PhaME

Release 1.0.2

PhaME Development Team

Aug 04, 2020

INTRODUCTION

Phylogenetic and Molecular Evolution (PhaME) analysis tool allows suite of analysis pertaining to phylogeny and moleuclar evolution.

CHAPTER 1

Introduction

1.1 What is PhaME?

PhaME or Phylogenetic and Molecular Evolution (PhaME) is an analysis tool that performs phylogenetic and molecular evolutionary analysis.

Given a reference, PhaME extracts SNPs from complete genomes, draft genomes and/or reads, uses SNP multiple sequence alignment to construct a phylogenetic tree, and provides evolutionary analyses (genes under positive selection) using CDS SNPs.

1.2 PhaME: Under the hood

PhaME's input consists of a set of genomes in fasta and/or fastq formats, and corresponding annotation files in gff3 format if downstream analyses will include coding regions. A detailed step by step explanation of how PhaME analyzes genomes is provided below.

1.2.1 Selecting Reference genome:

Since PhaME is a reference genome-based tool where all input genomes and metagenomes are aligned against a reference, the first step of PhaME's analysis is selecting a reference genome. Given a set of genomes (in a folder listed under the "refdir" parameter of the control file), the reference genome can be selected using one of three options: option 1- a random genome is selected from the provided set of genomes; option 2- a specific genome is selected from the set via input from the user; option 3- the MinHash distance is calculated between all genomes provided (complete genomes, draft genomes, and raw reads) to determine which reference genome has the shortest average distance to all of the other genomes. MinHash distances are calculated using its implementation in BBMap [Bushnell].

1.2.2 Self-nucmerization to remove repeats from reference genomes:

The genome alignment portion of PhaME is built on the tool nucmer2 for alignments of genomes in FASTA format. Each genome included is first aligned with itself using nucmer, called self-nucmerization, and then aligned regions called repeats are removed from the genomes for downstream analyses. The following nucmer command is used for the self-nucmerization step:

```
nucmer --maxmatch --nosimplify --prefix=seq_seq ref_genomeA.fasta ref_genomeA.fasta
```

The option –maxmatch, which reports all matches, is used to ensure that all possible alignments are reported for maximal removal of repeats.

1.2.3 Genome Alignments

All genomes that are in *refdir* are first aligned against the reference genome (see section 1) that have had its repeats removed (section 2). Likewise, incomplete genomes or contigs, the ones that are listed in the *workdir* with extension *.contig* are also aligned against the reference genome using *nucmer*. For aligning genomes in fasta format against each other, following command, same as the previous step for nucmer alignment is used:

```
nucmer --maxmatch refgenome.fasta genome.fasta
```

All other options in nucmer alignments are kept at default, some of the important ones are listed below:

```
-b|breaklen     Set the distance an alignment extension will attempt to
                 extend poor scoring regions before giving up (default 200)
 -c|mincluster   Sets the minimum length of a cluster of matches (default 65)
 -D|diagdiff     Set the maximum diagonal difference between two adjacent
                 anchors in a cluster (default 5)
 -d|diagfactor   Set the maximum diagonal difference between two adjacent
                 anchors in a cluster as a differential fraction of the gap
                 length (default 0.12)
 --[no]extend    Toggle the cluster extension step (default --extend)
 -g|maxgap       Set the maximum gap between two adjacent matches in a
                 cluster (default 90)
 -l|minmatch     Set the minimum length of a single match (default 20)
```

Also, *nucmer* only aligns *“A” “T” “C” “G”*, all other characters are ignored. So, if there are ‘”N”‘s in the provided genomes, thse positions are not included in the alignment.

Note: If an analysis requires running multiple iterations of PhaME on a same set of dataset or a subset of dataset, one doesn’t need to perform the alignment over and over again. PhaME provides an option where it can keep all possible pairwise alignment of genomes from *refdir* for future analyses. All the steps mentioned in this section are the same, except that all vs. all alignment is performed compared to just one reference. By doing all vs. all alignment one can also test the effect on their analyses with different reference genomes.

1.2.4 Mapping of raw reads to reference genome

Currently, PhaME only processes short, raw reads from Illumina. If raw reads, single or paired end, are included in the analyses, they are mapped to the reference genome using either bowtie2 or BWA based on users’ input. For reads mapping to the reference genome, the following commands are used:

First, it builds database from the reference genome.

```
bowtie2-build refgenome refgenome
```

:: or, if BWA was chosen as the preferred aligner:

```
bwa index refgenome
```

The raw reads are then mapped to the reference genomne using one of the following commands:

For bowtie2 and paired reads:

```
bowtie2 -a -x $refgenome -1 read1 -2 read2 -S paired.sam`;
```

:: The option *-a* reports all possible alignments.

