

Taxon-Specific Barcode Statistics For Amplicon Metagenomics

Zachary Foster

June 13, 2014

Introduction

Amplicon metagenomics, also known as metabarcoding, is a rapidly advancing field that has great potential for characterizing the microbial communities. Before metabarcoding, scientist were restricted to low-throughput techniques heavily biased by culturability. Although shotgun metagenomics has advantages over metabarcoding (e.g. no primer bias, functional genomic inferences), it is still too expensive to sample most communities exhaustively. Metabarcoding allows for up to hundreds of samples to be exhaustively characterized using a single Illumina MiSeq run, costing less than \$3000. With the ever decreasing cost of sequences and increasing reads lengths of high-throughput sequences like Illumina, metabarcoding is likely to become a routine technique in microbial ecology laboratories. Unfortunately metabarcoding is limited by the availability of reference databases and "universal" primer pairs. This reliance has stifled the application of metabarcoding to novel groups of organisms and unconventional barcode loci. Software to aid in the planning and evaluation of potential barcode loci and their effectiveness for different taxonomic groups would greatly accelerate the application of this powerful technique to new systems. Therefore I am developing a set of python and R scripts to characterize taxon-specific statistics relevant to metabarcoding. The pipeline can combine and reformat multiple different reference databases, randomly sub-sample the combined database by taxon, and calculate the optimal distance clustering threshold and predicted error rates using an arbitrary distance matrix. I am also looking into integrating *in silico* PCR data to predict the effects of different primers on the ability to detect and differentiate taxa. My scripts use reference database sequences to produce useful tables of statistics pertaining to each taxon and intuitive, information-rich graphics for evaluation of novel experimental protocols.

Methods

Database consolidation and standardization

In planning a amplicon metabarcoding experiment it is sometimes necessary to combine multiple reference database with different formats so they can be used in the same pipeline. This is especially the case when studying multiple kingdoms within a single community or an obscure group of organisms requiring specialized databases. Unfortunately most databases have their own custom FASTA header formats and few have an explicit taxonomy specified, so their taxonomy must be downloaded from NCBI by referencing an accession or taxonomy ID. This means that the first step in any pipeline that seeks to use sequences from multiple reference databases must be able to read in multiple format. Therefore, I made a script called `reformat.fasta.py` that reformats the FASTA headers of a variety of common reference database formats into a format similar to that used by UNITE (Figure 1). A single format also makes it easier for multiple scripts to communicate. The header format contains pipe-delimited positional fields for the organism name, genbank ID, reference database-specific ID, and taxonomy. The levels of the taxonomy field (Figure 2) are semi-colon-delimited and each is in the form "l_name", where "l" is a one or two character abbreviation of the taxonomic level and "name" is the taxon (e.g. "p_Ascomycota"). Currently the `reformat.fasta.py` can read in the formats of the following database: UNITE, ITS1, Genbank, RDP, Phyto ID, and Phyto DB. Of these, only Genbank, RDP, and UNITE have full taxonomy information included. However the others do have either genbank accession or taxonomy IDs, allowing their full taxonomy to be downloaded. A possible future improvement to `reformat.fasta.py` could

```

RDP: >S000415306 Sparassis crispa; MAFF 238626 Lineage=Root;rootr...
UNITE: >Lachnum|JF690990|SH189789.06FU|reps|k_Fungi;p_Ascomycota;c_Le...
ITS1: >EF093575_ITS1_GB|Caloplaca albopruinosa|tax_id:413252|represen...
Phyto DB: >PD_00795_ITS (Phytophthora infestans )
Phyto ID: >AF271224.1|Pythium vexans

```

Figure 1: Format of reference database FASTA headers.

```

d_Fungi;p_Glomeromycota
d_Fungi;p_Ascomycota;c_Dothideomycetes
d_Fungi;p_Ascomycota;c_Dothideomycetes;o_Pleosporales;f_Phaeosphaeriaceae;g_Phoma
d_Fungi;p_Ascomycota;c_Dothideomycetes;o_Pleosporales;f_Didymellaceae;g_Phoma

```

Figure 2: Format of taxon identifiers and FASTA taxonomy strings.

be an ability to automatically download the taxonomy information for these databases and insert it into their headers.

Initial database statistics

In order to make downstream scripts less memory intensive, I made a script called `fasta_taxon_statistics.py` to calculate taxon-specific statistics before the distance matrix is calculated. This is principally done to aid in randomly sub-sampling the database without loading it all into memory, as is described in the next section.

