University of Chicago

Master's Thesis

Modified Bootstrapping and K-Means Clustering for Taxonomic Binning

Author:
Anna Olson

Advisor:
Dr. Rick Stevens

October 28, 2014

Contents

Abstract

Metagenomics is the study of microbial ecology using genetics as an access point. We seek to understand the microbial communities in environments such as tidal pools, soil, mine runoff, or even the human gut, so that we can understand the impact that microbes have on our world and our health. Metagenomic analysis usually involves the determination of what species are present in a given sample, and, if possible, what each species' genome looks like. Because the vast majority of microbial organisms cannot be cultured easily in a generic lab setting, we look instead to the DNA taken from a sample, and try to assign the DNA to different species (the taxonomic binning problem). Many methods for solving the taxonomic binning problem exist, but they are extremely computationally intensive, reliant on woefully incomplete reference databases, or insufficiently accurate.

In this paper, we put forward a new taxonomic binning algorithm called HORATIO. HORATIO is a modified bootstrapping and k-means clustering algorithm for taxonomic binning that uses compositional information about DNA subsequences. HORATIO requires only moderate setup, and has tunable parameters that can reflect users' preferences. We compare and contrast HORATIO to other extant taxonomic binning algorithms and evaluate its performance against simulated and real metagenomic datasets.

Chapter 1

Introduction

There are many overlapping and complementary definitions and goals for metagenomics [1, 2, 3, 4], but they can be summarized as the study of microbiomes and microbial organisms using the data available from environmental DNA and RNA samples. The questions addressed by metagenomics can range from the simple, such as the identification of all species in a sample, to the deep, such as the characterization of an environment as a functioning whole made of many individual organisms.

Even the question of species identification is far more difficult than it might seem at first glance. The vast majority of bacteria cannot, at present, be cultured in laboratory conditions, and so the traditional method of culturing a sample to see what appears will miss a great deal of information about the environment it comes from. A natural approach to this problem is to examine the DNA present, and see if it matches against any known genomes, but this suffers from the fact that genome databases are necessarily limited largely to culturable organisms. An effective species identification method that is not subject to the limitations of extant databases is therefore sorely needed.

If we are able to accurately identify clusters of DNA that come from the same arbitrary source organism, we could start to construct full genomes without laboratory culturing

and learn about phylogenetic relationships between newly discovered organisms and well-understood organisms. Species identification, or taxonomic binning, could help pinpoint which species are responsible for which proteins, and thus would elucidate the ecology of an environment. It is already known that microbiomes in the human gut can differ from individual to individual and those differences seem to be related to overall health, but the mechanism is not well-understood; taxonomic binning could further research in this area.

Taxonomic binning is made difficult by three general factors: sequencing technology, the sparseness of extant databases, and a lack of understanding of the space of genetic sequences. Next-generation sequencing, while vastly faster and cheaper than traditional Sanger sequencing, produces lower-quality reads; resultant sequences are short, typically on the order of 100-500 base pairs, and read errors increase with sequence length. Current genomic databases are sparse, typically populated only with well-understood organisms, and so any matching of environmental DNA to those databases will inevitably have a skewed result. Finally, the information-theoretic and statistical properties of space of genomes are not known; we still do not have a clear understanding of what characteristics of genomes are most appropriate to use for species discrimination.

HORATIO is an algorithm that attempts to address the problem of database sparseness by using various clustering algorithms and known similarity scoring mechanisms to produce clusters of DNA from similar phylogenetic backgrounds. First a modified $k-$means clustering algorithm is used to produce small pure clusters, and then Affinity Propagation is used to group those clusters together. In order to reduce computational requirements, it divides the input based on size, and thus requires only a linear number of combinations, instead of quadratic. As a result, it runs relatively quickly compared to other reference-free algorithms. In this paper we identify the parameters that produce the best results, which are better than the results produced by simple implementations of the scoring mechanisms for certain classes of microbiomes.

Chapter 2

Background

Researchers often want to know what microbial species are present in an environment, and what role these species fulfill [1, 5, 6]. At a first glance, the identification of present species might seem an easy question to answer simply by culturing samples and examining the cultures for known species. However, up to 99% of all microbial species cannot be cultured in a laboratory setting with our present knowledge [7]. Simply culturing a sample will miss the vast majority of the species present, and and researchers cannot hope to get a good understanding of the system. Therefore, it is necessary to use another method for species identification. Researchers can sequence the DNA found in an environment to identify species, as this method does not require laboratory culturing beyond that required by sequencing [8]. From the sequenced DNA, researchers try to determine species makeup as well as gene functionality and the overall metabolism of the biome. Given sufficient sequencing depth, complete genomes for hitherto unknown species can be constructed from the sequenced DNA [6, 8].

2.1 Overview of Metagenomic Workflow

This work focuses solely on binning, a single part of the metagenomic analysis pipeline. Generally speaking, metagenomic workflows follow the following structure [1]:

Sampling The process of obtaining useful DNA. Care should be taken to avoid contamination, either from the environment, sampling procedures, or host organisms. Extraction procedures have a great effect on the quality of obtained DNA.

Sequencing Once DNA is obtained, it must be sequenced. Sanger sequencing is the most accurate method, and yields the longest DNA reads, but it is also the most expensive and most time-consuming option. Newer technologies, such as Illumina and 454, are much cheaper, but produce short reads of DNA, on the order of 100-200 bp for Illumina and 600-800 bp for 454.

Assembly There is a limit to how much analysis can be performed on short reads, and so reads must be assembled into longer contigs. Assembly is based on overlapping subsequences and read quality. Due to the presence of long mono-base repeats that exceed the length of a read, as well as regions that are highly similar between multiple sequences, assembly is necessarily limited.

Binning Binning is the process of assigning each sequence to a taxonomic clade [6]. Binning can be easier to perform on longer sequences, and so usually occurs after assembly. However, some binning can be performed prior to assembly, and assembly can be made easier by having a smaller query space. This means that there can be a feedback loop between assembly and binning steps.

Annotation and Analysis Once long contigs are constructed, and their provenance is identified, the workflow can extend to annotation and analysis. Annotation refers to the identification of genes and their functions and neighbors. Further analysis can include genome construction and identification of the roles that different species fulfill in their environment.

2.2 Binning

Binning consists of assigning each sequence to a taxonomic clade [1, 6]. Binning can be much easier with longer sequences to work from, so it is often performed on assembled contigs [9].

Ideally, it would be possible to assign each sequence to a species, or, if its provenance is unknown, determine its closest relationship on the phylogenetic tree. The computational requirements of binning algorithms should be minimized, given the vast amounts of data that need processing.

2.2.1 Similarity-Based Methods

Similarity-based binning methods are based on multiple sequence alignment. Each query sequence is aligned against sequences in a database, using alignment techniques like BLAST [1, 10]. For each query sequence, the closest database sequence is determined to be a match. The closeness of a match is reflective of evolutionary distance and the likelihood of different mutations and sequencing/reading errors.

Sequence alignment distance is viewed [10, 11] as a reflection of phylogenetic distance, given the nature of evolutionary mutation. Thus, sequence-based methods offer evolutionary insight as well as taxonomic binning. They are also generally highly accurate [10], as they are very unlikely to assign a query sequence to an unrelated genome.

To see how sequence alignment can be used to determine phylogenetic distances and evolutionary history, consider the 16S sequence. The 16S subunit consists of an RNA sequence that folds and binds to itself to create part of a prokaryotic ribosome, which is responsible for protein synthesis. Because ribosomes are absolutely necessary to an organism's survival, the rate of evolutionary change is low, and therefore there is a high degree of similarity between

16S subunits of different clades [11]. Thus, if a researcher has a 16S sequence from a metagenomic sample, they can align it against a database of 16S sequences and find the closest match. The closest match - and the distance between that match and the original sequence - will give valuable phylogenetic information [12]. As a bonus, determining the provenance of known common gene sequences in a metagenomic sequence can help researchers narrow down their search parameters when trying to find matches for other sequences in their metagenomic sample.

Unfortunately, sequence alignment techniques are computationally intensive and thus inappropriate for the vast amounts of metagenomic data needing classification [10], given the high demands it makes on computational resources. The canonical sequence alignment algorithms, Needleman-Wunsch and Smith-Waterman, use dynamic programming to find the optimal alignment between two sequences [13, 14]. Many alignment methods today focus on reducing the effective size of the database for each query sequence; for example, the Basic Local Alignment Search Tool (BLAST) uses a heuristic algorithm that searches for short substrings to focus only on database elements that are likely to offer good alignments [15].

Examples of alignment-based binning methods include, but are not limited to:

MEGAN Based on BLAST results and a taxonomy database, MEGAN determines a phylogeny for a query sequence. If there is no exact match for a query sequence available, MEGAN finds the lowest common ancestor of all the close matches and assigns that classification to the query sequence. [3]

Signature peptides Instead of trying to align every part of every sequence against a database, this method preprocesses a database of known genomes to find "signature peptide" sequences specific to given clades. Query sequences are given the best assignment possible based on the presence/absence of signature peptides. [16]

AMPHORA AMPHORA looks for protein-coding sequences in query contigs, and aligns

them against a core database of known universal bacterial genes. The search space is reduced through the use of HMMs. Previously unknown genes that are part of the same query contigs as known genes can thus be assigned to a known phylogeny. [17]

MTR First, reads are processed via BLAST, against a known database, to find provisional assignments. Instead of using a Least Common Ancestor (LCA) algorithm, MTR uses an iterative clustering and set-covering algorithm that optimizes for consistency with clustering at higher taxonomic ranks and a small number of larger bins. Only protein-coding sequences with close matches in the given database are considered in MTR in order to produce a fast and accurate algorithm. [18]

TANGO Query sequences are compared against a database using BLAST. TANGO presents another alternative to the LCA approach, i.e., the F-measure approach. For each candidate assignment of a query sequence to an entity in the taxonomic tree, sequence hits and misses among the entity (and its children) are compiled to produce a score that reflects the appropriateness of the assignment. [19, 20]

SOrt-ITEMS First, BLAST alignments are used to determine the appropriate taxonomic level at which a query sequence can be assigned with certainty. The degree of similarity between the query sequence and the hit sequences in each possible species-level assignment is then used to further push the query sequence down the taxonomic tree and to provide greater specificity than an LCA algorithm. [12]

DiScRIBinATE DiScRIBinATE is an improvement upon SOrt-ITEMS, specifically in the method by which query sequences are assigned to finer taxonomic levels after alignment. First a list of possible ancestors for the source of the query sequence is tabulated, and then the query sequence is assigned to the most likely ancestor as determined by the alignment scores and the levels at which the LCA and possible ancestors appear. [21]

2.2.2 Composition-Based Methods

Composition-based methods are based on the idea that details about composition are fairly constant across a given genome [1, 6], and are sufficiently different from other genomes to warrant use as a discriminator. From a mathematical perspective, this means the space of all genomes and most subsequences can be reduced to a much smaller space using some string kernel without loss of phylogenetic information. Many composition-based methods use k-mer frequency spectra - or derivatives thereof - to differentiate; that is, k-mers that are frequent or infrequent in one genome should be frequent in sufficiently large subsequences, and and the profile of frequencies should be unique to a monophyletic clade. Some methods look at the significance of k-mers based on the frequencies of $(k-i)$-mers [4, 22, 23], while others use the k-mer spectra to build Markov models [24].

Composition-based methods are, in general, much faster than their alignment-based relatives. This is largely a function of the computationally intensive nature of alignment-based binning [8].

Due to their nature, composition-based methods are ill-suited for small query sequences, as these sequences are not long enough to contain a strong compositional signal [1, 25]. As query sequences get shorter, each k-mer has a disproportionate impact. For example, if there is a k-mer w that appears 100x less frequently in the genome than average, and if it appears in a query sequence from that genome, then the algorithm will attempt to assign the query sequence to a genome with a much higher representation of w.

It should be noted that composition may not be constant across the entirety of a genome. For one thing, some regions of the genome are less variable than others, and will not drift as far in a compositional direction. For another, horizontal gene transfer can result in some regions of a genome looking drastically different from another.

Examples of composition-based binning methods include, but are not limited to:

TETRA An approximation of the significance of each $4-$mer, based on 3-mers and 2-mers, is computed and stored in a profile vector. The similarity between two sequences is defined as the Pearson correlation coefficient between the profile vectors of those two sequences. [22]

TACOA Like TETRA, TACOA looks at the significance of $k-$mers based on shorter sequences. In this case, the significance of a $k-$mer is computed relative to the sequence's GC ratio. The similarity between two TACOA profiles is the inner product of the two vectors. [23]

RAIphy RAIphy builds **r**elative **a**bundance **i**ndex (RAI) profiles for genomic sequences based on the log-odds ratio of each $k-$mer. The similarity between sequences is computed as the sum, over all $k-$mers, of the frequency of the $k-$mer in the query sequence multiplied by the log-odds ratio of that $k-$mer in the sequence to which the query sequence is to be matched. Note that this is not a commutative measure. RAIphy can be an iterative measure, wherein database elements are updated to reflect the composition of the query sequences that it matched to and the algorithm is run again until no query sequences are assigned to new database elements. [4]

INDUS Unlike TETRA, TACOA and RAIphy, INDUS does not assume that composition stays constant throughout the genome, for reasons cited above. Instead of comparing a query sequence to a profile representative of an entire genome, INDUS compares a query sequence to a profile representative of a short fragment. The INDUS database comes from a set of roughly 1000 genomes from the NCBI database. [8]

AbundanceBin This algorithm assumes that each present genome has a general frequency for each $k-$mer. Genomes are represented at different coverage levels giving a mixed Poisson distribution, where each bin has an associated $k-$mer frequency profile and

value of λ. The assignment of contigs to bins is driven by an Expectation-Maximization algorithm. [26]

Metawatt The frequency of each $k-$mer in each contig is computed, and the probability of a contig belonging to a given bin is a function of their respective $k-$mer profiles. The algorithm starts with a single bin, consisting of the longest contig. As each subsequent contig is examined, in order of size, new bins are created if contigs don't fit into any extant bins with appropriate probability. [27]

ClaMS The profile of a given sequence is its de Bruijn chain signature (DBC), which is robust to even short sequences. A contig is assigned to whichever database element - or training set - has the closest match to its DBC signature. [28]

CompostBin Not all $k-$mers are equally informative or have equal phylogenetic discriminative power. CompostBin performs weighted Principal Component Analysis on 6-mers in each input contig in order to reduce the dimensionality of the space. The space is then repeatedly bisected until the number of bins is equal to the number of expected clades. [9]

CONCOCT Like AbundanceBin, CONCOCT models the metagenomic sample as the output of a mixed model - in this case, of Gaussians. The $k-$mer profile and coverage of each query sequence is computed, and the query sequences are binned using an Expectation Maximization algorithm. The number of clusters is computed using the Bayesian information criterion. [29]

ESOM Tetramer frequency significance vectors, like those used in TETRA, are used as the composition profiles of query sequences. The profiles are input to an Emergent Self Organizing Map (ESOM) algorithm, and the results can be used for further binning. One advantage of using self organizing maps is that the results can be easily visually inspected. [30]

PhyloPythia $k-$mer frequency vectors are used as compositional profiles for query contigs, and the profiles are binned using support vector machines (SVMs) through an iterative process: first, every possible combination of two clusters from the supplied database are used for initial binning, with contigs being assigned to the cluster they were mostly frequently associated with by the SVM round, and then each resultant cluster is checked for consistency against the database. [31]

PHYMM Interpolated Markov models (IMMs) are built for both database sequences and query sequences, and then compared. The database sequence to which a query sequence's IMM is closest is that query sequence's assignment. [24]

MetaCluster 5.0 First, low-coverage reads are filtered out of the initial query set. This high-coverage set is binned in two steps, one using presence of $w-$mers (longer genetic substrings), and the second using $k-$mer profiles in a series of $k-$means clusterings. In the second step, the number of clusters is determined by starting with an overestimation and then successively joining clusters as dictated by the expected distribution of $k-$mer frequencies within a single genome. These two binning phases are performed again on the lower-coverage reads, with lower thresholds for similarity. [25, 32]

2.2.3 Hybrid Methods

Alignment-based methods offer greater accuracy, and composition-based methods offer greater speed; therefore, it is plausible that a combination of the two techniques would offer the best of both worlds, by using composition-based methods to reduce the search space of alignment-based methods [33], or by simply combining the results of both techniques to get a more robust measure [24].

SPHINX The alignment search space in SPHINX is reduced through the use of tetramers. Database sequences are clustered according to phylogenetic closeness, and their tetramer profiles are computed. The tetramer profile of a given query sequence is compared to

every cluster and the query sequence is aligned against the members of the cluster to which its tetramer profile was closest. [33]

PhymmBL The similarity between a query and a database sequence is a combination of the IMM score from PHYMM (see above) and the BLAST score. This combined score gives greater accuracy. [24]

2.3 Reference-Based and Reference-Free Methods

Another division between binning methods exists between reference-based and reference-free methods [10, 25]. Reference-based methods compare query contigs to elements in a database, or reference, whereas reference-free methods try to determine the underlying cladistic structure of a metagenomic sample.

Reference-based methods allow researchers to identify relationships between sample contigs and other known genetic entities. Unfortunately, as a consequence of the difficulty in culturing most bacteria [7], most bacteria are ill-represented in our databases. For example, less than 2% of the NCBI database consists of the phyla Fusobacteria and Chlorobi [8, 10], when there is no reason to believe that those phyla are particularly rare. Therefore, most query sequences cannot be assigned to sufficiently close matches in genomic databases, leading to large numbers of mis-assigned or unassigned sequences [1]. Contigs can also be queried against databases of environmental samples, i.e., environmental sequences that have been sequenced and assembled but have not been phylogenetically binned; this can aid in further assembly and future phylogenetic identification [6].

Instead of using a database as a reference, it is preferable in some cases to use reference-free methods to just examine extant structure within the set of query sequences and perhaps find clusters representing different genera or species. Given monophyletic clusters, further assembly of contigs can be performed, and species diversity can be estimated.

Technically, any of the above methods that computes a similarity between two genetic sequences could be adapted into a reference-free form by simply computing the similarity matrix for the set of query sequences. This comes at the expense of computational efficiency, particularly if a reference is much smaller in number than the metagenomic sample to which it is compared: if there are N query sequences, and M elements in the reference, and if $M << N$, then the number of comparisons for a reference based method is MN, which is much smaller than N^2, the number of comparisons for a full similarity matrix. Not all reference-free methods require a full similarity matrix. For example, Leveraged Affinity Propagation requires only a sparse similarity matrix [34].

One example of an alignment-based reference-free algorithm is ESPRIT-Tree [35]. The alignment distances between all sequences are computed, and then hierarchical clustering, using average-linkage, is performed on the result. The user can select the clustering with the most appropriate separation distance for their purposes. ESPRIT-Tree offers promising results, but is computationally intensive.

2.4 Clustering

Binning is, in essence, a clustering problem [27]. Reference-based methods offer an extant structure with which to bin - or cluster - query sequences, whereas reference-free methods search the query space in order to find a structure.

K-Means Clustering Given that, in essence, RAIphy is an iterative k-means algorithm [4], we decided to focus largely on k-means clustering. In a k-means clustering, there are k clusters, each with a single center that reflects the mean of all of the elements in that cluster. A k-means clustering algorithm seeks to minimize the distances between elements and the center within each cluster. An iterative k-means clustering algorithm starts by assigning k initial seeds, and assigning remaining elements to those seeds to make k clusters. The

centers of the clusters are updated to be the mean of all of the member elements, and then all elements are reassigned to the newly moved centers. This process is repeated until no elements are reassigned to new clusters. [36]

k is not always known in advance, and certainly would not be known for metagenomic purposes. One possibility is to start with an initial set of seeds and allow for clusters to merge and split based on distance, given user-defined parameters [36, 37]. If the centers of two clusters are separated by less than a given distance C, they are merged. If an element isn't within a distance R of any cluster center, it is defined as a seed for a new cluster. The algorithm used in HORATIO is a variation of this method. [37]

Affinity Propagation Another general option for clustering that, unlike general $k-$means, doesn't require the user to know the number of clusters in advance is Affinity Propagation. Affinity Propagation seeks to identify exemplars of clusters within a data set using message-passing. At each iteration, data points exchange messages with each other about how well-suited they are to be exemplars for each other. These messages are based on the input similarity matrix and initialized preferences for each data point, where the initial preference value for a data point reflects how likely it is to be an exemplar. Larger initial preference values result in fewer clusters. We chose Affinity Propagation as another clustering method worthy of further investigation due to the fact that general implementations are widely available. [34]

Chapter 3

HORATIO

Herein we describe HORATIO. First we will begin with motivation and a high-level overview of how HORATIO works, and then we will discuss its details and usage.

3.1 Motivation

HORATIO's key feature is its contig separation. In order to reduce the number of similarity computations, we decided to compare only two sets of contigs with each other at any given time, instead of comparing every contig to every other contig. We were inspired by raiphy's use of $k-$means clustering using database elements as centroids, and decided that we could use contigs themselves as initial seeds. By comparing only a subset of contigs to the seeds at any given time, fewer computational resources would be requires. Because more compositional signal exists in longer signals, we decided that the initial seeds would be the longest seeds, i.e., those with the most signal. We therefore decided to separate the contigs by size, reserving the largest contigs as initial seeds, and comparing the remaining contigs in stages. See Figure 3.1 for a schematic.

In order to conserve as much compositional signal as possible, we decided to concatenate seeds with their matching contigs to create pseudocontigs to be the seeds for the next stage

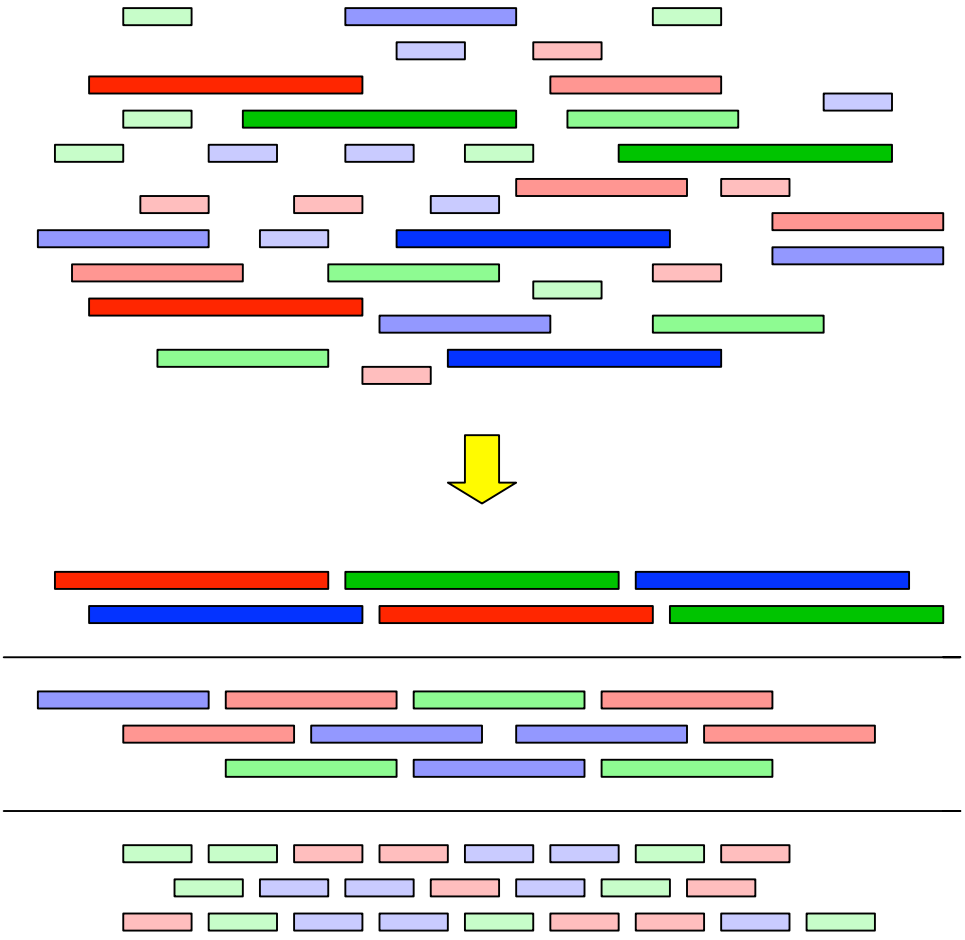

Figure 3.1: Schematic of contig separation by size

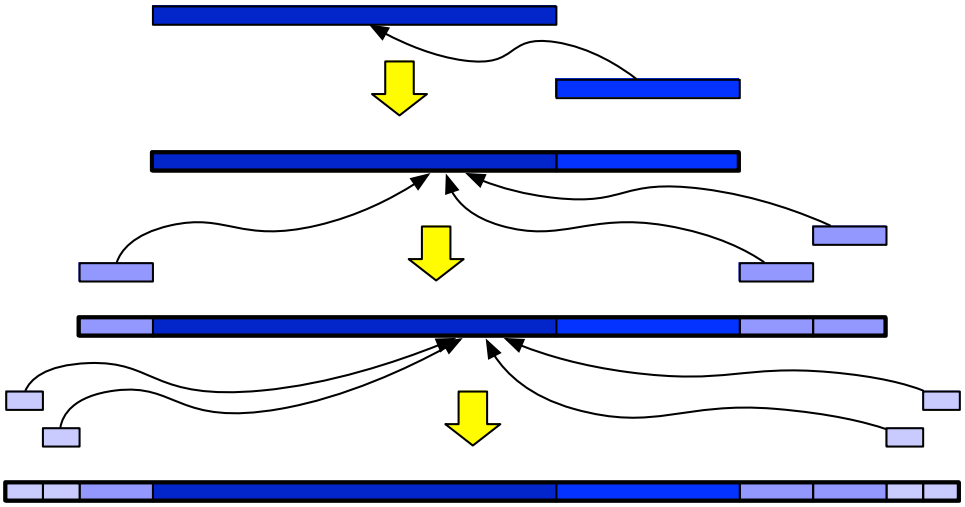

Figure 3.2: Matching contigs are concatenated to create pseudocontigs

of contigs (see Figure 3.2). Thus, the centroid of each cluster reflects all of its members, not just the original seed.

From this initial idea of using $k-$means clustering on subsets of contigs separated by size, we decided to build HORATIO. We decided to test HORATIO on three different similarity functions (raiphy, tetra and tacoa) and on multiple parameters in order to optimize it.

3.2 Description of Algorithm

3.2.1 Overview

Generally speaking, HORATIO proceeds thus: First, contigs are separated by size into different stages, where Stage 0 has the largest contigs and Stage k has the smallest. The Stage 0 contigs are used as initial roots (singleton clusters), and Stage 1 contigs are matched against these initial seeds. Each Stage 1 contig is assigned to the Stage 0 contig to which it is closest, as determined by the input similarity function. Once all Stage 1 contigs are assigned, every cluster that "recruited" Stage 1 contigs defines a pseudocontig by concatenating the current root with all contigs that were newly assigned to it. This pseudocontig becomes the

new root for the cluster, and the loop begins again, this time with clusters recruiting Stage 2 contigs. This process continues until no contigs are left to process.

In order to reduce false assignments, HORATIO will separate - or split off - contigs that don't match well to any extant cluster. This ill-matching contigs are cast as new roots for new clusters, and in the next round of HORATIO, they can recruit contigs. The threshold of similarity scores that all contigs must meet before being recruited is l, and is also known as the splitting coefficient.

Simply splitting off every ill-matching contig will result in a large number of singleton clusters, which would be useless. Therefore we need a mechanism to join clusters. We supposed that contigs would have high similarity scores with multiple clusters to which they could legitimately be assigned (i.e., clusters consisting of contigs from the same source), and that, by tracking the "runners-up" of each contig-to-cluster assignment, we could gain insight into what clusters should be joined. Suppose a contig is best-matched to a cluster with a certain similarity score `s`. Then all clusters to which the contig has similarity score `s'` are "runners-up", or neighbors, of the contig, provided that `s'` is sufficiently close to `s`. The closeness required for neighboring is n. Contigs bring their neighbor information with them to their cluster, and if the cluster sees the same neighboring clusters often enough, the two clusters will join. The frequency with which a cluster has to appear in another cluster's neighbor list before joining is the joining coefficient j.

Clusters are stored in HORATIO as trees. Each internal node is a pseudocontig, and its children are, in fact, the parental contigs, i.e., the contigs that were concatenated together to create the pseudocontig in question. See Figure 3.3 for a schematic. Clusters are merged simply by concatenating their roots to create a new root; for example, if clusters A and B are to be joined, a new root is computed by joining their two roots, and setting the old roots to be children of the new root. See Figure 3.4.

Despite the cluster-joining described above, there are still more clusters than there are genomes in the set. For a 10-genome set, up to 1100 clusters may be found. The vast majority of those clusters are singletons. To test further clustering, Affinity Propagation was chosen, as it does not require that the number of clusters as an argument [34]. Using the user-specified scoring mechanism, the similarities between the clusters are computed and the results stored in a similarity matrix. Then, using Scikit-Learn's Python implementation of Affinity Propagation, final true clusterings are computed [38]. All clusters placed in the same class by Affinity Propagation are merged, and output to the user.

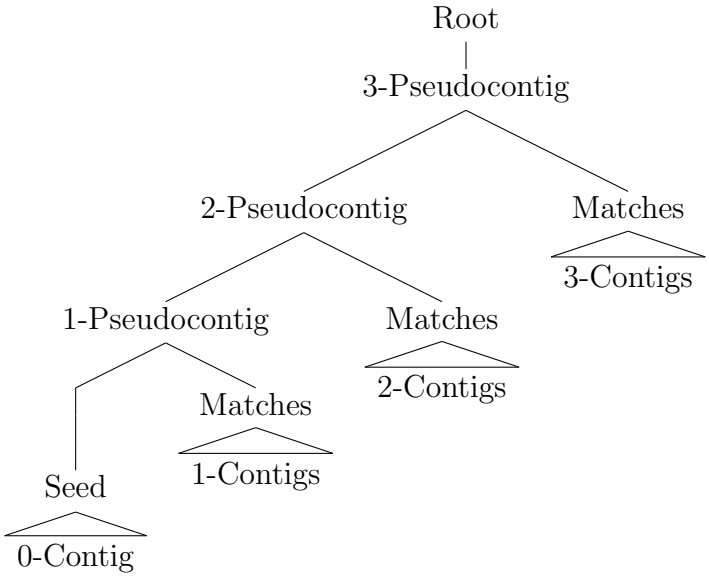

Figure 3.3: A simple cluster schematic

The pseudocode for HORATIO is laid out in Algorithm 1. Let m be a metagenome, s be a similarity function, $c = [c_0, c_1, ..., c_k]$ be a cut schedule, n be the neighboring coefficient, j be the joining coefficient, and l be the splitting coefficient.

Algorithm 1 HORATIO

procedure HORATIO(m, s, c, n, j, l, $doAP$, k, $pref$)
  for $contig \in m$ **do** ▷ Initialization
    if $|contig| \geq c_0$ **then**
      Add $contig$ to $Stage_0$
      Add $Cluster(contig)$ to $Clusters$
    else
      for $i \in [1, k]$ **do**
        if $c_{i-1} \leq |contig| < c_i$ **then**
          Add $contig$ to $Stage_i$

  for $i \in [1, k]$ **do** ▷ Main Loop
    Make database from $Stage_{i-1}$
    Initialize $NewClusters$, $MatchDictionary$
    for $contig \in Stage_i$ **do** ▷ Iterate through contigs
      for $cluster \in Clusters$ **do** ▷ Find best match
        if $s(contig, cluster) > bestScore$ **then**
          $best = s(contig, cluster)$
          $match = cluster$
      if $best \geq l$ **then** ▷ Good match
        Add $contig$ to $MatchDictionary[cluster]$
        for $cluster \in Clusters \setminus match$ **do** ▷ Find neighbors
          if $s(contig, cluster) \geq (1 - n) * best$ **then**
            Add $cluster$ to $contig.Neighbors$
      else ▷ Not a good match
        Add $Cluster(contig)$ to $NewClusters$
    for $cluster \in Clusters$ **do**
      Add $MatchDictionary[cluster]$ to $cluster$
      $cluster.Root =$ concatenation$(cluster.Root, MatchDictionary[cluster])$
    Initialize $NeighborGraph$
    for $cluster \in Clusters$ **do** ▷ Joining clusters together
      $neighbor, freq = cluster.MostCommonNeighbor()$
      if $freq \geq j$ **then**
        Add $(cluster, neighbor)$ to $NeighborGraph$
    for co in $NeighborGraph.components$ **do**
      Remove $cluster$ from $Clusters$ $\forall cluster \in co$
      Add $Join(co \; \forall \; cluster \in co)$ to $Clusters$

  if $doAP$ **then**
    $InClusters = \{|cluster| \geq k \quad \forall cluster \in Clusters\}$
    $Clusters = AffinityPropagation(InClusters, pref)$ ▷ Final clustering
  return $Clusters$

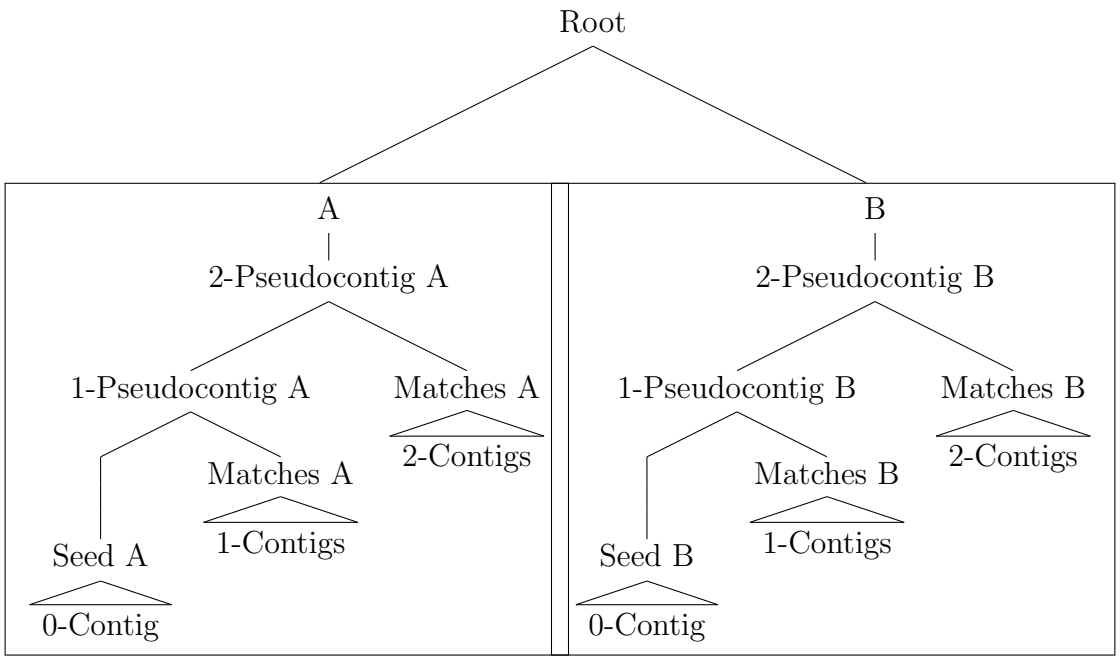

Figure 3.4: A cluster containing two merged clusters A and B

3.2.2 Parameters

3.2.3 Input

The input metagenome file must be a fasta-formatted file, but needn't be tab-delimited. If the input file is not tab-delimited, HORATIO will create a tab-delimited version containing exactly the same information, and that version will be available in the same folder as the original input file after completion. HORATIO expects that each contig will have a unique identifier, and that the contig will only be a string of DNA bases, with no quality information.

3.2.4 Output

There will be two output files. If the output argument is `<OUT>`, then the files will be `<OUT>_clusters` and `<OUT>_clusters_pickle`. `<OUT>_clusters` will be a human-readable text document containing a cluster's contig members on each line. `<OUT>_clusters_pickle` is a cPickle file containing all of the HORATIO cluster information, including the full tree and all scores for each and every cluster, designed to be used for further close analysis. Note

that HORATIO expects `<OUT>` to contain the desired file path for output files.

3.2.5 Cut schedule

The cut schedule is entered in the format $[x_k, x_{k-1}, ..., x_1, x_0]$, where $x_i \in \mathbb{N}$, and in reality reflects the schedule $[c_k, c_{k-1}, ..., c_1, c_0]$, where $c_i = x_i * 1000$. c_k denotes the lower limit on size for contigs processed by HORATIO; no contigs of size $< c_k$ will be considered. There is no upper limit on contig size, however; c_0 merely denotes the cutoff size for initial seeds, such that every contig with size $\geq c_0$ will be made a seed before the main loop starts.

3.2.6 Scoring Methods

Given that different scoring methods might be faster or better suited to a researcher's needs, HORATIO is designed to be able to use any given similarity-scoring method. The three methods that we built in and examined for this task were Raiphy, Tetra, and Tacoa, all of which are composition-based methods. These methods were introduced in 2.2.2.

In order to distinguish between the methods RAIphy, TACOA and TETRA and the scoring metrics used within those methods, we refer to the metrics as Raiphy, Tacoa and Tetra, respectively.

Raiphy If the user wishes to use the scoring function based on RAIphy, the user must have the version of raiphy available at `https://github.com/olsona/kiki`. This software performs exactly the same computations as the original version of raiphy, but outputs the results in a format usable by HORATIO. In order to use the raiphy scoring function, the user should specify `-s raiphy` and `-p <PATH>`, where `<PATH>` is the file path to the raiphy executables, contained in kiki/bin/.

Raiphy works by creating RAI (relative abundance index) profiles for each contig and genome it encounters. The profile is a vector of size 4^k, where k is recommended to be 7,

and each vector element corresponds to the relative frequency - and significance - of a given $k-$mer. In order to compare two contigs, the two contigs' RAI profiles are computed and then scored. It is worth noting that the raiphy program available for download appears to use different methods for computing scores and RAI profiles than is described in the original paper. The version of raiphy required for HORATIO uses the methods in the original downloadable code. [4]

Tetra Given the simplicity of the TETRA method, and the fact that no working TETRA executables are available, we wrote our own implementation of the Tetra similarity score in PERL. As might be expected from the name, Tetra examines over- and under-representation of $4-$mers. In particular, a contig profiled by Tetra will produce a 256-element vector, where each vector element contains the Z-score of a particular 4-mer. The Z-score computation is explained further in B.1. The similarity between two contigs is computed as the Pearson correlation coefficient of the two contigs' Tetra vectors.

