

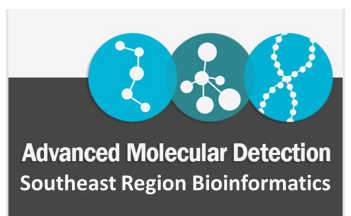
Florida Department of Health, Bureau of  
Public Health Laboratories  
Jacksonville, FL

# Generation of Genomic Epidemiology Reports **for External Use**

---

## Standard Operating Procedure

This document is the standard operating procedure for generating genomic epidemiology reports.

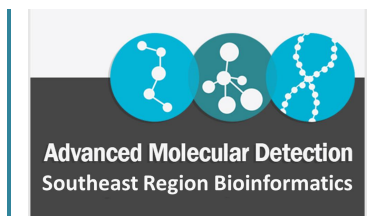


## Signature Page

Prepared by:	Effective	Supersedes Procedure #
Jianye Ge, PhD, Ibrahim Farag, PhD, Sam Bernhoft, MPH, and Molly Mitchell, PhD		

Revisions:	Date	Signature

Annual Review:	Date	Signature



## 1. Purpose:

This document establishes the standard operating procedure for generating comprehensive, genomic epidemiology reports. The report's primary purpose is to transform raw genomic data into actionable insights that inform public health decision-making. The SOP provides detailed guidelines for data collection, processing, analysis, interpretation, and dissemination of results, ensuring that each report is consistent, accurate, and reproducible.

In this SOP, *Mycobacterium Tuberculosis* (MTB) is used as an example.

## 2. Scope:

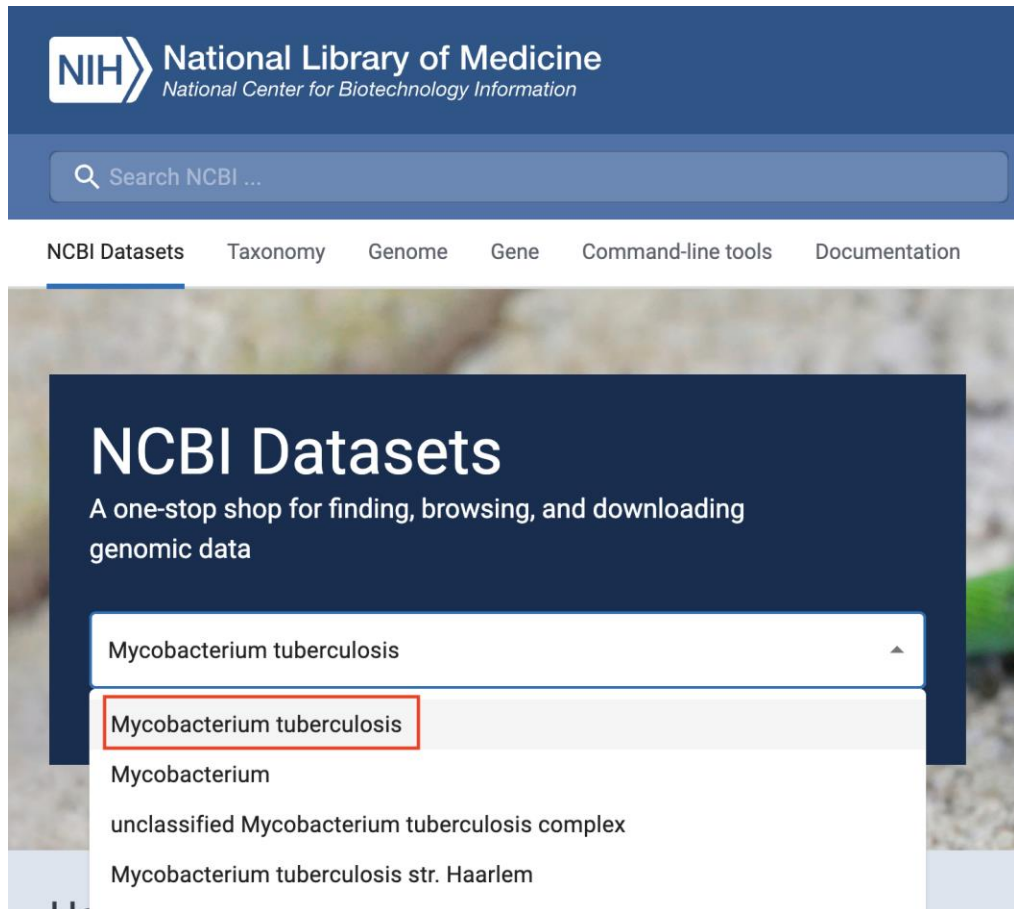
The Bioinformatics Team is responsible for generating the genomic epidemiology reports.

## 3. Disclaimer:

<Add in a disclaimer to protect the results included in the genomic epidemiology report if your pipeline and system is not yet validated. If your pipeline is not validated, the results may be used for surveillance and research purposes only. They are not to be used for diagnosis, treatment, or management of patient care.>

### I. Data Preparation:

- a. Sequence data
  - i. Add data to HiPerGator in the desired folder of the run(s) you'd like to compare.
- b. Reference Genomes
  - i. If reference genomes are needed, they can be downloaded from NCBI.
  - ii. Example of finding a reference genome for MTB
    1. Navigate to <https://www.ncbi.nlm.nih.gov/datasets/> in your preferred browser.
    2. Type "*Mycobacterium tuberculosis*" in the search box.
      - a. The species name will pop-up in a drop-down menu.
      - b. Click the species name.



NIH National Library of Medicine  
National Center for Biotechnology Information

Search NCBI ...

NCBI Datasets Taxonomy Genome Gene Command-line tools Documentation

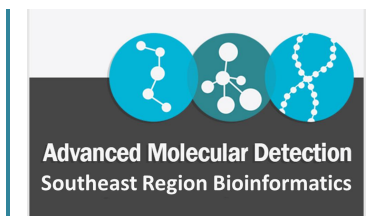
## NCBI Datasets

A one-stop shop for finding, browsing, and downloading genomic data

Mycobacterium tuberculosis

- Mycobacterium tuberculosis
- Mycobacterium
- unclassified Mycobacterium tuberculosis complex
- Mycobacterium tuberculosis str. Haarlem

3. Scroll down to the RefSeq genome section.
  - a. Click the blue “Download” button.



## Mycobacterium tuberculosis ☆

*Mycobacterium tuberculosis* is a species of high G+C Gram-positive bacteria in the family Mycobacteriaceae.

NCBI Taxonomy ID	1773
Taxonomic rank	species
Current scientific name	<b>Mycobacterium tuberculosis</b> (Zopf 1883) Lehmann and Neumann 1896 (Approved Lists 1980) <a href="#">Type Material</a>
Basionym	"Bacterium tuberculosis" Zopf 1883

[View taxonomic details](#)



Browse taxonomy

### Genome

[Browse all 8,217 genomes](#)

#### RefSeq genome

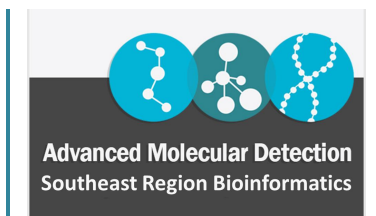
[ASM19595v2](#)

Sanger Institute (2013). Strain: H37Rv.

RefSeq GCF\_000195955.2

[Download](#)

4. A pop-up a window will appear.
  - a. Select the radio button for "RefSeq only" under "Select file source"
  - b. Select the check box next to "Genome Sequences (FASTA)"
  - c. Click the blue "Download" button to download a zip file containing the reference genome.