Statistics are calculated for each taxon at each taxonomic level. Taxa are differentiated by their lineage (i.e. the name of the taxon, as well all of the higher level taxa it is a part of). Using the lineage to define a taxon instead of just its name and level allows for correct handling of multiple taxa at the same level with the same name, but part of different higher level taxa (Figure 2). This is important for correct summary statistics and downstream tree visualization. Statistics calculated for each taxon include: the number of sequences, the taxonomic level, the parent taxon, and the child taxa.

Database sub-sampling

In order to focus on a specific subset of an entire reference database, I made a python script that sub-samples a FASTA reference database, taking into consideration the taxonomy encoded in the headers. This is often necessary since reference databases are very large and the time it takes to calculate a pair-wise distance matrix is proportional to the square of the number of sequences. Some taxa have much more sequences than are necessary. On the other hand, many taxa have too few sequences to characterize their sequence variation. These should be also removed so computational time can be devoted to more informative taxa. There also can be inconsistencies in the number of taxonomic levels specified, so some filtering is required to remove sequences with low-resolution taxonomies.

Therefore, I have made a script called `fasta_taxon_subsample.py` that has the ability to sub-sample a reference database in a taxonomy-specific manner. Taxa with too many sequences can be randomly sub-sampled and taxa with too few can be removed. It uses the output of `fasta_taxon_statistics.py` (Figure 3) to decide which sequences are to be filtered out before it reads the reference database. This allows it to read the database one sequences at a time and thus it is able to process nearly any size of reference database of data with minimal RAM. It can also filter out sequences that are not identified to a specific set of taxonomic levels. Specific taxa can also be required or removed, so that the user can focus on specific groups of organisms.

id	taxon	level	name	count	parent	children
1	d_Fungi	d	Fungi	8000		2;3;4;5;6;7
2	d_Fungi;p_Ascomycota	p	Ascomycota	4390	1	8;9;10;11

Figure 3: Format of `fasta_taxon_statistics.py` output.

Distance matrix calculation

One of the most important and computationally intensive steps in this procedure is that calculation of a pairwise distance matrix between all sequences. Any program or distance metric can be used; the choice depends of the type of sequence data available and the characteristics of the locus analyzed. If the reference database is a multiple sequences alignment, then a program such as `fdnadist` from the EMBOSS package can be used. If the sequences are unaligned and all of the same locus, than a pairwise global alignment tool, such as the `pair.seqs` command from the Mothur package can be used. If the sequences are unaligned and contain the locus of interest, but are not globally alignable, it might be possible to use *in silico* PCR, using a program such as `isPCR`, to extract amplicon sequences so they can be globally aligned. This method adds a whole new level of complexity and will be explained further in the discussion. Finally, if the sequences do not share any genes it might still be possible to obtain a rough distance using tetranucleotide frequency or GC content.

After a distance matrix is obtained using one the methods described above it might be necessary to rename the rows and columns as the sequence taxonomy headers. To do this I have written a simple script called `reformat_matrix.py`. Currently this script only accepts the Phylip format outputted by `fdnadist`, but I plan to add support for other formats as needed.

Calculation of taxon-specific distance statistics

once a correctly formatted distance matrix is produced, it can be sub-sampled for each taxon and statistics calculated. The most important of these are the optimal clustering threshold and the expected error rate at that threshold. The optimal clustering threshold can be calculated for any taxonomic level (e.g. clustering species vs clustering genera).

To do this and visualize the results, I wrote an R script. For clustering a given taxonomic level, it subsamples the entire distance matrix for each taxon and calculates the distribution of inter-taxon and intra-taxon distances, the optimal clustering threshold, and error rate. The clustering threshold optimization and error rate calculations are done using the `Spider` package function `threshOpt`. The optimal clustering threshold is determined by clustering over a range of distance thresholds and picking the threshold that minimizes the sum of the false positive and false negative error rates. False positives result from too low of a threshold, causing sequences to be classified differently when they are actually the same for a given taxonomic level. False negatives result from too high of a threshold, causing sequences to be classified as the same when they are different. Currently clustering at only one level at a time is supported, so the script needs to be run multiple times to get optimal thresholds for clustering different taxonomic levels.

