method, care must be taken when interpreting the results. There is always the risk of confounding variables tainting the results, recalling that correlation does not mean causation. There are ways of decreasing the risk of confounding variables, but the chance will never be totally eliminated.

8.5 Conclusion

The canonical discriminant analysis is an excellent technique for exploring metagenomic data, and more generally for any type of classification analysis. The CDA is really good at separating preexisting classes, and gleaning which variables make the difference. The ease and extent of visual representation are also extremely helpful, it is very clear what the results of the analysis are. The CDA can predict the class of a new sample, which is useful not only for understanding new samples but also for approximating the strength

of the model. Canonical discriminant analyses can be combined with many other statistical methods in order to improve their significance.

As with any statistical method, there are constraints and drawbacks to CDA. It is prone to overfitting, biased toward groupings, and requires samples of a certain size. There is also the risk of misinterpretation and confounding variables. However, if this analysis is performed under a watchful eye, and care is taken, it is extremely powerful and informative. As the database of metagenomic data continues to grow, and its study alongside, the significance of this technique will only increase.

9 Example

Now that several statistical techniques have been discussed, we will work through them all, using the data set "`all.data`". This set (`all.data`) consists of 203 samples, the proportion of their identified sequences pertaining to each of 27 functional groups[9], and their classification into 6 environmental groups. Removed from this data set are classes with extremely small sample sizes, samples that were contaminated during sequencing, and a few samples externally determined to be outliers. The environmental groups are human[10], hypersaline, mat community, spring[11], terrestrial[12], and water[13]. This section will display the strengths and weaknesses of these techniques with regard to `all.data`, and along the way we will discuss our findings.

9.1 Motivation

To date, there is only one published study on the total public database of metagenomic data, *Functional Metagenomic Profiling of Nine Biomes* by Dinsdale et al[9]. Their study was conducted when there were only 45 publicly available metagenome samples. Continuing in their footsteps, we performed analyzed all publicly available microbial metagenomic data, today consisting of over 200 samples.

The findings in the Dinsdale paper indicated that there are metagenomic differences between environments, and in part this study aimed to substantiate or refute their findings. We wanted to discover which functional groups, if any, can be used to differentiate between environments.

[9] Amino Acids and Derivatives; Carbohydrates; Cell Division and Cell Cycle; Cell Wall and Capsule; Cofactors, Vitamins, Prosthetic Groups, and Pigments; DNA Metabolism; Dormancy and Sporulation; Fatty Acids, Lipids, and Isoprenoids; Membrane Transport; Metabolism of Aromatic Compounds; Miscellaneous; Motility and Chemotaxis; Nitrogen Metabolism; Nucleosides and Nucleotides; Phages, Prophages, and Transposable Elements; Phosphorus Metabolism; Photosynthesis; Plasmids; Potassium Metabolism; Protein Metabolism; Regulation and Cell Signaling; Respiration; RNA Metabolism; Secondary Metabolism; Stress Response; Sulfur Metabolism; Virulence.

[10] Human gut samples, taken from separate studies of Americans and Japanese.

[11] Samples from hot springs

[12] Non-human, land-based animal gut samples.

[13] Samples from an assortment of marine and freshwater environments

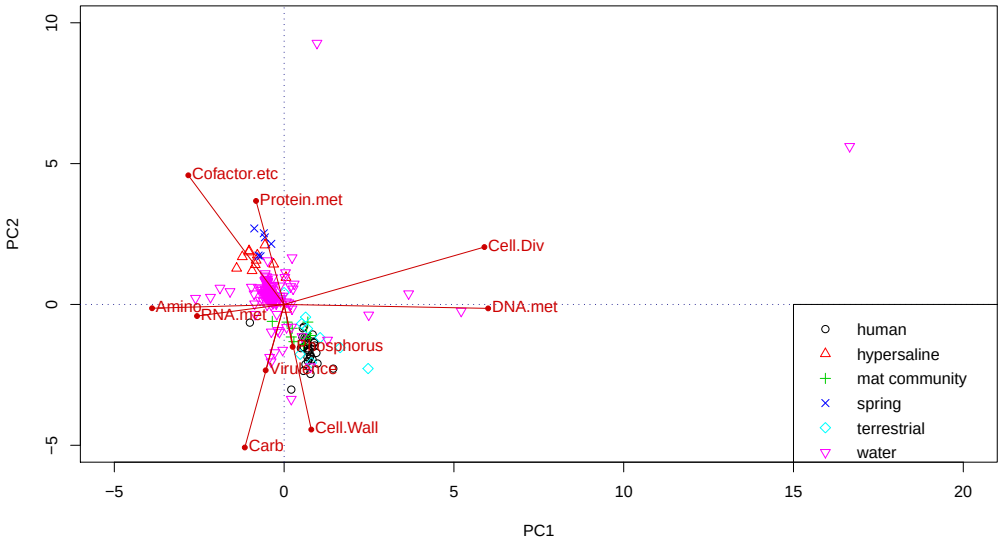


Figure 29: Plot of scaled PCA over 10 functional groups with highest variance.

9.2 PCA

We begin by performing a principal component analysis on `all.data`. We plot data along the first two principal components, coloring the resultant plot by environmental grouping. First we look at the scaled and unscaled variance analyses over all 27 variables, but these give rather messy and confusing results. Since the PCA acts on variance, we perform the analysis again, this time using the variables with the largest spread. There was a clean break after the top 10 variances, so the PCA was performed over those variables. The resultant plot can be seen in Figure 29. We see two water samples that are far away from the rest, and seem to be forcing the other samples into one large cluster. Note that these samples may be outliers. Figure 30 shows a zoomed-in view of the central data cluster. The environmental classes are separating, although the water class overwhelms the eye.

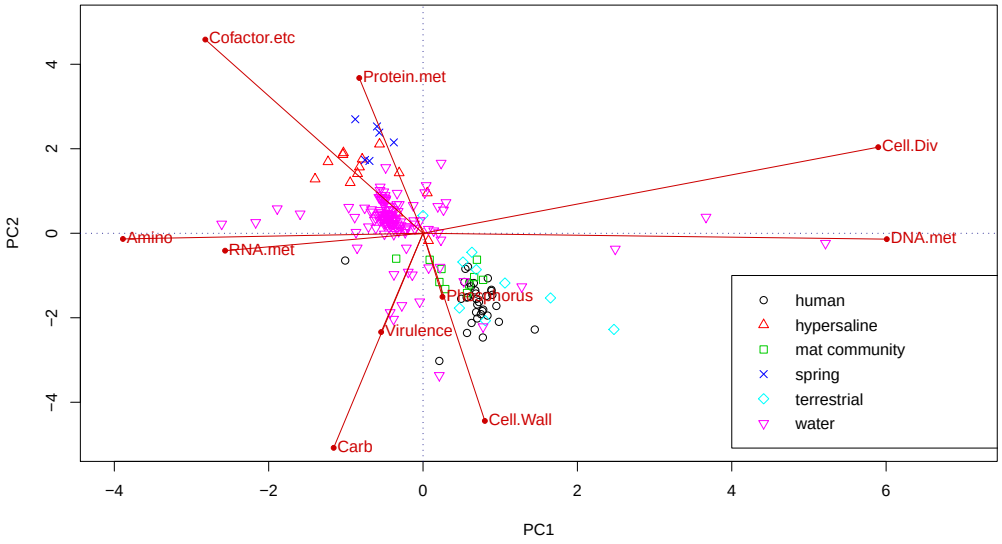


Figure 30: Zoomed view of central cluster from Figure 29

Cluster	Environmental data within Cluster
1	All but 1 hypersaline; water samples
2	1 water sample (probable outlier)
3	All spring samples; 1 human (possible outlier); water samples
4	3 water samples (possible outliers or contaminated data)
5	1 water sample (not shown, probable outlier)
6	1 water sample (not shown, probable outlier)
7	All mat community samples; 5 human; 3 terrestrial; water
8	1 water sample (possible outlier)
9	Remaining human samples; 2 terrestrial; water

Table 8: Breakdown of environmental data present in each of the 9-mean clusters.

9.3 K-means

The sum of squares plot for our data set, Figure 31, has no clear 'elbow', which would indicate the best number of clusters. In our experience with metagenomic data, this implies the existence of outliers. To understand the clusters that K-means found, we plot the PCA but include the clustering assigned by K-means. When the data is graphed and colored based on clustering for $K = 3$, the apparent elbow of the graph, the data clusters into 2 single points, and a central mass. This is not a helpful analysis.

