vRhyme

Binning Virus Genomes from Metagenomes

```
December 2021
Kristopher Kieft
kieft@wisc.edu
Anantharaman Lab
University of Wisconsin-Madison
```

Current Version

vRhyme v1.0.0

Citation

If you find vRhyme useful please consider citing our manuscript on bioRxiv:

Kieft, K., Adams, A., Salamzade, R., Kalan, L., & Anantharaman, K. vRhyme enables binning of viral genomes from metagenomes. bioRxiv, 2021.

Table of Contents:

Updates for v1.0.0 (December 2021):

- Initial release
- Fixed pip install bugs

Program Description

vRhyme Description

vRhyme is a multi-functional tool for binning virus genomes from metagenomes. vRhyme functions by utilizing coverage variance comparisons and supervised machine learning classification of sequence features to construct viral metagenome-assembled genomes (vMAGs).

IMPORTANT NOTE: vRhyme is built to run on viral sequences/scaffolds. A typical workflow is to predict viruses from a metagenome (e.g., with VIBRANT or VirSorter) and then use those predictions as input to vRhyme. vRhyme can take an entire metagenome as input, but the performance for a whole metagenome has not been fully evaluated. vRhyme is not meant to bin microbes.

Why "*vRhyme*"? The similarity in sequence features between two scaffolds can be used to identify fragments of the same genome, such as tetranucleotide frequencies, codon usage or GC content. It's almost like rhyming sequences together to create pairs that sound similar, at least metaphorically at the nucleotide level. Coverage variance helps to separate scaffolds that sound the same but are actually different genomes.

vRhyme Features

- Binning vMAGs
- Automatic bin refinement
- Automatic read coverage calculation and refinement
- Bin scoring and redundancy checks (Mmseqs2)
- Supervised machine learning: neural networks and decision trees
- Scaffold dereplication (Mash, Nucmer)
- Read alignment (Bowtie2, BWA)
- ORF prediction (Prodigal)

Installation

GitHub, pip, and conda

1. `git clone https://github.com/AnantharamanLab/vRhyme`
2. *optional* create a conda environment (see examples below)
3. *optional* activate conda environment if you made one
4. `cd vRhyme`
5. `pip install .` ← *NOTE*: don't forget the dot (`pip install [dot]`)

Installing with pip is optional but suggested. Using pip will collect Python dependencies and add vRhyme to your system PATH. Note that `vRhyme.egg-info` and `build` should be created

after the pip install. Without pip, vRhyme can still be executed directly from the git clone, just ensure executable permissions (`cd vRhyme/; chmod +x vRhyme scripts/*.py aux/*.py`). The conda environment is also optional but can be useful for downloading and managing program dependencies.

Example Conda Environments

- with pip install

  1. `conda create -c bioconda -n vRhyme python=3 samtools mash mummer mmseqs2 prodigal bowtie2 bwa`
  2. `conda activate vRhyme`

- directly from git clone

  1. `conda create -c bioconda -n vRhyme python=3 pandas numpy numba scikit-learn pysam samtools mash mummer mmseqs2 prodigal bowtie2 bwa`
  2. `conda activate vRhyme`

Test the Installation

Test and validate the installed dependencies for vRhyme. In these tests you're looking for *Success* statements. All Python dependencies must be *Success*. Please update them if prompted. Both machine learning models must be *Success*. For program dependencies, only Mmseqs is required to be *Success*, the others are optional depending on vRhyme usage. If you use any coverage input beside `-c` then Samtools is required to be *Success*. If you skip vRhyme `-p` or `-g` flags then Prodigal must be *Success*. If you plan to use vRhyme's dereplication function (see `--method`) then Mash and Nucmer must be *Success*. If you input reads (`-r/-v/-u`) then Bowtie2 and/or BWA (see `--aligner`) must be *Success*.

- with pip install
  `test_vRhyme.py`

- directly from git clone
  `aux/test_vRhyme.py`

Print the main vRhyme help page:
`vRhyme -h`

Requirements

Program Dependencies

Please ensure the following programs are installed and in your machine's PATH. Note: most downloads will automatically place these programs in your PATH. See each program's source website for installation guides.

