VIROME
VIRAL INFORMATICS RESOURCE FOR METAGENOME EXPLORATION
A WEB-BASED APPLICATION FOR ANNOTATION AND ANALYSIS OF
VIRAL METAGENOMES

by

Jaysheel D. Bhavsar

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Master of Science in Computer Sciences

Fall 2014

© 2014 Jaysheel D. Bhavsar
All Rights Reserved
VIROME
VIRAL INFORMATICS RESOURCE FOR METAGENOME EXPLORATION
A WEB-BASED APPLICATION FOR ANNOTATION AND ANALYSIS OF
VIRAL METAGENOMES

by

Jaysheel D. Bhavsar

Approved: ____________________________________________________________

Li Liao, Ph.D.
Professor in charge of thesis on behalf of the Advisory Committee

Approved: ____________________________________________________________

Errol Lloyd, Ph.D.
Chairperson of the Department of Computer and Information Sciences

Approved: ____________________________________________________________

Babatunde A. Ogunnaike, Ph.D.
Dean of the College of Engineering

Approved: ____________________________________________________________

James G. Richards, Ph.D.
Vice Provost for Graduate and Professional Education
ACKNOWLEDGMENT

I sincerely thank Dr. Li Liao and Dr. K. E. Wommack for their guidance, patience and sharing their keen insight with me throughout this project. I am grateful to you Dr. K. E. Wommack, for teaching me so many aspects of carrying out fruitful research and for your infective perseverance and knowledge. Thank you Dr. S. Polson, D. Nasko, K. Thaper and S. Dhankar and Dr. K. E. Wommack et. al for their patience over my ever-changing crazy ideas of what VIROME should be. I thank the Faculty of the Computer and Information Sciences department, Dr. Cathy H. Wu and Dr. Blake Meyer for making these years of my education truly memorable.
TABLE OF CONTENTS

LIST OF TABLES ............................................................................................................. v
LIST OF FIGURES .......................................................................................................... vi
ABSTRACT ..................................................................................................................... vii

Chapter

1 INTRODUCTION ........................................................................................................ 1

2 VIROME BIOINFORMATICS ANALYSIS PIPELINE ............................................. 6

3 VIROME DATABASE ............................................................................................... 20

4 VIROME WEB-APPLICATION ................................................................................. 25
     Level 1: Home View ............................................................................................... 34
     Level 2: Browse View .......................................................................................... 35
     Level 3: Statistics View ....................................................................................... 38
     Level 4: Search View ........................................................................................... 41
     Level 5: Sequence Detail View ............................................................................ 43
     Comparinator ......................................................................................................... 46

5 VIROME SUBMISSION PORTAL ............................................................................. 48

6 SEQUENCE DETAIL THEN AND NOW: A CASE STUDY ................................... 50

7 CONCLUSION .......................................................................................................... 61

REFERENCES .............................................................................................................. 61

Appendix

VIROME BIOINFORMATICS ANALYSIS PIPELINE DOCUMENTATION ................................. 67
LIST OF TABLES

Table 1  Compare features between VIROME, MG-RAST and MetaVIR ..........3
Table 2  Feature comparison of various workflow managers.........................7
Table 3  Server and Environment organization of VIROME analysis data........23
Table 4  Feature comparison between sequence detail view version 1
         and version 2. .............................................................................53
LIST OF FIGURES

Figure 1  General overview of data flow through VIROME bioinformatics analysis pipeline, VIROME database and VIROME web application........................................................................................................5

Figure 2  VIROME bioinformatics analysis pipeline overview...............8

Figure 3  Detail VIROME bioinformatics analysis pipeline.....................14

Figure 4  VIROME database organization..............................................20

Figure 5  APP_INFO database schema................................................22

Figure 6  VIR_DATA database schema. Library table is not part of VIR_DATA its shown here for visual purpose linking VIR_DATA to APP_INFO database..........................................................24

Figure 7  Benchmark test of data communication time over three different protocol and various data types.................................................................29

Figure 8  The components of VIROME web-application, and their method of communication..................................................................................30

Figure 9  VIROME web-application data exploration pyramid. Graphic by freepik.com .........................................................................................33

Figure 10  Level 2: Browse view shows organization of ORF by biological and genetic data. Pie charts from left to right VIROME ORF categories, ORF type and Functional taxonomy at domain level .........................37

Figure 11  Distribution 1407 ORF classified as potential functional protein ORFs for RVA library .................................................................................39

Figure 12  Function 1 distribution of 1390 ORF that were homologous to GO database for RVA library ...........................................................................39

Figure 13  Family level view of RVA functional taxonomy statistics........40

Figure 14  List of all ORF that are part of Biological Process as annotated by Gene ontology database .................................................................42
Figure 15  BLAST function and results with VIROME web-application ............42

Figure 16  Read tab of sequence detail view. Raw read information on the left, and distribution of ORFs across a read/contig .........................44

Figure 17  ORF tab of sequence detail view ..........................................................45

Figure 18  VIROME submission portal and its interaction with existing ColdFusion classes, and database .........................................................49

Figure 19  Sequence detail view v1.0 ORF view. A tab for each ORF, plus a read tab, ORF details on top left, a heat map across all database on bottom left, static blast imager on top right. Detail top blast hits per database bottom right .................................................................54

Figure 20  Sequence detail view v1.0 ..........................................................55

Figure 21  Long reads in tab view with over 100 ORFs. Based on screen resolution, with over 5 ORFs per read tab labels are not visible and navigation difficult. Read tab is also not visible. Notice multiple scroll bars to the right ..........................56

Figure 22  Need to use both scroll bars to view heat map of ORF in Figure 21. Notice individual tabs are not visible anymore. The ORF overlay on read nucleotide is rendered unusable in this view, as you can’t see entire read in one screen ..........................................................57

Figure 23  Read Detail on top right. Interactive and dynamic blast imager top right. Blast hits per database, functional taxonomy and environmental metadata available in interactive and downloadable tables in the bottom half of the view ..........................................................58

Figure 24  Figure 24: Detail view of UNIREF100 table ..............................59

Figure 25  Version 2.0 ORF view. Notice ORF navigation bar at the bottom. ORF navigation bar, and tabs on top are always in view in the same place no matter how large or small ORF content gets ........................................60
ABSTRACT

Recent developments in sequencing technology have allowed metagenomic studies to become an increasingly common method for assessment of microbial communities and viral assemblages. Many existing resources for analysis of metagenomic data are designed for the needs of bacterial metagenomic studies, but few address issues specific to viruses. Most notable among these limitations is the lack of solutions for functional and taxonomic annotation of viral metagenomes.

Presented is the Viral Informatics Resource for Metagenome Exploration (VIROME) a web-based application for annotation and analysis of viral metagenomes. VIROME leverages a large database of known and environmental proteins to assess the potential identity of open reading frames (ORFs) within sequences. The analysis pipeline combines a variety of existing and novel resources to automatically form consensus annotations for a given ORF or sequence. The VIROME web interface allows the user to search metagenomes by BLAST, protein family, and functional/taxonomic annotations (among others). Interactive library statistics provide key information summaries at a glance, while enabling the user to directly browse the underlying data by “drilling-down” to increasingly specific levels of resolution. With over 23,028,372 ORFs across 109 libraries from diverse range of environments VIROME acts as a repository for published and contributed metagenome sequence data from viral assemblages
Chapter 1

INTRODUCTION

At the time of inception of the Viral Informatics Resource for Metagenome Exploration (VIROME) in 2005/06 there was a lack of bioinformatics analysis tools/pipelines designed specifically for viral metagenomes, nor were there any visualization tools available that could empower the user to identify and generate categorized data on demand. The analysis tools that did exist were largely adapted from whole genome or bacterial genome pipelines. The pipelines developed for whole genome or bacterial genome analysis largely searched for homologs against sequence databases of known organisms and bacterial reference genomes, this approach does not work well for viral metagenomes because large amount of homologs found in viral metagenomes and viral genomes are to unknown or hypothetical proteins (Wommack, Bhavsar et al. 2008). With the introduction of 454 sequencing and other next generation sequencing technologies it became possible to attain larger sequencing depth and reduce sequencing cost when compared to Sanger sequencing. However increase of short read sequencing data (~ 400 bases) did not translate to an increase in biological information. Short reads failed to find distant homologs and missed approximately 72% of Clusters of Orthologous Groups (COG) (Tatusov, Galperin et al. 2000) hits when compared to long reads (Wommack, Bhavsar et al. 2008).

Today, there are few metagenome analysis pipelines available, and even fewer analysis pipelines focusing on viral metagenomes. Two of the most popular metagenome analysis pipelines are MG-RAST (Meyer, Paarmann et al. 2008) and MetaVIR (Roux, Faubladier et al. 2011). MG-RAST, which is based on the rapid annotation using subsystem technology (RAST) genome annotation tool (Aziz, Bartels et al. 2008), has over 117
thousand metagenomes containing 404 billion base pairs of sequence data. Of the 117 thousand metagenomes over 16 thousand metagenomes are publicly available but only small fraction of the publicly accessible metagenomes are viral metagenomes. The core analysis and annotation of MG-RAST depends heavily on the SEED database which is comprises of mainly of bacterial and archaeal genomes (Overbeek, Disz et al. 2004). With little being known about viral communities and the strong sequence divergence and broad gene richness found in viral metagenomes indicate a tremendous genetic diversity (Kristensen, Mushegian et al. 2010) large amount of data is not considered while annotating viral metagenomes. Of all the features available one of the greatest assets of MG-RAST is the metagenome overview summary, which lists several key pieces of information such as:

- Sequence annotation distribution e.g., annotated protein, unknown protein, ribosomal RNA and unknown
- K-mer profile,
- A histogram of hit distribution across subject databases
- A distribution of sequences among top level COGs
- Enzyme groups within the Kyoto Encyclopedia of Genes and Genomes (KEGG) (Kanehisa and Goto 2000)

MetaVIR is specifically designed for the analysis of environmental viral communities. MetaVIR compares reads to complete viral genomes from the RefSeq database and performs phylogenetic analysis to explore viral diversity. However MetaVIR has a similar drawback as MG-RAST, by employing a single database of just viral genomes MetaVIR ignores a significant fraction of reads that often exhibit significant overlap to other viral sequences from different locations (Polson, Wilhelm et al. 2010). MetaVIR does have a great metagenome comparative tool, with help of Krona (Ondov, Bergman et al. 2011), makes it possible to compare metagenomes through
• K-mer frequencies
• Taxonomic compositions
• Recruitment plots
• Phylogenetic tree

With just 30% of sequence finding homology to a known functional gene a lot of sequence data is left out of the analysis and annotation of a viral metagenome sequences by both tools. Another feature missing from both these tools is the seamless navigation through various level of data granularity. In each case the primary navigation bar is the only form of navigation from one level of data exploration to next which is an impediment to more intuitive data exploration. Table 1 shows a comparison of features between VIROME, MG-RAST and MetaVIR

<table>
<thead>
<tr>
<th>Features</th>
<th>VIROME</th>
<th>MG-RAST</th>
<th>MetaVIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of subject database</td>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Env. Annotation</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Comparative tool</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MIMARKS standards</td>
<td>Yes</td>
<td>Yes</td>
<td>Unknown</td>
</tr>
<tr>
<td>Homology search tool</td>
<td>BLAST</td>
<td>BLAT</td>
<td>BLAST</td>
</tr>
<tr>
<td>Interactive interface</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️ ✔️</td>
</tr>
<tr>
<td>Download data</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
</tr>
<tr>
<td>Easy of Navigation</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
</tr>
</tbody>
</table>

What distinguishes VIROME from MetaVIR and MG-RAST is the rigorous bioinformatics analysis pipeline; and clean, simple, and intuitive interface, which facilitates natural flow of data exploration. The VIROME bioinformatics analysis pipeline uses six well-known curated databases and an extensive environmental peptide database in contrast to
the single database used by MetaVIR for homology search. The VIROME bioinformatics analysis pipeline also uses BLAST for better resolution and accurate alignment as appose to BLAT which is used by MG-RAST. VIROME adheres to data collection and reporting standards as outlined in the minimum information about a marker gene sequence (MIMARKS) and minimum information about (x) sequence (MIxS) specifications (Yilmaz, Kottmann et al. 2011). Inclusion of such metadata empowers VIROME to generate and display well-annotated results, especially in case of environmental annotation. Ensuring extensive annotation with controlled vocabulary also helps in downstream comparative analysis outside VIROME. Based on a rigorous bioinformatics pipeline, VIROME classifies each open reading frame (ORF) into following six classifications:

- Possible functional protein
- Unassigned functional protein
- Top hit viral
- Viral only hit
- Top hit microbial
- Microbial only hit

Additionally VIROME identifies and marks stable RNA genes, in particular transfer RNA (tRNA) and ribosomal RNA (rRNA) genes. Details of each of these categories are available in Chapter 2.

VIROME is a combination of three main components. The VIROME bioinformatics analysis pipeline is the core software engine that conducts bioinformatic analysis and annotation of the sequence data within a viral metagenome library. The VIROME database stores all the data generated by VIROME bioinformatics analysis pipeline and serves data on-demand to the VIROME web application. The VIROME web application is the interface
that end-user uses to interact with and explore viral metagenome data. An easy to use interface within intuitive flow of data exploration from one level to next, dynamic data display and scalability were some of the key principles that drove the design of VIROME bioinformatics analysis pipeline, database and web application. Figure 1 shows general overview of data flow through VIROME bioinformatics analysis pipeline, VIROME database and VIROME web application.

Figure 1  General overview of data flow through VIROME bioinformatics analysis pipeline, VIROME database and VIROME web application.
Chapter 2

VIROME BIOINFORMATICS ANALYSIS PIPELINE

With the advancement in high throughput sequencing, millions of reads are available for analysis in a short amount of time. At the same time pipelines employed to analyze large amount of data have become complex, requiring a large amount of computing power. One approach would be to add more computing power or employ a sophisticated system that can manage and analyze large amounts of data and efficiently and quickly while maintaining accuracy. VIROME bioinformatics analysis pipeline employs one such pipeline management system known as Ergatis (Orvis, Crabtree et al. 2010). Ergatis comprises of two tightly coupled systems. A management system known as Workflow operates behind the scenes and does bulk of the heavy lifting with respect to job management. Workflow is responsible for submitting jobs for execution to a compute cluster or a personal computer, keeping track of job progress, reports on job status, logging and other overhead related to job scheduling. The second part of Ergatis is the visible user interface, which allows a user to setup, execute and maintain a pipeline. The Ergatis front end also shows status of the pipelines as well as its individual components. What differentiates Ergatis from other workflow management systems such as Galaxy (Giardine, Rimer et al. 2005), Taverna (Wolstencroft, Haines et al. 2013) and Kepler (https://kepler-project.org/) is its ability to run multiple components in parallel and serial and its ability to run several scatter/gather processes in parallel of each other. Ergatis is also able to use a single instance of a pipeline to run multiple inputs. For example if five libraries need to be analyzed using the VIROME bioinformatics analysis pipeline five different instances of the pipeline would need to be run
in Galaxy, Taverna or Kepler. However, Ergatis could run all five libraries in parallel of each other using a single instance of the VIROME bioinformatics analysis pipeline. This is a useful feature as it provides a single location to monitor, debug and modify multiple libraries.

