Simple Annotation of Metatranscriptomes through Sequence Annotation (SAMSA): A Pipeline for Metatranscriptome Analysis

Created by Sam Westreich
stwestreich@ucdavis.edu
http://www.github.com/transcript/

Introduction to SAMSA (and core dependencies)

Hello, and thank you for considering using SAMSA! This pipeline is designed to take raw sequencing reads from an Illumina run and match those reads against a reference database, and then examine the annotations for significant changes in either organism activity, or in the expression levels of various transcripts.

Although metatranscriptomics—the study of diverse microbial population activity based on RNA-seq data—is rapidly growing in popularity, there are limited options for biologists to analyze this type of data. Current approaches for processing raw metatranscriptome data rely either on restricted databases, a personal analysis server, or use metagenome-based approaches that have not been fully evaluated for use in processing metatranscriptomic datasets. I have created a new bioinformatics pipeline, SAMSA, designed specifically for metatranscriptome dataset analysis, which runs either in-house or in conjunction with Metagenome-RAST (MG-RAST) servers. Designed for use by researchers with relatively little bioinformatics experience, SAMSA offers a breakdown of metatranscriptome activity by organism or transcript function, and is fully open source.

SAMSA was constructed in a Mac OS X environment, and is designed to run in Unix systems. The majority of the pipeline is written in Python, with R scripts used in the final steps for performing statistical comparisons and generating visual graphs of the metatranscriptome composition.

Dependencies

- Unix/Mac OS environment. If running on a Windows computer, Cygwin may be used, although all files must be moved into the Cygwin directory.
- Python. This pipeline was constructed to run on Python 2.7.
- Python modules:
    - sys
    - os
    - subprocess
    - glob
    - time
    - gzip
    - operator
    - commands
- R, or more preferably, RStudio. Analysis R scripts use the following R packages:
    - DESeq2
    - ggplot2
    - gridExtra
    - scales
    - knitr

    - reshape
    - Plyr
- Trimmomatic, a flexible read-trimming tool for NGS data. Documentation and download links for Trimmomatic may be found here: http://www.usadellab.org/cms/?page=trimmomatic .
- FLASH, Fast Length Adjustment of SHort reads, a tool for merging paired end reads. If the sequencing data is not paired end, this tool is not necessary - although paired end data is more accurate. FLASH download and documentation links may be found here: https://ccb.jhu.edu/software/FLASH/ .
- A working internet connection. SAMSA needs to both upload and download files from MG-RAST's servers, which may require several hours of uninterrupted access.
- Memory: all SAMSA commands have been shown to work on a laptop with only 8 Gb of RAM, but more ram will increase processing speed.

Note that, at each stage in the pipeline, SAMSA generates output files, which serve as the inputs for the next stage in data processing. This allows for the pipeline to be paused at any step, and different steps may be run on different machines by transferring the files and pipeline programs.

Now, ready to get started?

SAMSA Pipeline Use: Streamlined Wrapper Script for pre-annotation

For ease of use, several wrapper programs exist to run the various steps in this pipeline with a minimum of interacting with the command line. These wrapper scripts are entitled:

1. SAMSA_pre_annotation_pipeline.py: this wrapper script performs the following steps:
    a. Read cleaning and adaptor removal with Trimmomatic
    b. Paired-end file aligning with FLASH (if the "paired" option is enabled)
    c. Uploading of files to MG-RAST
    d. Sequence statistics computation with MG-RAST
2. SAMSA_post_annotation_pipeline.py: this wrapper script performs the following steps:
    a. Downloading organism and functional annotations from MG-RAST
    b. Running SAMSA analysis scripts on downloaded annotation files
    c. Cleans up summary output files for import into R for analysis

Pre-annotation wrapper program

SAMSA_pre_annotation_pipeline.py has the following usage options:

REQUIRED:

-Ends (1 or 2) Specifies whether input sequence files are single or paired end

-A (string) Authorization key generated by MG-RAST

-D (path) Path to folder containing raw sequence files to be processed

OPTIONAL:

-Q Enables 'quiet mode'; no output printed to the console.

-F (path) Path to FLASH app, if not installed in /Applications/FLASH-1.2.11/flash* .

-T (path) Path to Trimmomatic.jar, if not in /Applications/trimmomatic-0.33/trimmomatic.jar .

