7-1-2016

PhamDB: A web-based application for building Phamerator databases

James G. Lamine
Calvin University

Randall J. DeJong
Calvin University

Serita M. Nelesen
Calvin University

Bioinformatics, 32(13), 2016, 2026–2028
doi: 10.1093/bioinformatics/btw106
Advance Access Publication Date: 26 February 2016
Applications Note

Genome analysis

PhamDB: a web-based application for building Phamerator databases

James G. Lamine[1], Randall J. DeJong[2] and Serita M. Nelesen[1,*]

[1]Department of Computer Science and [2]Department of Biology, Calvin College, Grand Rapids, MI, USA

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on September 14, 2015; revised on February 10, 2016; accepted on February 19, 2016

Abstract

Summary: PhamDB is a web application which creates databases of bacteriophage genes, grouped by gene similarity. It is backwards compatible with the existing Phamerator desktop software while providing an improved database creation workflow. Key features include a graphical user interface, validation of uploaded GenBank files, and abilities to import phages from existing databases, modify existing databases and queue multiple jobs.

Availability and implementation: Source code and installation instructions for Linux, Windows and Mac OSX are freely available at https://github.com/jglamine/phage. PhamDB is also distributed as a docker image which can be managed via Kitematic. This docker image contains the application and all third party software dependencies as a pre-configured system, and is freely available via the installation instructions provided.

Contact: snelesen@calvin.edu

1 Introduction

Phamerator is a tool developed for the purpose of comparing and analyzing bacteriophage (phage) genomes, allowing the visualization of the genomic mosaicism that characterize phage genomes (Cresawn *et al.*, 2011). Phamerator has been an important instrument of analysis for thousands of undergraduate students participating in SEA-PHAGES (Science Education Alliance – Phage Hunters Advancing Genomic and Evolutionary Sciences; Jordan *et al.*, 2014; http://seaphages.org). Students and faculty have used Phamerator in several publications (Hatfull, 2012, 2013; Pope *et al.*, 2011, 2014), including a recent comparison of a large number of phages (Pope *et al.*, 2015), greatly expanding knowledge of bacteriophage diversity and evolution.

Phamerator has largely been used with a single, shared database. However, it does have the ability to connect to other databases. As the SEA-PHAGES program has grown and diversified its approaches for isolating new bacteriophages, including using new bacterial hosts, there has been an increasing need for faculty and student researchers to create and manage custom Phamerator databases tailored to their bacterial host and unique research questions (Grose and Casjens, 2014; Merrill *et al.*, 2014).

Here we describe PhamDB, a web-based application which provides an improved workflow for the creation and management of Phamerator databases. A Phamerator database contains genome information as well as computed information about gene similarity. Pairwise comparisons of the predicted amino acid sequences of genes are used to assign them to families, termed 'phams'. Whole genome pairwise comparisons are also made at the nucleotide level. Computed phams and nucleotide comparisons are stored in a database, and Phamerator allows researchers to view data and visualizations from this database (Cresawn *et al.*, 2011).

Previously, databases for Phamerator were created using a series of scripts, posing significant barriers to entry to non-technical users. Creating databases this way required comfort with the Unix command line, the completion of a number of sequential steps, and responding to error messages. This required a significant investment of time, and troubleshooting sometimes required communication with the researchers who develop and maintain Phamerator. PhamDB eliminates this learning curve by providing an improved database creation workflow, including a web interface powered by a completely re-written database creation engine with a focus on input validation, early error detection, graceful error recovery, and actionable, human readable error reporting.

When creating a database, the user can upload GenBank files or select phages from existing databases. GenBank files are then checked for the necessary elements and format to be used with Phamerator, and

error messages for invalid files include a description of the problem and the line number where it occurs. The user can deselect uploaded phages or add additional phages before creating a database. Database creation operations are queued as jobs, with one job running at a time. This allows researchers to explore different research questions by queuing several new databases at once, rather than having to wait for the first database to be processed before creating another.

After database construction, users can remove or add phages and reconstruct the database. Instructions for connecting Phamerator to the correct server and databases are displayed for the user to copy and paste into Phamerator preferences. Existing databases can be imported as .sql files, and databases can be exported as .sql files for backup purposes.

2 Implementation

PhamDB is written in Python, allowing it to use the same bioinformatic libraries as Phamerator. Validation measures have determined that identical phams are constructed by PhamDB and the previously existing scripts method. PhamDB databases are stored in MySQL and use the same schema as Phamerator. The program is organized into three layers of abstraction—the web application front end, the web application back end, and the core database creation engine. The application can be installed by non-expert users on a server or on a single user machine. Note that we are not providing here a web hosted service, but rather PhamDB must be installed on a user's own hardware.