### Download Package

1 genome selected for download

#### Select file source

- ☐ All
- ☒ RefSeq only <sup>1</sup>
- ☐ GenBank only

#### Select file types

- ☒ Genome sequences (FASTA) <sup>2</sup>
- ☐ Annotation features (GTF)
- ☐ Annotation features (GFF)
- ☐ Sequence and annotation (GBFF)
- ☐ Transcripts (FASTA)
- ☐ Genomic coding sequences (FASTA)
- ☐ Protein (FASTA)
- ☐ Sequence report (JSONL)
- ☒ Assembly data report (JSONL)

Your selected data will be downloaded as a ZIP archive  
Estimated file size is 2 MB

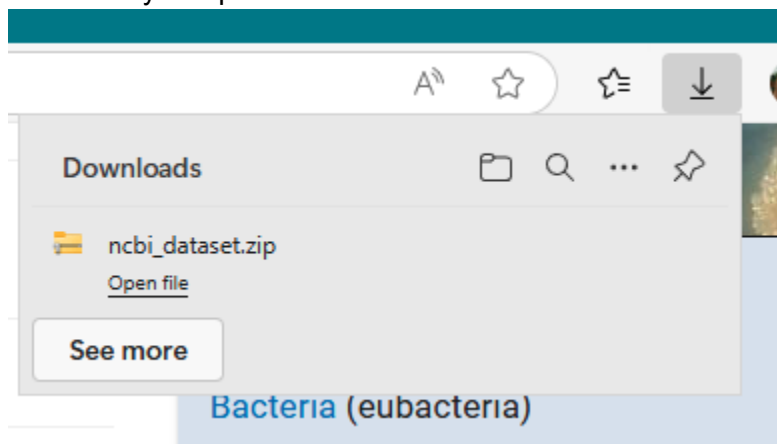
Name your file

ncbi\_dataset.zip

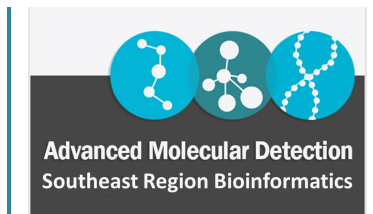
Cancel

Download <sup>3</sup>

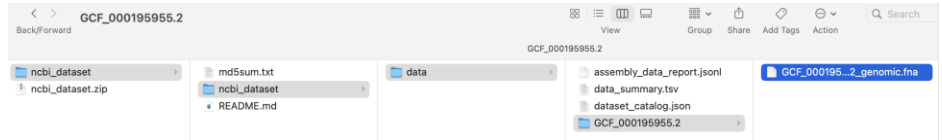
5. Unzip the reference genome file
  - a. Open the zipped file from your downloads by double clicking on it. It will automatically unzip.



i.



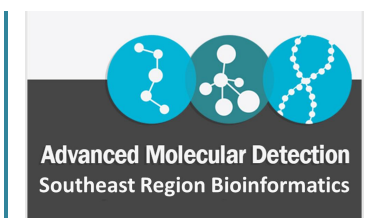
6. Click through the nested files (see below) to find the .fna (ref genome) file in the unzipped folder.



a.

c. Metadata

- i. Metadata for each sample is needed, as appropriate, for your report.
- ii. Metadata could include, the sample ID, isolation source, location source (county, hospital, etc.), date of collection, or anything else that may be relevant to the study.



## II. Example codes

- a. All examples code and data can be found at [BPHL-Molecular/ReportGen: FDOH's files for Outbreak report generation](#)
- b. "Report Generation SOP - External.docx" is this Standard Operating Procedure.
- c. <https://github.com/BPHL-Molecular/ReportGen/blob/main/environment.yml> is used to create a new conda environment.
- d. <https://github.com/BPHL-Molecular/ReportGen/blob/main/DataPreprocessing.ipynb> includes the example code used to preprocess data to create files for report generation.
- e. <https://github.com/BPHL-Molecular/ReportGen/blob/main/ReportGeneration.ipynb> includes the example code used to a generate report.
- f. <https://github.com/BPHL-Molecular/ReportGen/blob/main/states-fips.json> is used when plotting geomap with county information

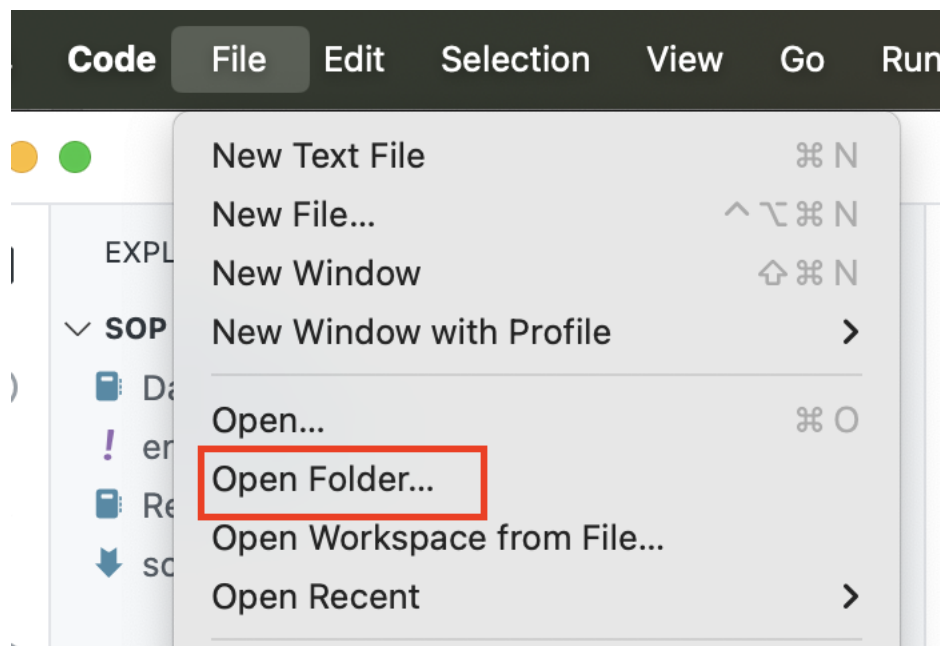
## III. Conda environment setup

- a. Create a conda environment on HiperGator (HPG).
  - a. An <https://github.com/BPHL-Molecular/ReportGen/blob/main/environment.yml> has been provided in the "Example Codes" folder to include all required python libraries to process the datasets and generate the report.
- b. First, in the terminal, type "**module load conda**" to load conda module in HPG
- c. Second, type "**conda env create --name myenv -f environment.yml**" to create a new conda environment
  - a. NOTE: Replace "myenv" with the name of the project or the task.
- d. In the following data preprocessing and report generation, please make sure you activate the conda environment first before executing your codes.
  - a. Activate the environment by "**conda activate myenv**".

