Spring 2017

Repeaterator: A tool for visualizing DNA repeat motifs in actinobacteriophage genomes

Grant A. Rybnicky
James Madison University

Follow this and additional works at: https://commons.lib.jmu.edu/honors201019
Part of the Bioinformatics Commons

Recommended Citation
https://commons.lib.jmu.edu/honors201019/272

This Thesis is brought to you for free and open access by the Honors College at JMU Scholarly Commons. It has been accepted for inclusion in Senior Honors Projects, 2010-current by an authorized administrator of JMU Scholarly Commons. For more information, please contact dc_admin@jmu.edu.
Repeaterator: a tool for visualizing DNA repeat motifs in Actinobacteriophage genomes

An Honors College Project Presented to
the Faculty of the Undergraduate
College of Science and Mathematics
James Madison University
by Grant Alexander Rybnicky
May 2017

Accepted by the faculty of the Department of Biology, James Madison University, in partial fulfillment of the requirements for the Honors College.

FACULTY COMMITTEE:

Project Advisor: Steven Gaines Cresawn, Ph. D
Associate Professor, Biology

Reader: Jonathan Monroe, Ph. D
Professor, Biology

Reader: Morgan Steffen, Ph. D
Assistant Professor, Biology

HONORS COLLEGE APPROVAL:

Bradley R. Newcomer, Ph.D.,
Dean, Honors College

PUBLIC PRESENTATION

This work is accepted for presentation, in part or in full, at JMU Biosymposium on April 20th, 2017.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>3</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>4</td>
</tr>
<tr>
<td>Abstract</td>
<td>6</td>
</tr>
<tr>
<td>Background</td>
<td>7</td>
</tr>
<tr>
<td>Implementation</td>
<td>11</td>
</tr>
<tr>
<td><em>Architecture</em></td>
<td>11</td>
</tr>
<tr>
<td><em>Critical Issues</em></td>
<td>12</td>
</tr>
<tr>
<td>Results and Discussion</td>
<td>14</td>
</tr>
<tr>
<td><em>User Interface</em></td>
<td>14</td>
</tr>
<tr>
<td><em>Comparison to Existing Software</em></td>
<td>17</td>
</tr>
<tr>
<td><em>Case Study</em></td>
<td>22</td>
</tr>
<tr>
<td><em>Future Development</em></td>
<td>25</td>
</tr>
<tr>
<td>Conclusions</td>
<td>27</td>
</tr>
<tr>
<td>Appendix I: Repeaterator Javascript</td>
<td>28</td>
</tr>
<tr>
<td>Appendix II: Repeaterator HTML</td>
<td>44</td>
</tr>
<tr>
<td>References</td>
<td>50</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Architecture of Repeaterator</td>
<td>12</td>
</tr>
<tr>
<td>2. Repeaterator user interface</td>
<td>15</td>
</tr>
<tr>
<td>3. Repeaterator options</td>
<td>16</td>
</tr>
<tr>
<td>4. Dot plot of OlympicHelado against itself generated using Gepard</td>
<td>18</td>
</tr>
<tr>
<td>5. Identification of repeat motifs for OlympicHelado using MEME</td>
<td>19</td>
</tr>
<tr>
<td>6. Identification and location of repeats for OlympicHelado using REPuter</td>
<td>20</td>
</tr>
<tr>
<td>7. Comparative genome map of 3 cluster BI1 phages generated using Phamerator</td>
<td>22</td>
</tr>
<tr>
<td>8. Repeat map of OlympicHelado generated using Repeaterator</td>
<td>24</td>
</tr>
<tr>
<td>9. Repeat map of OlympicHelado motifs 2 and 5</td>
<td>25</td>
</tr>
</tbody>
</table>
Acknowledgements

Foremost I would like to thank Dr. Steven Cresawn for his support and encouragement over the past 4 years. Dr. Cresawn has been a great friend and mentor, providing me with countless hours of collaboration, academic planning, and life advice. My experience at JMU would have been significantly different if we had not met. I would like to also thank Dr. Jon Monroe and Dr. Morgan Steffen for serving on my thesis committee and providing me prospective in my work that I otherwise would not have had.

My parents have supported me financially and emotionally throughout my undergraduate career. They have also been a source of strength and sanity while going through the rollercoaster that is applying to graduate school. I would like to thank them for their patience when listening to me about my research and for allowing me opportunities to explore my passion.

The JMU Biology Department has been my home for the last 4 years and has given me many opportunities to grow as a student and as a member of society. I would like to acknowledge the department for providing the framework in which I grew during my undergraduate career and their support. Additionally, the JMU Honors College has provided opportunities for me to pursue my passion for research as well as expose me to challenging course work within other areas.

The Hatfull lab at the University of Pittsburgh has been instrumental in organizing the SEA-PHAGES program which generated the data for our database as well as in affirming my desire to pursue a graduate degree. The time I spent as part of the Hatfull lab was formative and allowed me to better understand my own academic goals.
I would like to thank the Howard Hughes Medical Institute for funding my research, as well as the Honors Advisory Council for funding the Hillcrest scholarship that allowed me to train in Dr. Hatfull's lab. The Madison Achievement Scholarship and Judith Ramsey Hoffman Science & Mathematics Scholarship have assisted me financially in pursuing my undergraduate degree.

The past and present undergraduate members of the Cresawn lab have been indispensable in providing feedback on Repeaterator. I would like to specifically thank Elizabeth Lucas, Nicole Roberto, Andrew Lambrechts, Rachael St. Jacques, Mia Wenzel, Kelly Degnon, Brie Lewis, and Kaleigh Jaeger for their comradery over the years and for making the Cresawn lab a wonderful group to be a part of. I would also like to acknowledge John Wallace for his contributions to the first attempt at Repeaterator. John’s creative spirit and willingness to try new approaches allowed us to navigate many obstacles and to bring the first version of Repeaterator from idea to working code.
Abstract

Horizontal gene transfer plays a large role in microbial genetic diversity. Bacteriophages can mediate diversity within their hosts through transduction, the uptake and dispersal of host DNA between bacterial hosts. However, bacteriophages themselves experience horizontal gene transfer through mobile genetic elements and recombination. Unlike their hosts, bacteriophages cannot easily be mapped onto a phylogenetic tree as they do not all possess a common trait like the 16s RNA gene. However, their genomes are typically small enough to be analyzed using tools such as Gepard and Phamerator that compare nucleotide identity across entire genomes. However, Gepard lacks the ability to contextualize the analysis with respect to annotated genes, and Phamerator, by its very nature as a comparison tool, cannot highlight repeats within a single genome. Many programs identify repeat motifs, but lack the ability to display genomic information and regard repeats in an isolated manner. To address this problem, I have developed Repeaterator, a tool to visualize DNA repeat motifs within Actinobacteriophage genomes. Instead of comparing multiple genomes, Repeaterator compares a genome to itself to map the occurrence of DNA repeat motifs in the context of gene annotations. Additionally, other genomic information can easily be overlaid on the visualization, including GC content or the strand the gene is coded on. Repeaterator provides powerful new insight into the evolutionary origins of Actinobacteriophage genomes and can easily be adapted to analyze other genomes. Repeaterator displays bacteriophage genomes and their annotations as an interactive data visualization. It uses the same underlying technologies as Phamerator itself and will be available along with our other tools on http://phamerator.org for general use.
Background

*Mycobacterium tuberculosis* is a gram-negative bacterium and the primary causative agent of pulmonary tuberculosis (TB), a respiratory disease characterized by chronic lung inflammation. The World Health Organization (WHO) estimates that 10.4 million people worldwide contracted TB in 2015, with 1.4 million of the cases being fatal (World Health Organization, 2015). A projected 480,000 cases were caused by multidrug-resistant (MDR) *M. tuberculosis*, a subset of strains that have resistance to at least two antibiotics (World Health Organization, 2015). Traditionally, short-course TB treatment consists of administration of multiple antibiotics; however, patient compliance with established protocols is poor (Caminero, 2006). Poor compliance can lead to the rise of MDR strains that can be spread to other patients. Globally, only 20% of MDR-TB cases were treated in 2015 (World Health Organization, 2015). Additionally, the most common vaccine for TB, the Bacillus Calmette–Guérin (BCG) vaccine, has highly variable efficacy and confers no immunity to some geographic populations (Mangtani et al., 2014). The problem of MDR-TB and other antibiotic resistant pathogens poses a sizable public health risk globally and has warranted a large amount of research in novel diagnostics and treatment.

Antibiotic resistance can be conferred through various mechanisms including inactivation of the antibiotic, efflux of the antibiotic from the cell, and mutation of target proteins favoring forms that have a lower binding affinity for the antibiotic (Lin et al., 2015). Often, genes that confer resistance phenotypes reside on mobile genetic elements and are selected against without the presence of an antibiotic. It is only due to selective pressure caused by antibiotics that these mobile genetic elements present the
host bacteria with a fitness advantage and are therefore maintained within the population. Due to the widespread availability of and demand for antibiotics, selective environments are facilitated by medical professionals who improperly prescribe antibiotic treatment and patients who do not properly comply with prescribed treatments (Rönnerstrand & Andersson Sundell, 2015). Reservoirs of antibiotic resistance genes can also be found in the environment due to lax use of antibiotics in agriculture and the large presence of antibiotics in hygiene products (Allen et al., 2010).

Through the propagation of antibiotic resistance genes in natural reservoirs, the chance of transferring resistance between bacterial species increases. Plasmids can be transferred directly between bacteria through conjugation and indirectly by transformation. Bacteriophages, also referred to as phages, can likewise mediate transduction of genetic material. Horizontal gene transfer further bolsters the threat of antibiotic resistance genes, potentially passing the effects of antibiotic selection from nonpathogenic bacteria to virulent strains (Allen et al., 2010).

Bacteriophages are a significant area of research interest (Citorik, et al., 2014). This interests stems from the realization that mechanisms that have evolved in bacteria and confer resistance to antibiotics do not also confer resistance to phage infection (Bondy-Denomy et al., 2016; Lin et al., 2015). Although resistance to phage infection is possible, phages are among the most abundant biological entities known, numbering an estimated $10^{31}$ phages in the biosphere (Rohwer & Edwards, 2002). They thus represent an essentially unlimited pool of anti-bacterial agents. As a population, phages exhibit extreme diversity of hosts and environments (Clokie, et al., 2011). These viruses infect bacteria and harness the cellular machinery to replicate the viral genome and
produce virions, or infectious viral particles. Phages can be generally classified into two categories based on replication strategy: lytic and temperate. Both types of phages initiate the infection their bacterial host in a similar manner; by recognizing a surface receptor on the host’s cell wall, attaching, and adsorbing into the host cytoplasm. Upon entry into the host cell, the phage genome circularizes to prevent degradation by exonucleases and enacts one of its replication strategies. Lytic phages immediately employ lytic growth: the suppression of host gene expression followed by mass replication, transcription, and translation of the viral genome to form new viral particles. Once the quantity of viral particles reaches a critical number, lysins are produced and lyse the host cell to release the newly formed virions into the environment (Sulakvelidze, et al., 2001). Temperate phages employ a different strategy, lysogenic growth. Lysogeny is characterized by the replication of the viral genome alongside the host genome upon host-mediated DNA replication. This can be achieved through integration of the phage genome into the host chromosome via phage encoded integrases to form a prophage, or by propagating extrachromosomally as a plasmid (Little, 2005). Upon an environmental signal, the prophage becomes active and excises from the host genome to enter lytic growth. Both lysis and lysogeny manipulate the host’s cellular functions at multiple points, demonstrating a close functional and evolutionary relationship between virus and host.

Just as bacteria are subject to horizontal gene transfer, so are the viruses that infect them. Transposons (Sampson et al., 2009), homing endonucleases (Pope et al., 2013), and direct DNA repeats are prevalent within phage genomes, causing insertions, deletions, and substitutions (Pope et al., 2011). Mycobacteriophage, the most...
characterized subset of actinobacteriophages, genomes are organized into 2 common modules, the 5’ “left arm” and the 3’ “right arm” (Pham, et al., 2007). The left arms contain large (>750 bp) genes encoding virion structural proteins and tends to follow a conserved synteny. In contrast, the right arms of these genomes encode many metabolic genes and genes of unknown function and tend to have a shorter mean length (~250 bp). Rampant genomic reorganization within the right arm of these genomes contributes to the mosaicism and makes genome scale evolutionary studies difficult (Smith et al., 2013). However, many recombination events introduce or remove DNA repeats into or from the genome. These genetic scars can then serve to promote further recombination events and can possibly be used to trace the evolutionary history of genetic elements and the phage genomes within which they reside.

Here I describe a program to visualize the occurrence and distribution of DNA repeat motifs within genomes within the Actinobacteriophage database. These data are displayed in the context of annotated genomic features as well to promote a functional understanding of the role DNA repeat motifs play in genome content and organization. This tool is associated with the Actinobacteriophage database and is available to analyze Actinobacteriophage genomes at http://phamerator.org.
Implementation

Architecture

Repeaterator is coded in the JavaScript programming language and is embedded within the Phamerator web application available at http://phamerator.org. Actinobacteriophage genomic data are stored in the Phamerator Mongo database and added to through interactions between Repeaterator and MEME (Figure 1). When a new genome is added to the Phamerator MySQL database, Repeaterator is triggered to send the genome sequence to MEME in order to identify DNA repeat motifs. MEME returns the top 10 motifs with the lowest expectancy (E) value. The E-value is an the number of expected motifs of the same width and site count, given a database of similarly sized random sequences (Bailey et al., 2009). The request made to MEME sets a maximum motif length of 25 nucleotides and maximum number of instances of 50. Individual instances of a motif may differ in nucleotide sequence, but deviations from the consensus sequence adversely affect the E-value of the motif. User input triggers Repeaterator to access the Phamerator Mongo database and to use the Data-Driven Documents library (http://d3.js.org) to represent genomic data as document object model (DOM) elements within a HTML document that is displayed on the screen (Figure 1).
Figure 1. Architecture of Repeaterator. Black arrows represent automated processes. White arrows indicate processes triggered by the user. (1) The Phamerator MySQL database is converted to a Mongo database by a script belonging to the Phamerator family of code. This process is not directly part of the Repeaterator family of code, but is necessary for proper implementation. (2) Repeaterator retrieves genome FASTA files from the Phamerator MySQL. (3) Repeaterator calls MEME for each new genome in the MySQL database and passes it a FASTA file. (4) MEME returns repeat results to Repeaterator in the form of an HTML file. (5) Repeaterator parses out relevant information and stores it in the Phamerator Mongo database. [A] The user requests a repeat map through the user interface. [B] Repeaterator retrieves genome data stored in the Phamerator Mongo database. [C] Repeaterator uses the D3.js library to convert genome stored in JSON to SVG DOM Nodes for visualization in the user interface.
Critical Issues

Phamerator diagrams are linear maps of genomes which themselves are physically linear molecules. However, this representation does not lend itself to the display of intra-genome sequence repeats. A circular diagram would solve this problem, but it produces a new issue; the diagram can mislead the reader to believe the genome is in fact a circular molecule. I addressed the issue by representing the genome as a 355° arc, rather than a closed circle, which expressed that the two ends of the genome are not connected.
Results and Discussion

User Interface

Repeaterator can be accessed by selecting the “Repeat Map” option from the navigation menu presented at the top or left of the screen (depending on screen size). Subsequently, a hierarchical list of phage (sub)clusters is presented, which can be expanded by clicking or tapping (on touchscreen devices) to reveal individual genomes within those groupings. Users may select one or more phages, automatically initiating the drawing of the repeat maps which are then accessed by selecting the “View Map” tab (Figure 2).
Figure 2. **Repeaterator user interface.** The interface is organized such that the most general buttons are placed at the top of the page. (1) denotes the navigation bar that is always visible, (2) denotes the tabs to toggle between selecting phage and viewing the map, (3) denotes the subcluster selection, (4) denotes individual phage selection, (5) denotes the map within the workspace, and (6) denotes the options menu. When the Select Phages tab is selected, layout A) is shown. When the View Map tab is selected, layout B) is shown.
Up to 10 repeat motifs will be displayed for each genome, with a maximum of 50 instances drawn for each motif. To allow for the study of a subset of these motifs, users may access a settings menu (Figure 3) and from there toggle the visibility of motifs individually.

![Figure 3. Repeaterator options.](image)

Repeat maps are interactive, with interactivity triggered by hovering the mouse pointer (or tapping on a touch screen device) over various on-screen elements. Specifically, information about each motif is accessible by hovering or tapping on the various colored arcs that connect motif instances. This information includes the motif identification number, length, number of occurrences, and E-value. Details about each motif instance can be accessed by hovering or tapping the green (forward) or red (reverse) dots arrayed along the innermost portion of the genome ruler. This will reveal details such as motif identification number, position in the genome, probability (P) value,
and DNA sequence (including both upstream and downstream flanking regions). Furthermore, gene annotations can be color-coded according to their direction of transcription or the gene phamily to which they belong.

**Comparisons to Existing Software**

Gepard is a Java program that allows the user to draw a dot plot comparing two or more nucleotide sequences (Figure 4) (Krumsiek, et al., 2007). The program is optimized to compare large sequences including eukaryotic chromosomes and whole bacterial genomes. Gepard can compare a genome to itself and reveal regions with DNA repeats, however it is difficult to identify the motif responsible and the exact genomic coordinates of the repeat. While this tool is useful for visualizing large scale genome reorganizations, it does not provide any specific data concerning repeats or genome annotations.
The program GenomeMatcher provides a variation of dot plot analysis with enhanced functionality (Ohtsubo, et al., 2008). In addition to drawing dot plots, GenomeMatcher also displays user-provided genome annotations along the axes. As such, the user can compare any differences or duplications to the annotations in that region. GenomeMatcher also provides an enhanced zoom feature that allows for easier analysis of small alignments. The program is limited though in the fact that it does little to classify or identify repeat motifs and relies on user interpretation of dot plots to draw conclusions.
MEME is a program that allows the user to input nucleic acid or protein sequences, customize search parameters, and then run the program to identify and display repeat motifs (Bailey et al., 2009). The output of MEME is an HTML file that shows the sequence, motif logo, location of repeat instances, and a variety of statistical measures of significance (Figure 5). MEME’s strength is in detection and classification of repeat motifs within a sequence, and while it draws linear maps of these repeats, it fails to consider or display any sort of genome annotations, making data interpretation difficult.

![Figure 5. Identification of repeat motifs for Olympic Helado using MEME. Motifs are arranged in rows vertically, from lowest to highest E-value. Logos are created based on the frequency of a given nucleotide in each position of the repeat motif.](image)

REPuter, like MEME, is software that takes user input of a DNA sequence and identifies repeat motifs within the sequence. Unlike MEME, the processing time of REPuter is substantially less, making analysis of large sets of genomes much easier (Kurtz et al., 2001). However, the output from REPuter is less informative than that of
MEME. Data can be exported as plain text or the recommended route of an interactive viewer called REPvis. Within REPvis, the user can see the location of repeat pairs, choose the size of the repeat to visualize, and export the text output (Figure 6). In spite of being able to visualize the location of repeats in the input sequence, no genome annotations are offered to display nor are the identities of the repeats offered.

![REPvis interactive Viewer](image)

**Figure 6. Identification and location of repeats for OlympicHelado using REPuter.** Repeats are represented by lines colored by length or repeat. The horizontal lines labeled forward above and below the repeats denote the forward coding strand of the OlympicHelado genome.

Skittle is a repeat visualization program that is intended to complement repeat prediction software (Seaman & Sanford, 2009). The display allows the user to recognize patterns in the DNA sequence selected through the creation of a 2-D color-coded map. While the interface makes the manual detection of repeats easier, it does little to verify the presence of a repeat and limits the user to predetermined genomes.
Phamerator is the most similar tool to Repeaterator as it is a web application designed to visualize actinobacteriophage genome annotations and compare nucleotide identity between genomes (Figure 7). Comparison of similar genomes can shed light on the presence of DNA repeats within a genome, but the program offers no option to compare a genome against itself. Additionally, Phamerator does not reveal the sequence of alignments that are visualized or coordinates to the nucleotide level, making determining the identity of a repeat difficult. As repeats are not the main focus of this visualization program, the display can become crowded with multiple repeats present.
Case Study

The 9 Actinobacteriophages of cluster BI1 infect Streptomyces spp. and have an average GC content of 59.5%. These phages all have genomes of similar length, ranging from 55,349 bp to 56,470 bp. Within cluster BI1 genomes, there is an array of 18 forward coding genes each between 200 bp and 250 bp in length, exemplified by phage OlympicHelado genes 28 through 45. Nucleotide sequence similarity is observed.
between genomes, as well as in the intergenic spaces in the same genome (Figure 7), suggesting the presence of a DNA repeat motif.

When visualized using Repeaterator, it is apparent that there are many dispersed repeats on both strands of the genome with seemingly even distribution (Figure 8). By deselecting all of the repeat motifs in the map settings and inspecting each motif individually, motifs 2 and 5 are localized to the region containing phage OlympicHelado genes 28 through 45 (Figure 9). Both motif 2 and 5 are unique (Figure 5), but occur in very similar positions of the genome, with each instance on the forward coding strand and a length of 25 nucleotides.

These repeats may represent the vestiges of recombination events, which is further supported by insertion and deletion events within this region of cluster BI1 actinobacteriophages (Figure 7). The localization of motifs 2 and 5 to 8 kb of the genome may also indicate the potential for these sequences to act as binding sites for regulatory proteins, but the consensus sequence of both motifs 2 and 5 deviate from the standard Shine-Dalgarno consensus sequence.

It is possible that the genes in this array have similar evolutionary origin and may have come into the genome at different times from the same source. Alternatively, the genes in this array could be duplications of the same gene that have diverged over time, however they are classified into different phamilies and demonstrate less than 50% identity between any of the members.
Figure 8. Repeat map of OlympicHelado generated using Repeaterator. Genome position is denoted by placement relative to the genome ruler. Numbers on the ruler represent nucleotide position in kilobases. Arrows outside the genome ruler represent the gene annotations within that genome and are colored by protein family. Circles on the inner edge of the genome ruler mark the position of repeats in the genome. Green and red circles indicate location on the forward coding and reverse coding strands, respectively. All instances of a given repeat motif are connected by a series of arcs. 10 repeat motifs are present on this map.
Future Development

Future development projects include the addition of more options to color-code gene-specific data. By diversifying the types of biological data that can be displayed in the repeat maps, users can make more meaningful predictions about the origin of genes within the genome. Additional data types include phamily abundance, phamily conservation among clusters, and protein domains.

Another functionality that would help in understanding the role of DNA repeat motifs in genome evolution would be the classification of repeat motifs into families.

Figure 9. Repeat maps of Olympic-Helado motifs 2 and 5. Genome coordinates 23.0 kb to 36.5 kb are shown. Both motifs are found only in this region of the genome. Motif 2 is shown in A) and motif 5 is shown in B).
Through this process, the presence of repeat families could be tracked through different genomes and shed light on the origins of possible horizontal gene transfer events. This endeavor could further be bolstered through the incorporation of data from a program like t2prhd, which tracks and predicts the evolutionary history of repeated DNA sequences (Sipos, et al., 2008).

For ease of viewing specific instances of a repeat, an integrated repeat map zoom feature will be developed. Instead of the traditional zoom that enlarges the existing image, the integrated feature will allow the user to specify the start and stop coordinates within the genome that they would like to view. The map will then resize appropriately, displaying genomic data within that region spread over the 355° genome ruler. As such, the user can examine specific regions of the genome if desired and can space out repeat instances.
Conclusions

Repeaterator provides a framework to better visualizing DNA repeat motifs within Actinobacteriophage genomes than existing programs do. Insight gleaned from Repeaterator allows users to better understand the conservation and distribution of DNA repeats within these genomes and can be applied to study the transmission of genes between phage genomes and genomic reorganization events.

Currently, the International Committee on Taxonomy of Viruses classifies viruses into families chiefly based on particle morphology and then subdivides families into subfamilies and genera based on genetic similarity to a representative genome. With the improvement of sequencing technologies and bioinformatics programs to support genome analysis, the tools are in place to characterize and classify phages using phylogenetics. As DNA repeats are characteristic of the horizontal gene transfer events that make traditional phylogeny difficult, repeat motifs may serve as a better alternative to map evolutionary relationships than morphology.

Repeaterator is currently only available to analyze Actinobacteriophage genomes, but is not inherently limited to such a data set. Given a properly structured Mongo database, it would be possible to visualize any single chromosome or plasmid using Repeaterator and to explore the occurrence and distribution of repeat motifs contextualized by genome annotations.
selectedGenomes2 = new Meteor.Collection(null);

var formRight = function() {return document.getElementById("start_coor")};
var formLeft = function() {return document.getElementById("end_coor")};

Template.repeats.onCreated(function() {
    Meteor.call('getclusters', function(error, result) {
        if (typeof error !== 'undefined') {
            /////console.log('error getting clusters:', error);
        } else {
            Session.set('myMethodResult', result);
        }
    });

    //in rendered callback
    var key = function(d) {
        return d.phagename;
    }

    drawRepeats = function (svg) {
        console.log("tracker autorun has rerun");
        //d3.selectAll("#mappy").remove();

        svg.attr("height", function(d) {return (selectedGenomes2.find().count() * 750) });
        svg.attr("width", function (d) {
            return 1000
        });

        // Define the div for the tooltip
        var div = d3.select("body").append("div")
            .attr("class", "tooltip")
            .style("opacity", 0);

        var divr = d3.select("body").append("div")
            .attr("class", "tooltip")
            .style("opacity", 0);

        var divm = d3.select("body").append("div")
            .attr("class", "tooltip")
            .style("opacity", 0);

        //d3.select('.determinate').style("width", "0%");

        function x() {
            console.log(d3.select(this).data());
        }
    }
//bind the data
var phage1 = svg.selectAll(".phages")
  .data(function() {
    pnames = selectedGenomes2.find({}, {phagename: 1}).fetch().map(function(obj) { return obj.phagename; });
    phages = Genomes.find({phagename: {$in: pnames}}, {sort: {cluster: 1, phagename: 1}});
    //todo: get selected primary and secondary sort fields and ascending/descending
    return phages.fetch();
  }, key);

//move the images over
phage1
  .attr("transform", function (d, i) {
    return "translate(400, " + ((i * 750)+400) + ")";
  });

//enter new images and move appropriately
var newPhage = phage1.enter().append("g")
  .classed("phages", true)
  .attr("transform", function (d, i) {
    return "translate(400, " + ((i * 750)+400) + ")";
  });

//remnant of old design
  /*var keystone = d3.svg.arc()
   .innerRadius(230)
   .outerRadius(250)
   .startAngle(-0.0436332)
   .endAngle(0.0436332);*/

//add in phage name
newPhage.append("text")
  .attr("x", -400)
  .attr("y", -300)
  .attr("font-family", "sans-serif")
  .attr("font-size", "24px")
  .attr("fill", "black")
  .style("text-anchor", "left")
  .text(function (d) {
    return d.phagename;
  })
  .attr("fill-opacity": 1));

//FIX THIS NEED DATA BIND
  /* function genlen() {return
    selectedGenomes2.findOne(phageName).genomelength}
    var genleng = genlen();
    console.log(genleng);*/

var slider = document.getElementById('test5');
noUiSlider.create(slider, {
var range = { 'min': 0, 'max': 100 },
format: wNumb({ decimals: 0 })
});

/set parameters for gene arcs
var arcg = d3.svg.arc()
  .innerRadius(function(d){if (d.name%2==0)
    return 290;
  else if (d.name%2==1)
    return 250;})
  .outerRadius(function(d){if (d.name%2==0)
    return 330;
  else if (d.name%2==1)
    return 290;})
  .startAngle(function(d){
    return ((d.start/this.parentNode.__data__.genomelength)*6.2)+0.0436332;
  })
  .endAngle(function(d){
    return ((d.stop/this.parentNode.__data__.genomelength)*6.2)+0.0436332;
  });

//bind gene data and draw gene arcs
newPhage.selectAll(".paths")
  .data(function(d){return d.genes;})
  .enter()
  .append("svg:path")
  .attr("fill", function(d){
    return d.phamColor;
  })
  .style("stroke", "black")
  .attr("d", arcg)
  .attr("class", "garcs")
  .style("opacity", 1)
  .on("mouseover.arc", function() {
    d3.select(this).style("stroke", "#ffb973")
    .style("stroke-width", 2)
  })
  .on("mouseout.arc", function() {
    d3.select(this).style("stroke", "#000")
  })
  .on("mouseover.tip", function(d) {
    nodedata = this.parentNode.__data__;
    div.transition()
      .duration(500)
      .style("opacity", .9);
    div.html(nodedata.phagename + " gp" + d.name)
  });

  // the text of the tooltip ...
  .style("left", (d3.event.pageX) + "px")
  .style("top", (d3.event.pageY - 28) + "px");
}
document.getElementById("Phamily").onclick = function(d) {
    d3.selectAll(".garcs")
        .transition().duration(1000)
        .attr("fill", function (d) {
            return d.phamColor;
        })
        .style("fill-opacity", 1)
;
}

//color gene arcs by direction
document.getElementById("Direction").onclick = function(d) {
    d3.selectAll(".garcs")
        .transition().duration(1000)
        .attr("fill", function (d) {
            if (d.direction === "forward") {
                return "green"
            } else if (d.direction === "reverse") {
                return "red"
            } else {return "black"}
        })
        .style("fill-opacity", 1)
;
}

document.getElementById("%GC").onclick = function(d) {
    d3.selectAll(".garcs")
        .transition().duration(1000)
        .attr("fill", function (d) {
            return "black";
        })
        .style("fill-opacity", function(d){
            return Math.abs(Math.log((((this.parentNode.__data__.sequence.slice(d.start, d.stop+1)).match(/G/gi).length + (this.parentNode.__data__.sequence.slice(d.start, d.stop+1)).match(/C/gi).length) / (this.parentNode.__data__.sequence.slice(d.start, d.stop+1)).length)));
        });
};

console.log(Math.abs(Math.log((((this.parentNode.__data__.sequence.slice(d.start, d.stop+1)).match(/G/gi).length + (this.parentNode.__data__.sequence.slice(d.start, d.stop+1)).match(/C/gi).length) / (this.parentNode.__data__.sequence.slice(d.start, d.stop+1)).length))));

return Math.abs(Math.log((((this.parentNode.__data__.sequence.slice(d.start, d.stop+1)).match(/G/gi).length + (this.parentNode.__data__.sequence.slice(d.start, d.stop+1)).match(/C/gi).length) / (this.parentNode.__data__.sequence.slice(d.start, d.stop+1)).length))));

//color gene arcs by Pham Color
d.stop+1)).match(/C/gi).length) /
(this.parentNode.__data__.sequence.slice(d.start, d.stop+1)).length))

};

//set parameters for the ruler arc
var arc = d3.svg.arc()
  .innerRadius(230)
  .outerRadius(250)
  .startAngle(0.0436332)
  .endAngle(2*Math.PI - 0.0436332);

//draw the ruler arc
newPhage.append("path")
  .attr("d", arc)
  .style({ fill: "white", opacity: 1})
  .attr({"stroke": "black"})
  .attr({"stroke-width": 2});

//remnant of old style
/* Keystone
newPhage.append("path")
  .attr("d", keystone)
  .attr("fill", "white"); */

//Add code for line ticks/*
var r = d3.scale.linear()
  .domain([0, 1])
  .range([0, 200]);

//default accessor [[x1,y1]] => radian and angle
var line = d3.svg.line.radial()
  .radius(function(d) { return (r(d[1])); })  // will change between -1 and 1
  .angle(function(d) { return d[0];});

// radius axis
// cheat with CSS
/* var gr = newPhage.append("g")
  .attr("class", "r axis")
  .selectAll("g")
  .data(r.ticks(10).slice(0))
  .enter().append("g");
 +*/

//set parameters for thousand bp tick marks
var thoutick = newPhage.append("g")
  .attr("stroke", "black")
  .selectAll("g")
  .data(function(d){return d3.range(2.5, 357.5,
    (1000/d.genomelength)*355);})//replace 80000 with genome length
  .enter().append("g")
  .attr("transform", function(d) {
return "rotate(" + (d - 90) + ")"; }

// set parameters for 500 bp tick marks
var fhuntick = newPhage.append("g")
  .attr("stroke", "black")
  .selectAll("g")
  .data(function(d) { return d3.range(2.5, 357.5, (500/d.genomelength)*355); })// replace 80000 with genome length
  .enter().append("g")
  .attr("transform", function(d) {
    return "rotate(" + (d - 90) + ")"; });

// set parameters for 5000 bp tick marks
var fthotick = newPhage.append("g")
  .attr("stroke", "black")
  .selectAll("g")
  .data(function(d) { return d3.range(2.5, 357.5, (5000/d.genomelength)*355); })// replace 80000 with genome length
  .enter().append("g")
  .attr("transform", function(d) {
    return "rotate(" + (d - 90) + ")"; });

// draw 5000 bp tick marks
fthotick.append("line")
  .attr("x2", 250);

// draw 500 bp tick marks
fhuntick.append("line")
  .attr("x2", 235)
  .style({opacity: 1});

// draw 1000 bp tick marks
thoutick.append("line")
  .attr("x2", 240)
  .style({opacity: 1});

// draw circle to cover radial line and to draw repeats on
newPhage.append("circle")
  .attr({cx: 0, cy: 0, r: 229.1})
  .attr({stroke: "none"})
  .style({fill: "white", opacity: 1});

// rounds a number to nearest increment of 5
var roundToFive = function (x) { return (x % 5) >= 2.5 ? parseInt(x / 5) * 5 + 5 : parseInt(x / 5) * 5;};

// add text to 5000 bp tick marks
fthotick.append("text")
  .attr("x", 240)
  .attr("dy", ".85em")
  .style("text-anchor", "middle")
  .text(function(d) { return roundToFive(((d - 334)/355)*(this.parentNode.parentNode.__data__.genomelength/1000)); })// Replace 80000 with genome length
  .style("fill", "green")
  .attr({stroke: "none"})
  .attr({"font-size": "14px", "font-family": "Arial"})
var randomColor = (function() {
  var golden_ratio_conjugate = 0.618033988749895;
  var h = Math.random();

  var hslToRgb = function (h, s, l) {
    var r, g, b;
    if(s == 0) {
      r = g = b = l; // achromatic
    } else {
      function hue2rgb(p, q, t) {
        if(t < 0) t += 1;
        if(t > 1) t -= 1;
        if(t < 1/6) return p + (q - p) * 6 * t;
        if(t < 1/2) return q;
        if(t < 2/3) return p + (q - p) * (2/3 - t) * 6;
        return p;
      }
      var q = 1 < 0.5 ? 1 * (1 + s) : 1 + s - 1 * s;
      var p = 2 * 1 - q;
      r = hue2rgb(p, q, h + 1/3);
      g = hue2rgb(p, q, h);
      b = hue2rgb(p, q, h - 1/3);
    }
    return '#' + Math.round(r * 255).toString(16) + Math.round(g * 255).toString(16) + Math.round(b * 255).toString(16);
  }
  return function() {
    h += golden_ratio_conjugate;
    h %= 1;
    return hslToRgb(h, 0.5, 0.60);
  }
})();

var c10 = d3.scale.category10();

function colores_google(n) {
  var colores_g = ['#3366cc', '#dc3912', '#ff9900', '#109618',
    '#990099', '#0099c6', '#dd4477', '#66aa00', '#b8e2e2', '#316395', '#994499',
    '#22a999', '#aaaa11', '#6633cc', '#e67300', '#8b0707', '#651067', '#329262',
    '#5574a6', '#3b3eac'];
  return colores_g[n % colores_g.length];
}

var connectfunc = d3.svg.line()
  .x(function (d) {
    console.log(genomelength);
    return 230 * (Math.sin((d.pos / genomelength) * 6.19592) + ((2.5*Math.PI)/180));
  })
  .y(function (d) {
return -230 * (Math.cos(((d.pos / genomelength) * 6.19592) + ((2.5*Math.PI)/180)));
// Need to make hard coded length reactive
var xCirclePlace = function(d) {
    return 230 * (Math.sin(((d.pos / 56189) * ((355*Math.PI)/180)) +
    ((2.5*Math.PI)/180)));
};

var yCirclePlace = function (d) {
    return -230 * (Math.cos(((d.pos / 56189) / d.sequence_db.sequences.length) * ((355*Math.PI)/180)) +
    ((2.5*Math.PI)/180));
};

var site = reps.selectAll(".site")
    .data(function(d) { return d.motifs })
    .enter()
    .append("g")
    .each(function () {
        var sit = d3.select(this);
        sit.selectAll("sit")
            .data(function(d) { return d.sites; }) // second level data-join
            .enter()
            .append("circle")
            .attr("class", function (d) {return "motif" +
            this.parentNode.__data___.id})
            .attr("cx", function(d) {return xCirclePlace(d)})
            .attr("cy", function(d) {return yCirclePlace(d)})
            .attr("r", 2)
            .attr("fill", function(d) { if (d.rc === false) {
                return "green"
            } else if (d.rc === true) {
                return "red"
            } else{
                return "black"
        }
    })
    .on('mouseover.r', function (d) {
        d3.select(this).attr("stroke", "black").attr("r", "4")
    })
    .on("mouseover.tip", function(d) {
        // node data = this.parentNode.__data__;
        divr.transition()
            .duration(500)
            .style("opacity", .9);
        divr.html("ID: " + this.parentNode.__data___.id +
        "<br>" + "Position: " + d.pos + "<br>" + "P-Value: " + d.pvalue + "<br>"
+ "5\'-"+d.lflank + "<strong style='color:#FCB674'>" + "    " + d.match + 
"</strong>" + "    " + d.rflank + "-3\'\")

// the text of the tooltip ...
.style("left", (d3.event.pageX + 25) + "px")
.style("top", (d3.event.pageY - 28) + "px");
})
.on('mouseout.r', function (d) {
    d3.select(this).attr("stroke", "none").attr("r", "2")
})
.on("mouseout.tip", function(d) {
    divr.transition()
        .duration(500)
        .style("opacity", 0))
    ;
})

document.getElementById("Motif1").onclick = function(d) {
    if (document.getElementById("Motif1").checked === true) {
        d3.selectAll(".motif1")
            .style("visibility", "visible")
    } else if (document.getElementById("Motif1").checked === false){
        d3.selectAll(".motif1")
            .style("visibility", "hidden")
    }
}

document.getElementById("Motif2").onclick = function(d) {
    if (document.getElementById("Motif2").checked === true) {
        d3.selectAll(".motif2")
            .style("visibility", "visible")
    } else if (document.getElementById("Motif2").checked === false){
        d3.selectAll(".motif2")
            .style("visibility", "hidden")
    }
}

document.getElementById("Motif3").onclick = function(d) {
    if (document.getElementById("Motif3").checked === true) {
        d3.selectAll(".motif3")
            .style("visibility", "visible")
    } else if (document.getElementById("Motif3").checked === false){
        d3.selectAll(".motif3")
            .style("visibility", "hidden")
    }
}

document.getElementById("Motif4").onclick = function(d) {
    if (document.getElementById("Motif4").checked === true) {
        d3.selectAll(".motif4")
            .style("visibility", "visible")
    } else if (document.getElementById("Motif4").checked === false){
        d3.selectAll(".motif4")
            .style("visibility", "hidden")
    }
}
d3.selectAll(".motif4")
    .style("visibility", "hidden")
}

document.getElementById("Motif5").onclick = function(d) {
    if (document.getElementById("Motif5").checked === true) {
        d3.selectAll(".motif5")
            .style("visibility", "visible")
    } else if (document.getElementById("Motif5").checked === false){
        d3.selectAll(".motif5")
            .style("visibility", "hidden")
    }
}

document.getElementById("Motif6").onclick = function(d) {
    if (document.getElementById("Motif6").checked === true) {
        d3.selectAll(".motif6")
            .style("visibility", "visible")
    } else if (document.getElementById("Motif6").checked === false){
        d3.selectAll(".motif6")
            .style("visibility", "hidden")
    }
}

document.getElementById("Motif7").onclick = function(d) {
    if (document.getElementById("Motif7").checked === true) {
        d3.selectAll(".motif7")
            .style("visibility", "visible")
    } else if (document.getElementById("Motif7").checked === false){
        d3.selectAll(".motif7")
            .style("visibility", "hidden")
    }
}

document.getElementById("Motif8").onclick = function(d) {
    if (document.getElementById("Motif8").checked === true) {
        d3.selectAll(".motif8")
            .style("visibility", "visible")
    } else if (document.getElementById("Motif8").checked === false){
        d3.selectAll(".motif8")
            .style("visibility", "hidden")
    }
}

document.getElementById("Motif9").onclick = function(d) {
    if (document.getElementById("Motif9").checked === true) {
        d3.selectAll(".motif9")
            .style("visibility", "visible")
    } else if (document.getElementById("Motif9").checked === false){
        d3.selectAll(".motif9")
            .style("visibility", "hidden")
    }
document.getElementById("Motif10").onclick = function(d) {
    if (document.getElementById("Motif10").checked === true) {
        d3.selectAll("." + "motif10")
            .style("visibility", "visible")
    }
    else if (document.getElementById("Motif10").checked === false)
    {
        d3.selectAll("." + "motif10")
            .style("visibility", "hidden")
    }
};

// gets rid of phage from selection
phage1.exit().remove();

$('#preloader').fadeOut(300).hide();

Template.repeats.onRendered(function () {
    console.log('phages rendered');

    $('#preloader').fadeOut(300).hide();

    $(document).ready(function() {
        $('ul.tabs').tabs();
    });

    $('.collapsible').collapsible({
        accordion : false // A setting that changes the collapsible behavior
        to expandable instead of the default accordion style
    });

    $(document).ready(function() {
        // the "href" attribute of .modal-trigger must specify the modal ID
        that wants to be triggered
        $('.modal-trigger').leanModal();
    });

    $('.dropdown-button').dropdown({
        inDuration: 300,
        outDuration: 225,
        constrain_width: false, // Does not change width of dropdown to
        that of the activator
        hover: true, // Activate on hover
        gutter: 0, // Spacing from edge
        belowOrigin: false, // Displays dropdown below the button
        stoppropagation: true,
        alignment: 'left' // Displays dropdown with edge aligned to the
        left of button
    });

    svg = d3.select("#repeat-map")
        .append("svg");
svg.attr("id", "svg-repeat-map")
   .attr("border", "5px");
Tracker.autorun(function () {
   drawRepeats(svg);
});

updateSessionStore = function () {
   console.log('updating selected data');
   //console.log('names:', selectedGenomes.find({}, {fields: {phagename: 1}}).fetch().map(function (p) {return p.phagename;}));
   Meteor.user().selectedData['repeatMaps'] = selectedGenomes2.find({},
   {fields: {phagename: 1}}).fetch().map(function (p) {return p.phagename;});
};

Template.repeats.helpers({
   clusters: function() {
      return Session.get('myMethodResult');
   },
   selectedGenomes2: selectedGenomes2,
   selectedGene: function () { return Session.get('selectedGene'); },
   genomes_are_selected: function() {
      return selectedGenomes2.find({}).fetch().length > 0;
   }
});

Template.repeats.events({
   "change .clusterCheckbox": function (event, template) {
      console.log("event", event.target.checked);
      //console.log(selectedGenomes.find().count());
      if (event.target.checked) {Meteor.subscribe("genomesWithSeq", clusterPhageNames, {
         onReady: function () {
            clusterGenomes = Genomes.find({cluster: event.target.getAttribute("data-cluster"), subcluster: event.target.getAttribute("data-subcluster")}).fetch();
            console.log(clusterGenomes);
         }
      })
      if (event.target.checked) {
         clusterGenomes = Genomes.find({cluster: ", subcluster: "}).fetch();
         clusterPhageNames = clusterGenomes.map(function (obj) {return obj.phagename});
         Meteor.subscribe("genomesWithSeq", clusterPhageNames, {
            onReady: function () {
               clusterGenomes = Genomes.find({cluster: event.target.getAttribute("data-cluster"), subcluster: event.target.getAttribute("data-subcluster")}).fetch();
               console.log(clusterGenomes);
            }
         })
         if (event.target.checked) {
         }
      }
   });
"preloader").show(function () {
   if (event.target.id !== "Singletons") {
      clusterGenomes = Genomes.find({cluster: event.target.getAttribute("data-cluster"), subcluster: event.target.getAttribute("data-subcluster")}).fetch();
   } else {
      clusterGenomes = Genomes.find({cluster: "", subcluster: ",subcluster: "}).fetch();
      clusterPhageNames = clusterGenomes.map(function (obj) {return obj.phagename});
      Meteor.subscribe("genomesWithSeq", clusterPhageNames, {
         onReady: function () {
            clusterGenomes = Genomes.find({cluster: event.target.getAttribute("data-cluster"), subcluster: event.target.getAttribute("data-subcluster")}).fetch();
            console.log(clusterGenomes);
         }
      })
      if (event.target.checked) {
      }
   });

console.log("cluster checkbox checked: ", event.target.id);
});

("#preloader").show(function () {
   if (event.target.id !== "Singletons") {
      clusterGenomes = Genomes.find({cluster: event.target.getAttribute("data-cluster"), subcluster: event.target.getAttribute("data-subcluster")}).fetch();
   } else {
      clusterGenomes = Genomes.find({cluster: "", subcluster: "",subcluster: "}).fetch();
      clusterPhageNames = clusterGenomes.map(function (obj) {return obj.phagename});
      Meteor.subscribe("genomesWithSeq", clusterPhageNames, {
         onReady: function () {
            clusterGenomes = Genomes.find({cluster: event.target.getAttribute("data-cluster"), subcluster: event.target.getAttribute("data-subcluster")}).fetch();
            console.log(clusterGenomes);
         }
      })
      if (event.target.checked) {
      }
   });
});

console.log('names:', selectedGenomes.find({}, {fields: {phagename: 1}}).fetch().map(function (p) {return p.phagename;}));
clusterGenomes.forEach(function (element, index, array) {
    console.log("getting sequence for", element);
    selectedGenomes2.upsert({phagename: element.phagename}, {
        phagename: element.phagename,
        genomelength: element.genomelength,
        sequence: element.sequence,
        cluster: element.cluster,
        subcluster: element.subcluster
    }, function () {
        //Meteor.call('updateSelectedData', element.phagename, true);
    });
});

else {
    clusterGenomes.forEach( function (element, index, array) {
        console.log('removing', element.phagename);
        selectedGenomes2.remove({phagename: element.phagename}, function () {
        });
    });
});

},
"change .phageCheckbox": function (event, template) {
    $('#preloader').show(function () {
        // get a list of all phagenames on the client
        phagename = event.target.id.split("-")[0];
        console.log(event);
        //Session.set("selections", selections++);

        // if user just selected a phage, it doesn't yet exist on the
        client but should
        Meteor.subscribe("genomesWithSeq", [phagename], {
            onReady: function () {
                if (event.target.checked) {
                    console.log(phagename, 'was selected');
                    q = Genomes.findOne({phagename: phagename});
                    selectedGenomes2.upsert({phagename: q.phagename}, {
                        phagename: q.phagename,
                        genomelength: q.genomelength,
                        sequence: q.sequence,
                        cluster: q.cluster,
                        subcluster: q.subcluster
                    }, function () { /*Meteor.call('updateSelectedData',
                        phagename, true); */});
                }
            }
        });
    });
},
"change .phageCheckbox": function (event, template) {
    $('#preloader').show(function () {
        // get a list of all phagenames on the client
        phagename = event.target.id.split("-")[0];
        console.log(event);
        //Session.set("selections", selections++);

        // if user just selected a phage, it doesn't yet exist on the
        client but should
        Meteor.subscribe("genomesWithSeq", [phagename], {
            onReady: function () {
                if (event.target.checked) {
                    console.log(phagename, 'was selected');
                    q = Genomes.findOne({phagename: phagename});
                    selectedGenomes2.upsert({phagename: q.phagename}, {
                        phagename: q.phagename,
                        genomelength: q.genomelength,
                        sequence: q.sequence,
                        cluster: q.cluster,
                        subcluster: q.subcluster
                    }, function () { /*Meteor.call('updateSelectedData',
                        phagename, true); */});
                }
            }
        });
    });
if user just unselected a phage, it exists on the
client but shouldn't

else {
    console.log(phagename, 'was unselected');

    selectedGenomes2.remove({"phagename":phagename},function () {
        //Meteor.call('updateSelectedData',
        phagename, false);
        
    });

} 

"click .downloadRepeatMap": function (event, template) {
    console.log("downloadGenomeMap clicked");
    
    svg = $("svg").attr({
        version: '1.1',
        xmlns: 'http://www.w3.org/2000/svg'
    });

    svgData = $("svg-repeat-map")[0].outerHTML;
    svgBlob = new Blob([svgData], {type:"image/svg+xml;charset=utf-8"});
    svgUrl = URL.createObjectURL(svgBlob);
    downloadLink = document.createElement("a");
    downloadLink.href = svgUrl;
    downloadLink.download = "repeaterator_map.svg";
    document.body.appendChild(downloadLink);
    downloadLink.click();
    document.body.removeChild(downloadLink);
}

"click #clearSelection": function (event, template) {
    console.log("clearSelection clicked");
    d3.select("#clearSelection")
        .transition()
        .duration(500)
        .style("opacity", 0).each("end", function () {
            selectedGenomes2.remove({});
            //Meteor.call('updateSelectedData', '', true);
        });
}

Template.registerHelper('clusterIsChecked',function(cluster, subcluster) {
    if (input === "Singletons") { input = ""; }
    phagesInCluster = Genomes.find({cluster: cluster, subcluster: subcluster}, {fields: {"phagename": 1}}).fetch();
    r = true;
    phagesInCluster.forEach(function (phage, phageIndex, myPhageArray) {
        if (selectedGenomes2.find({"phagename": phage.phagename}).count() ==
        0) {
            r = false;
        }
    });
Template.registerHelper('phageIsChecked2', function(input) {
    return selectedGenomes2.find({"phagename": input}).count() > 0;
});

Template.cluster2.helpers({
    selectedCount: function (cluster, subcluster) {
        count = selectedGenomes2.find({cluster: cluster, subcluster: subcluster}).count();
        if (count === 0) {
            return "";
        }
        return count;
    },
    selectedClass: function (cluster, subcluster) {
        count = selectedGenomes2.find({cluster: cluster, subcluster: subcluster}).count();
        if (count === 0) {
            return "badge";
        }
        return "new badge";
    },
    dataBadgeCaption: function (cluster, subcluster) {
        count = selectedGenomes2.find({cluster: cluster, subcluster: subcluster}).count();
        if (count === 0) {
            return "";
        } else if (count === 1) {
            return "selected genome";
        }
        return "selected genomes";
    }
});
Appendix II: Repeatorator HTML

```html
<template name="repeats">
  <div id="geneData" class="modal modal-fixed-footer">
    <div class="modal-content">
      <span id="gene_dna_seq" style="width:90%; word-wrap:break-word; display:inline-block;">(upgrading)</span>
    </div>
    <div class="modal-footer">
      <button class="waves-effect waves-green btn-flat btn-copy-link" data-clipboard-target="#gene_dna_seq">Copy to Clipboard</button>
      <a href="repeats" class="modal-action modal-close waves-effect waves-green btn-flat">Close</a>
    </div>
  </div>

  <div id="mapsettings" class="modal">
    <div class="modal-content">
      <div class="col s12 m6 l4">
        <h5>Repeat Map Settings</h5>
        <table>
          <tr>
            <td>Show:</td>
            <td><label><input type="checkbox" id="Motif1" checked><span class="lever"></span></label></td>
            <td>Motif 1</td>
          </tr>
          <tr>
            <td rowspan = 4>
              <form action="#">
                <p><input name="colorBy" type="radio" id="Phamily" checked /></p>
                <label for="Phamily">Phamily</label>
              </form>
            </td>
            <td rowspan = 4>
              <form action="#">
                <p><input name="colorBy" type="radio" id="Phamily Abundance" /></p>
                <label for="Phamily Abundance">Phamily Abundance</label>
              </form>
            </td>
            <td rowspan = 4>
              <form action="#">
                <p><input name="colorBy" type="radio" id="Cluster Conservation" /></p>
                <label for="Cluster Conservation">Cluster Conservation</label>
              </form>
            </td>
          </tr>
        </table>
      </div>
    </div>
  </div>
</template>
```
<label for="Cluster Conservation">Cluster Conservation</label>  
</p-->  
<p>  
<input name="colorBy" type="radio" id="%GC" />  
<label for="%GC">%GC</label>  
</p>  
<p>  
<input name="colorBy" type="radio" id="Direction" />  
<label for="Direction">Direction</label>  
</p>  
<form>  
</form>  
</td>  
</tr>  
<tr>  
<td colspan = 1 rowspan = 1>  
<label>  
<input type="checkbox" id="Motif2" checked>  
<span class="lever"></span>  
</label>  
</td>  
<td>Motif 2</td>  
</div>  
</tr>  
<tr>  
<td colspan = 1 rowspan = 1>  
<label>  
<input type="checkbox" id="Motif3" checked>  
<span class="lever"></span>  
</label>  
</td>  
<td>Motif 3</td>  
</div>  
</tr>  
<tr>  
<td colspan = 1 rowspan = 1>  
<label>  
<input type="checkbox" id="Motif4" checked>  
<span class="lever"></span>  
</label>  
</td>  
<td>Motif 4</td>  
</div>  
</tr>  
</table>
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Motif 5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Motif 6</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Motif 7</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Motif 8</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Motif 9</td>
<td></td>
</tr>
</tbody>
</table>
<label for="(upgrading)">
  (upgrading)
</label>
<span class="(upgrading)" data-badge-caption="(upgrading)">(upgrading)</span>
</div>
<div class="collapsible-body">
  <div class="row">
    <div class="col s12 m6 l4">
      <input type="checkbox" id="(upgrading)" class="filled-in phageCheckbox" checked="(upgrading)"/>
      <label for="(upgrading)">
        (upgrading)
      </label>
    </div>
    <div class="col s12 m6 l4">
      <input type="checkbox" id="(upgrading)" class="filled-in phageCheckbox"/>
      <label for="(upgrading)">
        (upgrading)
      </label>
    </div>
    <div class="col s12 m6 l4">
      <input type="checkbox" id="(upgrading)" class="filled-in phageCheckbox"/>
      <label for="(upgrading)">
        (upgrading)
      </label>
    </div>
  </div>
</div>
References


American Society of Microbiology. doi:10.1128/9781555816506.ch3


and construction of compatible site-specific integration vectors for mycobacteria.

*Microbiology, 153*(8), 2711-2723. doi:10.1099/mic.0.2007/008904-0


