Taxon-Specific Barcode Statistics For Amplicon Metagenomics

Zachary Foster

June 13, 2014

Introduction

Amiplicon metagenomics, alos known as metabarcoding, is a rapidly advancing field that has great potential for characterizing the microbial communities. Before metabarcoding, scientist were reseticted to low-thruput techniqes heavly biased by culturability. Although shotgun metagenmics has advantages over metabarcoding (e.g. no primer bias, functional genomic inferences), it is still too expensive to sample most communities exhastivly. Metabarcoding allows for up to hundreds of samples to be exhanstivly characterized using a single Illumina MiSeq run, costing less than $3000. With the ever decreasing cost of sequences and increasing reads lengths of high-throuput sequences like Illuia, metabarcoding is likly to become a rutoine techniue in microbial ecology laboratories. Unofrtunatly metabarcoding is limited by the availbility of refence databases and "universal" primer pairs. This realiance has stiflied the application of metabarcoding to novel groups of organisms and unconvential barcode loci. Softeware to aid in the planning and evaluation of poential barcode loci and their effectivness for diffent taxonomic groups would greatly accelerate the application of this powerful techniqe to new systems. Therfore I am developing a set of python and R scripts to chaaracterize taxon-specific statitsics relevant to metabarcodings. The pipeline can combine and reformat multiple different reference databases, randomly subsample 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 differnt primers on the ability to detect and differntiate taxa. My scripts use refernece database sequences to produce useful tables of statsitcs 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 nessesary to combine multiple refernece database with differnt formats so they can be used in the same pipeline. This is especially the case when studing multiple kingdoms within a single community or an abscure group of organisms requiring specilized databases. Unfortunatly most databases have their own custom FASTA header formats and few have an explicit taxonomy specified, so their taxonomys must be downloaded from NCBI by refernecing an accession or taxonomy ID. This means that the first step in any pipline that seeks to use sequences from multiple refenrece 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 varity of common refenrece database formats into a format similar to that used by UNITE (Figure 1). A single format also makes it easier for multiple scirpts to communicate. The header format contians pipe-delimited positional fields for the organism name, genbank ID, refence 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 imporvemnt to `reformat_fasta.py` could be

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.

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

Initial database statistics

In order to make downsteam scripts less memory intensive, I made a script called `fasta_taxon_statistics.py` to caluculate taxon-specific statistics before the distance matrix is calculatd. This is principally done to aid in randomly subsampling 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 differntiated by their linage (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 handleing 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 visulaization. Statistics calculated for each taxon include: the number of sequences, the taxonomic level, the parent taxon, and the child taxa.

Database subsampling

In order to focus on a specific subset of an entire reference database, I made a python script that subsamples a FASTA reference database, taking into consideration the taxonomy encoded in the headers. This is often nessesary since refernece 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 nessary. 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 inconsienties in the number of taxonomic levels specified, so some filtering is required to remove sequences with low-resuluton taxonomies.

Therefore, I have made a script called `fasta_taxon_subsample.py` that has the ability to subsample a refenrece database in a taxonomy-specific manner. Taxa with too many sequences can be randomly subsampled 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 refenrece database. This allows it to reads the database one sequences at a time and thus it is able to process nearly any size of refernece 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 availble and the characterics of the locus analyzed. If the refernece datbase 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 sequenes 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 frequencey or GC content.

After a distance matrix is obtained using one the methods described above it might be nessesary 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 subsampled for each taxon and statistics cacluated. 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 visulize the results, I wrote an R script. For clustering a given taxonomic level, it subsamples the entire distance matrix for each taxon and calucluats the distribution of inter-taxon and intra-taxon distances, the optimal clustering threshold, and error rate. The clustering threshold optimaization 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 pciking the threshold that minimizes the sum of the flase positive and false negative error rates. False positives result from too low of a threshold, causing sequences to be classified differntly 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 differnt. Currently clustering at only one level at a time is supported, so the script needs to be run multiple times to get optimal thrtesholds for clusing differnt taxonomic levels.