Required

1. [Python3](#) (version >= 3.6)
2. [Mmseqs2](#)
3. [Samtools](#)

Optional (depends on usage)

4. [Prodigal](#)
5. [Mash](#)

6. [Nucmer](#)
7. [Bowtie2](#)
8. [BWA](#)

Python3 Dependencies

There are several Python3 dependencies that must be installed as well. You may already have most of these installed. See each package's source website for installation guides. Versions are important.

1. [Pandas](#) (version >= 1.0.0)
2. [Numpy](#) (version >= 1.17.0)
3. [Scikit-learn](#) (version >= 0.23.0)
4. [Numba](#) (version >= 0.50.0)
5. [PySam](#) (version >= 0.15.0)

Running vRhyme

Test run examples

Test out vRhyme on the provided example dataset. *NOTE*: If you choose to not install with pip, the `vRhyme` executable is within the `vRhyme/` subdirectory.

```
cd examples/
```

minimal coverage table input example

```
vRhyme -i example_scaffolds.fasta -c example_coverage_values.tsv -t 1
```

full coverage table input example

```
vRhyme -i example_scaffolds.fasta -o vRhyme_example_results_coverage-
table/ -c example_coverage_values.tsv -p example_scaffolds.prodigal.faa -
g example_scaffolds.prodigal.ffn -t 2
```

Quick run examples

minimum input example with bam files

```
vRhyme -i fasta -b bam_folder/*.bam
```

minimum input example with a coverage file

```
vRhyme -i fasta -c coverage_file.tsv
```

full BAM input example

```
vRhyme -i fasta -g genes -p proteins -b bam_folder/*.bam -t threads -o
output_folder/
```

reads input example with dereplication

```
vRhyme -i fasta -g genes -p proteins -r paired_reads_folder/*.fastq -t
threads -o output_folder --method longest
```

only use dereplicate function

```
vRhyme -i input_fasta -t threads -o output_folder/ --derep_only --method
longest
```

Output Explanations

Useful outputs

- *log_vRhyme_(-i).log*: main log file
- *vRhyme_best_bins.#.membership.tsv*: scaffold membership of best bins
- *vRhyme_best_bins.#.summary.tsv*: summary stats of best bins
- *vRhyme_best_bins_fasta/*: folder containing fasta files for best bins

Other outputs

- *log_vRhyme_paired_reads.log*: log of which reads files were paired together with `-r`. Exists with `-r`.
- *(-i).prodigal.faa/(-i).prodigal.ffn*: proteins/genes predicted by Prodigal for the length-filtered input sequences. Exists in the absense of `-p/-g`.
- *vRhyme_machine_distances.tsv*: raw distance calculation values used as input for the machine learning model(s)
- *vRhyme_coverage_files*: folder containing one file per sample with coverage values per scaffold. The row names can be found in *vRhyme_names.txt*. *vRhyme_coverage_values.tsv* is the combination of all sample coverages and is equivalent to `-c`. Does not exist with `-c`.
- *vRhyme_alternate_bins*: folder containing membership and summary files for each alternative binning iteration. *vRhyme_bin_scoring.tsv* contains the scoring information and rationale for the best iteration chosen as the best bins.
- *vRhyme_bam_files*: contains vRhyme-generated BAM files. Exists with `-r/-u/-v/-s`. Only sorted BAM files will exist unless `--keep_bam`.
- *vRhyme_sam_files*: contains vRhyme-generated SAM files. Exists with `-r/-u/-v` and `--keep_sam`.

Hierarchy

- '>' : folder
- '-' : file
- '*' : indicates file/folder may not exist

```
> main output folder (-o)
    - log_vRhyme_(-i).log
    - log_vRhyme_paired_reads.log *
    - (-i).prodigal.faa *
    - (-i).prodigal.ffn *
    - vRhyme_best_bins.#.membership.tsv
    - vRhyme_best_bins.#.summary.tsv
    - vRhyme_machine_distances.tsv
    > vRhyme_coverage_files *
        - (sample).coverage.tsv
        - vRhyme_coverage_values.tsv
        - vRhyme_names.txt
    > vRhyme_alternate_bins
        - #.membership.tsv
        - #.summary.tsv
        - vRhyme_bin_scoring.tsv
    > vRhyme_best_bins_fasta
        - vRhyme_bin_#.faa
        - vRhyme_bin_#.fasta
        - vRhyme_bin_#.ffn
    > vRhyme_bam_files *
        - (sample).sorted.bam
        - (sample).bam *
```

```
          - (sample).bam.bai *
    > vRhyme_sam_files *
          - (sample).sam
```