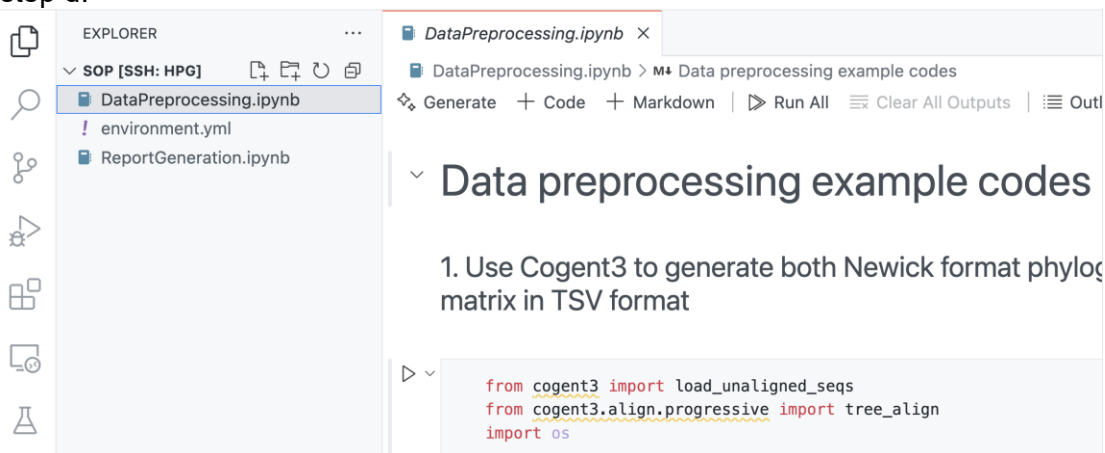
## IV. Jupyter Notebook with VS Code

- a. If you do not have VS Code installed on your computer, follow your IT requirements for requesting or installing new applications.
- b. Open VS code and follow the instructions provided by HPG remote development, [https://help.rc.ufl.edu/doc/VS\\_Code\\_Remote\\_Development](https://help.rc.ufl.edu/doc/VS_Code_Remote_Development), to log into HPG.
- c. After logging into HPG, click File --> Open Folder.
  - i. A pop-up a window will ask you the path of the folder.



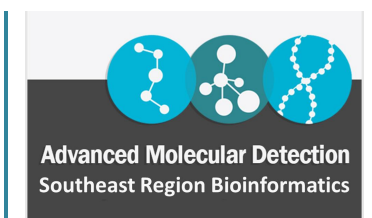


- d. Enter “/blue/bphl-state/user/SOP” in the path field
  - i. Click “OK”. This will open the folder and the example Jupyter Notebook.
- e. Copy these SOP files to your own folder
  - i. Copy the file by right-clicking on the file and selecting “copy”
  - ii. Navigate into the folder you wish to save the SOP in. Right-click the folder and select “paste.”
  - iii. Once you do this, change the path to the location of the SOP as mentioned in step d.



- iv.
- f. Note: more details on running and managing Jupyter Notebooks can be found at [https://help.rc.ufl.edu/doc/Jupyter\\_Notebooks](https://help.rc.ufl.edu/doc/Jupyter_Notebooks).
- g. Note: more details on managing Python environments and Jupyter kernels can be found at [https://help.rc.ufl.edu/doc/Managing\\_Python\\_environments\\_and\\_Jupyter\\_kernels](https://help.rc.ufl.edu/doc/Managing_Python_environments_and_Jupyter_kernels).

## V. Data pre-processing:

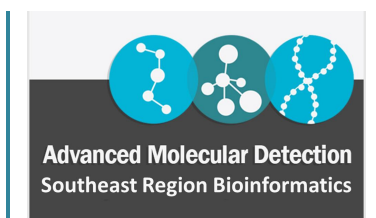


1. Collect all sequence data in .fastq format (when zippered is .fastq.gz) of a target species and copy them to a folder, usually folder “fastqs” under pipeline folder “Sanibel,” as explained below.
  - a. Please reference section I.a (page 3). on where to obtain the sequence data.
    - i. Many samples may have incorrect, inconsistent, or unconventional names with the samples in metadata.
      1. The isolate naming convention needs to match the name convention required by the pipeline to be used for data analysis.
2. Run appropriate pipeline(s) developed by either the FL Bioinformatics team, CDC, other state BRR, or your own!
  - a. **Sanibel – as an example:**
    - i. Under the folder “Sanibel”
      1. The folder “fastqs” contains the input fastq files
      2. The folder “output-RUNNUMBER” contains the output results.
        - a. Each input sample has one corresponding folder in the output folder. The output folder name is samplename.gff. The folder is under Sanibel > output-RUNNUMBER > sample name > sample name\_assembly > prokka > samplename.gff
        - b. “The AMR gene data is located in the file samplename\_amrfinderplus\_report.tsv. The file is located output-run name > sample name > sample name\_assembly > samplename\_amrfinderplus\_report.tsv.
    - ii. Complete a quality-check of the Sanibel output
      1. Under the Sanibel output folder for the run, there is a file titled “sum\_report.txt,” which contains the summary statistics of all samples. Only the samples that have a number of contigs (num\_contigs) <200 and read coverage (est\_coverage) ≥40x are considered to pass the quality check and will be used in the following analysis and report generation.
        - a. NOTE: Samples that fail QC will be repeated so they can be ignored for now.
        - b. This is FDOH’s QC standards. If you’re state has different standards, use those.

The following is an example of sum\_report.txt, in which only a few columns are shown for demonstration purpose and most columns are hidden. “Est\_coverage” is the read coverage and “num\_contigs” is the number of contigs.

**b. fl\_cg\_snp or Talbot**

- i. FL BPHL's core genome SNP pipeline for the identification of clusters of closely related bacterial isolates to support public health surveillance and outbreak investigations.
- ii. The output of fl\_cg\_snp should look like the following figure
  1. The output files, “pairwise\_matrix.tsv” and “SNPs\_boot.treefile,” will be used in generating the report.
- iii. After finishing the run, merge all consensus fasta files that pass quality control into one file.



1. Type: **cat \*.fa > merged.consensus.fa** and hit enter

### c. External tools

- i. **Cogent3** (version 2023.9.22a1), <https://github.com/cogent3/cogent3>. Cogent3 is a Python library for analysis of genomic sequence data. Cogent3 is used to align the consensus fasta files in Dengue analysis, and it can generate both a Newick-format phylogenetic tree file (.nwk) and a pairwise distance matrix in TSV format (.tsv) from consensus files.

1. The first time you use Cogent3, install Cogent3 to a conda environment of your choosing on HPG.

- a. To create a conda at a specific directory, you need to specify the path of the directory and give the directory a name. For example: **conda create -yp /blue/bphl-state/<user>/repos/bphl-molecular/cogent3** will create a directory named “bphl-molecular” in the repos folder. If you did not already have the repos directory, the -p option will create it.

- b. Type:

- i. **conda activate /blue/bphl-state/<user>/repos/bphl-molecular/cogent3**

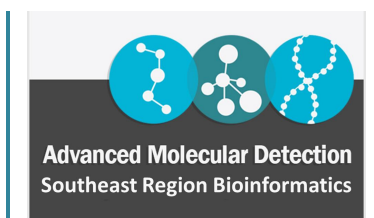
- ii. *conda install -c pip install cogent3=2023.9.22a1*

- c. If you have a conda environment as suggested in section II, you may activate it by typing **conda activate environment\_name**, and then install Cogent3 by **pip install cogent3=2023.9.22a1**

2. Run the codes in the section 1 in <https://github.com/BPHL-Molecular/ReportGen/blob/main/DataPreprocessing.ipynb> (also see the next figure) to the file to generate .nwk and .tsv format output.

- a. NOTE: Please update the file names in the codes accordingly before running the codes.