For bowtie2 and single end reads:

```
bowtie2 -a -x $refgenome -U read -S single.sam`;
```

For BWA and paired reads:

```
bwa mem refgenome read1 read2 | samtools view -ubS -| samtools sort -T tmp_folder -O␣
↪BAM -o paired.bam
```

For BWA and single end reads:

```
bwa mem refgenome read |samtools view -ubS - | samtools sort -T tmp_folder -O BAM -o␣
↪single.bam
```

1.2.5 Filtering genome alignments

Genome alignment produced using *nucmer* are filtered using *delta-filter* to only keep 1 to 1 alignments allowing for rearrangements. This filtering step is produced for all *nucmer* alignments.

```
delta-filter -1 genome.delta > genome.snpfilter
```

1.2.6 Calling SNPs from genome alignments

The pairwise *nucmer* alignments are then parsed to produce a SNP table using *show-snps*.

```
show-snps -CT genome.snpfilter > genome.snps
```

Here, option C and T specifies not to report SNPs from ambiguous alignments and report the output in tab delimited file respectively.

1.2.7 Reporting nucmer alignments

Each alignments are further parse to produce a tab delimited file that has information on regions and %ID of their alignments.

```
show-coords -clTr genome.snpfilter > genome.coords
```

The parameter flag -clTr implies different headers to be reported in the report.

```
-c          Include percent coverage information in the output
-l          Include the sequence length information in the output
-r          Sort output lines by reference IDs and coordinates
-T          Switch output to tab-delimited format
```

1.2.8 Calling SNPs from read mapping

bcftools mpileup is used for calling SNPs from read mapping results (bam file) of every genomes represented by raw reads. Maximum depth is set to 1000000 for both SNP and indel calling and minimum gaps for calling an indel is set to 3. The output vcf file is then used to call SNPs using *bcftools call* where ploidy is specified as *1* if its a haploid or bacterial genome, else it is called using default parameter. Furthermore, based on the user specified parameter in the control file, SNPs are further filtered based on percentage of SNPs. Here are the snippets of commmand that are run as part of this. All of them result in a vcf file.

```
bcftools mpileup -d 1000000 -L 1000000 -m 3 -Ov -f $refgenome $bam_output | bcftools␣
↪call --ploidy 1 -cO b > $bcf_output;
bcftools view -v snps,indels,mnps,ref,bnd,other -Ov $bcf_output | vcfutils.pl␣
↪varFilter -a$min_alt_bases -d$min_depth -D$max_depth > $vcf_output`;
bcftools filter -i '(DP4[0]+DP4[1])==0 || (DP4[2]+DP4[3])/
↪(DP4[0]+DP4[1]+DP4[2]+DP4[3]) > $snp_filter' $vcf_output > $vcf_filtered`
```

1.2.9 Calculating core genome alignments

As a first step in calculating the core genome, all alignments to reference are checked for linear coverage to assure the proportion of reference genome that was used in the alignment. If its lower than the threshold cutoff (default = 0.6) set in control file, that genome will be removed from further analyses. Then rest of the pairwise alignments that are either in vcf format or nucmer formats are then collated to calculate a core genome. Only the alignment position that are 100% conserved are kept, all other positions are removed from the final core genome alignment. PhaME produces multiple alignment files corresponding to core genome such as the one that has only the variant sites (*_all_snp_alignment.fna*), has variant and invariant sites (*all_alignment.fna*), and the ones that have SNPs from only the coding region (*_cds_snp_alignment.fna*). The coding region SNP alignment requires a GFF formatted annotation file.

1.2.10 Reconstructing core genome phylogeny

PhaME provides multiple tools (RAxML [Stamatakis 2014]_, FastTree [Price 2010]_, and IQ-Tree [Nguyen 2015]_) to reconstruct phylogeny from one core genome alignments that have invariant sites. If RAxML or FastTree option is chosen, users cannot modify the models as they are pre-selected. RAxML trees are reconstructed using GTRGAMMAI models that "GTR + Optimization of substitution rates + GAMMA model of rate heterogeneity (alpha parameter will be estimated)" with *I* but with estimate for invariable sites. FastTree uses GTR model only. IQ-TREE is run using option *-m TEST* that searches for the best model that fits the data before reconstructing the phylogeny. RAxML is the only option that is currently available that can also calculate the bootstraps.

1.2.11 Selecting genes for molecular evolutionary analyses

To perform selection analyses using PAML or HyPhy, codon alignments of genes are required. Based on the position of SNPs in the reference genome, if a SNP is within a coding region and if that coding region does not have a gap, they are extracted from the core genome alignment. The nucleotide sequences of the genes are translated to protein sequences, aligned using the program mafft 8, and then reverse translated back to nucleotide using the Perl code pal2nal.pl from http://www.bork.embl.de/pal2nal/.

1.2.12 Molecular Evoluationary analyses

The set of gene alignments are used for molecular evolutionary analyses using either PAML [Yang 2007]_ or HyPhy. Both packages can test for the presence of positively selected sites and lineages by allowing the dN/dS ratio (ω) to vary among sites and lineages. The adaptive branch-site REL test for episodic diversification (aBSREL) model in the HyPhy package is used to detect instances of episodic diversifying and positive selection. If PAML is selected, the M1a-M2a and M7- M8 nested models are implemented. In the latter case, the likelihood ratio test between the null models (M1a and M8) and the alternative model (M2a and M7) at a significance cutoff of 5% provides information on how the genes are evolving. The results for each gene are then summarized in a table containing information on whether the gene is evolving under positive, neutral, or negative selection, along with p-values. HyPhy is run with a model, which specifically looks for sign of positive selection in given sets of genes. The analysis produces a list of JSON files corresponding to each gene which can be uploaded to vision.hyphy.org/absrel for further analysis. We opted to provide PAML as an option, however we recommend using HyPhy for large projects due to its speed and concise output.

1.2.13 References

CHAPTER 2

How to run PhaME?

After installation, PhaME only requires a control file along with your input files.

```
$ phame phame.ctl
```

2.1 What is in control file?

```
   refdir = test/data/ebola_ref  # directory where reference (Complete) files are␣
↪located
  workdir = test/workdirs/ebola_complete # directory where contigs/reads files are␣
↪located and output is stored

reference = 2  # 0:pick a random reference from refdir; 1:use given reference; 2: use␣
↪ANI based reference
  reffile = KJ660347.fasta  # reference filename when option 1 is chosen

  project = t4  # main alignment file name

  cdsSNPS = 1  # 0:no cds SNPS; 1:cds SNPs, divides SNPs into coding and non-coding␣
↪sequences, gff file is required

  buildSNPdb = 0 # 0: only align to reference 1: build SNP database of all complete␣
↪genomes from refdir

FirstTime = 1  # 1:yes; 2:update existing SNP alignment, only works when buildSNPdb␣
↪is used first time to build DB

     data = 0  # *See below 0:only complete(F); 1:only contig(C); 2:only reads(R);
               # 3:combination F+C; 4:combination F+R; 5:combination C+R;
               # 6:combination F+C+R; 7:realignment  *See below
    reads = 2  # 1: single reads; 2: paired reads; 3: both types present;
```

(continues on next page)

(continued from previous page)

```
     tree = 1  # 0:no tree; 1:use FastTree; 2:use RAxML; 3:use both;
bootstrap = 0  # 0:no; 1:yes;  # Run bootstrapping  *See below
        N = 100  # Number of bootstraps to run *See below

PosSelect = 1  # 0:No; 1:use PAML; 2:use HyPhy; 3:use both # these analysis need gff␣
↪file to parse genomes to genes

     code = 0  # 0:Bacteria; 1:Virus; 2: Eukarya # Bacteria and Virus sets ploidy to␣
↪haploid

    clean = 0  # 0:no clean; 1:clean # remove intermediate and temp files after␣
↪analysis is complete

  threads = 2  # Number of threads to use

   cutoff = 0.1  # Linear alignment (LA) coverage against reference - ignores SNPs␣
↪from organism that have lower cutoff.

   * When using data option 1,2,5 need a complete reference to align/map to.
   * Use data option 7 when need to extract SNPs using a sublist of already aligned␣
↪genomes.
```

It is a simple text file similar to the ones used PAML analysis.

2.2 What do parameters in control file stand for?

1. *refdir* A directory with reference genomes (complete genomes) and their annotation file in gff format (optional). Each file should represent a genome and have following extensions. The path to the folder should be either absolute or relative to the location of the control file. Please avoid filenames that have multiple . or has special characters like *:* in their name.

- *.fasta
- *.fna
- *.fa
- *.gff (optional: to analyze Coding region SNPs of a selected reference file)

For example, a typical reference folder with reference genomes look like this:

```
$ ls ref/
GCA_000006925_2_ASM692v2_genomic.fna   GCA_000017745_1_ASM1774v1_genomic.fna  GCA_
↪000026245_1_ASM2624v1_genomic.fna   GCA_000227625_1_ASM22762v1_genomic.fna
GCA_000007405_1_ASM740v1_genomic.fna   GCA_000017765_1_ASM1776v1_genomic.fna  GCA_
↪000026265_1_ASM2626v1_genomic.fna   GCA_000245515_1_ASM24551v1_genomic.fna
GCA_000008865_1_ASM886v1_genomic.fna   GCA_000017985_1_ASM1798v1_genomic.fna  GCA_
↪000026265_1_ASM2626v1_genomic.gff   GCA_000257275_1_ASM25727v1_genomic.fna
```

Each of these files represent one genome. Each genome may have multiple sequences representing multiple replicons or contigs, but are all part of one genome. *gff* files corresponding to a genome must have the same exact name and in the same folder, just different extension. For example, *gff* file for genome *GCA_000006925_2_ASM692v2_genomic.fna* is *GCA_000006925_2_ASM692v2_genomic.gff*.

2. *workdir* This is the folder where intermediate and final results of analysis are stored. The path to the folder should be either absolute or relative to the location of the control file. Additionally, if the analysis includes incomplete genomes

or contig files and raw reads, they must be in this folder. Contigs file must have following extensions to be recognised as contig file.

- *.contig
- *.contigs

For example, a working directory with contigs folder look like this:

```
$ ls workdir/*.contig\
workdir/GCA_000155105_1_ASM15510v1_genomic.contig  workdir/GCA_000968895_2_ASM96889v2_
↪genomic.contig   workdir/GCA_001514825_1_ASM151482v1_genomic.contig
workdir/GCA_000190495_1_ASM19049v1_genomic.contig  workdir/GCA_000968905.2_ASM96890v2_
↪genomic.contig   workdir/GCA_001514845_1_ASM151484v1_genomic.contig
workdir/GCA_000191665_1_ecmda7_genomic.contig      workdir/GCA_001471755_1_
↪ASM147175v1_genomic.contig  workdir/GCA_001514865_1_ASM151486v1_genomic.contig
```

If the analysis includes reads, they must be in *workdir* as well and decompressed. If reads are paired, they must have same file name at the beginning of the name and *R1* and *R2* at the end of the name and needs to have *.fastq* as their extension (*_'R1.fastq '*_'R2.fastq). *Any file that have '*.fastq* as their extension but dont have paired reads will be treated as single reads. For example, a working folder with paired raw read files loole like this:

```
$ ls *.fastq
GGB_SRR2000383_QC_trimmed_R1.fastq  GGB_SRR2000383_QC_trimmed_R2.fastq  GGC_
↪SRR2164314_QC_trimmed_R1.fastq  GGC_SRR2164314_QC_trimmed_R2.fastq
```

3. ***reference***

   This is where you specify how do you want to pick your reference genome. The available options are:

   - 0: randomly pick a genome from *refdir* folder as the reference genome.
   - 1: use the specified genome as the reference. Genome's filename is specified in the *reffile* option.
   - 2: picks a *mid point* genome based on the Average Nucleotide Identity (ANI) among all genomes. It uses mash (implemented in BBMap) to calculate ANI.

4. ***reffile*** This is where you specify the reference genome, if option 1 is picked in previous option. File name of the genome is written here and the program will look for that file in *reffile* folder. For example, *KJ660347.fasta* in the control file example above is found in the *reffile* folder.

5. ***project*** The name of the project. All the important downstream output filenames will have the specified project name as their prefix.

6. ***cdsSNPS*** This option allows users to parse SNPs based on their position into coding and non-coding sequences. It can be turned ON (0) or OFF (1). If turned ON, the picked reference genome must have a corresponding gff file. This option is automatically turned ON, if Molecular evolutionary analyses is turned ON (see below).

7. ***buildSNPdb*** This option will turn ON (1) or OFF (0) database creation, which is essentially all possible pairwise alignment of all genomes in *refdir*. Turning this ON will significantly increase the runtime.

8. ***FirstTime*** This options default is 1, which reruns everything. The option 2, which only recalculates the SNP matrix only works when SNP database is turned ON in previous step.

9. ***data***

   Select the appropriate option based on the type of data that was included in the analysis.

   - 0: only full/complete(F);

  - Select this option if you only have full/complete genomes or you only want to analyze these genomes from the dataset. Full/COmplete genomes are the ones that are found in *refdir*.

- 1: only contig(C);
  - Select this option if you only have contigs and one reference that is complete. PhaME requires users to input a complete genome as reference. This option will only report contigs in the final alignments and the tree.
- 2: only reads(R);
  - Select this option if you only want to analyze reads file. Remember similar to option 1, a reference must be given.
- 3: combination F+C;
  - Select this option if you have full/complete genome and contigs.
- 4: combination F+R;
  - Select thsi option if you have full/complete genomes and only reads.
- 5: combination C+R;
  - Select this option if you have full/complete genomes and reads. However, it still requires at least one Full/Complete genome.
- 6:combination F+C+R;
  - Select this option if you have full/complete, contigs, and read datasets in your analysis.
- 7:realignment
  - Select this option if you want to realign using a subset of genomes that have already been aligned using one of the option above. It requires editing the *working_list.txt* file.

10. *reads* This option is dependent on option chosen in *data*. If the analysis contains only single reads, enter 1, if paired reads enter 2, and if both are present enter 3.
11. *tree* The option to generate tree. If 0 is entered, no tree is generted. If 1 is entered, only FastTree is used. If 2 is entered, only RAxML is used. If 3 is entered, both FastTree and RAxMl are used to make trees.
12. *bootstrap*
    - The option is valid if 2 or 3 is selected in *tree* option. It will calculate bootstrap trees using RAxML.
13. *N*
    - Specify the number of bootstrap trees to generate if its turned ON in *bootstrap* option.
14. *PosSelect* The option to turn ON and select type of molecular evolution analysis to be done. Enter 0 to turn OFF molecular evolutionary analysis, 1 to use PAML to do molecular evolutionary analysis, 2 to use HyPhy, and 3 to use both of them. Turning this option ON will significantly slow the runtime. If this option is turned ON, you must provide the gff file for the corresponding reference genome.
15. *code* This specifies the pre-calculated parameters during genome alignments.Option 0 which is specific for bacteria uses, *Bacteria* aligns using default option with *maxmatch* for nucmer. And, option 1 which is for'Virus' sets option for nucmer alignment with *maxmatch* turned ON and *-b 200 -c 65 -d 0.12 -g 90 -l 20*.
16. *clean* Turning this option ON (1) will remove intermediate files.
17. *threads* Specify the number of threads to run analysis ON.
18. *cutoff* This options lets user control the genomes to include based on how much of their region was included in the alignemnt against the reference genome. Linear alignment (LA) coverage against reference - ignores SNPs from organism that have lower cutoff.

2.3 What output files are produced?

Summary files (all files are found under folder *workdir/results*)

- SNP alignment files
  - all positions (including monomorphic sites)
    - ‘project‘_all_alignment.fna
  - **all detected SNPs** ‘project‘_all_snp_alignment.fna
  - **SNPs in CDS (coding sequence)** ‘project‘_cds_snp_alignment.fna
  - **intergenic SNPs** ‘project‘_int_snp_alignment.fna
- **Newick tree files**
  - **bootstrap mapped Maximum Likelihood trees**
    - RAxML_bipartitionsBranchLabels.‘project‘_all_best
    - RAxML_bipartitions.‘project‘_all_best
  - **bootstraps**
    - RAxML_bootstrap.‘project‘_all_b
  - **best ML tree**
    - RAxML_bestTree.‘project‘_all
  - RAxML tree using only CDS SNPs
  - **FastTree using all SNPs**
    - ‘project‘_all.fasttree
- **FastTree using SNPs from coding sequence** ‘project‘_cds.fasttree
- **Other files:**
  - **coordinates of gaps throughout the overall alignment** ‘project‘_gaps.txt
  - **the size of gaps between *reference* and other genomes.** ‘project‘_all_gaps.txt
  - **A tab delimited summary file containing information on the core genome size, total SNPs, etc.**

    ‘project‘_summaryStatistics.txt
    - Most rows are genome name (first column), attribute name (second column), and corresponding value (third column) - *Total_length* for genome size (total base pair) of the corresponding genome (first column)
    - *Gap_legnth* for total gaps in the corresponding genome (first column)
    - One row labeled *REPEAT* (first column) and ‘Gap_length‘(second column) correspond to repeat size (third column) of reference genome.
    - *Reference used* shows the name of the reference genome used.
    - *Total gap length:* shows the length of total gaps in the alignment.
    - *Core genome length:* shows the length of genomes that were aligned.
    - *Total SNPs:* shows the length of SNPs.
    - *CDS SNPs:* shows the subset of SNPs from Total SNPs that fall within coding regions.

  - **A pairwise list of all compared position with coordinates between references and samples** ‘project‘_comparison.txt ‘project‘_stats.txt (also contains if SNPs are in coding or non-coding regions)
  - **A matrix file that lists the number of SNPs present between genomes**
    * **all core regions** ‘project‘_snp_coreMatrix.txt
    * **CDS only** ‘project‘_snp_CDSmatrix.txt
    * **intergenic only** ‘project‘_snp_intergenicMatrix.txt
- **Log file** *project*.log
- **Error file** *project*.error

Directory structures

- ***working directory*/files** references (concatenated chromosomes)
- ***working directory*/results** All output files
- ***working directory*/results/snps**

  SNP coordinate files generated from NUCmer and bowtie

  - *g1_g2.snps*: contains pairwise snps between *g1* and *g2*. For example:

```
[P1] [SUB]   [SUB]   [P2]    [BUFF]  [DIST]  [FRM]   [TAGS]
127     T       C    127        22      127     1       1   KJ660347_
↪1_18959    ZEBOV_2002_Ilembe_1_18958
149     T       C    149        6       149     1       1   KJ660347_
↪1_18959    ZEBOV_2002_Ilembe_1_18958
155     C       A    155        6       155     1       1   KJ660347_
↪1_18959    ZEBOV_2002_Ilembe_1_18958
```

- ***working directory*/results/gaps**
  - Gap coordinate files generated from NUCmer and bowtie
- ***working directory*/results/stats**
  - Intermediate stat files generated when parsing NUCmer and Bowtie results - *g1_g2.coords* is a table file that contains regions of genome *g1* and *g2* that were aligned. - *g_repeat_coords.txt* is a table that contains region within genome *g* that were detected as similar. - *g_repeat_stats.txt* contains genome size, repeat segment, and repeat length of genome *g*. For example:

```
ZEBOV_2007_4Luebo size: 18958
Repeats segment #:  0
Repeats total length:   0 (0.00%)
```

    * *repeat_stats.txt* summary of all *g_repeat_stats.txt*.
- ***working directory*/results/temp**
  - Temporary files generated
- ***working directory*/results/PSgenes**
  - All gene fasta files that contain at least 1 SNP, along with their amino acid sequences and codon alignments
- ***working directory*/results/paml**
  - PAML results

- ***working directory*/results/hyphy**
  - HyPhy results
- ***working directory*/results/*_ambiguousSNPpositions.txt**
  - Positions in genomes represented as raw reads where there are ambiguous SNPs.

CHAPTER 3

Conda

PhaME can be installed using [conda](https://conda.io/docs/index.html). If you do not have anaconda or miniconda installed, please do so. Installation of miniconda or anaconda is rather straight forward. After installtion of conda, add channels for bioconda and conda-forge using:

```
$ conda config --add channels r
$ conda config --add channels defaults
$ conda config --add channels conda-forge
$ conda config --add channels bioconda
```

Then simply run

```
conda install phame
```

We recommend creating a separate conda environment for PhaME. You can create a conda environment by:

```
$ conda create -n my_env
$ conda install phame -n my_env
```

CHAPTER 4

Docker

To bypass the installation steps, we have provided a docker [image](https://stackoverflow.com/questions/23735149/what-is-the-difference-between-a-docker-image-and-a-container) for PhaME. To run PhaME from a docker image, follow these steps:

1. Install .. _Docker: https://docs.docker.com/install/
2. Download the latest PhaME image from [quay.io](https:quay.io/)

```
$ docker pull quay.io/biocontainers/phame:1.0.3--0
```

3. Check if the image is correctly downloaded by running the provided test:

```
$ docker run --rm quay.io/biocontainers/phame:1.0.3--0 phame -h
$ docker run --rm quay.io/biocontainers/phame:1.0.3--0 phame -vcheck
```

4. **Run your own data using docker. A step by step guide**

   - Create a folder to mount onto your docker

```
$ mkdir -p phame_analysis_folder
```

   - Create a *refdir* folder with complete genomes within *phame_analysis_folder*

     This folder will contain fasta files representing complete genomes.

```
$ cd phame_analysis_folder
$ mkdir -p refdir
```

     Copy or download genomes and their gff files (if needed) onto this folder.

   - **Create a *workdir* folder within the *phame_analysis_folder*.** This folder will have all the intermediate and final outputs of the analysis including input contigs and reference.

```
$ mkdir -p workdir
```

- **Create a control file.** All the inputs and parameters of a PhaME analysis is set in the control file. Using the provided template create a control file with apprpriate parameters and save it in the *phame_analysis_foler*.
- Run the analysis using docker.

```
$ docker run --rm -v $(pwd)/phame_analysis_folder:/data migun/phame src/phame␣
↪/data/ecoli.ctl
$ git clone https://github.com/mshakya/phame_examples.git
$ docker run --rm -v $(pwd)/phame_examples:/data migun/phame-1 perl src/phame␣
↪/data/ecoli/ecoli.ctl
```

CHAPTER 5

Web-interface.

5.1 Phame App

This Dockerized application contains all of the code you need to run PhaME as a standalone app. It has containers for the PhaME application, the web interface, Celery and Redis queues and a PostGREs database.

5.2 Step by step guide to running PhaME using a web interface on a local machine.

Docker and git are required.

1. clone the repo

```
$ git clone git@github.com:LANL-Bioinformatics/phame-app.git
```

2. cd to the project root directory *phame-app*

```
$ cd phame-app
```

3. run

```
$ cp .envs/.local/.postgres_template phame-app/.envs/.local/.postgres
$ cp .envs/.local/.email_template phame-app/.envs/.local/.email
```

4. Edit the *.postgres* file and change the values for *POSTGRES_USER* and *POSTGRES_PASSWORD*
5. Create docker containers.

```
$ docker-compose -f docker-compose-local.yml build
```

6. start docker

```
$ docker-compose -f docker-compose-local.yml up -d
```

7. initialize the database.

```
$ docker-compose -f docker-compose-local.yml run --rm web /bin/bash -c "python -c
↪'import database; database.init_db()'"
```

If all went well, you can go to *localhost* to see the phame webpage.

5.3 Step by step guide to running PhaME using a web interface on a production machine.

The user input files can require a lot of storage space. Use these instructions if you want to store the users' data on a data volume that is different from the main volume where the Docker container is created.

Go through steps 1-3 as for the local installation and then:

1. Run

```
mkdir -p /path/to/api/uploads
```

2. Update paths in *docker-compose-production.yml* to the volume where you want to store the users' upload files for the *phame* and *web* containers.

```
phame:
    volumes:
      - phame_data:/phame_api/media
      - /path/to/api/uploads:/api/static/uploads
web:
    volumes:
      - phame_data:/phame_api/media
      - /path/to/api/uploads:/api/static/uploads
```

For example set volumes to - */vol_d/api/uploads:/api/static/uploads* if you want to store the upload files on */vol_d*

5. Create docker containers.

```
docker-compose -f docker-compose-production.yml build
```

6. start docker

```
docker-compose -f docker-compose-production.yml up -d
```

5.4 Monitoring tasks

Browse to *localhost:5555* to see the Flower Dashboard. Here you can see the status of the celery workers and their tasks.

You can look at projects run by other users if you create an *admin* account and login to that account. Click on the admin user icon in the upper right corner and select the username for the projects you would like to view.

5.5 Email notifications

If you would like users to receive email notifications with the error and execution logs when their projects have finished running:

1. Setup an email client. We use https://www.mailgun.com/
2. Edit the *.email* file and change the values for *API_KEY*, *EMAIL_URL* and *SENDER*
3. Edit *phame-app/api/config.py* and set *SEND_NOTIFICATIONS = True*

CHAPTER 6

Use Cases

6.1 With only complete genomes

- Download complete genomes (with extension .fasta, .fna) into a folder. For example *ref*

```
mkdir -p ref
```

- **Create a control file. Specify path to *ref* in *refdir* option.** sHere is a typical control file setup for running PhaME with only complete genomes. The most important option here is *data = 0*, which specifies that the dataset only consist of complete genomes and those are in *ref* folder.

```
   refdir = ref  # directory where reference (Complete) files are located
  workdir = workdirs # directory where contigs/reads files are located and output␣
↪is stored

reference = 2  # 0:pick a random reference from refdir; 1:use given reference; 2:␣
↪use ANI based reference
  reffile = KJ660347.fasta  # reference filename when option 1 is chosen

  project = only_ref  # main alignment file name

  cdsSNPS = 1  # 0:no cds SNPS; 1:cds SNPs, divides SNPs into coding and non-
↪coding sequences, gff file is required

  buildSNPdb = 0 # 0: only align to reference 1: build SNP database of all␣
↪complete genomes from refdir

FirstTime = 1  # 1:yes; 2:update existing SNP alignment, only works when␣
↪buildSNPdb is used first time to build DB

         data = 0  # *See below 0:only complete(F); 1:only contig(C); 2:only␣
↪reads(R);
                           # 3:combination F+C; 4:combination F+R; 5:combination␣
↪C+R;
```

(continues on next page)

(continued from previous page)

```
                          # 6:combination F+C+R; 7:realignment  *See below
        reads = 2  # 1: single reads; 2: paired reads; 3: both types present;

         tree = 1  # 0:no tree; 1:use FastTree; 2:use RAxML; 3:use both;
bootstrap = 0  # 0:no; 1:yes;  # Run bootstrapping  *See below
                N = 100  # Number of bootstraps to run *See below

PosSelect = 0  # 0:No; 1:use PAML; 2:use HyPhy; 3:use both # these analysis need␣
↪gff file to parse genomes to genes

         code = 0  # 0:Bacteria; 1:Virus; 2: Eukarya # Bacteria and Virus sets␣
↪ploidy to haploid

        clean = 0  # 0:no clean; 1:clean # remove intermediate and temp files␣
↪after analysis is complete

  threads = 2  # Number of threads to use

   cutoff = 0.1  # Linear alignment (LA) coverage against reference - ignores␣
↪SNPs from organism that have lower cutoff.
```

Save the above file in the same directory where *ref* is, for example as *complete_phame.ctl*

- To run the PhaME using specified control file, one can do

```
$ phame complete_phame.ctl
```

6.2 With complete genomes and contigs

- Download complete genomes (with extension .fasta, .fna) into a folder *ref*. Download incomplete genomes or contigs into a folder *workdir*. If the contigs file do not have the extension *.contig*, then rename all the files to have this extension. One can quickly change the extension using *rename fa contig *.fa*.

```
$ mkdir -p ref
$ mkdir -p workdirs
$ cd workdirs
$ rename fna contig *.fna
```

- **Create a control file. Specify path to *ref* in *refdir* option and path to *workdirs* in *workdir* option.** Here is a typical control file setup for running PhaME with complete genomes and contigs. The most important option here is *data = 3*, which specifies that the dataset consist of complete genomes and contigs.

```
   refdir = ref  # directory where reference (Complete) files are located
  workdir = workdirs # directory where contigs/reads files are located and output is␣
↪stored

reference = 2  # 0:pick a random reference from refdir; 1:use given reference; 2: use␣
↪ANI based reference
  reffile = KJ660347.fasta  # reference filename when option 1 is chosen

  project = only_ref  # main alignment file name

  cdsSNPS = 1  # 0:no cds SNPS; 1:cds SNPs, divides SNPs into coding and non-coding␣
↪sequences, gff file is required
```

(continues on next page)

(continued from previous page)

```
  buildSNPdb = 0 # 0: only align to reference 1: build SNP database of all complete␣
↪genomes from refdir

FirstTime = 1  # 1:yes; 2:update existing SNP alignment, only works when buildSNPdb␣
↪is used first time to build DB

         data = 3  # *See below 0:only complete(F); 1:only contig(C); 2:only reads(R);
                          # 3:combination F+C; 4:combination F+R; 5:combination C+R;
                          # 6:combination F+C+R; 7:realignment  *See below
        reads = 2  # 1: single reads; 2: paired reads; 3: both types present;

         tree = 1  # 0:no tree; 1:use FastTree; 2:use RAxML; 3:use both;
bootstrap = 0  # 0:no; 1:yes;  # Run bootstrapping  *See below
                N = 100  # Number of bootstraps to run *See below

PosSelect = 0  # 0:No; 1:use PAML; 2:use HyPhy; 3:use both # these analysis need gff␣
↪file to parse genomes to genes

         code = 0  # 0:Bacteria; 1:Virus; 2: Eukarya # Bacteria and Virus sets ploidy␣
↪to haploid

        clean = 0  # 0:no clean; 1:clean # remove intermediate and temp files after␣
↪analysis is complete

  threads = 2  # Number of threads to use

   cutoff = 0.1  # Linear alignment (LA) coverage against reference - ignores SNPs␣
↪from organism that have lower cutoff.
```