vRhyme files and folders

- *aux/*: folder containing auxiliary scripts. These scripts are not used when running the main vRhyme software and are intended to be run either pre- or post-binning. For all scripts listed below used `-h` for a help message with usage.
    - *test_vRhyme.py*: test the installed dependencies for vRhyme.
    - *alternative_bins_to_fasta.py*: if you wish to use an alternative iteration from vRhyme binning this script will extract fasta files per bin for that iteration.
    - *bin_coverage.py*: calculate the average coverage of a bin per sample (combined averages of all bin members).
    - *coverage_table_convert.py*: convert a MetaBat2 'jgi_summarize_bam_contig_depths' generated coverage table into vRhyme format for -c input.
    - *extract_unbinned_sequences.py*: extract unbinned scaffolds into a separate fasta file.
    - *generate_bin_scores.py*: calculate vRhyme-equivalent scores per bin. This script will run the same protein redundancy and scoring method implemented by vRhyme. Note that in the output table the score will be listed in the final row.
    - *link_bin_sequences.py*: link scaffolds within a bin to generate a single sequence per bin. This may be useful for CheckV input. Note that after linking scaffolds, if Prodigal is used to predict open reading frames the Prodigal `-m` flag should be used.
    - *sequence_lengths_of_bins.py*: calculate the total nucleotide length of each bin.
- *example/*: folder containing example tests and output files for validating vRhyme installation and setup.
- *images/*: folder containing PNG images with the vRhyme flowchart, flag compatibility descriptions, and logo.
- *models/*: folder containing two machine learning models for the neural network (NN) and extra trees classifiers (ET).
- *scripts/*: folder containing essential scripts that perform vRhyme functions. These scripts are not intended to be used on their own.
- *vRhyme*: main executable.

vRhyme Flags

Flag and input compatibility

Flag	Description	Incompatibility
-i	Input nucleotide genomes, contigs, scaffolds in FASTA format. Always required	None
-o	Output folder to deposit results. Will create if doesn't exist or isn't provided	None
-g / -p	Input genes (-g) and proteins (-p) in FASTA format. If only one of -g or -p is provided then Prodigal will still be run	None
-s	Input SAM alignment file(s)	`--method composite` / -c
-b	Input BAM alignment file(s)	`--method composite` / -c
-c	Input coverage table	`--method composite` / -r / -u / -v / -b / -s
-r	Input paired reads files	-c
-u	Input unpaired reads file(s)	-c
-v	Input interleaved paired reads file(s)	-c
Notes		
The inputs -s/-b/-r/-u/-v are typically not provided together but are compatible. First, reads inputs (-r/-u/-v) will be aligned to create SAM files and then converted into BAM files. Next, SAM file inputs (-s) will be converted into BAM files and all BAM files, including any inputs (-b), will be used to create the coverage table (equivalent to -c).		
BAM inputs (-b) can be sorted/indexed. If not, then vRhyme will automatically sort and index them.		
The suffix/extension of input reads is important. For all input reads the extension must be .fastq or .fastq.gz. For paired reads (-r) the suffix should be _1.fastq for forward and _2.fastq for reverse to ensure proper pairing during alignment.		
Incompatibilities that are ignored (vRhyme will run): • "Read Alignment" flags with -b / -s / -c inputs • "Dereplication" flags with `--method none` • "Bin Filters" / "Read Alignment" flags with `--derep_only` • All input files besides -i / --interest with `--derep_only`		

Flag explanations

Commonly Used

(typical usage inputs and options)

- `-i`: (required) input nucleotide scaffolds. For best results, this is predicted virus scaffolds from a metagenome or entire virome assembly. Alternatively, an entire metagenome can be used as input, but this has not been extensively tested.

- `-o`: output folder to deposit all results. vRhyme will exit if the folder already exists.

- `-g`: input nucleotide genes in the format 'name_#'.

- `-p`: input amino acid proteins genes in the format 'name_#'.

- `-b`: BAM (extension '.bam') sequence alignment files. The files can sorted or unsorted (each will be checked individually) but vRhyme will use sorted and indexed BAM files for coverage. Example usage: `-b bam_file_folder/*.bam`

- `-t`: number of parallel processes (threads) to run. With `-c/-p/-g` inputs vRhyme runtime is sufficient with low `-t` values (e.g., 1). vRhyme will be most efficient when `-t` is set to the same value as `--iter` such that 1 iteration (`--iter`) is run on 1 thread (`-t`) simultaneously. The same is true for whole integer multiples. For example: `-t 10 --iter 10`, `-t 20 --iter 10`, `-t 30 --iter 15`. Note that any `-t` and `--iter` combination can be used.

- `-l`: minimum scaffold length to consider for binning. The default minimum is 2000bp (2kb) and cannot be reduced. Scaffolds <2kb do not contain enough information for accurate binning and can create erroneous results.

- NOTE: For FASTA inputs (`-i/-g/-p`) most characters are accepted with few exceptions (e.g., pipe symbol `|` and comma `,`).

- NOTE: `-g` and `-p` can be skipped and Prodigal will be used to predict open reading frames

- NOTE: For `-g/-p`, the format 'name_#' is Prodigal format. 'name' can be any sequence name with '_#' designating the protein/gene number. For example, 'example_sequence_1' and 'example_sequence_2' is protein/gene 1 and 2.

- NOTE: Prodigal is run with `-m -p meta`.

Other Inputs
(mostly inputs besides -b)

- `-s`: SAM (extension '.sam') sequence alignment files. vRhyme will convert these files directly to BAM format before processing. Example usage: `-s sam_file_folder/*.sam`

- `-r`: paired forward (_1.fastq/_R1.fastq) and reverse (_2.fastq/_R2.fastq) read files. The extensions and suffix terminology is important to ensure proper pairing during alignment. With this input a file called 'log_vRhyme_paired_reads.tsv' will be created to validate that correct pairing was identified. Example usage: `-r paired_reads_folder/*.fastq`

- `-u`: unpaired (.fastq) read file(s). This is for single end sequencing and will not be considered with `-r`. Example usage: `-u unpaired_reads_folder/*.fastq`

- `-v`: interleaved paired (.fastq) read file(s). Compatible with `--aligner bowtie2` only. Example usage: `-v interleaved_reads_folder/*.fastq`

- `-c`: coverage table input. See the example run for formatting if you want to generated this file manually. This may be useful if re-binning after a coverage table has already be generated by vRhyme or if an error caused vRhyme to exit after the coverage table was created. This input can also come from using cov_table_convert.py on a coverage table generated by 'jgi_summarize_bam_contig_depths'.

- `--interest`: file containing a list of scaffolds of interest to dereplicate and/or bin. Use this if your input (`-i`) file contains more scaffolds than what you want to bin or dereplicate. The list is new line separated.

- NOTE: all read inputs can be in gzip (.gz) format

- NOTE: It is highly suggested to bin using at least 2-3 samples for coverage covariance comparisons (e.g., 3 `-b` input files). vRhyme will still function and bin with 1 sample.

- NOTE: BWA is run with 'mem'. A BWA database/index will be created if it doesn't exist in the location of the `-i` file. Use Bowtie2 or separate read map software of choice for interleaved reads or if reads are improperly formatted for BWA.

- NOTE: Bowtie2 is run with '--no-unal --no-discordant' using `-r` or `-v`. A Bowtie2 database/index will be created if it doesn't exist in the location of the `-i` file.

Edit Outputs
(these typically do not need to be modified and do not effect binning results)

- `--keep_sam`: (bool, just use the flag) If used, do not remove SAM files generated by vRhyme. Otherwise all SAM files generated from vRhyme read alignment will be deleted. All input SAM files (`-s`) are not effected.

- `--keep_bam`: (bool, just use the flag) If used, do not remove unsorted/index BAM files generated by vRhyme. Otherwise all unsorted/index BAM files generated from vRhyme read alignment or SAM conversions will be deleted. All input and final sorted BAM files (`-b`) are not effected.

- `--speed`: (bool, just use the flag) If used, allow extra CPU multithreading per process (may use more CPUs than `-t #` per process). This is effected by the environments OMP_NUM_THREADS, OPENBLAS_NUM_THREADS, MKL_NUM_THREADS, VECLIB_MAXIMUM_THREADS, or NUMEXPR_NUM_THREADS.