A second way of looking for good clusters is the silhouette plot. When we make silhouette plots of our data (Figure 32) and plot their average width (Figure 33), we again are led to the conclusion that $K = 3$ should be ideal. We have already decided that it is not, and so look further. In Figure 33, there are peaks at 5 and 8. Looking at these as values for K, we find that 5 seems to indicate outliers, but $K = 8$ seems to form separations along groupings.

Looking at the plot for $K = 9$, however, a clearer picture appears. In Figure 34, several clusters contain only a single point, which indicates that they are probably outliers. Two of these points (both water samples) have been cropped out of Figure 34, for ease of visualization. All of the points put into their own categories are, in fact, water samples. Close inspection reveals something about the clusters, detailed in Table 8. Note that, while imperfect, there seems to be some separation along known environmental categories, which leads us to continue our investigation.

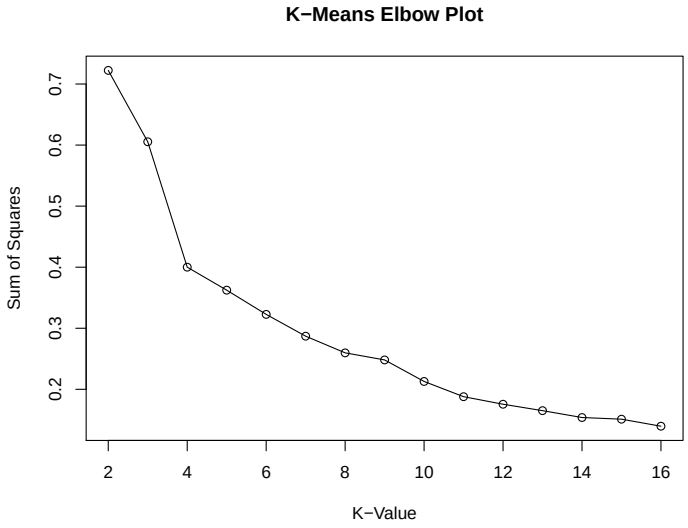


Figure 31: Sum of squared error versus K values, ranging 2:16, from K-means analysis of `all.data`

9.4 LDA

We performed a linear discriminant analysis (LDA) on `all.data`, with environmental groupings as the response. First we attempt the analysis over all 27 functional group variables. Plotting the data along the first two linear discriminant functions, Figure 35, it appears that the LDA succeeds in separating our data along environmental factors.

There is a lot of separation between the environmental groups. There are a few overlaps that may indicate outliers, specifically the human and terrestrial samples that mix in with water. The mat community sample that mixes in with hypersaline samples comes from a hypersaline environment, which may explain its proximity. There is a fair amount of mixing between spring, hypersaline, and water samples. This is not altogether unexpected, as they are all aquatic environments.

Using the leave-one-out error estimation technique for this LDA, this analysis has an error rate of approximately .167, that is, we expect to misclassify about 17% of samples with this technique.

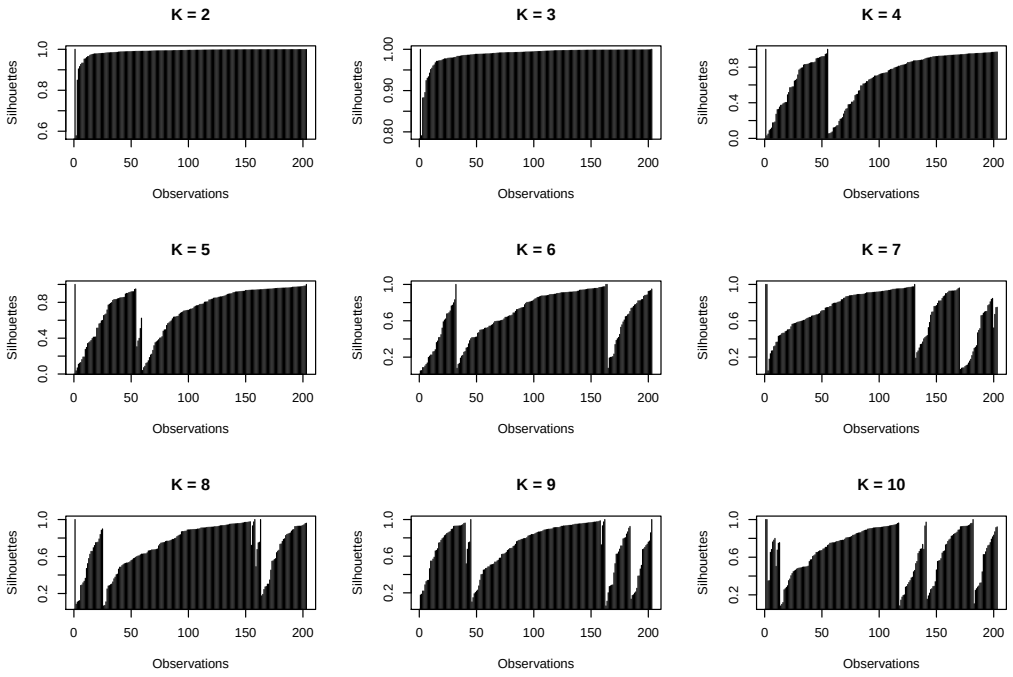


Figure 32: Silhouette plots from K-means analysis of `all.data`, K ranging 2:10

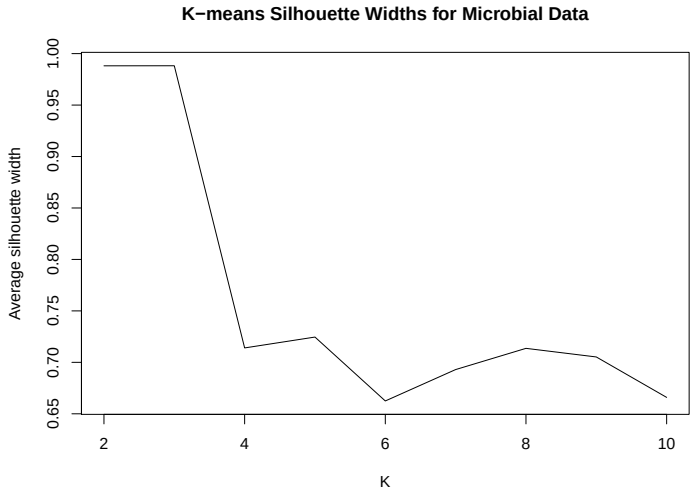


Figure 33: Plot of average silhouette widths for K-means analysis of `all.data`. The silhouette plots are in Figure 32.

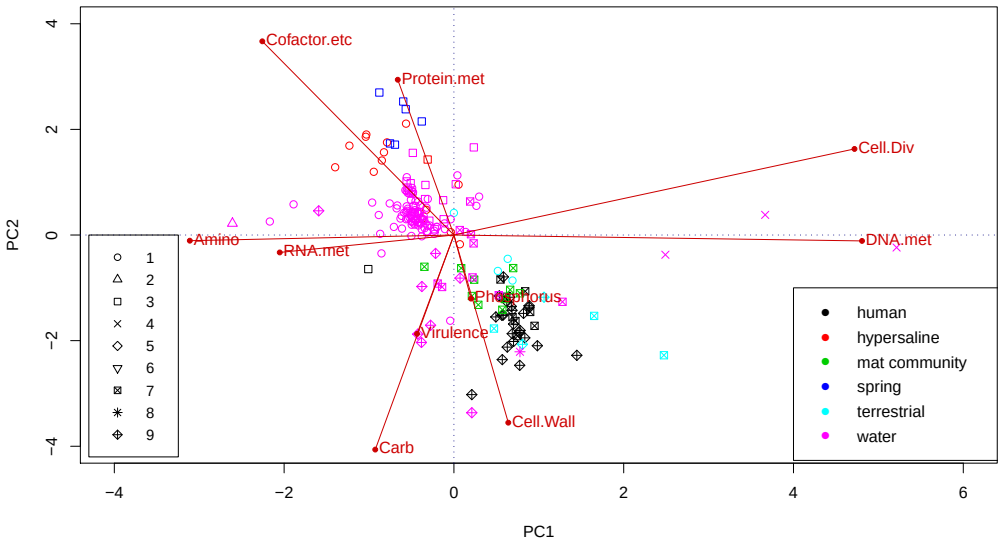


Figure 34: Central cluster of scaled PCA with 9-means clusters indicated.