Visualization of taxon-specific distance statistics

With such high dimension data, effective visualization is key to discovering patterns. I used the R packages `ggplot2` and `igraph` to make information-rich graphics for exploratory analysis. I designed a novel (as far as I know) form of displaying multidimensional data within a hierarchy by placing graphs produced by `ggplot2` into the nodes of a phylogenetic tree produced by `igraph`. This allows for the relationships of hundreds of taxa to be browsed at once in an intuitive manner. I also used colored nodes to display the distribution of taxon statistics over an entire phylogeny, making large-scale patterns over taxonomic levels and lineages apparent. In addition, more conventional graphs displaying the distribution of statistics at each taxonomic level are also produced. All these graphics are produces automatically and saved as PNG images in a defined file structure allowing for further automated analysis and integration of results.

Results

Construction of test data

To demonstrate the abilities of this pipeline, I analyzed the Ribosomal Database Project (RDP) LSU reference multiple sequence alignment, testing its ability to differentiate a range of taxonomic levels from species to class. The database contained 95,396 fungal LSU sequences and the aligned FASTA file was 2.0Gb.

After formatting the database using `fasta_taxon_statistics.py`, I used `fasta_taxon_statistics.py` to calculate initial taxon-specific statistics for the entire database. After appending genus and species information (if present in the organism name) onto the taxonomy string during formatting, 37,733 distinct taxa were present. I then filtered out any sequences that did not have the full taxonomy string ranging from domain to species, resulting in a filtered database with 64,038 sequences and 28,897 taxa. In order to reduce the database further and remove taxa with two few sequences, I sub-sampled each species to 10 sequences and removed species with less than 10 sequences. This reduced the database to 8,000 sequences comprising 1,339 taxa.

With this drastically reduced database I used `fdnadist` to calculate a distance matrix using the Kimura 2-parameter evolution model, which allows for different rates of transitions and transversions. This took about 2 hours to run on my desktop computer. The resultant distance matrix and taxon-specific statistics for the sub-sampled database were the only inputs of used for downstream analysis.

The ability of this reference alignment to differentiate sequences of each taxon into every taxonomic level between species and class was tested.

Visualizations

Graphs were produced summarizing the distribution of both the optimal clustering threshold (Figure 4) and the predicted error rate at that threshold (Figure 5). A more complicated set of graphs was made showing the false positive and false negative error rates of a range of clustering thresholds for each taxon (Figures 6 and 7). These are displaying the information from which the optimal threshold and error rate were calculated.

The average optimal clustering threshold for differentiating species is nearly equal across taxonomic levels at around 0.01 (Figure 4 (a)). For differentiating genera, it is also nearly equal, but around 0.03 (Figure 4 (b)). However in both cases the distribution of optimal clustering thresholds is far from normal and changes between taxonomic levels (Figure 4 (c) and (d)). Looking at optimal threshold values throughout the phylogeny, it appears that it is weakly associated with evolutionary history, but the extreme values are distributed throughout the tree (Figure 4 (e) and (f)). The effect of evolutionary history seems slightly more pronounced when clustering into genera.

The predicted error rate when clustering at the optimal threshold decreases with decreasing taxonomic level, when differentiating by both species and genera, but is consistently lower for genera (Figure 4 (a) and (b)). This is expected since genera should be more diverged than species. There seems to be an effect of evolutionary history on error rates since lineages tend to have similar values (Figure 4 (e) and (f)).

Discussion

0.1 Implications for conventional metabarcoding

With even this first attempt using a test data set on a single locus (LSU) intriguing patterns are already becoming apparent that violate many assumptions made by researchers out of convenience. Metabarcoding analysis pipelines and used distance-based clustering usually assume that the entirety of a large group has a similar inter-specific distance, such as 3% PID for fungi ITS sequence. Here we observe that the values for optimal inter-specific K2 thresholds of genera easily range from 0.0 to 0.1 (Figure ?? ??). The inter-genus distance covers twice that range (Figure ?? ??). Error rates at optimal thresholds show a similar amount of variation, even though they are calculated for each taxon separately (Figure 5). If they were calculated using a single universal threshold the variation in error rates would be even higher as a result of different optimal thresholds. This means that studies using LSU for all fungi at a single clustering threshold are often incorrectly differentiating species up to 50% of the time (Figure ?? ??).

