May 2020

Fitting of Lotka-Volterra Model for Coupled Population Growth Data Through Least-Squares Estimation of Parameters

Jessica Ann Harter
University of Wisconsin-Milwaukee

FITTING OF LOTKA-VOLTERRA MODEL FOR COUPLED POPULATION GROWTH DATA THROUGH LEAST-SQUARES ESTIMATION OF PARAMETERS

by

Jessica Harter

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Mathematics

at

The University of Wisconsin-Milwaukee

May 2020

ABSTRACT

FITTING OF COUPLED POPULATION DATA THROUGH ESTIMATION OF PARAMETERS USING THE LEAST SQUARES METHOD

by

Jessica Harter

The University of Wisconsin-Milwaukee, 2020
Under the Supervision of Professor Daniel Gervini

The population of two types of bacteria found in the Gulf Coast of Florida, V.chagasii and V. harveyi, can be described by the Lotka-Voltera competition model. Using data gathered in experiments conducted by Bury and Pickett (2015), we take a different approach to find parameter estimates using numerical methods in R. In particular, we find a numerical solution to the coupled set of ODEs and minimize the sum of squared errors in order to obtain the optimal parameter estimates that will fit the data best. In order to get a sense of accuracy of these parameter estimates, we use bootstrap estimation to compute the component wise standard deviations and construct confidence intervals for the estimates.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

The paper by Bury and Pickett (2015) studies the populations of two bacteria, *V. harveyi* and *V.chagasii*, at various temperatures. Their paper aimed to determine the interaction between these two populations by conducting an experiment in which they observed the population of each bacteria every hour, for six hours total. They conducted the same experiment at the temperatures 10°C, 15°C, 20°C, 25°C, 30°C, and 36°C, but disregarded data collected at 30°C and 36°C due to either extreme outliers, in which case parameter estimators could not be computed using their methods, or due to flawed execution of the experiment. In this paper we discuss the model used to describe the populations of these bacteria, how to estimate the parameters of the model and compare our parameter estimates to those found by Bury and Pickett (2015).

2 Lotka-Volterra Competition Model

For a given temperature, we denote the populations of *V. harveyi* and *V.chagasii* as $H(t)$ and $C(t)$, respectively, at a given time t. The interaction between these two populations is best modeled using the Lotka-Volterra Competition model, as follows

$$\frac{dC}{dt} = r_c C\left(1 - \left(\frac{\alpha_{cc}}{K_c}C + \frac{\alpha_{ch}}{K_c}H\right)\right)$$
$$\frac{dH}{dt} = r_h C\left(1 - \left(\frac{\alpha_{hh}}{K_h}H + \frac{\alpha_{hc}}{K_h}C\right)\right)$$

where the practical interpretation of each parameter is given in Table 2.1.

Parameter	Description
r_c	Growth rate of *V.chagasii*
r_h	Growth rate of *V.harveyi*
α_{ch}	Impact of *V.harveyi* cells on *V.chagasii* cells
α_{hc}	Impact of *V.chagasii* cells on *V.harveyi* cells
α_{cc}	Competition between *V.chagasii* cells with other *V.chagasii* cells
α_{hh}	Competition between *V.harveyi* cells with other *V.harveyi* cells
K_c	Carrying capacity of *V.chagasii* population
K_h	Carrying capacity of *V.harveyi* population

Table 2.1: A description of each parameter in the Lotka-Volterra Competition Model. The competitive ability coefficients, α_{ch}, α_{hc},α_{cc}, and α_{hh}, describe the interaction between the bacteria.

We rewrite the model by adopting the notation in chapter 13 of Jones, Maillardet, and Robinson (2014). We define $\mathbf{y}(t) = (C(t), H(t))$, which satisfies the vector differential equation

$$\frac{d\mathbf{y}(t)}{dt} = \mathbf{f}(t, \mathbf{y})$$

2.1 Redefining Parameters, and Least Squares Method

Bury and Pickett (2015) determined that due to the environment in which this bacteria is found, it is highly unlikely for the bacteria to encounter limiting nutrients and thus, it is meaningless to assign a value to the carrying capacities K_c and K_h. This leads us to redefining the parameters of the Lotka-Volterra model by letting

$$\delta_{cc} := \frac{\alpha_{cc}}{K_c} \quad \text{and} \quad \delta_{ch} := \frac{\alpha_{ch}}{K_c}$$
$$\delta_{hh} := \frac{\alpha_{hh}}{K_c} \quad \text{and} \quad \delta_{hc} := \frac{\alpha_{hc}}{K_c}$$

so that our model $\mathbf{f}(t, \mathbf{y})$ reduces to

$$\frac{dC}{dt} = r_c C\big(1 - (\delta_{cc} C + \delta_{ch} H)\big)$$
$$\frac{dH}{dt} = r_h C\big(1 - (\delta_{hh} K_h H + \delta_{hc} C)\big)$$

This simplified model depends on six unknown parameters, $\boldsymbol{\theta} = (r_c, \delta_{cc}, \delta_{ch}, r_h, \delta_{hh}, \delta_{hc})$, which we estimate from the data collected by Bury and Pickett (2015). Note that if we didn't redefine the parameters, as shown above, $\boldsymbol{\theta}$ would have eight components as opposed to six. Due to the small size of the data set, having to estimate more parameters than necessary could lead to issues, such as overfitting. This gives us more motivation, beyond the biological reasoning provided by Bury and Pickett (2015), to redefine the parameters as we did above. It is important to mention that we aim to estimate all six parameters simultaneously, whereas Bury and Pickett (2015) estimated the parameters in two phases. In their first phase, they estimated the growth rates using data from a separate experiment. In their second phase, they used the data that we also used in this paper to obtain estimates for the δ parameters. Estimation of these six parameters provides insight into how these two populations of bacteria interact.