Ergatis’s workflow manager is written in JAVA and currently supports Sun Grid Engine or Open Grid Scheduler (http://gridscheduler.sourceforge.net/) along with standalone personal computer. Workflow manager has been previously deployed on HTCondor grid manager (http://research.cs.wisc.edu/htcondor/) but its not actively supported by the Ergatis team. It is possible to add additional support for other grid scheduler such as PBS/Tourque. The Ergatis front end is written in PHP (http://php.net/) and uses common gateway interface requests and responses (CGI) (https://metacpan.org/release/CGI) a Perl module (http://www.perl.org/).

Table 2 provides an overview of features between Ergatis, Galaxy, Taverna and Kepler.

<table>
<thead>
<tr>
<th>Features</th>
<th>Ergatis</th>
<th>Galaxy</th>
<th>Taverna</th>
<th>Kepler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid support</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Grid type</td>
<td>SGE/OGE</td>
<td>All(^1)</td>
<td>Most(^2)</td>
<td>Most(^2)</td>
</tr>
<tr>
<td>Interface</td>
<td>Web</td>
<td>Web</td>
<td>Standalone(^3)</td>
<td>Standalone(^3)</td>
</tr>
<tr>
<td>Open source</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Language</td>
<td>Perl-CGI and JAVA</td>
<td>Perl-CGI and C</td>
<td>JAVA</td>
<td>JAVA</td>
</tr>
<tr>
<td>Scatter/Gather</td>
<td>✔️ ✔️ ✔️ ✔️ ✔️</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
</tr>
<tr>
<td>Multi-input</td>
<td>✔️ ✔️ ✔️ ✔️ ✔️</td>
<td>✔️ ✔️ ✔️</td>
<td>✔️ ✔️</td>
<td>✔️ ✔️</td>
</tr>
</tbody>
</table>

\(^1\) Galaxy uses Distributed Resource Management Application API (DRMAA) to submit grid jobs (https://code.google.com/p/drmma-python/).

\(^2\) Complete list of grid engine support are not listed

\(^3\) Web interface is available for Taverna and Kepler via additional library as web service calls
Figure 2 depicts four main divisions of VIROME bioinformatics analysis pipeline,

- Quality control
- Open reading frame prediction
- Annotation
- Statistics and classification
Version 1.0 of the VIROME bioinformatics analysis pipeline contains thirty-two unique components of which four components are precompiled algorithms/commands such as BLAST (Altschul, Gish et al. 1990), CD-Hit (Li and Godzik 2006), tRNAscan (Lowe and Eddy 1997), MetaGene (Noguchi, Park et al. 2006) and a UNIX command called “concat”. The other twenty-seven components are Perl scripts and algorithms that were developed by the VIROME team. Due to parallel and serial combinations of these thirty-two components a typical end-to-end Ergatis pipeline consist of sixty-two components. Figure 3 shows all sixty-two components in order or execution.
Execute tRNAscan-SE

Prepare rRNA clean Sequence for DB insertion

Concatenate sequences into one file

Split into two files

Database upload

Prepare rRNA contaminated sequences for DB insertion

Concatenate sequences into one file

Split into two files

Database upload

Convert raw output into VIROME DB compatible tab file

Concatenate output into one file

Split into two files

Database upload

Prepare rRNA BLAST output for DB insertion

Concatenate output into one file

Split into two files

Database upload
The VIROME bioinformatics analysis pipeline v1.0 accepts FASTA, FASTQ or SFF inputs. All input sequences are filtered for minimum length of 300 bases for unassembled
reads or 600 bases for assembled contigs. In the case of FASTQ and SFF input average phred base quality score (Ewing, Hillier et al. 1998) must be greater than 20. In the case of 454 pyrosequencing data (typically in SFF files) artificial duplicates are removed using CD-HIT-454 (Li and Godzik 2006) at 98% identity. Ambiguous base such as N, R, and Y must be under 8% per sequence. The UniVec (http://www.ncbi.nlm.nih.gov/tools/vecscreen/univec) database is used to screen reads for the presence of contaminating vector sequences within metagenome sequence reads/contigs. A taxonomically diverse collection of ~30,000 ribosomal RNA (rRNA) genes (5S, 16S, 18S, and 23S) is used to detect the presence of ribosomal RNA homologs within sequence libraries rRNA contaminant using NCBI BLASTN v2.2.28+ (Altschul, Gish et al. 1990) with minimum expected value of $1 \times 10^{-75}$. The VIROME bioinformatics analysis pipeline v1.0 scans for transfer RNA (tRNA) using tRNAScan-SE v1.3.1 (Lowe and Eddy 1997).

The VIROME bioinformatics analysis pipeline v1.0 is an open reading frame (ORF) based pipeline; it uses predicted ORFs for functional and environmental annotation. Long ORFs are especially helpful in identifying candidate protein coding regions in a DNA sequence. The VIROME bioinformatics analysis pipeline v1.0 uses MetaGene (Noguchi, Park et al. 2006) to predict open reading frames. Because of the short read length of Illumina and Ion Torrent sequences users must assemble these data and submit contigs of $\geq 600$ bases, rather than single pass reads. Assembly drastically reduces the CPU time necessary for analysis and contigs greater than 600 bases are known to provide substantially more biological information than short reads (Wommack, Bhavsar et al. 2008).

The VIROME bioinformatics analysis pipeline v1.0 performs homology searches against a functional and an environmental sequence database using NCBI BLAST. Functional homologs are found using BLASTP search against a well-curated database called UniRef100-Plus (UniRef100P). UniRef100P is made up of seven de-replicated databases:
1. ACLAME: A classification of mobile genetic elements, which contains the collection and classification of mobile genetic elements (MGEs) from various sources, comprising all known phage genomes, plasmids and transposons (Leplae, Hebrant et al. 2004)

2. COG: Cluster of orthologous groups of proteins from the sequenced genomes of prokaryotes and unicellular eukaryotes and the construction of clusters of predicted orthologs (Tatusov, Galperin et al. 2000)

3. GO: Gene ontology provides a set of structured, controlled vocabularies for community use in annotating genes, gene products and sequences (Ashburner, Ball et al. 2000)

4. KEGG: Kyoto encyclopedia of genes and genomes is a knowledge base for systematic analysis of gene functions, linking genomic information with higher order functional information. The genomic information is stored in the GENES database, which is a collection of gene catalogs for all the completely sequenced genomes and some partial genomes with up-to-date annotation of gene functions (Kanehisa and Goto 2000)

5. PhageSEED: A tool kit for high-quality annotation of available phage sequences (http://www.phantome.org)

6. SEED: A toolkit to view and manipulate genomic data, automatically append annotations to it, and exchange information (Overbeek, Disz et al. 2004)

7. UniRef100: The UniRef (UniProt Reference Clusters) provide clustered sets of sequences from the UniProt Knowledgebase (UniProtKB) and selected UniProt Archive records to obtain complete coverage of sequence space at several resolutions while hiding redundant sequences (Suzek, Huang et al. 2007). UniRef100 contains peptides (>11 aa in length) gathered from the major sequence repositories (e.g., GenBank NR).

Environmental homologs are found using a BLASTP search against a well-curated database known as Metagenomes Online (MgOl) (S. Polson, K. E. Wommack et al, unpublished). Currently the MgOl database contains 258 libraries, of those, 159 libraries are viral, 88 prokaryotic and remaining 10 are eukaryotic or mixed organism libraries. MgOl contains 56,254,299 proteins for a total of 6,480,011,292 bases of amino acid sequences. MgOl is a manually annotated resource of predicted proteins identified in viral and microbial shotgun metagenomes. MgOl libraries are annotated with an abundance of sample metadata.
including sample provenance, geographical description, environmental parameters, sampling and preparation methodologies, and Environmental Ontology (ENVO) (http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=ENVO) terms. Samples are further categorized using MgO1 sample descriptors, easy to understand terms allowing samples to be easily grouped for comparison (http://metagenomesonline.org).

The VIROME bioinformatics analysis pipeline v1.0 classifies each sequence into one of seven categories:

1. **Possible functional protein:** These are sequences that have an homolog against UniRef100P database with expected value (e-value) of at least $1 \times 10^{-3}$ and the top homolog sequence is well curated i.e. BLAST hit description not unclassified, hypothetical protein or unknown.

2. **Unassigned protein:** These are sequences that have an homolog against UniRef100P database with e-value of at least $1 \times 10^{-3}$ but the top homolog sequence is not well curated i.e. BLAST hit description is unclassified, hypothetical protein or unknown.

3. **Top viral:** These are sequences that have a homolog against MgO1 database with e-value of at least $1 \times 10^{-3}$ and the top homolog sequence is in one of the MgO1 viral libraries.

4. **Top microbial:** These are sequences that have a homolog against MgO1 database with e-value of at least $1 \times 10^{-3}$ and the top homolog sequence is in one of the MgO1 prokaryotic or eukaryotic libraries.

5. **Viral only:** These are sequences that have a homolog against MgO1 database with e-value of at least $1 \times 10^{-3}$ and all homologs are to MgO1 viral libraries.

6. **Microbial only:** These are sequences that have a homolog against MgO1 database with e-value of at least $1 \times 10^{-3}$ and all homologs are to MgO1 prokaryotic or eukaryotic libraries.

7. **ORFans:** These are sequences that do not show homology to any UniRef100P or MgO1 peptide with an e-value of $1 \times 10^{-3}$ or does not have any homologs at all to either UniRef100P or MgO1 database.

Aside from the above seven categories, each sequence is scanned for tRNA sequence using tRNAscan-SE v1.3.1 and for rRNA contamination. Transfer RNA results are stored in a
MySQL (https://www.mysql.com) database (See chapter 3: VIROME database) and can be visualized using the VIROME web-application, sequence detail view. Sequences containing ribosomal RNA are removed from the sequence library in quality control stage of the VIROME bioinformatics analysis pipeline v1.0 and are stored in MySQL database. These sequences are displayed by selecting the rRNA pipe slice in the browse view of the VIROME web-application, see Chapter 4 for details.

The VIROME bioinformatics analysis pipeline v1.0 generates several statistical summaries for each library. These analyses cover a broad range of levels of biological organization from groups of thousands of sequences to a single ORF. The statistical summaries generated by VIROME bioinformatics analysis pipeline v1.0 are:

1. **Analysis overview**: Analysis overview shows the distribution of ORFs, in raw counts across UniRef100P, MgOl, both UniRef100P and MgOl and ORFans.

2. **VIROME Classification**: VIROME Classification shows percent distribution of ORFs across all seven categories described above (possible functional protein, unassigned protein, ACLAME, COG, GO, KEGG, SEED, PhageSEED) along with percent distribution of transfer RNA and ribosomal RNA.

3. **Detailed hierarchical distributions** for each functional databases ACLAME, COG, GO, KEGG, SEED and PhageSEED are also generated. Due to arbitrary depth of ACLAME and GO hierarchy a limit of six levels is imposed while viewing these statistics using the VIROME application.

4. **Taxonomy**: Hierarchical distribution of functional taxonomic lineage of all ORFs with top BLAST hit to UniRef100P subject database.

Using the environmental feature annotations of MgOl sequences and the VIROME informatics pipeline, it is possible to group viral metagenome peptides according to significant BLAST hits against MgOl peptides. The MgOl environmental feature annotations used for grouping viral metagenome peptides are: MgOl library id; library type; genesis; sphere; ecosystem; or extreme environment; and its physiochemical characteristics. The summaries of MgOl BLAST hit data for viral metagenome peptides are provided according
to each of these environmental features using a weighted scheme (Wommack, Bhavsar et al. 2012).

Along with the above statistics, the VIROME bioinformatics analysis pipeline v1.0 also generates histograms of read/contig lengths, ORF lengths and GC content per library. Raw data produced by the VIROME bioinformatics analysis pipeline v1.0 can be downloaded via the VIROME application. There are several how-to videos related to VIROME application are available online at http://www.virome.dbi.udel.edu

All the components of the VIROME bioinformatics analysis pipeline are written in Perl, using an object oriented programing paradigm. All Perl modules and scripts related to VIROME bioinformatics analysis pipeline and all scripts and modules required to setup an Ergatis instance are available in a private Github (https://github.com) repository URL: https://github.com/bjaysheel/virome_pipeline. Efforts are under way to create purely Perl based workflow, which can then be distributed as a virtual machine or as open-source package, for individuals to maintain their private VIROME bioinformatics analysis pipeline.
Chapter 3

VIROME DATABASE

All data generated by VIROME bioinformatics analysis pipeline is stored in a MySQL (https://www.mysql.com/) database. The VIROME database is organized into two major databases. The first database, known as APP_INFO, holds all the metadata related to each library and all VIROME registered user information. The second database is known as VIR_DATA where all data generated by the VIROME bioinformatics analysis pipeline is stored. Figure 4 shows organization of VIROME database.

Figure 4  VIROME database organization

All metadata for each library, registered user information and bookmarks of saved searches are stored in APP_INFO database. Figure 5 shows detail schema of APP_INFO.
The user table contains basic information for each user such as name, email address, institute affiliation, username and password. Passwords are encrypted using external ColdFusion (http://www.adobe.com/products/coldfusion-family.html) encryption as opposed to inherent MySQL password encryption this allows the flexibility to retrieve user password if required.

Groups table creates a grouping of users. A group can contain multiple users, and a user can belong to more than one group. By default when a new user is generated a new group is generated by default with the new user as its member. All libraries uploaded by the user will automatically be assigned this user’s respective group ID. A user can request access to a private library not assigned to them by sending an email to virome@virome.dbi.udel.edu and once verified a confirmation email will be sent.

The library table contains basic library information such as library name, project name, library description, environment, sequencing method, and whether or not respective library should be made public to the world. Each library is assigned a group ID, which establishes ownership and user access, however a library can only be assigned to one group ID.

Lib_summary table holds metadata such as library type (viral/microbial), NCBI accession (if any), data source (e.g.: CAMERA, NCBI), sequencing type, amplification, number of sites, site ID, genesis, sphere, ecosystem, physical chemical modifiers, physical substrate, Envo terms, geographic region, geographic place and geographic coordinates. A complete list of metadata types in the Lib_Summary table can be found in the VIROME database documentation (in digital format).
The library table links data in APP_INFO with data in VIR_DATA. VIR_DATA stores all sequence data, transfer RNA, ribosomal RNA, results of open reading frame prediction, BLAST results and summary output from the VIROME bioinformatics analysis pipeline. Due to budgetary constraints the VIROME database is not hosted on a dedicated database machine, which could be capable of responding to large number of queries and complex queries quickly and efficiently. For this reason a choice was made to distribute the
data stored in the VIR_DATA database into four different databases across four different physical machines. This architecture served to reduce load on single machine, and enable quick queries over a smaller database as opposed to one large table in a single database. This architecture is depicted in Figure 4; each instance of VIR_DATA has an identical structure over all four servers (Calliope, Polyhymnia, Thalia and Terpsichore). The environment metadata type in the library table is used as a category for dividing the data over each of the four servers; Table 3 shows each server and environments they represent.