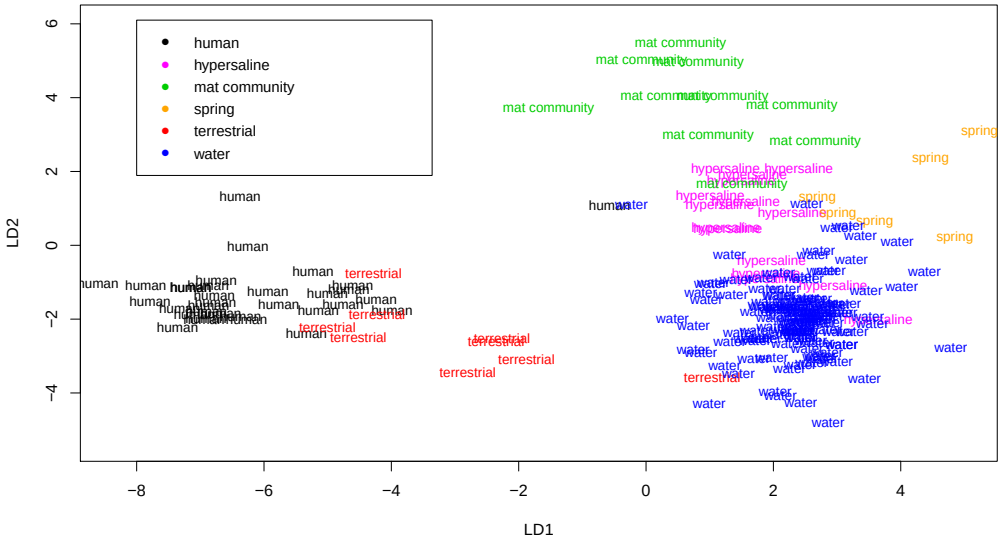


Figure 35: Plot of the first two dimensions of the linear discriminant analysis of all `all.data` over 27 variables, with 6 environmental groupings.

9.5 Trees

We also performed tree analyses on the data. The fully grown tree over all the data is Figure 36. To figure out the best tree, we ran a series of cross-validation experiments. Figure 37 shows the mean and standard error of misclassification over 100 trials at different tree sizes, and Table 9 shows the average deviance for the series. From these we can determine that 9 leaves is the best, and plot the pruned tree seen in Figure 38.

While this tree is not as definitive as we might like, there is information to be gleaned from it. At first glance we see that water is a leaf along several branches. This is less than ideal, and seems to indicate the wide range of values among water samples. This spread is partly due to the presence of both freshwater and marine saline samples in the 'water' class, as well as the large sample size. However, note that the bacteria samples from terrestrial guts (human as well as non-human) are all on the left of the first node, having a very low proportion of photosynthesis coding. That there are low levels of coding for photosynthesis here makes biological sense. Bacteria in eukaryotic guts have no access to sunlight, and so have nothing to photosynthesize. The human and non-human samples are split under membrane transport. On the other branch, with higher proportions of proteins dedicated to photosynthesis, we see that there are differences between these groups as well, though they seem a bit more muddled. This is biologically sensible, since hot springs, hypersaline, and water samples all come from aquatic environments, and the mat communities (though very different from the microbes that float in water), also live in pools of water of varying salinity. Note that the water grouping includes samples from a wide variety of marine environments, which accounts for the large variance. Again, the water samples mix in through the data, but the tree appears to differentiate between environments. Of course, these are not perfect leaves, and there is some overlap between the groups, but the majority of the data splits along environmental lines, though the vast water group mixes in.

9.6 Random Forest

As noted in Section 6, trees are highly unstable. In our research we have had more robust, and repeatable, results from Random Forest analyses, so that is the next technique to use.

With unsupervised Random Forests, we want to see if our groupings

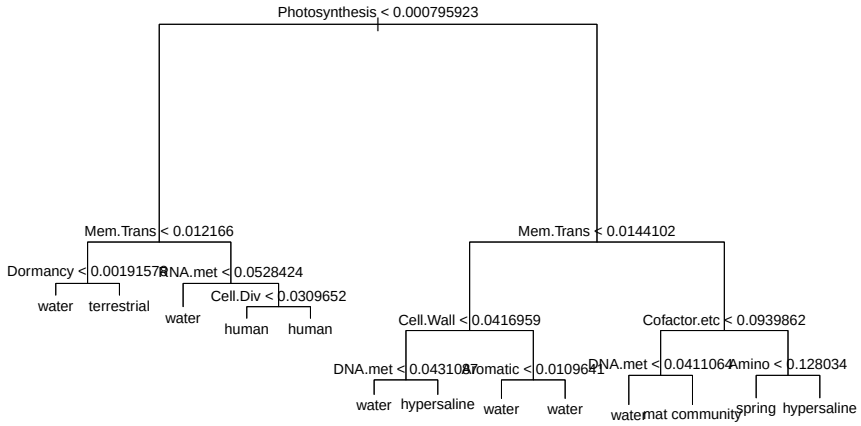


Figure 36: Full tree grown from `all.data` using all available variables.

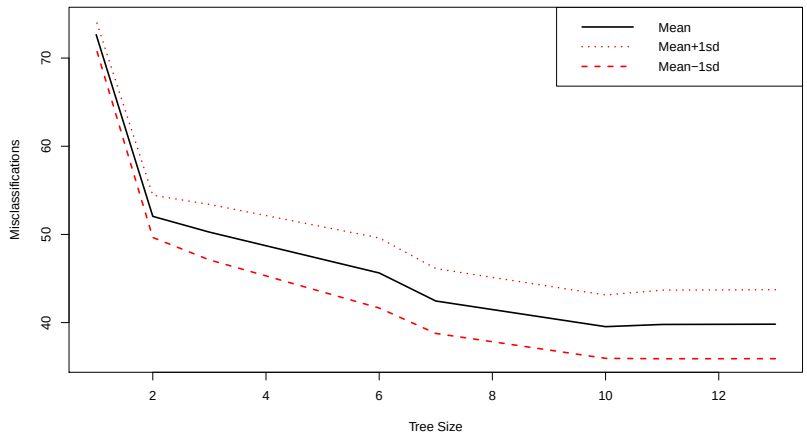


Figure 37: Plot of mean and standard errors of misclassifications for different sized trees over `all.data`.

Tree Size	Average CV Deviance
1	72.65
3	52.05
4	50.26
6	45.63
9	42.45
13	39.54
15	39.82

Table 9: Tree size and average deviance from a series of tree cross-validation experiments over `all.data`

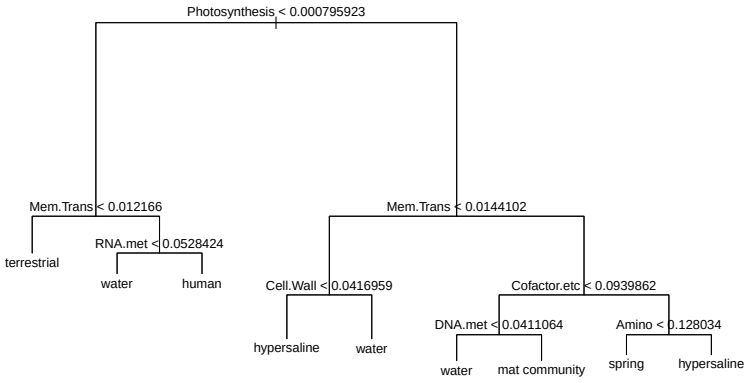


Figure 38: Tree of `all.data` pruned to 9 leaves, as indicated by a series of cross-validation experiments.