Future directions

There are many ways to enhance this analysis to provide more useful information. Generating a confidence interval for estimates of the optimal clustering threshold and error rates would aid interpretation of the data. The difference between false positive and false negative error rates for each taxon could be used to estimate the direction and degree of bias when estimating taxon diversity from numbers of OTUs. This information could be used to correct these counts. Estimating the optimal threshold at for differentiating taxa below a each taxon might allow for novel hierarchical clustering algorithms that iteratively cluster and classify amplicon sequences one taxonomic level at a time, applying the appropriate threshold for each taxon and assigning a confidence for each assignment. Including an *in silico* PCR step in the pipeline could take into account primer biases and allow for estimation of the relative effectiveness of a set of potential primers before an experiment. There are also many ways the pipeline could be seeded up using recursion that would make it more useful for large analyses. Finally, automated PDF report generation would allow researchers to quickly evaluate results by compiling the most important of the many graphs produced.

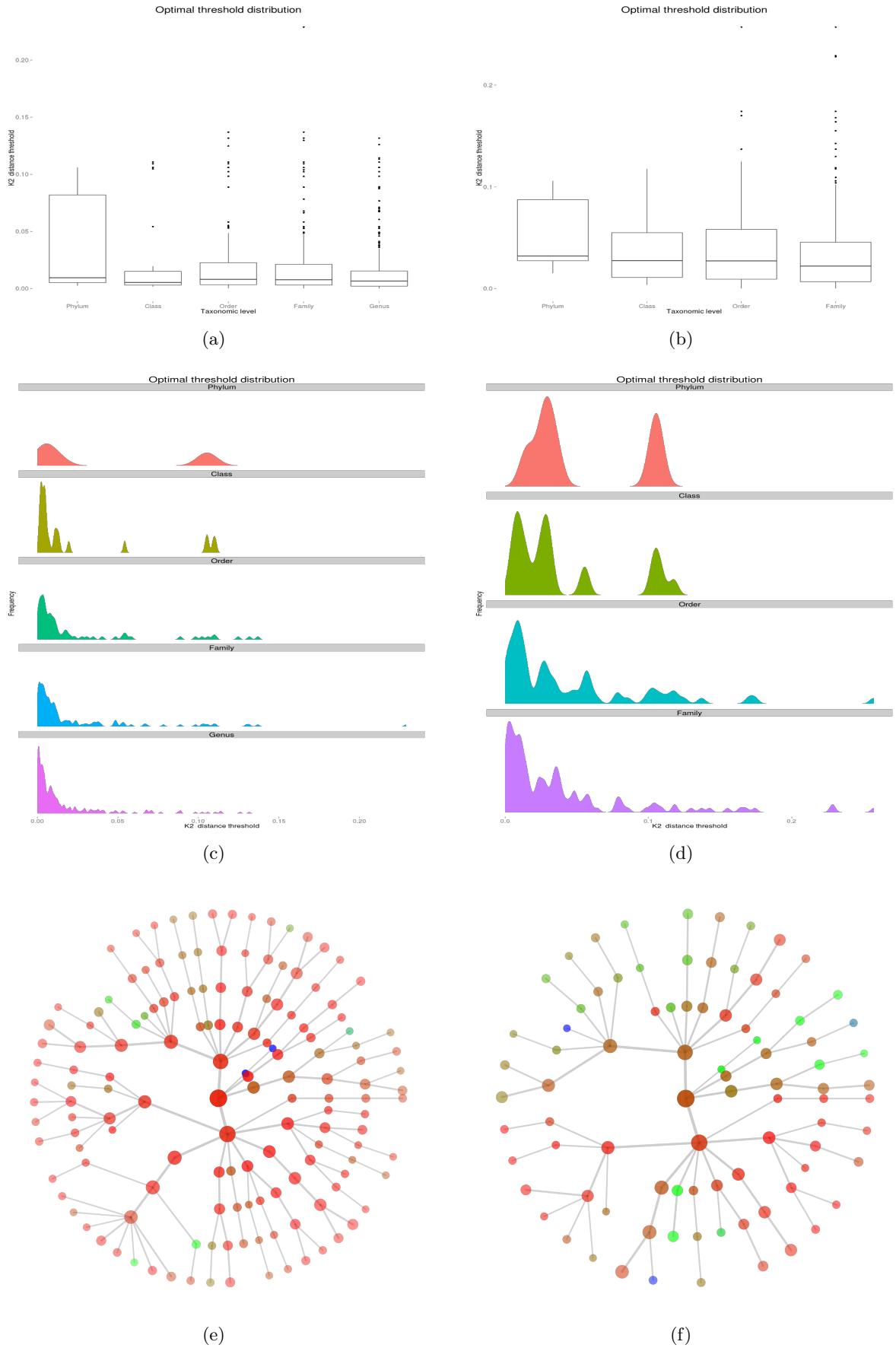


Figure 4: Optimal distance threshold statistics for differentiating species [(a) (c), (e)] and genera [(b) (d), (f)]. For figures (e) and (f) red is 0, blue is the maximum threshold, and green is the median.

