Blake Johnson
David Galambos

Metagenomic Binning

Introduction and Applications

Metagenomic binning is a process for determining genomes of several species at once when a biological sample has multiple types of organisms. This process is extremely useful for biologists and rather difficult for computer science. This can be very helpful when trying to determine the species present in an unknown sample, such as an extreme environment hosting bacteria and archaea. Another more practical application is monitoring gut microbiomes, such as a study observing changes in gut microbiomes for premature infants based on their hospital rooms[1]. However, the computational techniques for solving this problem can often be slow and very statistically rigorous.

This concept of metagenomic binning goes just slightly beyond what we learned in class. In class we learned how reads are assembled into contigs which can then be mapped onto genomes. These provide the input for binning, which is a subsequent step where contigs will be grouped into different clusters by species or taxonomic group. These clusters are often known as bins. In real life, samples will often have multiple unknown organisms in varying abundances. This is where metagenomic binning steps in: a step after assembly that will split up the contigs into their corresponding species. We will cover four different binning algorithms in this paper and analyse the strengths and weaknesses of each.

Algorithm Description

Common Methods

There are two common methods that all our algorithms use: Tetranucleotide frequency (TNF) and differential coverage. To understand tetranucleotide frequency, it's quite convenient to break the word down: four-nucleotide frequency. A single tetranucleotide is just a short sequence of 4 nucleotides, with 256 possible tetranucleotides in DNA. The frequency of tetranucleotides in a single contig will often be representative of the frequency of tetranucleotides in the genome of the organism. Therefore, one can group contigs by similar tetranucleotide frequency to separate contigs accurately into bins.

To determine whether this measure is effective at delineating contigs into species, we wrote an algorithm that would roughly determine how different two species are based on tetranucleotide frequency. We tested it with 2 different strains of *E. coli* and 1 genome each of *Desulfobacula toluolica* and *Staphylococcus aureus*, all of which were obtained from NCBI Genome (See Table 1-2 and Figure 1 in the appendix). We were able to accurately reconstruct the phylogenetic tree based on our data.

One common method for metagenomic binning that none of our algorithms use is supervised binning. Supervised binning uses reference genomes to help the binning process. However, this assumes that the user already knows what species to expect out of their sample.

[1] Brooks et al. (2017)

Additionally, this might ignore minute differences between species or specific genetic variations. To avoid these limitations, all of the algorithms we studied are unsupervised binning processes, which do not use any reference genomes.

The other common method that all of our papers use is differential coverage. Differential coverage is best understood through a biological lens. Within a sample, there will be lots of variation in the abundance of different organisms. Therefore, when the contigs are produced, certain organisms which were more abundant in the sample will have much higher mapping coverage. This means that contigs can be separated by species efficiently and effectively just by observing the differences in coverage.

GroopM and CONCOCT

GroopM and CONCOCT are some of the earliest algorithms to use differential coverage, with both papers coming out in 2014. They both consider differential coverage and TNF simultaneously. CONCOCT begins by calculating a single vector for each contig using both coverage and tetranucleotide frequency information. This creates a multi-dimensional matrix that is then plotted on only two of its dimensions using a technique called principal component analysis (PCA). After plotting all the contigs in the sample, CONCOCT uses a Gaussian mixture model to accurately estimate cluster boundaries. Put simply, a Gaussian mixture model assumes that every input variable is continuous along a normal distribution. These clusters are then output as the final bins.[2] This is very different from MetaBAT and BinSanity, which build clusters by choosing contigs that best represent a potential bin.

MetaBAT

MetaBAT is an algorithm designed for extremely large datasets, in this case defined as several million contigs. It prides itself on its ability to be fast and memory-efficient. One of the key ways it does this is unique to this algorithm-- iterative bin creation. MetaBAT has a simple 7-step process for creating its bins. The first step in this process is determining which contig has the highest coverage. It will then use this contig as a medoid, and find contigs with similar TNF scores and coverage. Fascinatingly enough, instead of letting the user decide the cutoff values for similar TNF scores or deciding based on the data, MetaBAT creators downloaded 1,414 genomes and fragmented them to get empirical evidence on where cutoff points should be for determining inter- vs intra-species variation. Then, once all contigs within the cutoff point of the medoid have been recruited, MetaBAT will look at each contig in this set and determine if there is a best new medoid. It will then repeat these steps until the medoid is unchanged during the comparison step. This means that the medoid is the best medoid for that group of contigs, and all contigs within the cutoff point of that medoid have been added to our bin. At this point, we are done with the bin, and MetaBAT will set the bin aside and select the next highest covered contig. After repeating this process enough, MetaBAT will have exhausted its pool of contigs, i.e., every contig has been assigned to a bin. MetaBAT will then dissolve bins that're too small to be a whole genome (the paper describes these small bins as <200kbp). Then, these freed contigs will be individually reassigned to the best-fitting bin based on abundance correlation.