- i. The file names (such as samples.tn93.tree) in the codes are just examples. TN93 is chosen as a model for analyzing nucleotide sequences. It assumes base substitutions occur at different rates.



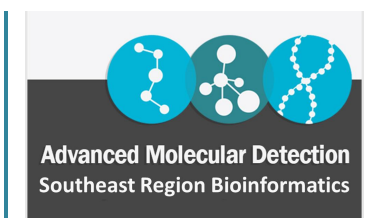
```
from cogent3 import load_unaligned_seqs
from cogent3.align.progressive import tree_align
import os

consensus_fasta_file = "merged.consensus.fa" # Replace with your file path
tree_newick_file = "samples.tn93.tree" # Replace with your file path
dists_csv_file = "samples.dist.tsv" # Replace with your file path

seqs = load_unaligned_seqs([consensus_fasta_file, moltype="dna"])
aln, tree = tree_align("TN93", seqs, show_progress=False)
tree.write(tree_newick_file)

dists = aln.distance_matrix(calc="tn93", show_progress=False)
dists_df = dists.to_table().to_dataframe()
dists_df.to_csv(dists_csv_file)
```

3. The code will output two files
  - a. Newick tree file (such as samples.tn93.tree)
  - b. Pairwise distance matrix file (such as samples.dist.csv)
- ii. MASH (<https://github.com/marbl/Mash>) is an alignment-free bioinformatics tool that rapidly estimates genomic and metagenomic distances by compressing large sequences into small, representative MinHash sketches. It is useful when the pairwise distances between the samples from multiple species needs to be calculated. MASH is typically used when multiple species are included in one report. The phylogenetic tree can be generated based on the pairwise distances.
  1. The first time you use MASH, install MASH to a conda environment of your choosing on HPG.
    - a. To create a conda at a specific directory, you need to specify the path of the directory and give the directory a name. For example: **conda create -yp /blue/bphl-state/<user>/repos/bphl-molecular/mash** will create a directory named “bphl-molecular” in the repos folder. If you did not already have the repos directory, the -p option will create it.
    - b. Type:
      - i. **conda activate /blue/bphl-state/<user>/repos/bphl-molecular/mash**
      - ii. **conda install -c bioconda mash**
  2. Run the following command to convert the FASTA file to mash files.
    - a. **mash sketch -o combined.msh \*.fasta**
  3. Run the following command to generate the pairwise distances.
    - a. **mash dist combined.msh combined.msh > distances.txt**
  4. Run the codes in <https://github.com/BPHL-Molecular/ReportGen/blob/main/DataPreprocessing.ipynb> to the file (also showing below). Please update the file names in the codes accordingly before running the codes.



5. Convert the MASH distances to a pairwise distance matrix. Then generate phylogenetic trees. Both can be achieved using the code below.

```
import pandas as pd
import os
import numpy as np

# MASH distance --> pairwise distance matrix in tsv
distances = pd.read_csv("mash_distances.txt", sep="\t") # Replace with your file path
distances.columns = ["sample1", "sample2", "distance", 'p-value', 'Matching Hashes / Total']
# remove ".fasta" from the sample names in first and second columns
distances["sample1"] = distances["sample1"].str.replace(".fasta", "")
distances["sample2"] = distances["sample2"].str.replace(".fasta", "")
df = distances.pivot(index="sample1", columns="sample2", values="distance")
df.to_csv("mash_distances.tsv", sep="\t")

# pairwise distance matrix in tsv --> phylogenetic tree in Newick
from Bio import Phylo
from Bio.Phylo.TreeConstruction import DistanceTreeConstructor, DistanceMatrix
import pandas as pd

# Load the pairwise distance matrix from a TSV file
def load_distance_matrix(file_path):
    df = pd.read_csv(file_path, sep='\t', index_col=0)
    names = list(df.columns)
    lower_triangle_matrix = [[0]] + [list(df.iloc[i, :i+1]) for i in range(1, len(names))]
    return DistanceMatrix(names, lower_triangle_matrix)

# Construct a phylogenetic tree using the UPGMA algorithm
def construct_tree(distance_matrix):
    constructor = DistanceTreeConstructor()
    tree = constructor.upgma(distance_matrix)
    return tree

# Write the tree to a Newick file
def write_tree_to_newick(tree, output_file):
    Phylo.write(tree, output_file, 'newick')

input_file = 'mash_distances.tsv' # Replace with your file path
output_file = 'mash_tree.newick' # Replace with your file path

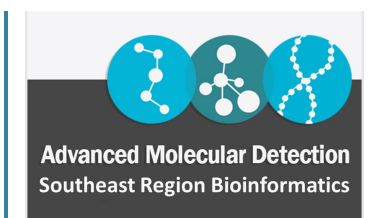
distance_matrix = load_distance_matrix(input_file)
tree = construct_tree(distance_matrix)
write_tree_to_newick(tree, output_file)
```

Python

- iii. **Varpipes-WGS**, <https://github.com/CDCgov/NCHHSTP-DTBE-Varpipes-WGS>. Varpipes-WGS is a whole-genome sequencing analysis pipeline developed by the CDC's Division of Tuberculosis Elimination that automates raw data quality control, read alignment, variant calling and strain typing to enable standardized, high-throughput genomic characterization of *Mycobacterium tuberculosis*. The docker image of Varpipes-WGS is available at HPG, /apps/varpipes-wgs/20230222/bin/varpipes\_wgs\_without\_refs\_latest.sif

The following are the program to run Varpipes-WGS

1. Create a new folder, such as "varpipes", and then copy all fastq.gz files of the samples to this folder. Make sure the fastq.gz file names following the



illumina file name convention. Namely, it has to contain “\_L001\_R1\_001” or “\_L001\_R2\_001” in the file names.

2. Download file <https://github.com/BPHL-Molecular/ReportGen/blob/main/examples/runvarpipe.sh> to the same folder you just created.
3. In a terminal, get into this folder. For example, you may type “**cd varpipe**” to get into the folder that contains all fastq.gz files.
4. Run the following command to activate the container

**singularity shell --bind ./varpipe\_wgs/data /apps/varpipe-wgs/20230222/bin/varpipe\_wgs\_without\_refs\_latest.sif**

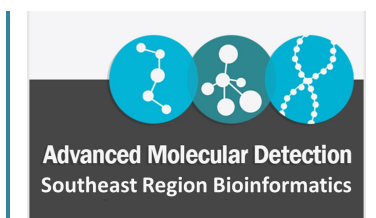
5. Once you get into the container, type “**cd /varpipe\_wgs/data**”
6. Then, type “**./runVarpipeline\_shortfilename.sh 3**” to run the program, in which “3” is the number of threads. You may change it to any number as you wish, but please keep it relatively low to not overwhelm the server.

Once the program is complete, there will be one folder created for each sample. You should see the analysis results in each folder. The following is one example, in which file ended with “\_DR\_loci\_Final\_annotation.txt”, “\_summary.txt”, and “\_Lineage.txt” are the files that containing results that may be used in generating report.