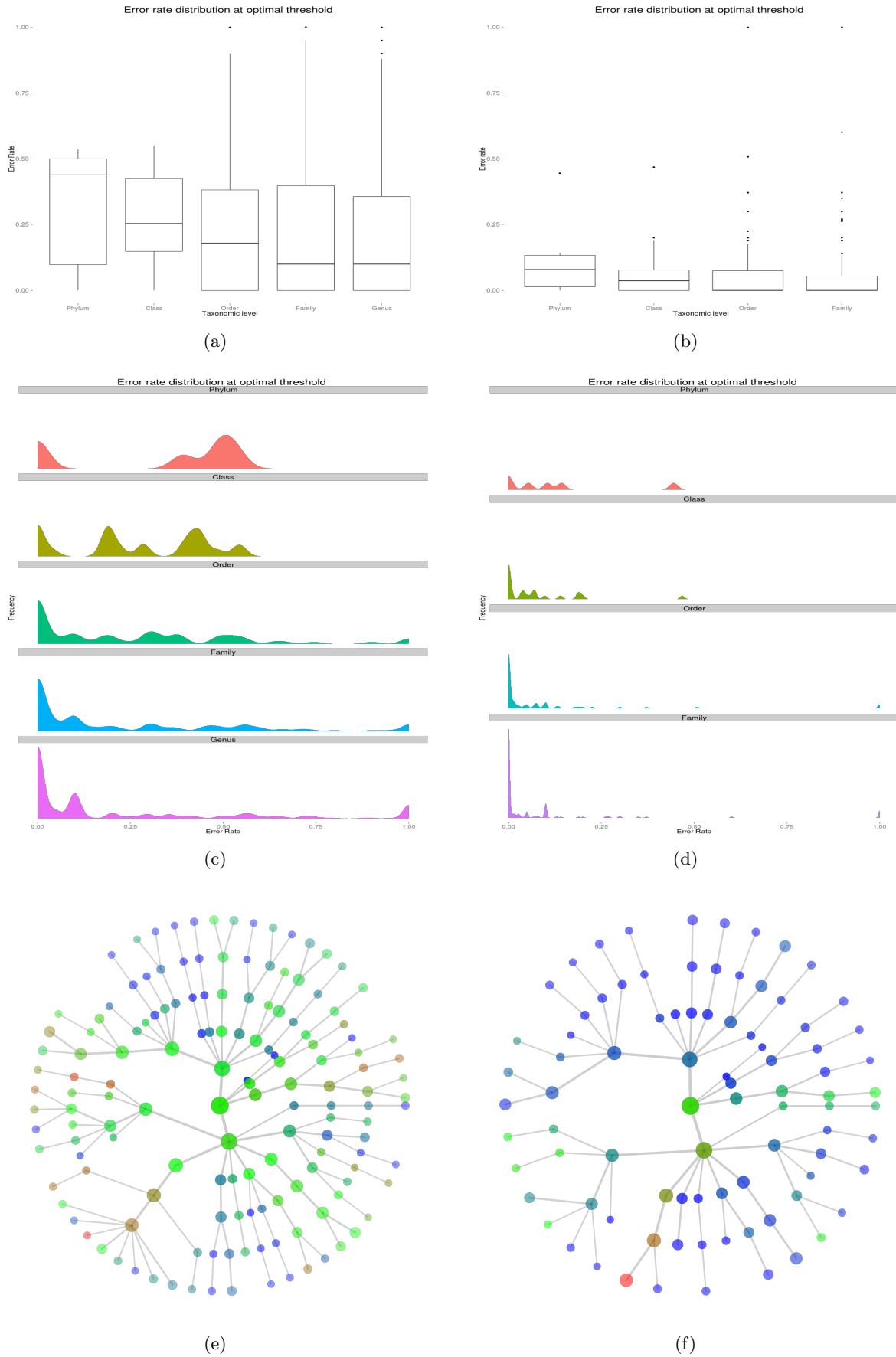


Figure 5: Error rates at optimal threshold statistics for differentiating species [(a) (c), (e)] and genera [(b) (d)]. For figures (e) and (f) red is 0%, green is 50%, and blue is 100%.

