Simple Analysis of Metatranscriptomes through Sequence Annotation (SAMSA) Pipeline

Version 2.0.0

User’s Manual/Documentation

Created by Sam Westreich

[swestreich@gmail.com](mailto:swestreich@gmail.com), <http://github.com/transcript/>

**Table of Contents**

1. Overview of SAMSA, functionality
2. Dependencies and included programs
3. The SAMSA Pipeline – Preprocessing
   1. Merging of paired-end reads
   2. Read cleaning
   3. Ribodepletion
4. The SAMSA Pipeline – Annotation
   1. Setting up DIAMOND databases
   2. Annotating against a DIAMOND database
   3. Converting DIAMOND results to readable format
5. The SAMSA Pipeline – Aggregation
6. The SAMSA Pipeline – Analysis
   1. Importing summarized results into R
   2. Differential Expression with DESeq2
   3. Generating stacked bar graphs with ggplot2
   4. Creating PCA plots and heatmaps with ggplot2
7. Further applications with SAMSA
   1. Subsystems hierarchy comparisons
   2. Organism-specific function filtering
8. Example files and workflow
9. Frequently asked questions
10. Contact and citation information

**Overview of SAMSA and its functionality**

In the analysis of microbiomes, complex microbial communities that occur in many different environments, there has been a recent shift towards high-throughput methods that capture both cultured and uncultured microbial species. Although 16S ribosomal profiling is still most commonly used, there is an increasing shift towards using deeper sequencing methods, such as metagenomics and metatranscriptomics. Of these high-throughput ‘omics methods, metatranscriptomics is advantageous in that it captures both organism abundance and species or strain level functional expression data, giving a more complete overview of the microbiome. However, there are still relatively few bioinformatics tools designed to handle this complex type of data.

SAMSA is the first open-source bioinformatics pipeline designed specifically for metatranscriptomic data. The first version of SAMSA was built specifically for researchers with a minimum of bioinformatics experience, who may not have a supercomputing cluster available for their use. SAMSA version 1 worked in conjunction with MG-RAST, a public annotation service capable of handling metagenomic and metatranscriptomic data. SAMSA version 1 helped with preprocessing and paired-end merging of RNA sequences, upload to MG-RAST, downloading of annotations, and analysis of the results.

While MG-RAST is still a valuable resource, some metatranscriptomics researchers may want more control over annotation specificity, databases and sequences used for matching, and speed of obtaining results. SAMSA version 2 is designed to meet those needs, and is built around DIAMOND, a superfast BLAST-like annotation algorithm that runs on a supercomputing cluster or on cloud-based services such as Amazon Web Services (AWS).

SAMSA version 2, this version, is intended to handle end-to-end analysis of metatranscriptomes. The reference databases can be chosen by the user, and it includes R scripts for running various analyses on the results, including measuring differential expression or activity for either the organisms present or the functional activity, as well as creation of graphics and figures.

**Dependencies and Included Programs**

**System Specs:** SAMSA is designed to run on a supercomputing cluster, or through a cloud-based cluster service, such as Amazon Web Services (AWS). All SAMSA scripts are designed to run in a POSIX (Linux, Unix, or Mac OS) environment.

While there are no explicit minimum memory requirements, DIAMOND-based annotation, the most RAM-intensive step in SAMSA, works much more rapidly on high-memory systems. Most testing was performed using 128 Gb of allocated RAM.

**External Programs:** SAMSA uses several other programs for steps in analysis; why reinvent the wheel, when someone else has built a perfectly useable version? The other programs used by SAMSA include:

* DIAMOND – this superfast BLAST-like algorithm is designed to handle annotation of many (millions) of input sequences, searching multiple reads in parallel against a formatted database. This forms the core of the annotation step in the SAMSA pipeline.  
  Version number:   
  DIAMOND can be accessed here: <http://ab.inf.uni-tuebingen.de/software/diamond/>
* FLASH – this lightweight program combines paired-end sequencing files to create a file of overlapping reads, as well as files of uncombined reads. If using single-end sequencing, this program can be skipped (although paired-end sequencing is recommended for metatranscriptomes).  
  Version number:  
  FLASH can be accessed here: <https://ccb.jhu.edu/software/FLASH/>
* Trimmomatic – this program removes low-quality sequences and checks for Illumina adaptor contamination in the raw sequence files.  
  Version number:  
  Trimmomatic can be accessed here:
* Python – all SAMSA Python scripts were built to run in Python 2.7, and are unfortunately incompatible with Python 3. SAMSA uses the following Python modules:
  + sys
  + os
  + subprocess
  + glob
  + time
  + gzip
  + operator
* R – the analysis scripts use R, and while they may be run from the command line, they are more easily used in RStudio. The following R packages are used:
  + DESeq2
  + ggplot2
  + gridExtra
  + scales
  + knitr
  + reshape
  + plyr
  + pheatmap
  + RColorBrewer

R version number: 3.0

**Included Programs:** Most of the SAMSA pipeline scripting is written in Python, version 2.7. NOTE: Much of the code is not compatible with Python 3. The following programs are included in the SAMSA pipeline:

* master\_script.bash – Don’t want to mess with flags, remembering where your files are saved, or typing out different commands for each step? Open up this file and insert your paths and values, and all pipeline scripts will check here first.
* analysis\_counter.py – After DIAMOND annotation is complete, this Python program compares the output back against the database to provide a sorted abundance table for use in R scripts.
* subsystems\_analysis\_counter.py – Similar to analysis\_counter, but this Python program compares the output specifically for the SEED Subsystems database, allowing for hierarchical functional clustering of results.
* subsystems\_hier\_condenser.py – Subsystems hierarchy results are provided at level 4 (function) by default; this Python program condenses hierarchy up to level 3, 2, or 1 (top-level, broadest categories).
* run\_DESeq\_stats.R – R program that calculates differential expression using DESeq2, either of organisms or of functions, from the sorted abundance tables generated by analysis\_counter.
* diversity\_stats.R – R program that calculates Shannon or Simpson diversity from the sorted abundance tables generated by analysis\_counter.
* make\_DESeq\_PCA.R – R program that uses DESeq2, pheatmap, ggplot2, and RColorBrewer to create PCA plots from the sorted abundance tables generated by analysis\_counter.
* make\_DESeq\_heatmap.R – R program that uses DESeq2 to create a heatmap from the sorted abundance tables generated by analysis\_counter.
* make\_combined\_graphs.R – R program that uses ggplot2 to create stacked bar graphs, one per sample, from the sorted abundance tables generated by analysis\_counter.

OTHERS WILL LIKELY BE ADDED HERE

**The SAMSA Pipeline – Overview**

****

**The SAMSA Pipeline – Preprocessing**

So, you’re looking to analyze some metatranscriptome data! You’ve got your fastq files, hot off the sequencer. Before these files can be annotated, they need to be merged together (if you performed paired-end sequencing) and cleaned to remove low-quality reads and adaptor contamination.

*Paired-end read merging*

If paired-end sequencing was performed, there will be two sequence files for each sample provided; they usually have names ending in “R1” and “R2”, or some variant. R1 contains the forward reads, while R2 contains the reverse reads.

The easy way to merge these two files together is using a read merging program, such as FLASH (Fast Length Adjustment of Short reads)( <https://ccb.jhu.edu/software/FLASH/)>. This lightweight merging program can run on a desktop computer, and can be downloaded through SourceForge. Once downloaded, run the following commands:

$ tar xvf FLASH-1.2.11.tar.gz $ cd FLASH-1.2.11/

$ make

To use FLASH for merging two paired-end reads, run it from the command line, using the following command:

$ ./flash MATES\_1.fastq MATES\_2.fastq

To see FLASH options, use the following command:

$ ./flash --help | less

**Result:** A fastq sequence file with overlapping paired-end reads merged together. Additionally, separate files are produced for the not combined reads; these may be included as well if the user chooses.

*Removal of adaptor contamination and low-quality reads*

Raw sequences may contain low-quality reads, or reads with “contamination” – the adaptor sequences used for the process of sequencing may have been accidentally read as part of the read. These contaminated and low-quality sequences should be removed to avoid skewing the results of a metatranscriptome analysis.