### 3. Report generation:

- a. Create a metadata table.
  - i. In <https://github.com/BPHL-Molecular/ReportGen/blob/main/ReportGeneration.ipynb>, Section 1 gives the example codes on how to show a table in Jupyter Notebook.
    1. Customize the code to fit the report needs by changing the column names if you have different column names in your metadata table.
    2. In the example codes, it uses “**meta.csv**” as input file name, but please update the file name if you have a different file name as input.
    3. You can also customize the fill color, text color, font size, text alignment, column widths, table width, and table height, if necessary.
- b. Plot a phylogenetic tree.
  - i. In <https://github.com/BPHL-Molecular/ReportGen/blob/main/ReportGeneration.ipynb>, Section 2 gives the example codes on how to plot a phylogenetic tree. The input data include three files:
    1. A metadata file (e.g., meta.csv), which is the metadata table and should include a column “Sample ID”
      - a. NOTE: Make sure you have a column “Sample ID” in your metadata. Otherwise, you may change “Sample ID” in the example codes to your sample id column name.
    2. A pairwise distance file (e.g., distances.tsv), which is a Tab-Separated Values (TSV) file and contains all the samples to plot





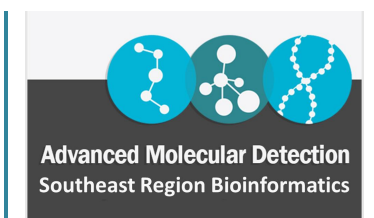
3. A tree file (e.g., tree.newick), which contains a tree in newick format.
  4. NOTE: The sample IDs in these files have to be the same. All these files should have been generated through data processing.
  5. NOTE: The metadata file and the tree files are essential to run the code. The distance file is only necessary if you need to add annotation lines.
- ii. In the majority of cases, you don't need to change the details in the functions as defined in this section.
  - iii. If you need more than 25 distinct colors, you may add more colors in the function `generate_distinct_colors()`.
  - iv. For the hover information codes in the `plot_tree()` function, you may change, remove, or add the column names to fit your metadata.
  - v. For the annotation codes in the `plot_tree()` function, you may change the annotation positions on x-axis and y-axis, if necessary, particularly when the sample size is getting big, such as more than 500 samples.
  - vi. To use the `draw_annotation_line()` function, you may update the sample ids in the example codes. The following example includes two samples ("ID1" and ID2"). You may update the sample names to draw annotation lines between the samples..
  - vii. Code example
    1. `draw_annotation_line(fig, "ID1", "ID2", "black", dists_df)`

#### c. Plot sub-trees.

- i. In <https://github.com/BPHL-Molecular/ReportGen/blob/main/ReportGeneration.ipynb>, Section 3 gives the example codes on how to plot sub-trees according to a column in metadata. The general idea is to separate the samples by a selected column, such as "speciesID\_mash", and plot the samples that share the same "speciesID\_mash" value into the same sub-tree. Thus, one additional input data is the column name (e.g., "speciesID\_mash") to separate the tree.

#### d. 3D plot of the samples.

- i. In <https://github.com/BPHL-Molecular/ReportGen/blob/main/ReportGeneration.ipynb>, Section 4 gives the example codes on how to plot samples in 3D space based on the pairwise distances. Either Multidimensional Scaling (MDS) or Uniform Manifold Approximation and Projection (UMAP) may be used for generating the coordinates.
  1. MDS tried to preserve distances between points
  2. UMAP tries to preserve the overall shape or structure of the data.



3. NOTE: There is no substantial difference from the end user point of view. Since both methods would rely on the provided random seed, namely, the 3D plot would change based on the random seed. It is recommended to try both methods with different seeds and see which plot visually makes better sense.
- ii. The input data of 3D plot includes metadata file, distance file, and a column name that to color the samples, but the tree file is not required.

**e. Show Antimicrobial Resistance (AMR) Genes.**

- i. In <https://github.com/BPHL-Molecular/ReportGen/blob/main/ReportGeneration.ipynb>, Section 5 gives the example codes on how to show the AMR genes in the notebook.
- ii. The AMR genes data were generated with AMRFinderPlus (v3.10.1), which is included in Sanibel pipeline.
- iii. The input for these codes is the path the output of Sanibel run, and it should look like "**Sanibel/output-date**".
- iv. You may show the table in the jupyter notebook or export to an Excel file, or both.
  1. NOTE: If the number of lines of the table is large (e.g., >500 lines), showing the table in html file may crash the browser and thus it is recommended to export the table to an Excel file.

**f. Geomap plot for metadata.**

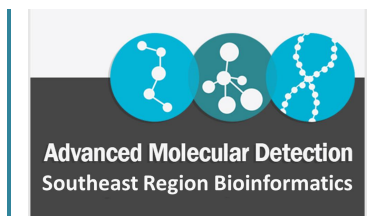
- i. In <https://github.com/BPHL-Molecular/ReportGen/blob/main/ReportGeneration.ipynb>, Section 6 gives the example codes on how to plot geo map by counties in Florida state.
- ii. The input data includes a geojson file "**states-fips.json**" and the metadata.
- iii. You may customize the codesto show certain metadata columns in the hover information.
- iv. To plot other states, you may change the FIPS code in the ReportGeneration.ipynb. FIPS (Federal Information Processing Standard) numeric codes for southeast region states are as follows:

Mississippi: 28  
Georgia: 13  
Alabama: 01  
Tennessee: 47  
Florida: 12  
Louisiana: 22

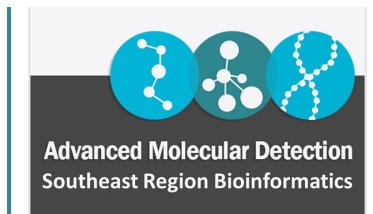
Please update the "12" in the following codes for your own state. For example, Alabama should use "01".

```
county_names = [  
    entry["properties"]["NAME"]  
    for entry in geojson["features"]
```