Visulization of taxon-specific distance statistics

With such high dimentional data, effective visulization is key to discovering patterens. 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 multidimentional data within a heirarchy 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 intuative manner. I also used colored nodes to dispaly the distrubution of taxon statistcs over an entire phylogeny, making large-scale patterns over taxonomic levels and linages apparent. In additon, more conventional graphs dsiplaying 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 abilitties of this pipeline, I analyzed the Ribosomal Database Project (RDP) LSU refenrece multiple sequence alignment, testing it ability to differntiate a range of taxaonomic leves 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 statistcs for the entire database. After appending genus and species information (if present in the organism name) onto the taxonomy string during formatiing, 37,733 dsitinct taxa were present. I then filtered out any sequences that did not have the ful 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 subsampled 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 databse I used `fdnadist` to calculate a distance matrix using the Kimura 2-parameter evolution model, which allows for differnt 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 subsampled database were the only inputs of used for downstream analysis.

The ability of this refernece alignment to differntiate sequences of each taxon into every taxonomic level between speceis and class was tested.

Visulaiztions

Graphs were produced summarixing 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 form which the optimal threshold and error rate were caluculated.

Discussion

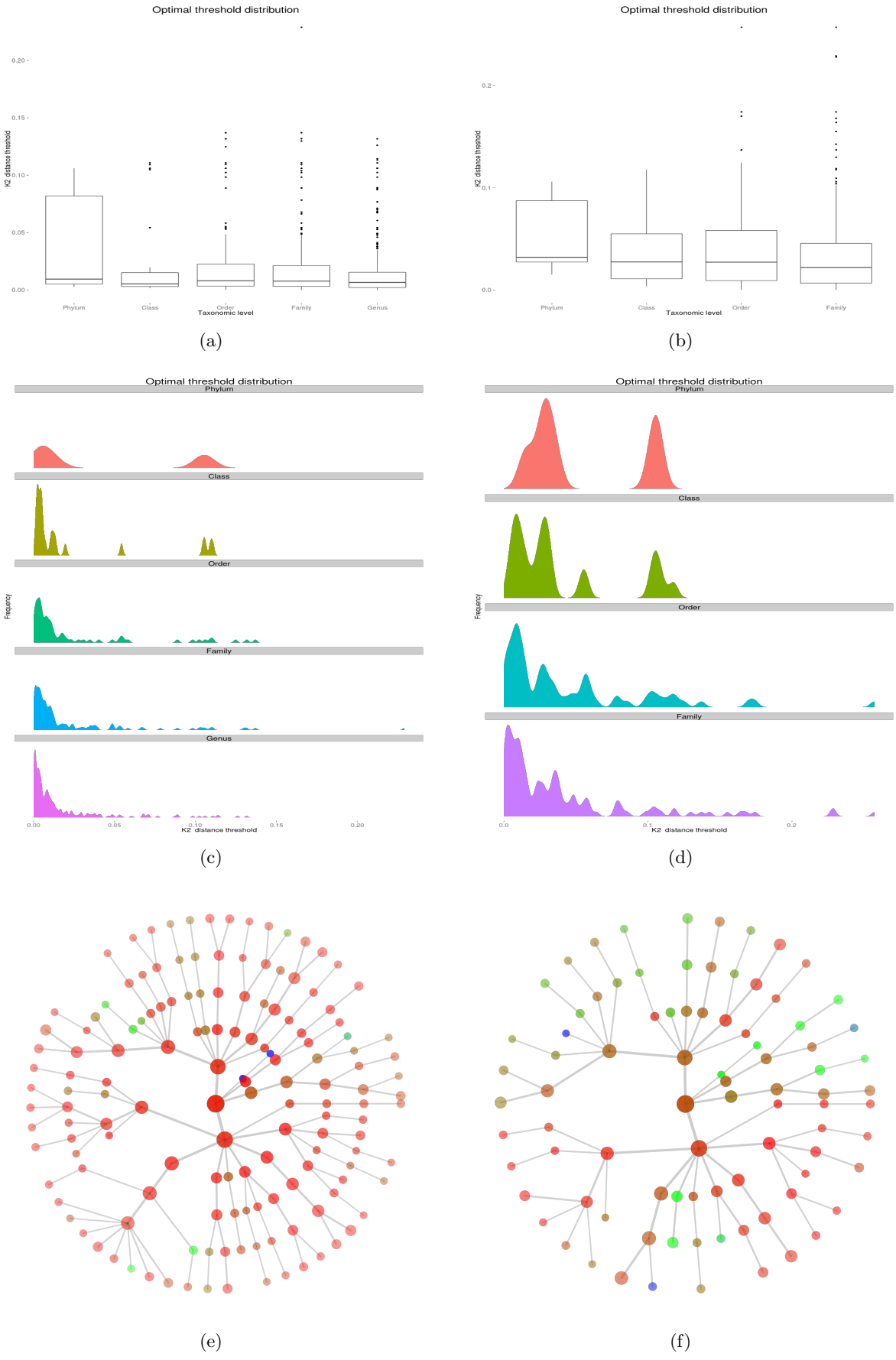

Figure 4: Optimal distance threshold statistics for differnetiating speceis [(a) (c), (e)] and genera [(b) (d), (f)]. For fingures (e) and (f) red is 0, blue is the maximum threshold, and green is the median.