Example pre-annotation command:

$ python SAMSA_pre_annotation_pipeline.py -A MG-RAST-KEY -D ~/path/to/sequences/ -Ends 2 -F ~/path/to/FLASH/program* -T ~/path/to/trimmomatic.jar

Submitting files on MG-RAST for annotation

After the SAMSA_pre_annotation_pipeline.py wrapper has finished, the files are trimmed, aligned (if paired end), uploaded to MG-RAST's inbox, and MG-RAST has computed sequence stats for each file.

To submit these files, log in to MG-RAST and select "Upload", and then "Next" at the bottom of the page. Select all files to be submitted, assign a project name, and make sure to DESELECT "dereplication." After the files have been submitted, their progress can be viewed by selecting "Browse" at the top of the page, and then "In Progress" on the left side.

Downloading annotations from MG-RAST

The best approach for downloading annotation files from MG-RAST is to use the MG-RAST_API_downloader.py, part of the SAMSA tools. This python script has the following necessary command line arguments:

REQUIRED:

-S (string)	Annotation source: RefSeq, UniProt, KEGG, etc.
-D (string)	Data type: organism, function, or ontology
-A (string)	MG-RAST authorization key
-I (number)	Annotation ID number; found on the MG-RAST results page
-O (name)	Save name for this annotation file (will be created in current directory).

OPTIONAL:

-Q	Enables quiet mode.
-usage	Prints these usage options to the command line output and then exits.

These commands can also be seen by running:

$ python MG-RAST_API_downloader.py -usage

A sample MG-RAST_API_downloader.py command:

$ python MG-RAST_API_downloader.py -S RefSeq -D organism -I 459902.3 -A UyTjAgVxLPwwdGDj73xhJ9w9C -O output_metatranscriptome.tab

Because this download program can be run from a single line of code, it is possible to create a very simple bash script containing download program commands for each of the metatranscriptomes to be downloaded. This allows for a “queue” of downloads.

NOTE: Given the size of these annotation files, the download process can take several hours. It’s advisable to run this process in a separate shell using the screen command (https://www.gnu.org/software/screen/manual/screen.html), allowing for logging off while the download continues to run.

Once these annotation files are downloaded, proceed to Step 5, in later instructions, for analysis.

SAMSA Pipeline Use: Detailed pre-annotation steps

Steps 1-3 are performed automatically if using the SAMSA_pre_annotation_pipeline.py, as described above. However, they can each be performed manually based off of the steps described here.

Step 1: Preprocessing

Raw files must be processed using Trimmomatic to remove adaptor contamination and low-quality reads before they can be submitted to MG-RAST.

For details using Trimmomatic, refer to the manual: http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/TrimmomaticManual_V0.32.pdf

If using paired-end reads:

For each sample, two files should be present, often designated as R1 and R2. The R1 file contains all forward reads for that sample, while the R2 file contains all reverse reads.

FLASH (http://ccb.jhu.edu/software/FLASH/) is recommended for joining these two files together to create a single file of all combined reads. FLASH 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
```

FLASH is run from the command line, using the following command:

```
$ ./flash MATES_1.fastq MATES_2.fastq
```

To see FLASH options, use the following command:

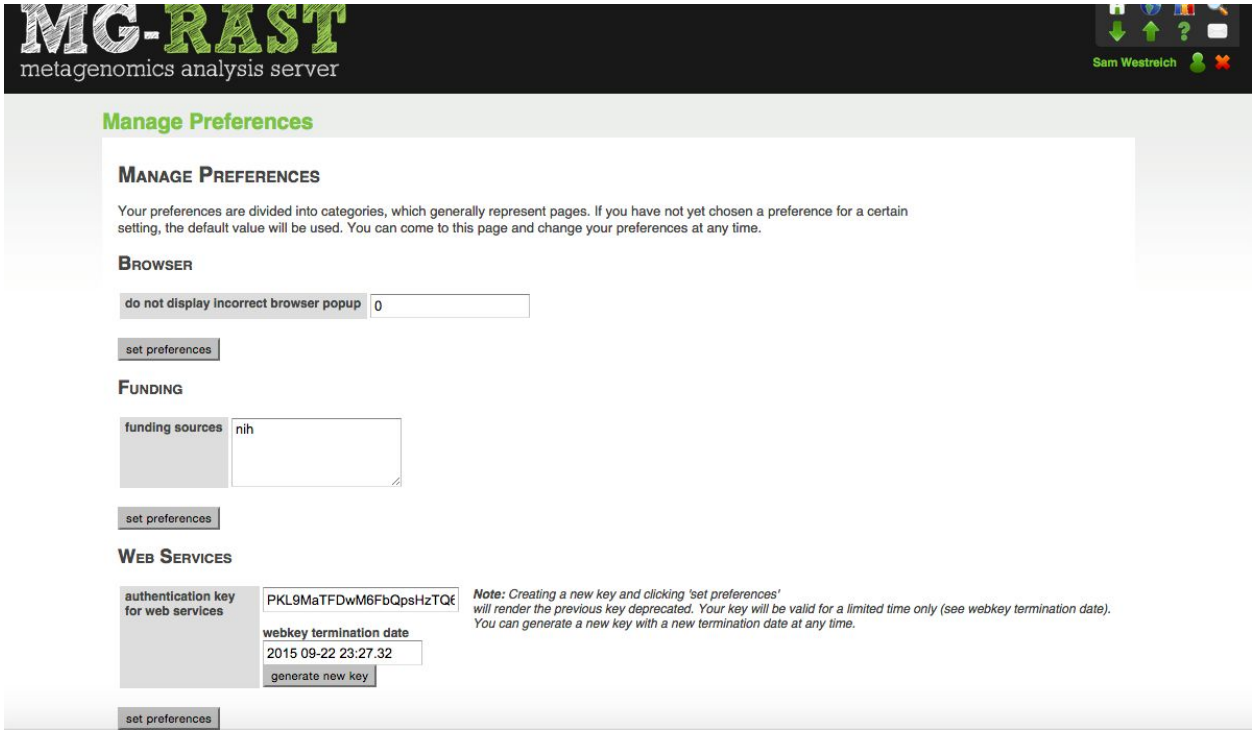
```
$ ./flash --help | less
```

Step 2: Uploading to MG-RAST

Once files have been cleaned (and joined if paired-end), the next step is to upload the file to MG-RAST for annotation.

If you do not have an MG-RAST, it is necessary to make an account at http://metagenomics.anl.gov/?page=Register. If you do have an account, you will need a current authorization key (generated within the last ten days). The authorization key can be found at http://metagenomics.anl.gov/?page=AccountManagement by selecting “Preferences”.

If you generate a new authorization key, be sure to save the key by selecting “Set Preferences” at the bottom of the page.



Although files can be uploaded to MG-RAST through the point-and-click interface, it is simpler and more rapid to use the API interface. To upload files through the MG-RAST API:

1. Navigate to the folder on the Unix system where the files are located.
2. Run upload_MG-RAST.py (part of the SAMSA tools). The following options must be specified:
    a. -A authorization_key
    b. -F file_to_be_uploaded
    c. For further usage options, run: upload_MG-RAST.py -usage
3. Files may be uploaded in zipped form (.gz) for faster transfer.

Example upload_MG-RAST.py command:

```
$ python upload_MG-RAST.py -A eZYmtTzx5yrYn8pDQx4kUFxC4 -F sequences.fastq
```

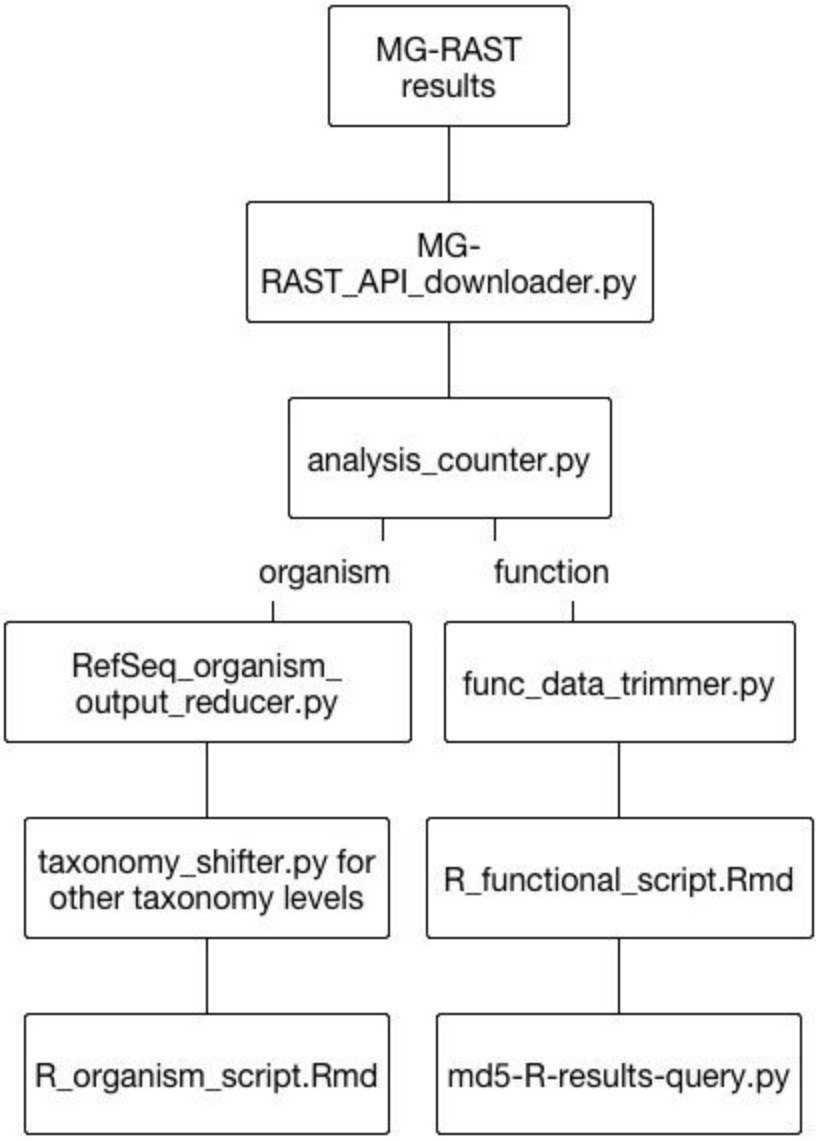
Step 3: Submitting to MG-RAST

Once all files are uploaded to MG-RAST’s inbox, access the inbox at http://metagenomics.anl.gov/Html/mgmainv3.html?mgpage=upload. MG-RAST must compute sequence stats on each file before it can be submitted. If files were uploaded in zipped format, they must be unzipped in the inbox.

To submit files for processing, click the “Next” button at the bottom right. When submitting a metatranscriptome, be sure to ensure that under “Pipeline Options”, “Dereplication” is DISABLED. *Warning: failing to disable dereplication will result in the removal of identical sequences and the loss of expression data!*

After submission, MG-RAST will process the files. This may take several days, depending upon the number of files submitted, the size of the files, and the number of other files currently waiting in the MG-RAST processing queue.

SAMSA Pipeline Use: Detailed post-annotation steps



Step 4: Downloading annotations from MG-RAST

Once MG-RAST has finished processing submitted files, the annotated results can be downloaded using MG-RAST_API_downloader.py (part of the SAMSA tools). This tool has the following options:

-Q	Quiet mode (no printing to STDOUT); optional
-S	Source (RefSeq, UniProt, KEGG, COG, Subsystems,GenBank, etc.); required
-D	Data type (Organism, Function, Ontology); required
-A	Authorization key, found under "Preferences" at metagenomics.anl.gov; required
-I	Annotation ID, found on metagenome page; required

-O Output save file name - should end with .tab; required
-usage Prints usage documentation and exits.

If required flags are not provided when running this tool, it will prompt for the information to be provided.

Sample MG-RAST_API_downloader.py command:

$ python MG-RAST_API_downloader.py -S RefSeq -D Organism -I 4628940.3 -A eZYmtTzx5yrYn8pDQx4kUFxC4 -O annotation_results.tab

Result: A downloaded annotation file, containing one line for each submitted, annotated sequence and its best match from the selected database.

Step 5: Analyzing annotations

The downloaded annotation file from MG-RAST contains a read-by-read annotation of the submitted metatranscriptome. The next step in the SAMSA pipeline is creating a summary file of this annotation, providing a sorted count of all organisms or transcripts by abundance. This summary file will be imported into R and used for analysis.

To create this summary file for ORGANISM results, use analysis_counter.py (part of the SAMSA tools). A sample usage of this script:

$ python analysis_counter.py annotation_file -o output_file_name

For pipeline use, -q will enable quiet mode.

For functional annotations, it is recommended that the MG-RAST internal identifier (M5nr) be preserved. To create this summary file for FUNCTION results, use analysis_counter.py with the "-m" flag, to preserve M5nr IDs. A sample usage of this script:

$ python analysis_counter.py annotation_file -o output_file_name -m

Result: An output file containing a sorted abundance list of all organisms (if using analysis_counter.py) or of all functions by M5nr ID (if using the "-m" flag). Summary data will be included in the header of the file.

Step 6: Preparing summary files for import into R

In each output file, the first few lines are summary statistics for the entire file. These lines need to be removed before the file can be imported into R.

To remove these lines, use either:

RefSeq_output_reducer.py for an organism file

func_data_trimmer.py for a functional transcript file

For these programs, the input file must be specified with the “-I” flag. An output file can be specified using the “-O” flag; if this is not included, all files will receive the “_simplified” appendix. For more information, run the script with “-usage” flag.

Result: For organism files, RefSeq_output_reducer.py will generate a summary file with overall data stripped from the first 6 lines, and with the species names reduced and consolidated down to Genus level. For functional transcript files, func_data_trimmer.py will generate a summary file with overall data stripped from the first 6 lines, and with excess unnecessary functional repetition removed.

Optional analysis: Examining different taxonomy levels

In normal use of this pipeline, organism files are annotated at the Genus level. However, it may be necessary to convert organism results to a higher taxonomic levels for analysis at the Family or Order level. The program “taxonomy_shifter.py” is capable of raising the taxonomic level of all entries in a results file (generated in Step 6, above) to a higher taxonomic level.