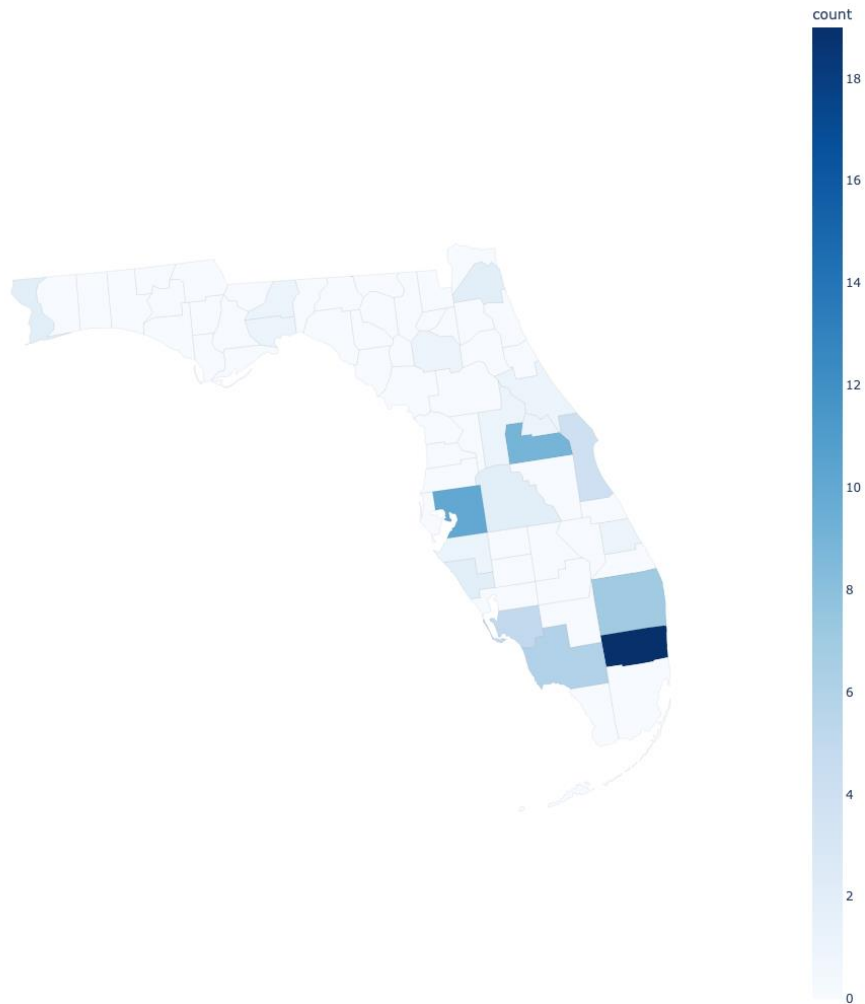


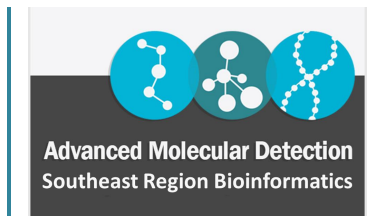


```
        if entry["properties"]["STATE"] == "12" # florida fips is 12
    ]
    county_counts = pd.DataFrame(county_names, columns=["County"])
    county_counts["FIPS"] = [
        entry["properties"]["GEO_ID"][-5:]
        for entry in geojson["features"]
        if entry["properties"]["STATE"] == "12" # florida fips is 12
    ]
```

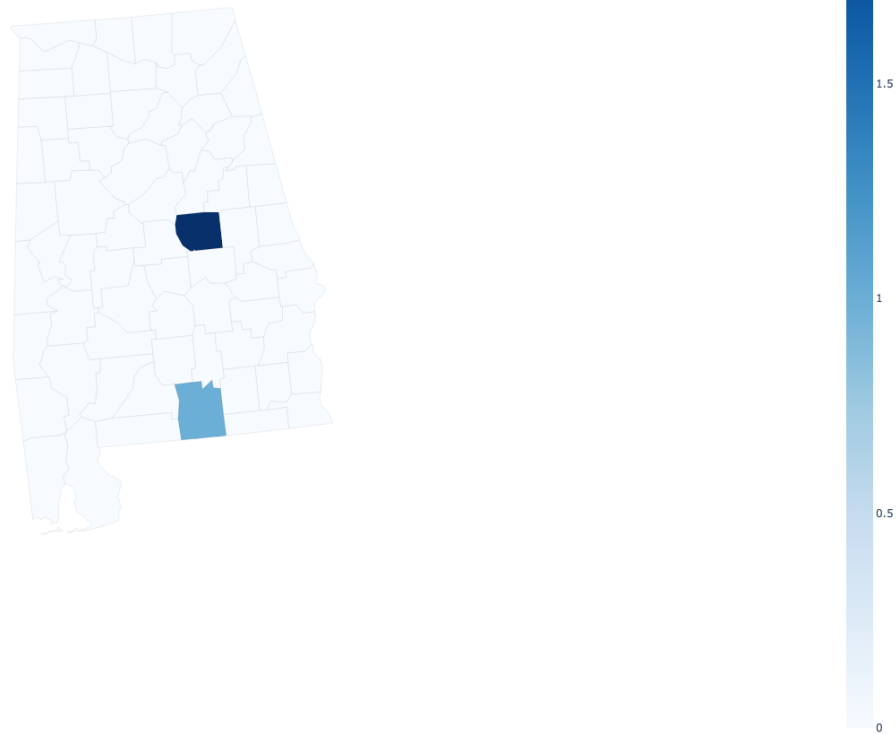


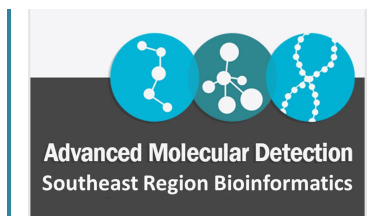
v. The following is an example geo map by county for Florida.



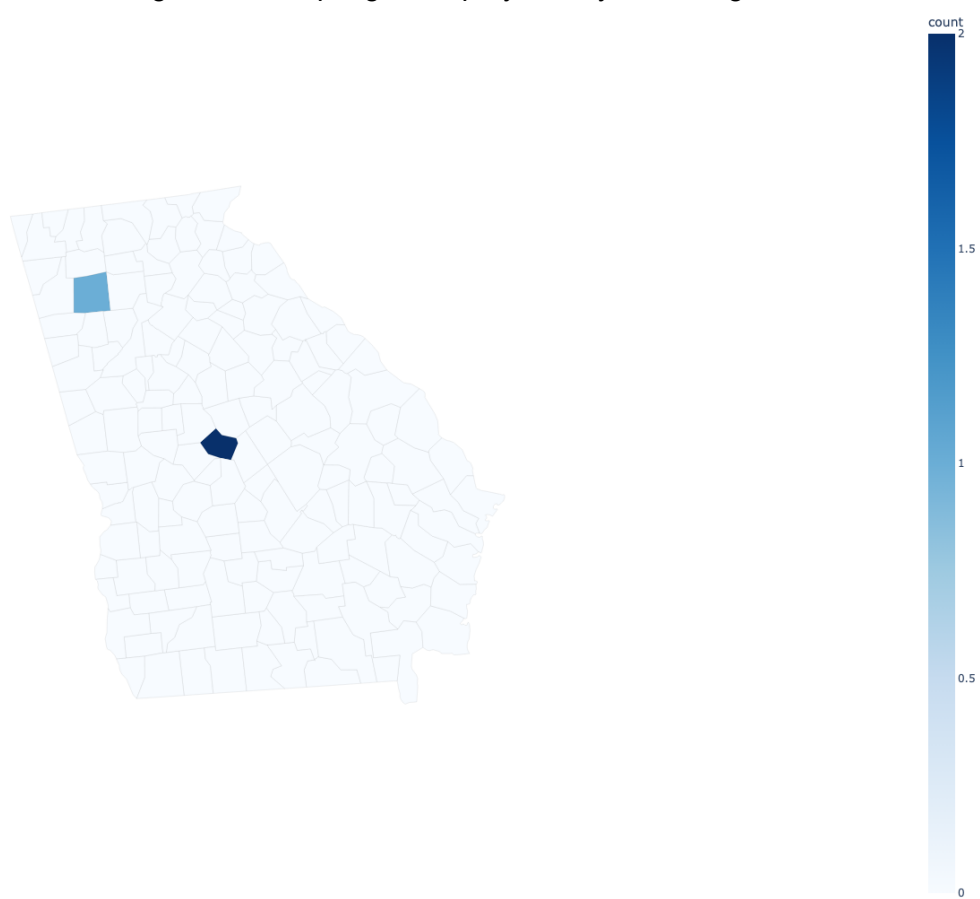


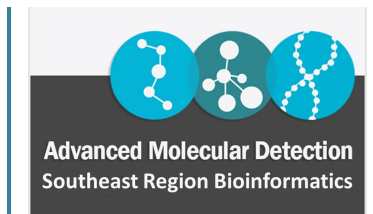
vi. The following is an example geo map by county for Alabama.



