May 06, 2025

Bacterial Travelers

DOI
https://dx.doi.org/10.17504/protocols.io.kxygxq3k4v8j/v1

Brittany Ott[1]

[1]US FDA

Brittany Ott
US FDA

DOI: https://dx.doi.org/10.17504/protocols.io.kxygxq3k4v8j/v1

Protocol Citation: Brittany Ott 2025. Bacterial Travelers. **protocols.io** https://dx.doi.org/10.17504/protocols.io.kxygxq3k4v8j/v1

Protocol status: Working
We use this protocol and it's working

Created: May 02, 2025

Last Modified: May 06, 2025

Protocol Integer ID: 210832

Keywords: bacterial sequences within sample, bacterial sequence, transcriptomic data, bacterial traveler, using transcriptomic data, associated microbiome, organism of origin, organism, larger organism, helpful if sample

Abstract

Using transcriptomic data, we want to examine the bacterial sequences within samples and determine their organism of origin and even function. This can be helpful if samples are taken of a larger organism that may have an associated microbiome.

Procedures for Gambierdiscus Bacterial Travelers Code - Overview

1 The goal was to examine the bacterial travelers amongst transcriptomic data of dinoflagellates, specifically *Gambierdiscus*.

Instead of using assembled data, a very useful pipeline generated by **Sam Westreich** utilizes reads. This eliminates the potential for chimeric assemblies that can throw off results.

This is a modified version of the pipeline to suit my needs. You know the old saying: never write code that has already been written. Well, maybe that's not the old saying. But it works for me.

Software required for the pipeline

1.1
- **SAMSA2**
- **diamond**
- **sortmerna**
- **subsystems**
- **Trimmomatic**
- **BBMerge**
- **RStudio**
- **FastQC**
- Python

SAMSA2 Pipeline

1.2 In order to determine what bacteria can be found in our transcriptomes, we decided to utilize the **SAMSA2** pipeline, with a couple of modifications.

Ensure you have all dependencies, software, and databases required to run the pipeline

2
- **Python 2.7.x** (with the following packages)
  *sys
  *os
  *subprocess
  *glob
  *time
  *gzip
  *operator

*commands
- **RStudio** (with the updated version of R, and the following libraries)
*optparse
*ggplot2
*data.table
*plyr
*gridExtra
*scales
*reshape2
*knitr
*vegan
*dplyr

2.1 Download and install all software listed above according to the manual. However, I do have some notes here that might help with some of the installation protocols (Note: I did everything on a remote Linux server, not a local MacOS or Windows machine, so all instructions will be working for Linux):

- **SAMSA2:** this is a set of scripts (the original pipeline).

- **diamond:** this installation is simple, follow the instructions in the README portion of the github page. Make sure you put the `diamond` executable into your **PATH**.

- **sortmerna:** use the **Github release**, as building the code is a major pain, and follow the README instructions. Once you have it installed, you then need to generate the indices before you run the code. There are 6 databases that need to be indexed (small and large rRNA subunits for bacteria, archaea, and eukarya). To do this, make sure the `indexdb_rna` script is in your PATH. I also placed my `sortmerna` directory into my home directory. Then run the following (note: everything inside of `< >` is dependent on your particular files; when entering the command, do not use the `< >` symbols):

$ indexdb_rna --ref ~/sortmerna-<version>/rRNA_databases/silva-arc-16S-id95.fasta,/home/<username>/sortmerna-<version>/index/silva-arc-16S-id95 -v

---

- refseq database: This will obtain a usable version of the refseq database for `diamond` searches. First, obtain the database:

$ wget ftp://ftp.ncbi.nlm.nih.gov/refseq/release/complete/.faa.gz*

Note: this database is HUGE, so make sure you have the available space. You will then need to pull all of the files together:

*$ cat *.faa.gz > refseq.faa.gz*

Then, you will need to prepare the database for `diamond`:

$ nohup nice diamond makedb --in refseq.faa.gz --db refseq &

- subsystems: This is to generate the SEED subsystems database, which will be used when doing `diamond` searches. First, you will need to grab the necessary files from the SEED ftp site using:

$ wget ftp://ftp.theseed.org/subsystems/subsys*

Then, unzip the `.gz` files using: $ gunzip <filename>.gz

Next, run the **subsys_db_rebuilder.py** script:

$ python2.7 subsys_db_rebuilder.py subsystems.complex subsystems2role subsystems2pg subsys.txt

This will produce a `subsystems.complex.merged` file. Note: this file will be used further down the line, so keep it somewhere you will remember. Next, run the following command:

$ sed 's/\t/ /g' subsystems.complex.merged > notabs.subsystems.complex.merged

Finally, run the following, which will generate a `reduced` file:

$ python2.7 duplicate_counter.py notabs.subsystems.complex.merged

Now, you will generate the `diamond` compatible database:

$ nohup nice diamond makedb --in notabs.subsystems.complex.merged.reduced --db seed_subsystems

---

- Trimmomatic: go to the **Trimmomatic** page, and under the `Downloads` section, choose the `binary` link. Make sure you have an updated version of `Java`.

- BBmerge: download the **BBtools package**, which has the bbmerge.sh script inside. Place this into your PATH.

- Rstudio: follow instructions on the downloads page.

- FastQC: follow instructions on the download page.

Trim the reads using Trimmomatic

3 Not all of your data needs to be trimmed. You can check if it needs to be trimmed or not using `FastQC`. For those that were of poor quality, here is what I used to trim:

```
# PE: indicates paired end reads; this is followed by your raw
data files, forward and reverse, respectively.
# Then the next four names are the new file names for your
forward-paired reads, your forward-unpaired reads, your
reverse-paired reads, and your reverse-unpaired reads (not all
reads will be able to be paired during trimming) --> you are
giving 4 new file names
# HEADCROP:8 will cut off the first 8 bp
# TRAILING:15 will cut off the end of the read below the
threshold
# ILLUMINACLIP:NexteraPE_BaltAdaptSeq.fa indicates a fasta file
containing adapter sequences used by your sequencing system; 2
indicates 2 mismatches allowed for seed matches of 16 bases; 30
indicates that if the paired end reaches a score of 30, seeds
will be extended and clipped (about 50 bases); 10 is the same
as 30, but for single end reads

$ java -jar trimmomatic-0.36.jar PE <forward reads>_R1.fastq.gz
<reverse reads>_R2.fastq.gz <sample>_forward_paired.fq.gz
<sample>_forward_unpaired.fq.gz <sample>_reverse_paired.fq.gz
<sample>_reverse_unpaired.fq.gz HEADCROP:8 TRAILING:15
ILLUMINACLIP:NexteraPE_BaltAdaptSeq.fa:2:30:10 MINLEN:36
```

Merge the reads using BBMerge

4 Once the reads are trimmed (if they need to be trimmed), the next step is to merge them back together. To do this, I used the `BBMerge` script from the `BBTools` package:

```
# in1=forward reads; this will either be from your trimmed
results, or your raw files
# in2=reverse reads; this will either be from your trimmed
results, or your raw files
# out=the merged reads file (new file) --> you are giving a new
file name
# outu1=the unmerged forward reads file (new file) --> you are
giving a new file name
# outu2=the unmerged reverse reads file (new file) --> you are
giving a new file name

$ bbmerge.sh in1=<sample>_forward_paired.fq.gz in2=
<sample>_reverse_paired.fq.gz out=<sample>_merged.fq.gz outu1=
<sample>_unmerged_forward.fq.gz outu2=
<sample>_unmerged_reverse.fq.gz
```

4.1 Now, BBMerge has difficulty pairing smaller reads, and this eliminates much of the data. While longer reads are preferable, we can still use the shorter reads as well. Using only one side of the paired end reads, concatenate your merged with either the forward or reverse of your unmerged reads:

$ cat <sample>_merged.fq.gz <sample>_unmerged_forward.fq.gz > <sample>_mergeunmerge.fq.gz

Using sortmerna to separate ribosomal sequences from "ribodepleted" sequences

5 For this step, you will use the <sample>_mergeunmerge.fq.gz from the previous step, as well as the '=`SAMSA/sortmerna-2.lb` directory (make sure you unzip the tarball first!)

5.1 sortmerna does not like gzipped files, so unzip:

$ gunzip -k <sample>_mergeunmerge.fq.gz

5.2 Then run the following code:

```
# note: change the PATH names as appropriate
# --ref is the reference databases (there are six, each with the
fasta file and the generated database, which should already be
available in the tarballed sortmerna directory
# --reads is your merged+unmerged fastq file
# --aligned tells sortmerna what to call the file containing the
ribosomal sequences
# --other tells sortmerna what to call the file containing the
ribodepleted sequences
# the remainder of the options were those specifically called to
work with scripts down the line, so they are not altered

$ sortmerna --ref <PATH TO DATABASE>/<dataset1>.fasta,<PATH TO
DATABASE>/index/<dataset1>:<PATH TO DATABASE>/<dataset2>.fasta,
<PATH TO DATABASE>/index/<dataset2>:<PATH TO
DATABASE>/<dataset3>.fasta,<PATH TO DATABASE>/index/<dataset3>:
<PATH TO DATABASE>/<dataset4>.fasta,<PATH TO
DATABASE>/index/<dataset4>:<PATH TO DATABASE>/<dataset5>.fasta,
<PATH TO DATABASE>/index/<dataset5>:<PATH TO
DATABASE>/<dataset6>.fasta,<PATH TO DATABASE>/index/<dataset6> --
reads <PATH TO READS FILES>/<sample>_mergeunmerge.fq --aligned
<PATH YOU WANT TO SEND FILES TO>/<sample>.ribosomes --other <PATH
YOU WANT TO SEND FILES TO>/<sample>_ribodepleted --fastx --
num_alignments 0 --log -v
```

5.3 House keeping step: remove the gunzipped `.fq` file, which takes up a lot of space and is no longer necessary:

$ rm <sample>_mergeunmerge.fq

Running DIAMOND blastx using the refseq database

6 This step will annotate each transcript for its function and the source (what organism it comes from). First, you must download diamond OR you can use the executable placed into the `SAMSA` directory.

6.1 Generate the indices for your database:

$ diamond makedb --in refseq.faa.gz --db refseq

OR use the already indexed refseq database in the `SAMSA/database` directory. But if you want to use the newest version of refseq, do the following to get the amino acid

database and re-index the new refseq.faa.gz, using the code above.

```
# get all of the `.faa.gz` files
$ wget ftp://ftp.ncbi.nlm.nih.gov/refseq/release/complete/*.faa.gz

# concatenate all of the `.faa.gz` files into one big file
$ cat *.faa.gz > refseq.faa.gz
```

6.2 Once you have the diamond executable (in your ~/bin folder which should be in your PATH, so it can be called from anywhere), and the database downloaded and indexed, you can now run `blastx` using the following command:
RIBODEPLETED example (run for both ribodepleted and ribsomal)

```
# running blastx using the refseq database
$ diamond blastx --db <PATH TO REFSEQ>/refseq -q <PATH TO
RIBODEPLETED SAMPLE>/<sample>_ribodepleted.fq -a
<sample>_ribodepleted_refseq -t ./ -k 1

# reformatting your outfile into a tab format
$ diamond view --daa <sample>_ribodepleted_refseq -o
<sample>_ribodepleted_refseq_final.out -f tab
```

Aggregation

7 Once you have run diamond blastx (first command in section 6.2) and reformatted the outfile (second command in section 6.2), use the `DIAMOND_analysis_counter.py` SAMSA script (found in the `SAMSA` directory). Also ensure that you have gunzipped the refseq.faa.gz file, as it will be looking for the unzipped version. You will do this twice, once for the organism, and once for the function:

```
# running the DIAMOND_analysis_counter.py script to obtain a list
of organisms (.tsv out)
$ nohup nice python <PATH TO FILE>/DIAMOND_analysis_counter.py -I
<sample>_ribodepleted_refseq_final.out -D <PATH TO
DATABASE>/refseq.faa -O &

# running the DIAMOND_analysis_counter.py script to obtain a list
of functions (.tsv out)
$ nohup nice python <PATH TO FILE>/DIAMOND_analysis_counter.py -I
<sample>_ribosomal_refseq_final.out -D <PATH TO
DATABASE>/refseq.faa -F &
```

R scripting

8 Once you are done with the aggregation of your diamond blastx files, which will produce `.tsv` files, you will take these `.tsv` files and move forward with the first set of R scripts: organism.

8.1 For the Organism: The things that you need to make sure you have done for the Rscript to work:

1. all control samples are structured: control_<samplename>_..._organism.tsv
2. all environmental samples (samples of interest and comparison) ate structured: experimental_<samplename>_..._organism.tsv
3. The R_for_SAMSA.rmd is in the folder containing all of your `*_organism.tsv` files

Do the same thing for Function

8.2 **Organism:** R_for_SAMSA_normalization.Rmd
R_for_SAMSA_normalization.nb.html

8.3 **Function:** R_for_SAMSA_normalization.nb.html
R_for_SAMSA_normalization.Rmd

8.4 **Ribosomal:** R_for_SAMSA.Rmd R_for_SAMSA.nb.html
R_for_SAMSA_normalization.nb.html

8.5 **Hierarchy:** SAMSA_R_COMBINED.nb.html

SAMSA_SEED_R_COMBINED.nb.html SAMSA_SEED_R_COMBINED.Rmd