Also included is a reference database, Bacteria_Genus_flattened.tsv, which includes all taxonomic levels for all current bacterial genera. This reference must be specified using the “-R” flag when running the program.

This tool has the following flags needed for usage:

-Q Enables quiet mode
-F Input file, necessary
-R Reference index file, necessary
-T Final taxonomy level desired: Kingdom, Phylum, Class, Order, Family, Genus, necessary
-O Custom output file name (default is input_file.shifted)
-V Verbose mode, shows exceptions
-E Exclusion, will exclude all exceptions if present

A sample usage of taxonomy_shifter.py:

$ python taxonomy_shifter.py -F input_file -R location_of_reference_file -T Phylum -O phylum_converted_input_file

Results: A summary file in which all lower taxonomic levels have been raised to the specified taxonomic level and combined into a single result. For example, running the above sample command on an output file will result in a summary output file, as generated in Step 6, but with all entries at the Phylum level.

Step 7: Evaluating in R

Once the analysis_counter program has run, it will generate output files for each metatranscriptome. These files must be trimmed, using either RefSeq_output_reducer.py for organism results, or using func_data_trimmer.py for functional results. These files may be converted to a different taxonomic level using taxonomy_shifter.py, but are ready for input into R for analysis.

Depending on whether these output files were for organism or transcript reads, they can be analyzed using the R scripts, either R_organism_script.Rmd or R_functional_script.Rmd (both part of the SAMSA tools).

Both of these R scripts are in Markdown format, with additional comments to explain the analysis being performed by each code block.

Before loading the R analysis programs, ensure that downloaded samples have the following name, either:

- experimental_filename.output
- control_filename.output

R sorts the imported files into two groups based upon this prefix; if it isn't present, R will not be able to perform pairwise comparison analysis between the two conditional groups.

After files have been imported into R and cleanup steps have been performed (the R program will simplify header names based upon filenames, merge the files into a single table, sort, and simplify naming conventions), further code blocks will allow for the generation either of summary stacked bar graphs, or for the performing of differential analysis between experimental or control conditions.

Further instructions and detailed breakdown of each step in the R analysis script is included in the R markdown scripts.

Results: The R scripts will generate two stacked bar graphs, showing either the relative or absolute abundance of each different entry across all imported metatranscriptome files. In addition, the DESeq2 section of the R scripts (last code block in the files) will generate a list of all results sorted by adjusted p-value, in tab-delimited form.

Further analysis: Converting M5nr internal MG-RAST IDs to RefSeq IDs

To merge duplicate annotations across multiple databases, MG-RAST uses its own internal identifier system, referred to as M5nr (the "nr" standing for "non-redundant"). Each M5nr internal ID is linked to an external annotation in one or more reference databases used by MG-RAST.

In order to determine the external IDs from other databases that are linked to a specific MG-RAST M5nr ID, an API call can be used to request the relevant information from MG-RAST.

For metatranscriptome results, however, it may be necessary to retrieve external IDs for a large number of M5nr IDs at a time. To speed up and automate this process, the program "md5-R-results-query.py" can add external IDs to the results generated by the "R_functional_script_w-md5s.Rmd".

NOTE: Currently, the M5nr IDs for organisms are not preserved. In order to use this script, the input tab-delimited file must contain the M5nr ID in the right-most column.

Sample usage:

```
$ python md5-R-results-query.py -I input_file_name -O output_file_name
```

Due to the relatively slow speed of retrieving multiple API requests, this program may take several hours to run completely; it can be terminated partly through its run, however, if only the IDs at the beginning of the file are required.

Results: The output file will appear identical to the input file, saved in tab-delimited form, but will include one extra column on the right, containing the external RefSeq ID corresponding with the MG-RAST M5nr internal ID.