* There are many programs available for cleaning and removing adaptor sequences from raw sequence files; this pipeline is set up to use Trimmomatic (<http://www.usadellab.org/cms/?page=trimmomatic>). This Java application includes primer sequences for Illumina machines, and can be used to filter out adaptor sequences from either single-end or paired-end sequencing.

The command is structured like so:

$ java -jar trimmomatic-0.33.jar SE –phred33 $infile $outfile\_name SLIDINGWINDOW:4:15 MINLEN:99

Details on these parameters, as well as other commands, can be found in the Trimmomatic manual. That manual can be accessed here: <http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf>

**Result:** A fastq sequence file with low-quality sequences and adaptor contamination removed.

*Removal of ribosome sequences*

One issue when sequencing extracted RNA is filtering out mRNA from the much more common ribosomal RNA, or rRNA. Although rRNA comprises the majority of all extracted RNA from microbiome communities, it can obscure the more important mRNAs. For best results, ribodepletion methods should be used on the biological samples after RNA extraction and before sequencing as a quality control step.

However, biological ribodepletion kits are not completely effective at removing all ribosomes. Stripping out remaining ribosomal reads will help increase the speed of downstream pipeline steps, by resulting in fewer total reads to be annotated (the slowest and most computationally intensive step) and analyzed.

SortMeRNA (<http://bioinfo.lifl.fr/RNA/sortmerna/>) is a robust ribosomal read filtering tool that can incorporate multiple databases (SILVA, GreenGenes, RDP) for rRNA identification. Note that SortMeRNA was originally designed to select rRNA sequences, rather than to remove them, and so the reads discarded by SortMeRNA are, in fact, the mRNAs needed for metatranscriptome analysis.

For detailed instructions on using SortMeRNA, be sure to consult the included user manual for version 2.1. SortMeRNA can be installed using the build command when downloaded from Github, or can be downloaded as a precompiled tarball release.

Note that the “--other" flag MUST be applied when using SortMeRNA! Without this flag, all reads that do not match the ribosomal RNAs in the reference will be discarded. These reads are, in fact, the mRNAs, and must be preserved for the following steps in the SAMSA pipeline.

Example SortMeRNA command (matching against the 16S SILVA bacterial database, included in SortMeRNA download):

$ sortmerna --refsilva-bac-16s-db --reads $file.fastq --aligned $file.ribosomes --other $file.ribodepleted --fastx --num\_alignments 1 --log –v

From this command, two files will be produced; the $file.ribosomes will contain all sequences from the original file identified as rRNA, while the $file.ribodepleted will contain all reads discarded by SortMeRNA (aka not identified as ribosomes, to be used in the next step of the SAMSA pipeline).

Note that while the identified ribosomal sequences can potentially be used for other analyses, the biological ribodepletion that is strongly recommended for all samples before sequencing will likely skew these results, making them unusable for organism-specific abundance measurements.

**Result:** A fastq sequence file with ribosomal sequences removed; additionally, a second file is created containing said ribosomal sequences for optional taxonomic profiling.

**The SAMSA Pipeline - Annotation**

Now that the initial sequences have been merged (if using paired-end sequencing), cleaned, and stripped of adaptor sequence contamination and ribosomal sequences, the next step is to annotate the mRNA reads to their corresponding match in a reference database. Given that a metatranscriptome may contain many tens of millions of reads, some of the older approaches to annotation, such as BLASTX, are simply too slow to complete the task in a reasonable amount of time.

SAMSA instead uses DIAMOND, a superfast BLAST-like aligner, to handle the job of annotation. This tool can process reads up to 10,000x as fast as BLASTX, with very little loss in accuracy. DIAMOND can also annotate against any provided database, allowing for custom databases to be created and searched against. For an example bash script that shows the different commands for interacting with DIAMOND, take a look at the “DIAMOND\_example\_script.bash” included in this pipeline’s suite of files.

*DIAMOND installation*

DIAMOND may be downloaded from Github as a binary; once downloaded, it can be extracted from the tarball and put to use.

*Creating a DIAMOND-indexed database*

Any database file needs to be indexed by DIAMOND and converted into a binary file (.daa extension) before it can be searched against. DIAMOND will convert any fasta file to a usable database with the following command:

diamond makedb --in $database --db $database

The --in flag specifies the starting fasta file that will be converted to a DIAMOND-searchable database.

While DIAMOND can be given any database file to be indexed, two databases that should sufficiently serve the needs of most microbiome researchers are the NCBI RefSeq database and the SEED Subsystems database. Maintained by NCBI, RefSeq is one of the most complete databases for general purposes and is generally accepted to contain high-quality annotations. SEED Subsystems offers the unique ability to sort specific functions into hierarchies, letting similar functions be grouped under a category heading, such as “cellular respiration” or “protein biosynthesis.” This can be very useful for examining overall functional activity within a metatranscriptome.

The RefSeq database can be accessed through NCBI’s FTP site, here: <ftp://ftp.ncbi.nlm.nih.gov/refseq/release/complete/> . The simplest approach is to download all non-redundant protein sequence files, use cat on the command line to merge them together into a single gzipped file:

$ cat file1.gz file2.gz file3.gz > all\_files.gz

The SEED Subsystems database can be accessed through their FTP site here: <ftp://ftp.theseed.org/subsystems/>

Note, however, that the SEED Subsystems database is not readily downloadable in a fasta format that can be indexed by DIAMOND. The different levels of Subsystems hierarchy are maintained in different files. For merging these files together to create a single, indexable database that contains all hierarchy information, see the relevant Github repository here: <https://github.com/transcript/subsystems>

*Annotating a file against a DIAMOND database*

Using DIAMOND to annotate a sequence file against a database is a two-step process; the first step is to perform the requested annotation search, while the second step converts the results file from a binary (.daa) format to a more readable BLAST m8 table.

To perform the annotation search:

diamond blastx --db $diamond\_database -q $filename -a $diamond\_output -t ./ -k 1

For an explanation of the different flags and options, see the “DIAMOND\_example\_script.bash” file included with the SAMSA Github repository.

To convert the generated results file to a viewable BLAST m8 data table:

diamond view --daa $diamond\_output -o $final\_output -f tab

The resulting data table is ready for step 3 in the SAMSA pipeline: aggregation.

**The SAMSA Pipeline - Aggregation**

Now that each metatranscriptome file has been annotated, the next step is to reduce the results down into a condensed and simplified format for statistical analysis. DIAMOND returns the best match for each read in the starting file that meets its parameters for sequence specificity, much like a line-item receipt from a grocery store. This step converts this large file into a condensed, sorted summary table that returns the total number of hits to each specific organism or function.

NOTE: This step will create two summary files for each starting metatranscriptome; one file will contain annotations grouped by organism (all Bacteroides reads will be grouped together), while the other file will contain annotations grouped by function (all reads coding for the enzyme lactase will be grouped together). Later steps will document the steps necessary to perform a search for all functions expressed by a specific organism or group of organisms, or vice versa, all organisms performing a specific function or set of functions.

This next step uses the Python program “DIAMOND\_analysis\_counter.py”, and for the RefSeq database, will require access to the original (readable, not DIAMOND-converted) database file.

To use this program for aggregating all reads by organism:

python DIAMOND\_analysis\_counter.py –I $infile –D database\_file –O

And to use this program for aggregating all reads by function:

python DIAMOND\_analysis\_counter.py –I $infile –D database\_file –F

The resulting files can either be viewed directly, or can be imported into R for further statistical analysis and figure generation.

**The SAMSA Pipeline – Analysis**

The aggregated results files generated by the Python script in step 3, aggregation, can be used for several different comparisons and analyses in R. Note that, while some of these scripts can be run in base (command line) R, they are more optimally handled using RStudio, which allows for evaluation of intermediate data tables and the ability to better isolate any issues that may arise due to differences in database and output structure.

*Statistical analysis and differential expression using DESeq2*

Perhaps the most important question when comparing metatranscriptomes is that of differential expression – which organisms or functions exhibit a significantly different level when comparing between control and experimental results? The R program “run\_DESeq\_stats.R” evaluates this question.

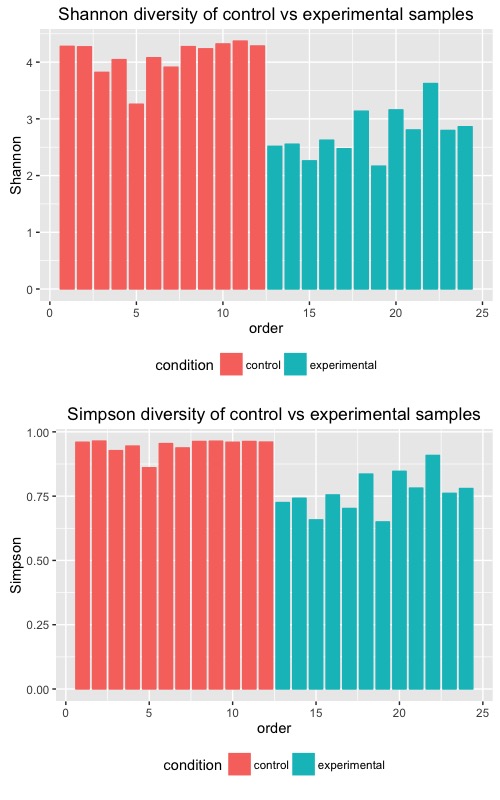
This R script, if run from the command line, takes two arguments: the directory location of the summary files that will be analyzed as ARGV1, and the name under which the results should be saved as ARGV2. In the specified directory, control file names should begin with “control\_”, and experimental file names should begin with “experimental\_”.

**Result:** This R program outputs a tab-delimited results table that, for each organism or function in the infiles, prints out the controlMean, the experimentalMean, the log2FoldChange, and the absolute and multiple-hypothesis-adjusted p-values.

*Diversity measures using organism data*

Two statistics, Shannon and Simpson diversity, can provide a useful overview of the diversity of a microbiome. Two R programs evaluate this measure: “diversity\_stats.R” will calculate the average diversity seen in each sample group, while “diversity\_graphs.R” will create graphs of the diversity statistics for each individual sample.

**Result:** “diversity\_stats.R” will return the mean Shannon and Simpson diversities for both the control and experimental groups. “diversity\_graphs.R” will create two graphs, one showing Shannon diversity calculated for each individual data sample, the other showing Simpson diversity calculated for each individual data sample.



*Combined graphs of top (most abundant) organisms or annotations*

A stacked graph showing the top organisms (or functions) within all metatranscriptomes in a project provides a useful overview of results. “make\_combined\_graphs.R” is an R script that creates these graphs for either the top most abundant organisms or functions. Similar to the above R programs, it reads in summaries of all metatranscriptome files and generates a stacked bar graph from this data.

**Result:** “make\_combined\_graphs.R” will generate a PDF containing two stacked bar graphs; the top bar graph will reveal relative expression (in percentages), while the lower bar graph will reveal absolute expression (count numbers). A legend is displayed beneath. Control samples are displayed on the left, while experimental samples are displayed on the right. Each bar represents a single metatranscriptome input file.



*PCA plots of metatranscriptomes*

A PCA plot is useful for showing clustering between different metatranscriptomes, demonstrating that different samples can be regarded as different groups or categories. The R program “make\_DESeq\_PCA.R” uses the DESeq2 package to generate the data structure for this graph, and then creates it with ggplot2.

**Result:** A PCA plot showing experimental vs. control samples. Further customization can be performed in the script if desired to provide different shapes for different individual samples.



*Creating a heatmap of differential organism or functional activity*

Especially with functional RNA expression, heatmaps are a useful method for displaying the degree of change between samples. The R script “make\_DESeq\_heatmap.R” creates a heatmap comparing control vs. experimental samples, and can take either organism or function summaries as its input. This script uses pheatmap, and RColorBrewer packages for creating a heatmap.

**Result:** A heatmap showing experimental vs. control samples. This can be run to show levels of change either in organisms, or in functions, when contrasting between RefSeq results.



**Further Applications with SAMSA - Subsystems**

**Subsystems Hierarchy Comparisons**

While the detailed information of a metatranscriptome is incredibly powerful, it can feel overwhelming to consider hundreds of thousands of different annotated functions, when multiple thousands may be noted as significantly differentially expressed. One unique method for simplifying the “at-a-glance” metatranscriptome is to use a hierarchical database, such as TheSeed’s Subsystems system of annotations.

TheSeed’s Subsystems database (referred to hereafter as “Subsystems”) features four increasing levels of hierarchy for each annotated function. They’re best imagined as a fractal tree, with specific functions forming the leaves. For example, a specific sequence might be linked to TcuB tricarballylate oxidation (level 4 subsystem). This, in turn, falls under the category of “Tricarballylate utilization” (level 3 subsystem). This falls into the level 2 subsystem of “organic acids”, which is in the level 1 subsystem category of “Carbohydrates.”

Another way of viewing it:

1. Carbohydrates
   1. Organic acids
      1. Tricarballylate utilization
         1. TcuB tricarballylate oxidation to cis-aconitate

The Subsystems database can be reworked into a SAMSA-searchable database, and metatranscriptome hits against this database can be compared at each hierarchy level. Thus, instead of sorting through 100,000+ specific functions, it’s possible to obtain a metatranscriptome “overview” by looking only at differences in the top 20 or so level 1 Subsystems categories.

**Annotating against a flattened Subsystems database**

Unfortunately, the Subsystems database, although available for download through an FTP portal (<ftp://ftp.theseed.org/subsystems/)>, is not pre-configured in a format that makes it indexable by algorithms like DIAMOND. For a more detailed view of obtaining a flattened, DIAMOND-indexable version of the Subsystems database, see the following Github page: <https://github.com/transcript/subsystems>. Using the Python program “subsys\_db\_rebuilder.py” and the other files mentioned in the Github readme, the Subsystems database can be flattened into a single file containing all hierarchy, which can then be indexed by DIAMOND and used as a searchable database, much like the NCBI RefSeq database used in earlier steps.

Once the Subsystems database is converted to a flattened format, running a DIAMOND annotation search against it is nearly identical to searching against RefSeq, in the earlier section of this guide:

diamond blastx --db $subsystems\_database -q $filename -a $diamond\_output -t ./ -k 1

To convert the generated results file to a viewable BLAST m8 data table:

diamond view --daa $diamond\_output -o $final\_output -f tab

**Converting annotations to hierarchies**

The DIAMOND results will return the match for each given read to the specific Fig ID, the base unit of reference in the Subsystems database. To obtain the full hierarchy, a Python program needs to go back to the database and recheck each read against the reference, adding in the necessary information.

The Python program that accomplishes this is called “DIAMOND\_subsystems\_analysis\_counter.py”, and operates very similarly to the “DIAMOND\_analysis\_counter.py” program used for RefSeq results. Similarly, this analysis counter program reads back through the DIAMOND output and condenses the reads down into a sorted abundance list – but it also includes all hierarchy information, if available, printed in additional columns in the tab-delimited output.

Usage:

python DIAMOND\_subsystems\_analysis\_counter.py -I $file -D $subsystems\_db -O $file.analyzed -P $file.receipt

Note that, as the Python program does not read binary files, the linked Subsystems database must be the flattened file created for conversion to a DIAMOND-indexable database (but NOT the converted .daa file!). The output will be saved under the name given with the –O flag, while the optional –P flag will create another file containing a line-by-line “receipt” with all hierarchy included.

The resulting output file from this is ready to be imported into R for analysis, either for calculating significantly differentially expressed categories of functions, or for the creation of pie charts reflecting the relative levels of reads towards each Subsystems category.

**Analyzing Subsystems annotations for differential expression in R**

Because the Subsystems results from the Python aggregating program contain additional columns with hierarchy information, a separate R script is needed to analyze the results. These two R scripts are named “Subsystems\_DESeq\_stats.R” and “Subsystems\_pie\_charts.R”.

For figuring out significantly differentially expressed genes at each of the different hierarchy levels, “Subsystems\_DESeq\_stats.R” loads in the Subsystems results files, parses the resulting data tables down to only the requested hierarchy level, and then runs DESeq2 to figure out differential expression. **Note that this script cannot be run without modification from command line R; in RStudio, only certain lines (choose from lines 74-91) need be run. Each block corresponds to a different hierarchy level.**

**Result:** A data table containing DESeq evaluated differentially expressed transcripts for the selected hierarchy level, saved in a tab-delimited format.

**Analyzing Subsystems annotations for creating pie charts in R**

Similar to the above program, “Subsystems\_pie\_charts.R” will load in the Subsystems files, compress down to only the selected level of hierarchy, and will then create a pie chart. Note that currently, “Subsystems\_pie\_charts.R” does NOT choose to make two different pie charts for experimental vs. control files. Instead, it will create a single pie chart. If two contrasting pie charts are desired, run the program twice, feeding it each different group of files (once with the experimental files, a second time, separately, with control files).

By default, this script will produce output graphs of level 1 results. For different levels, change the level indicator on line 49 (note that higher levels may provide too many files and will require the use of additional colors).

**Result:** A pie chart, saved as a PDF under the given save filename. Details can be altered by changing the ggplot2 graph created at the end of the program.

**Further Applications with SAMSA – Functions by Organism**

Although looking at the functional activity of all organisms can provide a useful overview of a metatranscriptome and may offer suggestions as to where to focus further investigation, it’s also important to have the ability to narrow the focus and only examine the outputs linked to specific organisms of interest. For example, a gut microbiome study may be particularly interested in the activity of Lactobacillus species, and would like to restrict functional annotations to solely this genus.

Fortunately, because each read processed by SAMSA receives both an organism and a functional annotation, this is possible! This is made possible using a Python program named “DIAMOND\_specific\_organism\_retreiver.py”.

This program takes three inputs – the input file, which is the DIAMOND results file from annotating against the RefSeq database (-I flag), the specific organism to be selected (-SO flag), and the RefSeq database file (-D flag)(note: make sure to specify the original fasta file, not the DIAMOND-indexed binary). The program first reads through the RefSeq database, constructing a dictionary of RefSeq IDs and their matching organisms. It next reads through the infile, checking each annotated RefSeq ID’s entry to see whether the organism of interest’s name is present in the dictionary entry. Only those entries with the chosen organism name present are passed on to the outfile (which is automatically generated, named after the combination of the target infile and the specific organism).

**Result:** The program produces an output, named after the infile and the specific organism it’s searching against. This output file contains all individual transcripts that originated from the specific organism; the next step in the pipeline is to use DIAMOND\_analysis\_counter.py to reduce this down to a sorted abundance list.

**Example Files and Workflow**

When pulling SAMSA version 2.0 from the Github repository, sample files are provided for testing at each step in the pipeline, as well as for comparison to see whether produced files match up with the samples. An example master script is also provided, set up to run on the provided sample files.

The following starting sample files are included in the Github repository for SAMSA v2.0:

**Frequently Asked Questions (FAQs)**

Currently, there are no questions asked as of yet. More will be added as they come in!

**Contact Information – How to Cite SAMSA**

This pipeline has been designed and constructed by Sam Westreich, working with Dr. Danielle Lemay at the Genome Center, at the University of California, Davis. For questions, please contact him at:

[stwestreich@ucdavis.edu](mailto:stwestreich@ucdavis.edu)

Or if there is a code issue, or you wish to adapt the SAMSA code for a different purpose, it can be forked from Github: <https://github.com/transcript/samsa_v2>

**Citation information**

This second version of SAMSA is not yet published, but the original SAMSA paper can be cited as follows:

Westreich, S.T., Korf, I., Mills, D.A., Lemay, D.G. (29 September 2016). SAMSA: A comprehensive metatranscriptome analysis pipeline. BMC Bioinformatics, 17(1):399.