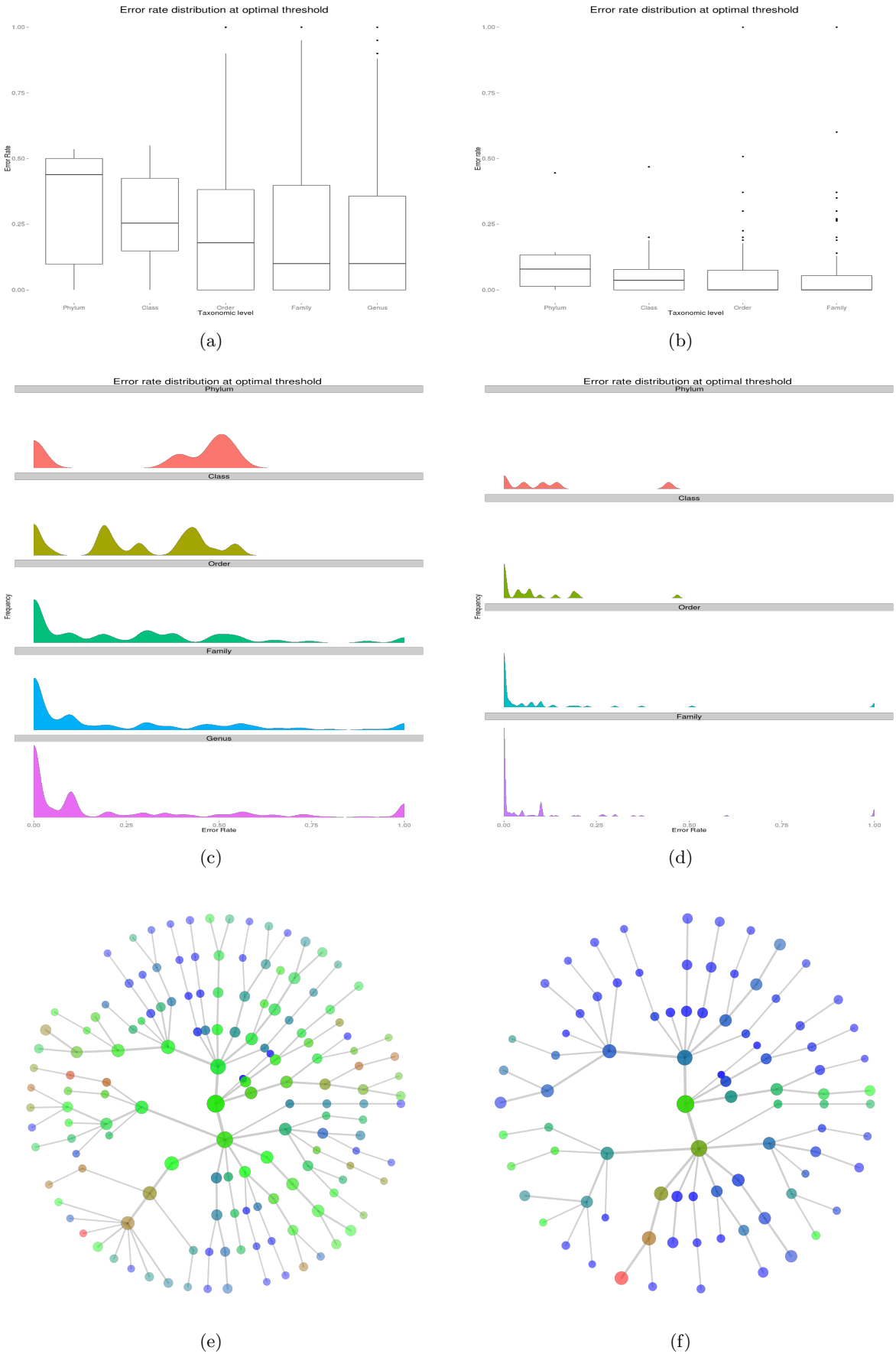

Figure 5: Error rates at optimal threshold statistics for differnetiating speceis [(a) (c), (e)] and genera [(b) (d), (f)]. For fingures (e) and (f) red is 0%, green is 50%, and blue is 100%.