[2] Alneberg et. al. (2014)

BinSanity

BinSanity is the most recent of the four binning algorithms that we analyzed, and it also takes the most distinctive approach. While many other binning algorithms conduct only one type of clustering step that is based on both contig coverage and composition, BinSanity includes two entirely separate clustering phases: One is based solely on differential coverage, and the second phase adds information about sequence composition. BinSanity utilizes a clustering algorithm called affinity propagation (AP), which uses iterative steps to compare every contig to all other contigs. Each comparison between a contig pair (i,k) evaluates whether contig *k* is a good exemplar (cluster center) for contig *i,* and if so, whether contig *i* should be clustered with *k* depending on the number of other contigs that have already clustered with *k*.[3] BinSanity compares all contigs to all others iteratively until it reaches the maximum iteration parameter (set by the user) or achieves stable cluster boundaries.

First, BinSanity creates a log-transformed matrix of the Euclidean distances between contig coverage values. It then runs AP clustering on this matrix. In many cases, contigs from different organisms will have different coverage values due to differing abundances in the sample. After this step is complete, BinSanity evaluates the quality of the resulting clusters as standalone bins based on completeness and redundancy (see section on 'Evaluating Bin Quality'). If there are any high quality bins, they are set aside for the final output. However, there are likely many clusters composed of contigs with similar coverage values that nevertheless came from different species. These low-quality bins go through another AP clustering step that retains previous coverage information while incorporating Euclidean distances based tetranucleotide frequency and GC content. This allows BinSanity to refine the lower-quality bins obtained during the first coverage-only clustering step. Finally, BinSanity outputs the high-quality bins obtained after the first step along with all bins produced in the second step.[4]

Evaluating Bin Quality

Completeness and Redundancy

When working with uncharacterized experimental data (ie. there is no reference), calculating completeness and redundancy is the most common method for assessing bin quality. Both measures involve searching for a core set of single copy genes (SCGs) in each bin. SCGs are a set of genes that every single bacterial or archaeal genome is expected to have exactly once. An example is provided in Campbell et. al. (2013). A bin is "complete" if it contains every SCG at least once. However, it is "redundant" if some SCGs appear more than once. This may indicate that the bin has contigs from more than one taxonomic group and should be refined. Completeness and redundancy are usually expressed as percentages, and the minimum/maximum threshold for each depends on the goals of the specific metagenomic study. There are several tools designed specifically for evaluating bin quality using these metrics; one of the more common ones is CheckM.[5]

[3] Frey and Dueck (2007)
[4] Graham et. al. (2015)
[5] Parks et. al. (2015)

V-Measure

Unlike completeness and redundancy, V-measure is most useful in assessing the performance of a binning algorithm by comparing the output to a known reference. The reference is usually a set of bins with known identity. V-measure is defined using two statistics: Precision and recall, both of which range from 0 to 1. Precision assesses whether each output bin contains contigs from only one reference bin. Recall measures whether all contigs from a reference bin are assigned to the same output bin. The V-measure for the whole set of output bins is then calculated as the harmonic mean of precision and recall.

Algorithm Comparison

Runtime

For the binning process, algorithms usually have to compare all contigs to all other contigs in order to calculate some type of distance between data points. This creates very slow $O(N^2)$ algorithms. However, MetaBAT manages to speed up this process through its iterative approach to binning, allowing it to create bins quickly after comparing all the contigs. Even though CONCOCT produced bins with higher completeness than MetaBAT in an error-free trial, MetaBAT took 14 minutes to complete and used only 3.9GB. In contrast, CONCOCT took a shocking 105 hours and 9.6GB.[6] The authors of BinSanity note that their algorithm is also comparatively slow. Although their choice of affinity propagation for clustering may lead to more accurate clusters, AP finds cluster centers by comparing every contig to every other contig. This process happens iteratively, so the clustering itself (not the calculation of distances) is an $O(N^2T)$ step, where N is the number of contigs and T is the number of iterations of comparing all N contigs against all contigs.

Closely Related Genomes

When the metagenomic sample contains many closely related species, some algorithms may have trouble distinguishing them. BinSanity has an advantage in accurately sorting these contigs into separate bins. This is because BinSanity has an initial clustering step that is based solely on coverage. While contigs from closely related species will always have similar tetranucleotide frequencies, coverage may differ. Therefore, other binning algorithms that include tetranucleotide frequency at every clustering step will tend to cluster contigs from two closely related species into the same bin, leading to high redundancy. BinSanity can avoid this issue by separating contigs from closely related species (with different coverage values) at the outset.

MetaBAT, on the other hand, struggles with closely-related genomes. Suppose we have 2 similar species, A and B. When MetaBAT is initially binning A, it will likely add a lot of the contigs from B into this bin as well, due to its iterative approach and predetermined cutoffs. Then, when it tries to form a bin for B, there will be very few contigs to add, because the bin for A has "stolen" a lot of these contigs that might fit better with B. Therefore, this bin for the B will be small and will be dissolved into other bins, leaving MetaBAT to overestimate the quantity of

[6] Kang et. al. (2015)

the species A and completely miss species B. This problem is only exacerbated when multiple similar species are present.

Low Abundance Genomes

When performing a binning analysis on an unknown sample, we are often interested in "rare" taxonomic groups with low abundance. This is a difficult challenge for all binning algorithms because coverage may be so low that complete bins are infeasible. Assuming that these low-abundance species aren't closely related to other groups, tetranucleotide frequencies might indicate that these are simply outliers of larger bins. Thus, some binning algorithms that compare contig composition and coverage simultaneously may be poorly suited for targeting low-coverage contigs. By contrast, BinSanity is better able to target these "rare" genomes due to its initial coverage clustering step. This enables BinSanity to separate all the low-abundance contigs into one large bin and then use traditional composition measures to split this cluster into smaller potential bins.[2]

Small Number of Samples

Having a low number of metagenomic samples (< 6, depending on the study) limits the amount of differential coverage data that is available. This can impact the success of the binning process because differential coverage depends on having a large number of samples across which to compare contigs. This makes it more difficult to cluster contigs that have similar coverage patterns across samples. Somewhat predictably, BinSanity performs comparatively poorly under these conditions due to its initial clustering step based solely on coverage. When there is little coverage information, the initial step is less accurate, and the mistakes carry over in the rest of the analysis. When the metagenomic data was limited to 2-3 samples, CONCOCT outperformed BinSanity as assessed by V-measure and Adjusted Rand Index (ARI).[7]

Conclusions

Learning metagenomic binning has taken the knowledge we've learned in class and made it even more applicable to real-world situations. It is also a wonderful example of a very simple-sounding problem requiring extremely hard computational answers. Many of these algorithms have specific specialities, so it is important to be aware of how these algorithms work so that the best option for a particular problem may be chosen. Fascinatingly enough, in the MetaBAT paper, they imply that for certain situations, CONCOCT and MetaBAT could be complementary algorithms, getting the best results by running both of them. This is a process we haven't considered in class before, and it's definitely something to be aware of when looking to use an algorithm. Why limit yourself to one algorithm when comparisons across several might actually give the best result? Well, with third-generation and next-generation sequencing, sometimes the sheer amount of data present makes running the data through multiple algorithms simple unfeasible. Overall, knowledge of the available algorithms and how they work is extremely important for deciding what to use. There are still many challenges and ways binning can be substantially improved, including analysis of low-abundance genomes and accurately binning extremely large datasets in a reasonable amount of time.

[7] Graham et. al. (2017)

Literature Cited

Alneberg, J., Bjarnason, B.S., de Bruijn, I., Schirmer, M., Quick, J., Ijaz, U.Z., Lahti, L., Loman, N.J., Andersson, A.F., and Quince, C. (2014). Binning metagenomic contigs by coverage and composition. Nat Methods *11*, 1144-1146.