Tacoa Like Tetra, and indeed Raiphy, Tacoa first computes a profile for each contig in the form of a vector of length 4^k, where each element in the vector contains a significance score for a given $k-$mer. In the case of Tacoa, $k = 4$. The over- or under-representation of a $k-$mer is computed relative to what would be expected given the contig's GC content. The similarity between two contigs is computed as a smooth function of the Euclidean distance between the two contigs' profiles. See B.2.

3.2.7 Neighboring Threshold

The neighboring threshold, denoted n, determines how close a cluster must be to a contig in order for it be considered a neighbor of that contig. More specifically, suppose that a contig co is closest to the cluster cl with score $s(co, cl) = b$. Then, another cluster cl' will be considered a neighbor of co if $s(co, cl') \geq (1 - n) * b$ if $b > 0$, and $s(co, cl') \geq (1 + n) * b$ otherwise. Therefore, higher values of n will result in more neighbors for each contig.

3.2.8 Joining Threshold

The joining threshold, denoted j, determines how frequently one cluster must appear in another cluster's neighbor list in order for them to be joined. If cluster cl has a neighbor list, in which cluster cl' is the most common and occurs with relative frequency f, then cl and cl' will be joined iff $f \geq j$. Note that only the most common cluster in a neighbor list is evaluated for possible merging.

3.2.9 Splitting Threshold

The splitting threshold, entered as a list $[l_k, l_{k-1}, ..., l_1]$, is a list that determines how close a contig must match to a cluster before it can be added. If, in stage i, a contig co is closest to cluster cl with score $s(co, cl) = b$, then co will only be added to cl if $b \geq y_i$. The splitting threshold is a list because as contigs get smaller and clusters get larger, the score distribution changes.

3.2.10 Affinity Propagation Preferences

The AP preference is a string indicating what initial preferences should be used for Affinity Propagation. The higher the initial preferences are, the fewer clusters are found in the final result. All items can be defined to have the same initial preference, indicating that they are all equally qualified to be exemplars, or items can have different preferences based on certain qualities. For the first case, the options 'min', 'median', 'mean' and 'max' are available, assigning initial preferences based, respectively, on the minimum, median, mean and max values of the similarity scores between the initial clusters. Also available are '40', '60', '70', '80' and '90', corresponding to the 40, 60, 70, 80 and 90% quantile for the similarity scores. For the second case, assigning preferences based on qualities inherent to the initial clusters, the user can choose 'len', 'len2', 'len3' or 'len4'. The option 'len' corresponds to scaling the lengths of the root (pseudo)contigs to the interval occupied by the similarity scores, so that

the smallest cluster (i.e., with the shortest root (pseudo)contig) has a preference equal to the least similarity score, and the largest cluster has a preference equal to the greatest similarity score. The options 'len2', 'len3' and 'len4' mean that the second, third or fourth power of the root contig is scaled. See B.3.

3.2.11 Minimum Number

One option for increasing computing speed is to reduce the number of data points that AP receives by filtering out small clusters. k is the minimum number of contigs that a cluster must have in order for that cluster to be included in AP. If k is set to 1, all clusters at the end of the main loop are included in AP. Otherwise, if $k \geq 2$, singleton clusters are excluded, thus speeding up AP computation, but reducing sensitivity.

Chapter 4

Evaluation

4.1 Testing Sets

One of the major problems of taxonomic binning is in testing an algorithm's accuracy without having ground truth for genuine data [2]. It is possible to produce simulations of metagenomic data where the provenance of each contig is known.

4.1.1 FAMeS

The Fidelity of Analysis of Metagenomic Samples (FAMeS) dataset [2], has been called the "gold-standard" for evaluating the performance of various metagenomics analysis algorithms" [10]. Each contig has known provenance, thereby allowing correctness testing to be performed.

The dataset consists of a set of simulated reads from different sources (high-, medium- and low-complexity, corresponding to what would be seen in soil, acid mine drainage, and sludge, respectively), and contigs built from those reads using different assemblers (Arachne, phrap and Jazz). The Jazz assembly process produces scaffolds, which are not usable by HORATIO, and so the Jazz data were discarded. The Arachne and phrap assemblies produced contigs

that were far too small to be useable, since the compositional signal from small contigs is insufficient for accurate matching.

4.1.2 New FAMeS

We therefore created a very large simulated metagenomic dataset where the identity of each contig was known. First, a set of complete genomes was downloaded from the NCBI Genome Database [39]. The set of genomes was drawn from the FAMeS set [2]: that is, if a complete genome was found to be a member of the FAMeS set, it was added to the NewFames set. Therefore, NewFames had 38 genomes, comprising 33 genera.

In order to examine the efficacy of HORATIO for metagenomes of different complexity, we looked to FAMeS for further inspiration. Using the estimated coverage for each simulated set - high-, medium-, and low-complexity - we built coverage distributions to be used on our smaller set of 38 genomes. To account for the increase in coverage in the 7 years since FAMeS was developed, the estimated coverage was multiplied by a constant such that the lowest coverage in our distributions was set to 60, or 200x the lowest coverage in the original FAMeS set, whichever was higher. The contig distributions are seen in Table 4.1. We made two metagenomes for each complexity level, called HC.1, HC.2, MC.1, MC.2, LC.1 and LC.2.

Simulated Set	Number of Genomes	Coverage
High complexity	1	165
	1	110
	36	90
Medium complexity	1	1735
	1	680
	1	100
	35	60
Low complexity	1	1285
	1	160
	36	60

Table 4.1: Contig length distributions for synthetic metagenomic sets

4.2 Definitions

Specificity While an ideal output would consist of one pure cluster per taxon, it was decided that purity, also known as specificity [23], would be more important than larger clusters. That is, an output of small but consistent clusters, with multiple clusters per taxon, is better than an output of large inconsistent clusters.

Purity per cluster, or precision, is a very simple calculation. Let N_i be the number of elements in the cluster i, and let m_i be the number of the elements from the most-represented taxon in the cluster. Then the purity of cluster i is [23]:

$$p_i = \frac{m_i}{N_i} \tag{4.1}$$

If there are N elements represented in the clustering, then the total purity is:

$$P = \sum_i \frac{m_i}{N} \tag{4.2}$$

A modified precision measure looks not at each cluster individually or at all clusters en masse, but at classes and the clusters that make them up. Let j be a class, let TP_j be the number of contigs in class j that were found in majority-j clusters (where purity $\geq$ some threshold), and let N_j be the total number of contigs that were found in majority-j clusters. Then the purity of class j is:

$$Sp_j = \frac{TP_j}{N_j} \tag{4.3}$$

Not all elements (i.e., contigs) in a taxonomic clustering are equal. It seems intuitive that it is more important to correctly classify longer contigs than shorter contigs, given that longer contigs are more likely to hold a useful genetic signal. Therefore, TP_j is equal to the combined length of true positive contigs, and N_j is equal to the combined lengths of all of

the contigs from source j. For our purposes, we computed overall specificity as:

$$Sp = \frac{\sum_j TP_j}{\sum_j N_j} \tag{4.4}$$

Sensitivity The sensitivity, or the recall, is a reflection of how well the algorithm can amass contigs correctly; specifically, greater sensitivity for a given class of contigs corresponds to fewer of those contigs being found in non-majority clusters. Let Z_j be the number of contigs that come from class j, and let TP_j be the number of contigs that were found in clusters with suitably high representation of contigs of class j (that is, the number of true positives). Then the sensitivity of the algorithm with respect to class j is [23]:

$$Sn_j = \frac{TP_j}{Z_j} \tag{4.5}$$

Therefore, total sensitivity is computed:

$$Sn = \frac{\sum_j TP_j}{\sum_j Z_j} \tag{4.6}$$

As above, TP_j and Z_j is equal to the combined lengths of contigs.

It is important to note that sensitivity and specificity were computed over all non-singleton clusters in the result; that is, TP_j and Z_j reflect the numbers of contigs found in non-singleton clusters, and N_j reflects the ideal number of contigs in a class j. As a result of this, not all contigs in the final clustering are considered in the computation of sensitivity and specificity.

Mutual Information Another approach to classifying clustering accuracy has been information-theoretic in nature. The basic premise is that if two clusterings are similar, then the information provided by one is similar to the information provided by the other, and thus the mutual information between them is high. Suppose we have two clusterings of N elements, $\mathbf{U}$ and $\mathbf{V}$, where $\mathbf{U}$ has R clusters and $\mathbf{V}$ has C clusters. Let i range from 1 to R, such

that U_i is a cluster in $\mathbf{U}$. Then $P(i) = |U_i|/N$. Likewise, if V_j is a cluster in $\mathbf{V}$, then $P(j) = |V_j|/N$. First, the entropy H of a clustering $\mathbf{U}$ is defined [40]:

$$H(\mathbf{U}) = -\sum_{i=1}^{R} P(i) \log P(i) \tag{4.7}$$

Let $P(i,j) = |U_i \cap V_j|/N$. Then the mutual information between clusterings $\mathbf{U}$ and $\mathbf{V}$ is defined [41, 40]:

$$I(\mathbf{U}, \mathbf{V}) = \sum_{i=1}^{R} \sum_{j=1}^{C} P(i,j) \log \frac{P(i,j)}{P(i)P(j)} \tag{4.8}$$

Mutual information may not tell the whole story, since once clustering may have much greater information/entropy than the other. Thus, the normalized mutual information is defined [40]:

$$NMI(\mathbf{U}, \mathbf{V}) = \frac{I(\mathbf{U}, \mathbf{V})}{\sqrt{H(\mathbf{U})H(\mathbf{V})}} \tag{4.9}$$

Unfortunately, even normalizing the mutual information may be insufficient, if chance is considered. Suppose that we are given a and b, vectors representing the number of elements in each cluster of $\mathbf{U}$ and $\mathbf{V}$. Given this, we can calculate $E\{I(M)|a,b\}$, that is, the expected information between two random clusterings with the prescribed number of elements in each cluster. This gives the adjusted mutual information measure [40]:

$$AMI(\mathbf{U}, \mathbf{V}) = \frac{I(\mathbf{U}, \mathbf{V}) - E\{I(M)|a,b\}}{\sqrt{H(\mathbf{U})H(\mathbf{V})} - E\{I(M)|a,b\}} \tag{4.10}$$

V-Measure The V-measure takes another information-theoretic approach to cluster similarity quantification, by looking at the harmonic mean between a homogeneity score and a completeness score. Homogeneity is analogous to purity/specificity, and completeness is analogous to sensitivity. Suppose $\mathbf{U} = \{U_i | i \in \{1, .., R\}\}$ is the true clustering of N elements, and $\mathbf{V}$ is the computed clustering. Let a_{ij} be the number of elements in cluster U_i and in cluster V_j. From equation 4.7, we have a definition for the entropy of any given clustering.

We then define the conditional entropy of one clustering on the other:

$$H(\mathbf{U}|\mathbf{V}) = -\sum_{i=1}^{R}\sum_{j=1}^{C}\frac{a_{ij}}{N}\log\frac{a_{ij}}{\sum_{i=1}^{R}a_{ij}} \tag{4.11}$$

Homogeneity is defined:

$$h(\mathbf{U},\mathbf{V}) = \begin{cases} 1 & \text{if the two clusterings are equivalent} \\ 1-\frac{H(\mathbf{U}|\mathbf{V})}{H(\mathbf{U})} & \text{otherwise} \end{cases} \tag{4.12}$$

Similarly, completeness is defined:

$$c(\mathbf{U},\mathbf{V}) = \begin{cases} 1 & \text{if the two clusterings are equivalent} \\ 1-\frac{H(\mathbf{V}|\mathbf{U})}{H(\mathbf{V})} & \text{otherwise} \end{cases} \tag{4.13}$$

Therefore, we have:

$$V_\beta(\mathbf{U},\mathbf{V}) = \frac{(1+\beta)*h(\mathbf{U},\mathbf{V})*c(\mathbf{U},\mathbf{V})}{(\beta*h(\mathbf{U},\mathbf{V}))+c(\mathbf{U},\mathbf{V})} \tag{4.14}$$

where β is the importance granted to completeness. If $\beta = 1$, both measures are weighed equally. If $\beta > 1$, completeness is more important. If $\beta < 1$, homogeneity is more important. For our purposes, $\beta = 1$, so:

$$V(\mathbf{U},\mathbf{V}) = \frac{2*h(\mathbf{U},\mathbf{V})*c(\mathbf{U},\mathbf{V})}{h(\mathbf{U},\mathbf{V})+c(\mathbf{U},\mathbf{V})} \tag{4.15}$$

Remember that in this case, $\mathbf{U}$ is the true clustering, and $\mathbf{V}$ is the computed clustering. [42]

Representation It is also important all genomes present in a sample are represented in a clustering. The representation value shows what fraction of original species have a cluster in which they are well represented, i.e., at least 90% of the contigs belong to that original species.

Chapter 5

Results

In order to minimize time spent on testing, we decided to first run a parameter sweep on all but AP preferences so that the parameters giving the highest purity could be found. Once those parameters were identified, we used those parameters to run another parameter sweep solely on AP preferences. Specifically, for each testing set, we looked at the following parameters:

Parameter	Options
Scoring measure	{Raiphy, Tacoa, Tetra}
Cut schedule	{[4,6,8,10,12,14,16,18], [4,6,8,10,14,18], [4,8,12,16]}
Joining parameter	{0.5, 0.7, 0.9}
Neighboring parameter	{0.01, 0.05, 0.10}
Splitting parameter	Raiphy: {-18.0,-17.5,-17.0,-16.5,-16.0,-15.5,-15.0} Tacoa: {0.2,0.24,0.28,0.32,0.36,0.4,0.44} Tetra: {0.3,0.35,0.4,0.45,0.5,0.55,0.6,0.65,0.7}

Table 5.1: Initial parameters

Below are detailed the results for each of the scoring metrics we used.

5.1 Specificity

The first, and perhaps most important, metric we looked at was specificity, described above (see equation 4.4). The stripcharts 5.1, 5.2, 5.3, 5.4 illustrate the distribution of specificities relative to different parameters.

One thing that is immediately obvious is that specificity is much higher for the HC set than it is for the MC or LC set. Cut schedule, N and J seem to have a minimal impact on specificity (see Figures 5.2, 5.3, and 5.4). For Raiphy and Tetra, the splitting coefficient has a marked effect on specificity (see Figures 5.7 and 5.5), but the same does not seem to hold for Tacoa (Figure 5.6).

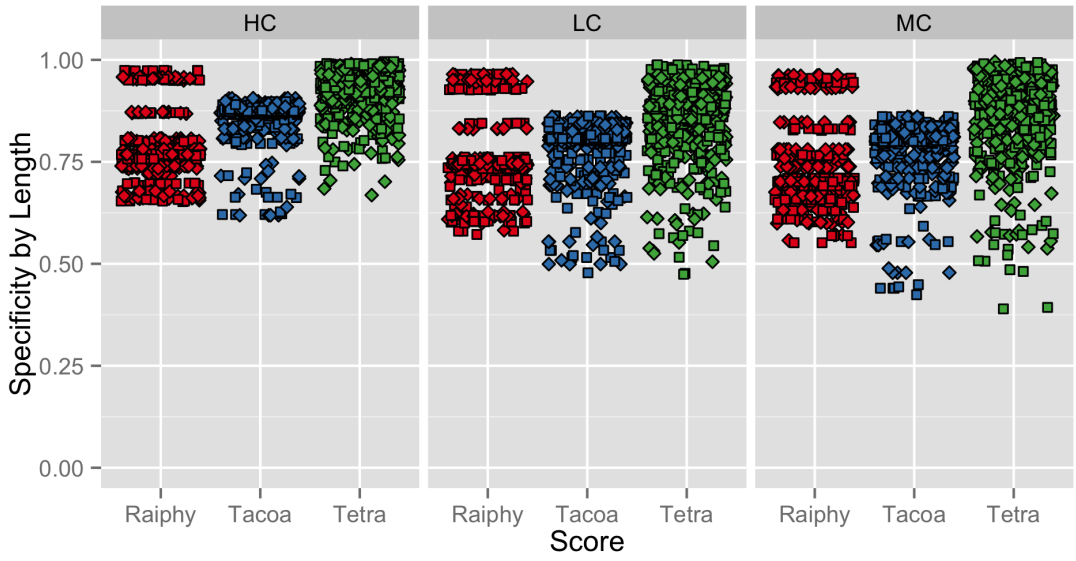

Figure 5.1: Scoring and specificity

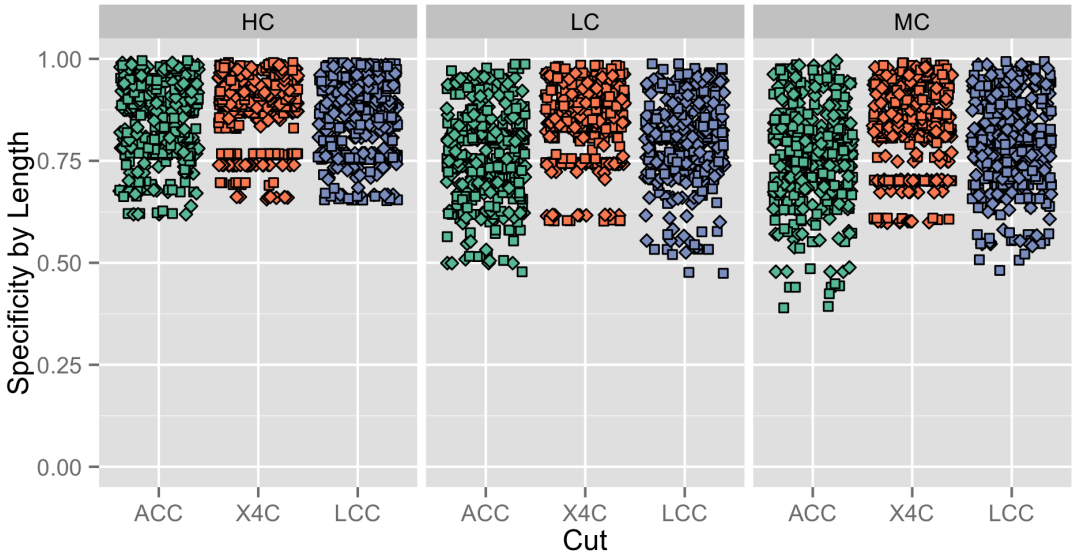

Figure 5.2: Cut schedule: ACC refers to [4,6,8,10,12,14,16,18], X4C refers to [4,8,12,16], LCC refers to [4,6,8,10,14,18]

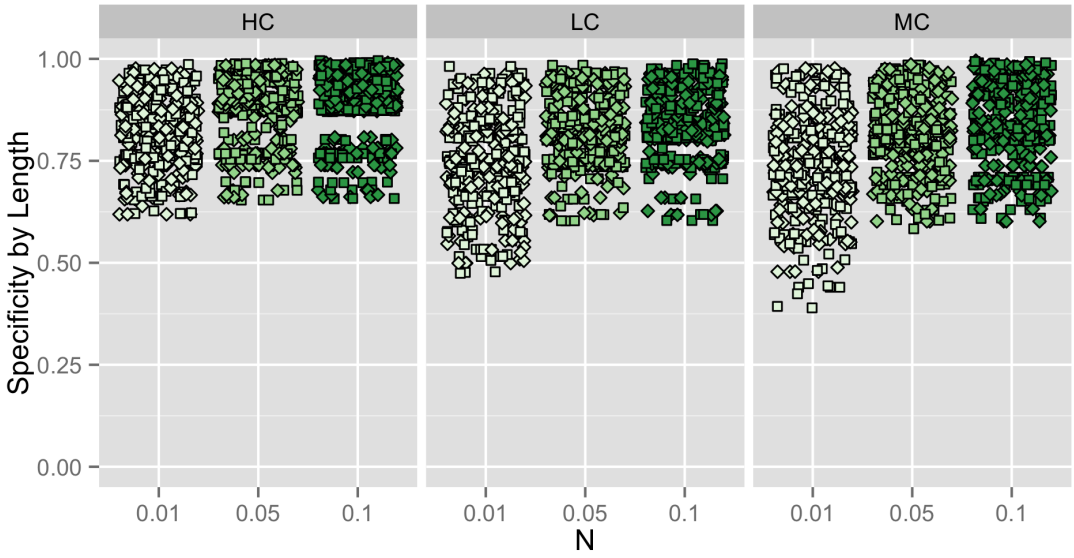

Figure 5.3: N

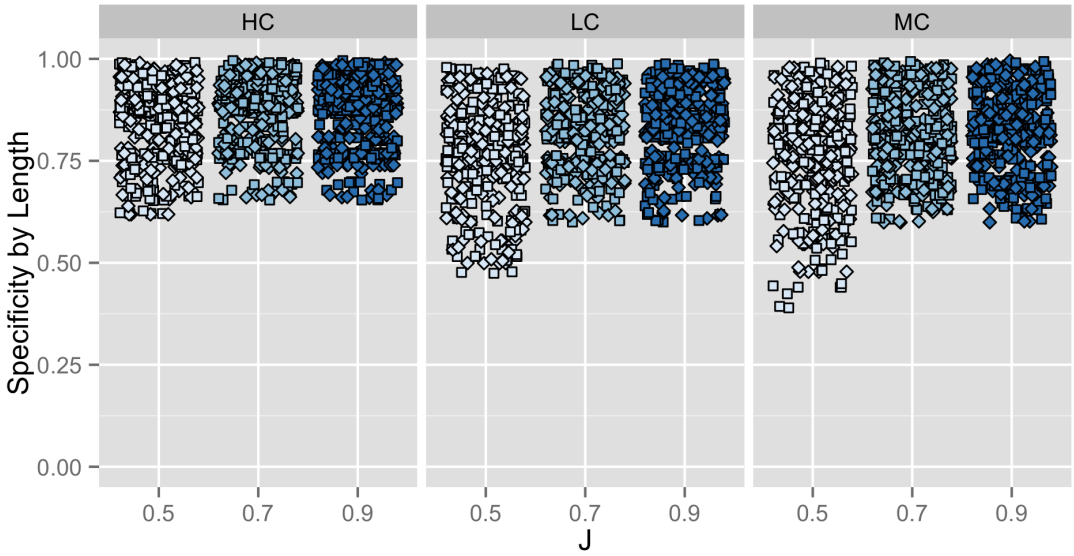

Figure 5.4: J

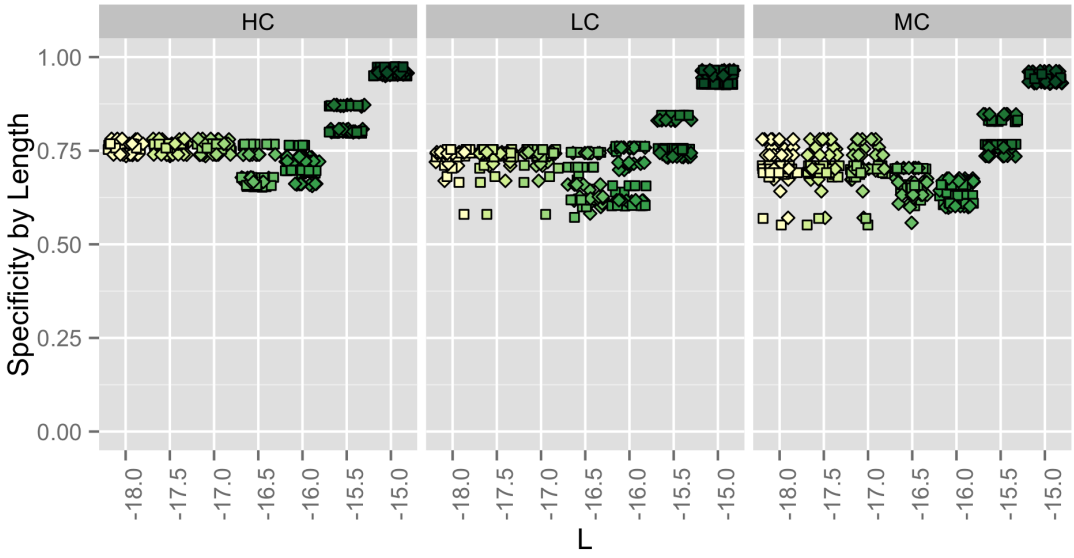

Figure 5.5: Splitting for Raiphy

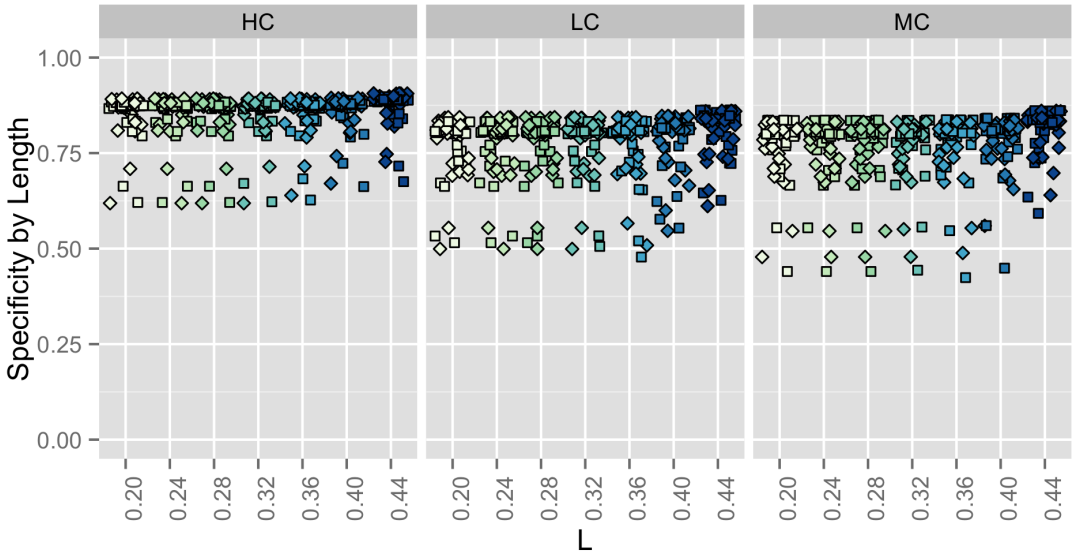

Figure 5.6: Splitting for Tacoa

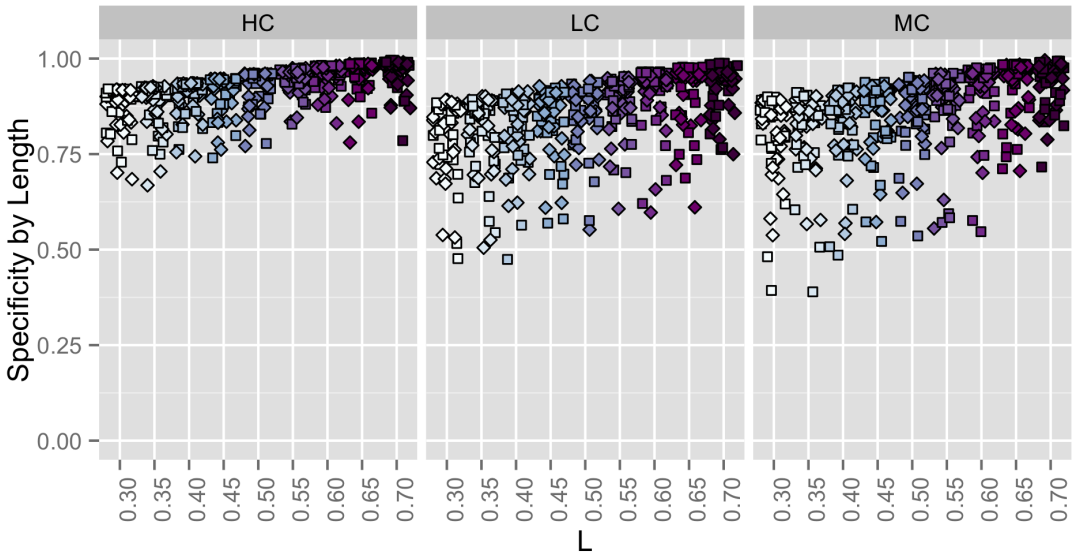

Figure 5.7: Splitting for Tetra

5.2 Sensitivity

As with specificity, sensitivity has a definite relationship with similarity metric (Figure 5.8) and with splitting coefficients (Figures 5.12, 5.13, and 5.14). Similarly, sensitivity does

not appear to reflect the cut schedule (Figure 5.9), the neighboring coefficient N (Figure 5.10), nor the joining coefficient J (Figure 5.11).

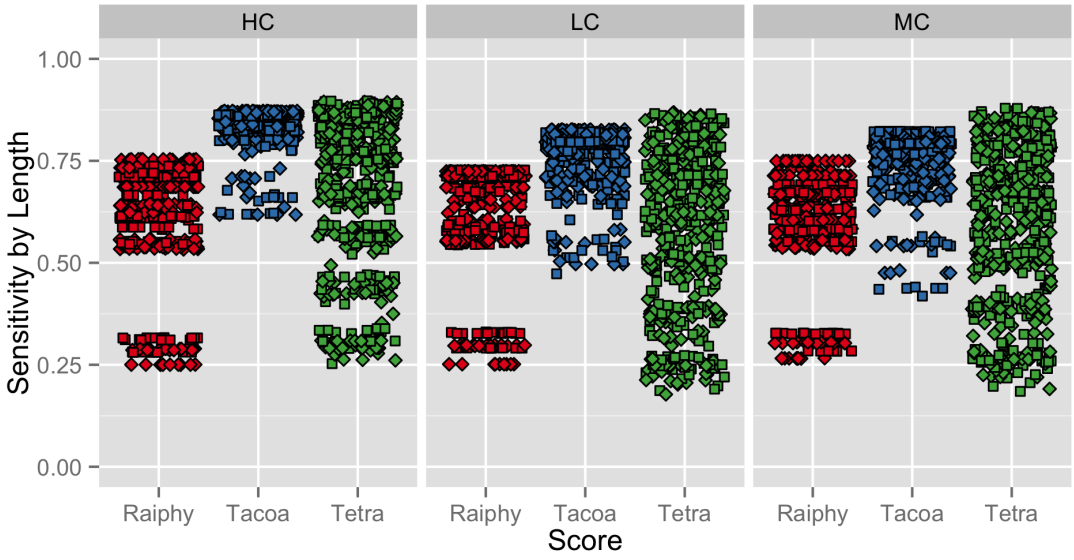

Figure 5.8: Sensitivity and scoring function

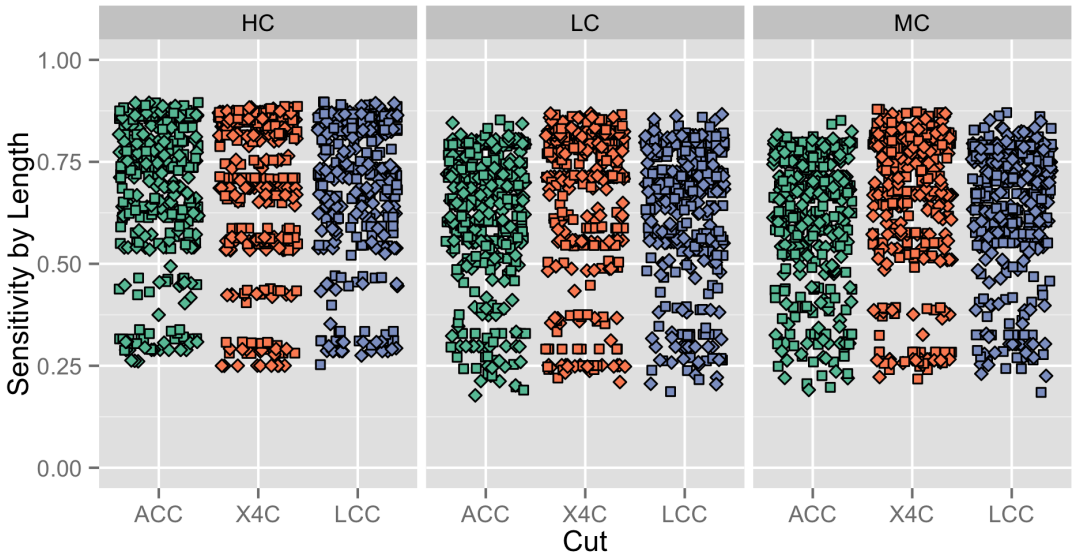

Figure 5.9: Sensitivity and cut schedule

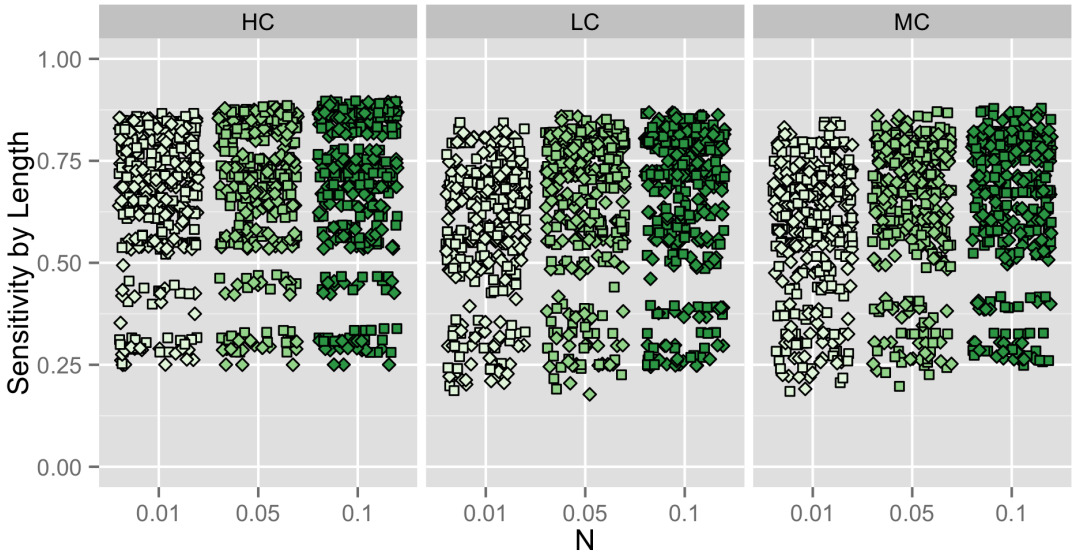

Figure 5.10: Sensitivity and N

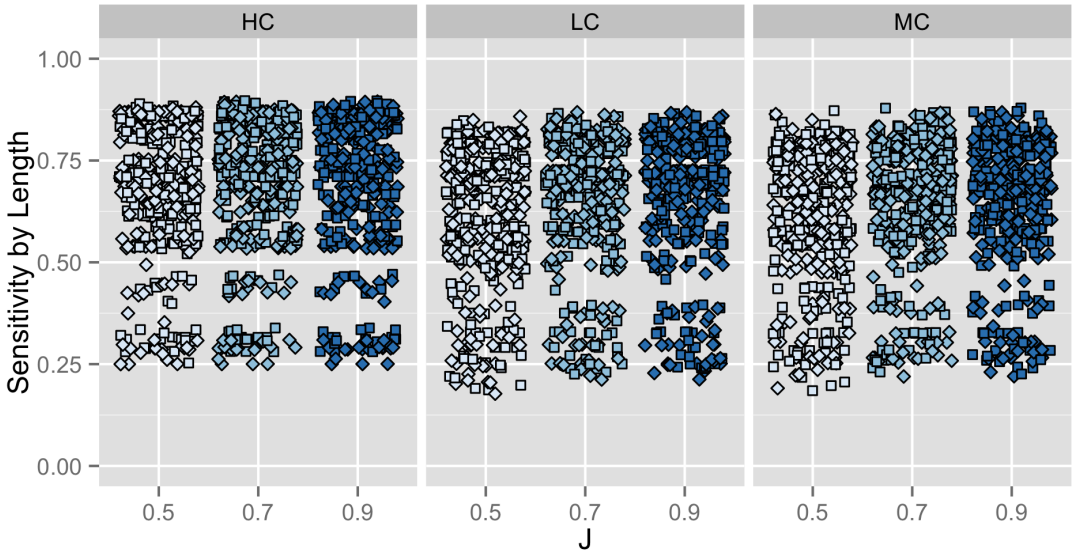

Figure 5.11: Sensitivity and J

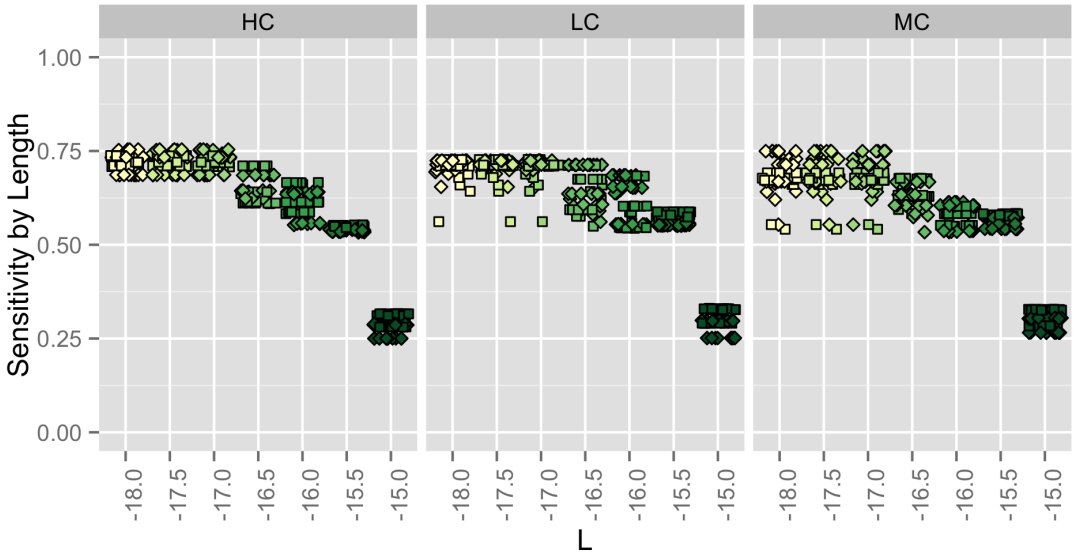

Figure 5.12: Sensitivity and L, Raiphy

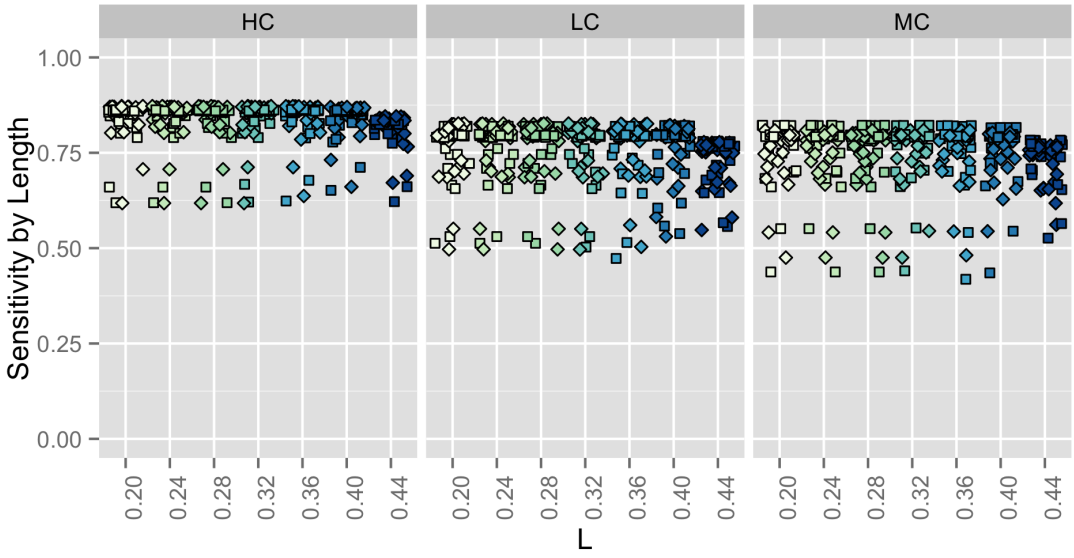

Figure 5.13: Sensitivity and L, Tacoa

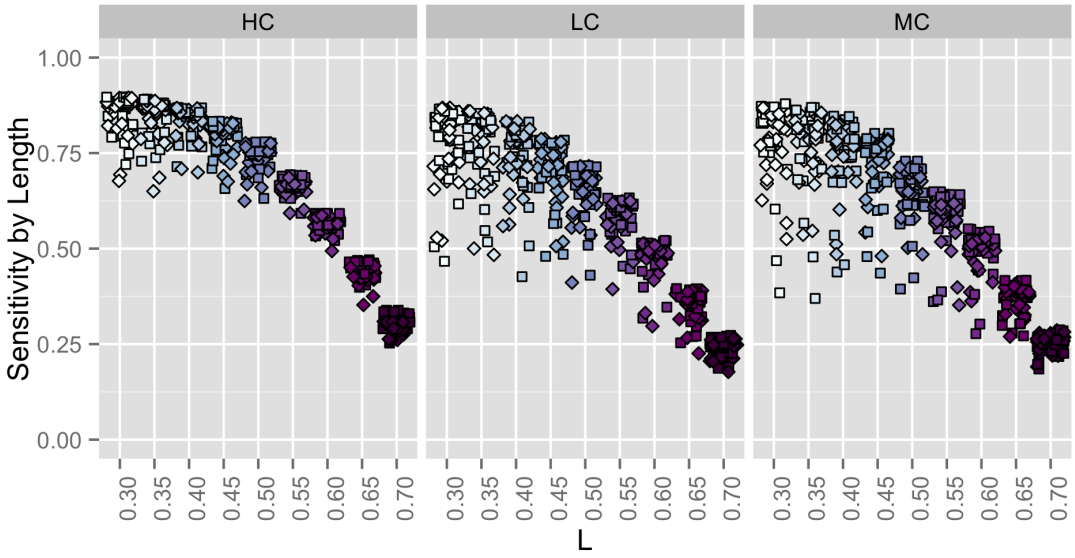

Figure 5.14: Sensitivity and L, Tetra

There does appear to be a positive relationship between sensitivity and specificity, when one controls for source. It is important to note, however, that the relationship is not perfect, and indeed the best linear relationships are obtained by further controlling for score mechanism and splitting threshold (see Figure 5.16). The clearest illustration of this is found by seeing that, for Tetra, specificity peaks around $L \in (0.5, 0.7)$, whereas sensitivity falls as L increases; this can be most easily seen in Figure 5.15. This suggests that if sensitivity is more important than specificity, then the researcher should tune the splitting parameter accordingly.

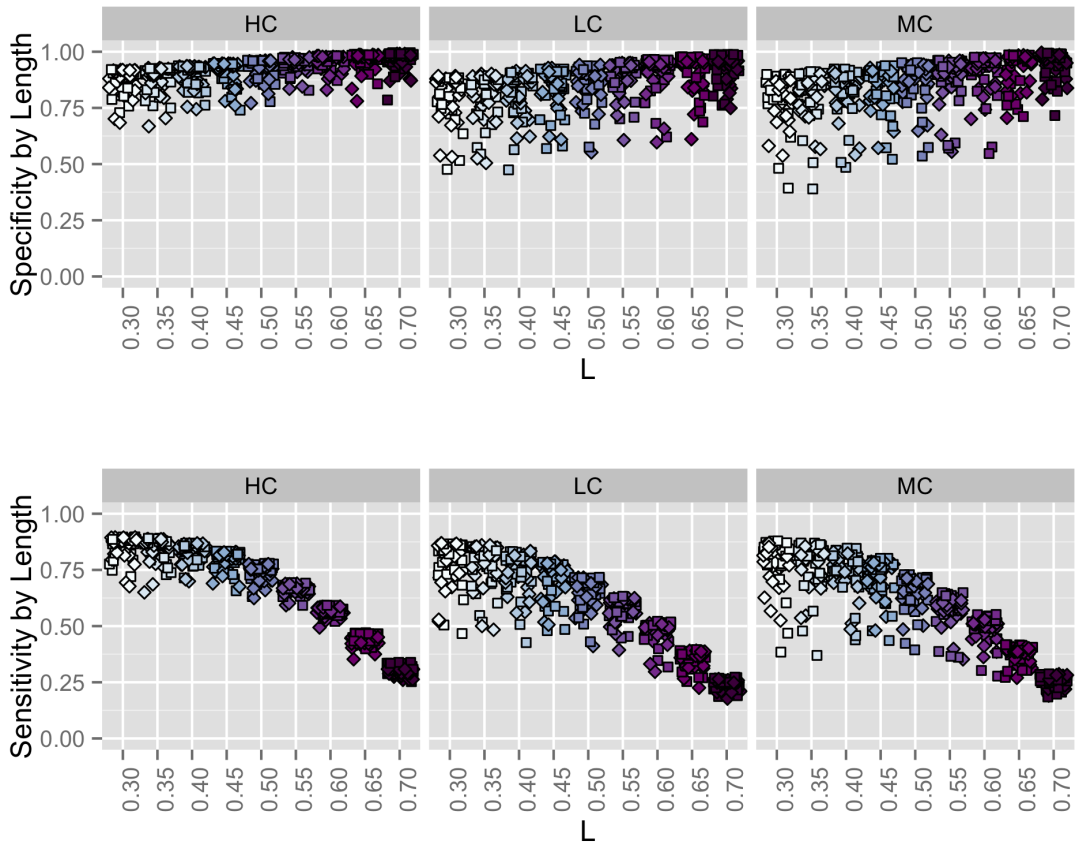

Figure 5.15: Specificity and sensitivity respond differently to L

It is also important to remember that sensitivity and specificity results are computed *only on non-singleton clusters*. While specificity is a reflection only of what appears in those non-singleton clusters, sensitivity is a reflection of what does not appear as much as it is a reflection of what does appear.

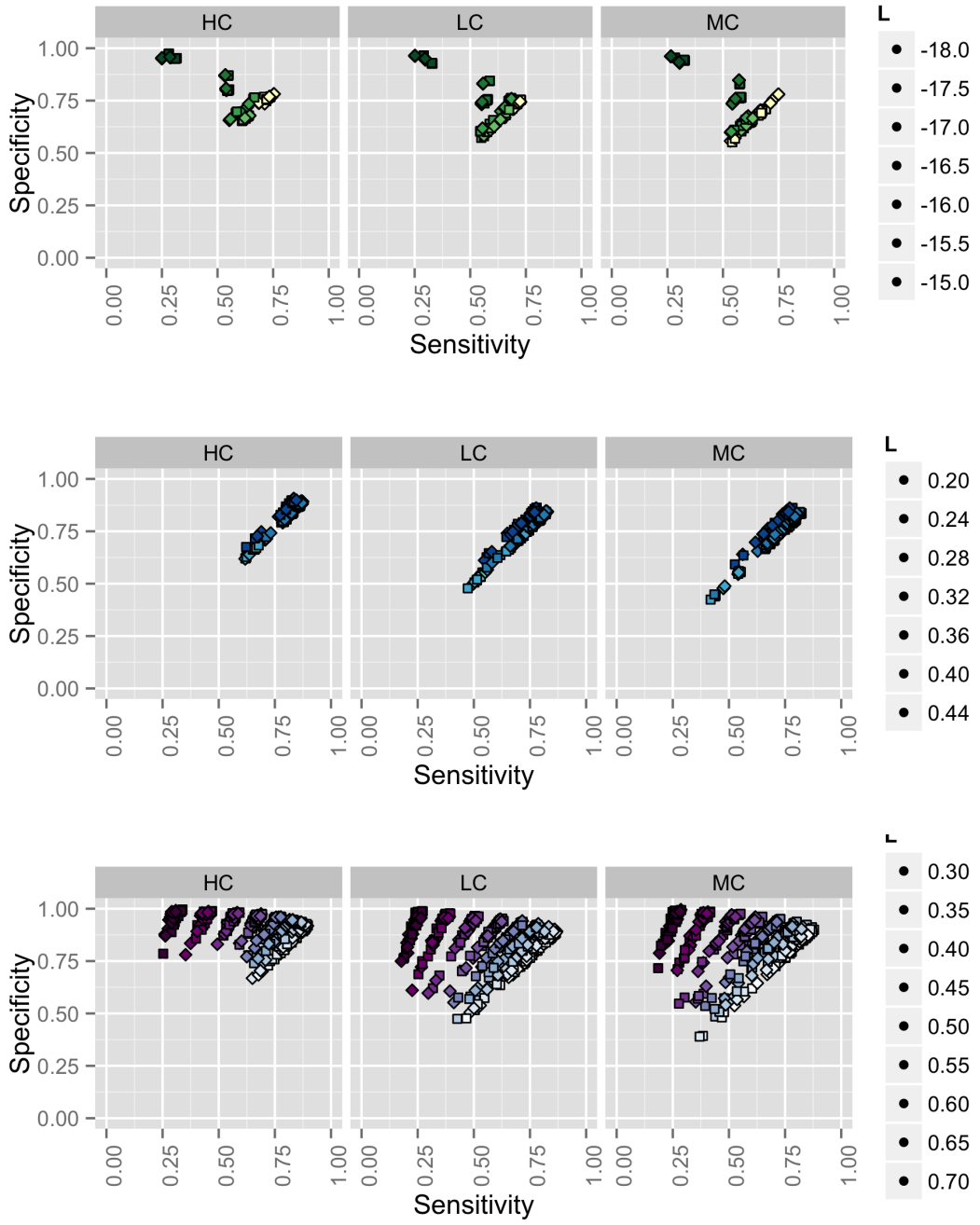

Figure 5.16: Sensitivity and Specificity vs. Source, Method, Splitting Parameter

5.3 Computation Time

At this point it was clear that Tetra was the superior similarity score. Since we wished to maximize accuracy and minimize computation time, we looked at Tetra computation logs

to see what parameters made HORATIO run faster or slower. We looked at cut schedule, N, J and L (Figures 5.17, 5.18, 5.19 and 5.20, respectively), and found that the greatest factor in computation time was the splitting parameter L. Therefore, we chose to restrict our subsequent tests to $L = \{0.50, 0.55, 0.60\}$.

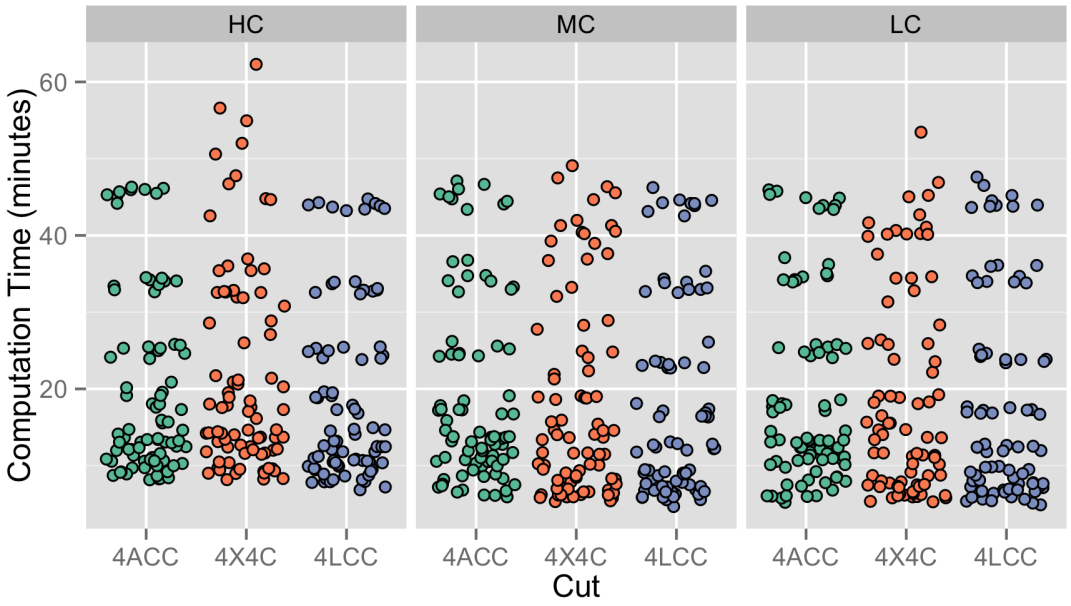

Figure 5.17: Computation time by cut schedule, for Tetra

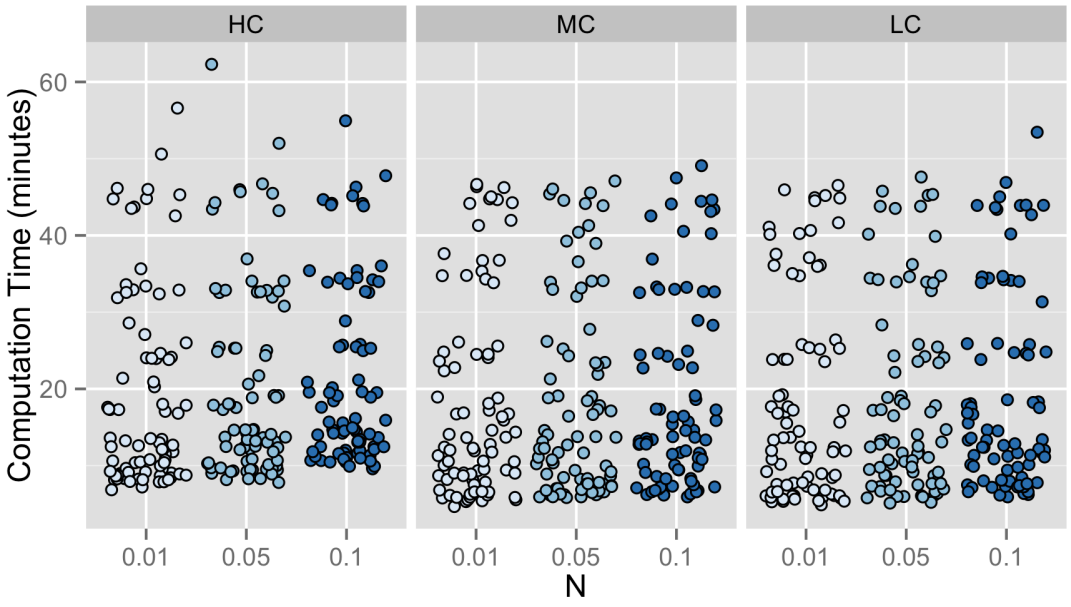

Figure 5.18: Computation time by N, for Tetra

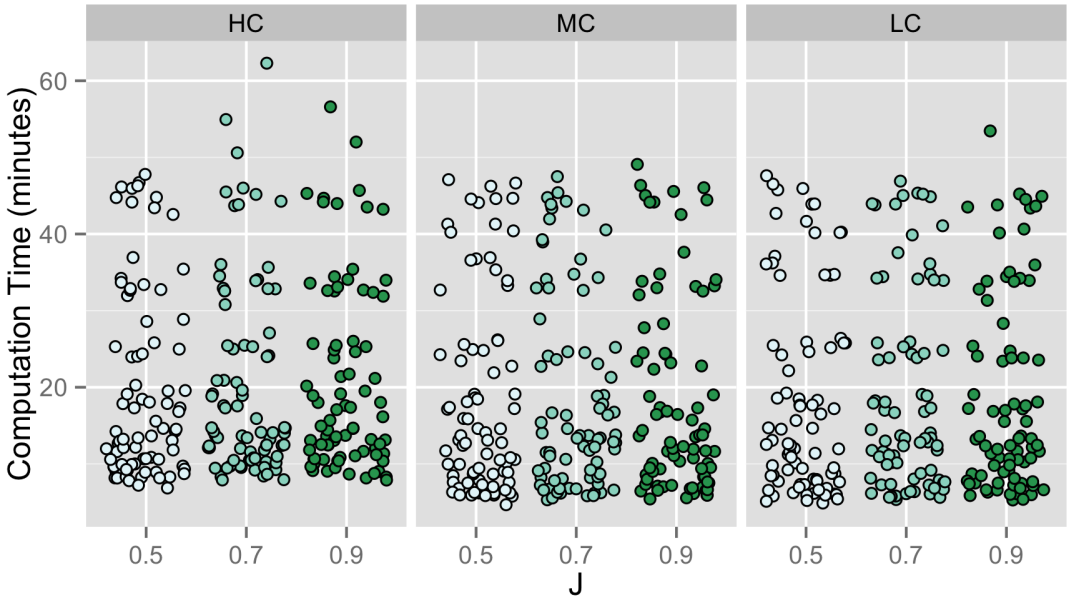

Figure 5.19: Computation time by J, for Tetra

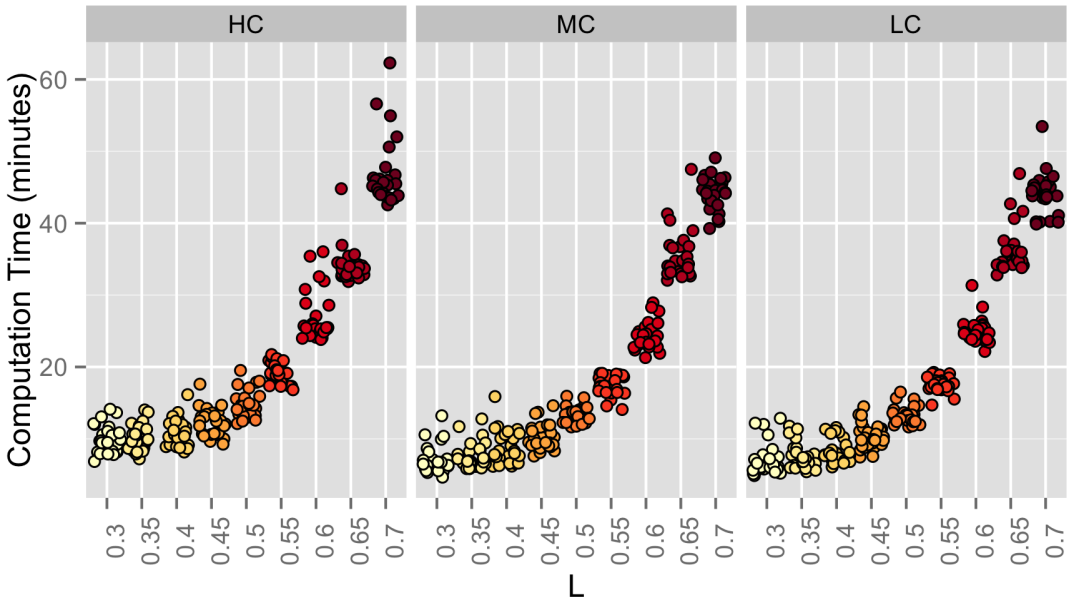

Figure 5.20: Computation time by L, for Tetra

5.4 AP Preferences

From these graphs, and from further analysis of the parameter sweep, and analysis of a more focused sweep on promising middle-group parameters, some promising sets of parameters began to emerge. Based on specificity, the following parameters merited further investigation:

Scoring measure	Tetra
Cut schedule	[4,6,8,10,12,14,16,18]
Joining parameter	{0.5,0.7,0.9}
Neighboring parameter	0.1
Splitting parameter	{0.50,0.55,0.60}

Table 5.2: Optimal parameters after initial sweeps

We also decided to investigate the possible use of Affinity Propagation for further clustering, and chose to examine the following inputs.

AP input	{Singleton, non-singleton}
AP preferences	{none, min, median, mean, max, 40, 60, 70, 80, 90, 95, len, len2, len3, len4}

Table 5.3: Affinity Propagation parameters

As with the initial sweep, we examined specificity and sensitivity as a function of AP preference. We also examined the number of clusters per genome, AMI, V score, and the fraction of represented genera. Given the qualitative difference between clustering based on equal preferences for each point and clustering based on unique preferences per point, the two types of preference assignment are displayed separately.

First, we see that AP preference has a marked effect on specificity (Figure 5.21). In particular, the preferences 'max', 'none' (N) and 'len4' seem to produce better results.

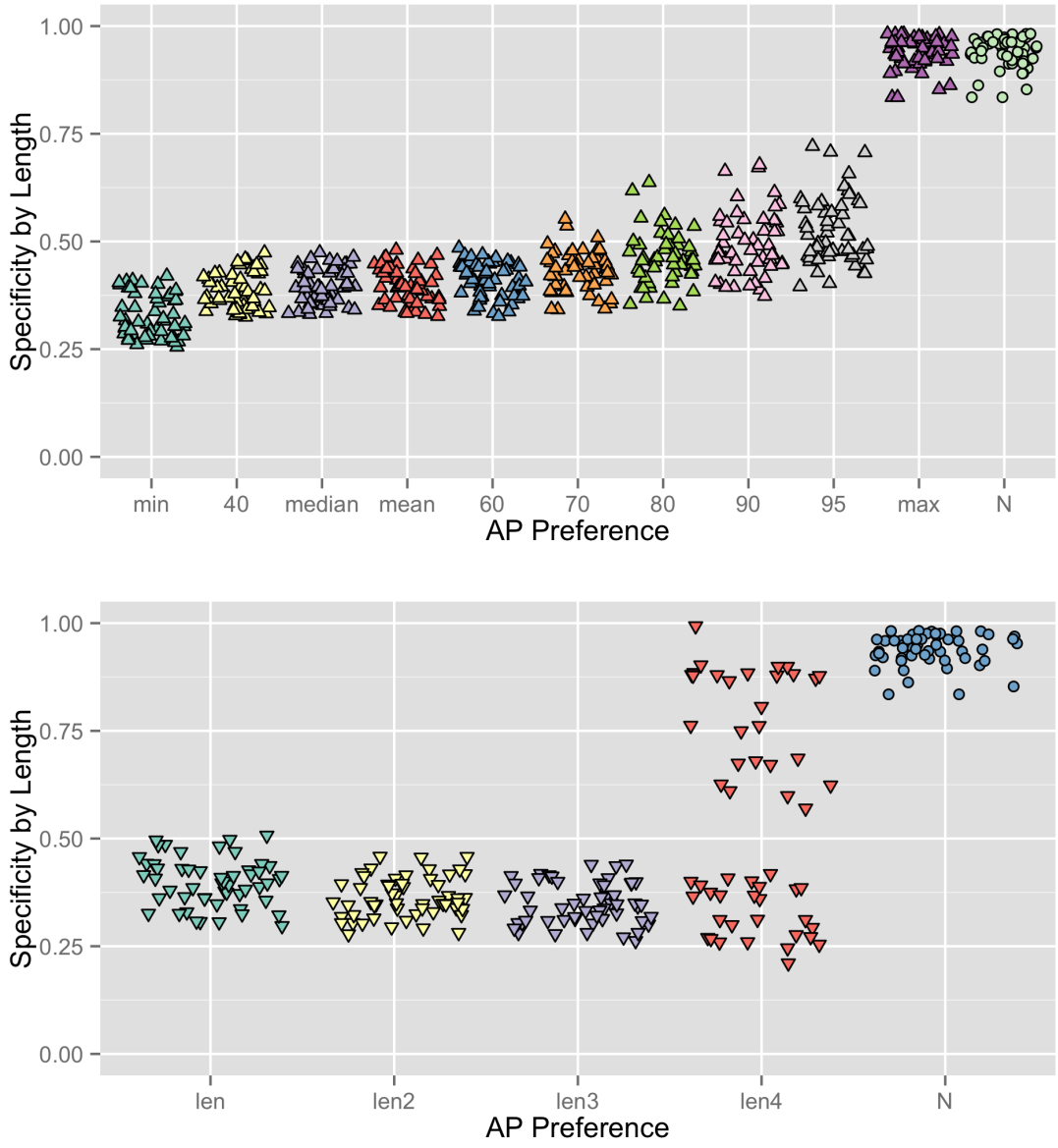

Figure 5.21: Affinity Propagation preference assignments affect specificity.

Next, we examine sensitivity in Figure 5.22. Again, notice that 'max', 'none' and 'len4' produce the highest sensitivities.

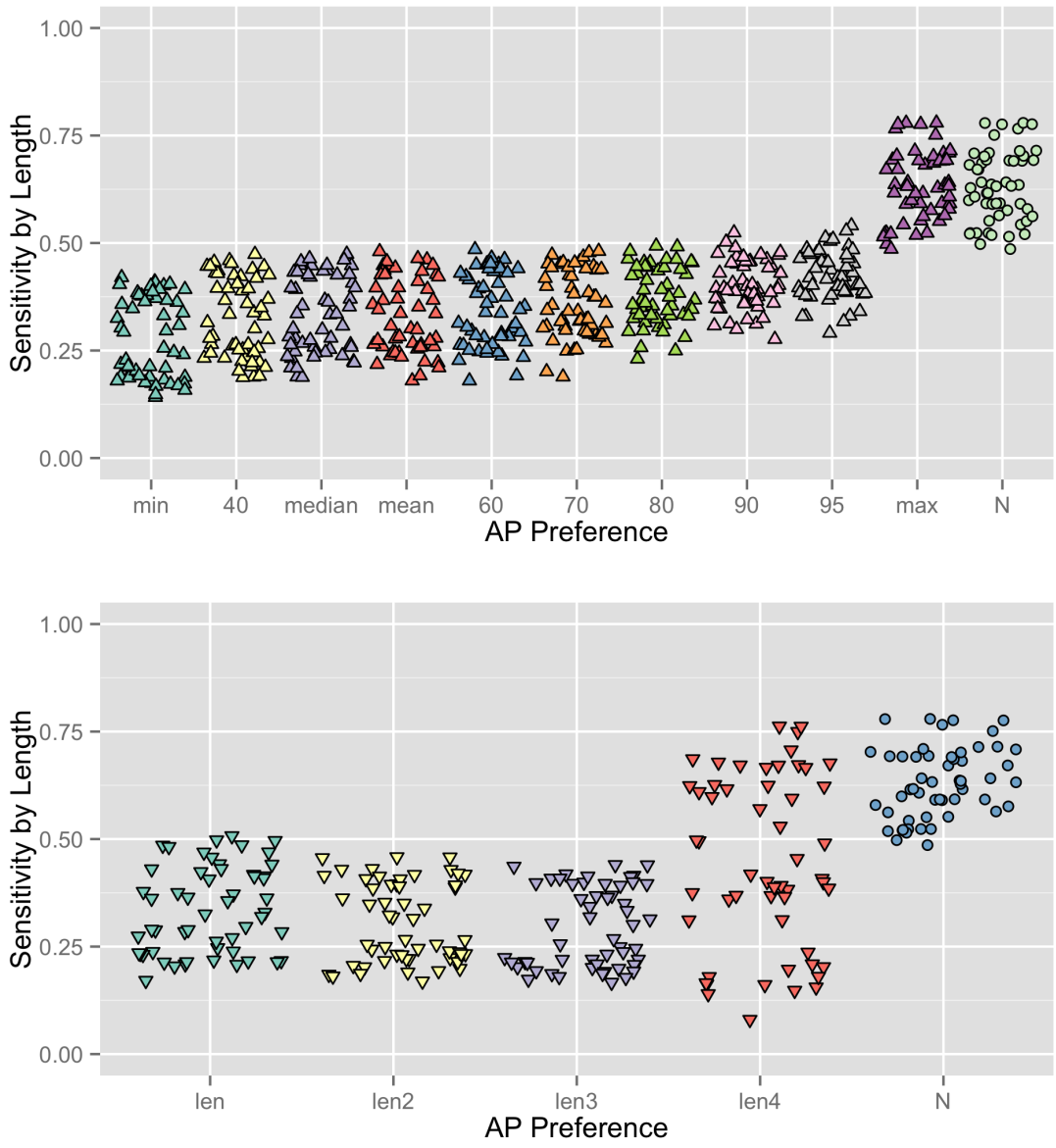

Figure 5.22: Affinity Propagation preference assignments affect sensitivity.

As mentioned in 4.2, mutual information-based measures can also be used as an aggregate measure for the quality of clustering. See 5.23 and 5.24. Consistent with 5.21 and 5.22, better results are obtained for 'max', 'none' and 'len4'.

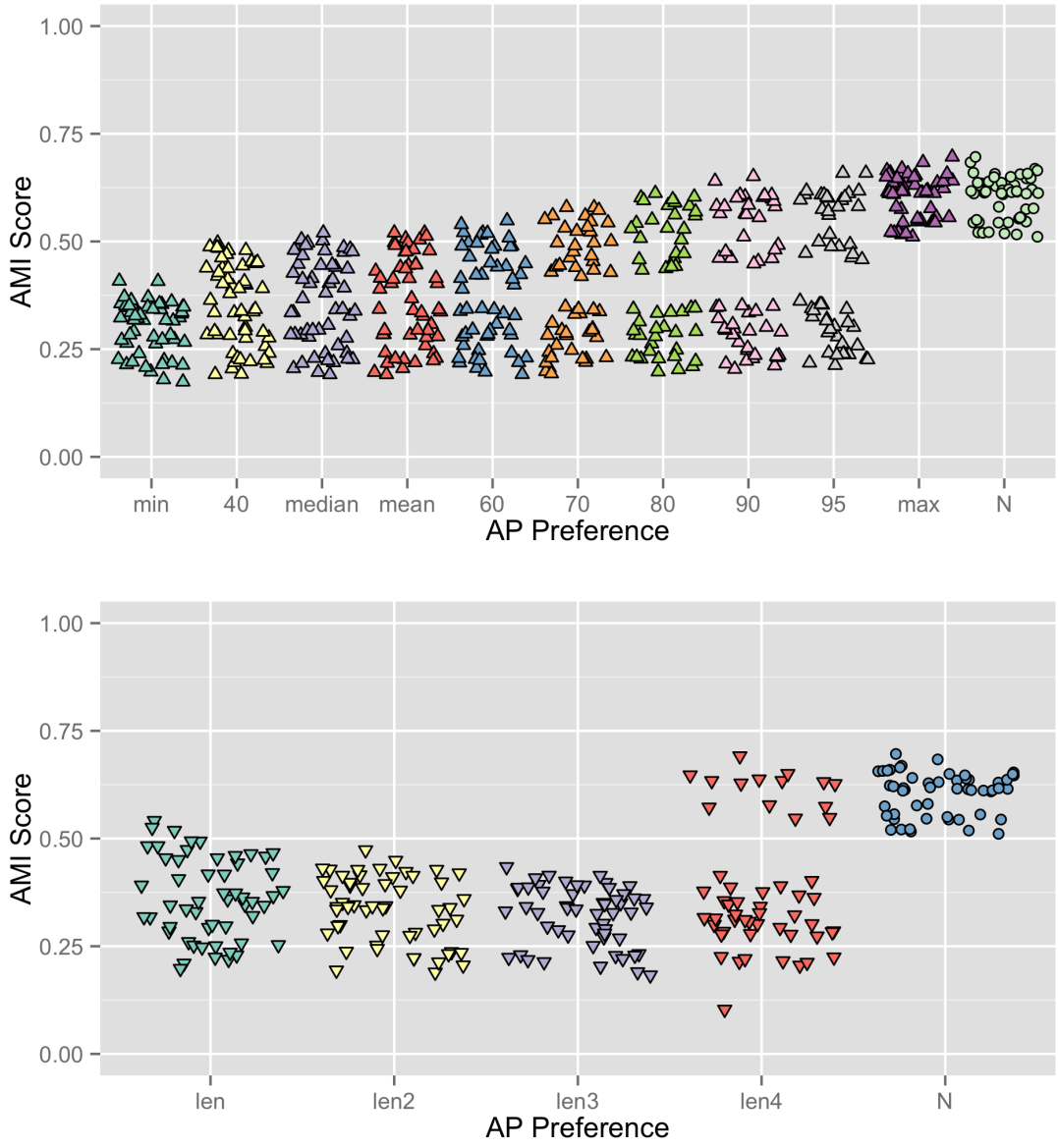

Figure 5.23: Adjusted mutual information as a function of AP preferences.

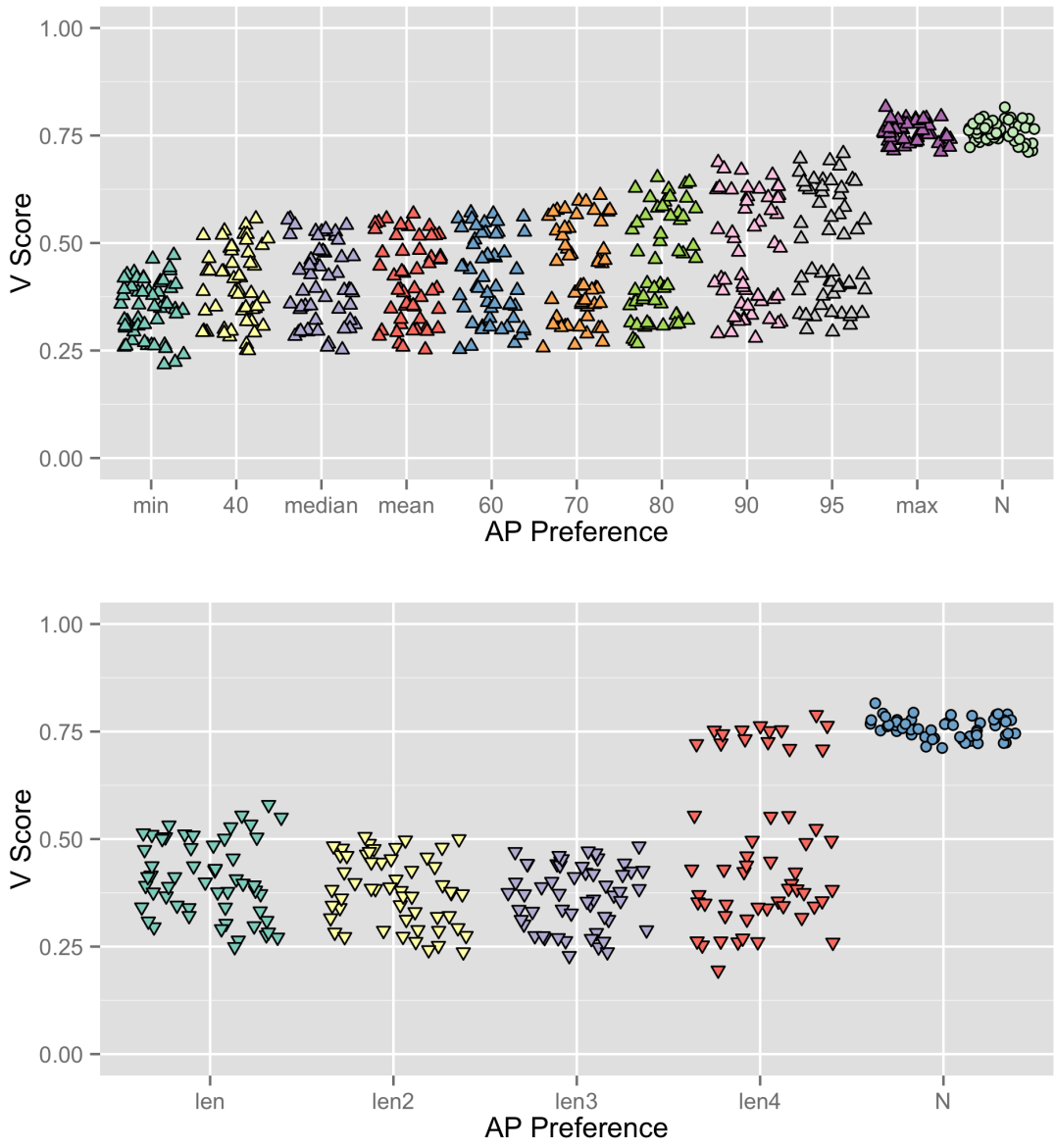

Figure 5.24: V score as a function of AP preferences.

Finally, we turn our attention taxonomic-specific values. In an ideal clustering, each clade at the appropriate level will be represented by a single cluster. In 5.26, we see that the top performers in terms of accuracy ('max', 'none' and 'len4') perform rather poorly in this regard. However, since we have deemed specificity and sensitivity to be far more important than the number of clusters per genome, this is not cause for worry. We also see in **??** that, for all preferences, a large number of genera have at least one cluster in which their contigs

make up at least 90% of the population. Interestingly, 'max', 'none' and 'len4' seem to have lower maximum values for representation than other preferences, but in general have larger values.

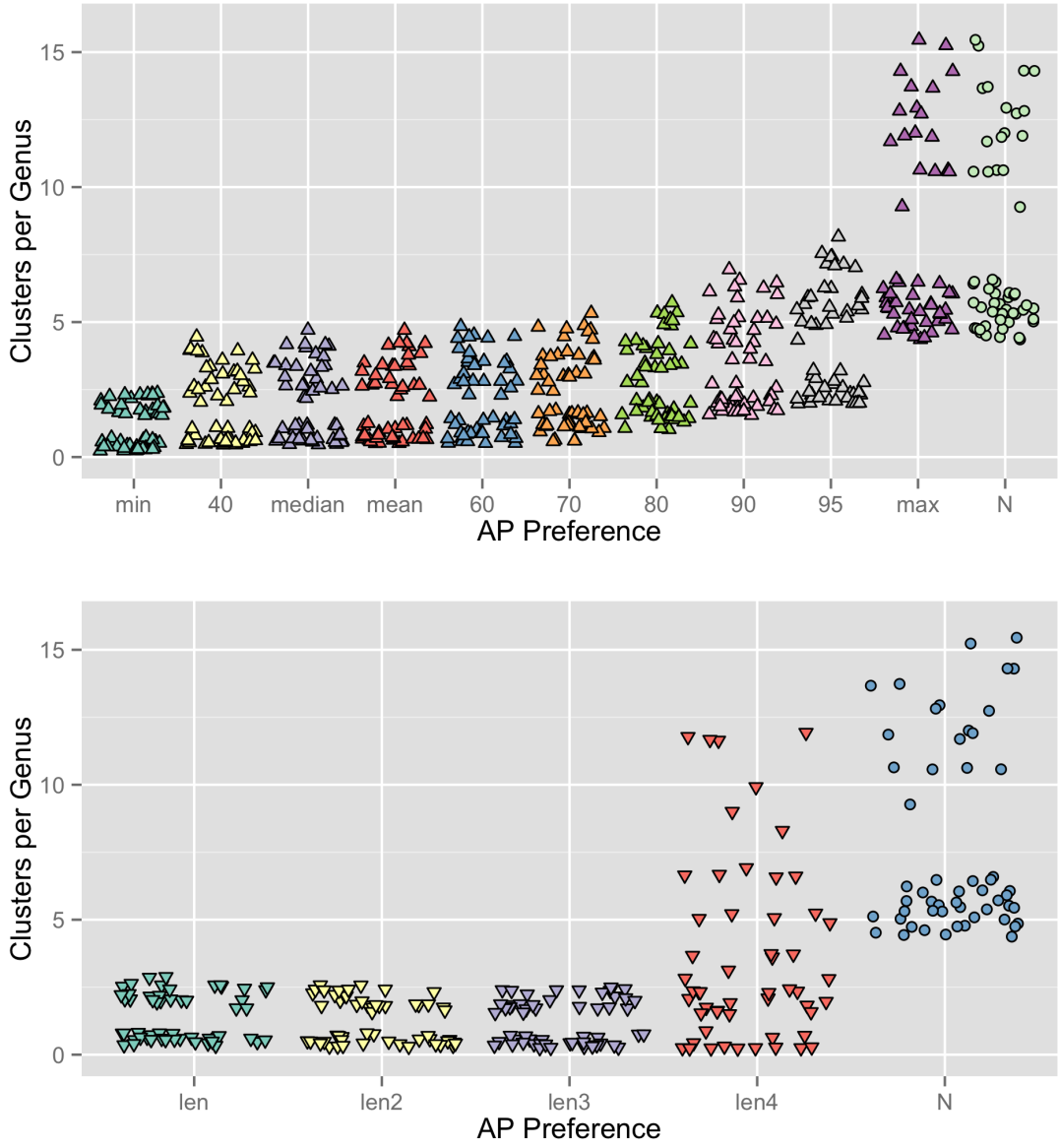

Figure 5.25: The number of clusters per known genus.

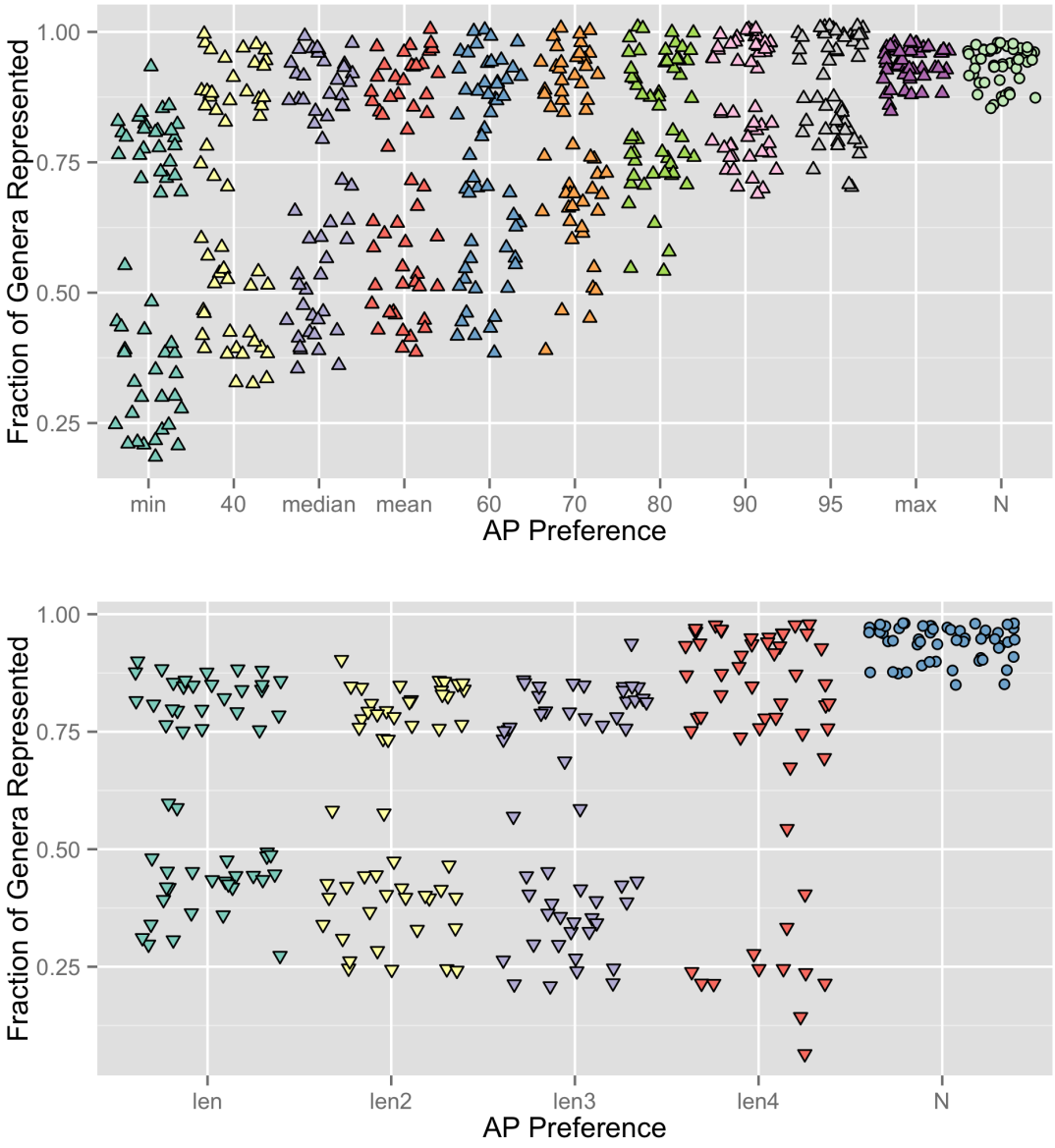

Figure 5.26: The number of genera represented in the final clustering.

The attentive reader will note that 'max' and 'none' have extremely similar results. In fact, setting the Affinity Propagation preference to 'max' is equivalent to allowing no Affinity Propagation at all. See B.3 for further explication of this.

No significant advantage was obtained through the use of Affinity Propagation. Indeed, the use of Affinity Propagation adds another 5 minutes to computation time while reducing

AMI, V score and sensitivity in all cases, and specificity in most cases. Therefore, the user is counseled not to use the AP option with HORATIO.

The best results, those that maximize AMI, V score and specificity, are obtained by using Tetra as the scoring function, with L=0.60 and J = 0.50. This provides representation values of at least 85%, specificity of over 90%, and V scores above 0.70.

Chapter 6

Discussion

6.1 Comparison With Other Methods

We tested the performance of standalone RAIphy, TACOA and TETRA on the same data sets we used for testing HORATIO. Using a large set of genomes, we built a database for each method. To run each method on each set, we first built profiles for each contig, and then computed the similarity. The results are listed in Table 6.1.

As HORATIO's metrics are computed relative to genus-level accuracy, we decided to do the same for the comparison metrics. For each contig, we found the database element to which it was most closely matched. If the contig and the database element were of the same genus, it was counted as a true positive; otherwise, it was a false positive. The number of clusters was the number of unique genera found in matching database elements. For comparison, there are 38 genomes, representing 33 genera, in each of these metagenomic sets. The time reflects the time taken to build the profiles for the contigs and the similarity computation.

Source	Score	Time	Sensitivity	Specificity	# of Clusters
HC	RAIphy	11m03.68s	0.18	0.72	45
HC	TACOA	10m27.21s	0.24	0.66	115
HC	TETRA	04m11.69s	0.61	0.86	212
MC	RAIphy	10m34.47s	0.19	0.73	44
MC	TACOA	10m07.38s	0.24	0.66	117
MC	TETRA	03m53.09s	0.59	0.84	211
LC	RAIphy	10m14.56s	0.19	0.75	47
LC	TACOA	10m02.30s	0.24	0.68	118
LC	TETRA	03m43.21s	0.68	0.86	211

Table 6.1: Comparison of reference-based algorithms

These results were obtained by ensuring that each source genome for a given metagenome was included in the input database. This is not a realistic assumption. Therefore, we also built databases where 10, 20, 50 and 80% of the source genera were included. By examining the representation fractions for each of these tests (shown in Figure 6.1), we see the major weakness of reference-based binning - the dependency on a well-populated database - laid clear.

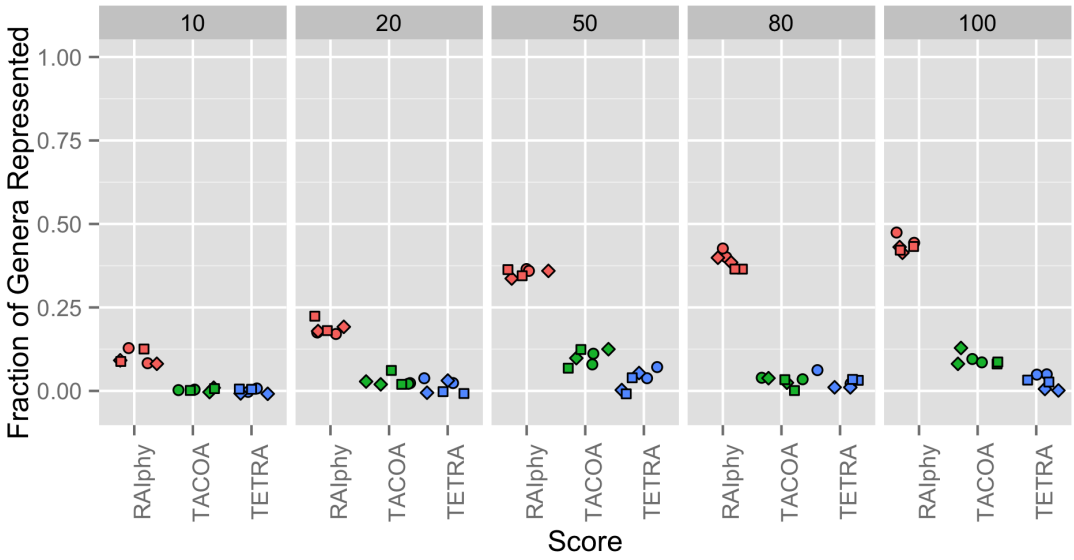

Figure 6.1: The quality of results for reference-based methods depends on the quality of the database.

We also uploaded a subset of the HC set to the INDUS service [8], with significant assistance from Monzoor Haque (personal correspondence, August 19 2014). The subset was much smaller than the original HC set, given the restrictions placed on contig size by the service. No results were incorrect per se, in that every contig was assigned to a clade containing its source species. Unfortunately, only a few contigs were assigned to the correct species or genus; the remainder were assigned somewhere between the family and phylum level. The INDUS service returned results for a 4MB file about half an hour after submission.

6.2 Strengths and Weaknesses

Compared to the other available reference-free method available for testing (ESPRIT-Tree; [35]), HORATIO is orders of magnitude quicker. While ESPRIT-Tree takes over a day to run on any of the simulated metagenomic sets used here, HORATIO takes closer to 15 minutes. HORATIO is slower than the fundamental methods employed (see Table 6.1), but it has the advantage of being reference-free (and thus being able to bin unknown contigs), and for high-complexity datasets, it has greater specificity and sensitivity. Therefore it could be used to further bin contigs for further assembly, and could help in the discovery of species in interesting microbiomes. It also has parameters that researchers can set to their personal preferences; for example, if a researcher is more interested in sensitivity than specificity when using Tetra as a scoring mechanism, they can simply set the splitting parameter lower.

On the other hand, HORATIO cannot provide taxonomy for any of the bins it identifies, and it is simply not as fast as other reference-based methods (Table 6.1). There is, as yet, no method for identifying "junk-heap" clusters, i.e., clusters composed of contigs from many different species. HORATIO may be difficult to set up, given the number of software packages it requires. Finally, HORATIO does not work well for medium- or low-complexity datasets. In such cases, the researcher is advised to use other algorithms, such as TETRA.

6.3 Future directions

6.3.1 Binning

Scoring

There appears to be some correlation between the score distributions within clusters and the purity of those clusters (Pearson correlation coefficient of around 0.5). It is highly likely that some aggregate or summary statistic of the score distribution of a cluster could be used to determine whether it is worthy of future analysis, or is instead probably a "junk-heap" of contigs from many different sources with no dominant member.

While the given metrics - Raiphy, Tetra and Tacoa - only look at contig composition, much more could be done with coverage information. This could mitigate the poor performance of HORATIO on medium- and low-complexity metagenomes. With dominant species present in a microbiome, coverage levels for contigs from different species would differ [6, 29]. Two contigs with similar composition profiles but different coverage profiles would be rightfully split, increasing performance.

More could also be done specifically with codon usage [26]. Codon usage differences appear largely in the final base pair, but this ordering information is lost in $k-$mer similarity scores. Codon usage is only applicable to genetic sequences that explicitly code for proteins, so this method would not be helpful for all metagenomic query sequences.

HORATIO could also be improved by using PCA to reduce the dimensionality of the composition profiles, as seen in AbundanceBin and CompostBin [26, 9]. Perhaps a PCA-based similarity metric would be optimal.

Lastly, alignment-based techniques could be used to improve HORATIO's accuracy without sacrificing too much speed [16]. The method of signature peptides is unfortunately restricted to protein-coding sequences, but there is no reason that the initial set could not be examined for known signatures before clustering.

Clustering

Many extant binning algorithms, such as ESOM, AbundanceBin and CompostBin show promising clustering techniques. Perhaps the initial clustering, made of the largest contigs, could be built using a self-organizing map like ESOM [30], and then $k-$means clustering would be used to assign smaller contigs to the extant clustering. Likewise, an expectation-maximization algorithm coupled with a mixture model, like that seen in AbundanceBin and CompostBin, could be used to create an initial clustering [26, 9].

Even staying with the simple HORATIO model, there is room for improvement. The method of joining clusters could perhaps be simplified to resemble MacQueen's method, and be based on nothing more than similarity between roots [37].

6.3.2 Evaluation

While we have many metrics at our disposal, no single metric shown in this work illustrates the full picture of the quality, and, more importantly, *biological usefulness* of a clustering. The F1 score is a possible starting point for aggregating information about sensitivity and specificity, and the relative importances of the two measures can be reflected in a weighted F1 score. The NMI, AMI and V score are also decent aggregate scores, but again, a direct relationship between clustering similarity and biological usefulness does not seem to have been established.

The testing sets, while a good starting point, do not fully reflect true metagenomic samples. It is not realistic to expect that there will only be one strain per species, or one species per

genus, in a real-world environment. The new FAMeS sets used in this work does have a few instances of two species per genus, or two strains per species, but this is still unrealistic: one would expect to find more strains per species. A better testing set would have a greater spread of species per genus or strains per species. Furthermore, 38 genomes in an environment is low [6], and a better set would have hundreds, if not thousands, of individual genomes represented [2]. Finally, a true metagenomic set would contain genomes of organisms that naturally co-occur, and the FAMeS (and new FAMeS) sets do not reflect that. Composition information might differ less (or, perhaps, more) between naturally co-occurring organisms' genomes.

6.3.3 Computation

The scoring code is single-threaded, but could be very easily parallelizable for greater speed. Suppose the database were divided into quarters, and the contigs to be matched against the database were divided into halves. Then the scoring could be split among 8 threads, and completed in a little longer 1/8 time (allowing for sorting of the final results).

Appendix A

HORATIO Usage

HORATIO is available at `https://github.com/olsona/Horatio`. Users can download it for their own use. In order for HORATIO itself to work, users must have Python 2.7+, along with Numpy and SciKit-Learn [43, 38]. Each scoring function has its own required package, detailed in later sections.

Use of HORATIO will look like this, with further argument details in Table A.1:

```
> python HORATIO.py -i <INPUT> -o <OUTPUT> -c <CUT> [ -s <SCORE> -p <PATH> -n
<NEIGHBOR> -j <JOIN> -l <SPLIT> -d <DOAP> -a <PREF> -k <MINNUM>]
```

`-i`	Input FASTA-formatted metagenome file.	3.2.3
`-o`	Prefix for all output files.	3.2.4
`-c`	Cut schedule. Defaults to [4,6,8,10,12,14,16,18].	3.2.5
`-s`	Scoring method. Defaults to 'tetra'. Other options include 'raiphy' and 'tacoa'.	3.2.6
`-p`	Computation path, i.e., path to scoring code for chosen scoring method. Necessary only if the score function is RAIphy.	3.2.6
`-n`	Neighboring threshold, i.e., the threshold for considering clusters to be neighbors to a contig, entered as a float between 0 and 1.	3.2.7
`-j`	Joining threshold, i.e., the threshold for merging two clusters, entered as a float between 0 and 1.	3.2.8
`-l`	Split threshold, i.e., is the threshold for letting a contig start its own cluster, entered as a list of floats ($[y_k, y_{k-1}, ..., y_1]$). Defaults to [-1000.0,...,-1000.0], which implies no splitting whatsoever.	3.2.9
`-d`	Indicator for whether AP is to be used. Defaults to 0 (no AP).	3.2.10
`-a`	AP clustering preference function, i.e., how to determine the initial preferences for each cluster going into AP clustering. Defaults to 'max'. Other options are 'min', 'median', 'mean', '40', '60', '70', '80', '90', '95', 'len', 'len2', 'len3' and 'len4'.	3.2.10
`-k`	Minimum size that a cluster needs to have in order to be included in the final AP clustering. The value can be any integer. Defaults to 1 (includes all clusters).	3.2.11

Table A.1: Usage details for HORATIO

Appendix B

Scoring computations

B.1 Tetra

The Tetra similarity metric given in the TETRA algorithm [22] is fairly simple, and is based on the idea that it is not overall frequency of a k-mer that matters, but the significance of its frequency given the frequency of its substrings. Here, $k = 4$, hence the name Tetra. Z-scores are computed for each k-mer to create a 4^k vector, and the similarity between two sequences is computed as the Pearson coefficient between their respective Z-score vectors. [22]

In Tetra, as in many other methods, sequence are viewed as the result of an n^{th}-order Markov chain model. In order to efficiently compute the Z-score for a given k-mer, one must make approximations [44]. One estimation for the Z-score is the martingale score [45]. Let W be a word in a genetic sequence, let $N(W)$ be the observed count of that word in the sequence, and let $N(w_x, ..., w_y)$ denote the observed count of the substring of W from position x to position y. Then, assuming that the sequence is given by a Markov chain of

order $m = h - 2$, the significance of W is given by:

$$z^M(W) = \frac{N(W) - \hat{\hat{N}}_m(W)}{\sqrt{\hat{\hat{V}}_m(W)}} \tag{B.1}$$

Where the estimated expected value of $N(W)$ is:

$$\hat{\hat{N}}_m(W) = \hat{\hat{N}}_{h-2}(W) = \frac{N(w_1, ..., w_{h-1}) * N(w_2, ..., w_h)}{N(w_2, ..., w_{h-1})} \tag{B.2}$$

And the estimated expected variance of $N(W)$ is:

$$\begin{aligned} \hat{\hat{V}}_m(W) = \hat{\hat{V}}_{h-2}(W) = & \frac{N(w_1, ..., w_{h-1}) * N(w_2, ..., w_h)}{N(w_2, ..., w_{h-1})^3} \\ & * \big(N(w_2, ..., w_{h-1}) - N(w_1, ..., w_{h-1})\big) \\ & * \big(N(w_2, ..., w_{h-1}) - N(w_2, ..., w_h)\big) \end{aligned} \tag{B.3}$$

For Tetra, $m = 2$, and $h = 4$.

These equations were encoded in `tetraZscores.pl`, using the PDL package for greater computational efficiency. `tetraZscores.pl` accepts a list of files (with associated sequence names), pulls the sequence for each file, computes the Z-score for every $4-$mer W, and writes the results out to a score file. `tetraCorrelation.pl` then computes the Pearson coefficient for every pair of vectors given in the two input score files (one vector from each file) and writes the results to a final correlation file.

Note that if $\hat{\hat{V}}_m(W) = 0$, then $z^m(W)$ is ill-defined. Upon advice from H. Teeling (personal communication April 2, 2014), the decision was made to set $z^M(W) = 0$ in that case. These cases are rare, but do occur.

See [22] for more details.

B.2 Tacoa

Like Tetra, Tacoa scores are based on the significance of a $k-$mer, not its raw frequency. Similarity between two profiles is computed using a smoothed kernel function. [23]

The significance of a $k-$mer is computed relative to GC content. Let $w = w_1 w_2 ... w_k$ be a $k-$mer whose significance we wish to compute in a sequence s. Suppose s has GC content r_s. Then the probability of seeing a nucleotide w_i in a given position is:

$$p_s(w_i) = \begin{cases} \frac{r_s}{2} & \text{if } w_i \in \{C, G\} \\ \frac{1-r_s}{2} & \text{if } w_i \in \{A, T\} \end{cases} \tag{B.4}$$

Let $N_s(w)$ be the observed count of w in s, and let $E_s[w]$ be the expected count:

$$E_s[w] = |s| \prod_{i=1}^{k} p_s(w_i) \tag{B.5}$$

From this, we can obtain the significance score of the $k-$mer w :

$$g_s(w) = \begin{cases} 0 & \text{if } N_s(w) = 0 \\ \frac{N_s(w)}{E_s[w]} & \text{if } N_s(w) > E_s[w] \\ -\frac{E_s[w]}{N_s(w)} & \text{if } N_s(w) \leq E_s[w] \end{cases} \tag{B.6}$$

Let $w^{[1]}, w^{[2]}, ..., w^{[4^k]}$ be the canonical ordering of all $k-$mers. We define our profile:

$$\mathbf{x}(s) = \langle g_s(w^{[1]}), g_s(w^{[2]}), ..., g_s(w^{[4^k]}) \rangle \tag{B.7}$$

Let s, t be two sequences. The distance between their two profiles is:

$$d\big(\mathbf{x}(s), \mathbf{x}(t)\big) = 1 - < \mathbf{x}(s), \mathbf{x}(t) > \tag{B.8}$$

where $< \mathbf{x}(s), \mathbf{x}(t) >$ is the dot product of $\mathbf{x}(s)$ and $\mathbf{x}(t)$. The smoothed kernel is:

$$K\big(\mathbf{x}(s), \mathbf{x}(t)\big) = e^{\left(-d(\mathbf{x}(s),\mathbf{x}(t))^2/2\right)} \tag{B.9}$$

See [23] for more details.

B.3 Affinity Propagation

Affinity Propagation, hereafter referred to as AP, is a message-passing clustering algorithm that does not require the user to know the number of clusters in advance. The simple form of AP takes as input a full similarity matrix between N points to be clustered, and a list of preferences for each data point, and returns a clustering of the data points, each cluster having its own exemplar data point. The initial preference for a data point is its suitability to be an exemplar for a given cluster. [34]

At each step, data points exchange messages detailing their suitability to a) be exemplars for other data points, and b) be exemplified by other data points; respectively, these quantities are called availability and responsibility. Suppose we have two points i and k. Then we define their responsibility and availability of i to j:

$$r(i,k) = s(i,k) - \max_{j \neq k} \big(a(i,j) + s(i,j)\big) \tag{B.10}$$

$$a(i,k) = \begin{cases} \min\Big(0, r(k,k) + \sum_{j \notin \{i,k\}} \max\big(0, r(j,k)\big)\Big) & \forall i \neq k \\ \sum_{j \neq k} \max\big(0, r(j,k)\big) & \forall i = k \end{cases} \tag{B.11}$$

$a(i,k)$ is initialized to 0, and $s(k,k)$ is initialized to be the preference provided for data point k.

At the end of each iteration, each point i "chooses" a provisional exemplar point k such that k maximizes the sum $a(i,k)+r(i,k)$ (note that this k might be i itself, in which case i is an exemplar). After iteration starts, values for r and a are damped according to a user-specified parameter λ in order to avoid uncontrolled behavior. Iteration continues until no change is seen in exemplar assignment for a given number of steps.

The assigned preferences can either be equal for all data points, indicating that there is no reason to believe that any one data point is more or less suited to be an exemplar than any other data points. Otherwise, differing values indicated greater suitability for some points, and lesser suitability for others. The higher the initial preferences, the greater the number of clusters.

If the initial preference for each point is equal to the maximum similarity score, then each data point will be in a cluster of its own. Let $C(i,k) = a(i,k) + r(i,k)$, the quantity maximized by the provisional exemplar k for i.

$$\begin{aligned} C(k,k) &= a(k,k) + r(k,k) \\ &= \sum_{j \neq k} \Big(\max\big(0, r(j,k)\big) \Big) + s(k,k) - \max_{j \neq k} \big(a(i,j) + s(i,j) \big) \end{aligned} \tag{B.12}$$

$$\begin{aligned} C(i,k) &= a(i,k) + r(i,k) \\ &= \min\Big(0, r(k,k) + \sum_{j \notin \{i,k\}} \max\big(0, r(j,k)\big)\Big) + s(i,k) - \max_{j \neq k} \big(a(i,j) + s(i,j) \big) \end{aligned} \tag{B.13}$$

We know that:

$$s(k,k) \geq s(i,k) \qquad \text{since } s(k,k) = \max_{i,j} \big(s(i,k) \big) \tag{B.14}$$

which gives:

$$\begin{aligned} C(i,k) &= a(i,k) + r(i,k) \\ &\leq \min\Big(0, r(k,k) + \sum_{j \notin \{i,k\}} \max\big(0, r(j,k)\big)\Big) + s(k,k) - \max_{j \neq k}\big(a(i,j) + s(i,j)\big) \\ C(i,k) &\leq a(i,k) + r(k,k) \end{aligned} \tag{B.15}$$

Now we examine the $a(i,k)$ terms.

$$a(i,k) = \min\Big(0, r(k,k) + \sum_{j \notin \{i,k\}} \max\big(0, r(j,k)\big)\Big) \tag{B.16}$$

This gives us two cases: either $a(i,k) = 0$, or $a(i,k) = r(k,k) + \sum_{j \notin \{i,k\}} \max\big(0, r(j,k)\big)$. Suppose $a(i,k) = 0$.

$$\begin{aligned} a(i,k) &= 0 \\ &\leq \sum_{j \neq k} 0 \\ &\leq \sum_{j \neq k} \Big(\max\big(0, r(j,k)\big)\Big) \\ a(i,k) &\leq a(k,k) \end{aligned} \tag{B.17}$$

Now suppose $a(i,k) = r(k,k) + \sum_{j \notin \{i,k\}} \max\big(0, r(j,k)\big)$. This means that $a(i,k) < 0$, since otherwise $a(i,k) = 0$.

$$\begin{aligned} a(i,k) &< 0 \\ &< \sum_{j \neq k} 0 \\ &< \sum_{j \neq k} \Big(\max\big(0, r(j,k)\big)\Big) \\ a(i,k) &< a(k,k) \end{aligned} \tag{B.18}$$

Therefore, we have:

$$\begin{aligned} C(i,k) &= a(i,k) + r(i,k) \\ &\leq a(i,k) + r(k,k) \\ &\leq a(k,k) + r(k,k) \\ C(i,k) &\leq C(k,k) \end{aligned} \tag{B.19}$$

Therefore, i will never be a better exemplar for k than k itself.

See [34] for more details.

Appendix C

Bibliography

[1] T. Thomas, J. Gilbert, and F. Meyer, "Metagenomics - a guide from sampling to data analysis," *Microbial Informatics and Experimentation*, vol. 2, no. 3, 2012.

[2] K. Mavromatis, N. Ivanova, K. Barry, H. Shapiro, E. Goltsman, A. C. McHardy, I. Rigoutsos, A. Salamov, F. Korzeniewski, M. Land, A. Lapidus, I. Grigoriev, P. Richardson, P. Hugenholtz, and N. C. Kyrpides, "Use of simulated data sets to evaluate the fidelity of metagenomic processing methods," *Nature Methods*, vol. 4, no. 6, pp. 495–500, 2007.

[3] D. H. Huson, A. F. Auch, J. Qi, and S. C. Schuster, "Megan analysis of metagenomic data," *Genome research*, vol. 17, no. 3, pp. 377–386, 2007.

[4] O. U. Nalbantoglu, S. F. Way, S. H. Hinrichs, and K. Sayood, "RAIphy: Phylogenetic classification of metagenomics samples using iterative refinement of relative abundance index profiles," *BMC Bioinformatics*, vol. 12, no. 41, 2011.

[5] R. Seshadri, S. A. Kravitz, L. Smarr, P. Gilna, and M. Frazier, "Camera: a community resource for metagenomics," *PLoS biology*, vol. 5, no. 3, p. e75, 2007.

[6] V. Kunin, A. Copeland, A. Lapidus, K. Mavromatis, and P. Hugenholtz, "A bioinformatician's guide to metagenomics," *Microbiology and Molecular Biology Reviews*, vol. 72, no. 4, pp. 557—578, 2008.

[7] R. I. Amann, W. Ludwig, and K.-H. Schleifer, "Phylogenetic identification and in situ detection of individual microbial cells without cultivation," *Microbiological Reviews*, vol. 59, no. 1, pp. 143–169, 1995.

[8] M. H. Mohammed, T. S. Ghosh, R. M. Reddy, C. V. S. K. Reddy, N. K. Singh, and S. S. Mande, "Indus - a compostion-based approach for rapid and accurate taxonomic classification of metagenomic sequences," *BMC Genomics*, vol. 12, no. Suppl 3, p. S4, 2011.

[9] S. Chatterji, I. Yamazaki, Z. Bai, and J. A. Eisen, "Compostbin: A dna composition-based algorithm for binning environmental shotgun reads," in *Research in Computational Molecular Biology*, pp. 17–28, Springer, 2008.

[10] S. S. Mande, M. H. Mohammed, and T. S. Ghosh, "Classification of metagenomic sequences: methods and challenges," *Briefings in Bioinformatics*, vol. 13, no. 6, 2012.

[11] C. R. Woese and G. E. Fox, "Phylogenetic structure of the prokaryotic domain: The primary kingdoms," *Proceedings of the National Academy of Science*, vol. 74, no. 11, pp. 5088–5090, 1977.

[12] M. Monzoorul Haque, T. S. Ghosh, D. Komanduri, and S. S. Mande, "Sort-items: Sequence orthology based approach for improved taxonomic estimation of metagenomic sequences," *Bioinformatics*, vol. 25, no. 14, pp. 1722–1730, 2009.

[13] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, 1970.

[14] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–197, 1981.

[15] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, no. 3, pp. 403–410, 1990.

[16] J. Berendzen, W. J. Bruno, J. D. Cohn, N. W. Hengartner, C. R. Kuske, B. H. McMahon, M. A. Wolinsky, and G. Xie, "Rapid phylogenetic and functional classification of short genomic fragments with signature peptides," *BMC Research Notes*, vol. 5, no. 460, 2012.

[17] M. Wu and J. A. Eisen, "A simple, fast and accurate method of phylogenomic inference," *Genome Biology*, vol. 9, no. 10, p. R151, 2008.

[18] F. Gori, G. Folino, M. S. M. Jetten, and E. Marchiori, "Mtr: taxonomic annotation of short metagenomic reads using clustering at multiple taxonomic ranks," *Bioinformatics*, vol. 27, no. 2, pp. 196–203, 2011.

[19] D. Alonso-Alemany, J. C. Clemente, J. Jansson, and G. Valiente, "Taxonomic assignment in metagenomics with tango," *EMBnet.journal*, vol. 17, no. 2, pp. 46–50, 2011.

[20] J. C. Clemente, J. Jansson, and G. Valiente, "Flexible taxonomic assignment of ambiguous sequencing reads," *BMC Bioinformatics*, vol. 12, no. 8, 2011.

[21] T. S. Ghosh, M. Haque, and S. S. Mande, "Discribinate: a rapid method for accurate taxonomic classification of metagenomic sequences," *BMC bioinformatics*, vol. 11, no. Suppl 7, p. S14, 2010.

[22] H. Teeling, J. Waldmann, T. Lombardot, M. Bauer, and F. O. Glöckner, "Tetra: a web-service and a stand-alone program for the analysis and comparison of tetra nucleotide usage patterns in dna sequences," *BMC Bioinformatics*, vol. 5, no. 1, p. 163, 2004.

[23] N. N. Diaz, L. Krause, A. Goesmann, K. Niehaus, and T. W. Nattkemper, "Tacoa - taxonomic classification of environmental genomic fragments using a kerneled nearest neighbor approach," *BMC Bioinformatics*, vol. 10, no. 1, p. 56, 2009.

[24] A. Brady and S. L. Salzberg, "Metagenomic phylogenetic classification with interpolated markov models," *Nature Methods*, vol. 6, no. 9, pp. 673–676, 2009.

[25] Y. Wang, H. C. Leung, S.-M. Yiu, and F. Y. Chin, "Metacluster 5.0: a two-round binning approach for metagenomic data for low-abundance species in a noisy sample," *Bioinformatics*, vol. 28, no. 18, pp. i356–i362, 2012.

[26] Y.-W. Wu and Y. Ye, "A novel abundance-based algorithm for binning metagenomic sequences using l-tuples," *Journal of Computational Biology*, vol. 18, no. 3, pp. 523–534, 2011.

[27] M. Strous, B. Kraft, R. Bisdorf, and H. E. Tegetmeyer, "The binning of metagenomic contigs for microbial physiology of mixed cultures," *Frontiers in Microbiology*, vol. 3, 2012.

[28] A. Pati, L. S. Heath, N. C. Kyrpides, and N. Ivanova, "Clams: a classifier for metagenomic sequences," *Standards in genomic sciences*, vol. 5, no. 2, p. 248, 2011.

[29] J. Alneberg, B. S. Bjarnason, I. de Bruijn, M. Schirmer, J. Quick, U. Z. Ijaz, N. J. Loman, A. F. Andersson, and C. Quince, "Concoct: Clustering contigs on coverage and composition," *arXiv preprint arXiv:1312.4038*, 2013.

[30] G. J. Dick, A. F. Andersson, B. J. Baker, S. L. Simmons, B. C. Thomas, A. P. Yelton, and J. F. Banfield, "Community-wide analysis of microbial genome sequence signatures," *Genome Biology*, vol. 10, no. 8, p. R85, 2009.

[31] A. C. McHardy, H. G. Martín, A. Tsirigos, P. Hugenholtz, and I. Rigoutsos, "Accurate phylogenetic classification of variable-length dna fragments," *Nature Methods*, vol. 4, no. 1, pp. 63–72, 2007.

[32] Y. Wang, H. C. Leung, S.-M. Yiu, and F. Y. Chin, "Metacluster 4.0: a novel binning algorithm for ngs reads and huge number of species," *Journal of Computational Biology*, vol. 19, no. 2, pp. 241–249, 2012.

[33] M. H. Mohammed, T. S. Ghosh, N. K. Singh, and S. S. Mande, "Sphinx - an algorithm for taxonomic binning of metagenomic sequences," *Bioinformatics*, vol. 27, no. 1, pp. 22–30, 2010.

[34] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, pp. 972–976, 2007.

[35] Y. Sun, Y. Cai, S. M. Huse, R. Knight, W. G. Farmerie, X. Wang, and V. Mai, "A large-scale benchmark study of existing algorithms for taxonomy-independent microbial community analysis," *Briefings in Bioinformatics*, vol. 13, no. 1, pp. 107–121, 2011.

[36] D. Steinley, "K-means clustering: A half-century synthesis," *British Journal of Mathematical and Statistical Psychology*, vol. 59, no. 1, pp. 1–34, 2006.

[37] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, 1967.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[39] A. Acland, R. Agarwala, T. Barrett, J. Beck, D. A. Benson, C. Bollin, E. Bolton, S. H. Bryant, K. Canese, D. M. Church, *et al.*, "Database resources of the national center for biotechnology information," *Nucleic acids research*, vol. 42, no. D1, p. D7, 2014.

[40] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, 2010.

[41] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*, vol. 1. Cambridge university press Cambridge, 2008.

[42] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure.," in *EMNLP-CoNLL*, pp. 410–420, ACL, 2007.

[43] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online; accessed 2014-08-18].

[44] S. Schbath, "An efficient statistic to detect over- and under-represented words in dna sequences," *Journal of Computational Biology*, vol. 4, no. 2, pp. 189–192, 1997.

[45] M.-Y. Leung, G. M. Marsh, and T. P. Speed, "Over- and underrepresentation of short dna words in herpesvirus genomes," *Journal of Computational Biology*, vol. 3, no. 3, pp. 345–360, 1996.