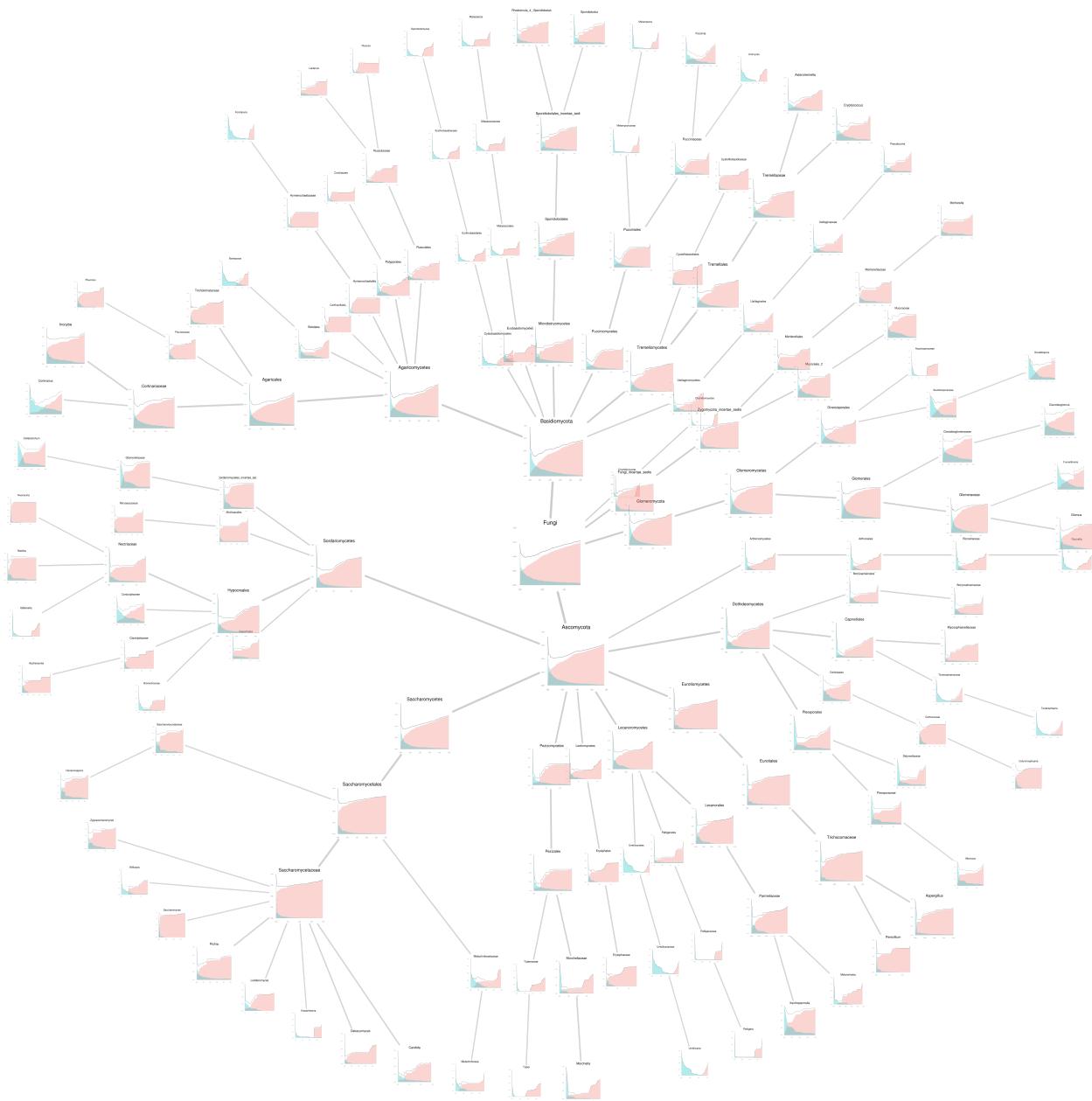


Figure 6: Threshold optimization graphs for differentiating species. Each graph is a kernel density plot of false positive (blue) false negatives (red) and cumulative error (black line). The y axis is relative frequency and the x axis is clustering threshold. The lowest point in the black line is the optimal threshold.