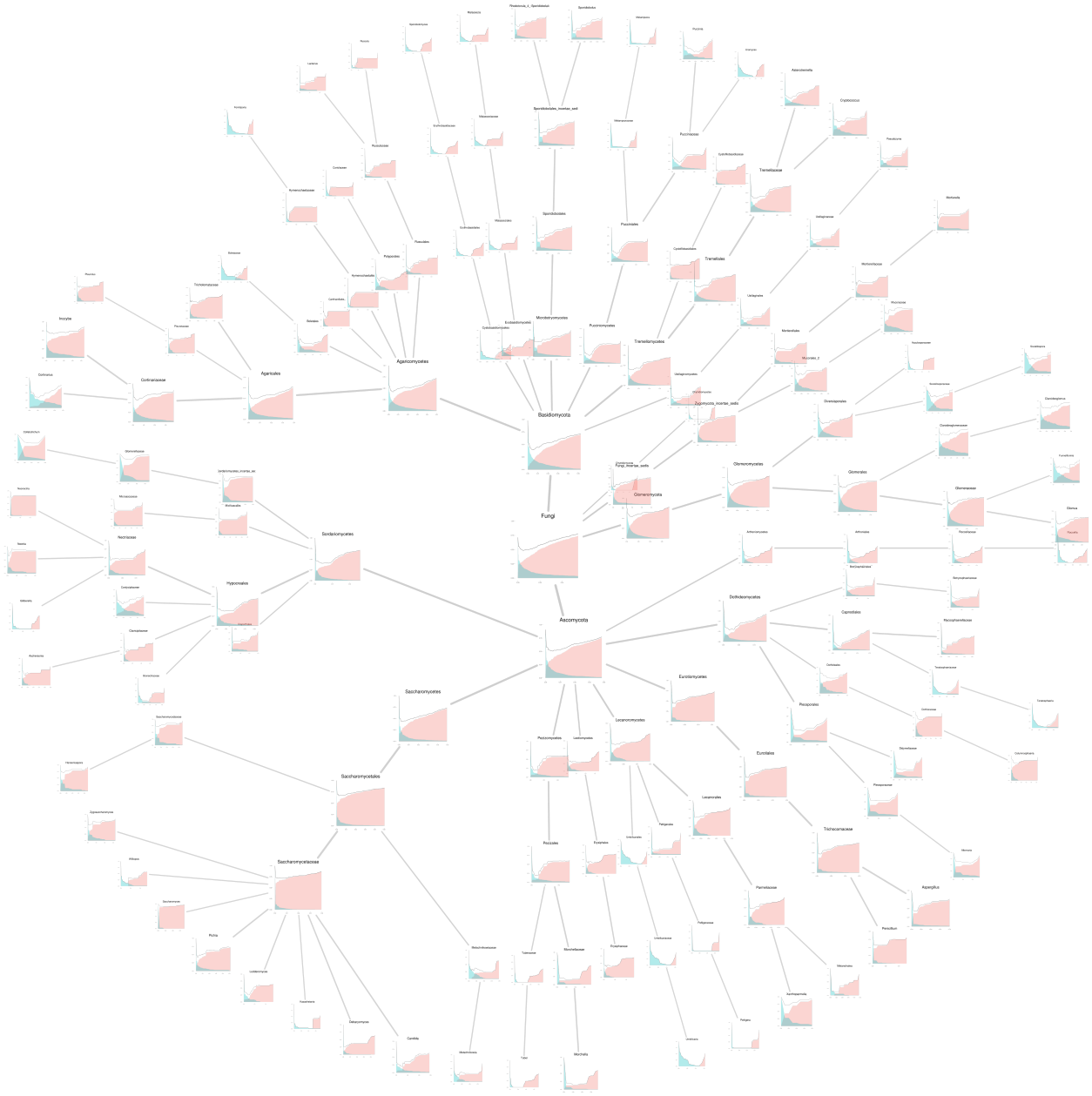

Figure 6: Threshold optimization graphs for differentating speceis. Each graph is a kernal density plot of false positive (blue) fasle negatives (red) and cumulative error (black line). The y axis is relative frequency and the x axis is clustering threshold. The lowest point in the black line is the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

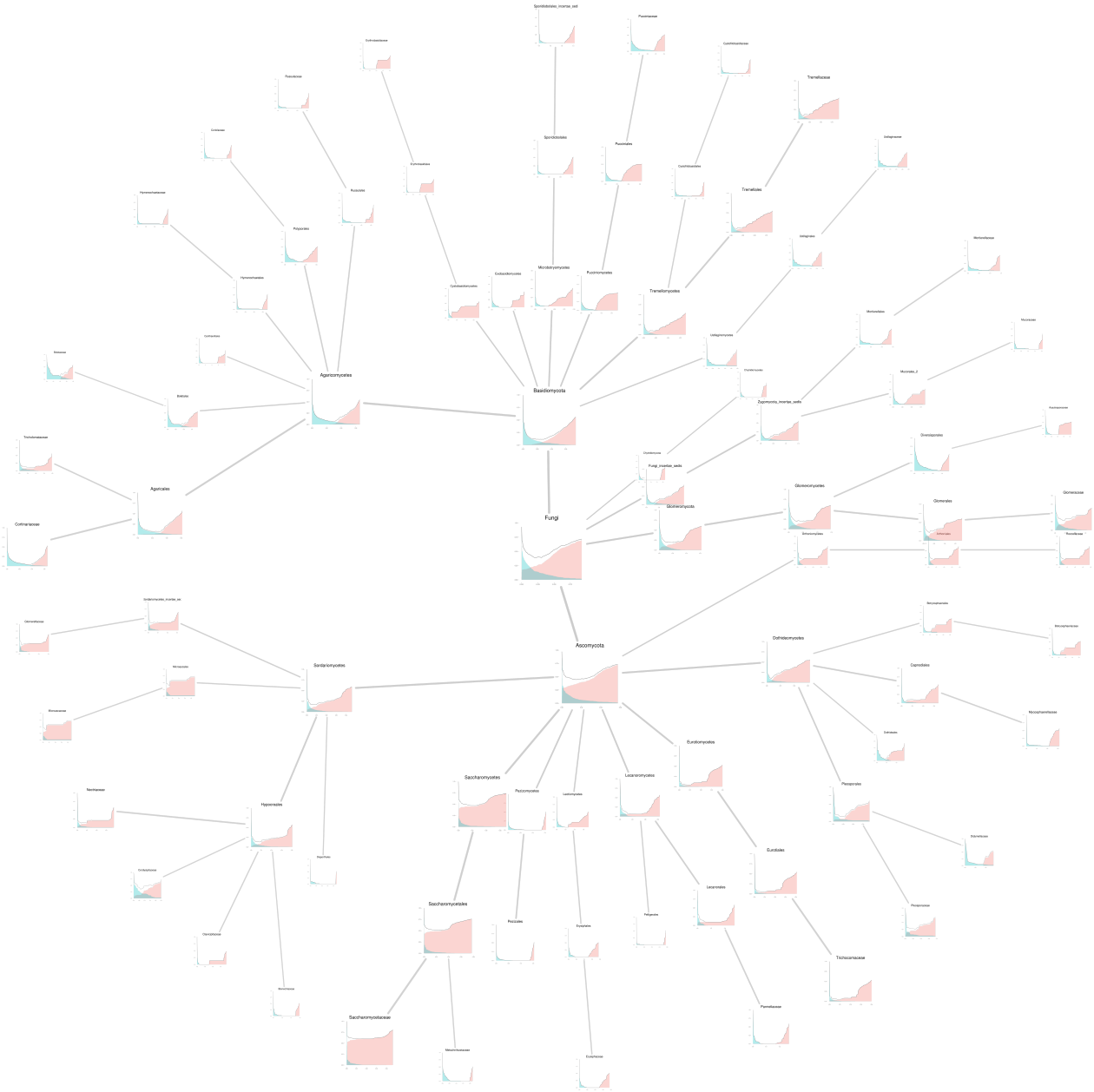

Figure 7: Threshold optimization graphs for differentating genera. Each graph is a kernal density plot of false positive (blue) fasle negatives (red) and cumulative error (black line). The y axis is relative frequency and the x axis is clustering threshold. The lowest point in the black line is the optimal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