Brooks, B., Olm, M.R., Firek, B.A., Baker, R., Thomas, B.C., Morowitz, M.J., and Banfield, J.F. (2017). Strain-resolved analysis of hospital rooms and infants reveals overlap between the human and room microbiome. Nat Commun *8*.

Campbell, J.H., O'Donoghue, P., Campbell, A.G., Schwientek, P., Sczyrba, A., Woyke, T., Soll, D., and Podar, M. (2013). UGA is an additional glycine codon in uncultured SR1 bacteria from the human microbiota. P Natl Acad Sci USA *110*, 5540-5545.

Frey, B.J., and Dueck, D. (2007). Clustering by passing messages between data points. Science *315*, 972-976.

Graham, E.D., Heidelberg, J.F., and Tully, B.J. (2017). BinSanity: unsupervised clustering of environmental microbial assemblies using coverage and affinity propagation. Peerj *5*.

Imelfort, M., Parks, D., Woodcroft, B.J., Dennis, P., Hugenholtz, P., and Tyson, G.W. (2014). GroopM: an automated tool for the recovery of population genomes from related metagenomes. Peerj *2*.

Kang, D.W.D., Froula, J., Egan, R., and Wang, Z. (2015). MetaBAT, an efficient tool for accurately reconstructing single genomes from complex microbial communities. Peerj *3*.

Parks, D.H., Imelfort, M., Skennerton, C.T., Hugenholtz, P., and Tyson, G.W. (2015). CheckM: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. Genome Res *25*, 1043-1055.

Appendix

The genome FASTA files and code can be found on our github.

Table 1. Pairwise tetranucleotide frequency differences from 4 bacterial genomes.

Genome comparison	Combined TNF Difference
S. aureus vs. *E. coli* IAI39	0.5956
S. aureus vs. *E coli* O83:H1 str. NRG 857C	0.5957
S. aureus vs. *D. toluolica* Tol2	0.3925
D. toluolica Tol2 vs. *E. coli* IAI39	0.3925
D. toluolica Tol2 vs. *E. coli* O83:H1 str. NRG 857C	0.3949
E. coli IAI39 vs. *E. coli* O83:H1 str. NRG 857C	0.0131

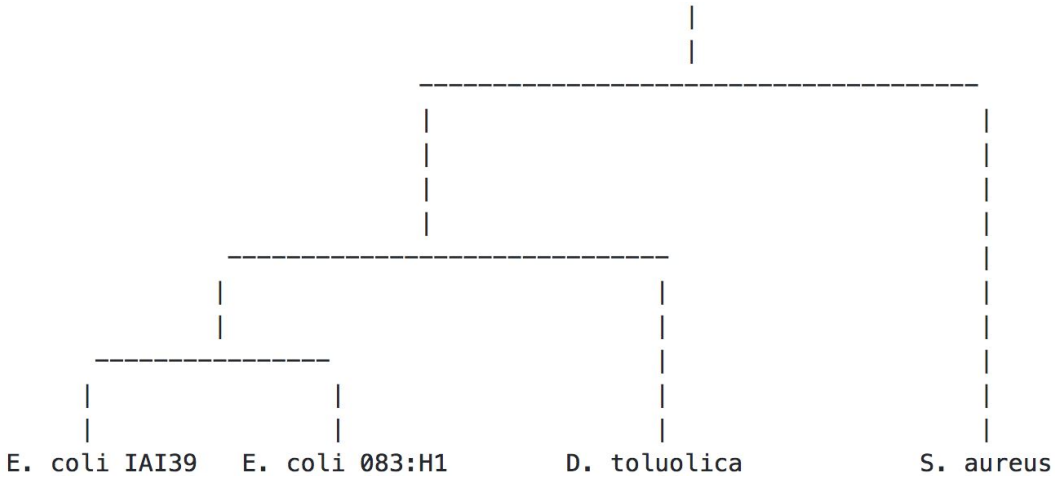

Figure 1. Phylogenetic tree based on our TNF data (identical to actual phylogeny).

Table 2. GenBank assembly ID for each genome used in our analysis. We selected complete genomes with varying expected levels of phylogenetic similarity from NCBI Genome.

Genome	GenBank Accession ID
Staphylococcus aureus	GCA_000013425.1
Desulfobacula toluolica Tol2	GCA_000307105.1
E. coli IAI39	GCA_000026345.1
E. coli O83:H1 str. NRG 857C	GCA_000183345.1