Spring 2018

Enhancing the sea-phages app to collect weather-related metadata from soil sampling locations

Kelly M. Degnon
James Madison University

Follow this and additional works at: https://commons.lib.jmu.edu/honors201019

Part of the Virology Commons

Recommended Citation
https://commons.libjmu.edu/honors201019/539
Enhancing the SEA-PHAGES App to Collect Weather-Related Metadata from Soil Sampling Locations

An Honors College Project Presented to the Faculty of the Undergraduate College of Science and Mathematics James Madison University

by Kelly Marie Degnon

May 2018

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

FACULTY COMMITTEE: HONORS COLLEGE APPROVAL:

Project Advisor: Steven Cresawn, Ph.D. Bradley R. Newcomer, Ph.D.,
Associate Professor, Biology Dean, Honors College

Reader: Elizabeth Doyle, Ph.D.
Lecturer, Biology

Reader: Pradeep Vasudevan, Ph.D.
Assistant Professor, Biology

PUBLIC PRESENTATION

This work is accepted for presentation, in part or in full, at the JMU Honors Symposium on April 18, 2018.
Abstract

The bacteriophage population is extremely diverse and abundant, with a seemingly limitless number of phages that have yet to be isolated and sequenced. The SEA-PHAGES program discovers and completely sequences the genome of hundreds of phages every year, contributing to the ever-growing database of phage sequences. However, there is little data about the conditions in which these phages are collected and if weather conditions can have an effect on the likelihood of isolating a phage. The SEA-PHAGES app, when used by students in the SEA-PHAGES program, will collect data on the location and weather conditions in which a soil sample was collected. These data could provide information in the future about the relationship between the weather and the bacteriophage community in soil.

Introduction

Bacteriophages

Bacteriophages, or viruses that infect bacteria, are ubiquitous and incredibly diverse. There are approximately $10^{31}$ bacteriophages globally, which turn over every few days. It is estimated that phage populations exist at a concentration of $10^9$ phages per gram of soil (Hatfull, 2015). Phages reproduce by utilizing their host bacterial cell’s resources and can undergo two different lifecycles: lytic or lysogenic. In both lifecycles, a phage infects the host bacterial cell and inhibits its metabolism. In the lytic lifecycle, the phages undergo reproduction within the host, then the viruses lyse the cell and the phages are released. However, in the lysogenic lifecycle, the phage normally integrates its DNA into the host cell’s DNA, then remains a part of the bacterial cell until environmental conditions are favorable for the viral replication and cellular lysis (Kazi & Annapure, 2016). Globally, there are an estimated $10^{23}$ phage infections.
per second, which displays how incredibly active these viruses are in microbial ecology (Hatfull & Hendrix, 2012).

The genomes of phages are relatively small, ranging in length from ~3.3 kbp to ~500 kbp. These genomes are mosaic, meaning that they are unique combinations shaped by horizontal gene transfer. Phages that are not closely related can share similar nucleotide segments, as their ancestors may have exchanged fragments of their DNA. This mosaicism may allow for phages to infect a bacterial host that is different from their “natural” host. Because of this mosaicism, it is difficult to track the evolutionary history of phages and to categorize them phylogenetically (Hatfull & Hendrix, 2012). Phages are categorized into clusters based on how closely related they are and then their genome information is stored online. The mosaicism of phage genomes and the alignment of one phage genome to anther can be displayed using Phamerator.org (Cresawn, 2017). Phamerator.org, the SEA-PHAGES comparative genomics platform written and maintained by the Cresawn lab at James Madison University, uses colorful columns to display which nucleotide segments align closely to segments of another phage genome (Figure 1).
Figure 1: A visualization of the genome alignment of phages Greg, BobSwaget, and Trixie, belonging to clusters A1, A2, and A2 respectively (Cresawn, 2017). Nucleotide segments that align closely to segments in another phage are shown as colorful columns that connect one genome map to another. The color spectrum represents pairwise BLAST-N E-values ranging from 0 (violet) to $10^{-4}$ (red).

Approximately 70% of sequenced phages infect one of 12 widely studied bacterial hosts. Because only a limited number of bacterial hosts have been extensively studied, this often leads to phages which have extremely similar genomes, as they possess the same type of genes which allow them to target a specific host. The current database of phages is not an accurate representation of the phage population as a whole due to this limit in sampling as well as the inability for many phages to be isolated and studied using typical laboratory approaches. In these cases, the viral genome must be studied using viral metagenomics or by searching for prophages, the phage genome integrated into the host genome, within the genetic material of a bacterial sample.
SEA-PHAGES Program

The Howard Hughes Medical Institute’s Science Education Alliance works alongside the Hatfull lab at the University of Pittsburgh and the Cresawn lab at James Madison University to facilitate a nationwide program which immerses undergraduate students in microbiologic and genomic research. This program, known as the Science Education Alliance-Phage Hunters Advancing Genomics and Evolutionary Science (SEA-PHAGES) program, enables students to isolate and identify phages from soil samples that they have collected. Over 150 colleges and high schools participate in this program, with numerous students contributing to the database of collected and sequenced phages. More than 800 mycobacteriophages and other Actinobacteriophages have been sequenced by students and added to the database of known bacteriophages (“SEA-PHAGES,” 2018). The inclusion of undergraduate students from across the world not only increases the number of phages that can be isolated and the range of areas that can be sampled, but it also increases the retention rate of students in the STEM field (Jordan et al., 2014).

Virus Viability in Soil

There are many factors which affect virus viability and adsorption in soil, such as water quality, moisture content, ambient temperature, soil pH, soil composition, the presence of other microorganisms, and soil texture (Blanc & Nasser, 1996). Temperature has been shown to have a significant effect on the viability of both bacteria and some viruses. In natural water, viruses have been found to survive longer at lower temperatures, remaining infectious for months at temperatures close to freezing (Nasser et al., 1993). This trend was consistent in viruses found in soil, such as MS-2 and PRD-1 bacteriophages, which had negligible die-off at temperatures
around 10°C. However, at ambient temperatures of about 23°C, significantly higher die off rates were observed in MS-2 and PRD-1 bacteriophages (Blanc & Nasser, 1996).

The relationship between soil moisture and virus survival has been shown to follow a nonlinear trend. When studying the effect of soil moisture on the survivability of poliovirus, the virus survival decreased as soil moisture increased up until it reached a 15% wet weight. After the 15% wet weight point, the virus survival then began to increase as soil moisture increased. This trend suggests that the viability of a phage in soil may be dependent on the moisture level of the soil. Therefore, precipitation may have an effect on the bacteriophage population in soil (Hurst et al., 1980).

The humidity of the environment has a different effect depending on the phage studied. One experiment tested the sensitivity of several phages, chosen because of their similarities in terms of size, nucleic acid composition, and envelope, to various levels of humidity. Certain phages, such as MS-2, seem to be stable at all levels of relative humidity. However, other phages, such as PR77, were sensitive to relative humidity levels above 50%, with no infectious phage particles observed after exposure. Therefore, the effect of humidity seems to differ from phage to phage, regardless of some structural similarities (Verreault et al., 2015). There is no clear way to predict the response of a phage to a change in humidity.

**SEA-PHAGES App**

Participants in the SEA-PHAGES program have not consistently recorded the location where the soil was collected, the conditions, the date and time, or whether a phage was found in any given sample. To attempt to solve this problem, a mobile app known simply as SEA-PHAGES was developed by Dr. Steven Cresawn, Melissa Wenzel, Roxana Behrooz, Joseph
Colman, Kaleigh Jaeger, Kelly Degnon, and Brie Lewis using the React Native and Meteor software libraries. React Native, a framework for building native apps for use on mobile devices, uses the JavaScript programming language. The app communicates with a server program, also developed in the Cresawn lab, using the Meteor JavaScript library. The data from this app is stored using MongoDB database software running on the JMU Center for Genome and Metagenome Studies (CGEMS) servers.

The SEA-PHAGES app currently collects data about the date and time of sample collection and the latitude and longitude at the location where it was collected. Students can also input a sample name and any relevant information about their sample. These samples will then appear on the “Map” screen of the app in the form of a pin located where the sample was collected. Students can update their submissions to include any additional details that they may have discovered after the time of collection. If a student identifies a phage in their sample, they can return to the app and edit their sample to reflect this by clicking on a button that states “Phage found in sample.”

Although the app already collects a large amount of data about each sample collected, there was no consistent method for recording the weather conditions in which these samples were collected. The goal of this project was the addition of new lines of code that integrate data from OpenWeatherMap and automatically collect data about the temperature, weather conditions, and humidity of the location where a sample is collected. It is hypothesized that a relationship between weather and the phage population will be identified. These data may allow for conclusions to be drawn about the likelihood of collecting a phage during a specific weather condition due to bacterial host and phage tolerances of various weather parameters.
Methods

React.js, a JavaScript library which contains a collection of functions that can be easily integrated into new code, was utilized through the use of React Native, which converts JavaScript code into native mobile applications. These frameworks were used to develop the SEA-PHAGES app for mobile devices running the Android and iOS operating systems, while Meteor and MongoDB were used on the server end. Meteor is a platform and framework for developing web applications and it has MongoDB built in. MongoDB, a non-relational document database which stores data collections in JSON format, was used to store data collected using the app. The SEA-PHAGES app allows users to record the coordinates where a soil sample was collected, the date and time, and any other relevant information about this sample.

Using the WebStorm programming environment, the existing code of the SEA-PHAGES app was edited and additional lines of code were added so that the app would automatically collect data about the weather at the time and location where the sample was collected. Multiple pages were edited, as each page contains code for a separate aspect of the app. The edited pages included “SampleAddPage.js,” “SampleEditPage.js,” and “MapPage.js” pages. A new page titled “api.js” was also created to contain code that would send latitude and longitude data to OpenWeatherMap and return the weather conditions for that specific place and time.

“api.js” Page

OpenWeatherMap, a company that collects a variety of Earth Observation data, including data about past, current, and predicted weather conditions, provides application programming interfaces (API) and keys which allow for their data to be pulled and integrated into various programs. An API key from OpenWeatherMap was acquired and integrated into the SEA-
PHAGES app code on a new page titled “api.js.” The purpose of this code was to submit the latitude and longitude of each potential sample, as determined by the phone’s geolocation, to OpenWeatherMap and to return data about the weather conditions at that specific location. This function was defined as `fetchWeather` (Figure 2). The function operates by accessing a url that states which fields are being inputted, submits an API key, and returns this data as a response. This `fetchWeather` function was then called on the “SampleAddPage.js”

```javascript
export default function fetchWeather(lat, lon) {
    let url = `https://api.openweathermap.org/data/2.5/weather?lat=${lat}lon=${lon}&units=metric&APPID=APIKEY`;
    return fetch(url).then(response => response.json()).then(responseJson => {
        return responseJson;
    }).catch((error) => {
        console.log(`Error with fetchWeather function: ${error}`);
    });
}
```

**Figure 2:** The “api.js” page of code which submits the latitude and longitude data to OpenWeatherMap and returns weather data for that specific location.

“SampleAddPage.js”

The `fetchWeather` function was imported into the “SampleAddPage.js” from the “api.js” page using the “import” function on line 14. Next, new variables for weather conditions were made on lines 28-32 and lines 36-38. These variables were defined as “null” so that they could use the data from OpenWeatherMap, once integrated.

Code also needed to be written to prevent users from submitting a sample with an empty weather field. To do so, lines 77-89 were written, which contain a conditional statement. This statement prevents the sample from being added if the temperature, weather, or humidity fields are incomplete. In this case, an error will appear stating “Error determining the current weather conditions.” It will also prompt the statement “Please make sure the app has permission to access
your location and try again.” Then users must click “OK” to close these messages and return to the Add Sample Page, where they can edit their entry to comply with the requirements.

The variable “newSample” was also edited on lines 94-96 to include the temperature, weather, and humidity data for each sample (Figure 3). This makes it so that the temperature, weather, and humidity data are recorded as being a part of the sample and will be stored with the sample data.

```javascript
import fetchWeather from './components/api.js';

const backAction = NavigationActions.back()

class SampleAdd extends Component {
  constructor(props) {
    super(props);
    this.state = {
      key: null,
      region: {
        latitude: null,
        longitude: null,
      },
      condition: {
        temperature: null,
        weather: null,
        humidity: null,
      },
      title: '',
      lat: '',
      lng: '',
      temp: '',
      weather: '',
      humid: '',
      date: '',
      description: '',
      phoneNumber: null,
      owner: '',
    };
  }

  if (this.state.temperature === 0 || this.state.weather === 0 || this.state.humidity === 0) {
    return (
      alert('Error determining the current weather conditions. 
      Please make sure the app has permission to access your location and try again.'),
      {
        text: 'OK', onPress: () => console.log('OK Pressed'),
      },
      {Cancelable: false}
    );
  }

  const newSample = {
    title: this.state.title,
    lat: this.state.latitude,
    lng: this.state.longitude,
    temp: this.state.temperature,
    weather: this.state.weather,
    humid: this.state.humidity,
    date: new Date(),
    description: this.state.description,
    phoneNumber: this.state.phoneNumber,
    owner: this.props.user_id,
  };
}
```

**Figure 3:** The “SampleAddPage.js” which was edited to import the *fetchWeather* function, define the temperature, weather, and humidity functions, create a conditional statement requiring weather data so that the sample can be submitted, and add the weather data to the sample.

The SEA-PHAGES app uses geolocation to collect the latitude and longitude of the phone being used. The geolocation data collected by the app must then be used to collect weather data by passing it into the *fetchWeather* function, defined on the “api.js” page. To do so, lines 157-164 were written in the form of a promise. In coding, a promise is a way to guarantee that
the code cannot proceed until the first operation is completed. In this case, the promise requires that the latitude and longitude data are collected using geolocation before this information can be submitted to OpenWeatherMap for the weather data. Once the latitude and longitude data are acquired, the coordinates will pass through the `fetchWeather` function, which will submit these coordinates to OpenWeatherMap and return the weather conditions at that location. The response from OpenWeatherMap will then be used to set the temperature, weather, and humidity variables set previously in the code (Figure 4).

![ComponentDidMount code snippet]

**Figure 4:** Lines of code that use the `fetchWeather` function from the “api.js” page to define temperature, weather, and humidity variables with their respective data from OpenWeatherMap at a specific latitude and longitude.

The “Temperature (C),” “Weather,” and “Humidity (%))” fields were needed on the “Sample Add Page” of the app as a way to visualize what data was being collected for each sample. These fields were created as disabled labels on lines 205-216, meaning that they cannot be edited by app users on this page and they can only contain data collected automatically from OpenWeatherMap. Although corrections to these fields cannot be made on this page, users can
later edit their samples on the Edit Sample Page. To import this data from OpenWeatherMap, the
values defined previously were called as values for each field (Figure 5).

**Figure 5:** Lines 205-216 on the “SampleAddPage.js” page were written to create the fields for
the “Temperature (C),” “Weather,” and “Humidity (%)” and to automatically fill these fields
with the data collected from OpenWeatherMap.

“MapPage.js”

The “MapPage.js” contains code which places a pin on the map at the location where a
sample is collected. When hovering over a pin, a text box appears with the sample name, date
and time of sample collection, and sample details. This code was edited on lines 59-61 to add the
temperature, weather, and humidity data to this text box. The code was written so that it would
state the name of the field, for example “Temperature (C): ,” and would then state the variable
being called, for example, “sample.temp.” This code was written to return the weather data for that sample on the text box shown above each pin when hovering (Figure 6).

**Figure 6**: Lines 59-61 were added to the “MapPage.js” code so that, when hovering over each sample pin, the temperature, weather, and humidity data would be shown.

**“SampleEditPage.js”**

The “SampleEditPage.js” was edited to show the temperature, weather, and humidity fields. These fields show the data currently in the database for the sample and allow for the data to be edited and updated in the database. The code added to lines 30-32 and lines 48-50 are needed to call the temperature, weather, and humidity information for each sample (Figure 7).
In order to have the temperature, weather, and humidity data appear on the Sample Edit Page, these variables were called so that they would appear in their respective labels. The data in the database was called as a placeholder, which could be edited by the user. This feature is useful in the event that the data collected from OpenWeatherMap was not completely correct, such as if the weather condition suddenly changed. If the user did not change these values, the data for that field would not be changed. However, if the user did alter this data, it would be edited in the database to reflect this change (Figure 8).
Figure 8: Lines of code added to the “SampleEditPage.js” which create a temperature, weather, and humidity field that can display the data in the database and which allows for the data to be edited and updated in the database.

To prevent a sample from being submitted without anything in the field, a conditional statement was added to lines 84-95 that will present an error message if the sample is submitted with an empty field (Figure 9). Similar to the error message that will appear on the Sample Add Page if a field is incomplete, this error message will alert users that there was an error in determining the weather conditions and it requests that they confirm that their app has permission to access their location. Users must click “OK” to close these messages and return to the Edit Sample Page.

Figure 9: Code added to the “SampleEditPage.js” page which prevents a sample from being resubmitted without a field being completed.
Simulation of the Code

To visualize the changes to the code and how they alter the appearance and performance of the app, Apple’s Xcode simulator was used. Xcode runs the app on a computer, without the need to load it on an iOS device (Figure 10). This simulation was vital to see how the changes made to the code would affect the appearance and functionality of the app.

Figure 10: A screenshot of a simulation of the About page of the SEA-PHAGES app using the Xcode simulator.

Results

“api.js” Page

To confirm the functionality of the `fetchWeather` function written on the “api.js” page, the “console.log(fetchWeather())” function was used. This function allowed for the results of the
API call to be viewed in a web browser’s JavaScript the console (Figure 11). The console shows the temperature, weather description, humidity, and additional data retrieved by the SEA-PHAGES app when communicating with OpenWeatherMap.

```
Promise {40: 0, 65: 0, 55: null, 72: null}
  _40: 0
  _55: base: "stations"
  _65: clouds: {all: 50}
  _72: cod: 200
  coord: {lon: 0, lat: 0}
  dt: 1522786168
  id: 0
  _main:
    _40: grnd_level: 1021.99
    _55: humidity: 80
    _65: pressure: 1021.99
    _72: sea_level: 1022.06
    _main: temp: 27.83
    _proto___: Object
    _proto__: name: ""n
  _sys: {message: 0.1711, sunrise: 1522735184, sunset: 1522778777}
  _weather: Array(1)
    _0: description: "broken clouds"
    _1: icon: "04n"
    _2: id: 803
    _3: main: "Clouds"
    _proto___: Object
    _proto__: length: 1
    _proto__: Array(0)
    _proto__: wind: {speed: 4.56, deg: 168.501}
    _proto__: Object
    _proto___: null
    _proto: null
    _proto__: Object
```

**Figure 11:** The data returned by the `fetchWeather` function, visualized using the `console.log` function.

“SampleAddPage.js”

The code added to the “SampleAddPage.js” resulted in the creation of three new fields: Temperature, Weather, and Humidity. These fields are automatically filled with data from OpenWeatherMap due to the integration of the API key into the code.
Figure 12: Screenshots of the Xcode simulation of the “Add Sample” page showing the Sample Name, Date and Time, Latitude, Longitude, Temperature, Weather, Humidity, and Details fields. The image on the left shows the page without scrolling, while the image on the right shows the fields that become visible after scrolling.

The date and time, latitude, longitude, temperature, weather, and humidity data should automatically fill these fields if geolocation is enabled. The Sample Name and Details fields must be filled in by the user and, if any of these fields are empty when the user attempts to submit the sample to the database, an error message will be presented. If all fields are completed on the “Add Sample” page and the “Submit Sample” button is pressed, the sample is stored in the MongoDB database. Each data sample stored in the database contains an entry for each field.
“MapPage.js”

When viewing the map page of the app, each sample is shown as a pin at the coordinate location where the sample was collected. The samples collected by the logged-in user are represented as a yellow pin, while samples collected by other app users would be represented as purple pins. When clicking on any pin, the sample name, date and time of collection, temperature, weather condition, humidity at the time of collection, and the description submitted by the person who collected the sample are displayed in a text box (Figure 13).

Figure 13: Screenshots of the Xcode simulation of the “Map” page, while hovering over a pin on the map. Each pin represents the collection location of the sample and provides information about the conditions in which it was collected.
The “SampleEditPage.js” displays the data currently in the database as fields that can be edited. Each field can be clicked on and typed in. Whatever the user types in will be submitted to the database to update the current sample. This page also contains the ability to mark whether the sample contained a phage, which is vital in collecting data about the distribution of phages and the weather conditions at the time of phage collection (Figure 14).

**Figure 14:** Screenshots of the Xcode simulation of the edit sample page, which allows for the editing of the fields of data collected during the time when the sample was submitted.
Discussion

The data that will be collected using this app will provide insight into any potential relationship between weather and the abundance and type of bacteriophages present in an environment. Such data may suggest whether there is an ideal weather condition for collecting phages from soil. Although there is much that this app can perform on its own, data collection is dependent on user input and relies heavily on students returning to the app to update their soil samples or to correct errors.

Based on previous experimentation, it is predicted that data collected from the SEA-PHAGES app will reflect that temperature has the largest effect on virus survival in soil, as compared to other weather conditions such as humidity. It is predicted that phages may be more abundant in colder temperatures, as they can remain infectious at temperatures close to freezing. Therefore, phages may be more likely to be isolated in colder temperatures. It is also predicted that the ideal temperature will differ between phages, especially between those who target a different host or those who are not closely related. There may be a correlation between the ideal temperature of the host bacteria and the ideal temperature of the bacteriophages that target it, as they are constantly engaging in a predator-prey relationship. Due to virus survival trends in various soil moistures, it is also predicted that phage survival will follow a nonlinear trend dependent on the wet weight of the soil, which is affected by precipitation. In terms of humidity, it is predicted that phages will respond to humidity differently unless the phages are extremely closely related.

If it is found that there is a temperature or weather condition in which phages are more likely to be isolated, these data may be used to increase the likelihood that a student in the SEA-
PHAGES program will isolate a phage. If more students are able to isolate phages, this will not only expand the database of known phages, but it will also increase the likelihood that students will be invested in their experiment and retain the information that they are learning. According to the Survey of Undergraduate Research Experiences, students who engage in undergraduate research are more likely to learn skills related to lab techniques, research processes, and scientific problems than students who do not engage in research. Students also cited personal development and stated that they planned to continue with an education and career in the sciences, which may help to fill the gap between the growth of the STEM field and the insufficient amount of scientists available to fill these positions (Lopatto, 2004).

The amount of data collected using this tool will be dependent on the participation of students in the SEA-PHAGES program. Although this tool will collect a large amount of data automatically, it cannot predict or confirm whether a phage will be found in a particular soil sample. Therefore, data about the isolation of phages cannot be collected unless students return to the SEA-PHAGES app and update their sample database entry to state whether a phage was found in their sample. A feature can be added so that, if notifications from the app are permitted on the user’s phone, reminder notifications can be sent following sample collection to prompt users to edit their samples. Additionally, there will likely be many phages in each sample of soil collected that will not be identified because the students only enrich for phages that target a specific type of bacteria. Therefore, even an area that seems to be heavily studied likely contains a multitude of phages that have yet to be identified. Although this information may become more complete as more data is collected, it cannot be considered a final determinant of what phages will be found where and in what conditions. The data can, however, be used to observe trends
and there is no limit to the number of phages and the amount of data that can be collected using this tool.

Another potential problem that may arise is the lack of data collected during poor weather conditions. For example, it is unlikely that many people would collect their soil sample while it is raining or during other harsh conditions. As a result, it will be difficult to make any conclusions about the likelihood of finding a phage during harsh weather conditions, as there will be insufficient data.

Because this app relies on geolocation to recorded latitude and longitude data as well as weather data, it cannot function without cellular service or Wi-Fi. This may prevent data collection about samples that are found in areas without cellular service, such as in the woods or in underdeveloped areas. Although users could record this data elsewhere and enter it into the app manually at a later time, there is a concern that users will not take the time to do so. This could limit the amount of data that is collected about these areas as people are less likely to collect soil samples from these remote locations and, if they do, they likely will not record this data in the app.

In the future, additional features can be added to the app to collect further data about the bacteriophages identified through the SEA-PHAGES program. A feature that would be useful is one which synchronizes soil samples on the app to the sequence and cluster data found on phamerator.org. This could make it easier to visualize the distribution of phage clusters and could provide information about whether closely related phages are more likely to be found in close proximity to one another. It may also provide an explanation of how the mosaic genome of phages works. For instance, if there are two phages found in close proximity to one another that
share a specific fragment of a gene but are not closely related, it could be hypothesized that these phages have somehow shared genes between one another. As additional features are added to the SEA-PHAGES app and as data is collected, the relationship between weather and the bacteriophage community in soil can be studied and potential trends could be identified.

Acknowledgements

Special thanks to the Howard Hughes Medical Institute, SEA-PHAGES Program, Dr. Steven Cresawn, Dr. Elizabeth Doyle, Dr. Pradeep Vasudevan, and the Cresawn Lab.

Additional thanks to the other members of the SEA-PHAGES app team: Melissa Wenzel, Roxana Behrooz, Joseph Colman, Kaleigh Jaeger, and Brie Lewis.

References