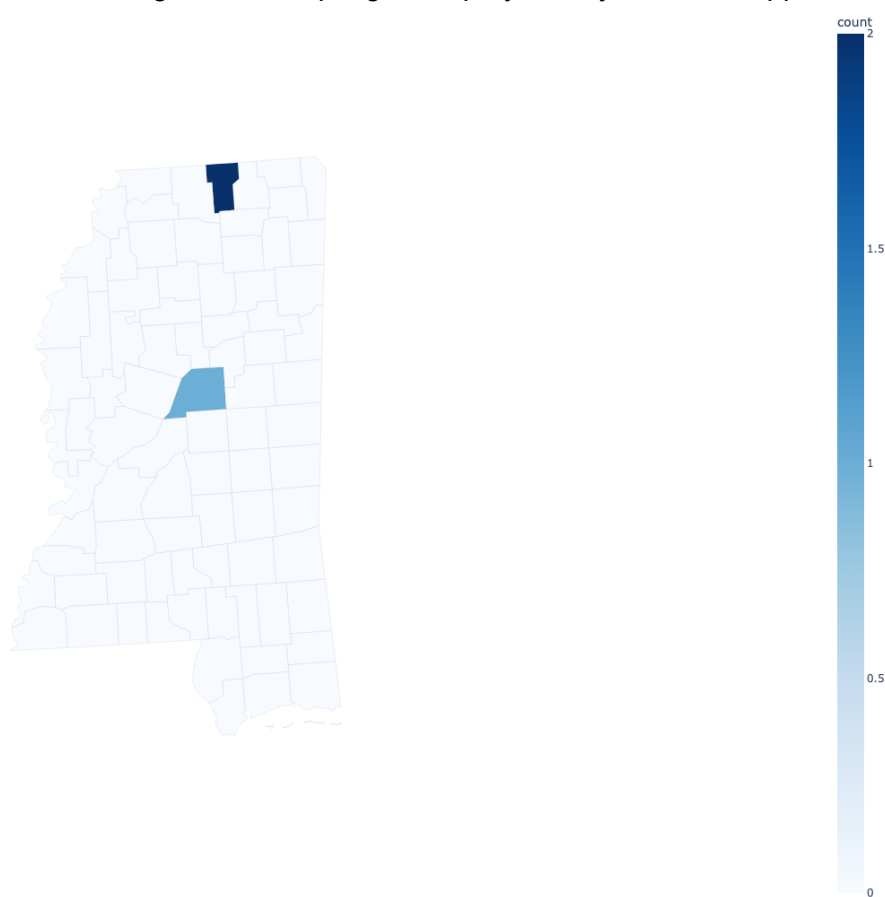


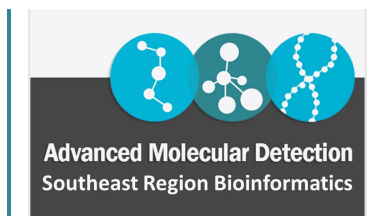
vii. The following is an example geo map by county for Georgia.



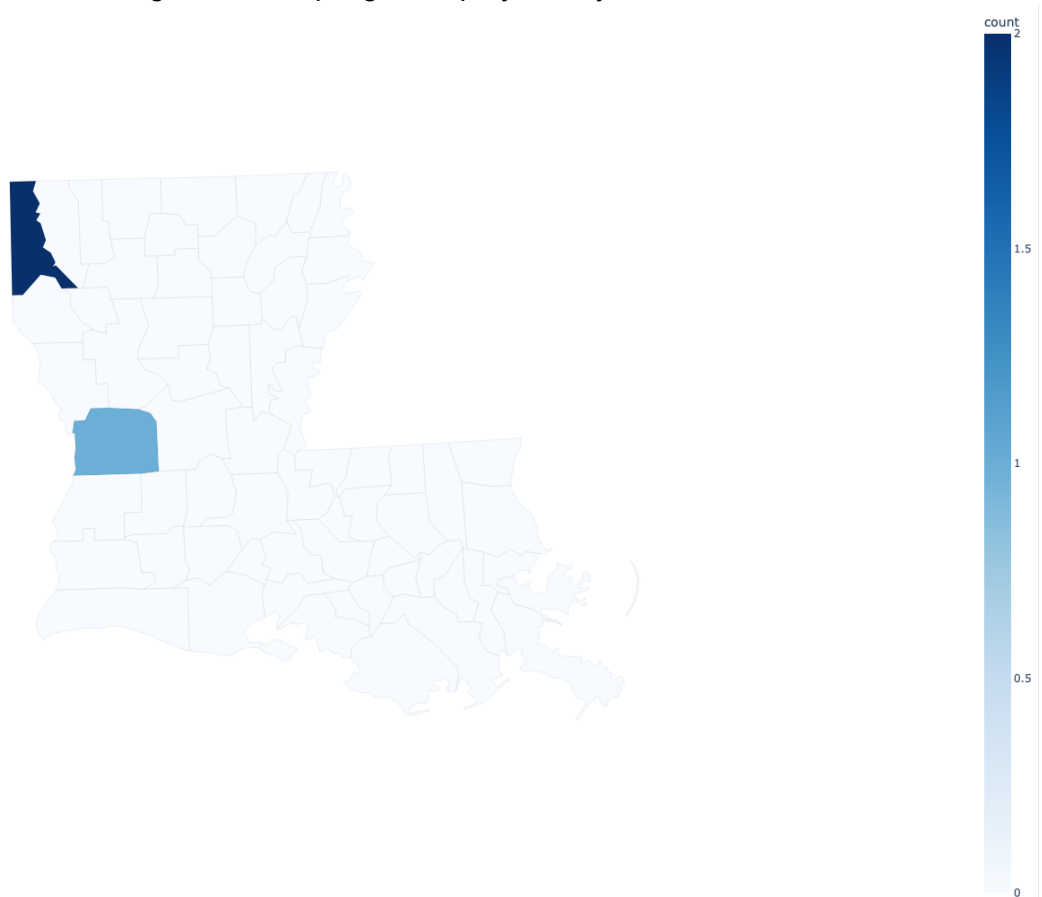


viii. The following is an example geo map by county for Mississippi.