- `--verbose`: (bool, just use the flag) If used, write log file information to standard out while running. This is identical to what is written to the log file.
- `--prefix`: The prefix to append to binned scaffold names. The default is 'vRhyme_#__'. Use the '#' symbol where the bin number will be noted. For example, by using `--prefix BIN_#__` bin 1 will have the prefix 'BIN_1__', bin 2 will be 'BIN_2__' and so on.

Read Alignment
(select read map software or modify alignment filtering)

- `--aligner`: read alignment software to use. Either 'bowtie2' (default) or 'bwa'. Other alignment software are not supported at this time. Use BAM (`-b`), SAM (`-s`) or coverage table (`-c`) inputs for alignments from other software.
- `--read_id`: minimum percent identity per aligned read for calculating coverage. The default is '0.97' (97%). Percent identity is calculated per read as the number of exact matches divided by the length of the aligned read. Exact matches are calculated by the length of the aligned read minus the number of gaps and mismatches. This is the quality trimming of the read alignment as some reads are aligned at poor identity depending on alignment software settings. This is compatible with reads file inputs or BAM/SAM inputs.
- `--mask`: mask coverage values <\int> bases on each end of a scaffold. The default is 150. For short read alignments the coverages can be skewed at scaffold ends and impact coverage covariance calculations. Masking (not including) these coverage values on each end can produce better results. The default of 150 is suitable for most situations, but changing this value to the average read length may be beneficial.

Bin Filters
(these typically do not need to be modified)

- `--bin_size`: minimum number of scaffolds per bin. The default is 2. For example, setting this to 5 means all reported bins must contain at least 5 scaffolds. Low values, such as 2, are best for binning viruses that have smaller genomes.
- `--iter`: number of binning iterations (presets) to run. The default is 10 and the available range is 8-15. The presets are immutable threshold settings for the machine learning model probability, Cohen's *d* coverage effect size, and network edge weight cutoffs. Each iteration will run a single preset. The minimum setting of 8 is usually sufficient to identify the best binning iteration. There is a maximum of 15 possible presets. Setting `--iter` higher (e.g., 15) has a slight reduction in speed and provides a better opportunity to identify the best binning iteration, though the presets that typically result in the best binning iterations are 0 to 7.
- `--red`: maximum number of redundant proteins per bin. The default is 50. This is a cutoff for how many of a bin's proteins can be redundant for generating/reporting the bin. The default of 50 means most bins are reported. For example, setting `--red 1` would result in only creating bins with <=1 redundant protein, which is a typical low-contamination bin. Redundant proteins contained on a single scaffold are not considered redundant. Modifying this value does not impact how bins are formed, only how bins are reported. Redundancy per bin is reported to allow for filtering post-binning. Note that some eukaryotic viruses, and few prokaryotic viruses, can normally have up to 10 redundant proteins. Uncommon exceptions may have greater than 10.
- `--cov`: coverage threshold to consider scaffold as present in sample. The default is 0.90. Any scaffold with an average coverage below this threshold is not considered as present. Typically, lowering this value increases recall along with contamination, whereas increasing this value lowers recall and contamination.