Table 3 Server and Environment organization of VIROME analysis data.

<table>
<thead>
<tr>
<th>Server</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calliope</td>
<td>Organismal Substrate</td>
</tr>
<tr>
<td></td>
<td>Sediment</td>
</tr>
<tr>
<td>Polyhymnia</td>
<td>Soil</td>
</tr>
<tr>
<td>Thalia</td>
<td>Extreme</td>
</tr>
<tr>
<td></td>
<td>Solid Substrate</td>
</tr>
<tr>
<td>Terpsichore</td>
<td>Water</td>
</tr>
</tbody>
</table>

Figure 6 shows details of the VIR_DATA database and relationships between each table. The sequence table is an important table in VIR_DATA as it ties all tables together within VIR_DATA and serves to connect VIR_DATA with APP_INFO. The sequence table contains all sequences reads, contigs and ORFs. Type ID column in sequence table identifies if a sequence is an ORF, read/contig or ribosomal RNA, while the sequence_relationship table establishes the relationship between read/contig and ORFs. The blastp and blastn tables both hold all homolog results from VIROME bioinformatics analysis pipeline.
Figure 6  VIR_DATA database schema. Library table is not part of VIR_DATA as it is shown here for visual purpose linking VIR_DATA to APP_INFO database.
Chapter 4

VIROME WEB-APPLICATION

Modern day metagenomics and metagenome analysis pipelines generate data at such an enormous scale that text files, shell scripts and spreadsheets are not a viable option for extracting meaningful biological information from analysis pipeline results. A level of expertise is required to write and execute scripts, not to mention a higher probability of introducing an error while manually curating results and running scripts. The central challenge in designing the VIROME web-application was to create a tool capable of providing key statistical information and data summary with intuitive ease of use. A tool, which would empower its user to search, bin and categorize data in multiple ways with few simple clicks and give complete access to input raw data. The VIROME web-application also needed to adapt and respond to large data requests efficiently and quickly. Instead of providing static tables, static charts and lots of text, I hoped to develop a tool that was adaptive, dynamic, interactive and responsive, a tool that would make data exploration exciting and engaging. It took several iterations and evaluation of various technologies to get the right combination of server side and client side technology to present VIROME web-application in its current state. Chapter 6 Sequence Detail then and now a case study, highlights how advancement in sequencing technology generated an explosion of data which prompted a major change in the VIROME user interface. The new interface not only allows the VIROME web-application to adapt to large influx of data in the future, it also gives the VIROME web-application an ability to display various data types such as assembled long contigs, raw reads that makeup a contig and ORFs predicted from assembled long contigs. The new user interface can also adapt and display N-number of subject databases, only
requirement is that VIROME bioinformatics analysis pipeline add respective BLAST results in appropriate tables.

At the time of initial VIROME web-application development in late 2005 early 2006 the technology choices for generating a high quality and interactive web-application were limited. Some of the popular options for server side and client side technology were:

- JAVA servlet ([http://docs.oracle.com/javaee/6/tutorial/doc/bnafd.html](http://docs.oracle.com/javaee/6/tutorial/doc/bnafd.html)),
- Perl-CGI ([https://metacpan.org/release/CGI](https://metacpan.org/release/CGI))

Combination of these technologies helped produce usable tools but VIROME needed more than long page view, multiple scroll bars, tables and graphs broken in to several sub sections. In particular, representation of hierarchical relationships was difficult with these technologies as multiple graphs needed to be pre computed and stored on the server. This made the process inefficient, requiring large amount of disk space and a cluttered user-interface.

HTML5 ([https://en.wikipedia.org/wiki/HTML5](https://en.wikipedia.org/wiki/HTML5)) was years away; JQuery ([http://jquery.com/](http://jquery.com/)) wasn’t introduced until August of 2006 and in its early versions wasn’t as powerful as it is today. The tug-of-war for browser supremacy among various vendors hampered the ability to create a sophisticated and interactive web-application will work flawlessly across all browser platforms. Even today with most browsers converging towards a single standard when it comes to displaying data using HTML5 and cascading style sheets (CSS) version 3, some code redundancy is required to ensure that data is presented in similar fashion between Microsoft Internet Explorer ([http://windows.microsoft.com/en-us/internet-explorer](http://windows.microsoft.com/en-us/internet-explorer)), Apple
Safari (https://www.apple.com/safari) and Mozilla Firefox (https://www.mozilla.org/en-US/firefox). With frequent updates and new version of browsers being released developers are frequently required to test and modify existing applications just to make sure that their applications are functioning to their potential or restrict end user to a specific browser version to ensure the application and its features are working as intended. This takes time away from future enhancement and feature addition that would improve user experience.

In its current version, the VIROME web-application performs exactly the same, on any browser making it truly platform independent. To achieve this end, the VIROME web-application was built using combination of open source and proprietary software. The VIROME web-application was developed using Apache Flex (http://flex.apache.org), previously know as Adobe Flex (https://www.adobe.com/products/flex.html), an open source application framework for all end-user facing content display and interaction, along with an Adobe ColdFusion (http://www.adobe.com/products/coldfusion-family.html) application server as the middleware, and a MySQL database server in the backend to store all data generated by VIROME bioinformatics analysis pipeline (see Chapter 2 VIROME bioinformatics analysis pipeline, and Chapter 3 VIROME database for details).

The advantage of using Flex was that the compiled application is turned into an interactive Adobe Flash (http://www.adobe.com/products/flashruntimes.html) movie which can be deployed on all major browsers, desktops and even mobile device, and in each instance the look, feel, and behavior of the web application remains consistent. The only requirement to launch the VIROME web-application is Adobe Flash player, which according to Adobe’s statistics about 99% of all desktop browsers have Adobe Flash player installed (https://www.adobe.com/products/flashruntimes/statistics.html). This provided a significant

---

4 Adobe Flash player plugin must be installed (https://www.adobe.com/products/flashplayer.html)
advantage over other available technologies at the time. Flex allowed VIROME web-application development to focus on features and enhancements impacting the end-user experience directly versus ensuring web-application behavior was consistent across various browsers.

The VIROME web-application employs ColdFusion as its middleware to communicate between client (end-user) and data stored on the server. ColdFusion was chosen over other application servers such as JAVA or PHP for its efficiency in communicating with Flex applications using Action Message Format (AMF) when compared to traditional approach to communication such as web services or HTTP post. AMF is a binary format used to serialize object graphs such as ActionScript objects (an object-oriented programming language originally developed by Macromedia Inc.) and XML (eXtended Markup Language), or send messages between an Adobe Flash client and a remote service. AMF is optimized to keep the size of the AMF messages small compared to the Simple Object Access Protocol (SOAP)/XML used by web services. Figure 7 shows benchmark test using three different communication methods from a Flex application across various data types.
MySQL database server is the de facto choice for backend database. MySQL is widely used, well-supported and open source.

The VIROME web-application followed a modular development practice, which loosely followed model-view-controller design fundamentals. The VIROME web-application is a combination of three components. Each component is modular and performs very specific roles. The visual element is the first component, a web-interface developed in Flex. The middle-ware which process all data and requests from user interface developed in
ColdFusion is the second; and the backend MySQL database which stores all the data is the third. All code for the user interface and middleware data processing is modular and follows object oriented programming practices. Figure 8, shows division of VIROME web-application and communication interface between each component.

The components of VIROME web-application, and their method of communication

The web-interface is an event based, asynchronous system. The application listens and anticipates events; upon hearing or receiving such event appropriate actions are taken. The asynchronous nature of the web-interface allows the user to explore other aspects of the web-interface while appropriate actions are performed in the background. The visual components for user interaction, such as buttons, accordions, tabs, bar charts, pie charts, etc. are written in XML-based user interface markup language introduced by Macromedia known as MXML. Organization of data, communication between Flex and the ColdFusion application server using AMF or remote procedure call (RPC), and logic to determine a course of action based on an event are all written in ActionScript.

Visual components and ActionScript classes are modular, based on their specific roles; for example, all visual components related to the statistics view of the VIROME web interface and organized together under “stats” package/module. Similarly, all components related to the search view are organized under “search” package/module. All ActionScripts,
which support or create custom visual components, are organized as “com.component” modules. In addition to these internal modules there are two external modules, user and bookmarks for the VIROME web-interface. The user module is responsible for all end-user actions such as login, registration, password retrieval and tracking user log-ins. The bookmark module is responsible for keeping track of all saved searches, recalling past bookmarks or creating new ones. Communication between the main VIROME web-interface and the User and Bookmark modules is handled via an Interface known as VIROMELib. The role of this interface is quite simple; VIROMELib interface passes messages and sets internal parameters of respective modules.

Middle-ware does the heavy lifting in the VIROME web-application. The middleware is responsible for interpreting requests from the user, gathering all the data from the database, organizing all the data as expected by the web-interface and returning the requested data. There are six major classes that support various aspects of the web-interface, one utility class and an application level class, which sets global parameters related to VIROME web-application. The six major classes are:

- Library
- Search
- Statistics
- ORF
- READ
- Export

Individual functions of each class, and detailed commands are available in an independent interactive web form application called VIROME ColdFusion Component
Documentation. Organization of the MySQL database can be found in Chapter 3: VIROME Database.

The VIROME web-interface was designed for seamless and intuitive exploration of data generated by the VIROME bioinformatics analysis pipeline. Data generated by the VIROME bioinformatics analysis pipeline can be explored at five different levels with the assistance of various views provided by the web-interface. Figure 9, shows the pyramid of data exploration from the highest level, Level 1: Home view to the most granular level individual ORF’s BLAST results Level 5: Sequence detail view.
Figure 9  VIROME web-application data exploration pyramid. Graphic by freepik.com
Level 1:
Home View

The home view gives a bird’s eye view of all the data currently stored in the VIROME database and available for exploration via the web-application. Data is presented in two ways, a summary table and an interactive bar chart. The summary table organizes data based on environmental classification it shows total number of libraries, total number of reads/contigs, megabases of reads/contigs, mean read/contig length, total number of ORFs, megabases of ORFs and mean ORF length. The interactive bar chart also shows the distribution of ORFs across each environment, but instead of displaying raw ORF counts, ORF counts are log_{10} normalized for ease of comparison across environments. Clicking on any one of the bars shows a deeper resolution of ORF counts across all VIROME ORF categories: tRNA, rRNA, possible functional protein, unassigned functional protein, top viral hit, top microbial hit, viral only hits, microbial only hits and ORFans. See Chapter 2: VIROME bioinformatics analysis pipeline for details on VIROME ORF categories.
Level 2:
Browse View

The browse view lists all libraries and three pie charts per library that shows the composition of a library based on biological and genetic data obtained through homology searches for each ORF. The three pie charts shown in the browse view are:

1. VIROME ORF categories (See Chapter 2: VIROME Bioinformatics analysis pipeline)
2. ORF status such as missing 3’, missing 5’, complete ORFs or missing both 3’ and 5’ ends
3. Functional taxonomy distribution at domain level.

The browse view is organized using two visual component tabs. The tabs divides libraries based on public or private domain and accordions within each tab organizes each library by environment. The following use case illustrates the importance of the browse view and how the browse view can be used as a visual comparison feature between two libraries of similar characteristics.

Use case one: Consider two libraries as shown in Figure 10. No background information is assumed except that the two libraries or part of same project. The first column of Figure 10 shows basic library statistics such as number of raw reads/contigs, megabases of reads/contigs, mega bases of ORFs, and library coding percentage

$$Library\ Coding\ Percentage = \frac{ORF\ megabases}{read/contig\ megabases} * 100$$

Low coding percentage could potentially indicate short read lengths, or a problem during sequencing/library preparation as the ORF predictor identified very few potential proteins in the library.
The second column shows the distribution of ORFs based on their biological classification from homology search. This pie chart can provide a wealth of information. For example a high percentage of ribosomal RNAs could point to potential contamination of cellular DNA within the viral metagenome library, while a high percentage of unassigned functional proteins warrants further investigation as an abundance of ORFs found homologs to only unclassified or hypothetical proteins.

The third column shows ORF distribution of a library based on completeness. Given the short read length of next-generation sequencing technologies many libraries show a low concentration of complete ORFs across all environments and library types. Fourth column shows the distribution of ORFs based on functional taxonomy at the domain level based on homology search.

Presentation of library data at Level 2 also facilitates a quick comparison between two libraries. The two libraries shown in Figure 10 RVA (top library) and RVB both have similar coding percentage and distribution of ORF completeness. However comparing VIROME ORF category distribution points to a potential contamination of the RVB library by cellular DNA as the ribosomal RNA pie slice is quite large.
Figure 10  Level 2: Browse view shows organization of ORF by biological and genetic data. Pie charts from left to right VIROME ORF categories, ORF type and Functional taxonomy at domain level
Level 3:
Statistics View

Similar to Level 2: Browse view, the statistics view also shows compositional features of a library using biological and genetic data obtained through homology search of each ORF. Unlike the browse view, the statistics view provides an in depth look at the distribution of ORFs within a library with the help of interactive bar charts and pie charts. The interactive charts provide a good way to explore hierarchical data such as functional taxonomy levels from domain level to species level or hierarchical structure of Kyoto Encyclopedia of Genes and Genomes pathway (KEGG). The data used to generate all charts in Statistics view and any other charts presented in VIROME web-application can be downloaded for further independent analysis outside of VIROME.

Use case two: Continuing with the same two libraries as presented in Use case one. From figure 10 we know that 1,407 ORFs, approximately 65% of ORFs for RVA library, have a homolog to the functional database UNIREF100P. Figure 11 of the statistics view shows a more detailed distribution of these 1,407 ORFs across various functional databases, and Figure 12 further explores the distribution of 1,390 ORF that have homologs to the Gene Ontology database. One could use this feature of the statistics view to bin ORFs, in this case get all ORFs that are classified as facilitators of biological process according to the Gene ontology database and download these ORFs for further independent analysis such as phylogenetic tree construction as demonstrated by Schmidt and colleagues (Schmidt, Sakowski et al. 2014). One could also use the functional taxonomy chart as show in Figure 13 to download all ORFs that had a homology to Tymoviridae family of functional taxonomy for fragment recruitment analysis as demonstrated by Douglas B. Rusch (Rusch, Halpern et al. 2007).
Figure 11  Distribution 1407 ORF classified as potential functional protein ORFs for RVA library

Figure 12  Function 1 distribution of 1390 ORF that were homologous to GO database for RVA library
Figure 13  Family level view of RVA functional taxonomy statistics
The search view provides an ability to search the VIROME database across various biological and environmental criteria and list individual ORFs. It is the search view that facilitates downloads of ORFs that are binned according to various biological and environmental criteria for subsequent analysis outside of VIROME see Figure 14.

The search view not only allows a user to search the VIROME database, but also BLAST their target sequence against any combination of VIROME libraries, from within VIROME web-application. Should the query sequence find any homologs to sequences in the VIROME database the search view provides the ability to explore and get details about the subject sequence all from within the VIROME web-application. Figure 15 shows the BLAST function of the VIROME web-application. Due to budgetary constraints and resources currently only one query sequence can be submitted to BLAST against VIROME database. To BLAST multiple sequences users are encouraged to submit a request to virome@virome.dbi.udel.edu.
### Figure 14  List of all ORF that are part of Biological Process as annotated by Gene ontology database

<table>
<thead>
<tr>
<th>Sequence Name</th>
<th>Hit Name/Accession</th>
<th>Description</th>
<th>Evalue</th>
<th>% QRY Coverage</th>
<th>% Identity</th>
<th>Organism</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAPUN11.1_146.1</td>
<td>Q00N9H</td>
<td>Reptile large subunit</td>
<td>1.0E-14</td>
<td>95.56 %</td>
<td>95.6</td>
<td>Reptile mild melito virus</td>
</tr>
<tr>
<td>RAPUN11.3_155.1</td>
<td>P25998</td>
<td>Reptile small subunit</td>
<td>1.0E-11</td>
<td>98.85 %</td>
<td>94.4</td>
<td>Reptile mild melito virus</td>
</tr>
<tr>
<td>RAPUN11.4_155.1</td>
<td>Q00N9H</td>
<td>Reptile large subunit</td>
<td>2.0E-6</td>
<td>77.78 %</td>
<td>51.7</td>
<td>Reptile mild melito virus</td>
</tr>
<tr>
<td>RAPUN11.6_151.1</td>
<td>P25998</td>
<td>Movement protein</td>
<td>3.0E-11</td>
<td>94.46 %</td>
<td>96.9</td>
<td>Reptile mild melito virus</td>
</tr>
<tr>
<td>RAPUN11.7_151.1</td>
<td>Q4RR96</td>
<td>1284 protein</td>
<td>4.0E-10</td>
<td>84.43 %</td>
<td>78.2</td>
<td>Reptile mild melito virus</td>
</tr>
<tr>
<td>RAPUN11.8_150.1</td>
<td>Q4RR96</td>
<td>Reptile large subunit</td>
<td>5.0E-13</td>
<td>92.86 %</td>
<td>90.3</td>
<td>Reptile mild melito virus</td>
</tr>
<tr>
<td>RAPUN11.11_254.1</td>
<td>A2K7T5</td>
<td>Reptile small subunit</td>
<td>2.0E-6</td>
<td>99.86 %</td>
<td>89.4</td>
<td>Tobacco mosaic virus</td>
</tr>
<tr>
<td>RAPUN11.12_168.1</td>
<td>Q4RR96</td>
<td>Movement protein</td>
<td>9.0E-16</td>
<td>79.83 %</td>
<td>73.0</td>
<td>Tobacco mosaic virus</td>
</tr>
<tr>
<td>RAPUN11.13_168.1</td>
<td>Q4RR96</td>
<td>Movement protein</td>
<td>3.0E-11</td>
<td>94.84 %</td>
<td>81.2</td>
<td>Citrus budworm-associated virus</td>
</tr>
<tr>
<td>RAPUN11.20_119.1</td>
<td>P0B820</td>
<td>Reptile-associated protein</td>
<td>2.0E-6</td>
<td>91.67 %</td>
<td>50.0</td>
<td>Citrus budworm-associated virus</td>
</tr>
<tr>
<td>RAPUN11.21_126.1</td>
<td>Q4RR96</td>
<td>Reptile large subunit</td>
<td>2.0E-10</td>
<td>98.40 %</td>
<td>76.7</td>
<td>Reptile mild melito virus</td>
</tr>
<tr>
<td>RAPUN11.34_162.1</td>
<td>P25998</td>
<td>Reptile small subunit</td>
<td>1.0E-19</td>
<td>94.23 %</td>
<td>96.8</td>
<td>Reptile mild melito virus</td>
</tr>
<tr>
<td>RAPUN11.29_175.1</td>
<td>Q4RR96</td>
<td>1284 protein</td>
<td>5.0E-25</td>
<td>96.43 %</td>
<td>100.0</td>
<td>Reptile mild melito virus</td>
</tr>
<tr>
<td>RAPUN11.31_175.1</td>
<td>Q4RR96</td>
<td>Movement protein</td>
<td>2.0E-11</td>
<td>90.27 %</td>
<td>79.3</td>
<td>Reptile mild melito virus</td>
</tr>
<tr>
<td>RAPUN11.32_175.1</td>
<td>Q4RR96</td>
<td>Movement protein</td>
<td>4.0E-9</td>
<td>94.12 %</td>
<td>91.1</td>
<td>Reptile mild melito virus</td>
</tr>
</tbody>
</table>

### Figure 15  BLAST function and results with VIROME web-application
Level 5: Sequence Detail View

The sequence detail view shows the finer details of each read/contig and ORF produced by the VIROME bioinformatics analysis pipeline. The sequence detail view is organized into two tabs, the read/contig tab and the ORF tab. The read tab contains raw read or contig information such as the number of ORFs, the number of transfer RNA(s), read length and the number of ORF(s) predicted. The read tab also includes an image that overlays all ORF(s) predicted along with location of transfer RNA as reported by tRNAscan-SE. The image shown in Figure 16 illustrates the distribution and coverage of ORFs across a read/contig and if the read contains any transfer RNA sequences. The ORFs in the image are color coded to indicate the completeness of the ORF:

- Green: An ORF with start and stop codon
- Orange: An ORF missing stop codon
- Blue: An ORF missing start codon
- Pink: An ORF missing both start and stop codons

The arrow on each ORF indicates direction of the ORF. Along with the image, the read tab also presents table of top BLAST hits against functional database UNIREF100P, the functional taxonomy lineage of ORFs that had homologs to the functional database, top BLAST hit against environmental database MgOl and an environmental summary of each ORF that had homolog to the environmental database. The information displayed in these tables can help infer functional taxonomy lineage, and environmental summary of a read/contig.
Similar to the read tab, the ORF tab shows an image of top BLAST hits across all functional databases and environmental database. The BLAST imager is a great way to quickly visualize the coverage and quality of the BLAST homologs. The ORF tab also shows a heat map of e-value of the top ten BLAST hits across all functional databases and the environmental database. Subtle change in the hit map across databases indicates strong homology against subject database as opposed to an abrupt change as shown in Figure 17. Selecting any square of a heat map will show a detailed tab delimited BLAST report of the top 50 hits as reported by the VIROME bioinformatics analysis pipeline.
Figure 17  ORF tab of sequence detail view
Comparinator

With advances in sequencing technology, and varying nature of the sample to sequence pipeline for each library is it difficult to predict all type of analysis and summarization needs users may desire. For this reason VIROME emphasizes the ability to put data in the hands of the user in a format that can easily be transferred to other analysis platforms. Aside from the data exploration levels, the VIROME web-application also facilitates comparative analysis across multiple libraries using Quantitative Insights Into Microbial Ecology (QIIME) package (Caporaso, Kuczynski et al. 2010). QIIME is an open source software package for comparison and analysis of microbial communities, primarily based on high-throughput amplicon sequencing data generated on a variety of platforms, but also supports analysis of other types of data such as shotgun metagenomic data. Data required for comparative analysis using QIIME package can be downloaded from the VIROME web-interface using a compare feature known as the Comparinator. The Comparinator allows an end-user to download biological observation matrix format files (McDonald, Clemente et al. 2012) for two or more libraries which can be directly plugged into various QIIME scripts. The Comparinator currently supports the following metrics for library comparison:

1. Functional taxonomy
2. A classification of mobile genetic elements (ACLAME) hierarchy
3. Clusters of Orthologous Groups (COG) hierarchy
4. Kyoto Encyclopedia of Genes and Genomes (KEGG) hierarchy
5. SEED hierarchy

Flex and ColdFusion code related to VIROME web-application are available in a private Github repository URL: https://github.com/bjaysheel/virome_app. Now that Apache Flex support for PHP application server is stable and has matured to support large data driven
application such as VIROME web-application, efforts are under way to convert all ColdFusion classes into PHP classes and modules. Moving to PHP as a middleware application server will remove dependency on proprietary software making all components of VIROME web-application open source. This will permit us to distribute VIROME web-application as a virtual machine or as open-source package that can be used as private mirrored application.
Chapter 5

VIROME SUBMISSION PORTAL

The VIROME submission portal allows users to submit their libraries for analysis via the VIROME bioinformatics analysis pipeline and subsequently explore analyzed data using the VIROME web-application. To ensure that data submitted is consistent across all libraries, and that the VIROME data is in accordance with minimum information about any sequence (MiXS) standard (Yilmaz, Kottmann et al. 2011) the VIROME data submission portal requires that all fields be completed for a successful library submission. By enforcing strict policy for collecting both primitive and derived metadata, the VIROME application provides a more complete disclosure of metadata that can be pertinent to comparative analyses between environmental metagenomes. This is especially important when considering that over 70% viral metagenome ORFs find homologs to only peptides within the environmental database.

The VIROME submission portal was developed using HTML5 and PHP. The submission portal interacts with ColdFusion server and leverages methods written for the VIROME web-application to allow end users to add, update and delete their libraries, see Figure 18. Each library goes through three stages when submitted to the VIROME submission portal:

1. Stand by: Library has been submitted once the data has been manually curated it will be queued for processing via the VIROME bioinformatics analysis pipeline

2. Processing: Library is being analyzed via VIROME bioinformatics analysis pipeline
3. Complete: Library has been analyzed via VIROME bioinformatics analysis pipeline, and is ready for data exploration via VIROME web-application. Manual intervention is required as a library progresses through each stage, to ensure data integrity and the quality of data. The VIROME submission portal URL: http://virome.dbi.udel.edu/submission/index.php, shares a single login between VIROME web application and VIROME submission portal.

Figure 18 VIROME submission portal and its interaction with existing ColdFusion classes, and database
Chapter 6

SEQUENCE DETAIL THEN AND NOW: A CASE STUDY

The Viral Informatics Resource for Metagenome Exploration (VIROME) is a web-based application specializing in analysis of viral metagenomes. Among the various modules that make up the VIROME web application, the sequence detail is an important module as it allows the user to dig deep into granular details of each ORF. At any given point a user is only one, at the most two clicks away from viewing: the distribution of ORF across an entire read/contig or whole genome; a heat map of e-value distribution across all databases; a detailed blast table highlighting the top functional hits (if any) and the top blast hits based on best e-value; and a quick view of top blast hits across all databases along with its functional lineage (if any). These views also allow a user to download data in various formats at the level of individual ORF, read/contig or whole genome level.

At the inception of VIROME, next-generation sequencing technology was in its infancy. Next-generation sequencing technologies such as 454-pyrosequencing dramatically reduced the cost of sequencing but the quality of information gathered, compared to Sanger sequencing, was less appropriate for phylogenetic and functional characterization of metagenomes (Wommack, Bhavsar et al. 2008). Therefore a conscious decision was made to design the sequence detail modules based on ORFs called from Sanger reads of approximate length of 750 bases.

The average number of ORF (> 50 bases) called from Sanger environmental metagenome libraries were ~2 and maximum of 5. Based on this finding a tabbed user interface was chosen see Figure 19 and 20. This interface quickly allowed a user to see a
number of surrounding ORFs, navigate to any one of the neighboring ORFs with a single click, and give access to read level information with a single click.

At the read level, a static image (designed similar to the BLAST imager view for NCBI-BLAST) provided a view of the distribution of ORFs across an entire read, as well as the position of tRNA(s) if present. Top BLAST hits based on e-value across all ORFs per database was displayed in a static table, which gave at quick glance a look at type and quality of BLAST hits. A functional taxonomy table across all ORFs provided a naive consensus for read, at the same time showing taxonomic similarities or differences in each ORFs. An entire read in nucleotide bases was displayed for convenience to copy and paste for further analysis.

At the ORF level, a static blast imager showed top BLAST hits across all databases. Top BLAST hits based on e-value across all databases along with functional lineage provides some indication of the possible taxonomic origin of the sequence. A heat map of the top ten blast hits across all databases shows at a glance similarities between BLAST hits, and how quickly quality of hits changes. Upon clicking on a specific region of the heat map a detailed BLAST report is presented where the top functional and top blast hits are highlighted along with functional taxonomy or environmental metadata. Reads in nucleotide format with the ORF sequence highlighted is displayed to give a visual representation of how an ORF spans over a given read, see Figure 19 and 20.

With the advancement in sequencing and improved assembly algorithms it is now possible to get better quality information from next-generation sequences. When displaying information from long read contigs from assembled metagenome or viral whole genomes the tabbed view became cumbersome to the point of unusable, see Figure 21 and 22. Labels on each of the tabs were unreadable, navigation to a desired ORFs was difficult involving, multiple scroll bars which made viewing the relevant information very difficult and
confusing. It became apparent that the user interface for the sequence detail view needed to be overhauled to account for long read data.

Thus the vision for version 2.0 was to present the same information in a more efficient manner, with interactive images and tables to allow a user to download information right from the sequence detail view.

Instead of multi tab view, there is now only one tab per feature (currently only the read and ORF tab). The static fixed length BLAST image was replaced by a dynamic, interactive, native BLAST imager, which scales based on read length that allowed for an even spaced ORF overlay. Dynamic tables that can be sorted by any given column replaced static simple tables. The entire content of the tables can now be downloaded in a tab-delimited file, for further analysis. Instead of displaying all bases of read sequences only the first few are displayed, and a mechanism is in place to display and download the entire read sequence see Figure 23 and 24.

There is now a single tab for ORF. Each ORF is displayed in a sliding window format. Navigation to an ORF is possible via “Next” and “Previous” buttons or by simply entering desired ORF number in the field provided. This eliminates the clutter of tabs as occurred in the previous version, while at the same time keeping navigation to neighboring ORFs simple and quick. A dynamic, high quality, native BLAST imager replaced the static BLAST imager from Figure 19 and 20. With the inclusion of a dynamic, high quality BLAST imager in the read tab, which provides a clear view of the ORFs distribution across the read, the long string of nucleotide read with an ORF overlay has been omitted from this version. The heat map of the top 10 BLAST hits across all databases, and view of top BLAST hits per database along with functional or environmental metadata lineage remains unchanged from the previous version, see Figure 25.
In conclusion with the overhaul of sequence detail view; information presented is more clear, concentrated, and easily accessible. The new sequence detail view provides multiple ways to navigate to a desired ORF, sort data in any way desired, make use of entire viewable screen, reduce the amount of scrolling, all this while still providing same high quality information. This version also provides a mechanism to download and bin data at an individual read level across all BLAST databases, taxonomy and environmental metadata for further analysis. Below is a table comparing two versions of the sequence detail view.

Table 4 Feature comparison between sequence detail view version 1 and version 2.

<table>
<thead>
<tr>
<th>Features / Version</th>
<th>v1.0</th>
<th>v2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive high quality image</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Interactive tables</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Download data</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Quickly identify ORF type</td>
<td>Yes (via tab color)</td>
<td>No (available via text only)</td>
</tr>
<tr>
<td>Easy of navigation</td>
<td>Yes (limited number of ORFs)</td>
<td>Yes</td>
</tr>
<tr>
<td>Scalability</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

There are few features that are missing from new version, most notable being the colored ORF tab based on ORF type (complete, missing stop codon, missing start codon or missing start and stop codons) and ORF amino acid sequence string from ORF view. Both of which were determined to be a low priority feature, and will be added in subsequent version release of VIROME web application.
Figure 19  Sequence detail view v1.0 ORF view. A tab for each ORF, plus a read tab, ORF details on top left, a heat map across all database on bottom left, static blast imager on top right. Detail top blast hits per database bottom right.
Figure 20  Sequence detail view v1.0
Figure 21  Long reads in tab view with over 100 ORFs. Based on screen resolution, with over 5 ORFs per read tab labels are not visible and navigation difficult. Read tab is also not visible. Notice multiple scroll bars to the right.
Figure 22  Need to use both scroll bars to view heat map of ORF in Figure 21. Notice individual tabs are not visible anymore. The ORF overlay on read nucleotide is rendered unusable in this view, as you can’t see entire read in one screen
Figure 23  Read Detail on top right. Interactive and dynamic blast imager top right. Blast hits per database, functional taxonomy and environmental metadata available in interactive and downloadable tables in the bottom half of the view.
Figure 24  Figure 1: Detail view of UNIREF100 table.
Figure 25  Version 2.0 ORF view. Notice ORF navigation bar at the bottom. ORF navigation bar, and tabs on top are always in view in the same place no matter how large or small ORF content gets.
Chapter 7

CONCLUSION

We have successfully created a framework, consisting of an end-to-end bioinformatics analysis pipeline coupled with a rich and dynamic web user-interface that is platform independent and a portal for data submission, all of which is backed by a single storage structure that shares data across all three applications. This framework addresses a need in the research community, namely a complete bioinformatics analysis pipeline for environmental viral metagenomes. This analysis pipeline leverages all the information available not just homology to known database(s) along with an adaptive, interactive and intuitive user interface that engages user in data exploration and puts the control of data in users hand.

The process begins with the VIROME submission portal, where a strict policy of collecting metadata insures that data is consistent across all libraries and that a controlled vocabulary is used, which can be leveraged downstream for exhaustive comparative analysis. The VIROME bioinformatics analysis pipeline runs all incoming data through extensive quality checks and filters. All sequences that pass various quality measures are analyzed using a suite of open source tools and custom scripts. The VIROME bioinformatics analysis pipeline leverages seven well-known databases and a comprehensive environmental metagenomes database to annotate ORFs based on their biological and genetic data obtained from homology search. This approach of using both environmental and known databases allows VIROME users to annotate a library using majority of the sequence data not just the 30% of sequences that find homology with known databases. Finally, the VIROME web-
application puts the user in the driver’s seat with dynamic, interactive, intuitive and engaging user-interface where an intuitive flow of data allows users to easily explore and download data for further analysis outside of VIROME using a powerful quantitative comparison and analysis tool known as QIIME.

Ongoing efforts directed towards making the VIROME framework independent of third party tools such as ColdFusion and Ergatis, would minimize dependencies on other tools and modules. This would enable VIROME framework to be installed easily and run locally with or without a computing grid and empower users to analyze and explore data in a seamless fashion at a push of button.

By providing a complete framework to annotate and explore environmental viral metagenomes freely to the community we have created an important resource. VIROME currently has over 250 unique active users across 10 countries. VIROME framework has analyzed over 300 libraries and seven thousand megabases of sequence data. Fifteen different peer review articles have cited VIROME.
REFERENCES


APPENDIX
VIROME BIOINFORMATICS ANALYSIS PIPELINE DOCUMENTATION

ALLFXNALDBBREAK.PL

NAME
   AllFxnalDBBreak.pl

SYNOPSIS
   USAGE: AllFxnalDBBreak.pl
   --input=/input/filename
   --outdir=/path/to/output/dir
   --server=server-name
   --env=dbi
   --library=libraryId
   [  
     --log=/path/to/logfile
     --debug=N
   ]

OPTIONS
   --input, -i OPTIONAL if server and library is defined Tab delimited file of library and its metadata information

   --outdir, -o REQUIRED Complete path to output dir.

   --server, -s OPTIONAL if input and/or server is defined. Server/database name where library data is stored

   --library, -l OPTIONAL if input and/or server is defined. Specific library ID

   --env, -e REQUIRED Specific environment where this script is executed. Based on these values DB connection and file locations are set.

       Currently supports
           igs
           dbi
           ageek or
           test

   --log, -l OPTIONAL Log file location

   --debug, -d OPTIONAL Debug level

   --help, -h This help message

DESCRIPTION
Create XML document that contains information to draw all functional count chart. i.e.: number of hits in KEGG, SEED, UNIREF100P, COG, and ACLAME

**INPUT**

The input is defined with --server, --library. Or by an input file which is a tab delimited text file with libraryId, prefix and library name

e.g. of input file (tab delimited):
#LibraryID NAME Prefix SERVER
1  ABC  ABC_NAME

**OUTPUT**

A well defined XML file with number of ORFs with significant hit to functional databases such as
SEED
KEGG
COG
ACLAME
GO and
PhageSEED

A secondary xml document with IDDOC prefix is created which store individual IDs for each ORF. IDDOC and XMLDOC files are linked via TAG_##

e.g.: IDDOC
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <TAG_1
    IDLIST="23125,23133,23142,23145,23169,23187,23212,23218,23225,..." />
  <TAG_2
    IDLIST="23149,23188,23226,23229,23255,23381,23383,23533,23541,235
    59,23583,23630,23654,23666,23672,23743" />
</root>

78
<TAG_3
IDLIST="23132,23142,23145,23149,23154,23165,23166,23188,23190,23196, ...,
" />
<TAG_4
IDLIST="23145,23188,23255,23288,23319,23336,23337,23381,23389,23416,23533,23541,23630,23666,23672,23743" />
<TAG_5 IDLIST="23288,23381" />
</root>

EXAMPLE
AllFxnalDBBreak.pl --server=calliope --env=dbi --library=31
BLAST-RESULT-PREP.PL

NAME

blast-result-prep.pl - prepare blast btab file for MySQL upload

SYNOPSIS

USAGE: blast-result-prep.pl
--input=/path/to/blast/btab/output
--liblist=/library/list/file/from/db-load-library
--lookupDir=/dir/where/mldbm/lookup/files/are
[  
  --log=/path/to/logfile  
  --debug=N
]

OPTIONS

--input, -i REQUIRED The full path to blast btab output

--liblist, -ll REQUIRED Library list file, and output of db-load-library.

--lookupDir, -ld REQUIRED Dir where all lookup files are stored.

--log, -l OPTIONAL Log file location

--debug, -d OPTIONAL Debug level

--help, -h This help message

DESCRIPTION

Given raw tab delimited BLAST results, this script will rearrange, edit and add information to each input row so that each row is in sync with the columns of blastp table of VIROME database.

Based on subject database a ranking integer is added to identify database hierarchy

INPUT

The input to this is defined using the --input. This should point to the BLAST btab output

Fields that are load are

1  query_name
2  query_length
3  algorithm
4  database_name
5  hit_name
6  qry_start
7  qry_end
8  hit_start
9  hit_end
percent_identity  
percent_similarity  
raw_score  
bit_score  
hit_description  
blast_frame  
qry_strand (Plus | Minus)  
hit_length  
e_value

if UNIREF100P blast result then following are also added

domain  
kingdom  
phylum  
class  
order  
family  
genus  
species  
organism  
fxn_topHit

OUTPUT
A tab delimited file of BLAST records, which are in sync with each columns of VIROME blast table.

Output of this script can be uploaded directly to VIROME blast table without any further modification or column specification.

EXAMPLE

blast-result-prep.pl --input=/path/to/file.name --outdir=/path/to/output/dir --liblist=/path/to/file.name --lookupDir=/path/to/file.name
BTABTRIMMGOL.PL

NAME
btabTrimMGOL.pl - remove limits the number of hits per query in a btab file

SYNOPSIS
USAGE: btabTrim.pl
--input_file_base=input_btab_file_base-name
--input_file_path=input_btab-file-path
--input_file_extension=input_btab-file-extension
--output=/path/to/trim_file.btab
[
  --log=/path/to/logfile
  --debug=N
]

OPTIONS
--btab_file_base, -n REQUIRED The base name of input btab file.
--output, -o REQUIRED btab blast file trimmed to allow a maximum of N hits per query
--log, -l OPTIONAL Log file location
--debug, -d OPTIONAL Debug level
--help, -h This help message

DESCRIPTION
This script is used to limit the number of hits per query in a btab blast result file

INPUT
The input to this is defined using the --btabIN_file_base, --btabIN_file_extension, --fasta_file_path and -btab option. This should point to the btab ncbi-blast output and original FASTA file information.

OUTPUT
The output is defined using the --output. This file is a btab format file.

CONTACT
Shawn Polson
polson@dbi.udel.edu
**NAME**

db-load-library.pl - load Library output to DB

**SYNOPSIS**

**USAGE:**
db-load-library.pl
   --id=/id the id of the library
   --user=/user/who/is/uploading/component
   --outdir=/path/to/output/dir
   --env=/env where to execute script
   [
   --log=/path/to/logfile
   --debug=N
   ]

**OPTIONS**

--id, -i REQUIRED The id number of the library you are running. You can find this in the library table on Jabba.

--user, -u REQUIRED User who is uploading library information.

--outdir, o REQUIRED Path to output dir where a tab file of library id and library prefix is stored.

--env, -e REQUIRED Specific environment where this script is executed. Based on these values DB connection and file locations are set.

Currently supports

    igs
dbi
ageek or
test

--log, -l OPTIONAL Log file location

--debug, -d OPTIONAL Debug level

--help, -h This help message

**DESCRIPTION**

Add Library Information to VIROME MySQL database

**INPUT**

The input to this is defined using the --id. This should be the id number of the library you wish to run. If the library you wish to run does not have an id than you will need to add the information to the table.

**OUTPUT**

Library information is added to VIROME library table.
CONTACT
  Daniel J. Nasko
dan.nasko@gmail.com
DB-LOAD-NOHIT.PL

NAME
db-load-nohit.pl

SYNOPSIS
USAGE:
  db-load-nohit.pl
  --server=server-name
  --env=dbi
  --library=libraryId
  --input=/path/to/input/file
  --outdir=/path/to/output/dir
  [                             
    --log=/path/to/logfile
    --debug=N                 
  ]

OPTIONS
--input, -i OPTIONAL if server and library is defined Tab delimited file of library and its metadata information
--outdir, -o REQUIRED Complete path to output dir.
--server,-s REQUIRED Server/database name
--library,-l REQUIRED Specific library ID
--env, -e REQUIRED Specific environment where this script is executed.
Based on these values DB connection and file locations are set.

Currently supports
  igs
  dbi
  ageek or
test

--log, -l OPTIONAL Log file location
--debug, -d OPTIONAL Debug level
--help, -h This help message

DESCRIPTION
For all ORFs that are defined as ORFans by VIROME pipeline, add a blank/dummy row in VIROME blast table, So they can be displayed and treaded similar to Functional or Metagenome BLAST hits.

INPUT
The input is defined with --server, --library.

OUTPUT
Add no-hit row to BLAST tables.
EXAMPLE

db-load-nohit.pl --server=calliope --env=dbi --library=31
DB-LOAD-UNIREF-LOOKUP.PL

NAME
db-load-uniref-lookup.pl - load uniref100+ cluster output

SYNOPSIS
USAGE: db-load-blast.pl
--input=/path/to/cluster/output

[ --log=/path/to/logfile
  --debug=N
]

OPTIONS
--input, -i REQUIRED The full path to cd-hit cluster output

--log, -l OPTIONAL Log file location

--debug, -d OPTIONAL Debug level

--help, -h This help message

DESCRIPTION
DEPRICATED: This script is used to load uniref100+ cd-hit cluster output in to MySQL database.

INPUT
The input to this is defined using the --input. This should point to the cd-hit cluster output

EXAMPLE
db-load-uniref-lookup.pl --input=/path/to/file.name
DB-TO-LOOKUP.PL

NAME
db-to-lookup.pl: Create a MLDBM lookup file

SYNOPSIS
USAGE: db-to-lookup.pl
    --input=/library/info/file
    --table=/tablename
    --env=/env/where/executing
    --outdir=/output/dir
    [ --log=/path/to/logfile
      --debug=N
    ]

OPTIONS
    --input, -i REQUIRED Tab delimited library info file.
    --table, -t REQUIRED MySQL DB table name
    --env, -e REQUIRED Specific environment where this script is executed. Based on these values DB connection and file locations are set.
    Currently supports
      igs
      dbi
      ageek or
test
    --outdir, -o REQUIRED Output dir where lookup file will be stored
    --log, -l OPTIONAL Log file location
    --debug, -d OPTIONAL Debug level
    --help, -h This help message

DESCRIPTION
Convert MySQL table into MLDBM file base database. This is used by VIROME pipeline as a lookup file per library, so as to reduce MySQL look ups and subsequently reduce load on MySQL server.

INPUT
A library and VIROME table name to convert into MLDBM database

Currently only supports VIROME sequence table

OUTPUT
A MLDBM database.

EXAMPLE

78
db-to-lookup.pl --input=/path/to/file.name --outdir=/path/to/output/dir
--table=table.name --env=dbi

DB-UPLOAD.PL

NAME
db-upload.pl: Upload file into MySQL DB

SYNOPSIS
USAGE: db-upload.pl
  --input=/file/name
  --table=/tablename
  --env=/env/where/executing
  --outdir=/output/dir/loc
  [  --log=/path/to/logfile
    --debug=N
  ]

OPTIONS
--input, -i REQUIRED The full path to tab delimited
--table, -t REQUIRED MySQL DB table name
--env, -e REQUIRED Specific environment where this script is executed. Based on these values DB connection and file locations are set.

Currently supports
  igs
dbi
  ageek or
test

--outdir, -o REQUIRED Output dir where lookup file will be stored
--log, -l OPTIONAL Log file location
--debug, -d OPTIONAL Debug level
--help, -h This help message

DESCRIPTION
This script is used to upload data already formatted using various prep scripts into VIROME MySQL tables

INPUT
A tab delimited file. Each column is in sync with respective VIROME SQL table

Currently following tables are supported
  sequence_relationship
sequence
blastx
blastp
blastn
tRNA and
ORF

OUTPUT
All data is uploaded into respective SQL tables

EXAMPLE
db-upload.pl --input=/path/to/file.name --outdir=/path/to/output/dir --
table=table.name --env=dbi
DEREP-BLAST-OUTPUT.PL

NAME
derep-blast-output.pl - format library header after cd-hit

SYNOPSIS
USAGE: derep-blast-output.pl
--file_part=/blast/btab/file/name
--file_path=/blast/btab/file/path
--clstr=/path/to/cdhit/clstr/list
--output=/full/path/to/output/file
[ --log=/path/to/logfile
  --debug=N
]

OPTIONS
--file_path, -f REQUIRED The full path to blast btab file output
--file_part, -p REQUIRED The file part of blast btab file
--clstr, -c REQUIRED The full path to cdhit clstr output file list
--output, -o REQUIRED The full path to output file
--debug,-d Debug level. Use a large number to turn on verbose debugging.
--log, -l OPTIONAL Log file location
--debug, -d OPTIONAL Debug level
--help, -h This help message

DESCRIPTION
This script is used to de-replication cluster file so blast btab includes all sequence part of a cluster.

INPUT
The input to this is defined using the --file_part. This should point to the blast btab file part, --file_path is the blast btab file path --clstr should point to cdhit clstr file list, --output points to full path to new btab output file.

OUTPUT
A tab delimited output file where each UNIREF100P has been expanded to its respective SEED, KEGG, COG, ACLAME and PhageSEED records with respective accession number, description and taxonomy. Other BLAST information such as e-value, bit score, length, hsp information remains the same

EXAMPLE
derep-blast-output.pl --file_part=/blast/btab/file/name --file_path=/blast/btab/file/path --clstr=/path/to/cdhit/clstr/list --output=/full/path/to/output/file
ENV_LIB_STATS.PL

NAME
  env_lib_stats.pl

SYNOPSIS
  USAGE: env_lib_stats.pl
  --server=server-name
  --env=dbi
  --library=libraryId
  --lookupDir=/location/of/mldbm/dir
  --input=/path/to/input/file
  --outdir=/path/to/output/dir
  [ --log=/path/to/logfile
    --debug=N
  ]

OPTIONS
  --input, -i OPTIONAL if server and library is defined Tab delimited
  file of library and its metadata information

  --outdir, -o REQUIRED Complete path to output dir.

  --server, -s OPTIONAL if input and/or server is defined.
  Server/database name where library data is stored

  --library, -l OPTIONAL if input and/or server is defined. Specific
  library ID

  --lookupDir, -ld REQUIRED Directory of MLDBM lookup files

  --env, -e REQUIRED Specific environment where this script is executed.
  Based on these values DB connection and file locations are set.

  Currently supports
    igs
    dbi
    ageek or
    test

  --log, -l OPTIONAL Log file location

  --debug, -d OPTIONAL Debug level

  --help, -h This help message

DESCRIPTION
  This script will process all libraries on a given server. Get a break
  down of environmental statistics as per
  genesis,
  sphere,
eco-system,
   extreme,
physio_chem,
library type and
various env. library type.

Counts for each category is stored in _cnt field, and all sequence IDs
each categories are stored in an external file.

INPUT
The input is defined with --server, which is a domain name only.
e.g.: calliope (if server name is calliope.dbi.udel.edu)

OUTPUT
All counts for each category are stored in "env_statistics" table on
the "server" given as input. All sequence IDs for each category are
stored in an external file, and its location is stored in db.

EXAMPLE
env_lib_stats.pl --server=calliope --env=dbi --library=31
FRAGGENESCAN2VIROMEORF.PL

NAME
fragGeneScan2viromeORF.pl

SYNOPSIS
USAGE: fragGeneScan2viromeORF.pl
-i=input dir
-p=inputFilePrefix
-r=originalReadFile.fsa
[
  --log=/path/to/logfile
  --debug=N
]

OPTIONS
--input_dir,-i REQUIRED Input directory where all fragGeneScan output
files are expecting PREFIX.frag_gene_scan.raw file PREFIX.faa
PREFIX.ffn and PREFIX

--inputFilePrefix, -p REQUIRED Input file name prefix

--read_file, -r REQUIRED Original read file input to FragGeneScan

--log, -l OPTIONAL Log file location

--debug, -d OPTIONAL Debug level

--help, -h This help message

DESCRIPTION
Convert FragGeneScan output (A JCVI in house ORF predictor) into VIROME
ORF predictor format so all ORF can be treated and displayed via VIROME
app the same.

INPUT
Raw fragGeneScan output along with the original FASTA file used to
predict ORF by fragGeneScan

OUTPUT
ORF predictor as expected by VIROME.
Currently support fragGeneScan to Metagene conversion.

EXAMPLE
fragGeneScan2viromeORF i=input dir -p=inputFilePrefix -r=originalReadFile.fsa
FXN-DB-TO-LOOKUP.PL

NAME

fxn-db-to-lookup.pl: Create a MLDBM lookup file

SYNOPSIS

USAGE: fxn-db-to-lookup.pl
--input=/file/name
--table=/tablename
--env=/env/where/executing
--outdir=/output/dir/loc
[
     --log=/path/to/logfile
     --debug=N
]

OPTIONS

--input, -i REQUIRED The full path to tab delimited

--table, -t REQUIRED MySQL DB table name

--env, -e REQUIRED Specific environment where this script is executed. Based on these values DB connection and file locations are set.

Currently supports
  igs
dbi
  ageek or
test

--outdir, -o REQUIRED Output dir where lookup file will be stored

--log, -l OPTIONAL Log file location

--debug, -d OPTIONAL Debug level

--help, -h This help message

DESCRIPTION

Convert functional tables from MySQL table to MLDBM database. For faster lookup and reduce MySQL load by VIROME pipeline

INPUT

A functional database name to be converted into MLDBM database.

Currently supports
  SEED
  PhageSEED
  KEGG
  COG
  ACLAME and
  UNIREF
OUTPUT
  A MLDBM lookup database

EXAMPLE
  fxn-db-to-lookup.pl --outdir=/location/of/outdir --env=dbi --table=seed
FXNALDBBREAK.PL

NAME
FxnalDBBreak.pl

SYNOPSIS
USAGE: FxnalDBBreak.pl
--input=/input/filename
--outdir=/path/to/output/dir
--server=server-name
--env=dbi
--library=libraryId
[
    --log=/path/to/logfile
    --debug=N
]

OPTIONS
--input, -i OPTIONAL if server and library is defined Tab delimited file of library and its metadata information

--outdir, -o REQUIRED Complete path to output dir.

--server, -s OPTIONAL if input and/or server is defined.
Server/database name where library data is stored

--library, -l OPTIONAL if input and/or server is defined. Specific library ID

--env, -e REQUIRED Specific environment where this script is executed. Based on these values DB connection and file locations are set.

Currently supports
    igs
dbi
    ageek or
test

--log, -l OPTIONAL Log file location

--debug, -d OPTIONAL Debug level

--help, -h This help message

DESCRIPTION
Create a XML document for each functional database with hierarchical structure as represented by each functional database annotation.

Since the hierarchical tree structure of GO and ACLAME database can be arbitrarily long, a hard limit of 6 levels is set for GO and ACLAME tree structure
INPUT

The input is defined with --server, --library.

OUTPUT

A well-defined XML file with number of ORFs within each database hierarchy.
A secondary xml document with IDDOC prefix is created which store individual IDs for each ORF. IDDOC and XMLDOC files are linked via TAG_

egg: XMLDOC

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <FUNCTION_1 NAME="Metabolism" LABEL="Metabolism" VALUE="15"
                 TAG="TAG_1" IDFNAME="KEGG_IDDOC_27.xml">
        <FUNCTION_2 NAME="Amino Acid Metabolism" LABEL="Amino Acid Metabolism" VALUE="8"
                     TAG="TAG_2" IDFNAME="KEGG_IDDOC_27.xml">
            <FUNCTION_3 NAME="Cysteine and Methionine Metabolism"
                         LABEL="Cysteine and Methionine Metabolism" VALUE="7"
                         TAG="TAG_3" IDFNAME="KEGG_IDDOC_27.xml"/>
            <FUNCTION_3 NAME="Glycine, Serine and Threonine Metabolism"
                            LABEL="Glycine, Serine and Threonine Metabolism" VALUE="1"
                            TAG="TAG_4" IDFNAME="KEGG_IDDOC_27.xml"/>
        </FUNCTION_2>
    </FUNCTION_1>

    <FUNCTION_2 NAME="Carbohydrate Metabolism" LABEL="Carbohydrate Metabolism" VALUE="1"
                 TAG="TAG_5" IDFNAME="KEGG_IDDOC_27.xml">
        <FUNCTION_3 NAME="Fructose and Mannose Metabolism"
                         LABEL="Fructose and Mannose Metabolism" VALUE="1"
                         TAG="TAG_6" IDFNAME="KEGG_IDDOC_27.xml"/>
        <FUNCTION_3 NAME="Butanoate Metabolism" LABEL="Butanoate Metabolism" VALUE="1"
                         TAG="TAG_7" IDFNAME="KEGG_IDDOC_27.xml"/>
    </FUNCTION_2>

    <FUNCTION_2 NAME="Lipid Metabolism" LABEL="Lipid Metabolism" VALUE="1"
                 TAG="TAG_8" IDFNAME="KEGG_IDDOC_27.xml">
        <FUNCTION_3 NAME="Linoleic Acid Metabolism" LABEL="Linoleic Acid Metabolism" VALUE="1"
                         TAG="TAG_9" IDFNAME="KEGG_IDDOC_27.xml"/>
    </FUNCTION_2>

    <FUNCTION_2 NAME="Xenobiotics Biodegradation and Metabolism"
                 LABEL="Xenobiotics Biodegradation and Metabolism" VALUE="1"
                 TAG="TAG_10" IDFNAME="KEGG_IDDOC_27.xml">
        <FUNCTION_3 NAME="Tetrachloroethene Degradation"
                         LABEL="Tetrachloroethene Degradation" VALUE="1"
                         TAG="TAG_11" IDFNAME="KEGG_IDDOC_27.xml"/>
        <FUNCTION_3 NAME="Bisphenol a Degradation" LABEL="Bisphenol a Degradation" VALUE="1"
                         TAG="TAG_12" IDFNAME="KEGG_IDDOC_27.xml"/>
    </FUNCTION_2>

    <FUNCTION_2 NAME="Nucleotide Metabolism" LABEL="Nucleotide Metabolism" VALUE="4"
                 TAG="TAG_13" IDFNAME="KEGG_IDDOC_27.xml">
```
<FUNCTION_3 NAME="Purine Metabolism" LABEL="Purine Metabolism" VALUE="3" TAG="TAG_14"
IDFNAME="KEGG_IDDOC_27.xml"/>
<FUNCTION_3 NAME="Pyrimidine Metabolism" LABEL="Pyrimidine Metabolism" VALUE="4" TAG="TAG_15"
IDFNAME="KEGG_IDDOC_27.xml"/>
</FUNCTION_2>
<FUNCTION_2 NAME="Energy Metabolism" LABEL="Energy Metabolism"
VALUE="3" TAG="TAG_16" IDFNAME="KEGG_IDDOC_27.xml">
...
...
...
</FUNCTION_2>

egg: IDDOC
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <TAG_1 IDLIST="23188,23190,23255,23288,23381,23389,23416,23456,23533,23541,23630,23654,23666,23739,23743"/>
  <TAG_2 IDLIST="23188,23255,23381,23389,23416,23630,23666,23743"/>
  <TAG_3 IDLIST="23188,23255,23389,23416,23630,23666,23743"/>
  <TAG_4 IDLIST="23381"/>
  <TAG_5 IDLIST="23190"/>
  <TAG_6 IDLIST="23190"/>
  <TAG_7 IDLIST="23190"/>
  <TAG_8 IDLIST="23190"/>
  <TAG_9 IDLIST="23190"/>
  <TAG_10 IDLIST="23190"/>
  <TAG_11 IDLIST="23190"/>
  <TAG_12 IDLIST="23190"/>
  <TAG_13 IDLIST="23288,23533,23541,23654"/>
  <TAG_14 IDLIST="23288,23533,23654"/>
  <TAG_15 IDLIST="23288,23533,23541,23654"/>
  <TAG_16 IDLIST="23381,23456,23739"/>
  ...
</root>

EXAMPLE
FxnalDBBreak.pl --server=calliope --env=dbi --library=31
GEN_LIB_STATS.PL

NAME

gen_lib_stats.pl

SYNOPSIS

USAGE: gen_lib_stats.pl
    --server=server-name
    --env=env
    --library=libraryId
    --lookupDir=/location/of/mldbm/dir
    --input=/path/to/input/file
    --outdir=/path/to/output/dir
    [ 
      --log=/path/to/logfile
      --debug=N
    ]

OPTIONS

--input, -i OPTIONAL if server and library is defined Tab delimited file of library and its metadata information

--outdir, -o REQUIRED Complete path to output dir.

--server, -s OPTIONAL if input and/or server is defined. Server/database name where library data is stored

--library, -l OPTIONAL if input and/or server is defined. Specific library ID

--lookupDir, -ld REQUIRED Directory of MLDBM lookup files

--env, -e REQUIRED Specific environment where this script is executed. Based on these values DB connection and file locations are set.

Currently supports
    igs
    dbi
    ageek or
    test

--log, -l OPTIONAL Log file location

--debug, -d OPTIONAL Debug level

--help, -h This help message

DESCRIPTION

This script will process all libraries on a given server. Get a breakdown of
  ORF types (missing 3', missing 5', incomplete, complete)
  ORF model (bacteria, archaea, phage)
LIB type (viral only, microbial only, top viral, top microbial)
FUNCTIONAL and Unclassified
Counts for each category are stored in _cnt field, and all sequence IDs for each category are stored in an external file.

INPUT
The input is defined with --server which is a domain name only.
e.g.: calliope (if server name is calliope.dbi.udel.edu)

OUTPUT
All counts for each category are stored in "statistics" table on the "server" given as input. All sequence IDs for each category are stored in an external file, and its location is stored in db.

EXAMPLE
gen_lib_stats.pl --server=calliope --env=dbi --library=31
METAGENE-PREP.PL

NAME
metagene-prep.pl - prepare metagene raw output for MySQL upload

SYNOPSIS
USAGE: metagene-prep.pl
    --input=/path/to/metagene/raw/output
    --liblist=/library/list/file/from/DB-load-library
    lookupDir=/dir/where/mldbm/lookup/files/are
    [ --log=/path/to/logfile
    --debug=N
    ]

OPTIONS
--input, -i REQUIRED The full path to metagene raw output
--liblist, -ll REQUIRED Library list file, and output of DB-load-library.
--lookupDir, -ld REQUIRED Dir where all lookup files are stored.
--log, -l OPTIONAL Log file location
--debug, -d OPTIONAL Debug level
--help, -h This help message

DESCRIPTION
DEPRICATE. Convert raw Metagene output into well-defined data that is in sync with VIROME ORF table

egg: raw Metagene output

    # EB8_MMETSP1129-20121227-18562
    # gc = 0.642596, rbs = -1
    # self:
    gene_1 1 891 + 0 01 94.2685 a - - -
    # EB8_MMETSP1129-20121227-18563
    # gc = 0.516753, rbs = -1
    # self:
    gene_1 39 1502 - 0 11 42.0969 p - - -
    # EB8_MMETSP1129-20121227-18564
    # gc = 0.60627, rbs = -1
    # self: p
    gene_1 1 7494 + 0 01 410.446 s - - -
    # EB8_MMETSP1129-20121227-18565
    # gc = 0.579521, rbs = -1
    # self: -
    gene_1 1 201 + 0 01 8.40845 p - - -
    gene_2 351 918 - 1 01 67.0183 p - - -
INPUT
The input to this is defined using the --input. This should point to
the metagene raw output file.

OUTPUT
A tab delimited file of ORF details in sync with VIROME ORF table

EXAMPLE
metagene-prep.pl --input=/path/to/file.name --
outdir=/path/to/output/dir --liblist=/path/to/file.name --
lookupDir=/path/to/file.name
MGA2SEQ_PEP.PL

NAME
mga2seq_pep.pl - convert metagene raw output to FASTA nucleotide and peptide file

SYNOPSIS
USAGE: mga2seq_pep.pl
--input=/path/to/FASTA
--mga=/path/to/mga/output
--prefix
--outdir
[ --log=/path/to/logfile
  --debug=N
]

OPTIONS
--input, -i REQUIRED The full path to FASTA sequence file.
--mga, mga REQUIRED Metagene output file
--log, -l OPTIONAL Log file location
--debug, -d OPTIONAL Debug level
--help, -h This help message

DESCRIPTION
This script is used to convert metagene output to nucleotide and peptide sequence file.

INPUT
The input to this is defined using the --input/-i or mga/-m. This should point to the FASTA file containing sequence(s), and metagene output file

OUTPUT
Two FASTA files one with predicted ORFs in amino acid format, and one with predicted ORFs in nucleotide format

EXAMPLE
mga2seq_pep.pl --input=/path/to/FASTA --mga=/path/to/mga/output --prefix=abc --outdir=/location/to/dir
**NT_FASTA_CHECK.PL**

**NAME**
nt_fasta_check.pl

**SYNOPSIS**

**USAGE:** nt_fasta_check.pl

**OPTIONS**

--FASTA, -f REQUIRED Input FASTA file

--outdir, -o REQUIRED Output directory to store results.

--log, -l OPTIONAL Log file location

--debug, -d OPTIONAL Debug level

--help, -h This help message

**DESCRIPTION**

Takes as input a FASTA file, and library prefix/library file Prefix each sequence id with PREFIX from library file, and check the quality of read bases.

**INPUT**

FASTA file in .fsa, .fa, .FASTA or .txt file format OUTPUT dir where updated input file is stored (output dir cannot be same as where input file is)

**OUTPUT**

An updated FASTA file with each sequence ID prefixed by PREFIX A ref file with original and new sequence ID

**EXAMPLE**

nt_fasta_check.pl -i=input.fsa -o=/output_dir -ll=library_list_file

or

nt_fasta_check.pl -i=input.fsa -o=/output_dir -lf=library_file
QC_FILTER.PL

NAME
QC_filter.pl

SYNOPSIS

USAGE: QC_filter.pl
      --FASTA FASTA-file
      --qual qual-file-list
      --cutoff cutoff-value
      --minlen min-length
      --outdir output-dir

OPTIONS

--FASTA, -f A FASTA file
--qual, -q A quality file is original input was SFF or FASTQ
--cutoff, -c Cuff off threshold
--minlen, -m Minimum length cutoff for each input read
--outdir, -o
--help,-h This help message

DESCRIPTION

Give input sequence check for minimum length of each read, and mean sequence quality

INPUT

Raw sequences

OUTPUT

FASTA file where low quality sequences and sequences less than minimum length requirement are removed from input file.

EXAMPLE

QC_filter.pl --FASTA=FASTA-file --qual=qual-file --cutoff=cutoff-value
             --minlen=min-length --outdir=output-dir
RESET-PROCESSING-DB.PL

NAME
reset-processing-db.pl

SYNOPSIS
USAGE: reset-processing-db.pl
--lib_file=/tab/delimited/lib/info/file
--env=/execution/loc
--output_dir=/path/to/output/dir

OPTIONS
--lib_file,-lf Absolute file path that contains tab delimited library
information

egg: id  name  prefix  server

--output_dir,-o Output directory

--env, -e REQUIRED Specific environment where this script is executed.
Based on these values DB connection and file locations are set.

Currently supports
    igs
    dbi
    ageek or
test

--log, -l OPTIONAL Log file location

--debug, -d OPTIONAL Debug level

--help, -h This help message

DESCRIPTION
Create backup of processing DB, empty processing DB, set ids of
processing DB to respective server info. This is only for igs
processing

INPUT
The input is defined with --lib_file --env.

OUTPUT
MySQL dump of processing db.

EXAMPLE
reset-processing-db.pl --lib_file=file_name.tab --env=igs
**RRNA-SCRUB.PL**

**NAME**

rRNA-scrub.pl - remove rRNA sequences identified by blast from the input FASTA.

**SYNOPSIS**

**USAGE:** rRNA-scrub.pl

```
--fasta_file_base=FASTA file base name
--fasta_file_path=/path/to/fasta_file
--fasta_file_extension=FASTA file extension
--btab=/path/to/input_file.btab
--outputA=/path/to/rRNA_minus_original.fasta
--outputB=/path/to/rRNA_identified_sequences.fasta
```

```
[   --log=/path/to/logfile
    --debug=N
]
```

**OPTIONS**

- `--fasta_file_base`, `-n` REQUIRED The base name of FASTA file.
- `--fasta_file_path`, `-b` REQUIRED The full path to FASTA file.
- `--fasta_file_extension`, `-e` REQUIRED The file extension
- `--btab`, `-b` REQUIRED The input btab blast output from the ncbi-blastn suite.
- `--outputA`, `-oA` REQUIRED The file to which all sequence except for rRNA identified sequence will be written.
- `--outputB`, `-oB` REQUIRED The file to which all rRNA identified sequences will be written
- `--log`, `-l` OPTIONAL Log file location
- `--debug`, `-d` OPTIONAL Debug level
- `--help`, `-h` This help message

**DESCRIPTION**

This script is used to scrub rRNA identified sequences from ncbi-blastn suite, and create a new FASTA file.

**INPUT**

The input to this is defined using the `--fasta_file_base`, `--fasta_file_extension` `--fasta_file_path` and `-btab` option. This should point to the btab ncbi-blastn output and original FASTA file information.

**OUTPUT**
The output is defined using the --outputA and -outputB option. These files created are FASTA files of original FASTA minus the rRNA sequences, and FASTA file of rRNA identified sequences.

**EXAMPLE**

```bash
rRNA-scrub.pl --fasta_file_base=FASTA-file-base-name --fasta_file_path=/path/to/fasta_file --fasta_file_extension=FASTA file extension --btab=/path/to/input_file.btab --outputA=/path/to/rRNA_minus_original.fasta --outputB=/path/to/rRNA_identified_sequences.fasta
```
SEQUENCE_RELATIONSHIP.PL

NAME
sequence_relationship.pl - prepare sequence info for upload to DB

SYNOPSIS
USAGE: sequence_relationship.pl
   --input=/library/list/file
   --outdir=/output/dir
   --env=execution env name
   [--log=/path/to/logfile
    --debug=N
   ]

OPTIONS
--input, -i OPTIONAL if server and library is defined Tab delimited file of library and its metadata information
--outdir, -o REQUIRED Complete path to output dir.
--env, -e REQUIRED Specific environment where this script is executed. Based on these values DB connection and file locations are set.

   Currently supports
      igs
dbi
     ageek or
test

--log, -l OPTIONAL Log file location
--debug, -d OPTIONAL Debug level
--help, -h This help message

DESCRIPTION
Create a sequence relation tree between reads/contigs -> ORFs (peptide) / ORF (nuclotide) and
reads/contigs -> rRNAs

INPUT
A tab delimited file with library and its metadata to be processed.
egg:
   #libraryID  libary name  library prefix  server
   220  M601K  POA  terpsichore

OUTPUT
A tab delimited file of read/contig, ORF and rRNA id parent/child relationship tree.
EXAMPLE

    sequence_relationship.pl --input=/path/to/input --
    output=/path/to/outdir --env=dbi
SEQUENCE-PREP.PL

NAME
sequence-prep.pl - prepare sequence info for upload to DB

SYNOPSIS
USAGE: sequence-prep.pl
   --input=/path/to/FASTA
   --outdir=/output/dir
   --libListFile=/library/list/file
   --type=sequence type
   --outdir=/output/dir
   [  
      --log=/path/to/logfile
      --debug=N
   ]

OPTIONS
--input, -i REQUIRED The full path to FASTA sequence file. # start a
   comment. # File format >ABC125234 .... # where ABC is three letter
   library prefix.
--outdir, -od REQUIRED Output dir location
--libListFile, -ll REQUIRED Library list file, an output of DB-load-
   library.
--type, -t REQUIRED Sequence type read=1, rRNA=2, orf (aa)=3, orf
   (dna)=4
--log, -l OPTIONAL Log file location
--debug, -d OPTIONAL Debug level
--help, -h This help message

DESCRIPTION
This script is used to prepare sequence for MySQL upload.

INPUT
The input to this is defined using the --input. This should point to
the FASTA file containing sequence(s). Input must be a multifasta
file, each file contain sequences for one library.

OUTPUT
A tab delimited file with columns in sync with VIROME sequence table.

EXAMPLE
sequence-prep.pl --input=/path/to/input --outdir=/path/to/outdir --
   liblist=/path/to/lib/list --typeId=2
SPLIT_BTAB.PL

NAME
split_btab.pl -- splits btab results

SYNOPSIS
split_btab.pl
  --btab /Path/to/infile.btab
  --out /Path/to/outdir/
  --splits=25
[--help]
[--manual]

DESCRIPTION
Split btab files such that they don't break up query results.

OPTIONS
-b, --btab=FILENAME Input file in BTAB format. (Required)
-o, --outdir=FILENAME Output directory. (Required)
-s, --splits=INT Number of splits. (Required)
-h, --help Displays the usage message. (Optional)
-m, --manual Displays full manual. (Optional)

DEPENDENCIES
Requires the following Perl libraries.

AUTHOR
Written by Daniel Nasko, Center for Bioinformatics and Computational Biology, University of Delaware.

REPORTING BUGS
Report bugs to dnasko@udel.edu

COPYRIGHT
Copyright 2012 Daniel Nasko. License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>. This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

Please acknowledge author and affiliation in published work arising from this script's usage <http://bioinformatics.udel.edu/Core/Acknowledge>.
**SPLITMULTIFASTA.PL**

**NAME**
split_multifasta.pl - split a single FASTA file containing multiple sequences into separate files.

**SYNOPSIS**
USAGE: split_multifasta.pl
   --input_file=/path/to/some_file.fsa
   --output_dir=/path/to/somedir
   [ --output_list=/path/to/somefile.list
     --output_subdir_size=1000
     --output_subdir_prefix=FASTA
     --seqs_per_file=1
     --total_files=1
     --compress_output=1
   ]

**OPTIONS**
--input_file,-i The input multi-FASTA file to split.

--output_dir,-o The directory to which the output files will be written.

--output_list,-s Write a list file containing the paths of each of the regular output files. This may be useful for later scripts that can accept a list as input.

--output_file_prefix,-f If defined, each file created will have this string prepended to its name. This is ignored unless writing multiple sequences to each output file using the --seqs_per_file option with a value greater than 1, else each file created will just be a number.

--output_subdir_size,-u If defined, this script will create numbered subdirectories in the output directory, each containing this many sequences files. Once this limit is reached, another subdirectory is created.

--output_subdir_prefix,-p To be used along with --output_subdir_size, this allows more control of the names of the subdirectories created. Rather than just incrementing numbers (like 10), each subdirectory will be named with this prefix (like prefix10).

--total_files, -t Used if the user wants to specify the total outputs files to be created. The script will determine the amount of sequences per file to meet this parameter. Cannot be used in conjunction with the seqs_per_file parameter.

--compress_output,-c Output FASTA files will be gzipped when written.
--debug,-d Debug level. Use a large number to turn on verbose debugging.

--log,-l Log file

--help,-h This help message

DESCRIPTION
This script is used to split a single FASTA file containing multiple sequences into separate files containing one sequence each.

INPUT
The input is defined with --input_file and should be a single FASTA file. File extensions are ignored. When creating this multi-entry FASTA file, one should take care to make the first *word* after the > symbol a unique value, as it will be used as the file name for that sequence. For example:

>gi53791237 Tragulus javanicus p97bcnt gene for p97Bcnt
ACAGGAGAAGAGACTGAGAGACAGTTCCAGGAGAGCAAGAGACGCCTAAAGAAATGCAAGAAGTTA
AACTCACCACATCCTTGAGAGAGGCAGTACATGACATTCCAAAAACTTCAAACTAGTTCTTTAA
AAAGGAAASACTCTCTTTTAATATGATGCATATTATTAATTTATTTACTCATTGGCGTGGAGAGGAGAAATG

>gi15387669 Corynebacterium callunae pCC1 plasmid
ATGCATGCTAGTGTGGTGAGTATGAGCACACACATTCATGGGCACCGCCGGGGTGCCAGGGGGCTTGCCC
CTTGTCATCGAGGGTGGTGGCTTGCCCCGGCGATAGGACCAGCGCCCAACATTGCAACCCGGTGTCGCG
GGTGATCGCGCACCACCGCGCCCGGCCGACTCTCCCCCTGTCTAGGCCATATTTACGGCCGTCCACTG

Whitespace is ignored within the input file. See the OUTPUT section for more on creation of output files.

OUTPUT
The name of each output sequence file is pulled from the FASTA header of that sequence. The first *word* after the > symbol will be used as the file name, along with the extension .fsa. The word is defined as all the text after the > symbol up to the first whitespace.

If the above example were your input file, two files would be created:

gi53791237.fsa
gi15387669.fsa

Any characters other than a-z A-Z 0-9 _ - in the ID will be changed into an underscore. This only occurs in the file name; the original FASTA header within the file will be unmodified.
You can pass a path to the optional --output_list to create a text file containing the full paths to each of the FASTA files created by this script.

Two other optional arguments, --output_subdir_size and --output_subdir_prefix, can be used on input sets that are too large to write out to one directory. This depends on the limitations of your file system, but you usually don't want 100,000 files written in the same directory.

If you have an FASTA file containing 95000 sequences, and use the following option:

    --output_dir=/some/path
    --output_subdir_size=30000

The following will be created:

    directory    file count
    ------------------------
    /some/path/1/    30000
    /some/path/2/    30000
    /some/path/3/    30000
    /some/path/4/    5000

If you choose to create a list file (and you probably want to), it will contain these proper paths.

You may not want the subdirectories to simply be numbers, as above, so you can use the --output_subdir_prefix option. For example:

    --output_dir=/some/path
    --output_subdir_size=30000
    --output_subdir_prefix=FASTA

The following will be created:

    directory    file count
    ------------------------
    /some/path/FASTA1/    30000
    /some/path/FASTA2/    30000
    /some/path/FASTA3/    30000
    /some/path/FASTA4/    5000

Finally, you can write multiple sequences to each output file using the --seqs_per_file option, which can be used along with --output_subdir_size and --output_subdir_prefix. The main difference to note is that, if you use --seqs_per_file, the FASTA file created will no longer be named using values taken from the header, since it will contain multiple headers. Instead, the file will simply be named using sequential numbers starting at 1 (like 1.fsa). For example:

    --output_dir=/some/path
--output_subdir_size=3000
--output_subdir_prefix=FASTA
--seqs_per_file=10

The following will be created:

<table>
<thead>
<tr>
<th>directory</th>
<th>file count</th>
</tr>
</thead>
<tbody>
<tr>
<td>/some/path.fasta1/</td>
<td>3000</td>
</tr>
<tr>
<td>/some/path.fasta2/</td>
<td>3000</td>
</tr>
<tr>
<td>/some/path.fasta3/</td>
<td>3000</td>
</tr>
<tr>
<td>/some/path.fasta4/</td>
<td>500</td>
</tr>
</tbody>
</table>

CONTACT
Joshua Orvis
jorvis@tigr.org
TRNASCAN-PREP.PL

NAME  
tRNAScan-prep.pl - prepare tRNAScan raw output for MySQL upload

SYNOPSIS
USAGE: tRNAScan-prep.pl
--input=/path/to/metagene/raw/output
--outdir=/output/directory
--liblist=/library/list/file/from/DB-load-library
--lookupDir=/dir/where/mlmdb/lookup/files/are
[  
  --log=/path/to/logfile
  --debug=N
]

OPTIONS
--input, -i REQUIRED The full path to input file
--output, -d REQUIRED Path to output dir
--liblist, -ll REQUIRED Library list file, an output of DB-load-library.
--lookupDir, -ld REQUIRED Directory of MLDBM lookup files
--log, -l OPTIONAL Log file location
--debug, -d OPTIONAL Debug level
--help, -h This help message

DESCRIPTION
Convert tRNAScan raw output into a tab delimited file with columns that are in sync with VIROME tRNA table.

INPUT
The input to this is defined using the --input. This should point to the tRNAScan-SE raw output file list. One file per line.

egg:
PVP_GO0QTYD02JISH2  1  218  146 Lys  TTT  0  0  75.04
PVP_GO0QTYD02GM67  1  58  131 Met  CAT  0  0  72.35
PVP_GO0QTYD02GM67  2  192  274 Leu  TAA  0  0  68.29
PVP_GO0QTYD02GM67  3  281  352 Cys  GCA  0  0  55.94

OUTPUT
A tab delimited output of tRNA information

egg:
#seqeunceid  tRNA #  start  stop  anti  intron  cove_start  cove_end  score
<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>1</td>
<td>265</td>
<td>336</td>
<td>Pseudoudo</td>
<td>GTT</td>
<td>0</td>
</tr>
<tr>
<td>28.16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>2</td>
<td>4186</td>
<td>4260</td>
<td>Met</td>
<td>CAT</td>
<td>0</td>
</tr>
<tr>
<td>123</td>
<td>3</td>
<td>3876</td>
<td>3804</td>
<td>Undefet</td>
<td>???</td>
<td>0</td>
</tr>
<tr>
<td>456</td>
<td>1</td>
<td>577</td>
<td>652</td>
<td>Undefet</td>
<td>???</td>
<td>0</td>
</tr>
<tr>
<td>989</td>
<td>1</td>
<td>149</td>
<td>217</td>
<td>Asn</td>
<td>GTT</td>
<td>0</td>
</tr>
<tr>
<td>22.12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25.99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**EXAMPLE**

tRNAscan-prep.pl --input=/path/to/input --outdir=/path/to/outdir --liblist=/path/to/lib/list --lookupDir=/path/to/mldbm/lookup/dir
UPDATE_MGOL_NAMES_IN_FASTA.PL

NAME
update_mgol_names_in_fasta.pl

SYNOPSIS

USAGE: update_mgol_name.pl
--FASTA=/path/to/input/file.fasta
--outdir=/path/to/output/dir
[  
  --log=/path/to/logfile
  --debug=N
]

OPTIONS

--FASTA, -f REQUIRED Fasta file name from where MGOL sequence(s) names needs to be updated

--outdir, -o REQUIRED Output directory where updated MGOL FASTA file will be stored.

--log, -l OPTIONAL Log file location

--debug, -d OPTIONAL Debug level

--help, -h This help message

DESCRIPTION

Update MGOL sequence name to have appropriate 3 letter prefix as defined in mgol_library table.

INPUT

The input is defined with --FASTA and --map.

OUTPUT

Updated sequence name in the given FASTA file

EXAMPLE

update_mgol_names_in_fasta.pl --FASTA=filename.fasta --outdir=/path/to/output-dir
UPDATEQUALITYFILE.PL

NAME
updateQualityFile.pl

SYNOPSIS
USAGE: updateQualityFile.pl
--FASTA=/path/to/fasta.fsa
--quality=/path/to/quality_file.qual
--outdir=/path/to/output/dir
--prefix=prefix

OPTIONS
--FASTA, -f REQUIRED The full path to FASTA sequence file.
--quality, -q REQUIRED The full path to the associated quality file.
--outdir, -o REQUIRED Path to output dir
--prefix, -p REQUIRED Unique 3 digit alpha/numeric library identifier
--help,-h This help message

DESCRIPTION
Script is designed to look at the sequence file and pull out only those relevant quality scores. This is necessary as the FASTA file is passed thru QC components that will throw out certain sequences.

INPUT
The input to this is defined using the --FASTA/-f AND --quality/-q flags. These should point to the files containing the FASTA and quality files, respectively.

CONTACT
Daniel J. Nasko
dan.nasko@gmail.com
VIROMECLASSIFICATION.PL

NAME
   viromeClassification.pl

SYNOPSIS
   USAGE: viromeClassification.pl
      -input=/input/filename
      -outdir=/path/to/output/dir
      -server=server-name
      -env=dbi
      -library=libraryId
      [ -log=/path/to/logfile
         -debug=N
      ]

OPTIONS
   --input, -i OPTIONAL if server and library is defined Tab delimited file of library and its metadata information

   --outdir, -o REQUIRED Complete path to output dir.

   --server, -s OPTIONAL if input and/or server is defined. Server/database name where library data is stored

   --library, -l OPTIONAL if input and/or server is defined. Specific library ID

   --env, -e REQUIRED Specific environment where this script is executed. Based on these values DB connection and file locations are set.
      Currently supports
         igs
dbi
ageek or
   test

   --log, -l OPTIONAL Log file location

   --debug, -d OPTIONAL Debug level

   --help, -h This help message

DESCRIPTION
   Create XML document with number of ORF binned per VIROME category i.e.:
      tRNA
      rRNA
      Possible functional
      Unassigned functional
      Top viral
Top microbial
Viral only
Microbial only and
ORFan

ORF counts are log normal.

INPUT
The input is defined with --server, --library.

OUTPUT
A XML document with ORF counts per VIROME category

egg:
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <CATEGORY LABEL="tRNA" CAT="tRNA" VALUE="1" />
  <CATEGORY LABEL="rRNA" CAT="rRNA" VALUE="0" />
  <CATEGORY LABEL="Possible functional protein" CAT="fxn" VALUE="20" />
  <CATEGORY LABEL="Unassignfxn protein" CAT="unassignfxn" VALUE="38" />
  <CATEGORY LABEL="Top-hit viral" CAT="topviral" VALUE="1" />
  <CATEGORY LABEL="Viral only" CAT="allviral" VALUE="22" />
  <CATEGORY LABEL="Top-hit microbial" CAT="topmicrobial" VALUE="0" />
  <CATEGORY LABEL="Microbial only" CAT="allmicrobial" VALUE="0" />
  <CATEGORY LABEL="ORFan" CAT="orfan" VALUE="21" />
</root>

EXAMPLE
viromeClassification.pl --server=calliope --env=dbi --library=31 --outdir=/path/to/output/dir
VIROMEORF-PREP.PL

NAME
viromeorf-prep.pl - prepare orf calls for MySQL upload

SYNOPSIS
USAGE: viromeorf-prep.pl
   --input=/path/orf/output
   --liblist=/library/list/file/from/DB-load-library
   --lookupDir=/dir/where/mldbm/lookup/files/are
   --outdir=/path/to/output/dir
   [ --log=/path/to/logfile
     --debug=N
   ]

OPTIONS
--input, -i REQUIRED The full path to input file
--output, -d REQUIRED Path to output dir
--liblist, -ll REQUIRED Library list file, an output of DB-load-library.
--lookupDir, -ld REQUIRED Directory of MLDBM lookup files
--log, -l OPTIONAL Log file location
--debug, -d OPTIONAL Debug level
--help, -h This help message

DESCRIPTION
DEPRECATED. Prepare and update ORF data to sync with VIROME ORF table

INPUT
The input to this is defined using the --input. This should point to
orf output file.

OUTPUT
A tab delimited file with ORF detail in sync with VIROME ORF table.

EXAMPLE
viromeorf-prep.pl --input=/input/dir --outdir=/path/to/output/dir --
liblist=31 --lookupDir=/path/to/lookup/mldbm/dir
VIROMETAXONOMYXML.PL

NAME
    viromeTaxonomyXML.pl

SYNOPSIS
    USAGE: viromeTaxonomyXML.pl
             --input=/input/filename
             --outdir=/path/to/output/dir
             --server=server-name
             --env=dbi
             --library=libraryId
             [ --log=/path/to/logfile
               --debug=N
             ]

OPTIONS
    --input, -i OPTIONAL if server and library is defined Tab delimited file of library and its metadata information
    --outdir, -o REQUIRED Complete path to output dir.
    --server, -s OPTIONAL if input and/or server is defined. Server/database name where library data is stored
    --library, -l OPTIONAL if input and/or server is defined. Specific library ID
    --env, -e REQUIRED Specific environment where this script is executed. Based on these values DB connection and file locations are set.
        Currently supports
          igs
          dbi
          ageek or
test
    --log, -l OPTIONAL Log file location
    --debug, -d OPTIONAL Debug level
    --help, -h This help message

DESCRIPTION
    Create hierarchical XML document of all Uniref functional ORFs

INPUT
    The input is defined with --server, --library.

OUTPUT
    XML output per library of taxonomic information
e.g.:<root>
<DOMAIN LABEL="Archaea" NAME="Archaea" VALUE="2"
IDFNAME="TAXONOMY_IDDOC_27.xml" TAG="TAG_1">
<KINGDOM LABEL="Archaea" NAME="Archaea" VALUE="2"
IDFNAME="TAXONOMY_IDDOC_27.xml" TAG="TAG_2">
<PHYLUM LABEL="Euryarchaeota" NAME="Euryarchaeota" VALUE="2"
IDFNAME="TAXONOMY_IDDOC_27.xml" TAG="TAG_3">
<Class LABEL="Thermoplasmata" NAME="Thermoplasmata" VALUE="2"
IDFNAME="TAXONOMY_IDDOC_27.xml" TAG="TAG_4">
<Order LABEL="Thermoplasmatales" NAME="Thermoplasmatales"
VALUE="2" IDFNAME="TAXONOMY_IDDOC_27.xml" TAG="TAG_5">
<FAMILY LABEL="Thermoplasmataceae"
NAME="Thermoplasmataceae" VALUE="2"
IDFNAME="TAXONOMY_IDDOC_27.xml" TAG="TAG_6">
<GENUS LABEL="Thermoplasma"
NAME="Thermoplasma" VALUE="2"
IDFNAME="TAXONOMY_IDDOC_27.xml" TAG="TAG_7">
<SPECIES LABEL="volcanium"
NAME="volcanium" VALUE="2"
IDFNAME="TAXONOMY_IDDOC_27.xml" TAG="TAG_8">
<ORGANISM LABEL="Thermoplasma volcanium GSS1"
NAME="Thermoplasma volcanium GSS1"
VALUE="2" IDFNAME="TAXONOMY_IDDOC_27.xml" TAG="TAG_9" />
</SPECIES>
</GENUS>
</FAMILY>
</ORDER>
</CLASS>
</PHYLUM>
</KINGDOM>
</DOMAIN>
</root>
<ORGANISM LABEL="Bacteroides fragilis"
NAME="Bacteroides fragilis" VALUE="1"
IDFNAME="TAXONOMY_IDDOC_27.xml" TAG="TAG_18" />
<ORGANISM LABEL="Bacteroides sp. 2_2_4"
NAME="Bacteroides sp. 2_2_4" VALUE="13"
IDFNAME="TAXONOMY_IDDOC_27.xml" TAG="TAG_19" />
<ORGANISM LABEL="Bacteroides sp. 9_1_42FAA"
NAME="Bacteroides sp. 9_1_42FAA" VALUE="15"
IDFNAME="TAXONOMY_IDDOC_27.xml" TAG="TAG_20" />
<ORGANISM LABEL="Bacteroides sp. D1"
NAME="Bacteroides sp. D1" VALUE="23"
IDFNAME="TAXONOMY_IDDOC_27.xml" TAG="TAG_21" />
<ORGANISM LABEL="Bacteroides sp. D4"
NAME="Bacteroides sp. D4" VALUE="1"
IDFNAME="TAXONOMY_IDDOC_27.xml" TAG="TAG_22" />
</SPECIES>
<SPECIES LABEL="caccae" NAME="caccae"
VALUE="34" IDFNAME="TAXONOMY_IDDOC_27.xml"
TAG="TAG_23">
  <ORGANISM LABEL="Bacteroides caccae ATCC 43185"
NAME="Bacteroides caccae ATCC 43185" VALUE="34"
IDFNAME="TAXONOMY_IDDOC_27.xml" TAG="TAG_24" />
</SPECIES>
<SPECIES LABEL="capillosus" NAME="capillosus"
VALUE="1" IDFNAME="TAXONOMY_IDDOC_27.xml"
TAG="TAG_25">
  <ORGANISM LABEL="Bacteroides capillosus ATCC 29799"
NAME="Bacteroides capillosus ATCC 29799" VALUE="1"
IDFNAME="TAXONOMY_IDDOC_27.xml"
TAG="TAG_26" />
  ...
  ...
  ...
</SPECIES>
<root>
EXAMPLE
  viromeTaxonomyXML.pl --server=calliope --env=dbi --library=31
</root>