Save the above control file in the same directory where *ref* is, for example as *contig_and_complete_phame.ctl*

- To run the PhaME using specified control file, one can do

```
$ phame contig_and_complete_phame.ctl
```

6.3 With raw reads, complete genomes, and contigs

- Download complete genomes (with extension .fasta, .fna) into a folder *ref*. Download incomplete genomes or contigs into a folder *workdir*. And, download fastq files in *workdirs* as well. If the contigs file do not have the extension *.contig*, then rename all the files to have this extension. Paired fastq files must have R1.fastq and R2.fastq as suffixes in their filename.

```
$ mkdir -p ref
$ mkdir -p workdirs
$ cd workdirs
$ rename fna contig *.fna
```

- **Create a control file. Specify path to *ref* in *refdir* option and path to *workdirs* in *workdir* option.** Here is a typical control file setup for running PhaME with complete genomes and contigs. The most important option here is *data = 6*, which specifies that the dataset consist of complete genomes and contigs.

```
   refdir = ref  # directory where reference (Complete) files are located
  workdir = workdirs # directory where contigs/reads files are located and output is␣
↪stored
```

(continues on next page)

(continued from previous page)

```
reference = 2  # 0:pick a random reference from refdir; 1:use given reference; 2: use␣
↪ANI based reference
  reffile = KJ660347.fasta  # reference filename when option 1 is chosen

  project = only_ref  # main alignment file name

  cdsSNPS = 1  # 0:no cds SNPS; 1:cds SNPs, divides SNPs into coding and non-coding␣
↪sequences, gff file is required

  buildSNPdb = 0 # 0: only align to reference 1: build SNP database of all complete␣
↪genomes from refdir

FirstTime = 1  # 1:yes; 2:update existing SNP alignment, only works when buildSNPdb␣
↪is used first time to build DB

         data = 6  # *See below 0:only complete(F); 1:only contig(C); 2:only reads(R);
                        # 3:combination F+C; 4:combination F+R; 5:combination C+R;
                        # 6:combination F+C+R; 7:realignment  *See below
        reads = 2  # 1: single reads; 2: paired reads; 3: both types present;

         tree = 1  # 0:no tree; 1:use FastTree; 2:use RAxML; 3:use both;
bootstrap = 0  # 0:no; 1:yes;  # Run bootstrapping  *See below
            N = 100  # Number of bootstraps to run *See below

PosSelect = 0  # 0:No; 1:use PAML; 2:use HyPhy; 3:use both # these analysis need gff␣
↪file to parse genomes to genes

         code = 0  # 0:Bacteria; 1:Virus; 2: Eukarya # Bacteria and Virus sets ploidy␣
↪to haploid

        clean = 0  # 0:no clean; 1:clean # remove intermediate and temp files after␣
↪analysis is complete

  threads = 2  # Number of threads to use

   cutoff = 0.1  # Linear alignment (LA) coverage against reference - ignores SNPs␣
↪from organism that have lower cutoff.
```

Save the above control file in the same directory where *ref* is, for example as *read_contig_and_complete_phame.ctl*

- To run the PhaME using specified control file, one can do

```
$ phame read_contig_and_complete_phame.ctl
```

6.4 Zoom-in analysis

PhaME has a unique feature that allows subsetting genomes from a PhaME analysis.

1. One can select genomes of interest by creating a copy of *working_list.txt*, a text file generated during the first run that contains list of all genomes then deleting lines containing names of genomes that are not of interest. For example,

```
$ cat working_list.txt
SRR1610032_pread
```

(continues on next page)

(continued from previous page)

```
    SRR1610033_pread
    GCF_000703365_1_Ec2011C_3609_genomic
    SRR1609871_pread
    SRR1609862_pread
    SRR160986_pread
    SRR1610031_pread
    SRR1610034_pread
    SRR1610028_pread
    SRR1610029_pread
```

2. Remove lines with genomes that are not of interest.

```
    $ cat working_list.txt
SRR1610032_pread
    SRR1610033_pread
    GCF_000703365_1_Ec2011C_3609_genomic
    SRR1609871_pread
```

3. Change the *data* option in control file to *data = 7*, maybe also change the *project* name, and save as zoom_in.ctl
4. Then rerun the PhaME as

```
$ phame zoom_in.ctl
```

Bibliography

[Bushnell] Bushnell B: BBMap. 37.66 edition. sourceforge.net/projects/bbmap/.