The front end is a combination of static HTML along with JavaScript and a JSON API to facilitate interactive user interface elements, such as uploading phages and live status information on running jobs. While the interface was designed with desktop users in mind, mobile first design practices were followed to ensure that the site is fully functional on devices with smaller screens, such as phones and tablets.

The web back end is built using the Flask micro framework. Metadata on PhamDB databases, jobs and uploaded GenBank files are stored in a MySQL database. Jobs are run on a separate worker process, and communication between the web back end and worker process is facilitated by RabbitMQ, an asynchronous message broker. The worker process calls the database creation engine and reports status back to the web back end.

2.1 Database creation engine

The database creation engine handles all PhamDB database operations. It provides operations which include creating a blank database, deleting a database, exporting a database as an SQL file, importing a database from an SQL file and building phams. The core of the database engine is the pham building function. Rather than providing separate operations for adding phages, removing phages and computing phams, these are combined into a single operation. This ensures that the results are the same regardless of what order phages are added or removed.

Specifically, the pham building operation is implemented as the rebuild function, whose arguments include a database name, list of GenBank files to add and optionally a list of phage ids to remove. This method reports status and errors through a callback method, making it possible to display live status information during long running jobs. Rather than exiting after the first error, the database creation engine is able to report all errors before aborting the operation. The rebuild method uses database transactions to ensure that in the event of invalid input or unexpected failure, the database is reverted to its original state. As a result, Phamerator databases never end up in an invalid intermediate state. Operations are either completely successful or no changes are made at all.

The rebuild operation runs as follows. First, uploaded GenBank files are validated. Then, the program checks for conflicts such as duplicate phages. Next, phams are calculated using the same method as recent versions of Phamerator database creation scripts, which involves two iterations of kClust (Hauser *et al.*, 2013; Older versions of the Phamerator scripts used CLUSTALW and BLAST to create phams (Cresawn *et al.*, 2011)). Finally, the new phams are uploaded to the database and, if the option is selected, rpsblast (http://www.ncbi.nlm.nih.gov/staff/tao/URLAPI/wwwblast/node20.html) is used to search the NCBI Conserved Domain Database for homology to each gene.

3 Conclusions

PhamDB provides an improved Phamerator database creation workflow that has been validated by the original Phamerator authors, and usability tests with undergraduate students and faculty confirmed a low barrier to entry. PhamDB also facilitates analyses that were previously unwieldy. For instance, phages that infect a newly studied bacterium in our laboratory have no close relatives in GenBank and their predicted amino acid sequences have few or no matches. When a Phamerator database was constructed containing these phages, shared features of genome architecture, gene function and amino acid sequence similarity were readily identified. Finally, PhamDB scales well, as we have successfully constructed databases of bacterial genomes.

Acknowledgements

We thank Steve Cresawn, Charlie Bowman and Graham Hatfull for encouragement, feedback and validation checks. We thank John Wertz for design inspiration and students in Computer Science 108 and Biology 224LN at Calvin College for usability testing. This article was presented at the 7th Annual SEA-PHAGES Symposium, HHMI Janelia Research Campus, Ashburn, VA, June 12-14, 2015.

Conflict of Interest: none declared.

References

Cresawn,S.G. *et al.* (2011) Phamerator: a bioinformatic tool for comparative bacteriophage genomics. *BMC Bioinf.*, **12**, 395.

Grose,J.H. and Casjens,S.R. (2014) Understanding the enormous diversity of bacteriophages: the tailed phages that infect the bacterial family Enterobacteriaceae. *Virology*, **468**, 421–443.

Hatfull,G.F. (2012) Complete genome sequences of 138 mycobacteriophages. *J. Virol.*, **86**, 2382–2384.

Hatfull,G.F. (2013) Complete genome sequences of 63 mycobacteriophages. *Genome Announcements*, **1**, e00847–e00813.

Hauser,M. *et al.* (2013) kClust: fast and sensitive clustering of large protein sequence databases. *BMC Bioinf.*, **14**, 248.

Jordan,T.C. *et al.* (2014) A broadly implementable research course in phage discovery and genomics for first-year undergraduate students. *MBio*, **5**, e01051–e01013.

Merrill,B.D. *et al.* (2014) Characterization of *Paenibacillus larvae* bacteriophages and their genomic relationships to firmicute bacteriophages. *BMC Genomics*, **15**, 745.

Pope,W.H. *et al.* (2011) Cluster K mycobacteriophages: insights into the evolutionary origins of mycobacteriophage TM4. *PloS One*, **6**, e26750.

Pope,W.H. *et al.* (2014) Cluster M mycobacteriophages Bongo, PegLeg, and Rey with unusually large repertoires of tRNA isotypes. *J. Virol.*, **88**, 2461–2480.

Pope,W.H. *et al.* (2015) Whole genome comparison of a large collection of mycobacteriophages reveals a continuum of phage genetic diversity. *eLife*, **4**, e06416.