Workflow for taxonomic and functional profiling of metagenomes
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.

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'.

Acknowledgements
A modified version of prokka is used to generate GFF files for count_features.py. The repository can be found at https://github.com/cnthornton/prokka. count_features.py uses a slightly modified version of the python library HTSeq [2].

References
[1] Seeman, T. (2014) Prokka: rapid prokaryotic genome annotation. *Bioinformatics*, 30(14): 2068-9. doi: 10.1093/bioinformatics/btu153.
[2] Anders, S., Pyl, P. T., Huber, W. (2015) HTSeq-a Python framework to work with high-throughput sequencing data. *Bioinformatics*, 31(2): 166-9. doi: 10.1093/bioinformatics/btu638.
[3] Abubucker S., et al. (2012) Metabolic reconstruction for metagenomic data and its application to the human microbiome. *PLoS Computational Biology*, 8(6): e1002358. doi: 10.1371/journal.pcbi.1002358.
[4] Ondov, B. D., Bergman, N. H., Phillippy, A. M. (2011) Interactive metagenomic visualization in a Web browser. *BMC Bioinformatics*, 12: 385. doi: 10.1186/1471-2105-12-385.
[5] Prestat, E., et al. (2014) FOAM (Functional Ontology Assignments for Metagenomes): a Hidden Markov Model (HMM) database with environmental focus. *Nucleic Acids Research*, 42(19): e145. doi: 10.1093/nar/gku702.

workflow overview:
- taxonomic profiling
  1. perform phylogenetic analysis on data
  2. cleanup output
- functional profiling
  1. predict and annotate coding DNA sequences
  2. calculate gene abundances
  3. obtain metabolic potential of the sample
  4 - 6. generate graphics

Note: this workflow assumes that you have at least one FASTA-formatted file of your assembled genome or metagenome.

Taxonomic Profiling

Investigate the composition of the microbial community:

```
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>
```

* preprocessing should already have been performed on the data prior to this point. The input reads will be the same reads used to generate the assembly.

(Optional) Compare phylosift results from multiple metagenomes in one table:

```
graphs2table.py -l 6 -o combined.phylosift.tsv <sample1>.html <sample2>.html
```

* the '-l 6' argument causes the table to be generated at taxonomic level 6 = family. level 5 is order, level 4 is class, etc.

Create an archive of unneeded files:

```
cd phylogeny_<singles_or_paired>
tar -cjf archive.tar.bz2 alignDir treeDir
cd ..
```

Functional Profiling

Find putative CDSs and annotate them using a user-specific protein database:

```
srun --cpus-per-task <threads> prokka --outdir features --prefix <sample> --proteins
/srv/databases/proteins/<gene_database>/<db> --metagenome --rfam --cpus <threads>
--mincontiglen <int> <sample>.contigs.fa 2> <sample>_prokka.log &
```

* prokka uses Prodigal for predicting coding DNA sequences (CDS), Aragorn for detecting tRNA and tmRNA genes, and Barrnap for predicting the location of rRNA genes. BLAST+ and HMMER3 are used for querying CDSs against the protein databases.
* three BLAST-indexed protein databases are available on the cluster: kegg, swissprot, and a filtered version of uniprotkb. The M5nr protein database is deprecated.

Calculate gene abundances, scale abundance estimates, and map genes to gene families:

```
srun count_features.py --format bam --order position --type CDS --attr gene --norm rpk
--id-mapping /srv/databases/function/<orthology>_idmapping.tsv --mode union
<sample>.mapped.sorted.bam features/<sample>.gff > <sample>.function.tsv 2>>
<sample>.annotations.log &
```

* the alignment file, <sample>.mapped.sorted.bam, is generated following assembly in the “MG processing” workflow. The BAM file should be sorted by either position in “genome” or by name. The default sorting method used by “samtools sort” is by position.
* HUMAnN2 requires the abundance counts in the gene table to be in units of reads per kilobase (rpk). RPK is calculated using the formula: (feature_count) / (feature_length / 1000).

* the ID mapping file associates genes with gene families. The appropriate mapping file to use will depend on the desired pathway ontology.

Determine the metabolic pathways that are present in the community:

```
srun humann2 --input <sample>.function.tsv --input-format genetable --output
<pathwaydb>_pathways --output-basename <sample> --o-log pathways.log --minpath on --xipe
on --pathways-database /srv/databases/pathways/<pathway_database> --memory-use maximum &
```

* currently there are four metabolic pathways databases available: FOAM, MetaCyc, UniPathway, and KEGG ontology (ko). Both ko and FOAM associate KEGG orthologs to a hierarchical representation of metabolic pathways. MetaCyc associates sets of genes from the UniProt Knowledge Base (UniProtKB), clustered according to 90% sequence similarity, to the reactions that they catalyze; the reactions are then associated with metabolic pathways. UniPathway uses the SwissProt proteins database, a manually curated subset of the UniProtKB.
* to use MetaCyc, both the metacyc1 and metacyc2 pathways files will need to be passed to the --pathways-database argument (separated by a comma with no spaces).

Make a multi-sample table of pathway abundances:

```
srun table_from_humann2.py --labels “<sample1>, <sample2>, …” <pathabund1> <pathabund2>
… > table_pathabundance.tsv
```

* this only applies if multiple samples have been run through the workflow and you wish to compare the pathways found by HUMAnN2.

Generate a Krona graph of the resultant pathways:

```
srun krona_from_humann2.py pathways/<sample>_pathabundance.tsv >
pathways/<sample>_pathabundance.krona
srun ktImportText -o pathways/<sample>_pathabundance.krona.html
pathways/<sample>_pathabundance.krona
```

* you can include multiple samples in the same Krona graph by specifying additional .tsv.krona files at the end of this command