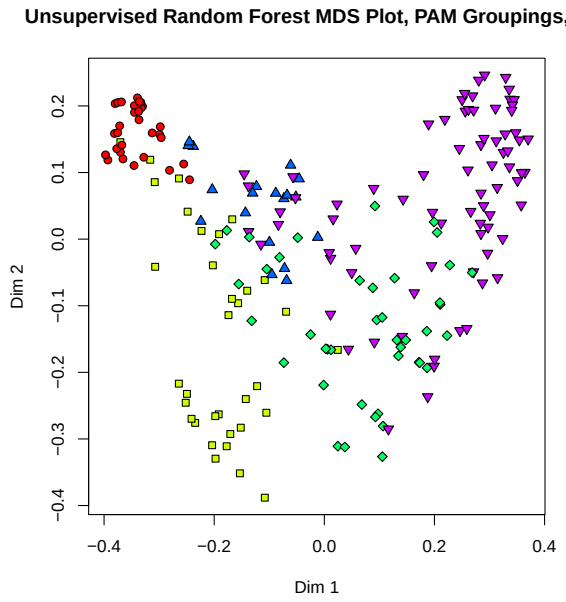
match up with clear clustering in unmarked data. Similar to K-Means, PAM silhouettes help determine the best number of clusters. In the plot of PAM silhouette widths (Figure 40), we see that $K = 5$ is the best point. Constructing MDS plots using PAM coloring (Figure 39a) and known grouping coloring (Figure 39b), reveals its accuracy. We see that PAM seems to have picked a cluster of human and terrestrial samples (circles, upper left), a cluster of mostly hypersaline data (triangles), a cluster of mat community, spring, and various extras (squares), and split the rest of the water samples into two clusters (inverted triangles and diamonds). The PAM coloring is not a perfect match to our outside clustering, but it is similar.

We use supervised Random Forests to calculate the mean decrease accuracy (plotted in Figure 41a) and mean decrease Gini (Figure 41b) of each variable. These graphs were discussed in Section 7.1. The variables with the highest mean decrease accuracy are those judged most important for accurate classification by the Random Forest, and those with highest mean decrease Gini are those that have the most impact on node purity. It is clear that the functional group 'Phages' is one of the most important to separating `all.data`, as it is at the very top of both plots. Looking at the clean breaks, we take the top variables from each and combine them to form a set of the most important variables. This is an excellent set of variables to create trees from, and a good set to start from when using any classification or clustering technique.

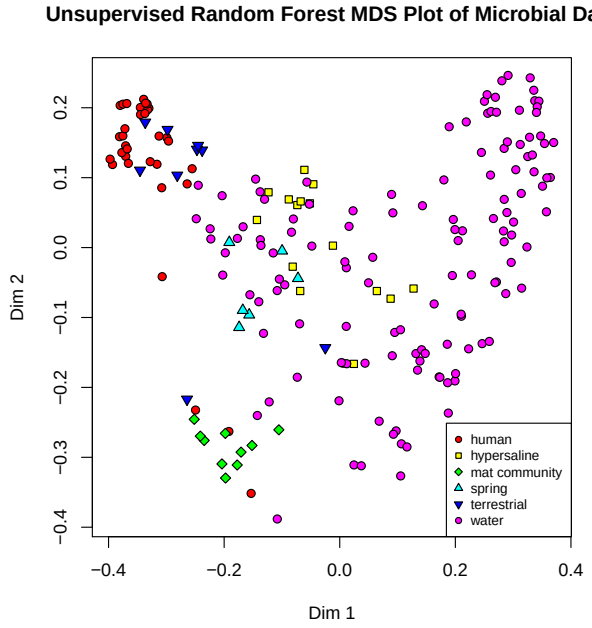
Constructing a tree from this set of variables, as shown in Figure 42, we see that it has similar leaves to the tree constructed using all the variables, which is in Figure 36. The difference is the purity of the nodes. In the Random Forest assisted tree, the leaves are much more definitive. Originally, in the node that leads to the human class, there was approximately 40% chance of being human, but in the new tree, we have 85% probability of being human and less than 10% chance for anything else. Using a smaller set of variables also increases the stability of the trees.

9.7 CDA

The next step was to perform a canonical discriminant analysis on `all.data`. We first did a CDA over all the variables, and looked at the vectors that seemed most important from that plot. We also calculated error, using an LDA on canonical scores. Combining the information from all these analyses, we settle on a set of 8 functional hierarchies:



(a) PAM cluster coloring.



(b) Known class coloring.

Figure 39: MDS plots of unsupervised Random Forest analysis on `all.data`.

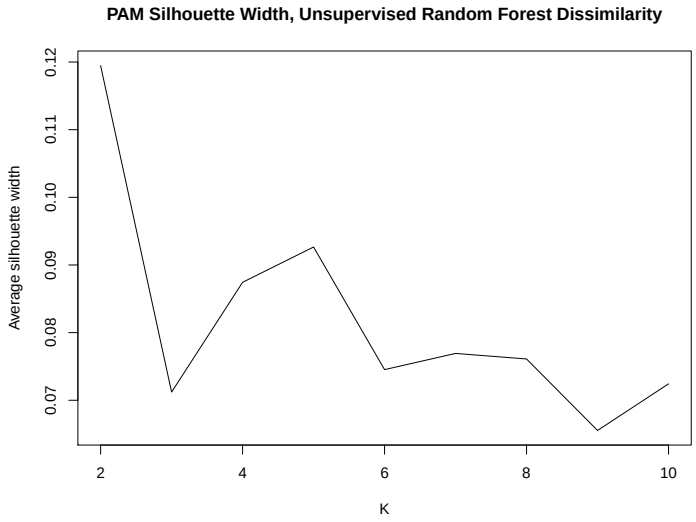
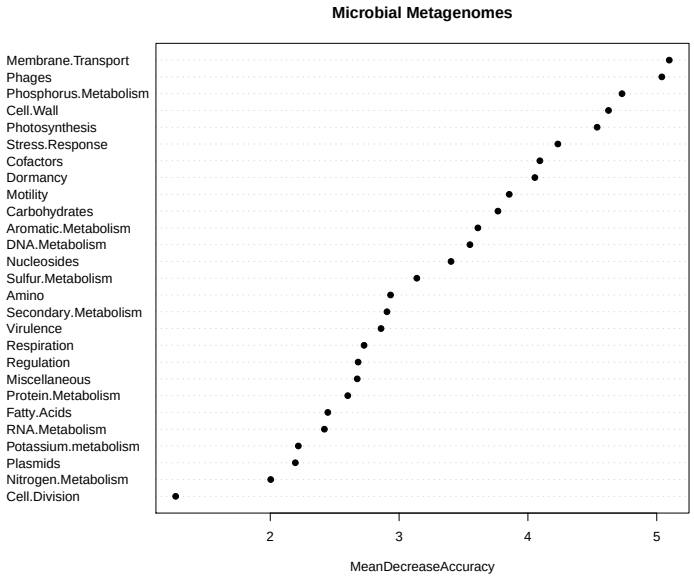


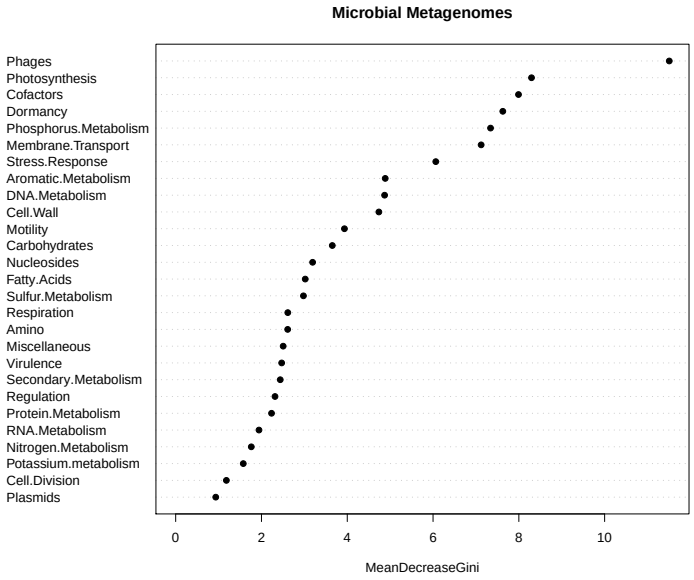
Figure 40: PAM silhouette widths for `all.data`.

- Cell Wall and Capsule
- Cofactors, Vitamins, Prosthetic Groups, Pigments
- Dormancy and Sporulation
- Membrane Transport
- Phages, Prophages, Transposable Elements
- Photosynthesis
- Respiration
- Stress Response

We performed our analysis using those variables, and plotted the results (Figure 43). This plot contains a lot of information. First, we see that samples from organisms split from water samples along the first canonical component, with the primitive mat communities in the middle. The first canonical component also separates the hot spring samples from the rest of the water. The splitting along the second canonical component separates human and non-human samples, as well as hypersaline and hot spring samples from the rest of the water. There is some overlap, but overall we see good separation into the general environmental categories. These findings are similar to those of Dinsdale et al. The four-fold increase of the metagenomic database has further defined environmental clustering.



