Binning metagenomic data using ESOM
Brazelton Lab, May 2016
https://baas-becking.biology.utah.edu/

This workflow was used for the following publications:
Brazelton et al. (submitted) Metagenomic identification of active methanogens and methanotrophs in serpentinite springs of the Voltri Massif, Italy.

This workflow utilizes the Databionics Emergent Self-Organizing Map (ESOM) program and accompanying scripts by Gregory Dick's lab at the University of Michigan (https://github.com/tetramerFreqs/Binning and citations below). ESOM examines sequences as individual objects and groups them by tetranucleotide composition, generating clusters of 'like' sequences and separating 'unlike' sequences.

All Brazelton lab scripts are provide at our github site: https://github.com/Brazelton-Lab

Note that these commands are written exactly as implemented on the Brazelton lab server. Other users would have to adapt for their own system. For example, we use the cluster workload manager SLURM (http://slurm.schedmd.com/), which is why many commands begin with 'srun'.

References
Dick, G. J., Andersson, A. F., Baker, B. J., Simmons, S. L., Thomas, B. C., Yelton, A. P., & Banfield, J. F. (2009). Community-wide analysis of microbial genome sequence signatures. *Genome biology*, *10*(8).

Ultsch, A., Moerchen, F.: ESOM-Maps: tools for clustering, visualization, and classification with Emergent SOM, Technical Report Dept. of Mathematics and Computer Science, University of Marburg, Germany, No. 46, (2005)

Further reading on Self Organizing Maps: http://edugi.uji.es/Bacao/SOM%20Tutorial.pdf

Starting Point:
Everything below assumes that your metagenomic data has been processed and assembled according to our Metagenome Processing workflow. Specifically, you will need a fasta file of your assembled contigs and a bam file from mapping the short reads onto this assembly. You will also need the fastq.gz files of forward, reverse, and single reads.

Prepare metagenome assembly for ESOM using esomWrapper.pl:
1. The program esomWrapper.pl takes the following arguments:
-p, which requires a path to a directory containing the fasta files to be esommed (for the current working directory, simply enter '.' for the -p argument).

-ext, the suffix that each of them has. NOTE: the suffix does *not* include the '.' that precedes the suffix. This means that if all the files you hope to prepare for ESOM analysis end in ".fa", the argument passed for the program should be simply "fa".

Other options are: -prefix (which allows you to name your output files), -DIR (which allows you to name the directory the output files will be stored in), and -min and -max. -min is the minimum size of a contig that will be included in the map. Contigs that are bigger than -max will be split until further splits would produce fragments smaller than -max. With default settings (min=2500, max=5000), a 2.4 kb contig will be ignored, and a 14 kb contig will be split into 5 kb and 9 kb fragments.

The esomWrapper.pl program will generate 6 files: a .lrn file, a .cls file, and a .names file, as well as a .suffix file, a .ann file, and a .log file.

Example:

```
mkdir ESOM
cd ESOM
ln -svi ../assembly/Contigs.fasta
```

(Repeat for each set of contigs you want to include in the ESOM map)

```
srun esomWrapper.pl -p /home/bbrazelton/Liguria-ESOM-tracing/ESOM -ext fasta -DIR output
```

Link Phylosift results to the contigs:

2. We will use Phylosift to generate taxonomic classifications of the contigs in order to define the ESOM bins with additional information. You may have already run Phylosift if you already followed our Metagenome Profiling workflow, but if not, you can do it now. Use the same reads that were used to generate the assembly:

```
srun --cpus-per-task <CPUs> phylosift all --output phylogeny_paired --threads <CPUs>
--paired <forward_reads> <reverse_reads> &

srun --cpus-per-task <CPUs> phylosift all --output phylogeny_singles --threads <CPUs>
<single-end_reads>
```

3. Combine the Phylosift results from phylogeny_paired and phylogeny_singles:

```
cat phylogeny_paired/sequence_taxa_summary.txt
phylogeny_singles/sequence_taxa_summary.txt > all_sequence_taxa_summary.txt
```

4. Use this Python script to match the Phylosift results of short reads to the contigs to which those reads map. Use the bam file from the Metagenome Processing workflow and the .names file generated by esomWrapper.pl above.

```
srun esom_tracer2.py --bam <sample>.mapped.sorted.bam --names
ESOM/output/Tetra_esom_2500.names --taxonomy all_sequence_taxa_summary.txt --taxa_level
order --output <sample-phylosift-order>.cls
```

This script generates two files: [output].cls and [output].taxa. [output].cls will color contig chunks by their most probable phylogeny from Phylosift when loaded into ESOM after training.

[output].taxa correlates class numbers to taxa. Note that there are more taxa listed then actually present in the assembly; this is normal.

Optional: Link metabolic pathways to the contigs:
In the same way that we can link phylosift results to contigs, we can also link predicted protein functions and metabolic pathways to contigs visualized with an ESOM map. There are two complementary approaches for this, one that is very specific to a list of user-selected protein functions (e.g. KEGG IDs) and one that visualizes entire metabolic pathways selected by the user.

Approach 1: list of user-selected protein functions
5. If you have a list of protein functions of interest, we can use the script genes2contigs.py to find which contigs have those proteins and create a class from them in ESOM. The example below uses the key-genes-kegg-ids.txt list. Another option is methanogenesis-kegg-ids.txt. This assumes that you have already predicted genes in your assembly with Prokka.

```
srun genes2contigs.py /srv/data/mg/annotations/key-genes-kegg-ids.txt <assembly>.gff
<name-of-new-file> <Tetra_esom_2500>.names
```

6. Open this new file in Excel. The second column has contig names, and the third column has the corresponding data point numbers from the ESOM .names file. Copy and paste the data point numbers (third column) into TextWrangler and replace all of the commas with newlines (\n). Copy and paste these back into a new worksheet Excel so that the first column has this list of data point numbers corresponding to the contigs that contain the protein functions of interest. In the second column next to this list, assign a class number to each group of protein functions. For example, maybe you want to assign the same class number to all genes involved in methylotrophy.

7. Once you have a list of data point numbers and the new class numbers you want to assign to them, save this a tab-delimited file and make sure that the file has Unix linebreaks, not Mac or Windows linebreaks. (You can check this with TextWrangler or other text editing programs.) Then transfer this file to the server and use it to edit an existing ESOM .cls file:

```
srun edit-esom-class-file.py <existing-class-file>.cls
<your-list-of-data-point-numbers-and-class-numbers.tsv <name-of-new-class-file>.cls
```

8. You will have to manually add your new class numbers to the header of this new .cls file. Then load this new .cls file in ESOM, and you can color-code your classes as you wish.

Approach 2: selected metabolic pathways
This approach assumes that you have already predicted protein functions and metabolic pathways according to the Metagenome Profiling workflow. If not, you can come back to this step later.

9. First, we need to create a table that cross-references contig names and the metabolic pathway predictions generated by HUMAnN2 in the Metagenome Profiling workflow:
```
srun pathways2contigs.py <assembly>_pathcoverage.tsv <assembly>.gff
/srv/databases/pathways/foam
```

*You might want to replace `/srv/databases/pathways/foam` depending on which pathways you gave to HUMAnN2.

10. We can filter this table down to the pathways we are most interested in. Here is an example that includes all pathways with greater than 0.5 coverage:
```
srun filter-pathways-table.py <assembly>.IDs.contigs.tsv
<assembly>.IDs.contigs.0.5.all.tsv 0.5 all
```

Here is an example that includes only one specific pathway, regardless of coverage:
```
srun filter-pathways-table.py <assembly>.IDs.contigs.tsv
<assembly>.IDs.contigs.0.methanogenesis_formate.tsv 0 'Methanogenesis;From_formate'
```

11. Run esom_tracer3 with each of your filtered tables, which will produce a .cls file for ESOM specific to that filtered pathway table. Continuing with the same two examples above:
```
srun esom_tracer3.py --names <Tetra_esom_2500.names> --pathways_table
<assembly>.IDs.contigs.0.5.all.tsv --output <assembly>.IDs.contigs.0.5.all.cls

srun esom_tracer3.py --names <Tetra_esom_2500.names> --pathways_table
<assembly>.IDs.contigs.0.methanogenesis_formate.tsv --output
<assembly>.IDs.contigs.0.methanogenesis_formate.cls
```

12. Next, I opened all of the .cls files generated for each of my filtered pathway tables in Excel and combined them into a single spreadsheet so that I could see how many pathways were assigned to the same contig. I manually resolved any cases where a single contig was assigned multiple pathways and created a consensus class file.

Someday maybe we can automate many of the steps above with software, but some of it is inherently human because it involves an engaged user choosing protein functions and metabolic pathways that are interesting and/or important for some reason.

Generate ESOM map:

13. Log onto the Brazelton lab server with X11 forwarding enabled (i.e. add -Y to the ssh argument such as `ssh -Yp <port number> username@winogradsky.biology.utah.edu`).

14. Enter the command “esomana”. This will open the GUI for ESOM.

15. Load the .lrn, .cls, and .name files generated by esomWrapper.pl. (File > load .lrn etc.)

16. Normalize the data by selecting the ‘Robust ZT’ option under the Data tab.

17. In order to ‘train’ the data select Tools > Training.

18. Save the file under a new name in order to not overwrite the previous.lrn file.

19. Parameters: use default parameters with the following exceptions

- Training algorithm: Online training
- Number of rows/columns in map: The [prefix].log file will contain a suggestion for the number of rows x columns suitable for your data.
- Search method: local search with constant radius
- Start value for radius = 50 (increase/decrease for smaller/larger maps).

20. After entering these parameters, click ‘start’. Training of data can take anywhere from minutes to hours, depending on the size of the data set. This process will generate three output files: .wts, .umx, and .bm. The information will be displayed on the Umatrix viewer. For detailed instructions on interpretation and analysis, see: https://github.com/tetramerfreqs/binning and http://databionic-esom.sourceforge.net/user.html

Visualization and Binning:

21. After training, load the new [output].cls file generated above by esom_tracer2.py or esom_tracer3.py. If you do not already have the ESOM map loaded (i.e. if you quit esomana after training and launched it again), then load the .wts file that was generated by the training step. Check the "messages" tab in the esomana window to make sure that other files (.lrn, .umx, .bm, .names) were loaded correctly. After loading the [output].cls file, Phylosift results (or metabolic pathways, if using esom_tracer3.py) should appear as color-coded points in the ESOM map.

22. The ESOM interface can be used to reclassify groups of data points. In order to get a less-fragmented, more “full” view of the borderless map, try using the ‘tiled display’ option under the ‘View’ tab. Some other suggestion viewing options: yes countours, invgray gradient, color 8, clip 100%, bestmatch size 2. These are all optional and do not change the results; they are only visualization preferences. Feel free to experiment.

23. Click on the "Classes" tab in the bottom window. Draw polygons around each cluster of points that seem to be self-organizing into a bin. Click once with the mouse to begin drawing the polygon, click at each vertex of the polygon you want to draw, and then shift-command-click (on a Mac, at least) on the final vertex to finish the polygon. You can use shift-click to cancel the polygon. A new entry will appear at the bottom of the Classes list. You can edit the name of the class to remember what it corresponds to (e.g. Methanococcales). Save the .cls file frequently to record your changes.

Recover contigs in original assembly associated with each ESOM bin:

24. The script getClassFasta.pl can then be used to recover the full sequences of contigs inside each bin that you created. GetClassFasta.pl takes a .cls, .names, and the original contig file in

the folder that esomWrapper created (esom.fasta), and returns the fasta sequences of a given class. Run getClassFasta.pl for each class that you want to work with downstream. Example:
```
srun getClassFasta.pl -cls <CLASS File> -names output/Tetra_esom_2500.names -fasta output/esom.fasta -num <CLASS NUMBER>
```

25. Optionally, you can remove reference sequences that you included in the ESOM map but that you don't want to include in downstream analyses. The script fastachanger.py requires three arguments: the newly-created fasta file from the previous step, an output file name, and the start of the header sequences associated with the sequences to be removed.

26. If you have used Prokka to predict genes and proteins in the original assembly, you can recover the genes and proteins that correspond to each ESOM bin with these Python scripts:
```
srun get-sub-prokka.py <ESOM_bin>.fasta <assembly>.gff <assembly>.faa <assembly>.ffn
srun get-sub-prokka-gff.py <assembly>.gff fasta
```
*As currently written, the first script must be run on each bin separately. The second script will run on every file with the provided extension (e.g. fasta) in that directory. It would be nice if somebody merged these two scripts into a better one.

These scripts will generate three new files for each bin: a .faa file of the predicted proteins, a .fna file of predicted genes, and a .gff file of protein annotations.

27. Evaluate the completeness and contamination in each bin with CheckM. The command below will use the predicted proteins in the .faa files you generated above for each bin.
```
srun --ntasks 1 --cpus-per-task 4 checkm lineage_wf --genes -f checkm_table.tsv --tab_table -t 4 --pplacer_threads 4 -x faa <path/ESOM_bins> <path/checkm> 2> checkm.log &
```

The most useful output is the checkm_table.tsv containing the completeness and contamination numbers. You can also visualize these results with a plot:
```
srun checkm bin_qa_plot -x fasta <path/checkm> <path/ESOM_bins> <path/checkm/plots> &
```

28. (Optional) At this point you can try to merge bins together in an attempt to improve the checkm results (i.e. improving completeness without increasing contamination too much). I have been doing this manually, for example:

```
cat <bin1>.faa <bin2>.faa <bin3>.faa > <merged-bin>.faa
cat <bin1>.fasta <bin2>.fasta <bin3>.fasta > <merged-bin>.fasta
cat <bin1>.gff <bin2>.gff <bin3>.gff > <merged-bin>.gff

srun --ntasks 1 --cpus-per-task 4 checkm lineage_wf --genes -f checkm_table_<merged_bin>.tsv --tab_table -t 4 --pplacer_threads 4 -x faa <path/ESOM_bins> <path/checkm_merged> 2> checkm.log &
```

You can also try automated bin-merging software such as CONCOCT or ABAWACA (not described in this document). CheckM also has a merging function; here is an example with bins that are classified as Euryarcheota:

```
srun checkm taxon_set phylum Euryarchaeota euryarchaeota.ms

srun checkm merge -x fasta euryarchaeota.ms <path/ESOM_bins> <path/checkm/merged> &
```

--> When I tried this, nothing merged. Not sure why or how to evaluate it.

Identify Metabolic Pathways Separately for Each ESOM Bin

This section describes how to take the metabolic pathway annotations of the master assembly and partition them into your ESOM bins. It uses the .gff files generated above for each bin.

29. Run count_features and humann2 on each ESOM bin separately. These commands assume you want to predict protein functions with KEGG and group them into pathways with FOAM:

```
srun count_features.py --format bam --order position --type CDS --attr gene --norm rpk
--id-mapping /srv/databases/function/kegg_idmapping.tsv --mode union
<assembly>.mapped.f1.sorted.bam <ESOM-bin>.gff > <ESOM-bin>.kegg.function.tsv 2>>
<ESOM-bin>.kegg.function.log &

srun humann2 --input <ESOM-bin>.kegg.function.tsv --input-format genetable --output
<ESOM-bin>.kegg.foam --output-basename <ESOM-bin>.kegg.foam --o-log
<ESOM-bin>.kegg.foam.log --minpath on --xipe on --pathways-database
/srv/databases/pathways/foam --memory-use maximum &

srun krona_from_humann2.py <ESOM-bin>.kegg.foam_pathabundance.tsv >
<ESOM-bin>.kegg.foam_pathabundance.krona
```

(Repeat for each ESOM bin)

30. Make a Krona graph and summary tables that compare the pathway abundances and coverages for all bins:

```
srun ktImportText -o <set_of_ESOM_bins>_kegg_foam_pathabundance.krona.html
<ESOM-bin1>.kegg.foam_pathabundance.krona <ESOM-bin2>.kegg.foam_pathabundance.krona
<ESOM-bin3>.kegg.foam_pathabundance.krona

srun table_from_humann2.py --labels <ESOM-bin1,ESOM-bin2,ESOM-bin3>
<ESOM-bin1>.kegg.foam_pathabundance.tsv <ESOM-bin2>.kegg.foam_pathabundance.tsv
<ESOM-bin3>.kegg.foam_pathabundance.tsv > <set-of-ESOM-bins>.kegg.foam_pathabundance.tsv

srun table_from_humann2.py --labels <ESOM-bin1,ESOM-bin2,ESOM-bin3>
<ESOM-bin1>.kegg.foam_pathcoverage.tsv <ESOM-bin2>.kegg.foam_pathcoverage.tsv
<ESOM-bin3>.kegg.foam_pathcoverage.tsv > <set-of-ESOM-bins>.kegg.foam_pathcoverage.tsv
```

31. In addition to the tables of whole pathway abundances and coverages generated above, it might be useful to compare the abundance of a few select predicted protein functions (every

protein in one pathway, for example) across many samples. The compare-features.py script will find the abundance (from count_features.py) for each provided KEGG ID in every function.tsv file in the current directory:

```
srun compare-features.py [list-of-KEGG-Ids] [name-of-new-file]
```