AN ABSTRACT OF THE THESIS OF

Rebecca C. Pankow for the degrees of Honors Baccalaureate of Science in Biochemistry & Biophysics and Honors Baccalaureate of Computer Science presented on July 15, 2011. Title: The Development of Computational Methods for Studying Plant-Pathogen Interactions.

Abstract approved:

_____

Jeff Chang

Modern technology has enabled the advancement of biological research through the use of powerful machines and computers as well as innovative computer programs. Advances in sequencing technology and software enable us to make de novo assemblies of organism genomes, and the development of specialized computer programs can automate routine but tedious tasks that are done by hand.

We are studying plant-bacteria interactions. The immune response of a plant to a pathogen can be quantified by counting callose spots in pictures of leaves. The need for an objective and fast method to count these spots led to the development of AutoSPOTs. This computer program, written in Perl, identifies and counts spots in images based on spot color and size, and includes features that previous spot-counting software did not have. AutoSPOTs has user-configured filters that can be used to both include and exclude different types of spots from being counted.

We have also de novo assembled a draft genome of the plant pathogen *Rhodococcus fascians*. This draft genome has been assembled from both mate-pair and paired-end reads. I wrote a set of scripts to filter out poor-quality reads from the dataset in order to improve the quality of our assembly. We are confident in our assembly due to the presence of a contig that is a potential match for *R. fascian*'s linear plasmid, as well as its consistency when assembled from different sets of reads.

Key Words: microbial genomics, pathogenesis, software development

Corresponding e-mail address: pankowr@onid.orst.edu

The Development of Computational Methods for Studying Plant-Pathogen Interactions

by

Rebecca C. Pankow

A PROJECT

Submitted to

Oregon State University

University Honors College

in partial fulfillment of
the requirements for the
degree of

Honors Baccalaureate of Science in Biochemistry & Biophysics (Honors Associate)
Honors Baccalaureate of Science in Computer Science (Honors Associate)

Presented July 15, 2011

Honors Baccalaureate of Science in Biochemistry & Biophysics and Honors Baccalaureate of Science in Computer Science project of Rebecca C. Pankow presented on July 15, 2011.

APPROVED:

Mentor, representing Botany and Plant Pathology

Committee Member, representing Computer Science

Committee Member, representing Molecular and Cellular Biology

---

Chair, Department of Biochemistry and Biophysics

Chair, Department of Electrical Engineering and Computer Science

---

Dean, University Honors College

I understand that my project will become part of the permanent collection of Oregon State University, University Honors College. My signature below authorizes release of my project to any reader upon request.

Rebecca C. Pankow, Author

ACKNOWLEDGEMENTS

AUTOSPOTs: AUTOMATED IMAGE ANALYSIS FOR ENUMERATING CALLOSE

I thank Caitlin A. Thireault, Allison Smith, and Philip Hillebrand for their assistance. I gratefully acknowledge Jim Carrington for use of his light microscope, and William Thomas for taking numerous pictures of callose deposits and providing the data for AutoSPOTs.

DE NOVO ASSEMBLY OF THE *RHODOCOCCUS FASCIANS* GENOME

I thank Jeff Chang for preparing our DNA samples, and the University of North Carolina High Throughput Sequencing Facility and the Oregon State University Center for Genome Research and Biocomputing for sequencing our samples. I also acknowledge Allison Creason for her hard work with Velvet 1.1.02 as well as Dr. Loper and Melodie Putnam for their collaborative work.

CONTRIBUTIONS OF CO-AUTHORS

AUTOSPOTs: AUTOMATED IMAGE ANALYSIS FOR ENUMERATING CALLOSE

Jason Cumbie and Jeff Chang contributed writing to this chapter.

DE NOVO ASSEMBLY OF THE *RHODOCOCCUS FASCIANS* GENOME

Jeff Chang contributed to the sequencing portion of Materials and Methods as well as the descriptions of *Rhodococcus fascians* and Illumina sequencing. Allison Creason contributed to the descriptions of the assembly parameters that we used for Velvet 1.1.02.

TABLE OF CONTENTS

LIST OF FIGURES

DEDICATION

To my mother and sister for all their love and support.

To my mentor for his help, patience, teaching, and all the time he has taken to work with me.

To the Chang lab for all their hard work.

THE DEVELOPMENT OF COMPUTATION METHODS FOR STUDYING PLANT-PATHOGEN INTERACTIONS

INTRODUCTION

Plant Immunity

One of our goals is to understand how bacteria infect and survive within plants. While plants do not have an immune system analogous to those of animals, plants still have defense systems to protect them from pathogens. Each plant cell has an innate immunity against a variety of pathogens, and this immunity, or defense response, is triggered by the cell's detection of pathogen-associated molecular patterns (PAMPs) and pathogen effectors. The detection of PAMPs outside the cell elicits PAMP-triggered immunity (PTI), while the detection of effectors inside the cell leads to effector-triggered immunity (ETI). These defense systems protect plants from infections by pathogens, while in turn plant pathogens must overcome these systems in order to successfully reproduce and thrive inside the plant (Jones et al, 2006).

The sum total of the effects that host plants and plant pathogens have on each other can be described as bacteria-plant interactions. Better understanding of these interactions may lead to new treatments for pathogens affecting crops and ways to genetically modify agricultural plants to be resistant to pathogens. Crop plants genetically modified to be resistant to pathogens would be less susceptible to disease, thereby increasing yields (Dodds et al., 2010).

Increases in crop yields are of importance to food security and areas that have trouble producing enough food to meet its population's needs. Furthermore, as some plant pathogens are closely related to certain animal pathogens, information learned about the ways these plant pathogens interact with their hosts may be applicable to how animals can be infected by similar pathogens (Letek et al., 2010).

PAMP-triggered Immunity (PTI)

The presence of pathogen-associated molecular patterns (PAMPs) and danger-associated molecular patterns (DAMPs) outside of a plant cell trigger PTI. A pathogen-associated

molecular pattern is generally a part of the pathogen itself, such as flagellin. DAMPs are parts of the plant cell itself, such as the cell wall, which come from the damages that the pathogen causes to the host cell (Dodds et al., 2010). PAMPs and DAMPs are recognized outside of the plant cell by PRRs, or pattern recognition receptors, most of which are located in the plant cell membrane (Jones et al, 2006).

Plants have different types of PRRs, such as Toll-like receptors and C-type lectin receptors, which recognize different classes of PAMP molecules (Zipfel, 2008). PAMPs are highly conserved among bacteria and are not released easily inside the plant (Jones et al, 2006). This latter feature is understandable, given this would decrease the likelihood of eliciting PTI and thereby increasing the bacteria's chances of survival and infecting the plant.

The detection of PAMPs triggers complex cellular signaling pathways and cascades and results in the production of reactive oxygen and nitrogen species in the cell, callose deposition in the cell wall, transcription factor activation, and the pumping of calcium ions outside the cell (Ausubel, 2005; Zipfel et al., 2008). PTI also involves the suppression of the response of the plant to its own growth hormones (Jones et al., 2006). These immune responses increase the ability of the plant to resist infection by a pathogen, and are the first line of a cell's innate immunity defense (Jones et al., 2006).

One way to measure host plant/bacteria interactions is to use the deposition of callose in cells as a molecular readout of defense. The detection of PAMPs by a plant cell elicits the deposition of callose in the plant cell wall. This deposition thickens the cell wall and thereby increases the cell's protection from pathogens, which reside outside the cell wall (Zipfel et al., 2008). This would also decrease the ability of the pathogen to inject virulence factors into the plant cell. It is thus in the pathogen's interest to decrease the amount of callose deposition in the plant cell wall. If the pathogen is able to counteract the deposition with its effector proteins, it would increase its virulence.

We compared images of callose deposits in cells exposed to non-pathogenic bacteria to images of callose deposits in cells exposed to pathogenic bacteria. Pathogenic bacteria are able to inject effectors into the plant cell in order to suppress the immune response of callose deposition in the cell wall. Comparing the number of callose deposits between the cells exposed to non-pathogenic bacteria and the cells exposed to pathogenic bacteria provides a baseline of plant immune response. The immune response provoked by non-pathogenic bacteria with a

gene that suppresses callose deposition can be compared to this baseline in order to determine the magnitude of suppression that the non-pathogenic bacteria are able to elicit.

In order to get information on how much callose was deposited in the cell walls, the number of callose spots must be counted. This task was initially done by hand. This was time-consuming, as one set of images could contain hundreds of pictures, each of which could contain tens to hundreds of spots. An additional problem with this method was its subjectivity. A human may have problems with consistency as well as deciding whether or not something should be considered a spot. By its nature, a computer program would have neither of these problems. The slow, inconsistent, and subjective nature of counting spots by hand needed to be replaced with a fast, consistent, and objective solution. This led to the development of AutoSPOTs, an image analysis program that automatically finds spots in a set of images.

Effector-triggered immunity (ETI)

Effector-triggered immunity is an immune response to the detection of effectors injected into the cell by a pathogen. ETI has a higher amplitude of defense than PTI. A plant cell synthesizes NB-LRR proteins, which detect effectors inside the cell environment. These NB-LRR proteins are capable of recognizing a wide variety of pathogen effectors from many different species. An NB-LRR protein is specific to a pathogen effector, and the link between NB-LRR genes and their corresponding effector genes is an example of the gene-for-gene relationship that exists between pathogens and the hosts that they try to infect. Virulence genes help a pathogen infect a host, while resistance genes in the host try to counteract the effectors that the virulence genes encode. If a host cell's NB-LRR proteins are successful, they identify the presence of effector proteins in the cell environment and elicit ETI. If a pathogen's effectors are successful, they evade detection by NB-LRR proteins and suppress the host cell's immune response (Jones et al., 2006).

Pathogen Virulence Proteins

Plant pathogens synthesize and use a variety of molecules to try to infect a plant. These molecules, which contribute to the virulence of a pathogen, include phytohormone mimics and effector proteins. These virulence molecules are injected into the plant cell by the pathogen,

which lives in the intercellular spaces of the plant. The proteins that a plant pathogen introduces into a plant cell are known as effectors, which suppress and interfere with PTI. If a pathogen is able to suppress PTI, then it is more likely to successfully infect the plant. Pathogen effectors operate together and redundantly to disrupt the plant cell's cellular functions and target host proteins involved in PTI. As such, collections of effector proteins are necessary for virulence, but no single effector is necessary. Only a small number of effectors have been characterized for their molecular function. *HopM1* of *Pseudomonas syringae* for example, functions to degrade a protein of the host and disrupt vesicle transport. AvrPto, another *P. syringae* effector, is thought to be a PRR kinase inhibitor, possibly for the suppression of plant ETI response. Other effectors, such as AvrB, AvrRPM1, and AvrRpt2, are theorized to act in concert to target RIN4, a host protein that regulates PTI and ETI (Dodds et al., 2010).

Effector Toxins

Plant pathogens produce toxins, such as antimetabolites and lipodepsipeptidic toxins, in order to increase their virulence against their hosts. These toxins are known to disrupt plant metabolic pathways, destabilize host cell membranes, and suppress plant salicylic-acid defense mechanisms (Lindeberg et al., 2008). Together, effectors and toxins are two classes of pathogenic molecules that interact with plant cells, and increase pathogenic fitness as well as the likelihood that the pathogen will successfully infect the plant.

Some of the characteristic symptoms of infection by plant pathogens are due to their toxins. For example, bean plants affected by halo blight disease commonly have yellow halos around leaf lesions. These halos are caused by phaseolotoxin, a phytotoxin produced by the pathogen *Pseudomonas syringae* (Lindeberg et al., 2008).

Some toxins across different species of pathogenic bacteria are orthologous to each other, making it possible to compare the sequenced genome of a pathogen with the sequences of toxin-producing genes from different species. If sequences similar to known virulence determinants are found within the genome, the genome can be said to contain candidate toxins. Candidate toxins found within a bacterial plant pathogen's genome may not only be similar to other bacterial toxins, they may also be similar to toxins used by bacterial pathogens against insects (Lindeberg et al., 2008). The identification of candidate toxins within a pathogen's genome may lead to the identification of new mechanisms that aid pathogen infection of plants.

Rhodococcus fascians

Much of what we understand regarding plant-pathogen interactions have been derived from studying the interaction between the model plant, *Arabidopsis thaliana* and the model Gram-negative plant pathogen, *P. syringae*. However, characterization of other Gram-negative phytopathogenic bacteria has unveiled new mechanisms of pathogenesis. We are focusing on an understudied plant pathogen, *Rhodococcus fascians*. Isolates of this species has some peculiar and interesting characteristics.

R. fascians, unlike most plant pathogens, is a Gram-positive actinomycete. *R. fascians* is capable of infecting at least 87 plant genera, including ornamental herbaceous perennials (Putnam, et al., 2007). *R. fascians* also has a unique strategy in that it stimulates the host plant to redirect its primary metabolism to synthesize nutrients in the area infected by the pathogen. This results in the conversion of the infected area into immature plant tissue that receives a steady supply of nutrients from the plant and aids the pathogen. It is thought that this redirection occurs either through six phytohormone mimics secreted by *R. fascians* (Stes et al., 2011).

The *Rhodococcus* genus is currently not resolved; of the twelve currently established Rhodoccoci, it is possible that a subset may experience some taxonomic movement or merging (Bell et al., 1998). *Rhodococcus* also includes one animal pathogen, *Rhodococcus equi. Rhodococcus equi* is capable of infecting horses, pigs, cattle, and humans with compromised immune systems (Letek et al., 2010). In humans, *Rhodococcus equi* infects and causes lesions in the lungs and other organs, and up to 55% of infections in AIDS patients can be fatal (Bell et al., 1998). As most of *Rhodococcus equi*'s potential virulence-associated genes are also found in other *Rhodococci* bacteria, it is thought to these genes may have evolved to adapt to animal hosts (Letek et al., 2010).

Members of the *Rhodococcus* genus, except for *R. equi*, have promise as a natural way to degrade pollutants such as substituted hydrocarbons, chlorinated phenols, polychlorinated biphenyls, and pesticides (Bell et al., 1998). *Rhodococcus* may also have other uses such as using its pigment gene as a way to confirm successful transformations, to degrade bitter compounds in fruit juice and thereby improve its flavor, and as a biosensor, due to *Rhodococcus*'s enzymes' ability to degrade substituted phenols and hydrocarbons. Furthermore, *Rhodococcus* bacteria

have also been found in a wide and extreme variety of environments, such as low-nutrient soil, antarctic ice, and marine sediments. The ability to adapt to multiple varied environments, infect plants and animals, and having several potential industrial applications make *Rhodococcus* of particular interest to humans (Bell et al., 1998). At this point, it is not known whether or not *R. fascians* will also possess this trait.

The genome of *R. fascians* has not yet been reported. Furthermore, the mechanism by which *R. fascians* infects plants is largely unknown, and may be related to its linear plasmid (Stes et al., 2011). This linear plasmid is an anomaly among bacterial pathogens, most of which have circular plasmids. A sequenced *R. fascians* genome may be useful in answering questions such as how it loses or obtains virulence and whether it may be susceptible to pesticides.

Next-Generation Sequencing

The genomes of plant pathogens can be used to increase our understanding of bacteria-plant interactions, plant immunity, and pathogen virulence. Genomes can be compared between species, and if two species contain similar genes, then the understanding of how one of the species infects plants may be applicable to the other. Furthermore, the genomes can be screened for potential new genes whose products can be searched for in the pathogen's transcriptome (Lindeberg et al., 2008). If such products are found, then the purpose of the gene can be researched. Verification of potential genes and determination of their purposes in the pathogen can increase understanding of the mechanisms behind their virulence.

Methods for genome sequencing have improved dramatically over the past decade. Next-generation high-throughput sequencing is the sequencing of a large amount of DNA strands at the same time. The sequences of fragments of DNA sample are referred to as reads. The length of a read varies depending on the chemistry of the chosen high-throughput sequencing technique, but reads are generally at least 36 nucleotides long. Modern high-throughput sequencing machines can sequence about 25 billion base-pairs in one day (www.illumina.com).

Next-generation sequencing can be applied to the task of *de novo* genome assembly. High-throughput sequencing is an attractive option for *de novo* assembly due to its rapid and cheap generation of a large amount of reads, as well as the development and existence of specialized programs to reassemble the fragments and produce the original genome.

Furthermore, a finished genome is not always necessary; often times a draft genome is enough for a task (Chain et al., 2009).

De novo assembly is, however, not without its challenges. These include the dependence genome assembly accuracy on seemingly arbitrary assembler program input parameters (MacLean et al., 2009), as well as need to determine the quality of reads used in the assembly. Assemblers are far from perfect, and the pool of reads that assembler programs use to generate a draft genome is often contaminated with erroneous reads (Salzberg et al., 2005). It is therefore necessary to develop methods to identify and filter out these bad reads in order to improve assembly quality. While filtering out reads is not the only step necessary to improve an assembly, it is still required for the generation of a working draft genome (Chain et al., 2009).

AUTOSPOTs: AUTOMATED IMAGE ANALYSIS FOR ENUMERATING CALLOSE DEPOSITION

Background

Computational methods are essential to any genomicist's toolkit. With the continual advances in sequencing technology, there are demands for computational approaches that can keep pace with the different data structures. It is with these in mind that we have developed software programs to further enable integration of genomics with plant-pathogen research. In this chapter, we describe AutoSPOTs, one of the programs that we developed to facilitate high-throughput characterization of bacteria-plant interactions.

The type III secretion system (T3SS) is used by many Gram-negative bacteria to establish interactions with their hosts (Grant et al., 2006). The T3SS is a conduit that deploys bacterial encoded type III effector proteins directly into host cells where they function to manipulate the host for the benefit of the infecting bacterium. In the case of plant pathogenic bacteria, type III effectors are necessary to engage and dampen one layer of plant defense called PAMP-triggered immunity (PTI; Jones and Dangl, 2006). A number of events have been associated with PTI, including the deposition of callose in cell walls (Zipfel, 2009). Callose, a β-1,3 linked glucan, along with cellulose, pectin, lignin, and hydroxyproline-rich proteins, are deposited as an agglomeration believed to function as an apposition to infecting bacteria located in the apoplastic space and to other penetrating-type microbes (Bestwick et al., 1995; Bestwick et al., 1998).

Pseudomonas syringae is an excellent model pathogen of plants. The genome sequences for several strains of *P. syringae* have been completed and mined for candidate type III effector genes (Buell et al., 2003; Feil et al., 2005; Joardar et al., 2005; Almeida et al., 2009; Reinhardt et al., 2009; Studholme et al., 2009). Functional approaches that relied on the availability of the genome sequence have also been used (Chang et al., 2005). One strain in particular, *P. syringae* pv *tomato* race DC3000 (*Pto*DC3000), is intensively studied because of its ability to infect the model host plant, *Arabidopsis thaliana*. *Pto*DC3000 has approximately 30 type III effector genes (Schechter et al., 2006). The challenge now is to understand the functions of all type III effector proteins and how a system of deployed type III effectors is coordinated in the host cell to dampen PTI for the benefit of the infecting bacterium.

AutoSPOTs – for Automated Batch Enumeration of Callose Deposition

Enumerating the deposition of callose is an often-used assay for quantifying PTI and perturbations to PTI. The wet-lab manipulations for this assay are relatively straightforward. The robustness of the assay, however, is affected by the variable host response to pathogen challenge and the obvious solution is to simply increase the number of samples. But, this simple solution is often outweighed by the onerous nature of the callose assay and its analyses.

We have therefore developed AutoSPOTs to mitigate the labor-intensive steps associated with image analyses and their potential associated biases. With user-defined criteria based on size and color, AutoSPOTs automates and batch enumerates aniline-stained callose deposits from JPEG images. AutoSPOTs will also automatically execute a series of standard statistical analyses. We have used AutoSPOTs to analyze thousands of images on a laptop computer. AutoSPOTs is an open-source Graphical User Interface (GUI) written in Perl and C. The software program and user's manual can be downloaded from our website at: http://changlab.cgrb.oregonstate.edu/.

Requirements for AutoSPOTs

Methods for sample preparation have been described (Kim and Mackey, 2008). Yet, some simple steps taken during sample preparation and microscopy can greatly improve the quality of the images for more accurate identification and enumeration of callose deposits. It is important to clear leaves as completely as possible subsequent to sample collection because autofluorescence of the chlorophyll will lead to background fluorescence. Insufficient staining can result in weakly fluorescent callose deposits. We have found that the simple act of staining leaves in aniline blue overnight improves the clarity of callose fluorescence. Proper mounting of leaves is another crucial step in sample preparation; wrinkling of leaves or bubbles in the mounting medium can result in multiple focal planes in a single field of view, making resolution of the entire field difficult. Finally, it is important to use an appropriate exposure time for capturing high-quality images (this may require some trial and error). While the customizable color filter settings make AutoSPOTs functional over a range of exposures, extremes in exposures pose potential problems. Exposure settings that are too low will result in faint or dim

callose deposits whereas exposure settings that are too high will wash out fluorescent spots. Both result in a reduction in the accuracy of AutoSPOTs.

We typically take ten JPEG images per leaf and sample fifteen leaves per treatment. A minimum of two treatments is required. For fully automated batch analysis, AutoSPOTs requires the user to properly name and store JPEG images in a recognizable manner. The two recognized formats are as single numbers (e.g., sample1.jpg, sample2.jpg), or as number-number (e.g., treatment1-1, treatment1-2). Additionally, there must be the same number of JPEG images per sample (leaf) per treatment group. JPEG images should be saved in directories labeled according to treatment groups. If these conditions are not met then some of the automated functions of AutoSPOTs cannot be used.

Defining Filters

AutoSPOTs requires the user to define a size filter and one of two types of color filters. In a subsequent section of this chapter, we show the effects that different color filters have on results. AutoSPOTs will apply the filters on a pixel-by-pixel basis to identify callose deposits for each JPEG image to be analyzed. It is therefore important for the user to capture high quality JPEG images and to establish the proper filter settings for the most uniform, sensitive and accurate identification of aniline-blue stained callose deposits across an experiment.

For the size filter, we recommend starting with minimum and maximum sizes of 20 and 100, respectively, and to refine as needed (see discussion on previewing below). We have included two types of color filters: the RGB and ratio filters. To simplify selection, we have included a ‘color selection assistance’ feature. By selecting pixels of callose deposits from several representative images, the color selection assistance feature will provide the user with the values for each of the criteria required of the RGB or ratio filters. Other criteria include ‘Trip’ and ‘Drop’ thresholds. The former is used by AutoSPOTs to determine which pixels will be considered as part of a stained callose deposit and ‘trips’ AutoSPOTs into expanding a callose deposit. The latter is used by AutoSPOTs to exclude pixels from a stained callose deposit and forces AutoSPOTs to ‘drop’ the pixel from expanding the callose deposit. The user can then determine the average values from multiple pixels of multiple images and set the color filters accordingly.

In most cases, AutoSPOTs performs better with grayscale JPEG images; this may depend

on the camera and staining/de-staining of leaves. We have added a feature that enables all images to be automatically converted to grayscale. When defining the color filter, note that the red, green, and blue channels will have the same value so the ratio filter cannot be used. In contrast, the RGB filter must be used and simply becomes an RGB intensity filter.

AutoSPOTs allows the user to preview the sensitivity and accuracy of the filters. A screenshot of a preview and the GUI is presented (Fig. 1.1). The image will be displayed and each identified callose deposit will be demarked. The total number of callose deposits identified will also be displayed. It is strongly recommended that the user carefully examine several images and adjust the filter settings to find the desired level of sensitivity and accuracy. It is important to preview images with few and many callose deposits (see Fig. 1.3). We caution the user to pay close attention to identification of leaf features such as veins or trichomes as callose deposits as well as incomplete demarcation or over-extension of callose deposits. Incorrect identification of leaf features as callose deposits suggests the filters are too sensitive, whereas inaccurate demarcation of callose suggests the Trip and Drop distances are not correctly set.

It cannot be stressed enough that the successful use of AutoSPOTS will depend on consistent, high-quality images, control treatments to assess the accuracy of filters, application of filters uniformly on all samples of all treatments being compared, and a sufficient number of JPEG images and samples to obtain good statistical power for analysis. Not all callose deposits will be identified, especially those in different focal planes, but as along as all leaves were prepared in a similar manner and JPEG images were photographed under similar settings, there will not be any biases in the results.

We have provided a detailed step-by-step Users Manual available by download from our website.

Image Analysis

Once the user has identified a satisfactory filter setting, AutoSPOTs can automatically batch process all images. Analysis begins by examining each pixel of each image individually to identify those that pass the ‘Trip’ threshold for a color filter. Once the pixels that pass the ‘Trip’ threshold are located, all adjoining pixels are analyzed using a ‘Drop’ threshold, which is usually a more relaxed threshold allowing for spot fading near the edges. Pixels are then continually counted outward until no more adjoining pixels can be found that match the ‘Drop’ threshold

criteria. The number of pixels in a given 'spot' is tallied, and then analyzed using the size threshold. Those that are within the minimum and maximum values set by the user are counted as a single callose deposit.

AutoSPOTs calculates the average number of callose deposits by averaging per JPEG image per leaf per treatment. AutoSPOTS has built-in statistical analysis tools and will generate a statistical report for all treatments against the user-defined control treatment. AutoSPOTs will also plot the data for visual representation. At each step of analysis all the data is saved to text files and directories specified by the user. Copies of every image analyzed with their demarked callose deposits are also stored so the user can inspect the sensitivity and accuracy of the filters.

Demonstration of AutoSPOTs

We used one size filter setting and six different color filter settings in AutoSPOTs to demonstrate their effects on sensitivity and accuracy in enumerating callose deposits from JPEG images (Fig. 1.2). Four of the tested color filter settings used RGB (intensity) values to identify callose deposits from JPEG images that were converted to grayscale. In these cases, the color filter setting was set from least sensitive to overly sensitive by using different values – we noted the drop and trip values had the largest effect on sensitivity. Two of the color filter settings used a ratio or RGB filter to analyze the original color JPEG images.

The treatments we tested were Arabidopsis infected with PtoDC3000, a T3SS-deficient mutant of *Pto*DC3000 (*hrcC*), a soil bacterium with an integrated T3SS-encoding region (EtHAn), and EtHAn carrying the type III effector gene, *hopM1*. *Pto*DC3000 deploys 30 type III effector proteins into Arabidopsis and sufficiently dampens PTI to cause disease. Its ability to dampen the deposition of callose has been repeatedly demonstrated (Hauck et al., 2003; DebRoy et al., 2004; Nomura et al., 2006; Ham et al., 2007). In contrast, since the *hrcC* mutant is incapable of delivering type III effectors, it cannot dampen the deposition of callose or PTI, nor cause disease on Arabidopsis (Niepold et al., 1985; Lindgren et al., 1986; Roine et al., 1997; Hauck et al., 2003; Thilmony et al., 2006). EtHAn was engineered from P. fluorescens Pf0-1 and is devoid of any endogenous type III effectors (Thomas et al., 2009). EtHAn therefore elicits PTI. The type III effector, *HopM1*, is sufficient to dampen the deposition of callose (DebRoy et al., 2004; Nomura et al., 2006; Thomas et al., 2009). A total of fifteen leaves were challenged per treatment, and ten images were randomly taken from each leaf. AutoSPOTs took less than 45 minutes to

automatically analyze the 600 JPEG images.

In general, the trends were similar for each of the six filter settings (Fig. 1.2). However, when the automatically generated statistics were analyzed, it is clear that the filter settings do indeed affect interpretation of data. Based on previous findings, we expected significant differences between *Pto*DC3000 versus the *hrcC* mutant and EtHAn + *hopM1* versus EtHAn treatments. The color filter settings 1-3 resulted in no differences in the conclusions – both comparisons within each of the three settings were statistically significant. However, the color filter setting 1 was clearly the poorest of the three in terms of sensitivity. In contrast, increasing the sensitivity of grayscale analysis (setting 4) or use of color JPEG images (setting 5 and 6) resulted in less desirable results. Thus, increased sensitivity to identify the highest number of callose deposits is not necessarily the most recommended approach.

We visually examined the analyzed JPEG images to understand the results of the different color filter settings (Fig. 1.3). In general, most of the color settings performed fairly well in analyzing areas with few callose deposits. Color filter settings 2, 3 and 6 were the more accurate. In contrast, the different color filter settings resulted in dramatic differences in the analysis of areas in which callose deposits were abundant. Settings 2 and 3 performed fairly well. However, very few callose deposits were identified in JPEG images with dense staining spots when AutoSPOTs used color filter settings 4 - 6. This was a consequence of AutoSPOTs failing to drop pixels and categorizing several callose deposits as one larger spot. These large spots would exceed the maximum of 100 as defined by the size filter and not be counted. Changing the size filter could potentially alleviate this problem to a certain extent. We have analyzed JPEG images provided by another research group and results from analysis of the color images were superior to grayscale images. This could be a consequence of differences in staining/de-staining of leaves or in the microscope camera. It is recommended to try different combinations of filters.

The differences in performance when analyzing JPEG images with sparse and dense callose deposits can lead to very misleading results. For example, we could not detect a significant difference between treatments with *Pto*DC3000 and its *hrcC* mutant under color filter setting numbers 4 and 5. This is because AutoSPOTs was sufficiently accurate in identifying the sparse callose deposits resulting from infection with *Pto*DC3000 but was inadequate in identifying densely distributed callose deposits resulting from infection with the *hrcC* mutant.

Conclusion

We developed AutoSPOTs a simple, user-friendly, and open-source software program to facilitate the high-throughput analysis of JPEG images. AutoSPOTs mitigates labor-intensive data analysis by automating and batch analyzing large sets of JPEG images for callose deposits and comparing results between treatments. AutoSPOTs therefore provides the opportunity to examine larger numbers of type III effectors or host genetic backgrounds for their effects on PTI.

We purposefully developed AutoSPOTs to be a simple program. As a consequence, the filtering scheme that AutoSPOTs uses relies on the user to identify the most suitable combination of filters through careful visual examination of their JPEG images. It is therefore expected that the user will design a properly controlled experiment and capture high-quality and uniform JPEG images for analysis. AutoSPOTs was developed for identification and enumeration of aniline-stained callose deposits but it has potential uses in other applications in studying plant-pathogen interactions, such as enumerating GFP-expressing bacteria.

Figure Legends

Figure 1.1. Screenshot of the Graphical User Interface of AutoSPOTs.

The AutoSPOTs GUI divides its various functions into four tabs. This screenshot of the Filter Settings tab illustrates the Preview Filters functions. Filters are defined and added in the top right section of the tab (settings for the Size filter are shown here). The desired filters are then selected from the Existing Filters menu (note that both color and size filters must be selected). The Use Grayscale option has been selected. Once a set of images has been loaded and an image selected in the lower left portion of the screen, the selected image will be displayed in the lower right display window. Callose deposits identified by the Preview Filters function will be indicated with a box and the total number of deposits identified will be displayed above the JPEG image.

Figure 1.2. Enumeration of callose deposits by AutoSPOTs using different color filter settings.

We infected leaves of Arabidopsis with four different strains of bacteria. We used AutoSPOTs to identify and quantify callose deposits with six different color filter settings. For filters 1 – 4, JPEG images were converted to grayscale. The RGB, Trip, and Drop values respectively, were 130, 40, and 100 for color filter 1; 100, 80, and 100 for filter 2; 90, 50, and 80 for filter 3; and 90, 100, and 100 for color filter 4. For color filters 5 and 6, JPEG images were

analyzed as color images using the color ratio and RGB filters, respectively. Fifteen leaves were infected per treatment and ten images were taken per leaf. Standard errors are shown. For each color filter setting, we compared results of *Pto*DC300 versus the *hrcC* mutant and EtHAn + *hopM1* versus EtHAn. Significant differences are denoted; *p-value ≤ 0.05; **p-value ≤ 0.01.

Figure 1.3. Effects of different color filter settings on the accuracy of AutoSPOTs.

Each set of panels represents the same section of the same two JPEG images analyzed using the six different color filter settings (1-6) described previously. One JPEG image had few callose deposits (A) while the other image was dense with callose deposits (B). Callose deposits identified by the program are indicated with a box. The analyses for color filters 5 and 6 used color images, which we have converted to grayscale for publication purposes.

Figures

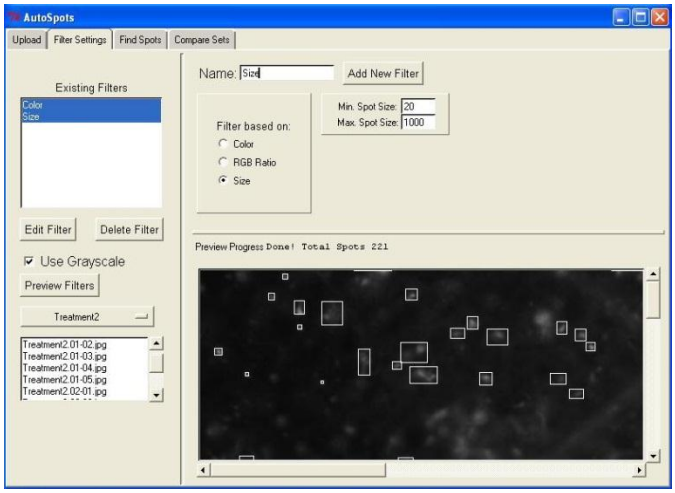

Figure 1.1

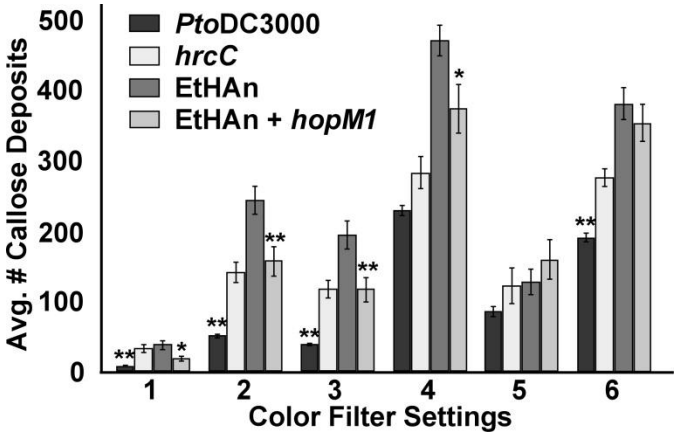

Figure 1.2.

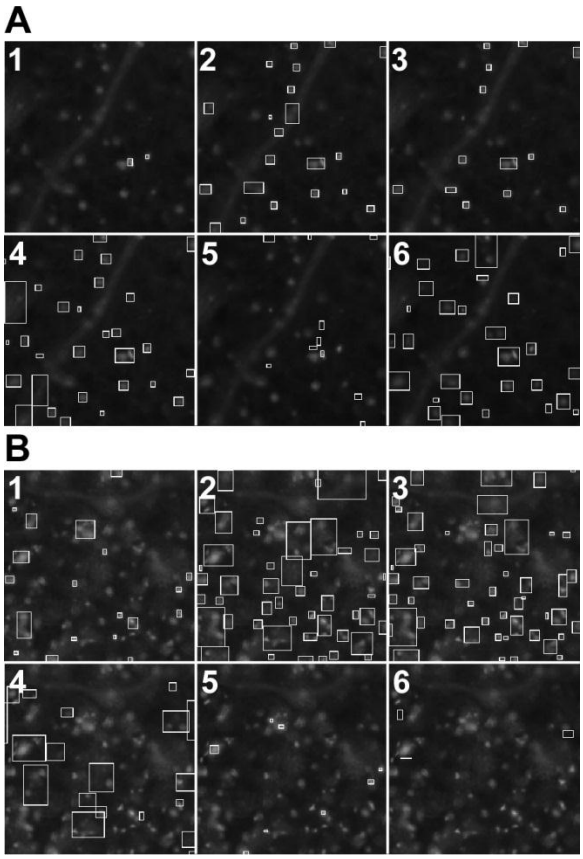

Figure 1.3.

DE NOVO ASSEMBLY OF THE *RHODOCOCCUS FASCIANS* GENOME

Background

Why *Rhodococcus fascians*?

Rhodococcus fascians is a Gram-positive pathogen of plants (Stes et al., 2010). Additionally, *R. fascians* may possess metabolic capabilities with potential applications in bioremediation, as isolates have been found in NASA clean rooms as well as polluted environments (La Duc et al., 2007; Gesheva et al., 2010). This biodegradative potential is a well-known characteristic for members of the *Rhodococcus* genus (Martinkova et al., 2009). Therefore, we are interested in characterizing *R. fascians* to understand its genetic diversity, metabolic capabilities and in particular, its virulence strategies. As a first step, we used next-generation sequencing and *de novo* assembly methods to assemble the *R. fascians* genome. We anticipate the sequenced genome of *R. fascians* will help in the identification of candidate toxins, the discovery of previously unknown genes, fast testing for the presence of *R. fascians* in infected plant tissues, and give insights into the virulence and physiology of this species of plant pathogenic bacteria (Putnam, et al., 2007).

Next-Generation Sequencing Techniques

Next generation (next gen) sequencing has dramatically changed genomics. This technique has dramatically reduced the cost, labor, and time associated with sequencing genomes (Chain et al., 2009). Several next gen platforms are currently available that use different chemistries but share the common characteristic of highly parallelized sequencing. We used the Illumina Genome Analyzer IIx to sequence the genome. Illumina is a next-generation sequencer that sequences DNA in parallel based on imaging of DNA amplified as local clusters on a solid matrix. During each cycle of DNA sequencing, a nucleotide with a fluorescent label is added to the end of the strand. The color of the fluorescent label determines the identity of the nucleotide added to the strand. The Illumina machine captures images of the clusters after each cycle and infers the identity of the added nucleotide based on the color of clusters in the images (www.illumina.com).

Despite major advantages with next gen sequencing, there are nevertheless some substantial hurdles that need to be addressed. First and foremost, the sequencing reads generated by next gen technology are typically short. Our reads, for example, were only 72 nucleotides long. This is very little information per read, as compared to the length of an entire genome. It is also common for genomes to contain regions that are duplicates of each other, and many of these repeat sections are longer than the length of the reads of longer than the sizes of the sequenced fragments (Salzberg et al., 2005). There is therefore increasing ambiguity about read position with decreasing read length. This ambiguity adds to the challenge of *de novo* assembly: it is difficult to determine which duplicate region a read originated from without additional information.

Different types of sequencing approaches can be used with next gen sequencing, with some more suitable than others for addressing the challenges of assembling genomes with repeat sequences. The simplest is the single direction sequencing method. The DNA fragments produced by random fragmentation of the genome can be sequenced 'as-is'; no additional preparation or steps are needed to sequence the fragments. Only one end of a DNA fragment is sequenced. This is the method that requires the least amount of preparation work, but the resulting reads contain the least amount of information as compared to other sequencing methods. A disadvantage is that the sequences are not only very short, as is the case for all Illumina-derived reads, but the reads also contain no information about their location relative to each other in the genome. Thus, this type of sequencing is the least apt for overcoming challenges with sequencing through repeated genome sequences. As a consequence, assemblies that rely solely on single-direction sequencing tend to be comprised of 1000s of short contiguous DNA sequences (contigs), i.e. the assembly is highly fragmented with less information on the relationship between fragments.

A slightly more complicated method is called paired-end sequencing. With paired-end sequencing, both ends of a DNA fragment are sequenced (www.illumina.com). The approximate distance between the pairs of reads can be known if only DNA fragments of a specific size are selected for sequencing. Paired-end reads provide additional information about read location during assembly; each read is known to be about a certain distance away from its corresponding paired read, unlike with single-direction sequencing. This can help with the problem of extremely short reads, as it acts a "longer" read and gives more sequence information over a longer distance in the genome. However, while the fragments of paired-end sequencing are

"longer" than those of single-direction sequencing, they are still not large enough to cover and resolve genome repeats that are larger than the sequenced fragment size.

Finally, the most technically challenging technique is referred to as mate paired sequencing. The one distinction between paired end and mate pair sequencing is the size of the DNA fragment that is sequenced. With mate paired sequencing fragments of 2 kb to 5 kb are sequenced. This single difference helps substantially to assemble reads that correspond to genome-wide repeats, reduce the number of contigs and generate high-quality genome assemblies. Library preparations for mate pair sequencing are, however, not trivial, and can lead to downstream bioinformatic challenges. The first step of mate-pair sequencing involves randomly fragmenting the genome into fragments that are several kb in length. The ends of these fragments are then biotinylated, circularized, and fragmented once more into fragments hundreds of bp long. The fragments with biotin attached to it, i.e. the fragments that contain both the beginning and the end of the initial circular fragment, are isolated and sequenced, giving reads that are approximately 3000 bp apart (www.illumina.com).

Next-generation sequencing introduces new challenges for genome assembly. A typical dataset for genome assembly can contain millions of short reads. In order to create a high-quality draft assembly, poor-quality reads must be filtered out of the dataset and the assembly must cover at least 90% of the original genome (Chain et al., 2009). An additional challenge is the reduction of the number of assembly contigs. Poor-quality reads contain incorrect information, such as sequencing errors and in the case of paired end of mate pair sequencing, the wrong distance between pairs. For our purposes, we were most concerned with errors that arise from mate pair sequencing as this is a known challenge.

Biotin is used to facilitate the attachment of the ends of a DNA fragment to each other, resulting in a circular fragment of DNA with its beginning and end attached to each other. These circular fragments of DNA can then be re-fragmented, and the fragments containing the beginning and end of the original fragment can be isolated and sequenced as two reads. If the original fragment was of a known size, then the distance between the beginning and ending reads is also known. However, during preparation, some of the isolated fragments will not have ends that are the expected distance apart (Phillippy, et al., 2008). Another known challenge for assembling *de novo* genomes from short reads is the generation of artifacts. Instead of generating a linear sequence of nucleotides, a preliminary, unprocessed draft genome may contain loops and branches (Chain et al., 2009). Such errors reduce the accuracy and quality of a

draft genome, and this can be aggravated by the existence of poor-quality reads in the pool used to assemble the genome. By removing poor-quality reads from a dataset, one can reduce the number of artifacts present in a draft genome.

Assuming perfect sequencing and preparation of the genomic samples, one would expect the pairs of reads to be evenly distributed across the entire genome. However, this is not the case in practice. Sequencing reads will be biased for or against certain regions of the genome. These biases lead to sequence gaps and physical gaps in the assembly. Physical gaps exist between a set of contigs; as the sequences between the contigs and the order of the contigs relative to each other are unknown. This lack of information prevents associating contigs together. Sequence gaps are sections in a contig that are completely unknown. They arise from knowing the flanking sequences of a mate pair but not the sequence in-between the mate pair reads (Fig. 2.1). While sequence gaps can be filled in easily since the flanking sequences of the gap are known, physical gaps are more difficult since this information is not known.

Meeting Next-Generation Sequencing Challenges

Our goal is to develop the highest quality assembly for *R. fascians*. In order to have a quality assembly, the pool of reads used to generate the assembly must be as accurate as possible. Poor-quality reads need to be filtered out of the pool of reads used for the assembly of the genome. It is necessary to check and assess read quality before the final assembly of a genome. To this end, I developed and used Perl scripts to help filter out reads that are artifacts from library construction and reduce misassemblies.

Materials and Methods

Sequencing

R. fascians was grown in LB media for two days with shaking at 30 ºC. Total genomic DNA was extracted using the Promega Wizard Genomic DNA preparation kit, according to the instructions of the manufacturer (Promega, Madison, WI.). The DNA was sequenced using two separate methods. One was sequenced via mate pair sequencing, and the other via paired-end sequencing. For the former method, fragments of 200 bp were size-selected and prepared for sequencing as recommended by the manufacturer (Illumina, San Diego, CA). The first step of paired-end sequencing requires the attachment of adapters to the ends of the DNA fragments.

Each strand in the DNA fragment has an adapter attached to it, and the adapters are at opposite ends of each other. These adapters attach to the flowcell, resulting in the strands of the DNA fragment separating and being opposite of their normal orientation. This antiparallel alignment enables both ends of the DNA fragment to be sequenced at the same time. The paired-end sequencing was done by Oregon State University's Center for Genome Research and Biocomputing. The Illumina Genome Analyzer IIx was used to sequence the sample for 2x80 cycles. 25 million paired-end reads of length 80 bp were obtained. For the latter, library preparation and sequencing were done by the High Throughput Sequencing Facility at University of North Carolina (2x80 cycle sequencing). Fragments of 3kb were size-selected, and 62 million mate pair reads of length 80 bp were obtained after sequencing (Table 2.1).

Filtering Out Poor-Quality Reads

I developed a Perl script to check the quality of our reads by comparing the distances between each read in a mate pair after being mapped back to a very strict initial Velvet 1.1.02 assembly with a hash length of 63. The two reads of a mate pair are expected to be within a certain distance of each other, based on the size-selected fragments of 3000 nt for the mate pair sequencing. Reads that were not the correct distance apart were removed from the dataset, and another, less strict, assembly was formed using the remaining reads and a hash length of 61. This process was repeated once more with a hash length of 55.

A final assembly was created using the 55 million remaining reads and the following parameters: for velveth, the parameters used were a hash length of 61 and two short Paired libraries, one for the paired-end file and one for the mate pairs file. The velvetg parameters were an expected coverage of 100, a coverage cutoff of 20, a minimum contig length of 100, and insert length of 200 for the paired end library, and an insert length of 3000 for the mate pair library.

Results

A well-known challenge with mate pair sequencing is the generation of artifactual mate pair reads that are less than 3000 bp apart. Depending on the library preparation, these reads can represent anywhere from 5% to 40% of the total reads. If they are not filtered out prior to genome assembly, artifactual reads could potentially cause misassembly errors.

I used Perl to develop a script for filtering out artifactual reads from the pool (Appendix). This was done by comparing the distances between the reads in the assembly versus their actual approximate distance based on the technique used to sequence them. However, one challenge we faced was the absence of a known suitable reference genome to infer mate pair distances. Thus, as a first step, we used only the paired-end reads and Velvet with a high hash length parameter, to generate a high confident but fragmented genome assembly. Even though the resulting contigs were small, our logic here was that with strict parameters, the contigs would be more reliable and contain fewer misassemblies.

In the next step, we used the short read alignment program, CASHX (Fahlgren et al., 2009) to align the mate pair reads to the preliminary reference genome assembly. A CASHX file contains the identification strings of reads and the locations of where these reads were found in an assembly (if they were found at all). The following is a typical sample of part of a CASHX file:

ID=62DULAAXX:6:1:2993:945#0_1_RC HITS=0
ID=62DULAAXX:6:1:2993:945#0_2_RC HITS=0
ID=62DULAAXX:6:1:3178:948#0_1_RC HITS=0
ID=62DULAAXX:6:1:3178:948#0_2_RC HITS=0
ID=62DULAAXX:6:1:3401:931#0_1_RC HITS=1
Accession=NODE_66_length_1681433_cov_73.324013 Start=47984 End=48055
Strand=1
ID=62DULAAXX:6:1:3401:931#0_2_RC HITS=1
Accession=NODE_66_length_1681433_cov_73.324013 Start=51269 End=51340
Strand=-1
ID=62DULAAXX:6:1:5277:953#0_1_RC HITS=1
Accession=NODE_49_length_2179869_cov_94.802094 Start=1033722 End=1033793
Strand=-1
ID=62DULAAXX:6:1:5277:953#0_2_RC HITS=1

In this format, a read's identification string is displayed (for example, "62DULAAXX:6:1:5277:953#0_1_RC"), how many times it "hit" the genome it was compared against (HITS=n), the identity of where the read matched against the draft genome (for example, "NODE_66_length_1681433_cov_73.324013"), and where the read matched the draft genome

(given by the “START” and “END” -- also known as start and stop -- coordinates). Regular expressions and substring functions can easily be used to parse a CASHX file.

The distances between pairs of reads were obtained from a CASHX output file in several steps. Only the pairs of reads that had at least one of the reads map back to the assembly unambiguously to the genome were used. A script was written to identify the pairs of reads from the output file that met this criterion, using regular expressions, and calculate the distances between them (if necessary). These distances were output to a text file.

The information in the text file was then simplified: each distance in the text file was divided by 100 and the decimal part of the result was discarded. The number of times each result was obtained was tracked (ex. 33 was obtained 2 times, 34 was obtained 6 times). The tracking was done with an array, where the ith position in the array corresponded to a result of i, and where the value of the array at position i equaled the number of times i had been calculated. The size of the array was determined by first scanning the file for the largest number, which, divided by 100 and floored, gave the needed array size.

This same script also identified and filtered out poor-quality reads from the dataset. The steps for this task were as follows: those pairs of reads that had each read in the pair map back exactly once to the genome and had a distance difference greater than or equal to 3 kb, or those pairs of reads that had either read “hit” the genome exactly once and the other not at all, were classified were classified as “good reads”. If the distance between reads in a pair was between 300 to 400 base-pairs, inclusive, such pairs were classified as “bad reads”.

The CASHX output files did not contain the original sequences of the reads. This information is contained in the reads’ “fasta” files. The following illustrates the format of a fasta file, where the ID of a read is listed after a “>”, and the sequence of the read follows after the ID line.

>62DULAAXX:6:97:8719:11753#0_1_RC
TCGTTCACCGGCCCGCTGATCGGATCGGTACTGCTGGTCGTCGACTTCCGCGTTGCCTGCACGGTCGCTG
CC
>62DULAAXX:6:97:8719:11753#0_2_RC
CCTACAACAAGCTCAGCGAGATCGCACCGACCGTCGCGCGGCCGGTTGGCACTGCGGCGTACGCGGTCC
CTC
>62DULAAXX:6:108:15261:1072#0_1_RC

TGTGATCGACAAAAGCAGCCCAGCTGTACGTAGCTCAGCTACGAAGAAGAGTGCAACAGCCGGTAGCG
GCGA
>62DULAAXX:6:108:15261:1072#0_2_RC
GCACGACGCGACCTCGTGTTGGAACGCGACGAGATCGCGTCCACCATCGAAATCCGACGCCTTGCACGT
GTC

Since CASHX output files only contained the information as to where reads mapped back to the genome, while the fasta files contained only the read sequences, it was therefore necessary to use both a fasta file and its CASHX file to create a new fasta file that contained only the "good reads". The previous script, in addition to identifying how many times each read in a pair mapped back to the genome and the distances between reads in a pair, output a new fasta file containing all reads that met the criteria for a "good" pair of reads, and another fasta file that contained all the reads that met the criteria for a "bad" pair of reads. In this way, the high-quality reads were extracted from the larger pool of reads (Table 2.1).

The histogram of the distances between mate pairs mapped back to an assembly shows a bimodal distribution (Fig. 2.2). Most mate pair reads were within 2.5kb and 3.8 kb of each other, while a small minority of the reads is between 0 and 500 bp. It was expected that between 5-40% of paired-end reads will be of poor quality. In our case, approximately 5% of our paired-end reads were incorrect, indicating that the genomic DNA was prepared well. This bimodal distribution demonstrates the easy identification of poor-quality reads and the feasibility of using distances between pairs of reads to filter out erroneous reads from a dataset.

Discussion

High-throughput sequencing was used to generate a draft genome. The Illumina Genome Analyzer IIx was used to sequence two sets of reads. The University of North Carolina prepared and sequenced mate-pair reads using the Illumina Genome Analyzer IIx. We prepared and sequenced paired-end reads using the CGRB's Illumina Genome Analyzer IIx at OSU. These two sets of reads were assembled into our draft genome using Velvet 1.1.02.

Scripts and methods were developed for the identification and filtering of poor-quality reads from our dataset. These were used to identify pairs whose reads were either too close or two far from their expected distance, and filter these out from the dataset. Suspect regions of the assembly were also identified using a script that mapped which sections of the assembly had no reads mapping back to it.

We are confident in our draft genome assembly and are in the process of annotating and analyzing it. Results that support our confidence include the following: one of our contigs matched the expected size of a known *R. fascians* plasmid, and this contig also contains genes known to be on that plasmid. We have also assembled our draft genome using different sets of reads, and we consistently get the same contigs after each assembly. We are also currently sequencing another *R. fascians* genome. After the draft of this second genome is completed, we will compare the synteny, or order of genes in each draft assembly. If the gene order in both assemblies is identical, this supports that our assembly is good (Salzberg et al., 2005).

Figures and Tables

	Length of Reads	# of Reads	# of Reads used to assemble
Paired End	72bp	25 million	25 million
Mate Pairs	72bp	62 million	55 million

Table 2.1. The filtering of contaminated mate pairs from the dataset.

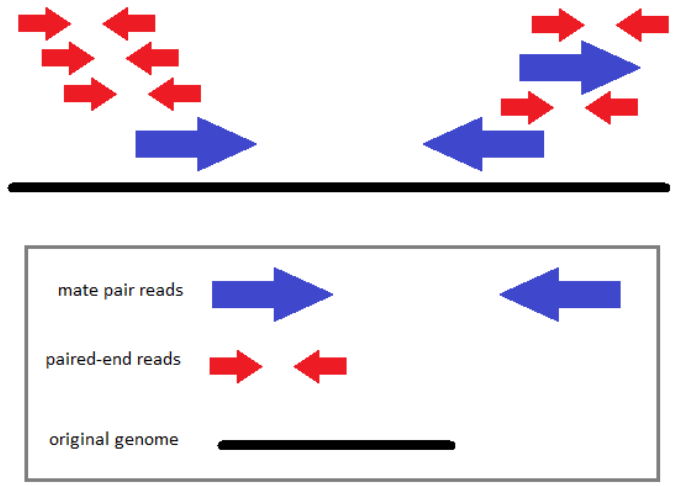

Figure 2.1. A typical sequence gap.

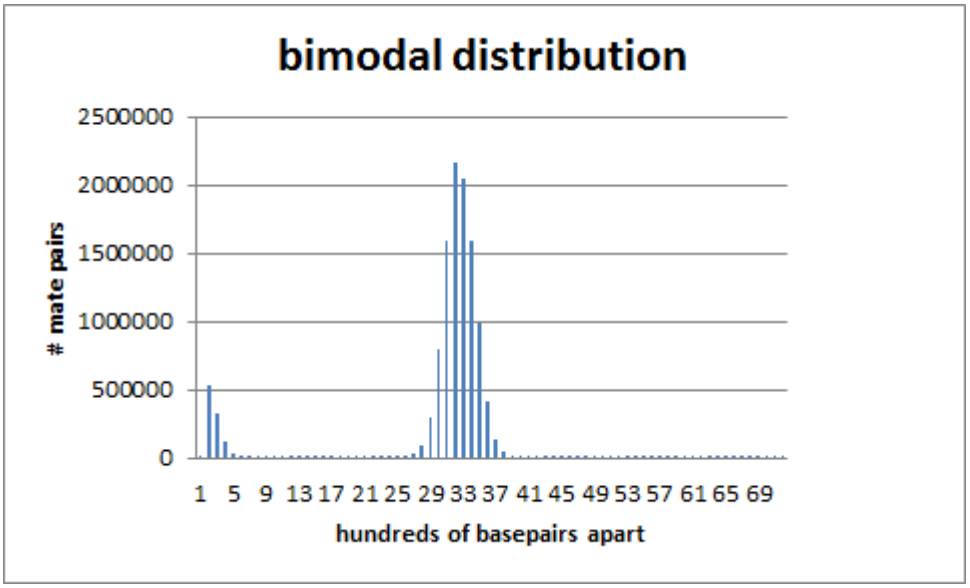

Figure 2.2. The bimodal distribution of mate pairs mapped back to a strict assembly.

BIBLIOGRAPHY

Almeida, N.F., Yan, S., Lindeberg, M., Studholme, D.J., Schneider, D.J., Condon, B., Liu, H., Viana, C.J., Warren, A., Evans, C., Kemen, E., Maclean, D., Angot, A., Martin, G.B., Jones, J.D., Collmer, A., Setubal, J.C., and Vinatzer, B.A. 2009. A draft genome sequence of *Pseudomonas syringae* pv. *tomato* T1 reveals a type III effector repertoire significantly divergent from that of *Pseudomonas syringae* pv. *tomato* DC3000. Mol. Plant-Microbe Interact. 22:52-62.

Ausubel, F. 2005. Are innate immune signaling pathways in plants and animals conserved? Nature Immunology. 6:973-979.

Bell, K., Philip, J., Aw, D., and Christofi, N. 1998. The Genus *Rhodococcus*. Journal of Applied Microbiology. 85: 95-210.

Bestwick, C.S., Bennett, M.H., and Mansfield, J.W. 1995. *Hrp* mutant of *Pseudomonas syringae* pv *phaseolicola* induces cell wall alterations but not membrane damage leading to the hypersensitive reaction in lettuce. Plant Physiol. 108:503-516.

Bestwick, C.S., Brown, I.R., and Mansfield, J.W. 1998. Localized changes in peroxidase activity accompany hydrogen peroxide generation during the development of a nonhost hypersensitive reaction in lettuce. Plant Physiol. 118:1067-1078.

Buell, C.R., Joardar, V., Lindeberg, M., Selengut, J., Paulsen, I.T., Gwinn, M.L., Dodson, R.J., Deboy, R.T., Durkin, A.S., Kolonay, J.F., Madupu, R., Daugherty, S., Brinkac, L., Beanan, M.J., Haft, D.H., Nelson, W.C., Davidsen, T., Zafar, N., Zhou, L., Liu, J., Yuan, Q., Khouri, H., Fedorova, N., Tran, B., Russell, D., Berry, K., Utterback, T., Van Aken, S.E., Feldblyum, T.V., D'Ascenzo, M., Deng, W.L., Ramos, A.R., Alfano, J.R., Cartinhour, S., Chatterjee, A.K., Delaney, T.P., Lazarowitz, S.G., Martin, G.B., Schneider, D.J., Tang, X., Bender, C.L., White, O., Fraser, C.M., and Collmer, A. 2003. The complete genome sequence of the Arabidopsis and tomato pathogen *Pseudomonas syringae* pv. *tomato* DC3000. Proc. Natl. Acad. Sci. U S A 100:10181-10186.

Chain, P., Grafham, D., Fulton, R., Fitzgerald, M., Hostetler, J., Muzny, D., Ali, J., Bruce, D., Buhay, C., Cole, J., Ding, Y., Dugan, S., Field, D., Garrity, G., Gibbs, R., Graves, T., Han, C., Harrison, S., Highlander, S., Hugenholtz, P., Khouri, H., Kodira, C., Kolker, E., Kyrpides, N., Lang, D., Lapidus, A., Malfatti, S., Markowitz, V., Metha, T., Nelson, K., Parkhill, J., Pitluck, S., Qin, X., read, T., Schmutz, J., Sozhamannan, S., Sterk, P., Strausberg, R., Sutton, G., Thomson, N., Tiedje, J., Weinstock, G., Wollam, A., Genomic standards Consortium Human Microbiome Project Jumpstart Consortium, Detter, J. Genome Project Standards in a New Era of Sequencing. 2009. Science 326: 236-237.

Chang, J.H., Urbach, J.M., Law, T.F., Arnold, L.W., Hu, A., Gombar, S., Grant, S.R., Ausubel, F.M., and Dangl, J.L. 2005. A high-throughput, near-saturating screen for type III effector genes from *Pseudomonas syringae*. Proc. Natl. Acad. Sci. U S A 102:2549-2554.

DebRoy, S., Thilmony, R., Kwack, Y.B., Nomura, K., and He, S.Y. 2004. A family of conserved bacterial effectors inhibits salicylic acid-mediated basal immunity and promotes disease necrosis in plants. Proc. Natl. Acad. Sci. U S A 101:9927-9932.

Dodds, P., and Rathjen J. 2010. Plant immunity: towards an integrated view of plant-pathogen interactions. Nature Reviews 11:539-548.

Feil, H., Feil, W.S., Chain, P., Larimer, F., DiBartolo, G., Copeland, A., Lykidis, A., Trong, S., Nolan, M., Goltsman, E., Thiel, J., Malfatti, S., Loper, J.E., Lapidus, A., Detter, J.C., Land, M., Richardson, P.M., Kyrpides, N.C., Ivanova, N., and Lindow, S.E. 2005. Comparison of the complete genome sequences of *Pseudomonas syringae* pv. *syringae* B728a and pv. *tomato* DC3000. Proc. Natl. Acad. Sci. U S A 102:11064-11069.

Gesheva V, Stackebrandt E, Vasileva-Tonkova E. 2010. Biosurfactant Production by Halotolerant *Rhodococcus fascians* from Casey Station, Wilkes Land, Antarctica. Current microbiology. 61:112-117.

Grant, S.R., Fisher, E.J., Chang, J.H., Mole, B.M., and Dangl, J.L. 2006. Subterfuge and manipulation: Type III effector proteins of phytopathogenic bacteria. Ann. Rev. Microbiol. 60:425-449.

Ham, J.H., Kim, M.G., Lee, S.Y., and Mackey, D. 2007. Layered basal defenses underlie non-host resistance of Arabidopsis to *Pseudomonas syringae* pv. *phaseolicola*. Plant J. 51:604-616.

Hauck, P., Thilmony, R., and He, S.Y. 2003. A *Pseudomonas syringae* type III effector suppresses cell wall-based extracellular defense in susceptible Arabidopsis plants. Proc. Natl. Acad. Sci. U S A 100:8577-8582.

Joardar, V., Lindeberg, M., Jackson, R.W., Selengut, J., Dodson, R., Brinkac, L.M., Daugherty, S.C., Deboy, R., Durkin, A.S., Giglio, M.G., Madupu, R., Nelson, W.C., Rosovitz, M.J., Sullivan, S., Crabtree, J., Creasy, T., Davidsen, T., Haft, D.H., Zafar, N., Zhou, L., Halpin, R., Holley, T., Khouri, H., Feldblyum, T., White, O., Fraser, C.M., Chatterjee, A.K., Cartinhour, S., Schneider, D.J., Mansfield, J., Collmer, A., and Buell, C.R. 2005. Whole-genome sequence analysis of *Pseudomonas syringae* pv. *phaseolicola* 1448A reveals divergence among pathovars in genes involved in virulence and transposition. J. Bacteriol. 187:6488-6498.

Jones, J.D., and Dangl, J.L. 2006. The plant immune system. Nature 444:323-329.

Kim, M.G., and Mackey, D. 2008. Measuring cell-wall-based defenses and their effect on bacterial growth in Arabidopsis. Methods Mol. Biol. 415:443-452.

La Duc MT, Dekas A, Osman S, Moissl C, Newcombe D, Venkateswaran K. 2007. Isolation and characterization of bacteria capable of tolerating the extreme conditions of clean room environments. Applied and Environmental Microbiology. 73:2600-11.

Letek, M., Gonzalez, P., MacArthur, I., Rodriguez, H., freeman, T., Valero-Rello, A., Blanco, M., Buckley, T., Cherevach, I., Fahey, R., Hapeshi, A., Jolyon, H., Leadon, D., Navas, J., Ocampo. A., Quail, M., Sanders, M., Scortti, M., Prescott, J., Fogarty, U., Meijer, W., Parkill, J., Bentley, S., and Vazquez-Boland, J. 2010. The Genome of a Pathogenic *Rhodococcus*: cooptive Virulence Underpinned by Key Gene Acquisitions. Public Library of Science Genetics 6:1-17.

Lindenberg, M., Myers, C., Collmer, A., and Schneider D. 2008. Roadmap to New Virulence Determinants in *Pseudomonas syringae*: Insights from Comparative Genomics and Genome Organization. Molecular Plant-Microbe Interations. 21:685-700.

Lindgren, P.B., Peet, R.C., and Panopoulos, N.J. 1986. Gene cluster of *Pseudomonas syringae* pv. "*phaseolicola*" controls pathogenicity of bean plants and hypersensitivity of nonhost plants. J. Bacteriol. 168:512-522.

MacLean, D., Jones, J., and Studholme, D. 2009. Application of 'next-generation' sequencing technologies to microbial genetics. Nature Reviews. 7: 287-296.

Martínková, L., Uhnákováa, B., Páteka, M., Nešveraa, J., Křena, V. 2009. Biodegradation potential of the genus *Rhodococcus*. Environ Int. 35:162-77.

Niepold, F., Anderson, D., and Mills, D. 1985. Cloning determinants of pathogenesis from *Pseudomonas syringae* pathovar *syringae*. Proc. Natl. Acad. Sci. USA 82:406-410.

Nomura, K., Debroy, S., Lee, Y.H., Pumplin, N., Jones, J., and He, S.Y. 2006. A bacterial virulence protein suppresses host innate immunity to cause plant disease. Science 313:220-223.

Phillippy, A., Schatz, M., and Pop, M. 2008. Genome assembly forensics: finding the elusive mis-assembly. Genome Biology. 9:R55.

Putnam, M., and Miller, M. 2007. *Rhodoccocus fascians* in Herbaceous Perennials. Plant Disease. 91:1064-1076.

Reinhardt, J.A., Baltrus, D.A., Nishimura, M.T., Jeck, W.R., Jones, C.D., and Dangl, J.L. 2009. *De novo* assembly using low-coverage short read sequence data from the rice pathogen *Pseudomonas syringae* pv. *oryzae*. Genome Res. 19:294-305.

Roine, E., Wei, W., Yuan, J., Nurmiaho-Lassila, E.L., Kalkkinen, N., Romantschuk, M., and He, S.Y. 1997. Hrp pilus: an hrp-dependent bacterial surface appendage produced by *Pseudomonas syringae* pv. *tomato* DC3000. Proc. Natl. Acad. Sci. U S A 94:3459-3464.

Salzberg, S., and Yorke, J. 2005. Beware of mis-assembled genomes. Bioinformatics. 21:4320-4321.

Schechter, L.M., Vencato, M., Jordan, K.L., Schneider, S.E., Schneider, D.J., and Collmer, A. 2006. Multiple approaches to a complete inventory of *Pseudomonas syringae* pv. *tomato* DC3000 type III secretion system effector proteins. Mol. Plant-Microbe Interact. 19:1180-1192.

Stes, E., Vandeputte, O., El Jaziri, M., Holsters, M., and Vereecke, D. 2011. A Successful Bacterial Coup D'état: How *Rhodococcus fascians* Redirects Plant Development. Annual Review of Phytopathology. 49:1-18.

Studholme, D.J., Ibanez, S.G., MacLean, D., Dangl, J.L., Chang, J.H., and Rathjen, J.P. 2009. A draft genome sequence and functional screen reveals the repertoire of type III secreted proteins of *Pseudomonas syringae* pathovar *tabaci* 11528. BMC Genomics 10:395.

Thilmony, R., Underwood, W., and He, S.Y. 2006. Genome-wide transcriptional analysis of the *Arabidopsis thaliana* interaction with the plant pathogen *Pseudomonas syringae* pv. *tomato* DC3000 and the human pathogen *Escherichia coli* O157:H7. Plant J. 46:34-53.

Thomas, W.J., Thireault, C.A., Kimbrel, J.A., and Chang, J.H. 2009. Recombineering and stable integration of the *Pseudomonas syringae* pv. *syringae* 61 *hrp/hrc* cluster into the genome of the soil bacterium *Pseudomonas fluorescens* Pf0-1. Plant J. 60:919-928.

Zipfel, C. 2008. Pattern-recognition receptors in plant innate immunity. Current Opinion in Immunology 20:10-16.

Zipfel, C. 2009. Early molecular events in PAMP-triggered immunity. Curr. Opin. Plant Biol. 12:414-420.

APPENDIX

MATE PAIR FILTERING SCRIPT

```
#!/usr/bin/perl

use warnings;
use strict;

my $BIN_SIZE = 50;

my $bins = {};

my $non_number_bins = {};
$non_number_bins->{"overlap"} = 0;
$non_number_bins->{"ok"} = 0;

my $cashx_file = $ARGV[0];
chomp($cashx_file);

open (CASHX, $cashx_file) || die("Could not open file!");

print "Processing cashx!\n";

while (my $first_read = <CASHX>) {

        chomp $first_read;

        my $first_hits = -1;

        if ($first_read =~ m/S=0/) {

                $first_hits = 0;
        }


        $first_read =~ m/^ID=(.*)[\s\t]+/;

        #$first_read =~ m/^ID=(.*)\tHITS=.*$/;
        my $first_id = $1;
        my $first_node = "-1";
        my $first_start = -1;
        my $first_stop = -1;
        my $first_strand = 0;

        #change this to accommodate more than one hit...

        my @first_nodes = ();

        if ($first_hits != 0) {
```

```
                my $accession_present = 1;

                while ($accession_present == 1) {

                        my $accession = <CASHX>;

                        $accession =~ m/^Accession=(.*)[\t\s]+Start=.*$/;
                        $first_node = $1;

                        $accession =~ m/Start=(.*)[\t\s]+End.*$/;
                        $first_start = $1;

                        $accession =~ m/End=(.*)[\t\s]+Strand.*$/;
                        $first_stop = $1;

                        $accession =~ m/Strand=(.*)$/;
                        $first_strand = $1;

                        my @hit_info = ($first_node, $first_start,
$first_stop);
                        push (@first_nodes, \@hit_info);

                        my $pos = tell CASHX;
                        my $next_line = <CASHX>;
                        seek CASHX, $pos, 0;

                        if (!($next_line =~ m/Accession/)) {

                                $accession_present = 0;

                        }


                }

        }

        my $second_read = <CASHX>;

        my $second_hits = -1;

        if ($second_read =~ m/S=0/) {

                $second_hits = 0;
        }

        $second_read =~ m/^ID=(.*)[\t\s]+HITS=.*$/;
        my $second_id = $1;

        my $second_node = "-1";
        my $second_start = -1;
        my $second_stop = -1;
        my $second_strand = 0;

        my @second_nodes = ();
```

```
	if ($second_hits != 0) {

		my $accession_present = 1;

		while ($accession_present == 1) {

			my $accession = <CASHX>;
			$accession =~ m/^Accession=(.*)[\t\s]+Start=.*$/;
			$second_node = $1;

			$accession =~ m/Start=(.*)[\t\s]+End.*$/;
			$second_start = $1;

			$accession =~ m/End=(.*)[\t\s]+Strand.*$/;
			$second_stop = $1;

			$accession =~ m/Strand=(.*)$/;
			$second_strand = $1;

			my @hit_info = ($second_node, $second_start,
$second_stop);
			push (@second_nodes, \@hit_info);

			my $pos = tell CASHX;
			my $next_line = <CASHX>;
			seek CASHX, $pos, 0;

			if (!($next_line =~ m/Accession/)) {

				$accession_present = 0;

			}

		}
	}

	my $smallest_distance = -1;
	my $overlap_present = -1;

	foreach my $first_info (@first_nodes) {

		my @first_array = @$first_info;

		my $first_node = $first_array[0];
		my $first_start = $first_array[1];
		my $first_stop = $first_array[2];

		foreach my $second_info (@second_nodes) {

			my @second_array = @$second_info;

			my $second_node = $second_array[0];
			my $second_start = $second_array[1];
			my $second_stop = $second_array[2];

			if ($first_node eq $second_node) {
```

```
                                my $lower_start;
                                my $lower_stop;
                                my $higher_start;
                                my $higher_stop;

                                if ($first_start < $second_start) {

                                        $lower_start = $first_start;
                                        $lower_stop = $first_stop;
                                        $higher_start = $second_start;
                                        $higher_stop = $second_stop;

                                } else {
                                        $lower_start = $second_start;
                                        $lower_stop = $second_stop;
                                        $higher_start = $first_start;
                                        $higher_stop = $first_stop;
                                }

                                my $difference = $higher_start -
$lower_stop;
                                my $overlap_exists = 1;

                                if ($lower_stop < $higher_start) {

                                        $overlap_exists = 0;

                                }

                                if ($overlap_exists == 1) {

                                        $overlap_present = 1;

                                } else {

                                        if ($smallest_distance == -1) {
                                                $smallest_distance =
$difference;

                                        } elsif ($difference <
$smallest_distance) {

                                                $smallest_distance =
$difference;

                                        }
                                        my $bin_number = $difference /
$BIN_SIZE;
                                        $bin_number = int($bin_number);
                                        if (!defined($bins-
>{$bin_number})) {

                                                $bins->{$bin_number} = 0;
                                        }

                                        $bins->{$bin_number} = $bins-
>{$bin_number} + 1;
```

```
                                }

                        }
                }

        }

        if ($overlap_present == 1) {

                $non_number_bins->{"overlap"}++;


        } elsif ($smallest_distance != -1) {

                my $bin_number = $smallest_distance / $BIN_SIZE;
                $bin_number = int($bin_number);
                if (!defined($bins->{$bin_number})) {

                        $bins->{$bin_number} = 0;
                }

                $bins->{$bin_number} = $bins->{$bin_number} + 1;


        } else {

                $non_number_bins->{"ok"}++;

        }

}

print "\tDone processing cashx!\n";


open (BIN_OUT, ">bins.txt");

print BIN_OUT "overlap\t".$non_number_bins->{"overlap"}."\n";
print BIN_OUT "ok\t".$non_number_bins->{"ok"}."\n";

my @keys = sort {$a <=> $b} keys % {$bins };

foreach my $key (@keys) {

        my $upper_value = $key * 50 + 50;
        my $lower_value = $key * 50;
        print BIN_OUT "$lower_value - $upper_value bp\t".$bins-
>{$key}."\n";

}

close (BIN_OUT);

print "Done with binning.\n";
```