For different values of $\boldsymbol{\theta}$, we obtain different functions $\mathbf{y}(t)$, so to be more specific in the notation, we write the system of differential equations in vector form as

$$\frac{d\mathbf{y}_\theta(t)}{dt} = \mathbf{f}(t, \mathbf{y}_\theta, \boldsymbol{\theta})$$

where $\mathbf{y}_\theta(t)$ is the solution to the differential equation for a given $\boldsymbol{\theta}$. Because the Lotka-Voltera model has no closed form solution, it is necessary to numerically solve for $\mathbf{y}_\theta(t)$ for a given $\boldsymbol{\theta}$. This can be done using the ODE solver in the deSolve package in R.

Using the data points $(t_0, \mathbf{y}_0) \dots (t_6, \mathbf{y}_6)$ from Bury and Pickett, we can find an optimal $\boldsymbol{\theta}$ by minimizing the sum of squared errors,

$$L(\boldsymbol{\theta}) = \sum_{i=0}^{6} \|\mathbf{y}_i - \mathbf{y}_\theta(t_i)\|^2,$$

where $\| \cdot \|$ is the two dimensional Euclidean norm. We call the minimizer of this function $\hat{\boldsymbol{\theta}}$. Since there is no closed form solution for $\hat{\boldsymbol{\theta}} = (\hat{r}_c, \hat{\delta}_{cc}, \hat{\delta}_{ch}, \hat{r}_h, \hat{\delta}_{hh}, \hat{\delta}_{hc})$, we must find the solution numerically using the `optim` function in R.

3 Implementation

To find the minimizer, $\hat{\boldsymbol{\theta}}$, we need to implement four functions in R. See Appendix A for the R code.

We first need to initialize a few values. The initial populations for a given temperature need to be pulled from the data set. There also needs to be an initial guess of the parameter $\boldsymbol{\theta}$, as explained below. Finally, we need to initialize a time grid which the ode solver uses. This time grid needs to go from 0 to 6, incremented by 0.1 or smaller.

The first function, which is called `f.model`, that needs to be implemented is $\mathbf{f}(t, \mathbf{y}_{\theta}, \boldsymbol{\theta})$, which returns both components of the differential equation.

The second function, which is called `sde`, solves the differential equation using the ode solver in R, using the Classical Runge-Kutta 4th Order Integration method. It takes $\boldsymbol{\theta}$ and the first function as inputs. This function returns the solution to the system, $\mathbf{y}_{\theta}(t)$, which is an array with the number of rows being equal to the grid size, and three columns - the time, $C(t)$, and $H(t)$.

The third function, which is called `LS`, computes the sum of squared errors. This function calls the second function to compute $\mathbf{y}_{\theta}(t)$. However, the data points t_i might not be in the time grid that was initialized, leading us to have more $\mathbf{y}_{\theta}(t)$ values than data points $\boldsymbol{y}(t_i)$. Thus, we only keep the values $\mathbf{y}_{\theta}(t_i)$ for $i = 0, 1, \ldots, 6$.

The fourth function, which is called `minimizer`, finds the minimizer $\hat{\boldsymbol{\theta}}$ by calling the R function optim, which uses the Nelder-Mead method to do the minimization. One input of this function is an initial parameter guess. A "good" initial guess was found looking at the plots of several $\mathbf{y}_{\theta}(t)$, as well as using the previous results from Bury and Pickett. Depending on the initial guess of the parameter, we get a different value out of the least squares function. If the initial parameter guess is too far off, a minimum can not be found. By trying various reasonable initial parameters, one can run the R code until the model visually looks like a good fit to the data, or by comparing the different least square error values.

3.1 Normalizing the Data

An important note to make about the data set is the magnitude of each population. In an effort to be able to conceptualize the size of each population at given time steps, we normalize the populations by either 10^{-7} or 10^{-8}, depending on the temperature. If we manipulate the data in such a way, we need to ensure that the value for each parameter we find in R is the actual value of the parameter and if it is not, we need to account for this and use the parameters from R to find the actual parameters.

Let β be the constant we are normalizing the data by, so that the populations we use in R are now $\tilde{C} = \beta C$ and $\tilde{H} = \beta H$. Then for the first component of $\mathbf{f}(t, \mathbf{y}_\theta, \boldsymbol{\theta})$, we have

$$\begin{aligned} \frac{d}{dt}C &= r_c C\big(1 - (\delta_{cc} C + \delta_{ch} H)\big) \\ \frac{d}{dt}C\beta &= r_c C\beta\big(1 - (\delta_{cc} C + \delta_{ch} H)\big) \\ \frac{d}{dt}\tilde{C} &= r_c \tilde{C}\big(1 - (\delta_{cc} C + \delta_{ch} H)\big) \end{aligned}$$

Letting $\tilde{\delta}_{cc} = \frac{\delta_{cc}}{\beta}$ and $\tilde{\delta}_{ch} = \frac{\delta_{ch}}{\beta}$, we get

$$\frac{d}{dt}\tilde{C} = r_c \tilde{C}\big(1 - (\tilde{\delta}_{cc} \tilde{C} + \tilde{\delta}_{ch} \tilde{H})\big)$$