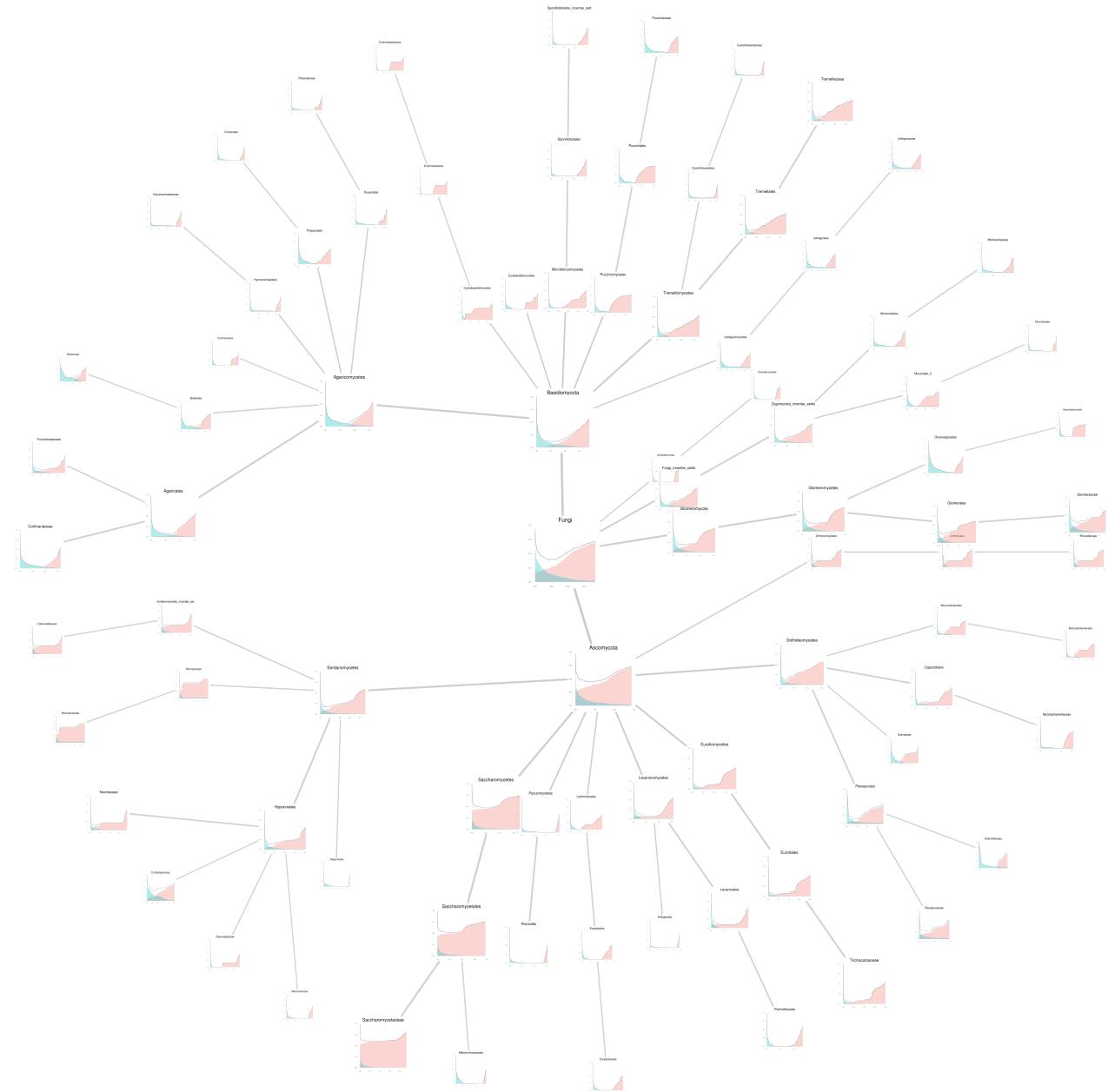


Figure 7: Threshold optimization graphs for differentiating genera. Each graph is a kernel density plot of false positive (blue) and false negatives (red) and cumulative error (black line). The y axis is relative frequency and the x axis is clustering threshold. The lowest point in the black line is the optimal threshold.