**PhageMiner Documentation**

The explosion of whole genome sequence data has expedited the identification and genetic analyses of prophages. A handful of in silico prophage detection tools are currently available, among which Prophinder and PHAST are perhaps the most well-known [1-2]. These tools generally identify prophages based on some similarity to a database of previously known prophage genomes and, as a result, their performance is directly dependent on the size and composition of their database, and is hampered by the fact that prophages have variable genome sizes and often little sequence similarity to other prophages. Consequently, in order to ensure a thorough discovery of previously unidentified prophages, a manual curation of annotated genomes currently remains an indispensable step. This process generally involves generating a list containing the location and the annotation of each open reading frame (ORF) in the host genome, which is then manually reviewed for clusters of bacteriophage-related genes. The content, order and orientation of the ORFs within the putative prophage genome (and its flanking ORFs) are then carefully evaluated using a variety of different software. However, despite being able to discover more prophages than currently available automatic screening tools, this process is time-consuming, labour intensive, and not scalable [3-6].

To address these issues, we developed PhageMiner, a new bioinformatics tool that employs a user-supervised semi-automated approach in order to streamline the manual curation process for prophage sequence discovery. Notably, while PhageMiner significantly facilitates the manual curation process, it is not a fully automated pipeline and requires manual input by the user during key decisions, thus ensuring careful inspection of putative prophage clusters.

The PhageMiner pipeline consists of the following steps:

1) The bacterial genome of interest is annotated using the RAST annotation server (<http://rast.nmpdr.org>) in order to create an annotated GenBank file, which is then input into the PhageMiner Python script.

2) The location and the annotated name of each ORF in the host genome is retrieved from the annotated GenBank file and saved to a comma-separated value (CSV) file using the Biopython package (<http://biopython.org>).

3) A number of predefined user-adjustable phage-associated keywords are used to scan the CSV file generated in the previous step. Any ORF containing a matching string (e.g., “phage”, “lytic amidase”, "tail fiber protein", etc.) in its annotation name is deemed a ‘hit’.

4) An additional set of predefined user-adjustable keywords are used to discard any matching hits with annotation names that resemble phages but are not prophage genes (e.g. ‘macrophage’).

5) Using the mean shift clustering method in Scikit-Learn machine learning library (<https://scikit-learn.org>), the location of the remaining phage hits respective to each other and to the size of the host genome are used to identify clusters of bacteriophage-related genes. During this step, minimal manual inputs by the user are requested in order to ensure correct identification of prophage regions. If necessary, clustering can be repeated with a different sensitivity as redefined by the user, or alternatively, the coordinates corresponding to each suspected prophage region can be entered manually. The pipeline is aborted at this stage if no clusters of bacteriophage-related genes are detected or manually defined by the user.

6) Once clusters of bacteriophage-related genes are identified, PhageMiner creates various figures and tables related to each of the suspected prophage regions, the most important of which are: a schematic diagram of the coding regions; the location of the prophage region in the chromosome, including the flanking genes adjacent to the prophage region; the presence of any assembly gaps; and the nucleotide sequences of the ORFs in the cluster. If necessary, the number of flanking genes displayed in each figure can be manually adjusted.

7) Based on the decisions made by the user, the putative prophage genomes are either rejected or extracted as a separate GenBank file and categorised into three groups: full-length prophages, satellite prophages and unknown phage-related regions.

The source code of PhageMiner is deposited in GitHub at <https://github.com/RezaRezaeiJavan/PhageMiner>.

**References**

1. Lima-Mendez G, Van Helden J, Toussaint A, Leplae R. Prophinder: a computational tool for prophage prediction in prokaryotic genomes. *Bioinformatics*. 2008;24(6):863-865.

2. Zhou Y, Liang Y, Lynch K, Dennis J, Wishart D. PHAST: A Fast Phage Search Tool. *Nucleic Acids Res*. 2011;39(1):347-352.

3. Brueggemann AB, Harrold CL, Rezaei Javan R, van Tonder AJ, McDonnell AJ, Edwards BA. Pneumococcal prophages are diverse, but not without structure or history. *Sci. Rep.* 2017;7(1).

4. Crispim J, Dias R, Vidigal P, de Sousa M, da Silva C, Santana M *et al.* Screening and characterization of prophages in *Desulfovibrio* genomes. *Sci. Rep*. 2018;8(1).

5. Castillo D, Kauffman K, Hussain F, Kalatzis P, Rørbo N, Polz M *et al.* Widespread distribution of prophage-encoded virulence factors in marine *Vibrio* communities. *Sci. Rep.* 2018;8(1).

6. Fu Y, Wu Y, Yuan Y, Gao M. Prevalence and diversity analysis of candidate prophages to provide an understanding on their roles in *Bacillus Thuringiensis*. *Viruses*. 2019;11(4):388.