Thus, the parameters that are computed numerically, $\tilde{\delta}_{cc}$ and $\tilde{\delta}_{ch}$, are the true value of each parameter, scaled by $1/\beta$. This means that the true values of the parameters are

$$\delta_{cc} = \beta\tilde{\delta}_{cc}$$

$$\delta_{ch} = \beta\tilde{\delta}_{ch}$$

Similarly, for the second component of $\mathbf{f}(t, \mathbf{y}_\theta, \boldsymbol{\theta})$, we have

$$\frac{d}{dt}\tilde{H} = r_h \tilde{H}\big(1 - (\tilde{\delta}_{hh}\tilde{H} + \tilde{\delta}_{hc}\tilde{C})\big)$$

where $\tilde{\delta}_{hh} = \frac{\delta_{hh}}{\beta}$ and $\tilde{\delta}_{hc} = \frac{\delta_{hc}}{\beta}$. Thus, the true values of the parameters are

$$\delta_{hh} = \beta\tilde{\delta}_{hh}$$
$$\delta_{hc} = \beta\tilde{\delta}_{hc}$$

Note that the parameters r_c and r_h remain unchanged, so the values for these two parameters obtained in R are the true values.

3.2 Bootstrap Estimation of Standard Deviations

Upon obtaining point estimators $\hat{\boldsymbol{\theta}}$ of the parameters for a given temperature, it is important to get a sense of accuracy of these parameters. The easiest way to do this is by parametric bootstrap, which allows us to obtain estimates of the standard deviations of $\hat{\boldsymbol{\theta}}$ and construct confidence intervals for $\boldsymbol{\theta}$.

In general, bootstrapping is used to draw inferences about populations. The bootstrapping procedure treats the single sample originally obtained as only one of the many random samples that the study could have obtained. The process takes the sample obtained, then re-samples it multiple times to create many simulated samples. It ends with our simulated data sets having many different combinations of the values that exist in the single, original data set and uses the distribution of the sample statistics across the simulated samples as the sampling distribution. This method is outlined in chapter 9 of [3].

The idea of bootstrapping for our purpose is to re-sample a single data set to create many simulated data sets. For each simulated data set, we find the optimal parameter $\hat{\boldsymbol{\theta}}$ from which we can draw inferences about the parameter, such as the standard deviation.

Suppose the data follows the model

$$\mathbf{y}_i = \mathbf{y}_\theta(t_i) + \varepsilon_i, \qquad i = 1, \dots, n$$

where $\boldsymbol{\varepsilon}_i = (\varepsilon_{ic}, \varepsilon_{ih})$ is random error of the V.Chagassi and V.Harveyi, respectively. We can assume $\varepsilon_{ic} \sim N(0, \sigma_c^2)$ and $\varepsilon_{ih} \sim N(0, \sigma_h^2)$ are independent. We estimate the variances σ_c^2 and σ_h^2 from the residuals of the least-squares fits already obtained. Let

$$\hat{\boldsymbol{\varepsilon}}_i = \mathbf{y}_i - \mathbf{y}_{\hat{\theta}}(t_i), \qquad \text{for } i = 1, \dots, n$$

Then $\hat{\sigma}_c^2$ is the sample variance of the $\hat{\varepsilon}_{ic}$s and $\hat{\sigma}_h^2$ is the sample variance of the $\hat{\varepsilon}_{ih}$s.

Now we can generate bootstrap samples of the data and of $\hat{\boldsymbol{\theta}}$ in the following way:

1. Calculate the error terms $\hat{\varepsilon}_{ic}$ and $\hat{\varepsilon}_{ih}$ and their corresponding standard deviations.
2. Create bootstrap pseudo-data as $\mathbf{y}_i^* = \mathbf{y}_\theta(t_i) + \boldsymbol{\varepsilon}_i^*$ and compute the respective parameter estimator $\hat{\boldsymbol{\theta}}^*$. This is done by creating a function, which is called `boot`, that takes the parameter and the number of pseudo-data points to generate, $\boldsymbol{B}$, as inputs. This function repeats the following $\boldsymbol{B}$ times (note, we take $\boldsymbol{B} = 300$):
   (a) Generate n random error terms ε_{ic}^* and ε_{ih}^* that are normally distributed with mean zero and using their corresponding standard deviations computed in step 1.

   (b) Compute $\mathbf{y}_i^*$ for each population using the respective error terms found in step (a). Column combine these two vectors, creating a 7x2 matrix.

   (c) Use the `minimizer` function to find $\hat{\boldsymbol{\theta}}^*$, using the bootstrap pseudo-data generated in part (b) and $\hat{\boldsymbol{\theta}}$ as the initial parameter guess.

   (d) Continue this process, adding each new $\hat{\boldsymbol{\theta}}^*$ to a matrix every iteration.

Because the data set we use when performing computations in R is scaled by β, it is necessary to scale the components of $\hat{\boldsymbol{\theta}}^*$ that correspond to a competitive ability by β and to not scale the components of $\hat{\boldsymbol{\theta}}^*$ that correspond to a growth rate.