- `--model`: machine learning model(s) to use. The default is 'hybrid', which means both the Neural Network (NN) and Extra Trees (ET) models are used. There is a minor reduction in runtime when using both models, but it is more accurate than either on their own. When both models are used the probability output from both models needs to meet the given threshold.
- `--max_gc`: maximum GC distance for pre-filtering. The default is 0.20. This is the difference in %GC between two scaffolds being compared. Scaffolds from the same genome tend to have similar GC content. Lowering this value (e.g., 0.10 or 0.15) will minorly increase speed for large datasets because fewer scaffolds will be compared with more computationally expensive tasks. Setting this value to 1.00 will turn off any filtering.
- `--min_kmer`: minimum k-mer distance for pre-filtering. The default is 0.60. This is the calculated distance between two scaffolds' tetranucleotide frequencies. Scaffolds that are more alike will have higher values (e.g., >0.80) and those that are more dissimilar will be lower (e.g., <0.40). Increasing this value (e.g., 0.70 or 0.80) will minorly increase speed for large datasets because fewer scaffolds will be compared with more computationally expensive tasks. Setting this value to 0.00 will turn off any filtering. Setting this value too high (e.g., >0.90) may greatly reduce recall but only the most similar scaffolds by k-mer usage will be compared.
- `--max_edges`: maximum number of edges per node in network clustering. The default is 6. Lowering this value will decrease connections between scaffolds and increase the reliability of the connections at the expense of false negative interactions. With values lower than 4 large viral genomes will likely be split between bins if highly fractionated.
- `--penalty_w`: penalty weight for Cohen's *d* distance calculations. The default is 0.20. Each Cohen's *d* value is compared to the respective preset threshold (see `--iter`). Any values greater than the threshold are given a penalty. Lower penalty weights will be less sensitive to coverage differences between scaffolds, and likewise greater penalty weights will be more strict.
- `--penalty_n`: maximum number of penalties for Cohen's *d* distance calculations. The default is 2. Each Cohen's *d* value is compared to the respective preset threshold (see `--iter`). Any values greater than the threshold are given a penalty. Increasing the maximum allowed penalties will be less sensitive to coverage differences between scaffolds, and likewise decreasing maximum allowed penalties will be more strict.
- `--mems`: refine bins with at least N members. The default is 4. Any potential bin with at least N members will be subject to bin refinement. Decreasing this value typically has minimal effects. Increasing this value may lead to greater contamination, but be less likely to refine (e.g., split apart) large viral genomes with many scaffolds.

Dereplication
(options to modify when using dereplication function)

- `--derep_only`: (bool, just use the flag) Only use dereplication function, skip binning. This is off by default. If set, binning will be skipped along with any binning-specific flags. Do not use this flag if you want to bin **and** dereplicate.
- `--method`: method of input scaffold dereplication. Options: none, longest, composite. The default is 'none', meaning dereplication by default is off. To turn on dereplication, set this flag to either 'longest' or 'composite'. Setting to 'longest' will dereplicate scaffolds and keep the longest representative. Setting to 'composite' will dereplicate identical scaffolds and combine overlapping scaffolds into composite sequences. 'Composite' is compatible with reads input or `--derep_only`.

- `-m`: minimum scaffold length to consider for dereplication. The default minimum is 1000bp (1kb) and cannot be reduced. Note that this value is different than `-l`.
- `--derep_id`: minimum percent similarity for dereplication. The default for 'composite' method is 0.99 and for 'longest' method is 0.97. The values are input as decimal points where 0.97 = 97%.
- `--frac`: minimum coverage in overlap percent for dereplication. The default for 'composite' method is 0.20 and for 'longest' method is 0.70. The values are input as decimal points where 0.70 = 70%.

(these typically do not need to be modified)
- `--mash_k`: equivalent to mash sketch -k. The default is 31.
- `--mash_s`: equivalent to mash sketch -s. The default is 1000.
- `--sens_ends`: free ends sensitivity, as fraction of scaffold length, to merging over regions of complexity; requires `--method composite`. The default is 0.10.
- `--sens_overlap`: alignment overlap sensitivity, as fraction of scaffold length, to merging over regions of complexity; requires `--method composite`. The default is 0.25.
- `--nucmer_c`: equivalent to nucmer -c. The default is 1000.
- `--nucmer_b`: equivalent to nucmer -b. The default is 1000.
- `--nucmer_g`: equivalent to nucmer -g. The default is 1000.
- `--nuc_split`: maximum fragmentations allowed per scaffold in nucmer alignment. The default is 3. Increasing this value will allow for more gapped alignments and decreasing will be more strict.

Contact

Please contact Kristopher Kieft (kieft@wisc.edu or GitHub Issues) with any questions, concerns or comments.

Thank you for using vRhyme!

```
_____
_

                    ## ## ## ##
                    ##        ##  ##          ##        ##    ## ## ##     #  ## ##
  ##          ##    ##        ##  ##            ##      ##  ##      ##    ##  ##          #
   ##      ##       ##      ##    ##              ## ##  ##      ##      ##  ## ## ##
    ##    ##        ## ####       ## ## ##        ##     ##      ##      ##  ##
     ## ##          ##    ##      ##      ##      ##     ##      ##      ##  ##
       ###          ##        ##  ##      ##      ##     ##      ##      ##    ## ## ##
_____
_
```

Copyright