(a) Mean decrease accuracy.



(b) Mean decrease Gini.

Figure 41: Variable importance plots generated using supervised Random Forest analysis on `all.data`.

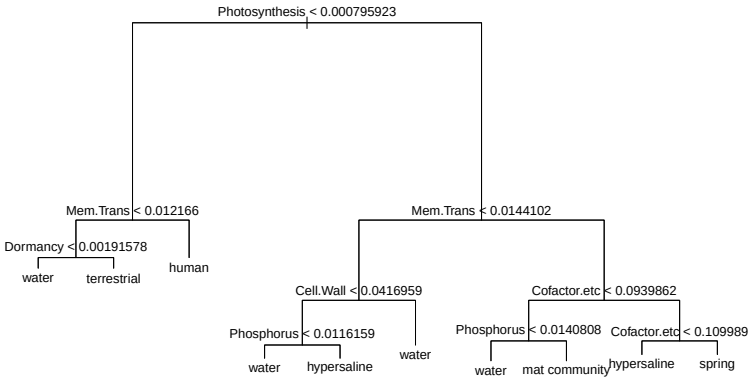


Figure 42: Tree grown with the top 8 variables indicated by the variable importance plots.

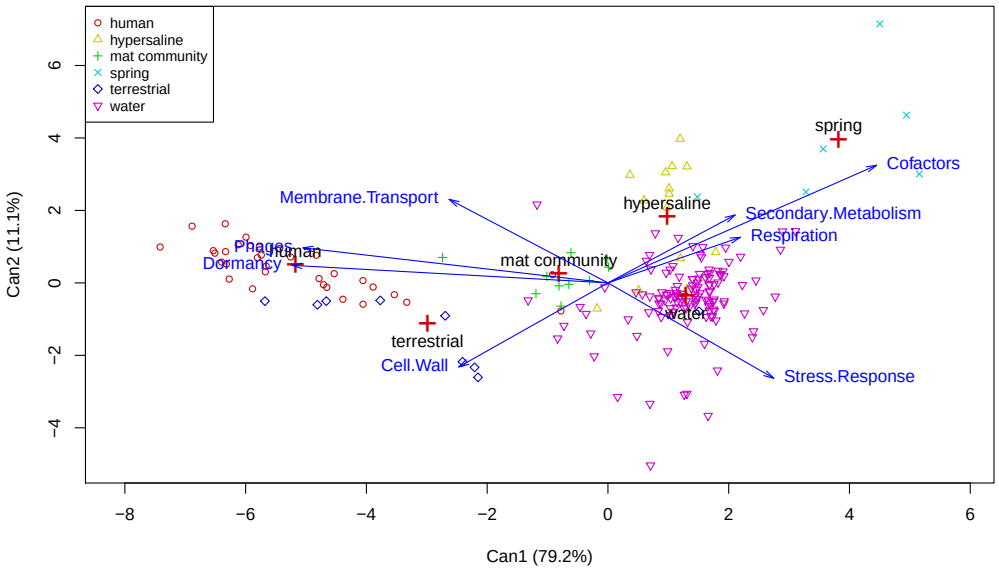


Figure 43: Plot of first two canonical components from CDA of `all.data`, computed over 8 variables.

	human	hyper-saline	mat community	spring	terrestrial	water	class error
human	6.025	0.000	0.490	0.000	0.390	0.095	0.139286
hypersaline	0.000	1.760	0.000	0.000	0.000	1.240	0.413333
mat community	0.000	0.000	1.990	0.000	0.000	0.010	0.005000
spring	0.000	0.135	0.000	0.865	0.000	0.000	0.135000
terrestrial	0.970	0.000	0.000	0.000	0.805	0.225	0.597500
water	0.105	0.775	0.445	0.145	0.095	24.435	0.060192

Table 10: Confusion Matrix for CDA

9.8 Strength of Analysis

Using the LDA on CDA error estimation technique (the second discussed in Section 8.3), this model has an estimated 12.5% misclassification error, which is very low. This was calculated by running the algorithm 200 times and averaging the misclassifications. By doing this we improve our confidence level. The confusion matrix of this CDA is Table 10. Note that the row sums are the number of *OOB* samples for each class, in this case 20% of the original class sizes. Looking at the last column, "class error," we can see the accuracy of our CDA as a predictor for each class, and averaging all these scores (weighted by number of samples) gives us .125, our overall error estimate. Looking more closely at the misclassifications, we see that most of them are benign. We see water, hypersaline, and spring mixing a little, which is not surprising since they are just different types of aquatic habitats. We are not surprised by the mixing of terrestrial organisms and humans either, since humans are terrestrial, but the mixups of human samples with mat community and water samples are a little disturbing. Taking into account our sample sizes and the possibility for outliers, the error is small. Our model is not perfect, but it is a very good predictor.

A Appendix: R Code

This appendix includes the R[21] code used to generate the graphics[14] in Section 9. We reference several packages that we installed in R, more information is available about each package (after it has been installed once) using the R commands:

```
> library(package.name)
> help(package.name)
```

For this overview, we'll call our data frame "`data`". We read the data frame in from a tab-delimited text file "data.txt" that contains all the information we need to perform the analyses discussed in this section.

```
> data<-read.table("data.txt", header=T, sep="\t")
> data
```

[1]	[2]	[3]	...	[29]	[30]
name	env	Amino	...	Virulence	Genome.ID
coa1143	water	16125	...	4742	4441143.3
coa1144	water	17110	...	5226	4441144.3
⋮	⋮	⋮		⋮	⋮
gut452.4	human	14826	...	5688	4440452.4

Note that columns `[3:29]` are the 27 variables (functional groupings). In order to compare our data samples, we normalized based on number of total identified hits, that is, we turned the number of hits in each functional group into a proportion, using a simple for-loop. Recall that we have n samples. In generating the graphs in Section 9, we used $n = 203$.

```
> cbind(env=data$env, name=data$name, data[3:29]/rowSums(data[3:29]), Genome.ID=data$Genome.I
```

Now `all.data` contains n rows and 30 columns, and the data in columns `[3:29]` are the proportions of recognized hits that fall under that hierarchy.

A.1 Utilities

Here is a function that creates a formula from a data set `y`, predictors, and response variables.

[14]Note: This is the code used to generate original analyses and graphs. Since then many of the figures have been remade in other programs, whose graphical abilities are different from those in R.

```
> createFormula.fun <- function(y, response, predictors){
   ifelse(is.character(predictors[1]), for(i in 1:length(predictors)){
      (1:length(names(y)))[names(y)==paste(predictors[i],sep="")]
      ->predictors[i]}
   ,predictors<-predictors);
predictors.names<-paste(names(y)[predictors], sep="");
ifelse(is.character(response),
   (1:length(names(y)))[names(y)==paste(response,sep="")]
   ->response,response<-response);
response.name<-paste(names(y)[response], sep="");
fmla<-as.formula(paste("cbind(",paste(predictors.names,collapse=","),")
   ","~ ",paste(response.name,collapse=""),collapse=""));
return(fmla) }
```

If response is given as a character, `getSingleResponse.fun` gets the numeric value:

```
> getSingleResponse.fun <- function(y, response){
if (is.character(response)){
   for (i in 1:length(names(y))){
      if (names(y)[i] == response) response = i}}
return(response)}
```

This is code for breaking data into *in-bag* and *out-of-bag* sets. The function takes `outPercent` as the percentage of data that should be out of bag, rounds to the nearest whole, then splits the data up and returns `bag` and `oob`.

```
> bagData.fun <- function(data, outPercent, response=c(1)){
envColumn <- getSingleResponse.fun(data,response)
vCategories <- levels(data[,envColumn])
num_levels <- length(vCategories)
for (i in 1:num_levels){
   subData <- data[grep(vCategories[i],data[,envColumn]),]
   sampleSize <- round(outPercent * length(subData[,1]))
   randValues <- sample(1:length(subData[,1]), sampleSize)
   bData <- subData[-1 * randValues,]
   outData <- subData[randValues,]
   if (i == 1) {
```

```
        bagData <- bData
        oobData <- outData}
    else{
        bagData <- rbind(bagData, bData)
        oobData <- rbind(oobData, outData)}}
return(list(bag=bagData, oob=oobData))}
```

A.2 Principal Component Analysis

R has built-in principal component analysis functions, but we used the package `bpca`[12]. Recall that the PCA is an unsupervised technique, and so it does not take into account groupings, so the input data frame will be `all.data[3:29]`, a table of values only. To make a raw PCA analysis (unscaled or scaled):

```
> pca.unscale<-bpca(data[3:29],var.scale=F)
> pca.scale<-bpca(data[3:29], var.scale=T)
```

These can then be graphed. However, for sake of clarity, we want to analyze only the variables with highest variance. So we run:

```
> order(apply(data[3:29],2,sd),decreasing=T)+2
[1] 22 4 5 8 3 7 25 29 6 18 10 24 19 14 16 27 12 11 28 23
[21] 17 15 21 13 26 9 20
```

The resultant row vector (1×27) is the list (ordered high to low variance) of column names. When plotted, we found a clean break after the ten highest variance values, so we compute:

```
> pca10scale<-bpca(data[c(22,4,5,8,3,7,25,29,6,18)], var.scale=T)
```

Note that, while the PCA is blind to the classes of the data while performing its analysis, R has access to the groupings from `data`. This means that we can indicate, on a plot of the PCA, which points are from which class. This is useful for finding where natural breaks in the data coincide with those expected. To obtain Figure 29, we ran the code:

```
> plot(pca10scale, obj.name=F, obj.cex=1,
       obj.pch=c(1,2,0,4,5,6)[unclass(data$env)],
       obj.col=c(1,2,3,4,5,6)[unclass(data$env)],
```

```
xlim=c(-4,7), ylim=c(-5,5), var.factor=.5, var.cex=1)
```

A.3 K-Means

K-means is the first technique discussed that uses randomness. To exactly replicate results you get from an analysis that includes a random factor, use `set.seed(x)` for some x. When the seed value changes, the random numbers generated will also change. Setting the seed will ensure replicable results, simply reset the seed to repeat the pattern.

To perform a K-means analysis for some K, averaging over 10 independent random starts:

```
> kmeans(data[3:29], K, nstart=10)
```

To make a sum of squares plot (Figure 31):

```
> temp<-rep(NA,15)
> for(k in 1:15){
   temp[k]<-sum(kmeans(data[3:29],k+1,nstart=10)$withinss)}
> plot(2:16, temp, type="o")
```

We wrote code to make both a silhouette plot (Figure 32) and a plot of average silhouette widths (Figure 33), based off code by Marden[15]. The default is to run the K-means algorithm with 10 random starts, but the argument `iter` allows that to change. The argument `maxK` (also with default 10) is the maximum value of K that you want silhouettes for. This function takes a data frame `"x"` that has only data, no classes.

```
 > doSilhouette.fun <- function(x, iter=10, maxK = 10){
kms <- vector("list", maxK)
w <- NULL
for(k in 2:maxK) {
   kms[[k]] <- kmeans(x, centers=k, nstart=iter)
   w <- c(w,sum(kms[[k]]$withinss))}
ss <- NULL
for(k in 2:maxK) {
   a <- sil(x,kms[[k]]$centers)
   s <- NULL
   for(j in 1:k) {
      s<-c(s,sort(a[kms[[k]]$cluster==j]))}
```

```
      ss <- cbind(ss, s)}
par(mfrow=c(ceiling((maxK - 1) / 3),3))
   for(k in 2:maxK) {
      plot(ss[,k-1], type="h", ylim=c(0,1), xlab="Observations",
         ylab="Silhouettes")
      title(paste("K =", k))}
a <- apply(ss,2,mean)
win.graph()
par(mfrow=c(1,1))
plot(2:maxK,a,type="l",xlab="K",ylab="Average silhouette width")}
```

After running the function code, with 50 as our number of starts, we input:

```
> doSilhouette.fun(data[2:29],iter=50)
```

After choosing some K-values from the elbow plot and silhouettes, we like to plot the PCA analysis with K-means cluster choices. In Figure 34, we graphed the scaled PCA analysis over the 10 highest-variance functional groups, and used $K = 9$. We've colored the image according to our classes, but shaped the data points according to their assigned cluster. To make this image:

```
> kmeans(data[c(22,4,5,8,3,7,25,29,6,8)], 9, nstart=10)-> cl9
> plot(pca10scale, obj.name=F, obj.cex=1,
      obj.pch=c(1,2,0,4,5,6)[unclass(cl6$cluster)],
      obj.col=c(1,2,3,4,5,6)[unclass(data$env)],
      xlim=c(-5,20), ylim=c(-5,10), var.factor=5, var.cex=1)
```

A.4 Linear Discriminant Analysis

The linear discriminant analysis is one of the simpler functions to run in R. The function `lda` is in the package `MASS`[31] that is built in to R. To perform an LDA against the environmental groups, simply run:

```
> lda.data<-lda(data[3:29],data$env)
```

To plot the LDA, just use:

```
plot(lda.data, col=c(1,2,3,4,5,6)[unclass(data$env)])
```

To plot just the first two linear functions, add the argument `dimen=2`.

To estimate the misclassification rate of the LDA, we wrote a function that runs a *leave-one-out* cross-validation. The data frame inputed into this function needs to have the classification in its first column, the data in the rest. First we run the code to create the function:

```
> error_est<-function(x,response){
n<-length(x[,1]);
k<-0;
getSingleResponse.fun(x,response)->response;
nlevels(x[,response])->levels.n;
confusion<-matrix(0,nrow=levels.n,ncol=levels.n);
row.names(confusion)<-levels(x[,response]);
for(i in 1:n){
   lda(x[-i,-response],x[-i,response])->x_lda;
   predict(x_lda,x[i,-response])->x_predict;
   ifelse(x_predict$class==x[i,response],k+0->k,k+1->k);
   (1:levels.n)[levels(x[,response])==x[i,response]]->confusion.r;
   (1:levels.n)[levels(x[,response])==x_predict$class]->confusion.c;
   confusion[confusion.r,confusion.c]+1->confusion[confusion.r,confusion.c]};
output<-vector("list",2);
output[[1]]<-confusion;
output[[2]]<-k*1/n
output}
```

With this function, you can run LDA leave-one-out cross-validation on any data frame, as long as it has the classes in the first columns, and it will return the misclassification rate. For our example:

```
> error.est(data[2:29])
      [1] 0.167
```

A.5 Trees

To exactly replicate results you get from an analysis that includes a random factor, use `set.seed(x)` for some x. When the seed value changes, the random numbers generated will also change. Setting the seed will ensure replicable results, simply reset the seed to repeat the pattern.

To do tree analyses in R, we used the `tree`[22] package.

```
> tr.data<-tree(as.factor(env)~.,data[3:29])
> plot(tr.data);text(tr.data, digits=1)
```

Optional: add argument `label="yprob"` to print the proportion of samples in each leaf that belong to each class.

We wrote code to find the average deviance from a series of cross-validation experiments. The function formats the names of predictor variables (`predict`) and `response`, grows a tree using the control parameters `mincut` and `minsize`, repeats a cross validation experiment `n` times, then outputs a data frame of tree size and average deviance.

```
 > table.deviance<-
function(y, predict, response, n=100, kfold=10, mincut=0, minsize=0){
createFormula.fun(y, response, predict)->fmla;
getSingleResponse.fun(y, response)->response;
ifelse(mincut!=0, tree.control(length(y[,response]), mincut, minsize)->control,
   tree.control(length(y[,response]),5,10)->control);
y.tr<-tree(formula=fmla, data=data.frame(y), model=T, control=control);
y.tr$call<-call("tree", fmla, data=quote(y));
length(cv.tree(y.tr,,prune.misclass,K=kfold)$dev)->n.dev;
x<-rep(0,n.dev);
for(i in 1:n){
   cv.tree(y.tr,,prune.misclass,K=kfold)$dev+x->x};
cbind(cv.tree(y.tr,,prune.misclass,K=kfold)$size,x/n)->x;
rev(x[,1])->x[,1];
rev(x[,2])->x[,2];
data.frame(x)->x;
names(x)<-c("Tree Size","Average CV Deviance");
x;
}
```

A second function formats the names of predictor variables (`predict`) and `response`, grows a tree using the control parameters `mincut` and `minsize`, keeps track of total misclassifications over `n` cross-validation trials, then finds and plots the mean and standard error over all the trials.

```
 > plot.misclass<-
```

```
    createFormula.fun(y, response, predict)->fmla;
    getSingleResponse.fun(y, response)->response;
    ifelse(mincut!=0, tree.control(length(y[,response]), mincut, minsize)->control,
       tree.control(length(y[,response]),5,10)->control);
    y.tr<-tree(formula=fmla, data=data.frame(y), model=T, control=control);
    y.tr$call<-call("tree", fmla, data=quote(y));
    length(cv.tree(y.tr,,prune.misclass,K=kfold)$dev)->n.dev;
    x<-matrix(0,nrow=n,ncol=n.dev);
    for(i in 1:n){
       cv.tree(y.tr,,prune.misclass,K=kfold)$dev->x[i,]};
    apply(x,2,sd)->x.sd;
    apply(x,2,mean)->x.mean;
    matplot(cv.tree(y.tr,,prune.misclass,K=kfold)$size,
       cbind(x.mean,x.mean-x.sd,x.mean+x.sd), type="l",
       ylab="Misclassifications", xlab="Tree Size",
       col=c(1,2,2), lwd=2,main=" ");
    legend("topright",legend=c("Mean","Mean+1sd","Mean-1sd"),
       lty=c(1,3,2),lwd=2,col=c(1,2,2));
}
```

These functions were used to make the plots in Section 9.5. Specifically, for Figure 36:

```
> tree(env∼.,big[,2:29])->big.tree
> plot(big.tree);text(big.tree)
```

Figure 37:

```
> plot.misclass(big,c(3:29),2,n=100)
```

Table 9:

```
> table.deviance(big,c(3:29),2,n=100)
```

Figure 38:

```
> prune.tree(big.tree,best=4)->big.tree4
> plot(big.tree4);text(big.tree4)
```

A.6 Random Forests

To exactly replicate results you get from an analysis that includes a random factor, use `set.seed(x)` for some x. When the seed value changes, the random numbers generated will also change. Setting the seed will ensure replicable results, simply reset the seed to repeat the pattern.

To do Random Forest analyses, we used the `randomForest`[17] R package. To compute a supervised Random Forest analysis:

```
> rfAll <- randomForest(all.data[3:29], all.data$env,
  importance=TRUE, proximity=TRUE, ntree=5000)
```

Then we want to plot the variance important plots from this analysis (Figures 41):

```
    > varImpPlot(rfAll, type=1, pch=19, col=black, cex=.8)
> varImpPlot(rfAll, type=2, pch=19, col=black, cex=.8)
```

And the unsupervised:

```
> urfAll <- randomForest(all.data[3:29],proximity=TRUE,ntree=5000)
```

With the unsupervised Random Forest analysis, we can create MDS plots. This is the coding for (Figure 39b):

```
> MDSplot(urfAll, all.data$env,
bg=rainbow(length(levels(all.data$env)))[unclass=all.data$env],
pch=c(rep(21:25,plot.cycles),
21:(21 + plot.remainder))[unclass=all.data$env],
palette=rep("black",length(levels(all.data$env))))
```

A.7 Canonical Discriminant Analysis

For our canonical discriminant analyses, we used the R package `candisc`[13]. To perform a CDA, we first build a linear model using the functional groups (here referenced by their shortened column names) as our explanatory variables, and environment class (`env`) as our response. Any set of variables can be used in the formula, and several good methods of choosing variables have been discussed previously.

```
   > data.mod <- lm(cbind(Amino, Carb, Cell.Wall, Cofactor.etc, DNA.met,
      Dormancy, Fatty.Acids, Mem.Trans, Aromatic, Motility, Nitrogen,
      Nucleo, Phages.etc, Phosphorus, Photosynthesis, Plasmids,
      Potassium, Protein.met, Regulation, Respiration, RNA.met,
      Secondary.met, Stress.Response, Sulfur, Virulence)
      ~ env, data=data)
```

Only then can we use the canonical discriminant function:

```
   > data.can <- candisc(data.mod, data=data)
```

The `candisc` object can then be plotted. To obtain Figure 43:

```
   > data2.mod <- lm(cbind(Cell.Wall, Cofactor.etc, DNA.met, Dormancy,
      Mem.Trans, Aromatic, Phages.etc, Respiration, Secondary.met,
      Stress.Response) ~ env, data=data)
> data2.can <- candisc(data2.mod, data=data)
> plot(data2.can, pch=c(1,2,3,4,5,6)[unclass(data$env)],
      col=c(1,2,3,4,5,6)[unclass(data$env)], conf=0)
```

Two methods for error-estimation of CDA were discussed in Section 8.3, the first is here:

```
   > can_error<-function(y,response,predictors){
getSingleResponse.fun(y,response)->response
for(i in 1:length(predictors)){
   getSingleResponse.fun(y,predictors[i])->predictors[i]} createFormula.fun(y,response,predictor
length(y[,1])->row.n;
length(y[1,])->col.n;
levels.n<-nlevels(y[,response]);
error.n<-0;
confusion<-matrix(0,nrow=levels.n,ncol=levels.n);
row.names(confusion)<-levels(y[,response]);
for(i in 1:row.n){
   y[-i,]->x;
   y.mod <- lm(fmla,data=x);
   y.can <- candisc(y.mod,data=x);
   y_norm<-y;
   apply(x[,predictors],2,mean)->x_mean;
```

```
    for(j in 1:length(predictors)){
        y[,predictors[j]]-x_mean[j]->y_norm[,predictors[j]]};
    as.numeric(y_norm[i,predictors])%*%y.can$coeffs.raw->score;
    means.n<-length(y.can$mean[,1]);
    mean_dist<-rep(0,means.n);
    cov.list<-vector("list",levels.n);
    for(h in 1:levels.n){
        grep(levels(y.can$scores[,1])[h],y.can$scores[,1])->class.rows;
        ifelse(i>=min(class.rows),class.rows,class.rows-1->class.rows);
        cov(y.can$scores[class.rows,-1])->cov.list[[h]]};
    for(j in 1:means.n){
        mahalanobis(as.numeric(score), as.numeric(y.can$mean[j,]),cov.list[[j]])
            ->mean_dist[j]};
    (1:levels.n)[levels(y[,response])==y[i,response]]->confusion.r;
    confusion[confusion.r,order(mean_dist)[1]]+1
        ->confusion[confusion.r,order(mean_dist)[1]];
    ifelse(row.names(y.can$mean)[order(mean_dist)[1]]==y[i,response],
        error.n->error.n,error.n+1->error.n);
};
output<-vector("list",2);
output[[1]]<-confusion;
output[[2]]<-error.n/row.n;
output;
};
```

This is the second:

```
    > cdaErrorEst.fun <- function(metagenome, response, predictors,
        outPercent=.2, trials=length(metagenome[,1])){
totalTest = 0
totalError = 0
fmla <- createFormula.fun(metagenome,response,predictors)
envColumn <- getSingleResponse.fun(metagenome,response)
envFactor <- levels(metagenome[,envColumn])
facLength <- length(envFactor)
confuMatrix <- matrix(rep(0, facLength * (facLength+1)),
nrow=facLength, ncol=(facLength+1),
```

```
dimnames=list(envFactor, c(envFactor, "class.error")))
for (i in 1:trials){
   metagenomeBags <- bagData.fun(metagenome, outPercent, response)
   bagMG <- metagenomeBags$bag
   oobMG <- metagenomeBags$oob
   metagenome.mod <- lm(fmla, data=bagMG)
   metagenome.can <- candisc(metagenome.mod, data=bagMG)
   can.scores <- metagenome.can$scores
   can.env <- can.scores[,1]
   metagenome.lda <- lda(can.scores[,-1], can.env)
   bag.means <- apply(bagMG[,predictors],2,mean)
   oobMG.norm <- oobMG[,predictors]
   for (j in 1:length(oobMG[,1])){
      oobMG.norm[j,] <- oobMG.norm[j,] - bag.means}
   oobMG.scores <- as.matrix(oobMG.norm) %*% as.matrix(metagenome.can$coeffs.raw)
   metagenome.predict <- predict(metagenome.lda, oobMG.scores)
   for (k in 1:length(oobMG[,1])){
      totalTest = totalTest + 1
      currentRow <- oobMG[k,]
      trueClass <- as.character(oobMG$env[k])
      predictClass <- as.character(metagenome.predict$class[k])
      confuMatrix[trueClass,predictClass]
         <- confuMatrix[trueClass,predictClass] + 1
      if (predictClass != trueClass){
         totalError = totalError + 1}
   }}
confuMatrix <- confuMatrix / trials
for (m in 1:facLength) {
   confuMatrix[m,facLength+1] = 1 - (confuMatrix[m,m] / sum(confuMatrix[m,]))
   }
return (list(error=(totalError / totalTest), confusion=confuMatrix))}
```

A.8 Packages Used

The R packages we used in the analyses presented here are:

- bpca[12]
- tree[22]
- randomForest[17]
- candisc[13]
- cluster[18]
- RColorBrewer[20]

References

[1] Thomas L. Madden Alejandro A. Schaffer Jinghui Zhang Zheng Zhang Webb Miller Altschul, Stephen F. and David J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

[2] FE. Angly, B. Felts, M. Breitbart, P. Salamon, RA. Edwards, C. Carlson, AM. Chan, M. Haynes, S. Kelley, H. Liu, JM. Mahaffy, JE. Mueller, J. Nulton, R. Olson, R. Parsons, S. Rayhawk, CA. Suttle, and FL. Rohwer. The marine viromes of four oceanic regions. *PLoS Biology*, 4(11):21212131, 2006.

[3] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[4] JM. Brulc, DA. Antonopoulos, ME. Berg Miller, MK. Wilson, AC. Yannarell, EA. Dinsdale, RE. Edwards, ED. Frank, JB. Emerson, P. Wacklin, PM. Coutinho, B. Henrissat, KE. Nelson, and BA. White. Gene-centric metagenomics of the fiber-adherent bovine rumen microbiome reveals forage specific glycoside hydrolases. *PNAS*, 106(6):1948–1953, 2009.

[5] N.A. Campbell and William R. Atchley. The geometry of canonical variate analysis. *Systematic Zoology*, 30(3):268–280, 1981.

[6] S. Dasgupta. The Mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1–18, 2000.

[7] Glenn De'ath and Katharina Fabricius. Classification and regression trees: A powerful yet simple technique for ecological data analysis. *Ecology*, 81(11):3178–3192, 2000.

[8] C. Desnues, B. Rodriguez-Brito, S. Rayhawk, S. Kelley, T. Tran, M. Haynes, H. Liu, M. Furlan, L. Wegley, B. Chau, Y. Ruan, D. Hall, F. E. Angly, R. A. Edwards, L. Li, R. Vega Thurber, R. P. Reid, J. Siefert, V. Souza, D. Valentine, B. K. Swan, M. Breitbart, and F. Rohwer. Biodiversity and biogeography of phages in modern stromatolites and thrombolites. *Nature*, 452(20 March 2008):340–343, 2008.

[9] EA. Dinsdale, RA. Edwards, D. Hall, F. Angly, M. Breitbart, JM Brulc, M. Furlan, C. Desnues, M. Haynes, L. Li, L. McDaniel, MA. Moran, KE. Nelson, C. Nilsson, R. Olson, J. Paul, B. Rodriguez-Brito, Y. Ruan, BK. Swan, R. Stevens, DL. Valentine, RV. Thurber, L. Wegley, BA. White, and F. Rohwer. Functional metagenomic profiling of nine biomes. *Nature*, 452:629–632, 2008.

[10] EA. Dinsdale, O. Pantos, S. Smirga, RA. Edwards, F. Angly, L. Wegley, M. Hatay, D. Hall, E. Brown, M. Haynes, L. Karuse, E. Sala, SA. Sandin, RV. Thurber, BL. Willis, F. Azam, N. Knowlton, and F. Rohwer. Microbial ecology of four coral atolls in the northern line islands. *PLoS ONE*, 3(2):1–17, 2008.

[11] T. Disz, R. Edwards, R. Olson, R. Overbeek, G. Pusch, A. Rodriguez, and V. Vonstein. A novel use of k-mers to support accurate and rapid annotation of prokaryotic genomes and environmental samples.

[12] Jose Claudio Faria and Clarice Garcia Borges Demetrio. *bpca: Biplot of Multivariate Data Based on Principal Components Analysis*. UESC and ESALQ, Ilheus, Bahia, Brasil and Piracicaba, Sao Paulo, Brasil, 2009.

[13] Michael Friendly and John Fox. *candisc: Generalized Canonical Discriminant Analysis*, 2009. R package version 0.5-15.

[14] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Leanring*. Springer, 2001.

[15] Marden J. *Multivariate Statistical Analysis Old School*. lulu, 2008.

[16] K. Kurokawa, T. Itoh, T. Kuwahara, K. Oshima, H. Toh, A. Toyoda, H. Takami, H. Morita, VK. Sharma, TP. Srivastava, TD. Taylor, H. Noguchi, H. Mori, Y. Ogura, DS. Ehrlich, K. Itoh, T. Takagi, Y. Sakaki, T. Hayashi, and M. Hattori. Comparative metagenomics revealed commonly enriched gene sets in human gut microbiomes. *DNA Research*, pages 1–13, 2007.

[17] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[18] Martin Maechler, Peter Rousseeuw, Anja Struyf, and Mia Hubert. Cluster analysis basics and extensions. Rousseeuw et al provided the S original which has been ported to R by Kurt Hornik and has since been enhanced by Martin Maechler: speed improvements, silhouette() functionality, bug fixes, etc. See the 'Changelog' file (in the package source), 2005.

[19] F Meyer, D Paarmann, M D'Souza, R Olson, EM Glass, M Kubal, T Paczian, A Rodriguez, R Stevens, A Wilke, J Wilkening, and RA Edwards. The metagenomics rast server - a public resource for the automatic phylogenetic and functional analysis of metagenomes. *BMC Bioinformatics*, 9(1):386, 2008.

[20] Erich Neuwirth. *RColorBrewer: ColorBrewer palettes*, 2007. R package version 1.0-2.

[21] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2009. ISBN 3-900051-07-0.

[22] Brian Ripley. *tree: Classification and regression trees*, 2007. R package version 1.0-26.

[23] T. Shi and S. Horvath. Unsupervised learning with random forest predictors. *Journal of Computational and Graphical Statistics*, 15(1):118–138(21), 2006.

[24] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195 – 197, 1981.

[25] Leo Breiman Statistics and Leo Breiman. Random forests. In *Machine Learning*, pages 5–32, 2001.

[26] Terry M Therneau and Beth Atkinson. R port by Brian Ripley. *rpart: Recursive Partitioning*, 2009. R package version 3.1-45.

[27] RLV. Thurber, KL. Barott, D. Willner-Hall, H. Liu, B. Rodriguez-Mueller, C. Desnuesa, RA. Edwards, M. Haynes, FE. Angly, L. Wegley, and FL. Rohwer. Metagenomic analysis indicates that stressors induce

production of herpes-like viruses in the coral porites compressa. *Proceedings of the National Academy of Sciences, USA*, 105(47):1841318418, 2008.

[28] RV. Thurber, D. Willner-Hall, B. Rodriguez-Mueller, C. Desnues, RA. Edwards, F. Angly, E. Dinsdale, L. Kelly, and F. Rohwer. Metagenomic analysis of stressed coral holobionts. *Environmental Microbiology*, 11(5):1–16, 2009.

[29] PJ. Turnbaugh, M. Hamady, T. Yatsunenko, BL. Cantarel, A. Duncan, RE. Ley, ML. Sogin, WJ. Jones, BA. Roe, JP. Affourtit, M. Egholm, B. Henrissat, AC. Heath, R. Knight, and JI. Gordon. A core gut microbiome in obese and lean twins. *Nature*, 457:480–484, 2008.

[30] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, 2002.

[31] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0.

[32] L. Wegley, R. Edwards, B. Rodriguez-Brito, H. Liu, and F. Rohwer. Metagenomic analysis of the microbial community associated with the coral porites astreoides. *Environmental Microbiology*, 9(11):2707–2719, 2007.

[33] D. Willner, RLV. Thurber, and F. Rohwer. Metagenomic signatures of 86 microbial and viral metagenomes. *Environmental Microbiology*, 2009.