From the bootstrap sample $\hat{\boldsymbol{\theta}}_1^*, \ldots, \hat{\boldsymbol{\theta}}_B^*$ that has been scaled properly, we compute the component-wise sample standard deviations. These are the j^{th} component of estimated standard deviations of $\hat{\boldsymbol{\theta}} = (\hat{r}_c, \hat{\delta}_{cc}, \hat{\delta}_{ch}, \hat{r}_h, \hat{\delta}_{hh}, \hat{\delta}_{hc})$, denoted by $\sigma_{j\hat{\boldsymbol{\theta}}}$. Thus, for each component j of $\hat{\boldsymbol{\theta}}$, a 95% confidence interval is given by

$$(\hat{\theta}_j - z_{0.025}\sigma_{j\hat{\boldsymbol{\theta}}}, \hat{\theta}_j + z_{0.025}\sigma_{j\hat{\boldsymbol{\theta}}})$$

4 Results

For different initial parameter guesses, the minimizer $\hat{\theta}$ and its corresponding least squares value are different. For each temperature, after finding a "good" initial guess, that is, one that could find a minimum, we used different initial parameter guesses close to this initial guess until we got a small error (compared to other guesses). See Appendix B for the initial parameters used at each temperature.

In both plots of Figure 4.1, notice the outlier at $t = 5$ hours. The least squares is minimized when the fitted values are that of Figure 4.1, however, the shape of the curve does not describe typical population growth very well.

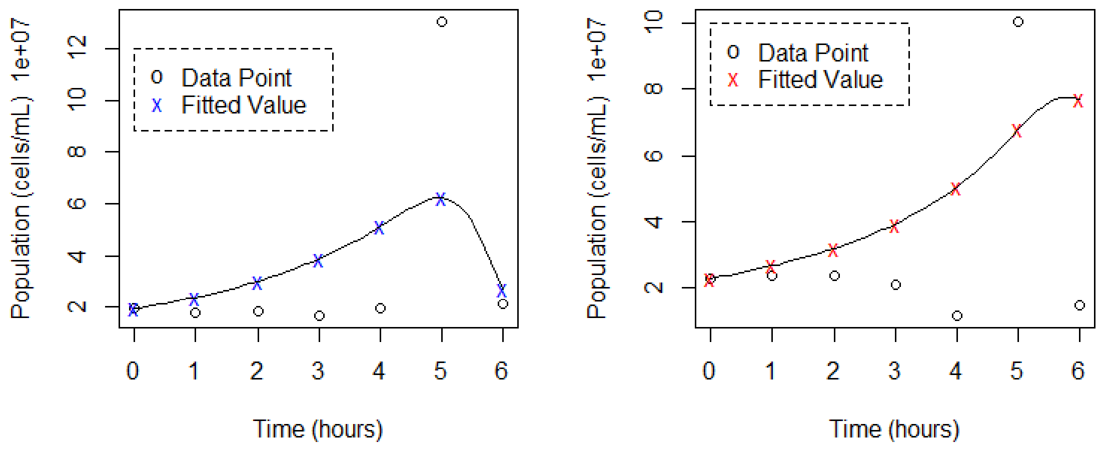

Figure 4.1: Plot of V.chagassi (left) and V.harveyi (right) at 10° C

The plots in Figures 4.2 through 4.4 do not have extreme outliers like Figure 4.1 did, so the shape of these curves when the least squares is minimum is what we would expect for population growth.

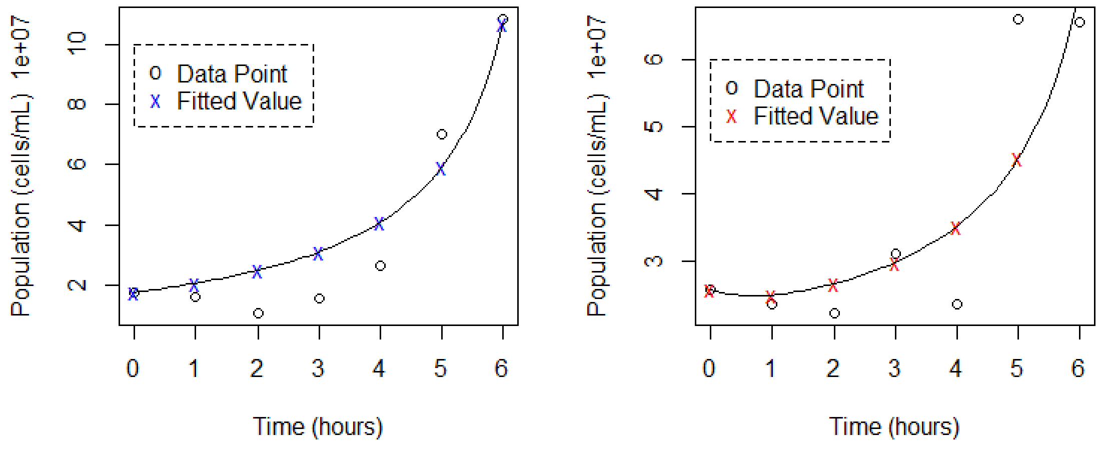

Figure 4.2: Plot of V.chagassi (left) and V.harveyi (right) at 15° C

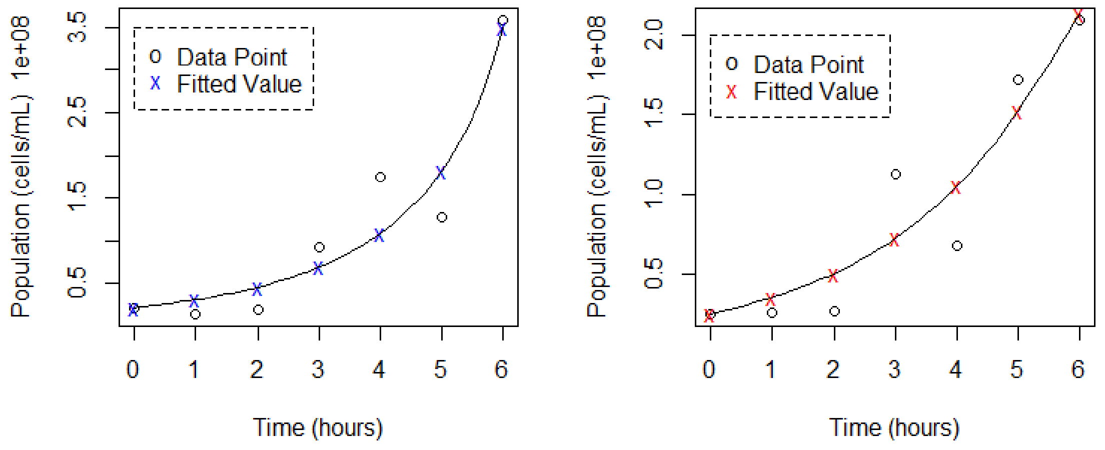

Figure 4.3: Plot of V.chagassi (left) and V.harveyi (right) at 20° C

While Bury and Pickett (2015) were unable to estimate all the parameters at 30° C due to the outliers in the data, we were able to fit the data, as shown in Figure 4.5, and get estimates for all the parameters, as shown in Table 4.4. In the V.harveyi plot of Figure 4.5, there is concern of overfitting the data to the model, however, this was difficult to avoid given the small sample size. At 36° C, the experiment was not executed correctly, so we do not have proper data at this temperature.

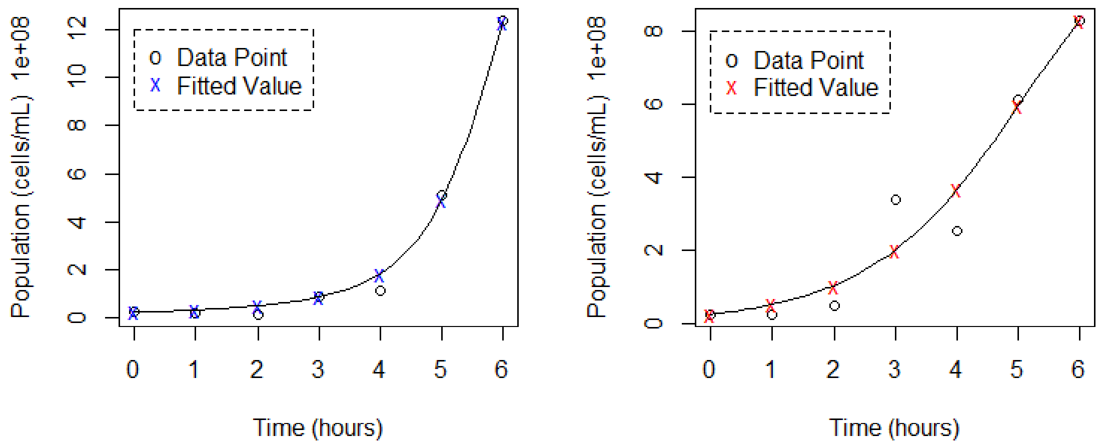

Figure 4.4: Plot of V.chagassi (left) and V.harveyi (right) at 25° C

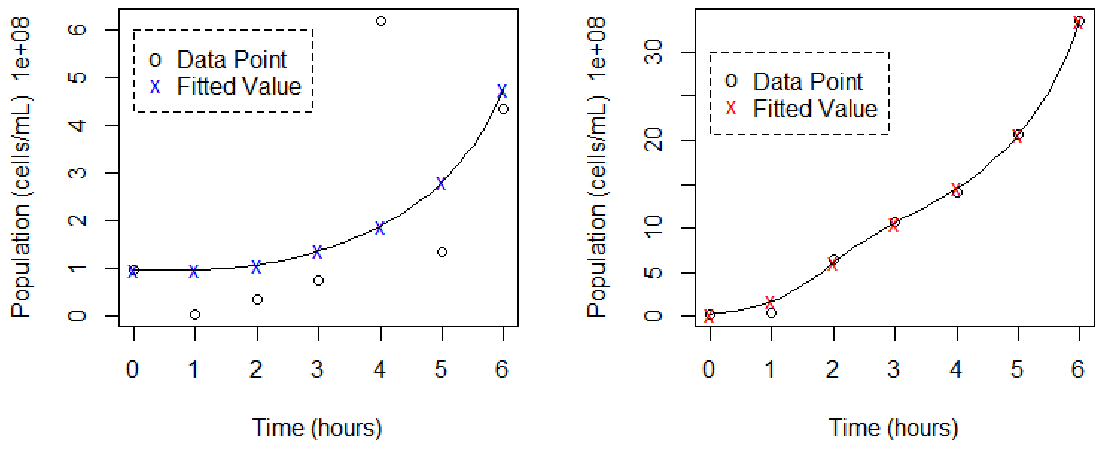

Figure 4.5: Plot of V.chagassi (left) and V.harveyi (right) at 30° C

	$10°C$	$15°C$	$20°C$	$25°C$	$30°C$
$\hat{r}_c$	0.4036	0.0544	0.3450	0.2830	0.1032
$\hat{\delta}_{cc}$	-1.0095×10^{-7}	-1.5579×10^{-7}	-3.4267×10^{-9}	4.1309×10^{-9}	1.3354×10^{-8}
$\hat{\delta}_{ch}$	1.0815×10^{-7}	4.3322×10^{-8}	-1.5157×10^{-10}	-7.8962×10^{-9}	-3.4459×10^{-9}
$\hat{r}_h$	0.0969	0.6687	0.3303	0.7491	0.5669
$\hat{\delta}_{hh}$	5.9978×10^{-8}	8.3226×10^{-8}	-5.0235×10^{-9}	6.8160×10^{-10}	3.6458×10^{-9}
$\hat{\delta}_{hc}$	-9.6182×10^{-8}	-5.4683×10^{-8}	3.5836×10^{-9}	8.0646×10^{-11}	-2.5652×10^{-8}

Table 4.1: The minimizer $\hat{\theta}$ for each temperature. These were found using the R code in the Appendix.

The estimated parameters that we found for each of these temperatures are shown in Table 4.4. An important note to make is that the values in Table 4.4 are rounded to the fourth decimal place. If one were to solve the ODE with these rounded parameter values, the solution would vary slightly from that obtained with the non-rounded parameters. The previous results obtained by Bury and Pickett are shown in Table 4.2. Because these parameters are rounded to the first or second decimal place, we were unable to reproduce anything close to the solution they obtained.

	$10°C$	$15°C$	$20°C$	$25°C$	$30°C$
$\hat{r}_c$	0.04	0.3	0.25	0.47	0.5
$\hat{\delta}_{cc}$	2.3×10^{-7}	4.1×10^{-10}	5.1×10^{-15}	4.8×10^{-9}	*
$\hat{\delta}_{ch}$	3.5×10^{-7}	-8.0×10^{-8}	-1.8×10^{-9}	-2.5×10^{-8}	*
$\hat{r}_h$	-0.02	0.05	0.2	0.34	0.37
$\hat{\delta}_{hh}$	-2.2×10^{-8}	1.8×10^{-8}	4.7×10^{-10}	2.3×10^{-9}	*
$\hat{\delta}_{hc}$	5.2×10^{-13}	2.1×10^{-8}	1.0×10^{-16}	1.9×10^{-17}	*

Table 4.2: The parameter estimates found by Bury and Pickett.

When comparing the parameter estimates we found to the ones previously obtained by Bury and Pickett (2015), notice that the sign of the parameter estimates sometimes differ. The sign of each parameter indicates the type of relationship between the bacteria, and thus, our differing parameter estimates have us drawing different conclusions regarding the interaction between the bacteria.

It is important to discuss why our parameter estimates differ from those obtained by Bury and Pickett (2015). As mentioned before, they found their parameter estimates in two phases. Once they obtained growth rate estimates in the first phase, these estimates remained fixed when they moved on to obtain estimates of the competitive ability coefficients in the second phase. A consequence of this is that it is more difficult to obtain a closer fit to the data since only four parameters were allowed to move freely. Conversely, we allowed all six parameters to vary, which is why we were able to obtain a closer fit to the data.

Using section 3.2, we compute the standard deviation and a 95% confidence interval of each parameter estimate, at each temperature. These results are listed in Table 4.3 and help us determine which parameters can be considered different from zero.

From Table 4.3, we conclude that the parameters r_c and r_h are significantly different, unlike the four δ parameters, which are rarely different from zero.

	Std Dev	Confidence Interval	Std Dev	Confidence Interval
For 10° C:			For 15° C:	
$\hat{r}_c$	0.3381	(-0.2591, 1.0664)	0.0212	(0.0127, 0.0961)
$\hat{\delta}_{cc}$	1.9630×10^{-7}	$(-4.8570, 2.8380) \times 10^{-7}$	8.6712×10^{-8}	$(-3.2574, 0.1416) \times 10^{-7}$
$\hat{\delta}_{ch}$	3.2127×10^{-7}	$(-5.2154, 7.3784) \times 10^{-7}$	1.2498×10^{-7}	$(-2.0164, 2.8829) \times 10^{-7}$
$\hat{r}_h$	3.7247×10^{-8}	$(-6.3318, 8.2690) \times 10^{-8}$	1.5538×10^{-7}	$(-2.3768, 3.7141) \times 10^{-7}$
$\hat{\delta}_{hh}$	1.3110×10^{-7}	$(-1.9697, 3.1693) \times 10^{-7}$	6.7863×10^{-7}	$(-1.2468, 1.4133) \times 10^{-6}$
$\hat{\delta}_{hc}$	2.5663×10^{-7}	$(-5.9918, 4.0681) \times 10^{-7}$	4.5283×10^{-7}	$(-9.4222, 8.3285) \times 10^{-7}$
For 20° C:			For 25° C:	
$\hat{r}_c$	0.1224	(0.1049, 0.5850)	0.0619	(0.1616, 0.4043)
$\hat{\delta}_{cc}$	1.2307×10^{-8}	$(-2.7549, 2.0696) \times 10^{-8}$	5.4482×10^{-9}	$(-6.5474, 14.8092) \times 10^{-9}$
$\hat{\delta}_{ch}$	1.2468×10^{-8}	$(-2.4589, 2.4286) \times 10^{-8}$	7.9110×10^{-9}	$(-2.3401, 0.7609) \times 10^{-8}$
$\hat{r}_h$	1.1067×10^{-9}	$(1.1333, 5.4717) \times 10^{-9}$	1.3051×10^{-9}	$(4.9332, 10.0492) \times 10^{-9}$
$\hat{\delta}_{hh}$	1.2842×10^{-8}	$(-3.0193, 2.0146) \times 10^{-8}$	1.4460×10^{-9}	$(-2.1525, 3.5157) \times 10^{-9}$
$\hat{\delta}_{hc}$	1.1342×10^{-8}	$(-1.8647, 2.5814) \times 10^{-8}$	1.3262×10^{-9}	$(-2.5188, 2.6801) \times 10^{-9}$
For 30° C:				
$\hat{r}_c$	0.0586	(-0.0117, 0.2182)		
$\hat{\delta}_{cc}$	2.2043×10^{-8}	$(-2.9849, 5.6558) \times 10^{-8}$		
$\hat{\delta}_{ch}$	3.0394×10^{-9}	$(-9.4031, 2.5114) \times 10^{-9}$		
$\hat{r}_h$	1.7050×10^{-9}	$(2.3277, 9.0112) \times 10^{-9}$		
$\hat{\delta}_{hh}$	1.2143×10^{-9}	$(1.2658, 6.0258) \times 10^{-9}$		
$\hat{\delta}_{hc}$	1.278×10^{-8}	$(-5.0715, -0.0589) \times 10^{-8}$		

Table 4.3: For each temperature and component of the estimated parameter, $\hat{\theta}_j$, a standard deviation and confidence interval is computed. All values are rounded to the fourth decimal.

References

[1] Bury, A. & Pickett, M. (2015) Seasonality in Vibrio Bacteria Population Structure: A practical application of the Lotka-Volterra competition model. *Proceedings of The National Conference On Undergraduate Research*. Eastern Washington University. pp.1006-1012.

[2] Jones, O., Maillardet, R. and Robinson, A. (2014). Introduction to Scientific Programming and Simulation Using R, 2nd edition, CRC Press/Taylor & Francis, FL.

[3] Givens, G.H. and Hoeting, J.A. (2013). Computational Statistics, 2nd edition, Wiley, NJ.

Appendices

Appendix A: R Code

```
# Read in data

vcdata <- as.matrix(read.table("vcdata.txt"))

vhdata <- as.matrix(read.table("vhdata.txt"))


vc.x <- as.matrix(as.integer(vcdata[2:8,1]))

vh.x <- as.matrix(as.integer(vhdata[2:8,1]))


beta <- 10^-7 # value which data is scaled by

vc.y <- matrix(as.double(c(vcdata[2:8,2:7]))*beta,nrow=7,ncol=6)

    # entries are scaled by beta

vh.y <- matrix(as.double(c(vhdata[2:8,2:7]))*beta,nrow=7,ncol=6)

    # entries are scaled by beta


# Define initial population at j deg celcius

temperature <- c(10,15,20,25,30)

j <- 1  # Indicates column of data in vc.y and vh.y you are working with,

    # i.e. j=1 : 10 deg, j=2 : 15 deg, etc.

pop0 <- c(C=vc.y[1,j], H=vh.y[1,j])


data.c <- vc.y[,j]

data.h <- vh.y[,j]

y.data <- cbind(data.c,data.h)  # matrix of data at jth temperature
```

```
# Initial parameter (theta) guess
# This is the initial guess at temperature 10 deg C
# Change these values at each temperature
rc <- 1
dcc <- 0.4*(10^(-9))
dch <- 0.4*(10^(-9))
rh <- 0.2
dhh <- 0.5*(10^(-9))
dhc <- 0.45*(10^(-9))

param0 <- c(rc,dcc,dch,rh,dhh,dhc)
t <- seq(0,6,by=.1) # time grid
l <- length(t)

# Lotka Voltera Model function
f.model <- function(time,pop,param){
  dC <- param[1]*pop[1]*(1-(param[2]*pop[1] + param[3]*pop[2]))
  dH <- param[4]*pop[2]*(1-(param[5]*pop[2] + param[6]*pop[1]))
  return(list(c(dC,dH)))
}

# Solve differential equation function
sde <- function(param,time,f,pop){
  sol <- ode(y = pop,times = time, func = f, parms = param, method = "rk4")
  return(sol) # returns a matrix length(time) by 3
}
```

```
# Use this function to help find initial parameters
# This solves the ODE for a given theta guess
guess <- function(param,time,f,pop){
  y.theta.all <- c()
  y.theta <- c()
  call.sde <- sde(param,time,f,pop)
  y.theta.all <- call.sde[,2:3] # only keep C and H data, disregard t column

  # only keep y.theta(t.i) values for each data point t.i
  for(i in 1:length(time)){
    if(time[i]%%1 == 0){
      y.theta <- rbind(y.theta,y.theta.all[i,]) #should be a 7 by 3 matrix
    }
  }
  return(y.theta)
}

# Least squares function
LS <- function(param,time,f,pop,y.values){
  # y.values will usually be the y values of the data, except when bootstrap
  # call sde function to compute y.theta(t) after initializing
  y.theta.all <- c()
  y.theta <- c()
  call.sde <- sde(param,time,f,pop)
  y.theta.all <- call.sde[,2:3] # only keep C and H data, disregard t column
  # only keep y.theta(t.i) values for each data point t.i
  for(i in 1:length(time)){
    if(time[i]%%1 == 0){
      y.theta <- rbind(y.theta,y.theta.all[i,]) #should be a 7 by 3 matrix
    }
  }
  error <- sum((y.values - y.theta)^2)
  return(error)
}

# Find theta.hat by minimizing least squares function LS
minimizer <- function(param,time,f.min,pop,f,y.values){
  result <- optim(par = param, fn = f.min, gr = NULL, time = time,
  pop = pop, f = f, y.values = y.values, method = c("Nelder-Mead"))
  return(result)
}

# Store output from minimizer function (theta.hat is "parameter")
result <- minimizer(param0,t,LS,pop0,f.model,y.data)
y.fitted <- guess(result$par,t,f.model,pop0) # 7 by 2 matrix,
    # with first column population C, second population H
parameter <- result$par

# Plots that help when trying to find good param0
plot(vc.x,guess(param0,t,f.model,pop0)[,1],xlab = "Time (hours)",
ylab = paste("Population (cells/mL) ", beta^-1), main = "V.chagasii")
```

```
plot(vh.x,guess(param0,t,f.model,pop0)[,2],xlab = "Time (hours)",
ylab = paste("Population (cells/mL) ", beta^-1), main = "V.harveyi")

# Plots of data, fitted values, and line connecting fitted values
par(mfrow=c(1,2))

plot(vh.x,y.data[,1],xlab = "Time (hours)",
    ylab = paste("Population (cells/mL) ", beta^-1))
points(vh.x,y.fitted[,1],col = "blue", pch = "x")
lines(t,sde(result$par,t,f.model,pop0)[,2])
legend(x=0,y=12,legend=c("Data Point", "Fitted Value"),
       box.lty=2, col=c("black", "blue"), pch = c("o","x"))

plot(vh.x,y.data[,2],xlab = "Time (hours)",
    ylab = paste("Population (cells/mL) ", beta^-1))
points(vh.x,y.fitted[,2],col = "red", pch = "x")
lines(t,sde(result$par,t,f.model,pop0)[,3])
legend(x=0,y=10,legend=c("Data Point", "Fitted Value"),
       box.lty=2,col=c("black", "red"), pch = c("o","x"))

print(result)
print(cbind(par = c("rc","dcc","dch","rh","dhh","dhc"),
    value = c(result$par[1],beta*result$par[2:3],result$par[4],
    beta*result$par[5:6])))

# Bootsrap Estimation of SD
ec <- y.data[,1] - y.fitted[,1] #error terms for c(t)
```

```
sd.c <- sd(ec)


eh <- y.data[,2] - y.fitted[,2] #error terms for h(t)

sd.h <- sd(eh)


boot <- function(par,B){

  param_star <- c()

  for(k in 1:B){

    ec_star <- rnorm(7,mean = 0, sd = sd.c)

    eh_star <- rnorm(7,mean = 0, sd = sd.h)

    y_star.c <- y.data[,1] + ec_star

    y_star.h <- y.data[,2] + eh_star

    y_star <- cbind(y_star.c, y_star.h) # 7 by 2 matrix

    param_star <- rbind(param_star,

        (minimizer(par,t,LS,pop0,f.model,y_star))$par)

  }

  return(param_star)

}


theta.star <- boot(parameter,300)


st.dev <- c() # each column is the standard deviation of the

    # ith component of theta_star

for(i in 1:6){

  if(i == (1|3)){

    st.dev <- cbind(st.dev, sd(theta.star[,i])) # do not scale growth rate

  }
  else{
    st.dev <- cbind(st.dev, sd(beta*theta.star[,i]))
        # competition coefficients of "theta.star" are scaled
  }
}


# 95% CI
# Scale competitive ability components of "parameter", but not growth rate
ci <- c()
for(i in 1:6){
  if(i == (1|3)){
    ci.upper <- parameter[i] + qnorm(0.025,lower.tail = FALSE)*st.dev[i]
    ci.lower <- parameter[i] - qnorm(0.025,lower.tail = FALSE)*st.dev[i]
    ci <- rbind(ci,cbind(ci.lower,ci.upper))
  }
  else{
    ci.upper <- beta*parameter[i]+qnorm(0.025,lower.tail = FALSE)*st.dev[i]
    ci.lower <- beta*parameter[i]-qnorm(0.025,lower.tail = FALSE)*st.dev[i]
    ci <- rbind(ci,cbind(ci.lower,ci.upper))
  }
}
```

Appendix B: Initial Parameters

	$10°C$	$15°C$	$20°C$	$25°C$	$30°C$
r_c	1	0.05	0.8	1	0.5
δ_{cc}	0.4×10^{-9}	0.35×10^{-4}	0.4×10^{-4}	0.4×10^{-4}	0.4×10^{-4}
δ_{ch}	0.4×10^{-9}	0.35×10^{-5}	0.4×10^{-5}	0.4×10^{-5}	0.4×10^{-4}
r_h	0.2	0.05	0.9	0.2	0.4
δ_{hh}	0.4×10^{-9}	0.25×10^{-4}	0.5×10^{-6}	0.5×10^{-7}	0.5×10^{-4}
δ_{hc}	0.4×10^{-9}	0.25×10^{-4}	0.45×10^{-6}	0.45×10^{-6}	0.45×10^{-4}

Table 4.4: Initial parameter guesses that lead to the least squares being minimized, for each temperature.