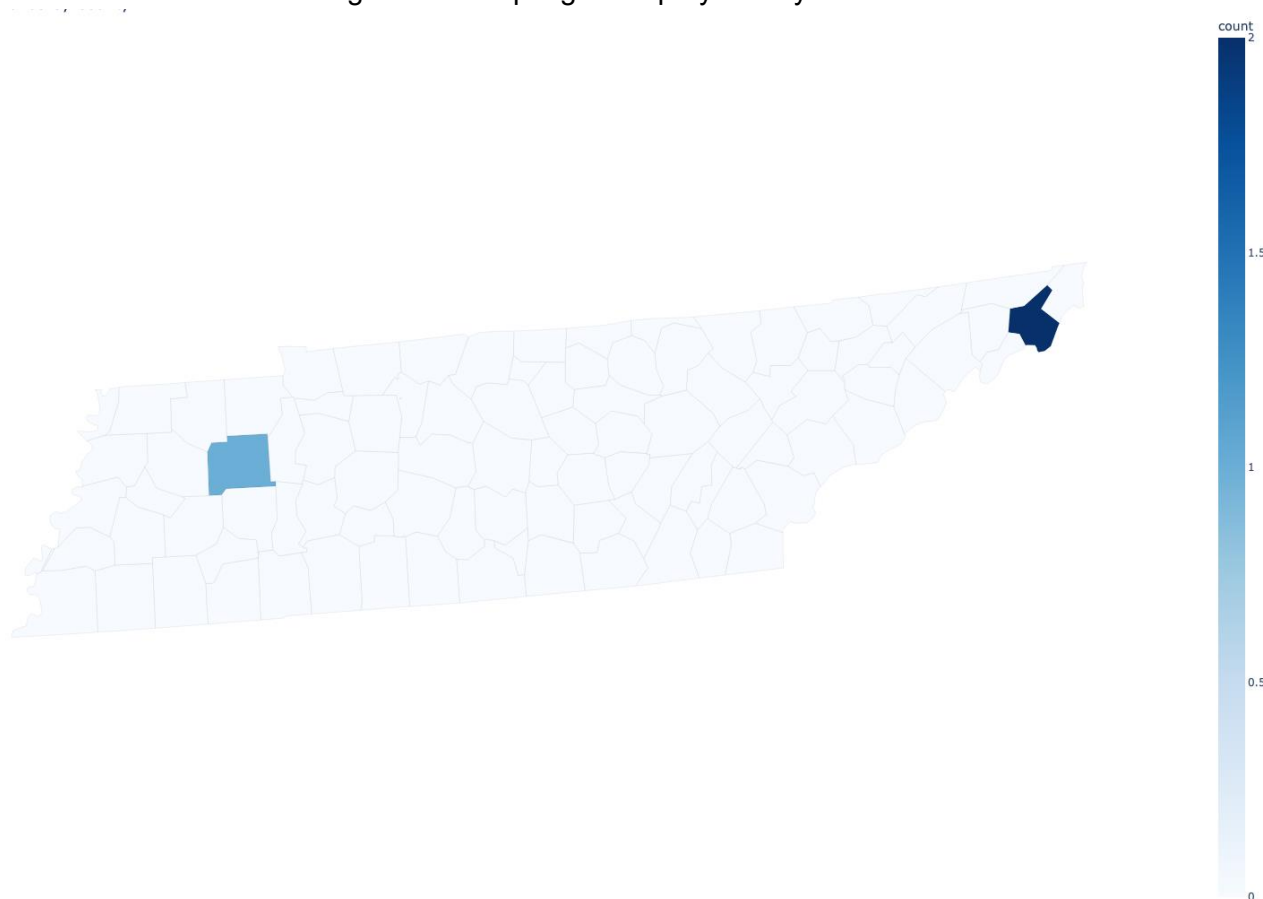




ix. The following is an example geo map by county for Louisiana.



x. The following is an example geo map by county for Tennessee.

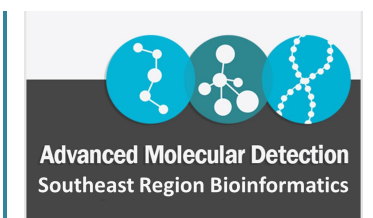


g. **Genetic modifications in loci associated to drug resistance generated by Varpipe-WGS**

This section is only for *Mycobacterium tuberculosis* (TB).

- In <https://github.com/BPHL-Molecular/ReportGen/blob/main/ReportGeneration.ipynb>, Section 7 gives the example codes on how to generate table from the results of Varpipe-WGS.
- If there are less than 100 samples, the codes will generate a table in the jupyter notebook. If there are 100 or more samples the codes will export the table to a csv file. This 100 threshold is configurable. The reason we need to export the table to csv file is because a big table could crash the browser when this html file is opened.
- Make sure you update the file paths of meta\_data and fastq\_varpipe path (which is the folder where you run Varpipe-WGS).

h. **Generate report in html format.**

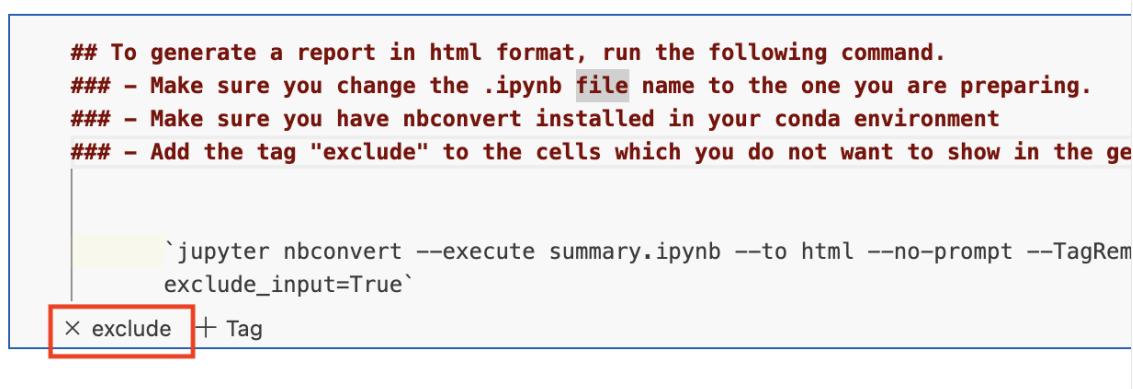
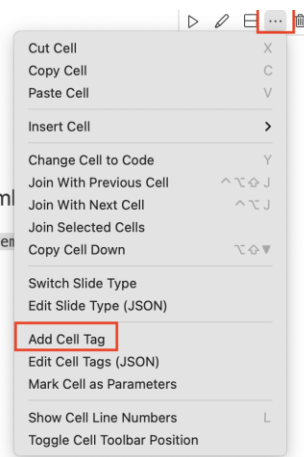


- i. You need to have “nbconvert” installed in your Conda environment.
  1. To install nbconvert, please activate the conda environment you are using, and type `“conda install -c conda-forge nbconvert”`.
- ii. For any cell in the Jupyter notebook that you do not want to display in the final html file, please add an “exclude” tag in the cell, pictured below.
  1. The following figures provide the instruction on how to add a cell tag.

To generate a report in html format, run the following command.

- Make sure you change the .ipynb file name to the one you are preparing.
- Make sure you have nbconvert installed in your conda environment
- Add the tag "exclude" to the cells which you do not want to show in the generated html

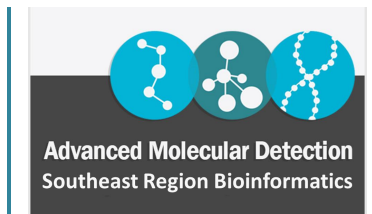
```
`jupyter nbconvert --execute summary.ipynb --to html --no-prompt --TagRemovePreprocessor.remove_cell_tags "exclude" --TemplateExporter.exclude_input=True`
```



- iii. Finally, to generate the html report, run the following command. Make sure update the jpynb file name to the one you prepared. The following example uses “summary.jpynb”.

```
jupyter nbconvert --execute summary.ipynb --to html --no-prompt --TagRemovePreprocessor.remove_cell_tags "exclude" --TemplateExporter.exclude_input=True
```





#### 4. References:

BPHL-Molecular GitHub Repository: <https://github.com/BPHL-Molecular/https://github.com/BPHL-Molecular/>

#### 5. Glossary of Terms:

1. **BPHL:** Bureau of Public Health Laboratory
2. **ST:** Sequence Type
3. **HPG:** HiPerGator, High Performance Cluster hosted by the University of Florida
4. **DNA:** deoxyribonucleic acid, carrier of genetic material in organisms
5. **QC:** Quality Controls
6. **BMGAP:** Bacterial Meningitis Genome Analysis Platform
7. **AMR:** Antimicrobial Resistance
8. **HAI:** Hospital Acquired Infection
9. **MDS:** Multidimensional Scaling ()
10. **UMAP:** Uniform Manifold Approximation and Projection

#### 6. Related Documents: