GENETIC MECHANISMS OF OZONE TOLERANCE IN SOYBEAN AND METHODS OF EVOLUTIONARY DISTANCE

by

Adam Michael Whaley

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Bioinformatics and Computational Biology

Charlotte

2015

Approved by:

_____
Dr. Jessica Schlueter

_____
Dr. Dan Janies

_____
Dr. Xinghua Shi

_____
Dr. Cory Brouwer

_____
Dr. John Diemer

ABSTRACT

ADAM MICHAEL WHALEY. Genetic mechanisms of ozone tolerance in soybean and methods of evolutionary distance. (Under the direction of DR. JESSICA SCHLUETER)

At ground level, ozone is a pollutant that poses serious risk to both human health and crop yields. In addition to being a vital economic crop in the US and abroad, conventional soybean lines are susceptible to damage caused by ozone exposure. In order to better understand the genetic differences between existing soybean lines which show extreme differences in sensitivity to ozone, RNA-seq and qRT-PCR based transcriptome analyses were performed. In utilizing these techniques, this study has shown that despite being the same plant, each cultivar exhibits a unique response in both amount and timing of gene expression. These unique responses, in addition to identifying the genes responsibility for ozone tolerance, hint at a secondary, physiological reason for ozone tolerance.

Evolutionary distances are measures based on the divergence of two sequences and are used to characterize the time since divergence and selective pressures acting on a pair of duplicated genes. Despite being a useful method of characterizing duplicate genes, the applications for applying these methods have largely gone unchanged in recent years and the community which uses them is underserved. To this end, this study has created two additional analysis pipelines, one which recreates existing methodologies and another which applies a new method to better characterize localized selective forces within subsets of a gene pair. Additionally, a complex and biologically relevant sequence simulation program has been created that, in addition to parameter tuning, has identified overestimates in the measures of the evolutionary distance. This provides a framework

for future additions to the pipeline which will automate parameter tuning for more accurate estimates.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Soybeans

Soybean belongs to a clade of plants known as legumes. Legumes such as soybean possess a number of traits that have made them valuable as agricultural crops throughout human history. Legumes are characterized by their unique root nodule structures. Within these structures, a symbiotic relationship with nitrogen fixing Rhizobia bacteria allows the plant to supply its own nitrogen. In the context of growth for agricultural purposes, this symbiotic relationship with the Rhizobia bacteria precludes the need for expensive and possibly environmentally damaging fertilizers (Singh). While legumes are grown for a number of purposes, most of these legumes belong to the Papilionoideae subfamily and are grown for their fruits, which tend to be high in proteins and oils. These proteins and oils lend themselves to use in a large number of industries including human and animal consumption, industrial oils and solvents, and cosmetics. In light of these favorable traits, it is unsurprising that legumes are one of the most commonly grown crop families worldwide. In developing nations, legumes such as pigeonpea can serve as a cheap, vital source of protein (Varshney et al.). In the United States, soybean is the second most-produced crop behind corn. It is a summer crop, typically planted in May, with approximately 85 million acres of soybeans being planted each year in the US; additionally, “biotech” varieties of soybean make up anywhere from 90-99% of soybeans

grown, varying by state. Soybean crops, due to their nitrogen fixing ability, are often used in the US in rotation with other crops such as corn to help replenish soil nitrogen.

In addition to its importance to agriculture, one of the more interesting aspects of the soybean is its large and complex genome. Soybeans have a paleotetraploid genome that has gone through 2 whole genome duplication events, each followed by a return to a diploid state (Schmutz et al.; Roulin et al.). These duplicate and reduction events have left soybeans with a larger gene content when compared to other legumes as well as significant retention of duplicated blocks within the genome. Approximately 75% of the genes in the soybean genome have at least one copy, with some gene families consisting of six gene duplicates (Roulin et al.). This has a profound effect on gene expression and makes soybean an ideal candidate for examining the proposed models for the fates of duplicate genes.

The wealth of knowledge available in soybean research is due to the availability of a complete soybean genome. Prior to its publication in 2010, the closest available genome that served as a model was A. thaliana. In contrast to the ~1Gb soybean genome, A. thaliana has a 120Mb genome with very little duplication. The availability of a reference genome has led to a wide number of resources readily accessible to researchers. In addition to the published genome, which has undergone several revisions since its initial release and are available through JGI (http://jgi.doe.gov), there are also a number of knowledge base websites dedicated to hosting soybean genome data and analysis tools for researchers. Foremost amongst these are SoyKB and SoyBase, which in addition to genome browsers and genetic maps, also provide tools such as GO-term enrichment analysis, BLAST analyses, and microarray probe ID mapping (Joshi et al.; Grant et al.).

The availability of a reference genome has also enabled the study of the effects of gene duplication on expression, which in turn lends support for the models of fates of duplicate genes. A 2013 study by Roulin et al. used RNA-seq to investigate neofunctionalization and subfunctionalization within the soybean genome following a whole genome duplication event. Roulin et al. identified 18,000 duplicated gene pairs and found that near half had differential expression between paralogs, and showed that different tissues selectively expressed a different gene in each pair in nearly all cases. By examining the synonymous (a change in a nucleotide which does not alter the codon) and non-synonymous (a change in a nucleotide which alters the codon) distances between pairs, Roulin et al. were able to find evidence for gene retention through subfunctionalization. A previous study in cotton found similar results, but this study was limited by the lack of a reference genome at the time and as such far fewer gene pairs could be investigated (Chaudhary et al.). Additionally, a 2014 study by Shen et al. found that the frequency of alternative splicing in duplicate genes is negatively correlated with the size of a gene family (Shen et al.). Larger gene families within the study also tended to have shorter introns, less exons, and lower levels of expression. Shen et al. were also able to show that over half of the duplicate genes they studied had different patterns of alternative splicing when both pairs were expressed.

1.2 Methods of RNA-seq Analysis

RNA-seq is a relatively new technique that utilizes current generation sequencing techniques to quantify transcript abundance. Where cost would normally be a significant barrier to entry, sequencing costs have reduced dramatically and the ability to multiplex, or run multiple samples in the same sequencing lane, have dramatically reduced the cost

to sequence which in turn has led to a significant increase in the amount of sequencing work done in labs around the world. This has been especially advantageous in less studied species with smaller research communities, where the cost would have too prohibitive to generate genome and transcriptome sequences. With this proliferation of sequence availability comes an increased need for diligence in selecting the appropriate algorithms for analyzing the data. Obviously the consequences of improperly handling such a vast amount sequencing data would be severe, and this task falls solely on bioinformaticians.

As is common with any new technology, there has been a large amount of recent investigations into the accuracy and efficacy of RNA-seq in comparison to other sequencing technologies. This scrutiny into biases and errors in sequences are not only important for the validation of projects using this technology, but for the refinement of procedures for generating libraries and the algorithms used to characterize the results. A study by Maroni et al. investigated the strengths, weakness, and differences between current sequencing technologies, specifically comparing the results of RNA-seq to the then more commonly used microarray expression arrays (Marioni et al.). Comparing the two technologies, Maroni found a .73 correlation between RNA-seq and microarray differential expression (DE) calls, with a large amount of that difference attributed to true positives that were only detected in the RNA-seq study. Maroni also found limited evidence of lane effects, differences in read counts attributable to the lane on the sequencing chip the sample was run in and not actual differences in expression, in the RNA-seq data. One lane of RNA-seq was able to detect 70% of the DE genes found in 3 microarrays, and also showed that additional lanes of RNA-seq only added roughly 7%

additional data. This indicates a sharp dropoff in the effectiveness in adding lanes to increase data for the cost of sequencing. This latter finding was backed by McGettin et al., who found only small differences between the technical replicates of RNA-seq samples (McGettigan). McGettin also point out that while there was a bias in the results introduced during the PCR-amplification step of RNA-seq library preparations, the current generation of strand-specific paired end RNA-seq libraries effectively eliminate this bias. One major, currently standing problem with RNA-seq is the inability to effectively quantify expression of rare transcripts and isoforms. While RNA-seq was instrumental in demonstrating the importance of alternative gene isoforms in cell differentiation, the inability of RNA-seq to quantify and qualify rare transcripts and low expression genes is a product of both limitations of the technology and limitations of quantification methods. This problem has only been highlighted by more recent technologies such as PacBio sequencing, which can capture and sequence full transcripts (Quail et al.). Because there is no reconstruction of transcripts based on smaller reads, this technology is not limited to statistically informed isoform reconstructions and abundances, but absolute abundances and qualification.

Simply put, RNA-seq works by sequencing the short fragments of a cDNA library of the mRNA transcripts from a sample. The resulting fragments vary in length and count depending on the experiment and sequencer used, but are usually 100-200bp in length. As a consequence of this process, there is a bias towards an under-representation of sequences from smaller transcripts (Oshlack, Wakefield, and others). The first major challenge in analyzing these sequences is properly attributing each read to the gene sequence that it was generated from. While several de novo transcript assembly methods

are available in the event a reference genome is unavailable, these methods are far more computationally taxing. Algorithms that use a reference genome tend to use short sequence mapping algorithms such as the Burrows-Wheeler alignment algorithm used in Bowtie (Langmead et al.).

There are three major computation considerations for algorithms in this stage. Foremost among these is mapping reads across splice junctions. There are two strategies used by algorithms during this step. The exon first strategy, where reads which map unambiguously to full exons are mapped first, and then can be eliminated from attempts to map across splice junctions. Remaining reads are then broken into smaller fragments and aligned to exons. This strategy is less computationally intensive but less accurate than the seed-extend method, where small fragments of reads are first mapped to the genome, then more sensitive methods are used to examine this candidate region (Garber et al.). The other two computational considerations are problems in mapping by paralogous gene families and pseudogenes. Paralogous genes and pseudogenes create scenarios where reads will fail to map to only a single location, and create problems downstream when read counts are used to quantify gene expression: this phenomenon is called transcript shadowing. While it may be tempting to simply exclude any gene that fails to map to only a single location, in the case of gene families these reads may be relevant information especially in a highly duplicated genome. As such, careful consideration must be taken when selecting software and parameter sets.

After the reads are assigned to genes, they are used as a measure or strength of gene expression. This creates two normalization problems as the number of reads generated per sequencing run will not be consistent, and longer transcripts will inherently generate

more reads than a shorter sequence would, even at the same abundance. Both of these sources of variability affect the raw count of reads mapped to a gene in ways that would not be consistent across samples or replicates. The most common method of dealing with this variability is the assignment of an RPKM value, or reads per kilobase per million reads. In this strategy, the read count of a gene is normalized by the length of an mRNA and by the number of mapped reads in the sequencing run. RPKM normalization functions best when read coverage is uniform across a gene, a standard which is not always possible depending on the library preparation method (Pepke, Wold, and Mortazavi). While this method is common to most RNA-seq analysis software, there are subtle differences in the calculation, as well as the method by which RPKM values are assigned across duplicate genes. Variability in RPKM estimates across samples necessitates further normalization before samples can be compared for differential expression.

A comparison of software packages for RNA-seq and their methods for addressing the computational problems posed by RNA-seq analysis can be found in Figure 1.

1.3 Fates of Duplicate Genes

Eighty years ago the first models for the fate of duplicate genes were introduced. In 1933 Haldane proposed that should a gene be duplicated, one copy would begin to accumulate mutations such that it would lose function while the other copy retained the original gene function (Haldane). This would become the basis for all models to follow. In 1970, the concept that would eventually be defined as neofunctionalization was introduced by Ohno (Ohno and others). Under his model, one member of the copied gene

pair would maintain normal function, freeing the other copy to begin accumulating mutations. While most of these mutations would be deleterious, Ohno proposed that some of these mutations might lend the gene novel functionality, and both copies would be kept. He hypothesized that duplicated genes would provide a more likely basis for the creation of new genes than a de novo process. While much of Ohno's work is historically significant, the revelation of gene duplication as the provider of novel genetic function is perhaps the most lasting.

In 1983 Kimura introduced the concept of relaxed selective pressure (Kimura, *The Neutral Theory of Molecular Evolution*). Under this model, purifying selection is relaxed after a duplication because of the redundant extra copy; this allows for the accumulation of mutations that would normally be selectively negative. Like Ohno's model, most mutations would eventually cause one copy to become a pseudogene but positive mutations could lead to fixation of both copies with one having a new or modified function.

As sequencing technologies in the last two decades have become more practical and cheaper, the amount of available genomic data increased allowing for scientists to build upon these foundational models by studying gene duplication at a genomic scale instead of solely at a single gene or theoretical level. These studies provided the foundation for the field of comparative genomics. This had led to many advancements and refinements of the classical models.

One of the first challenges to the classical models was the findings by groups such as Walsh, Nadeua and Sankoff, and Wagner et al (Walsh; Nadeau and Sankoff; Wagner). In his 1995 Genetics publication, Walsh showed that for large populations the probability of

a duplicated gene developing a novel function and becoming fixed in the population is higher than that of a pseudogene becoming fixed. This was supported by the findings of Nadeau and Sankoff who, using human and mouse gene families, showed that there is lower rate of gene loss than would be predicted by the classical models. They found that roughly half of gene duplications led to novel gene functions. These findings led to the development of the duplication-degeneration-complementation (DDC) model. The key distinction of this model was the addition of a third possible outcome following a duplication event: subfunctionalization. In addition to loss of function and neofunctionalization outcomes, subfunctionalization describes an outcome where degenerative mutations in both copies lead to the fixation of both alleles as both are required to maintain original function. Evidence for subfunctionalization has come from a number of sources and systems including zebrafish, maize, arabidopsis, cotton, and soybean (Liu et al.; Kleinjan et al.; Sheehan et al.; Duarte et al.; Chaudhary et al.).

In 2000 Lynch and Conery published a study estimating the rates of both duplication events and loss of function using genomic data available at the time from nine different species (Lynch and Conery). They found that most genes undergo a period of relaxed selection or increased evolution immediately after duplication. They also found a high rate of duplication events, on average .01 per gene per million years. While this rate is remarkably high, their estimated rate of gene loss is similarly high. They predicted a half-life for a gene duplicate to be only 4 million years. This contrasts with the high rate of duplicate preservation seen in studies such Nadeau and Sankoff. However, Lynch and Conery suggest that their rate of duplicate loss can be reconciled with the high retention

of duplicates found in other studies when considering the special circumstances surrounding whole genome duplications.

A recent review by Innan and Kondrashov categorized and described four categories of models duplicate gene fates (Innan and Kondrashov). The first category is models that describe a scenario where the selection on the new copy is neutral; this includes several classic models including Ohno's neofunctionalization and the DDC model. The second category describes a scenario where the duplication event has a beneficial effect on fitness through mechanisms such as dosage compensation or or protection from deleterious mutations. The third category describes the duplication of genes with genetic variation within the population where fixation occurs due to the positive effects of the genetic variability. Finally, the fourth category describes the fates of genes duplicated by events such as whole genome duplications, where dosage balances of protein complexes play important roles in fixation.

Modern genomic sequencing techniques have led to the proliferation of genomic data for countless species, and in turn, has provided an excellent resource for examining the various models of duplicated gene fates. In 2004, Adams et al. used expression arrays to show in cotton that duplicated genes showed different patterns of expression immediately following duplication, patterns which differed by tissue (Adams, Percifield, and Wendel). Similar results were found in soybean by Roulin et al., who found not only differing patterns of expression in duplicated genes by tissue, but also that duplicated genes in collinear blocks were coexpressed (Roulin et al.). Findings such as these have challenged base assumptions of the current models. Recently, Qian et al. proposed a new model of subfunctionalization (Qian et al.). This model, expression reduction, would

prevent the loss of daughter genes following duplication by hypothesizing that a high enough reduction in expression of both daughters would be such that, if one daughter gene were lost, the total expression by the remaining daughter would not be sufficient to retain ancestral function and would be deleterious. This provides a mechanism for maintaining long-term functional redundancy in duplicated genes

The goal of categorizing these models was to provide a structured framework from which focused questions of duplicate gene fate could be asked. However, it also underscores two major problems within the discussion: first, that many of these models are not formal models and are therefore limited, and secondly that many of these models apply to specific cases of duplication events and are dependent on many extraneous factors that are not always known. This limitation affects the ability to properly implement and test the predictions of these models in the absence of complete metadata. Categorizing specific duplications to their appropriate model is often reliant on functional information about not just the gene pairs but also knowledge of the ancestral gene. Determining these data in most organisms would be expensive and time consuming even for a limited set of genes, assuming an appropriate ancestral gene is known or available.

1.4 Ozone as a Pollutant

Human beings have a complicated relationship with ozone. When ozone is present in the stratosphere, it acts as a shield from UV radiation and is essential for human well-being. However, when ozone is present in the troposphere it acts as a pollutant. Tropospheric ozone comes from two main sources. First, wind patterns can draw stratospheric ozone down to ground level. Secondly, ozone can be formed at ground level through reactions with hydrocarbons, sunlight, and heat. These reactions

can be biotic, through naturally occurring volatile organic compounds (VOCs) and nitrogen oxides emitted by plants or abiotic reactions from emissions stemming from anthrogenic sources (Ebi and McGregor). Because the photochemical reactions which create ground level ozone are temperature sensitive, the levels of tropospheric ozone, especially in urban areas, is expected to rise in response to global warming (Aw and Kleeman; Seinfeld and Pandis).

Although most of the projections regarding ozone levels have the subsequent effects on human health have been to Europe and North America, they do not paint a hopeful picture. The WHO considers ozone concentrations in the 40-50ppb range to be safe (Vingarzan; Organization and others). The International Panel on Climate Change, IPCC, established a clean air standard in 2001 that listed acceptable ozone levels as below 80ppb. However, the air quality index (AQI) lists anything above 65ppb as "unsafe for sensitive individuals" (Bell et al.). Ozone levels regularly exceed 60ppb, especially in the summer months, with some areas in North America having yearly maximum ozone concentrations above 100ppb (Vingarzan).

The modeling of future ozone concentrations and the direct effects this will have on humans are complicated for a number of reasons. Chiefly amongst these is that the factors leading to tropospheric ozone creation are themselves so complex. Wind patterns, urbanization, emissions patterns, and temperature all play important and complex roles in ozone concentrations. Global warming especially, and what new emissions regulations may be imposed because of it, make the modeling of ozone difficult. Despite this, several sources have successfully made projections using a variety of modeling strategies, factors, and scopes (Aw and Kleeman; Ebi and McGregor; Bell et al.). Forkel and

Knoche examined the projections of ozone concentrations in Germany, and found a projected increase of 2-6ppb by 2030 (Forkel and Knoche). This increase would correspond to a 300% increase in the number of days ozone concentrations would exceed 90ppb. Localized studies in North America had similar findings: projections of the Los Angeles basin predicted a 24% increase in ozone concentration. Some of these projections also predict the effect this will have on human mortality; predictions by Knowlton et al projected a 4-5% increase in ozone relative fatalities in the New York metropolitan area; Bell et al projected a 5-10ppb increase in 50 major US cities by 2050, which they projected to lead to a .11%-.27% increase in total mortality due to ozone related COPD and other respiratory ailments (Bell et al.; Knowlton et al.). Likewise, Hwang et al. projected a .08 to .46 increase in daily mortality in Los Angeles and San Diego, while work by Anderson et al predicted a 10% increase in premature death due to ozone in the UK by 2020, increasing to 40% by 2080 (Hwang, Burer, and Bell; Anderson, Derwent, and Stedman). Finally, West et al. examined 10 world regions and predicted 500,000 additional deaths in the year 2030 due to changes in ozone concentrations (West, Szopa, and Hauglustaine). Patz attributes the large effects even small ozone increases can have directly on human health to the sheer number of individuals living in these urban areas of high ozone (Patz et al.).

In addition to the direct effects increasing ozone concentrations will have on human health and mortality, is it important to also consider the effect of ozone concentration on crop yield and human nutrition. The respiratory distress caused in plants by ozone exposure has been appreciated for a long time, in the 1980s ozone was posited as an explanation for forest decline (Pye; Reich; Blank). More recent studies

have expanded these findings, most notably the work by Hayes et al. which characterized the ozone tolerance of 83 species and organized the data into a database for meta-analyses (Hayes et al.) . Additionally, work by Wittig et al. found that current ambient ozone levels reduced tree biomass by 7%, and high ozone levels reduced biomass by 17%, in addition to affecting leaf and root growth (Wittig et al.). Ozone exposure, in addition to causing changes in biomass and growth, also causes leaf injury and negatively affects photosynthesis. These effects are not limited to trees, but are found in crop plants as well. Recent work by Singh et al. found that ozone exposure in soybeans negatively affected photosynthesis, as well as reducing stomatal conductance and overall capacity for carbon fixation; Singh et al. also suggest that differences in stomatal conductance, as a reaction meant to limit ozone exposure could account for difference in outcomes between soybean varieties (Singh, Tiwari, and Agrawal). Biomass reductions are, especially in the roots and leaves, are likely caused by this reduction in carbon-fixation, compounded by the need to produce antioxidant compounds (Ainsworth et al., "The Effects of Tropospheric Ozone on Net Primary Productivity and Implications for Climate Change"). In the United States alone, ozone is estimated to cause 3-5 billion dollars in crop loss (Fiscus, Booker, and Burkey).

1.5 Plant Defense Pathways and Their Relationship to Ozone

Plant pest resistance and stress pathways are complex genetic mechanisms that allow plants to resist and adapt to a wide variety of biotic and abiotic stresses. These pathways, depending on the type and severity of the stress can have many different outcomes, in the most severe cases leading to apoptosis to limit the spread of a pathogen. These pathways are controlled by a number of factors, including compounds

such as jasmonic acid, ethylene, salicylic acid. Jasmonic acid is a plant hormone derived from linoleic acid and is part of the jasmonate family of plant hormones. It plays a number of roles in plant growth and development, as well as stress resistance. Jasmonic acid levels are increased dramatically following an attack by an herbivore, which in turn activates the accumulation of anti-herbivory metabolites (Yang et al.). Salicylic acid is another plant hormone involved in both growth as well as defense; salicylic acid is a part of the systemic acquired resistance pathway whereby an attack on a specific part of a plant will increase resistances in other parts. Ethylene is another plant hormone, mostly commonly known for its role in ripening and senescence. In the context of plant pathogen and stress resistance, ethylene works alongside salicylic acid to regulate apoptosis (Rao, Lee, and Davis).

While stress response mechanisms are fairly ubiquitous, there are a number of varying responses when considering abiotic stress, ozone in particular. When ozone enters plant tissue it creates a burst of active oxygen species, which in turn activate signal transduction pathways for pathogen response including jasmonic acid, salicylic acid, and ethylene (Rao and Davis). Ethylene and salicylic acid, signaling molecules for oxidative stress, are negatively regulated by jasmonic acid and act as both a source of resistance and susceptibility to ozone by regulating plant cell death (Overmyer, Brosché, and Kangasjärvi). Rao et al. showed that Arabidopsis plants treated with ethylene inhibitors had reduced foliar damage, as well as plants treated with salicylic acid prior to ozone exposure had increased foliar damage. Conversely, Yoshida et al. showed that Arabidopsis mutants lacking ethylene signaling and salicylic acid production had increased foliar damage when compared to the wild type (Yoshida et al.). In aspen,

ethylene was shown to increase or decrease injury level depending on the severity of stress, indicating a complex relationship between ethylene, salicylic acid and oxidative stress (Vahala et al.).

Class	Category	Package	Notes	Uses	Input
Read mapping					
Unspliced aligners[a]	Seed methods	Short-read mapping package (SHRiMP)[41]	Smith-Waterman extension	Aligning reads to a reference transcriptome	Reads and reference transcriptome
		Stampy[39]	Probabilistic model		
	Burrows-Wheeler transform methods	Bowtie[43]			
		BWA[44]	Incorporates quality scores		
Spliced aligners	Exon-first methods	MapSplice[52]	Works with multiple unspliced aligners	Aligning reads to a reference genome. Allows for the identification of novel splice junctions	Reads and reference genome
		SpliceMap[50]			
		TopHat[51]	Uses Bowtie alignments		
	Seed-extend methods	GSNAP[53]	Can use SNP databases		
		QPALMA[54]	Smith-Waterman for large gaps		
Transcriptome reconstruction					
Genome-guided reconstruction	Exon identification	G.Mor.Se	Assembles exons	Identifying novel transcripts using a known reference genome	Alignments to reference genome
	Genome-guided assembly	Scripture[28]	Reports all isoforms		
		Cufflinks[29]	Reports a minimal set of isoforms		
Genome-independent reconstruction	Genome-independent assembly	Velvet[61]	Reports all isoforms	Identifying novel genes and transcript isoforms without a known reference genome	Reads
		TransABySS[56]			
Expression quantification					
Expression quantification	Gene quantification	Alexa-seq[47]	Quantifies using differentially included exons	Quantifying gene expression	Reads and transcript models
		Enhanced read analysis of gene expression (ERANGE)[20]	Quantifies using union of exons		
		Normalization by expected uniquely mappable area (NEUMA)[82]	Quantifies using unique reads		
	Isoform quantification	Cufflinks[29]	Maximum likelihood estimation of relative isoform expression	Quantifying transcript isoform expression levels	Read alignments to isoforms
		MISO[33]			
		RNA-seq by expectation maximization (RSEM)[69]			
Differential expression		Cuffdiff[29]	Uses isoform levels in analysis	Identifying differentially expressed genes or transcript isoforms	Read alignments and transcript models
		DegSeq[79]	Uses a normal distribution		
		EdgeR[77]			
		Differential Expression analysis of count data (DESeq)[78]			
		Myrna[75]	Cloud-based permutation method		

Figure 1. A table adapted from Garber et al. that explains the characteristics of a number of popular RNA-seq analysis programs.

CHAPTER 2: OZONE RESPONSE IN TOLERANT AND SENSITIVE SOYBEAN

2.1 Introduction

Ozone pollution is often perceived as an urban issue, but the problem is, in reality, regional and includes many agricultural areas. This problem occurs in rural areas because nitrogen oxides and hydrocarbons produced by electric power generation or urban sources such as automobiles move as plumes into rural areas. Those precursors then react with oxygen in the air and sunlight to form ozone (Prather et al.). The impact of ozone on human health receives significant public attention, but many plants are also sensitive to this toxic air pollutant (EPA; Ainsworth et al., "The Effects of Tropospheric Ozone on Net Primary Productivity and Implications for Climate Change*"; Mills et al.).

As an economically important crop, soybean suffers from several abiotic stresses that severely affect its production and quality in most countries (Purcell, L.C. and Specht, J.S.; Booker et al.). However, studies to identify genetic materials that are tolerant to abiotic stress, and specifically ozone stress, have been limited in soybean. Fortunately, recent studies have identified a small group of soybean genotypes from a breeding program in Fiskeby, Sweden, that are tolerant to several of the environmental stresses to which they have been challenged (Burkey and Carter Jr.). These closely related soybean genotypes are all tolerant or partially tolerant to drought, iron deficiency chlorosis, toxic soil aluminum, salt, cold, and atmospheric ozone pollution. Burkey and Carter (2009)

found that these Fiskeby varieties were significantly more tolerant to ozone stress than many other soybean varieties (Burkey and Carter Jr.). Ozone has been shown to reduce growth and yield in many crops, and soybean is known to be an ozone sensitive species that experiences stippling and necrosis of the leaves under elevated ozone conditions (Mauzerall and Wang; Wang and Mauzerall; Keen and Taylor; Heagle, Miller, and Pursley; Bou Jaoudé et al.). Leaf tissue damage is the combined result of activated oxygen species and an ethylene stress response that is interpreted as a pathogen invasion and leads to cell death as a means to limit the spread of the perceived pathogen (Kettunen, Overmyer, and Kangasjärvi). Impacts on a specific farm in a specific year can be difficult to quantify due to variations in the frequency and intensity of ozone pollution during the growing season and from year to year. Modeling studies that combine measurement of ambient ozone concentration with empirically derived yield response curves suggest that current soybean yields are reduced up to 10% by ozone pollution (Tong et al.; Fishman et al.). Research at the SoyFACE project in Illinois, where soybeans are exposed in the field to elevated ozone levels, predicts an additional 20% yield reduction by the year 2050 if ambient ozone levels rise (Morgan et al.). These yield losses could be exacerbated due to the current urbanization trend.

Ozone exposure directly affects the leaves by entering through the stomata where it is converted to reactive oxygen species including hydrogen peroxide, superoxide anions and hydroxyl radicals (Kanofsky and Sima). These reactive oxygen molecules are responsible for inducing the oxidative stress response. Upon exposure to ozone, plants may employ several response mechanisms some of which are conserved among species. Following exposure, ozone has an effect on photosynthesis resulting in an increase in

carbon dioxide concentration that in turn reduces stomatal conductance (Reich, Schoettle, and Amundson). Once ozone has entered a leaf, much of the defense response is thought to focus on either preventing the formation of reactive oxygen signaling molecules or reducing the concentration of plant-derived reactive oxygen molecules once they form. Superoxide dismutase, peroxidase and glutathione in conjunction of ascorbic acid work to remove these reactive oxygen molecules from the cell (Kubo et al.; Conklin and Barth; Chen and Gallie). Expression studies in tobacco showed an increase in beta-1,3-glucanase, catalase and chitinase (Ernst et al.). Physiologically, as the stomates are closing there is a decrease in mRNA levels of both subunits of Rubisco in potato plants exposed to ozone (Reddy et al.). Runeckles and Chevone described increased respiration, decreased photosynthesis, peroxidation of the membrane lipids, reduced transpiration, closing of the stomates and increased senescence (Runeckles and Chevone). Ethylene and salicylic acid, signaling molecules for oxidative stress, are negatively regulated by jasmonic acid and act as both a source of resistance and susceptibility to ozone by regulating plant cell death (Overmyer, Brosché, and Kangasjärvi).

In soybean, much like pathogen response, ozone elicits a hypersensitive resistance response that leads to a heightened production of flavonoids; in turn, this increase in flavonoids is thought to be a key factor in toxicity in the leaves that appears in the form of stippling and necrosis (Keen and Taylor). Maccarrone et al. showed that in soybean, lipoxygenase (LOX) transcription is increased in response to ozone exposure (Maccarrone, Veldink, and Vliegenthart). More recent work by Galant et al. supported the importance of the redox pathways in response to ozone exposure (Galant et al.). Using a proteomics approach, Galant et al. also showed that oxidative stress alters the

thiol oxidative state of proteins and leads to increased expression of peroxidase, methionine, and glutathione (Galant et al.). Ozone stress QTLs have been identified in rice, but limited QTL studies to date have addressed this stress in soybeans (Frei, Tanaka, and Wissuwa). Although a study underway by Burton et al. has led to the discovery of several putative ozone QTLs, the genetic mechanisms for ozone tolerance in soybean are largely unknown.

Within soybean, an important clue to understanding and dissecting these stress responses is found in the unique germplasm, geography, and geology of Fiskeby, Sweden, the site of a soybean breeding program that produced varieties from the 1940's through ~1975. Fiskeby is near Stockholm, Sweden and is far north of other soybean growing areas of the world (60 degrees north latitude, approximately 18 hour day-length in July). Sven Holmberg, the breeder at Fiskeby, was motivated to produce extremely early maturing varieties adapted to the short growing season of his country.

The stress-resistant nature of the varieties released by the Swedish program, Fiskeby III (USDA Plant Introduction (PI) 438471) being an example, was not discovered until recently when genetic studies included screening the ancestral base of North American varieties (Fiskeby types being part of the ancestral base for Canada) for various traits (Burkey and Carter Jr.). Resistance to ozone may have been a by-product of Holmberg's selection of varieties adapted to the cool climate of Sweden. It has been postulated that thicker leaves associated with cold tolerant Fiskeby genotypes could contribute to ozone tolerance by mechanisms yet to be identified, however testing of this hypothesis is ongoing.

In order to identify the genetic components of ozone resistance, we conducted an

RNA-seq analysis of tolerant and sensitive soybean cultivars under both control (low ozone) and stressed (high ozone) conditions. We focused on early response to ozone stress as we hypothesize that this point is when gene expression in Fiskeby III supports tolerance mechanisms.

2.2 Materials and Methods

Mandarin (Ottawa), PI 548379, and Fiskeby III, PI 438471, soybean cultivars were grown in a greenhouse under charcoal-filtered air conditions (<10 ppb ozone) for 25 days in October-November 2010 in the USDA-ARS greenhouse facilities in Raleigh, NC, USA. Plants were moved into continuously circulated tank reactors (CSTRs) in an adjacent greenhouse bay for a 3-day acclimation period prior to ozone exposure. CSTRs are cylindrical exposure chambers covered with Teflon film, designed for rapid mixing of gases (Rogers et al.; Heck and Philbeck). Ozone exposures began on the 28^{th} day following planting using eight CSTRs divided into four experimental blocks, each block consisting of two CSTRs designated as either a low (25 ppb target) or high (75 ppb target) ozone treatment. Each CSTR contained one plant of each cultivar variety. Actual ozone concentrations measured during exposure were 27±2 ppb and 74±1 ppb in the low and high treatments, respectively. CSTR temperature, light level, and relative humidity averaged 34±1 ºC, 254±13 μmol m^{-2} s^{-1} PAR, and 61±1 %, respectively, during the exposure period.

For physiological measurements, plants were exposed to ozone as described above for 2 days prior to assessment. Leaf foliar injury was determined by estimating the percentage of adaxial leaf surface area exhibiting necrotic or chlorotic lesions and/or pigmented stipple on selected leaves (Burkey and Carter Jr.). Leaf air space was

determined gravimetrically by weighing freshly harvested leaves before and after infiltration with water and reporting the difference in weight as a surrogate for the volume of air space within the leaf. Leaf specific weight was determined by removing six disks from each leaf using a cork borer of known area, drying the disks, and measuring the dry weight.

The fifth main stem trifoliate leaf (three leaves from each leaf stem) from the bottom of the plant was harvested from each genotype in one experimental block at each of 4 distinct time points from five to seven hours after exposure began at 0900, samples taken at 1400 and continued to be taken in 40 minute intervals. Leaf tissue was flash frozen in liquid nitrogen prior to transportation back to UNC-Charlotte on dry ice. Whole leaf tissue from each of the three leaflets from the 5^{th} trifoliate leafs was pooled and extracted using the Qiagen Plant RNAEasy extraction kit (Qiagen, CA) followed by Ambion DNA-free treatment (Life Technologies, NY). RNA-seq libraries were subsequently created using Illumina TruSeq RNA-seq kit (Illumina, CA). Each library was quantified on the Bioanalyzer using RNA nano chips (Agilent, CA) prior to submission for sequencing at the David H. Murdock Research Institute Core Facility (Kannapolis, NC). Each library was run on a separate lane of a HiSeq2000, generating 100 base pair single end reads.

RNA-seq quality figures and statistics were produced by RSeQc, a package for assessing read and mapping quality, using the *read_duplication* script (Wang, Wang, and Li) (Figure 6). The results of sequencing were mapped back to the Williams 82 soybean reference genome *Glycine max* Wm82.a2.v1 using Tophat (Trapnell, Pachter, and Salzberg). Mapping was optimized for non-mammalian data sets as per program

instructions by changing the maximum allowed intron size to 5Kb. The GFF file from the Williams82 reference genome was also provided in the mapping step. Differential gene and isoform analysis of the mapping results was determined using the Cufflinks *cuffdiff* program using the -u and -b options for individual time points, as well as across all four time points using the –T option. Differential expression was assessed by the *cuffdiff* program and corresponds to those genes returning a $p<.05$ as determined by *cuffdiff*. Analysis of the timing of differential gene expression across the four time points was accomplished using custom Python scripts and statistical measures from *cuffdiff* (See Appendix A.1.2). Functional analysis of expressed genes was determined using associated gene ontology (GO) terms for soybean gene models available on Soybase (www.soybase.org) and custom Python scripts (See Appendix A.1.2 and A.1.3) (Grant et al.). Single enrichment analysis (SEA), a function of AgriGO was used to examine GO term enrichment in sample queries; results were confirmed using the Gene Ontologies distribution tool available at Soybase.org (Grant et al.; Du et al.). SEA results return abundant GO term categories found in the sample set based on an adjusted p-value<.05. CummeRbund, an extension of the cufflinks package for data visualization, was used for visualization of results and read dispersion (Goff, Trapnell, and Kelley) (Figure 5). Soybean orthologs of genes used in the Le Provost et al. study were identified using BLASTp (Altschul et al.).

For PCR verification of the RNA-seq results, primers for an anti-oxidant gene Glyma.02g141800 (Glutathione), a photosynthesis gene Glyma.13g028200, and a 60S reference Glyma.13g318800 were designed with a T_m of ~60 degrees C with 40-60% GC content and 50-200bp product size. Primers and targets can be found in Table 7. Growth

conditions and extraction protocols were identical to the RNA-seq conditions. Tissue from the 4th trifoliate leaf was take and used to generate cDNA libraries using a poly-A procedure. Libraries were arranged on an 11-plate pPCR design such that each sample was arrayed in triplicate as well as with the 60S control in triplicate on each plate. qRT-PCR was performed on an ABI Systems 7500 using SYBR Select Master Mix (Applied Biosystems). Each sample was analyzed in triplicate with 100ng of cDNA. The thermal profile of the qRT-PCR was set at 95 degrees C for 10 minutes, then 40 cycles at 95 degrees for 15 seconds per cycle, and finally 60 degrees C for one minute. Amplification of the product was confirmed using a melt curve analysis. Threshold cycle was determined for each sample using ABI Systems 7500 software. Relative quantification was calculated in Excel, significance of each sample was determined using a two-tailed T-test.

2.3 Results and Disscussion

Although Holmberg bred several lines in his program that exhibited abiotic stress tolerance, Fiskeby III was chosen as the representative for this project because it exhibited the greatest ozone tolerance of the Fiskeby germplasm tested in a follow-up open-top chamber field trial after the initial greenhouse screening (Burkey and Carter Jr.). Similarly, Fiskeby III was chosen as one of the parental lines in a recombinant inbred population derived from a cross with Mandarin (Ottawa), an abiotic stress sensitive line being used by Burkey and Carter to investigate several abiotic stresses including ozone tolerance. This will facilitate the eventual comparison and integration of results from both classical genetic and molecular genomic investigative methods.

RNA-seq libraries were sequenced on the Illumina HiSeq2000 and generated 40

million to 200 million reads per lane (Figure 1). Following quality control, Tophat mapped between 84-95% of reads back to *Glycine max* Wm82.a2.v1 reference genome (Trapnell, Pachter, and Salzberg; Schmutz et al.). For each sample, only a small number of reads, ranging from ~3,000 to 72,000 reads failed to meet QC standards in the Tophat mapping process. Read quality figures for read duplication as well as dispersion were also created to assess read quality (Figure 5,6.

2.3.1 Analysis of RNA-seq Results to Determine Ozone Response in Ozone-Sensitive Soybean

Using *cuffdiff* (Trapnell et al.), differential analysis of the sequencing results of the Mandarin (Ottawa) at low and high ozone concentrations results yielded 135 to 275 differentially expressed genes per time point, or less than 1% of the total number of annotated genes in the reference genome (Table 1). In total, 535 unique genes showed differential expression at one or more time points. Genes with annotated functions or processes matching phenylpropanoids, and lignin biosynthesis were found in all four time points, while genes corresponding to glutathione, lipoxygenase, and phenylalanine ammonia-lyases were found in one or more time points but not in all four time points. Of the entire set of genes differentially expressed in all four time points 267 genes had a consistent pattern of up or down regulation across all four time points. Of these genes, 230 showed higher expression in the high ozone condition, and 15 correspond with GO functions related to oxidative or general stress response in soybean: Glyma.09G156700, Glyma.11G078400 and Glyma.17G036200 are putative oxidative stress response genes that code for perioxidase genes. Glyma.05G231900 is a gene involved in phenylpropanoid biosynthesis, but also matches GO terms for systemic acquired

resistance, a pathway in plants that "warns" surrounding plant tissues after a localized pathogen exposure, making its expression here particularly interesting. Glyma.10G162400 is a transcription-regulating gene that matches GO terms implicating it in auxin, cadmium, and osmotic stress response. The remaining 10 genes, Glyma.02G142500, Glyma.04G003200, Glyma.04G088500, Glyma.04G248500, Glyma.05G109600, Glyma.07G018000, Glyma.10G001800, Glyma.10G179400, Glyma.14G141000, Glyma.14G216200, all match GO terms related to salt stress, and other terms such as jasmonic acid response, response to cold, cell wall biogenesis, and oxidation-reduction (Figure 2). Because of the prevalence of these pathways in the results, it is likely that many of the genes in salt and cold stress response also play a yet uncharacterized role in oxidative stress response.

In addition to the enrichment of expected oxidative stress responses in ozone exposed samples, an analysis of individual time points using gene ontology (GO) term enrichment showed a shifting enrichment in transcriptional regulation over the time course. During time points 1-3, a higher proportion of differentially expressed genes came from larger FPKM values in the high ozone exposure samples (165:100, 92:43, 214:1, respectively), and the high exposure sample had a larger number of genes matching GO terms related to transcriptional control (11:7, 1:1, 36:0) . In the final time point, this was reversed: more differentially expressed genes were due to larger FPKM values in the low ozone samples (57:126), and more differentially expressed genes matched GO terms related to transcriptional control (5:16). This is likely consistent with the expected decrease in metabolic and photosynthetic functions in response to ozone exposure (Reddy et al.; Mittler) (Burkey, pers. comm.).

2.3.2 Analysis of RNA-seq Results to Determine Ozone Response in Ozone-Tolerant Soybean

Differential analysis of the sequencing results of Fiskeby III samples comparing low to high ozone exposure showed fewer differentially expressed genes than the Mandarin (Ottawa) samples; the number of differentially expressed genes ranged from 88 to 384 genes, a similar number of genes to the previous analysis of Mandarin (Ottawa) at low and high ozone. (Table 1). In contrast to the Mandarin (Ottawa) series, only lignin biosynthesis and lipoxygenase showed some degree of differential expression across all four time points. Phenylalanine ammonia-lyases showed expression in the first two time points only. Phenylpropanoid genes showed differential expression in all but the final time point, while glutathione had minimal differential expression with only one instance of differential expression, occurring at the final time point. 470 unique genes were found to have differential expression in one or more time point, and these genes did show a significant GO term abundance for oxidation reduction when analyzed through AgriGO ($p=0.00044$, Figure 3). When the entire time series was examined as a whole, however, 1682 genes were found to have a consistent pattern of differential expression, by far the largest number of differentially expressed genes in any sample comparison. Of these 1682 genes, 863 had higher expression in the high ozone samples. Significant GO terms returned from AgriGO were mostly related to lipid, carbohydrate, and fatty acid metabolic processes. However, it did also show a significant number of GO terms related to negative regulation of molecular function ($p=3.7e-5$). There were a number of GO terms related to oxidative stress, however the p-value for those terms was not statistically significant ($p=0.051$). An expression heatmap of these genes can be found in Figure 4.

2.3.3 Comparison of Tolerant and Sensitive Genotypes

Comparing the high ozone exposure between both Mandarin (Ottawa) and Fiskeby III at individual time points resulted in 258 to 536 differentially expressed genes, a similar number to the comparisons of each cultivar at low and high exposure described above. Lipoxygenase and lignin genes were found at all four time points, whereas phenylpropanoids and glutathione genes were found at time points 2-4. Phenylalanine ammonia-lyases were found at the first, third, and fourth time points. The 258 to 536 differentially expressed genes per time point correspond to 564 unique genes with differential expression at a minimum of one time point. An AgriGO analysis of these 564 genes identified a significant abundance of GO terms related to oxidation reduction ($p=5.1e-6$). When the time series was examined as a whole, only 64 genes were shown to have a consistent pattern of expression across all four time points. Of these 64 genes, 58 had higher expression in the ozone sensitive Mandarin (Ottawa), and only 6 had higher expression in Fiskeby. Of the 6 genes that had higher expression in Fiskeby, we were only able to identify GO terms for four. Glyma.14G164900 and Glyma.14G165000 matched identical GO terms for oxidation-reduction and response to light stimulus, and are likely tandemly duplicated genes. Glyma.11G197300 is a cytochrome P450 gene that matches a large number of biological process GO terms including defense response, response to water deprivation, and induced systemic resistance. Glyma.08G287500 matches GO terms for cell wall organization, anthocyanin accumulation, and regulation of hormone levels. 58 genes had higher expression in Mandarin (Ottawa), and 24 of these matched GO terms related to oxidative or general stress response. A summary of these genes and a brief description of their matching biological proces GO terms can be

found in Table 2. Of particular note in these results are the two transcription regulators, Glyma.13G203700 and Glyma.10G035500, as well as Glyma.15G182600 which matched GO terms for regulation of hydrogen peroxide and systemic acquired resistance. The previously described SAR gene, Glyma.05G231900 was expressed in both cultivars and not deemed significantly different.

While these results highlight some of the differences in the oxidative stress responses between tolerant and sensitive genotypes, also of interest is the timing of responses as the samples transition from one time point to the next when comparing the two cultivars at high ozone exposure. One posited explanation for the ozone tolerance seen in the Fiskeby cultivar was a delay or limited ozone uptake through some yet uncharacterized mechanism. This made an examination of the timing of differential gene expression between cultivars of particular interest. We examined a number of patterns (Table 3) and identified a number of different genes that showed delayed or accelerated expression in each cultivar. The 39 genes that showed an accelerated expression in Mandarin (Ottawa) were not sufficient to generate significant GO terms from AgriGO, but could still be examined individually. Of these 39, there were 11 that could be implicated as part of oxidative or general stress response. These include 3 oxidation-reduction process genes, Glyma.07G225300, Glyma.10G203500, and Glyma.20G051700. There were also three genes that matched GO terms relating to transcriptional control, Glyma.08G194900, Glyma.19G035300, and Glyma.10G082500. This last gene, Glyma.10G082500, is of particular interest, as it matched with over 52 biological process GO terms, many of which are implicated in oxidative stress and other stress response pathways. The analysis of genes matching a pattern of accelerated

expression in Fiskeby, 27 genes in total, was also insufficiently large to produce significant GO abundances from AgriGO. In addition to there being fewer genes matching this pattern of expression, there were far fewer genes within the results related to oxidative stress. Only Glyma.03G162700, an ethylene signaling pathway gene, and Glyma.18G087000, a defense response gene, matched GO terms that could be attributed to stress response. The apparent lack of an accelerated response from the ozone tolerant plant is one of the more compelling results that lend evidence to the hypothesis that the physical characteristics of the Fiskeby leaf limit the severity of ozone exposure and therefore limit leaf damage.

Another pattern of expression examined were the genes that were expressed in both cultivars at the first time point at high ozone exposure but were turned off at a later time point in one cultivar while the other maintained expression under high ozone conditions. Both possible configurations of this pattern had a small number of genes associated with it: 65 genes in which expression was maintained in Fiskeby III, 39 genes in which expression was maintained in Mandarin (Ottawa) (Table 3). In the first scenario, where expression is maintained in Fiskeby III but reduced or eliminated in Mandarin (Ottawa), these genes were overwhelmingly related to defense or stress response. Of these genes, 10 matched GO terms related to systemic acquired resistance, 4 additional genes matched GO terms for oxidative stress response, 7 more were related to oxidation-reduction, and 6 others related to jasmonic acid or ethylene response . In addition to there being fewer genes with expression maintained in Mandarin (Ottawa), the genes did that did match this pattern did not show the same prevalence as the previous pattern towards stress or defense responses. Only one gene in this set,

Glyma.01G153300 was related to oxidation-reduction, and no genes matched GO terms for oxidative stress response. One additional gene matched GO terms related to jasmonic acid response and several abiotic stress responses, another matched terms related to hydrogen peroxide response.

Table 4 shows genotypic differences in leaf physiology in response to ozone treatment in both Fiskeby III and Mandarin (Ottawa). Leaf foliar injury, leaf air space and the specific weight of leaves all confirm that Mandarin (Ottawa) is exhibiting greater ozone sensitivity than Fiskeby III. The leaf air space data shows that there is much more exposure of cell surface area in leaves from Mandarin (Ottawa), a potential basis for the greater ozone sensitivity. The specific leaf weight data shows that there is more cellular mass in Fiskeby III leaves as a result of the leaves being either thicker and/or having more densely packed cells.

2.3.4 Analysis of Wax and Cutin Biosynthesis Pathway

One pathway of particular interest to our investigation was the cuticle wax biosynthesis pathway. During our initial probing of the data using Cufflinks 2.0.2 and the soybean reference genome (version 1.89), we identified an abundance of genes belonging to the cuticle wax biosynthesis pathway that matched expression pattern 3 (expression in both cultivars, expression tapers or ends in Fiskeby). This result was unexpected for two reasons: first, cuticle wax biosynthesis is not well documented as a part of ozone tolerance and has only been documented in a limited set of species; and secondly, if it is down regulated in the ozone tolerant cultivar, what can the presence of this pathway tell us about ozone tolerance. A literature review of ozone tolerance in other plant species yielded some promising information regarding its possible role in

ozone tolerance and resistance in plant species. In plums, cranberries and spruce, ozone exposure results in a significant decrease in cuticle depth (Crisosto et al.; Norton, Charig, and IE; Elstner, Osswald, and Youngman). Furthermore, ozone is more readily deposited onto the leaf surface when the cuticular layer is reduced leading to greater foliar injury in response to ozone, and ozone has been previously implicated in the degradation of cuticular wax (Hellgren et al.; Shepherd and Wynne Griffiths). Work by Zhao et al. showed that the application of an exogenus chitosan relieved some of the damage caused by ozone exposure in soybean (Zhao et al.).

When the read mapping was updated for the new genome version, using Cufflinks 2.2.1, the script for determining differentially expressed genes matching certain patterns no longer returned the same abundance of wax biosynthesis pathway genes that our previous analysis had. At this time, we hypothesize that this is likely due to the overall lower number of differential gene calls,, which in turn causes there to be too few genes to identify the pathway through AgriGO or other GO term abundance tools. In the initial pattern analysis which revealed the cuticle wax pathway we observed 143 genes where we only observe 43; with the exception of the Fiskeby low exposure to Fiskeby high ozone exposure time series comparison, all other comparisons saw a decrease in the number of differentially expressed genes of a similar magnitude. Additionally, the exclusion of cuticle wax genes from the pattern results could be also attributed to the new genome or annotations as well; only 66 genes in the new annotations match GO terms for wax biosynthesis. If we examine the individual gene annotations from the pattern script, there are several genes we can attribute to the wax biosynthesis pathway. These include two ketoacyl-CoA synthases (Glyma.10G179400 and Glyma.U033000), which are

involved in fatty acid elongation for cuticle wax formation (Kunst and Samuels). There is also a wax ester synthase gene (Glyma.06G291700). Using the 66 genes matching GO terms for wax biosynthesis, we were able to query the data for those genes specifically and were able to identify three genes which showed differential expression. Of these genes, Glyma.11G185100, Glyma.12G183400 showed significant differential expression at time point 2, and had a fold change corresponding to higher expression in Fiskeby at that time point. However, by time point four the fold change had changed to correspond to higher expression in Mandarin (Ottawa), although it was not found to be significant (log2(fold change) = 8.925). The other gene, Glyma.13G317600, showed differential expression at time points one, two, and four. At the first two time points, the fold change showed higher expression in Fiskeby, though by time point four Glyma.13G317600 had higher expression in Mandarin (Ottawa). All three of these genes are substantiate the previously observed pattern of expression, where initial expression favors Fiskeby or neither cultivar, and over the time points shifts to higher expression in Mandarin (Ottawa). Additionally, we also used a list of Arabidopsis cuticle wax biosynthesis genes from Le Provost et al., a study that linked cuticle wax and drought tolerance (Le Provost et al.). Soybean orthologs to the *A. thaliana* genes from Le Provost were identified using BLAST, and the cufflinks results for those genes, in addition to the 66 genes matches the GO term for wax biosynthesis, can be seen in Appendix A.1.1. While only a limited number of these genes have statistically significant differential expression, there is a strong overall trend that matches that seen in the pattern 3 genes. Specifically, we see more genes with fold changes values corresponding to higher expression in Fiskeby in

the first time point, which decreases to more genes showing higher expression in Mandarin (Ottawa) by the final time point.

Since literature supports the idea of the cuticular wax pathway being an important component of ozone tolerance, then the critical question becomes why do we observe this pattern of expression that favors the ozone intolerant cultivar? To answer this question we reviewed the physical characteristics of the leaves of each cultivar as seen in Table 4. Reviewing these characteristics reveals what we believe to be a critically important factor when attempting to explain the expression of cuticular wax pathways in the ozone intolerant Mandarin (Ottawa). What becomes immediately evident from the physical data is that the leaves of the ozone tolerant Fiskeby cultivar are more massive and the cell density of the leaf tissue is much higher than that in Mandarin (Ottawa). We believe that because of this the ability for ozone to penetrate the leaf tissue and inflict injury upon the leaf tissue is severely limited. Because Mandarin (Ottawa) lacks this specific benefit, we see expression of cuticle wax pathways in an attempt to bolster the physical barrier to ozone penetration of leaf tissues. This may also explain the differences in anti-oxidant expression between the two cultivars: if Fiskeby, by virtue of its thicker leaf structure, does not need to express cuticular wax biosynthesis in order to build a physical barrier, than perhaps it is able to devote those transcriptional resources towards anti-oxidant pathways. And because this response is already dealing with a less significant influx of active oxygen species compared to that in Mandarin (Ottawa), it is able to significantly limit the degree of leaf damage the plant suffers.

2.3.5 Verification by qRT-PCR

Although the results described here are only an initial sampling of the future PCR work

that will be done, they sufficient to report as verification of our RNA-seq work. The two genes tested, Glyma.02g141800 and Glyma.13g028200, both showed similar patterns of expression and nearly identical DE calls to those in our RNA-seq analysis. In all cases, the photosynthesis related gene Glyma.13g028200 failed to generate a significantly differentially expressed call at any time point at high ozone (Table 6). The glutathione gene did have one DE call in the qRT-PCR results (Table 5), but since this time point was not sampled in the RNA-seq experiment it cannot serve as a validation of the RNA-seq work. Although these genes failed to show differential expression, the similar levels of expression do serve to validate the RNA-seq expression.

2.4 Conclusions

The analysis of the RNA-seq data generated from a time series exposure of ozone tolerant Fiskeby and ozone intolerant Mandarin (Ottawa) soybean varieties was remarkable not just for the genes identified during the study, but also for how limited the set expressed genes was. Only one comparison made was able to find differential expression in more than 1% of soybean genes. This observation is likely due to combination of factors, the relatively short ozone exposure times means we are only observing initial gene expression responses and likely the start of response pathways. We also must consider that the low ozone controls means we can reliably say that pathways and genes unrelated to oxidative stress would not be differentially expressed when we compare the low and high exposure sample sets of each cultivar. When comparing the high exposure samples for each cultivar to each other, it is likely that the low number of differentially expressed genes is a function of how similar the two cultivars are, which in turn limits the number of genes which would be expressed differently between the two cultivars. Despite the

small number of differentially expressed genes in each comparison set, we were able to identify differentially expressed genes that correspond to each of the known or expected categories of genes identified in other species. Despite the presence of the expected responses in each cultivar, the timing of expression of these expected genes and responses is distinct between each cultivar, and likely this is at least partly responsible for the difference in the outcome from ozone exposure. Another important finding, related to the small number of genes, is the apparent lack of response observed in Fiskeby in both the high exposure comparison to Mandarin (Ottawa) in both the number of differentially expressed genes and also in the gene expression pattern analysis. The fact that we observe only 3 genes in Fiskeby related to oxidative stress expressed first in Fiskeby before Mandarin (Ottawa), as well as the comparison of Fiskeby and Mandarin (Ottawa) at high ozone which only returned 6 genes which showed higher expression over the entire time series lead us conclude that there is an overall diminished genetic response to ozone exposure in the ozone tolerant variety compared to the ozone intolerant variety. When combined with the evidence of cuticle wax expression in the ozone intolerant variety and the differences in the physical characteristics of each leaf, we believe that at least part of the observed ozone tolerance of Fiskeby is due to its thicker, denser leaves provide passive resistance which limits the degree of ozone exposure. The observed diminished genetic response is then likely a consequence of this reduced exposure.

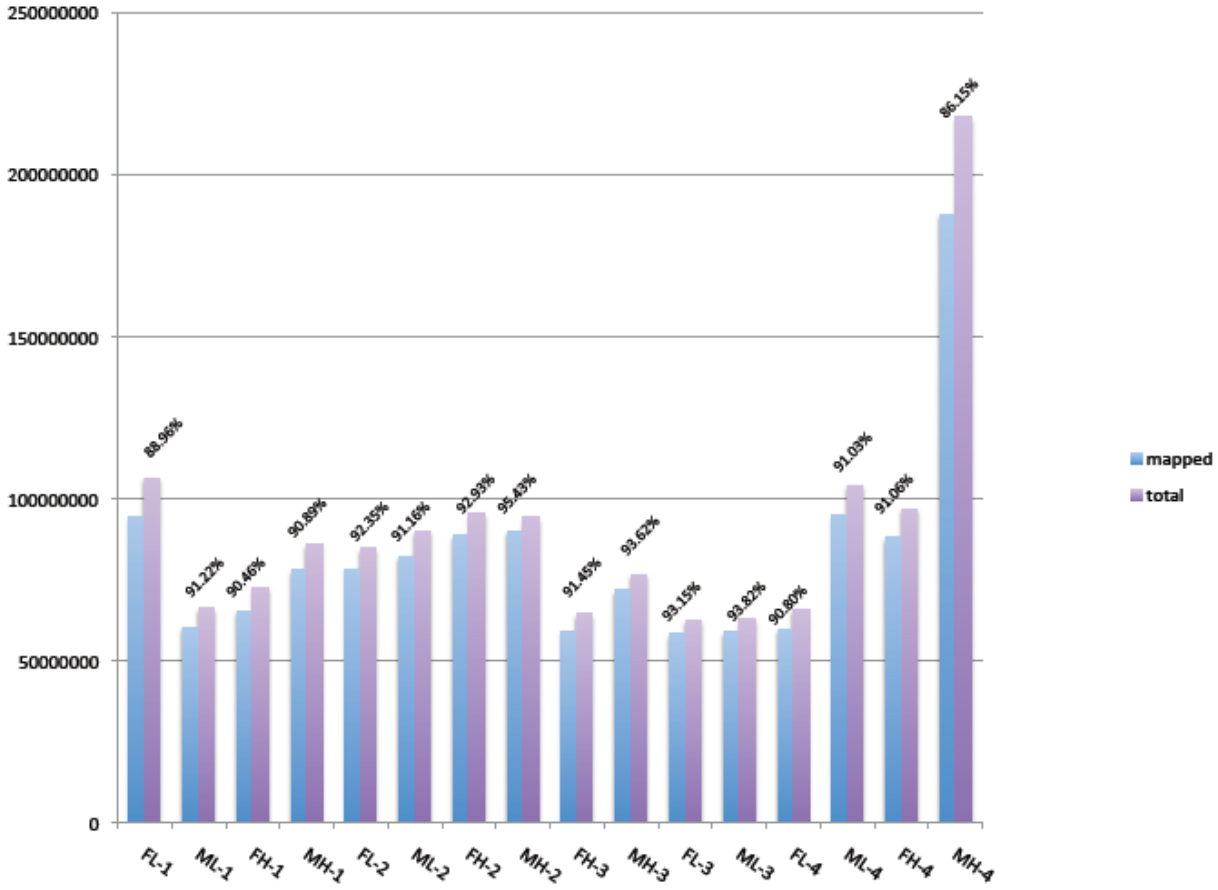

Figure 1. Graph showing total reads per sample, and the number of those reads which mapped to the G. max genome as a percent.

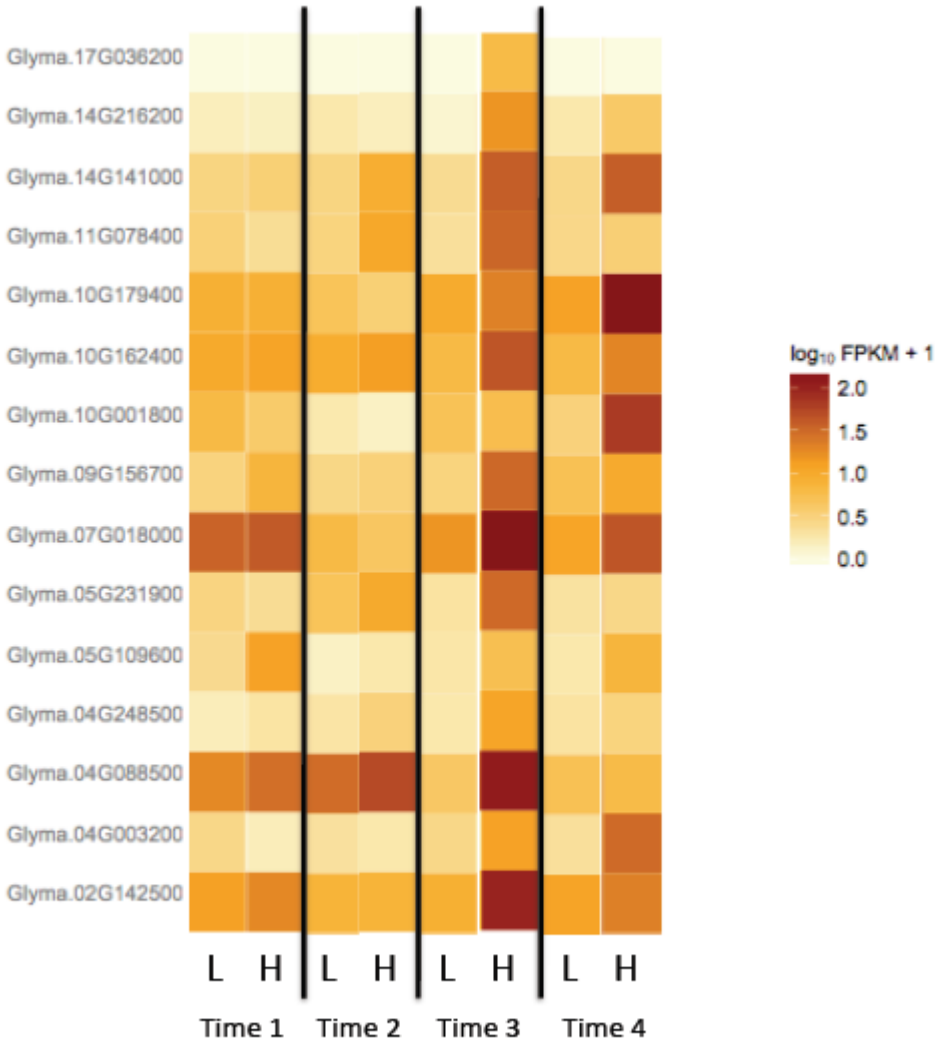

Figure 2: Heatmap of stress response genes identified as differentially expressed in the comparison of Mandarin (Ottawa) at low and high ozone exposures.

A heatmap of 15 genes identified as differentially expressed by cufflinks ($p<0.05$) in the Mandarin (Ottawa) low and high exposure comparison. These genes were selected from amongst the 230 total genes with higher expression in the high ozone treatment because of GO terms matching oxidative or general stress response pathways. Strength of expression (FPKM) is indicated by color, ranging from very low yellow to deep red at high FPKM values.

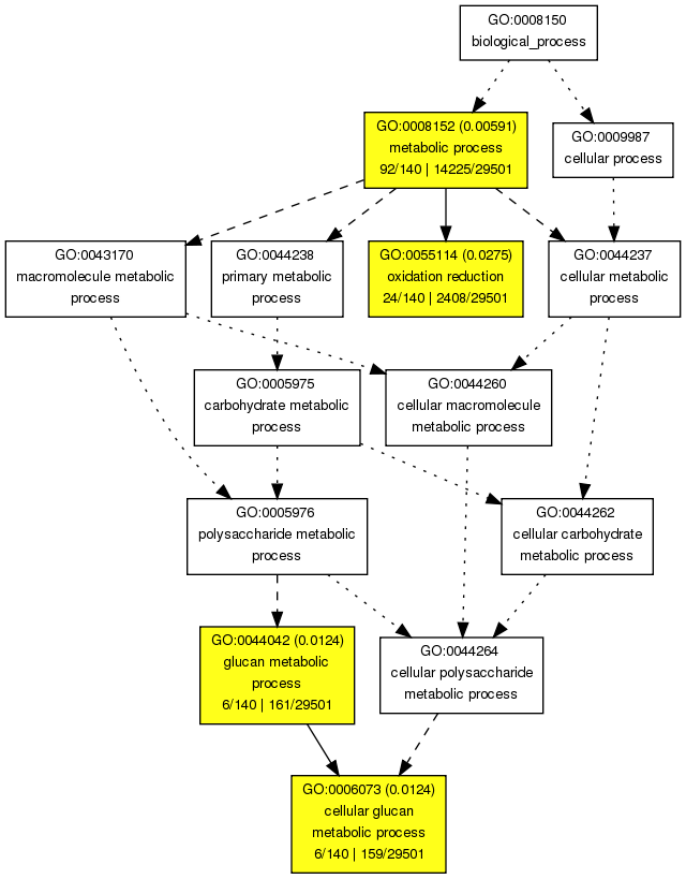

Figure 3: agriGO results of genes differentially expressed in Fiskeby low to high ozone

Results of a SEA Analysis in AgriGO of genes identified as differentially expressed by cufflinks ($p<0.05$) in at least one time point in the Fiskeby low and high exposure time series. Significance level of enrichment is displayed by color scale, where white indicates no significant enrichment, then transitioning from yellow to red to indicate strength of significance. Ratios at the bottom of each GO box represent the number of genes in the input list that matched that GO term, the number of total genes in the input list, total genes in the background genome set that match that GO term, total genes in the background set. At the top of each colored box, next to the GO term, is the adjusted p-value for each enriched GO term.

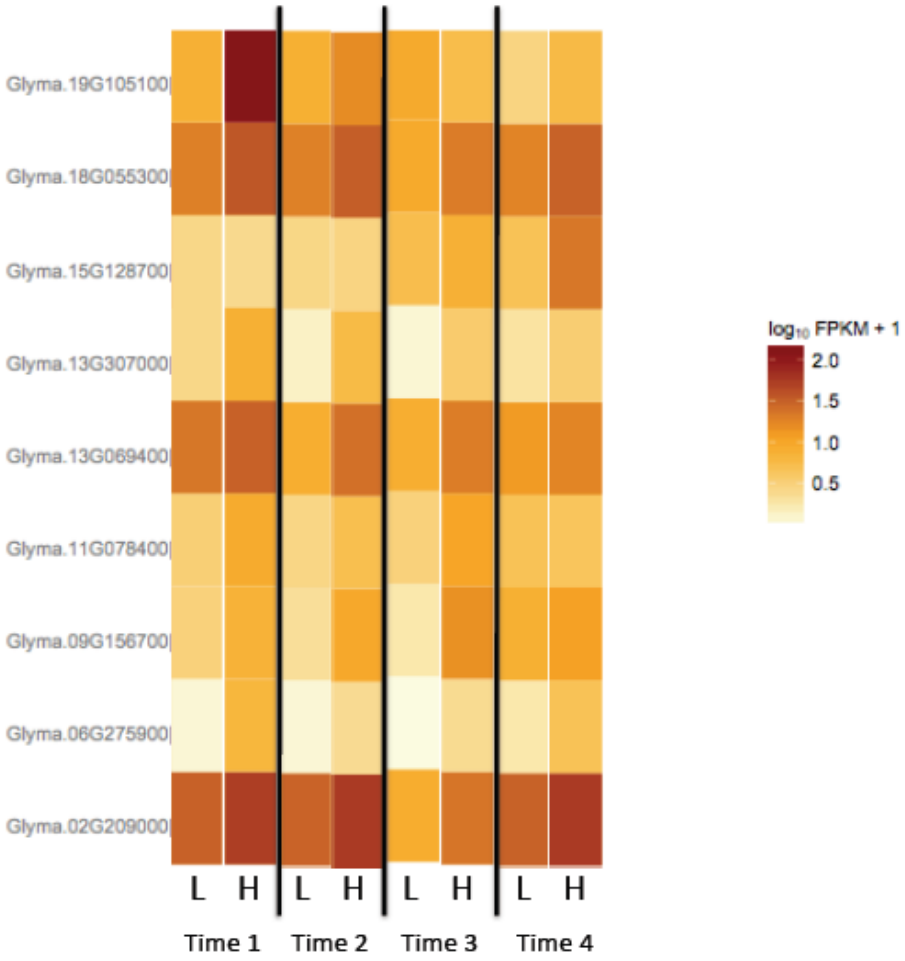

Figure 4: Heatmap of FPKM values of oxidative stress genes in Fiskeby low and high exposure across all time points
A heatmap of FPKM values for 9 genes identified by agriGO as oxidative stress genes.
Strength of expression (FPKM) is indicated by color, ranging from very low yellow to deep red at high FPKM values. The genes shown were found to be significantly differentially expressed by cufflinks ($p<0.05$), but were not sufficient to pass an agriGO analysis for statistically significant GO term enrichment ($p=0.051$).

Table 1. A description of comparison and labeling methodology. The number of differentially expressed genes identified by *cuffdiff* is also shown.

Sample 1	**Sample 2**	**Time Point**	**Label**	**Diff. Expressed Genes**
Mandarin-Ottawa Low Exposure	Mandarin-Ottawa High Exposure	1	ML-MH-1	275
Mandarin-Ottawa Low Exposure	Mandarin-Ottawa High Exposure	2	ML-MH-2	135
Mandarin-Ottawa Low Exposure	Mandarin-Ottawa High Exposure	3	ML-MH-3	215
Mandarin-Ottawa Low Exposure	Mandarin-Ottawa High Exposure	4	ML-MH-4	183
Fiskeby Low Exposure	Fiskeby High Exposure	1	FL-FH-1	384
Fiskeby Low Exposure	Fiskeby High Exposure	2	FL-FH-2	88
Fiskeby Low Exposure	Fiskeby High Exposure	3	FL-FH-3	129
Fiskeby Low Exposure	Fiskeby High Exposure	4	FL-FH-4	139
Fiskeby High Exposure	Mandarin-Ottawa High Exposure	1	FH-MH-1	310
Fiskeby High Exposure	Mandarin-Ottawa High Exposure	2	FH-MH-2	332
Fiskeby High Exposure	Mandarin-Ottawa High Exposure	3	FH-MH-3	258
Fiskeby High Exposure	Mandarin-Ottawa High Exposure	4	FH-MH-4	536

Table 2. Table of soybean genes and biological process GO terms associated with those genes matching stress response functions in the sensitive and tolerant comparison.

Gene ID	Biological Process GO terms
Glyma.02G205800	cell wall biogenesis; "cell wall macromolecule metabolic process"; "hydrogen peroxide biosynthetic process";
Glyma.03G145600	defense response to nematode; "oxidation-reduction process"; "response to oxidative stress"
Glyma.03G222000	"response to cold"; "response to high light intensity"; "response to hydrogen peroxide";"response to salt stress"; "response to water deprivation"
Glyma.04G063800	"defense response to bacterium"; "defense response to fungus"; "plant-type cell wall biogenesis"; "response to osmotic stress";
Glyma.04G088500	"response to salt stress"
Glyma.04G227700	"lignin biosynthetic process"; "phenylpropanoid metabolic process"; "positive regulation of flavonoid biosynthetic process"; "response to wounding"
Glyma.06G065000	"defense response to bacterium"; "defense response to fungus"; "plant-type cell wall biogenesis"; "response to osmotic stress"; "response to water deprivation"; "xylan biosynthetic process"
Glyma.06G158800	"hydrogen peroxide biosynthetic process"; "proteolysis";
Glyma.07G133900	"lignin biosynthetic process"; "oxidation-reduction process"; "phenylpropanoid metabolic process";
Glyma.07G157100	"defense response"; "lignan biosynthetic process"
Glyma.09G005600	"hyperosmotic response"; "lipid metabolic process";"protein targeting to vacuole"; "response to salt stress"; "response to temperature stimulus"
Glyma.09G038500	"cell wall macromolecule catabolic process"; "hydrogen peroxide biosynthetic process"; "lignin biosynthetic process";
Glyma.09G051100	cell wall biogenesis;"defense response to bacterium"; "defense response to fungus";
Glyma.10G035500	regulation of transcription, DNA-dependent
Glyma.11G053200	glucuronoxylan metabolic process; "secondary cell wall biogenesis";
Glyma.12G233700	"response to endoplasmic reticulum stress"; "response to heat"; "response to high light intensity"; "response to hydrogen peroxide"; "signal transduction"
Glyma.13G094900	"plant-type cell wall organization"; "response to chitin"; "response to cold"; "response to heat"; "response to mechanical stimulus"; "response to wounding"

Table 2 (Continued). Table of soybean genes and biological process GO terms associated with those genes matching stress response functions in the sensitive and tolerant comparison.

Glyma.13G203700	plant-type cell wall modification; "regulation of transcription, DNA-dependent"; "transmitting tissue development"
Glyma.13G301900	response to wounding
Glyma.15G110200	Golgi organization; "hyperosmotic response"; "lipid metabolic process"; "response to salt stress"; "response to temperature stimulus";
Glyma.15G143600	"cell wall macromolecule catabolic process";"hydrogen peroxide biosynthetic process"; "lignin biosynthetic process"; "xylan biosynthetic process"
Glyma.15G182600	"regulation of hydrogen peroxide metabolic process"; "response to hypoxia";"systemic acquired resistance, salicylic acid mediated signaling pathway"
Glyma.17G072200	cell wall biogenesis;"hydrogen peroxide biosynthetic process";
Glyma.19G008200	pollen tube growth; "regulation of defense response"

Table 3. A description of each gene expression pattern examined as part of the expression timing analysis. The number of genes that matched a particular pattern is also included.

Number of genes matching pattern	Pattern Description
34	Starts with 0 expression in both cultivars, faster response in Mandarin (Ottawa)
27	Starts with 0 expression in both samples, faster response in Fiskeby
65	Starts with expression in both samples, expression lowers in Mandarin (Ottawa) while Fiskeby remains expressed
39	Starts with expression in both samples, expression lowers in Fiskeby while Mandarin (Ottawa) remains expressed

Table 4. Genotype differences in leaf physiological parameters

Physiological Parameter	Genotype	Treatment (ppb O_3)	Leaf 3	Leaf 4	Leaf 5
Foliar Injury	Fiskeby III	27	0	0	0
(% leaf area)	Fiskeby III	74	0	0	0
N = 4	Mandarin (Ottawa)	27	0	0	0
	Mandarin (Ottawa)	74	33	38	3
Air Space	Fiskeby III	27	0.49 ± 0.02	n/a	0.45 ± 0.02
(ml gFW^{-1})	Fiskeby III	74	0.50 ± 0.02	n/a	0.43 ± 0.02
N = 4-6	Mandarin (Ottawa)	27	0.70 ± 0.02	n/a	0.74 ± 0.02
	Mandarin (Ottawa)	74	0.69 ± 0.01	n/a	0.83 ± 0.01
Specific weight	Fiskeby III	27	n/a	57 ± 2	n/a
(gDW m^{-2})	Fiskeby III	74	n/a	53 ± 1	n/a
N = 6	Mandarin (Ottawa)	27	n/a	36 ± 2	n/a
	Mandarin (Ottawa)	74	n/a	31 ± 1	n/a

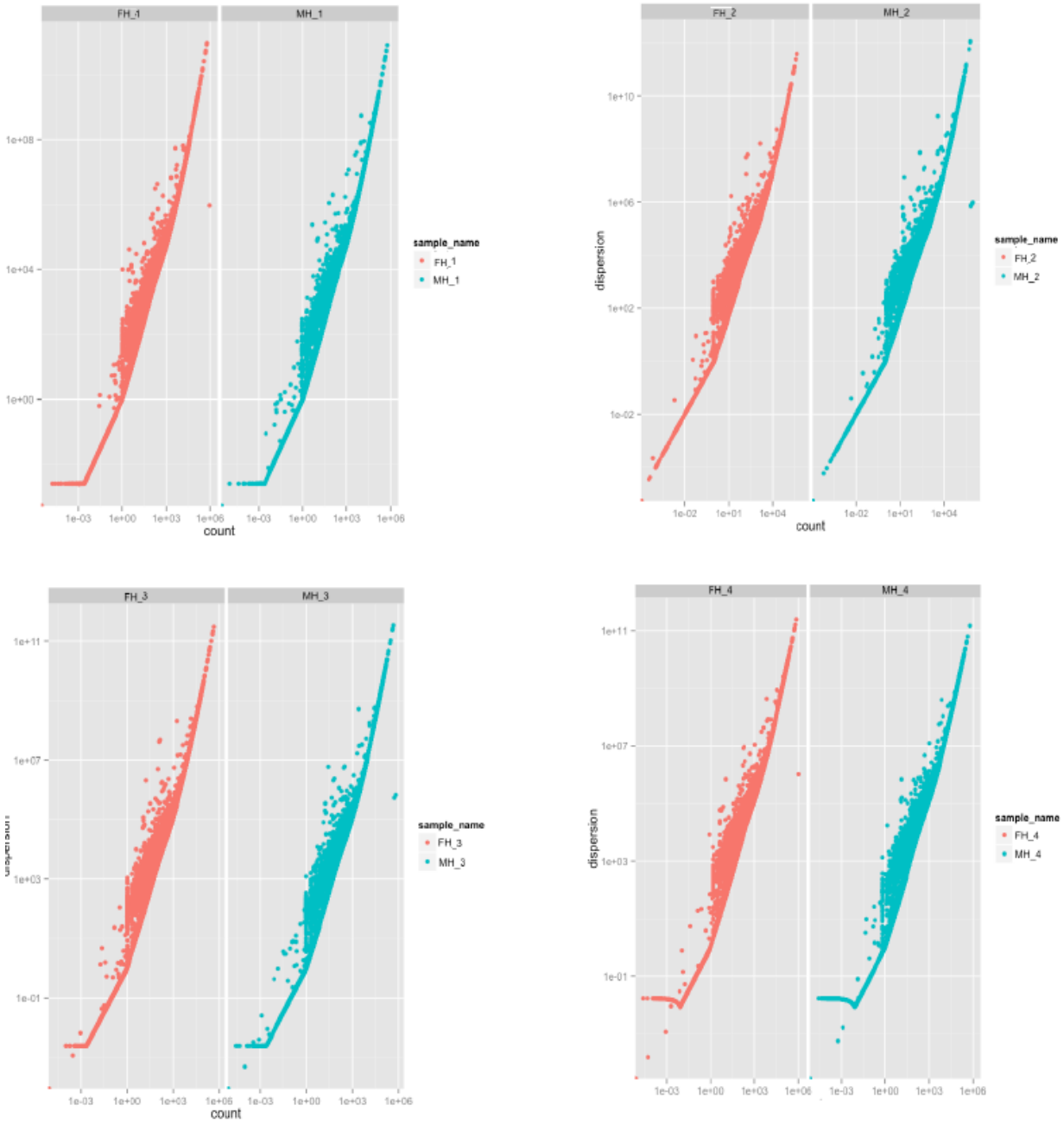

Figure 5. CummeRbund overdispersion graph assessing read quality in the high ozone samples.

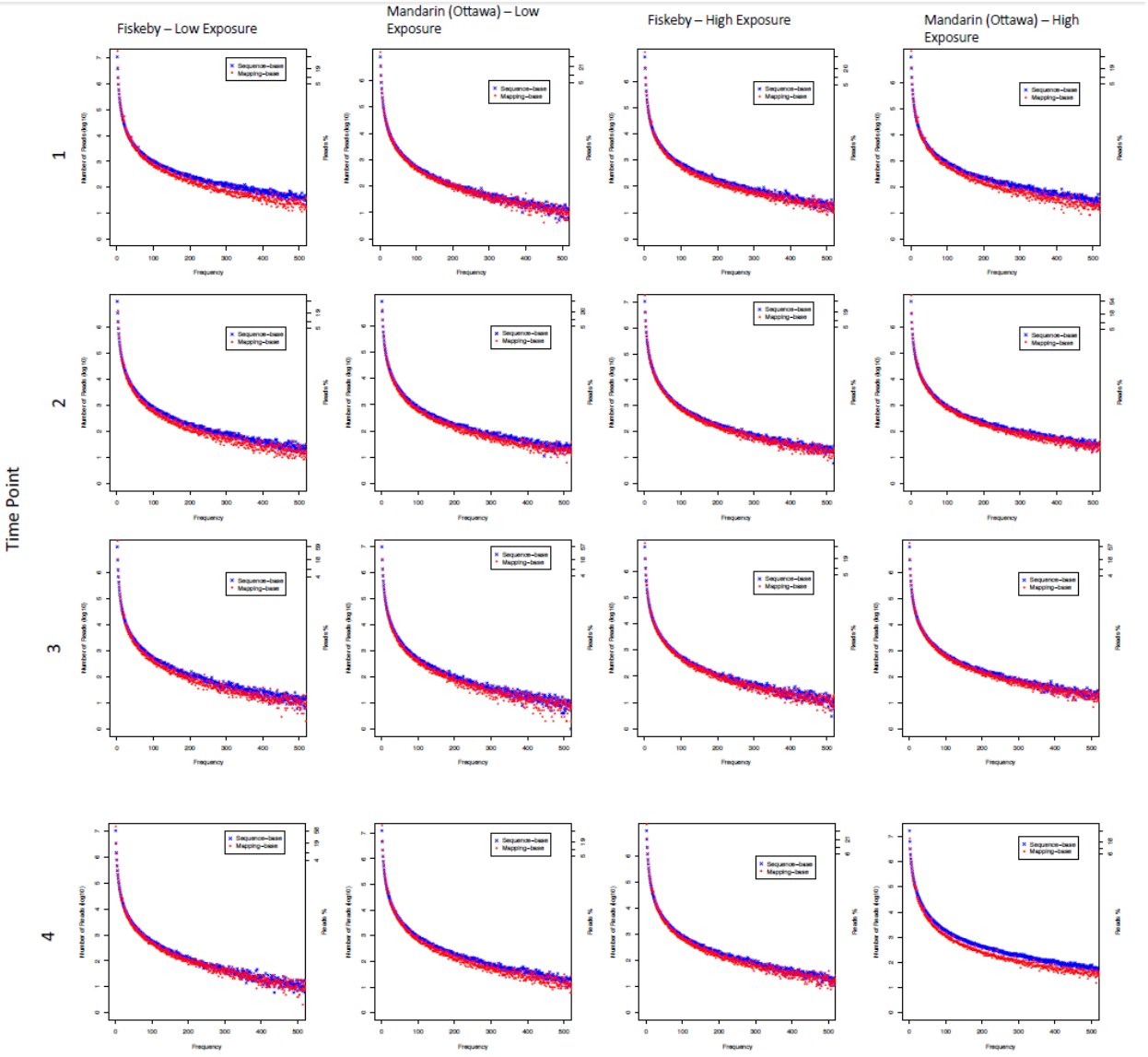

Figure 6. Read quality assessment using RSeQC. Here, number of identical reads (blue) is mapped against the number of reads mapped to identical locations (red) for each sample.

Table 5. qRT-PCR results of the glutathione gene chosen for analysis. Numbers in the sample label correspond to the number of hours post exposure at which the sample was taken. RQ is the relative quantification versus a control sequence.

	Glutathione				
	RQ #1	RQ #2	RQ #3	RQ mean	RQ SD
M0	2.1919857	1.9237738	3.584791	2.56685028	0.891705
F0	0.0592608	1	0.797949	0.6190701	0.495223
		Significance	0.029723		
M1	6.920359874	4.2645264	1.025523	4.07013629	2.952222
F1	4.337932168	2.2868746	1.079313	2.56803989	1.647404
		Significance	0.484485		
M2	1.0619288	1.2494702	1.297521	1.20297333	0.124489
F2	1.0935994	0.2302034	0.12472	0.48284093	0.531555
		Significance	0.084348		
M4	0.795882	0.453722	0.886567	0.71205713	0.228273
F4	0.869205	0.6762475	0.623538	0.72299693	0.129334
		Significance	0.945893		
M8	0.0241709	0.3263804	1.83379	0.72811373	0.969393
F8	0.3693302	0.2870487	0.584942	0.4137736	0.153839
		Significance	0.608652		
M12	0.6310072	0.4795139	0.11624	0.40892043	0.264545
F12	0.9498962	0.2736172	0.138298	0.45393723	0.434809
		Significance	0.885661		
M24	0.7847364	1.2079639	0.296587	0.7630956	0.456074
F24	1.8861796	1.6293425	0.907289	1.47427034	0.507535
		Significance	0.145351		

Table 6. qRT-PCR results of the photosynthesis-related gene chosen for analysis. Numbers in the sample label correspond to the number of hours post exposure at which the sample was taken. RQ is the relative quantification versus a control sequence.

	pD1				
	RQ #1	RQ #2	RQ #3	RQ mean	RQ SD
M0	0.924041	1.264936	1.985988	1.391655	0.542196
F0	0.086688	1	1.15829	0.748326	0.578436
		Significance	0.232594		
M1	0.51466	1.675365	3.001244	1.730423	1.244206
F1	0.393083	0.626345	0.394885	0.471438	0.134157
		Significance	0.156376		
M2	0.764242	0.001197	0.001204	0.255548	0.440542
F2	0.865188	0.482722	0.723177	0.690362	0.193333
		Significance	0.192539		
M4	4.209941	1.582349	0.001138	1.931143	2.12597
F4	0.211019	0.080885	0.559169	0.283691	0.247285
		Significance	0.253326		
M8	4.77E-05	0.252302	1.008245	0.420198	0.52465
F8	0.103745	0.284445	0.32389	0.23736	0.117383
		Significance	0.587488		
M12	0.245975	0.227334	0.009381	0.160897	0.131547
F12	0.122172	0.177858	0.200462	0.166831	0.040293
		Significance	0.944033		
M24	2.388457	1.893501	0.171876	1.484611	1.163487
F24	0.703364	0.797539	0.487856	0.66292	0.158753
		Significance	0.292207		

Table 7. Primer sequences and targets used in qRT-PCR

1. 60s- **Glyma.13g318800** (reference gene) p66) Left：AAAGTGGACCAAGGCATATCGTCG p67) Right：TCAGGACATTCTCCGCAAGATTCC
2. Glutathione**- Glyma.02g141800** (antioxidant gene) P17) Left: GGATGTGTGCCGAAGAAGTT P18) Right: TCTTTCCATGGCCTTCAATC
3. Protein D1- **Glyma.13g028200** (photosynthesis-related gene) p35) Left: TCCCGCTACTGCTGTTTTCT p36) Right: AATAAGGAGCCGCCGAATAC

CHAPTER 3: DEVELOPMENT OF EVOLUTIONARY DISTANCE PIPELINES

3.1 Introduction

In 1970, Ohno introduced the concept of neofunctionalization. This process, in which a duplicated gene would serve as the starting material for novel gene function, has been expanded upon to include a number of possible fates for duplicated genes. The statistical modeling of these fates are not limited just in their applications in molecular evolutionary theory, but can also inform researchers about the specific evolutionary history of an organism (Kimura, *The Neutral Theory of Molecular Evolution*; Lynch and Conery). Both of these areas of research leverage the ability to calculate the evolutionary distance between duplicated protein coding regions of a genome. Evolutionary distances are the number of synonymous changes (selectively neutral with no change in the resulting amino acid) and nonsynonymous changes (nucleotide changes that alters the codon resulting in an altered amino acid) in a pair of duplicated genes. These values are of great consequence in a number of fields, in that they can be used to inform researchers about the length of time that has passed since the genes duplicated, as well as the selective pressures acting on that gene. For instance, the selective pressure acting on a gene can serve as evidence for or against the neutral theory of molecular evolution, which posits that most of the mutations accumulated between species are selectively neutral, and that Darwinian selection plays only a very small role. Selective pressure can also

inform researchers about the possible outcomes of the duplication event modeled under the various fates of duplicate genes. Observing Darwinian selection could be evidence of neofunctionalization. Conversely, neutral selection could be evidence of subfunctionalization, where each gene in a duplicated pair shares a portion of the ancestral gene function. Purifying selection can be evidence for nonfunctionalization, where one of the genes in the duplicate pair becomes a pseudogene and loses function. The ability to detect the effects of selection on a gene comes from the ratio of synonymous and nonsynonymous substitutions. Because each of these values is calculated using the number of each change as the numerator and the number of possible sites as the denominator, if selection forces were neutral one would expect the ratio of these values to be approximately equal to one. In practice, however, we more often see values less than one, as nonsynonymous changes alter the coding sequence and can adversely affect gene function and are therefore less likely to become fixed in a population.

In its simplest form, the calculation of these values takes the form seen in equation 1 wherein each value is described as the number of observed synonymous or nonsynonymous changes divided by the mean number of those sites in each sequence

$$dN = Nd/N \qquad dS = Sd/S$$

(Hughes and Nei; Nei and Gojobori). The simplicity of this method makes it unsuitable for real data, as changes in the same nucleotide position will overwrite previous changes. These methods fall into two categories: counting methods, which count the number of changes and estimate multiple substitutions, and maximum likelihood methods, which use likelihood probabilities based on codon substitution models (Yang and Bielawski).

These methods all do a good job of estimating distances when the number of changes in a given set of codons is low. However, when a codon position is very different between gene pairs the accuracy of the estimate of the number of synonymous and nonsynonymous changes within that codon position is adversely affected, leading to high standard error values associated with those estimates.

By far the gold standard for deriving dN and dS is the PAML software package, which includes the *yn00* and *codeml* programs, both of which implement several well-documented maximum likelihood-based dN and dS estimation methods ; other popular alternatives include MEGA and HyPhy (Kumar et al.; Pond and Muse). The current version of PAML implements five methods of dN and dS estimation: Nei and Gojobori 1986; Yang and Nielson 2000; Li, Wu, Luo 1985; Li 1993; and Yang 2006 (referred to as LWL85, LPB93, and LWL85m in PAML, respectively) (Nei and Gojobori; Yang and Nielsen; Li, Wu, and Luo; Li; Ziheng Yang). All of these methods of estimation are counting methods with varying nucleotide substitution models. The methods of estimation of multiple substitutions for NG86, LWL85, LPB93, and LWL85m are based on the Jukes-Cantor nucleotide substitution model, while Yang and Nielson is more closely related to HKY or T93 nucleotide substitution models (Jukes and Cantor; Hasegawa, Kishino, and Yano; Tamura and Nei).

One of the major limitations in using these models to test selective pressures is the underlying assumption, when examining a pair of full length sequences, that all sites experience the same degree of selective pressure. Under this assumption, selective pressure is effectively averaged over the length of the gene, making Darwinian or purifying selection only detectable when the average dN/dS ratio across an entire

sequences exceeds or is below one. However, it has been known for quite some time that this assumption is false (Yang and Bielawski). In structural terms, not every codon is created equal, and will not have equal importance in the structure and function of the gene product. In an effort to deal with this limitation, two strategies have been proposed in recent years. The first category of these is a lineage based approach (Schmid and Yang). These methods utilize gene information from large number of species in order to reconstruct an ancestral sequence. Using the ancestral sequence, the aligned gene families are compared to identify codons which show higher than average occurrence of nonsynonymous substitutions at a specific site. These methods are effective at finding single codons under Darwinian selection, but are constrained by the ability to create a high quality ancestral sequence and possible errors based on the assumptions from deriving it. The second strategy used to find regions of purifying or Darwinian selection is the sliding window strategy (Schmid and Yang; Steinway et al.). These strategies break input sequences down in smaller, regional windows for dN/dS testing purposes. Although these strategies were initially reported to successfully identify regions of Darwinian selection, these findings have come under scrutiny recently as the sliding window methods have been shown to be sensitive to artifacts leading to incorrect results. Because the number of synonymous sites in a sequence is roughly one third the number of nonsynonymous sites, sliding window strategies are particularly sensitive to changes in dS caused by variations in the number of synonymous sites (Schmid and Yang).

3.1.1 Design Goals

The motivation to implement both versions of the pipeline on Bioextract is to provide the user with the best possible user experience (Lushbough, Jennewein, and Brendel). By implementing the pipeline on Bioextract, we eliminate the need for the user to download and install any portion of the software used on their own work station. This characteristic of Bioextract also allows the user to leverage the computing power of the Bioextract server, which in many cases will be far more powerful than the resources that may be available to them. The system also provides a GUI, as well as illustrated break-down of the pipeline steps, which customizable parameters. Because each step exists separately, it allows for review of output and input, as well as eventual alternate steps, such as providing new alignment algorithms. Finally, the system also allows the user to search for publically available datasets, as well as providing custom input, making data acquisition and storage very flexible.

The main goal for the first pipeline was to first recreate the steps available in other pipelines. By recreating the standard pipeline, but offering the advantages of a web tool and GUI. One frequent criticism of existing pipelines is the negative user experience, as well as the blackbox way most of these pipelines function. By housing the entirety of the pipeline online, we can ensure a consistent user experience, and the modular presentation of the pipeline opens up the user experience and explains what happens in each step. The goal was to also expand upon the statistics of each step, providing for the first time a statistical framework for interpreting results. Specifically, we sought to implement statistical methods for determining error value cutoffs in the final results, as well as parameter tuning functions.

The main goal of the second pipeline was to test and apply a new methodology to the examination of localized regions of selective pressure. Because existing methods have proven troublesome in a number of aspects, we propose a simple solution that involves dividing gene products into regions based on biologically relevant information. Specifically, we have implemented a version of the pipeline that divides gene pairs into conserved protein domains based on information obtained from NCBI's Conserved Domain Database (Marchler-Bauer et al.). By eliminating the need for a theoretical ancestral sequence or the issues inherent in the sliding window strategy, this strategy provides a simple, computationally fast, biologically-informed way of examining selective pressures on subunits of duplicated genes.

3.2 Materials and Methods

3.2.1 Description of Pipeline Implementation

The first pipeline, the recreation of the standard dN/dS discovery pipeline on Bioextract, starts with the duplicate gene discovery step. Since the input for this pipeline is CDS sequences and not genomic sequence, we have chosen to accomplish this step using BLASTn instead of other options. The BLAST parameters for this step do not make use of the full flexibility of the BLAST algorithm in order to lower the complexity for unfamiliar users. Instead, we provide the user with a range of options after the BLAST run has completed, which we feel streamlines the user experience. These options parse the BLAST results for gene pairs based on set stringency options. These options include, but are not necessarily limited to, selecting pairs based on sequence identity, e-value, and sequence match coverage. While the BLAST search does run at a default e-value cutoff of e-10, the results can be parsed at a higher e-value should the user decide to. Once the

user has selected their stringency options, the program then parses genes pairs matching the criteria, and prepares individual FASTA files of those pairs for the remainder of the program. The next step of the program is a two step process, which first converts the CDS sequences to protein sequences using *TRANSEQ* from the EMBOSS software package, then aligns the sequences using MUSCLE (Rice et al.; Edgar). Default MUSCLE parameters are used during this step, but parameters such as gap and mismatch penalties can be altered. Converting the gene pairs to amino acid sequence prior to alignment maintains proper reading frames that MUSCLE could possibly alter by introducing single base gaps in a nucleic acid alignment; these gaps would not be acceptable as input for PAML (Z. Yang). For simplicity, the MUSCLE alignments are output in FASTA format. Before the alignments can be run through PAML, they must be back-translated to their original underlying nucleic acid sequences, which is done using a custom python script. Finally, the PAML package program *codeml* is used to generate dN and dS for each input sequence pair. A custom python script assembles the results from each the PAML output of each gene pair and several R scripts are also included, which generate useful graphic summaries of the output for parameter and error tuning.

The second pipeline begins with a BLASTn run of a user provided CDS fasta file. In this pipeline, rather than provide the user with a series of stringency options for defining gene pairs, only the top hits of each run are used to generate gene pairs. While this is the stringecy method currently implemented on Bioextract, future iterations of the pipeline may relax this stringency by allowing more customizability. The resulting gene pairs are then used as input in a RPS-BLAST run, identifying sequence identity to the NCBI curated Conserved Domain Database (CDD). This step identifies conserved

domains within each gene pair. The ranges of each gene which correspond to a match to the CDD are matched between each gene, and, using those ranges, an amino acid FASTA file for each domain pair is generated using a custom python script and a peptide file input of the original CDS sequences. Having the user provide the protein sequences directly reduces the overall computation requirements of the pipeline, which can be substantial. From here, the pipeline is nearly identical to the first pipeline, with the exception of a tracking file used in the background to ensure that each domain pair remains matched with the duplicate gene pair it originated from. Identified conserved domains (domain pairs) are aligned using MUSCLE and then are back-translated to nucleic acid sequences with a custom pythonl script. The sequences are then run through PAML, and dN/dS values are generated. A custom python script assembles the results arranged by gene, and contain the dN, dS, and dN/dS information for each domain within a duplicated gene pair.

3.2.2 Pipeline Validation Methods

The dN/dS pipeline was validated using a selection of published genomes and dN/dS data. Because the focus of the pipeline is to recreate and streamline the process, we sought only in the first step to demonstrate that the dN/dS values obtained from the pipeline were approximately the same as values obtained from other available pipelines, and compare run times between programs. Genomes used for comparison are soybean, *Medicago t.* and *Lotus j.* (Schmutz et al.; Young et al.; Handberg and Stougaard). Evolutionary distance measures for the testing of the error correction methods were generated from soybean, as well as simulated data from the HKY sequence mutation generator (Refer to Chapter 4).

In order to access the success of the second pipeline, genome data from three well annotated genomes, *Glycine max*, *Arabidopsis thaliana*, and *Medicago trunculata*, were run through the first pipeline up to the PAML report step. These same sequences were then run through the second pipeline. A custom Python script was written to identify sequences where the reported dN/dS ratios were highly varied through the reported domains in the second pipeline. These sequences were then manually checked against the reported whole sequence dN/dS in the original pipeline. Genes which demonstrate the predicted "averaging" effect in which strong selection in certain domains are muted by selection effects in other parts of the gene

3.3 Results and Discussion

3.3.1 Validation and Comparison of Pipelines

There were multiple strategies employed to validate the two analysis pipelines. For the first dN/dS analysis pipeline, which examines genes as a whole, we validated it for internal and external consistency. To show external consistency, we compare the results to those obtained by competing pipelines in order to demonstrate that the results of this pipeline is equivalent to those offered by other lab groups. To validate internal consistency, we show that the results are consistent across multiple runs of the pipeline. We also show internal consistency by demonstrating that the results for some individual gene pairs, when compared with the second pipeline, have dN/dS values that are consistent across the length of the gene and across domain matches found by the second pipeline, since we would not expect all genes to have domains with significantly different dN/dS ratios to the whole gene. This also serves as a validation of the second pipeline, the goal of which was to find genes which demonstrate different strengths of

selection forces across domain comparisons than the gene as a whole. By finding genes that are consistent in selection strength, as well as genes with stronger or weaker dN/dS ratios, we can validate the second pipeline. To do this, gene pairs with data from both pipelines were compared and their alignments visualized. These gene pairs were selected for specific functions which would hopefully yield interesting differences in selection strength.

The first pair analyzed consisted of Glyma.14G099200.1 and Glyma.17G225500.1; their alignment can be found in Figure 1. This pair was selected by searching the soybean gene model annotations for genes related to plant growth and development. The domain identified that was shared between the two was a S-adenosylmethionine-dependent methyltransferase. This domain made up 108 of the 340 and 341 total length of each gene respectively. The dN/dS ratio reported for the whole gene was 0.2219; the dN/dS ratio of the domain was reported to be 0.1982. This internal consistency of dN/dS values found between the two methodologies is a validation of the first pipeline, and is an expected negative result of the second pipeline since these values are too similar to call . The result itself though is interesting in that while the dN/dS value of the whole gene is indicative of strong purifying selection, there does not appear to be any bias in that purifying selection against changes in the domain region of the gene.

The second pair analyzed consisted of Glyma.20g224700.1 and Glyma.10g162300.1; their alignment can be found in Figure 2. This gene pair was selected by identifying genes that were DNA-binding, and related to defense response pathways. The intention in this search was to identify genes that whose functions would

be important enough to warrant stronger selective pressures than those observed in pair one. The shared domain structure was a helix-loop-helix DNA binding domain, which made of 56 amino acids of the total genes 333 and 329 amino acid lengths respectively. The dN/dS ratios returned by the two pipelines were 0.8506 for the full length of the genes, and -1 (corresponding to dN=0 and dS=.1617, specifically) for the helix-loop-helix domain. The critical binding specificity of the helix-loop-helix structure is reflected in the strongly negative dN/dS ratio reflecting strong purifying selection to maintain the integrity of the helix-loop-helix. This is in sharp contrast to the remaining ~80% of the gene length, which evidently has a much higher selective tolerance for non-synonymous substitutions. This gene pair serves as a validation of both pipelines, in that the functionality of the first pipeline is demonstrated while also validating the hypothesis that identifying conserved domains could serve as an alternative to the sliding window strategy of dN/dS analysis.

The third pair analyzed consisted of Glyma.03g183600.1 and Glyma.19g184300.1; their alignment can be found in Figure 3. This third pair of genes was selected to examine the possible outcomes in multidomain proteins. This pair of genes represent an interesting test case, in that the proteins coded by these two genes contain two unique domains that are repeated twice, for a total of four domains in the protein structure. These domains are two transmembrane transport domains, and two ATP-binding domains. The repeating structure of this protein offers an opportunity to observe different selection pressures acting on identical or nearly identical domains within a single protein.

The reported dN/dS values from PAML for the two transmembrane domains were 0.0833 and 0.0505. The values for the ATP-binding domains were 0.0449 and 0.0709. The reported dN/dS ratio for the gene in its entirety was 0.1118. While these ratios are fairly similar, it is important to note that the ratio of the entire gene is considerably higher than any single domain, especially those with the lowest ratio values. While this may seem underwhelming, it is noteworthy because these domains constitute approximately 75% of the total residues in the coding sequence of the gene. This means that the dN values in the non-domain portions of the coding sequence must be considerably higher than the domain-containing portions, indicating that the domain portions experience higher purifying selection than the remaining 25% of the residues. In order to prove this hypothesis, however, one would expect to observe approximately the same values of dS in all 5 cases (the four domains and the total for the gene), and that most of the difference in dN/dS to be attributable to changes in dN. The values of dS from this sample range from approximately .0812 to .135, indicating that these values are likely from the most recent whole genome duplication event soybean underwent. The dN values range from .004 to .013. Despite there being a greater range of values and therefore variation in the dS values, dN has a much higher correlation to the dN/dS ratio (.92) than dS does (.35). This shows that there is a very strong relationship between changes in dN/dS and dN, and no linear relationship between dN/dS and dS.

The fourth pair of genes analyzed consisted of Glyma.01g018700.1 and Glyma.01g096800.1; the alignment of these genes can be found in Figure 4. These genes were selected using an annotation search for growth regulating genes. These genes contain a single conserved domain, the VQ motif domain. Although this motif is

conserved in a variety of plant species, its PFAM description page (PF05678) contains very little functional annotation information. As such, this pair works well as a test case, as we cannot infer a likely dN/dS outcome based on functional information.

The PAML dN/dS output for the full gene and domain portion of gene were .2441 and .0831 respectively. While not dramatically different, there is an appreciably stronger degree of purifying selection in just the domain containing portion of the gene versus the dN/dS ratio for the entire gene. Because the VQ motif is very small, its 28 residues make up less than 10% of the total gene length, it is worth looking at the dN and dS values used to calculate the ratio of both regions. Although we would expect to find identical dS values for both the full gene and the domain region, because the accumulation of synonymous changes occurs at a predictable rate and should be consistent across the length of a gene, in practice the actual estimates of these values can vary. These variations can be attributed to both how the values are estimates, the smaller number of synonymous sites in the domain region will change the calculation of the estimate, as well as the error in the estimate itself. As expected, we found nearly identical dS values (.7230+-0.11 and .5109+-.25 respectively). Conversely, we found very different dN values across the full gene and the domain region, .1765+-.02 and .0425+-.03 respectively. Examining these numbers, we can safely say that the source of the difference in dN/dS is from increased selective pressure against nonsynonymous changes in the domain portion of the coding sequence.

3.4 Conclusions

Detecting and characterizing evolutionary distances between duplicated gene pairs is a complex process that, until now, had not been brought into line with

contemporary Bioinformatics practices. By removing the need for platform dependencies, program installation, and increased speed through parallelization, we have successfully met our goal of providing biologists with a simple, easy to use and understand method of determining evolutionary distances within their sequencing data. Additionally, we have introduced a successor to both the sliding window and lineage based methods of localized selection characterization. While the lineage based methods can detect localized selection pressures, it is constrained by requiring that a high quality ancestral sequence be constructed before the analysis can take place. In contrast, the sliding window method has been shown to be artifacts caused by changes in the number of synonymous sites, and cannot provide trustworthy results. Our method is superior to both of these previous methods, specifically in the way it divides genes into subsections. Because we have opted to divide genes in a way that is biologically informed, we specifically avoid the arbitrary divisions of the sliding window methods while simultaneously eliminating the need for *a priori* knowledge of ancestral sequences. While very small domains could potentially see the same pitfalls found in the sliding window methods, the risk is small and the advantage of using domains in terms of relevance, simplicity, and computational ease cannot be understated. We have shown that the pipeline can detect changes in selective pressure between domains and with respect to the gene as a whole. We observe these changes in a selection of soybean genes of differing functions and selective pressures. Through the careful selection of our methods and the platform on which we have made them available, we have created superior pipelines to others currently available.

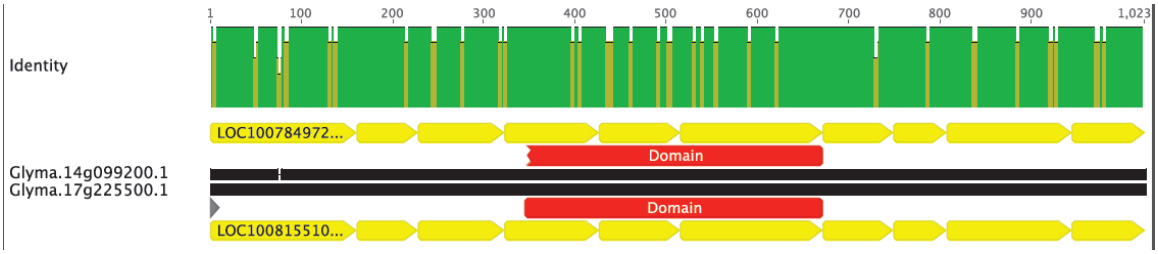

Figure 1. Alignment showing the exon and domain structure of a pair of genes identified by the domain centered pipeline.

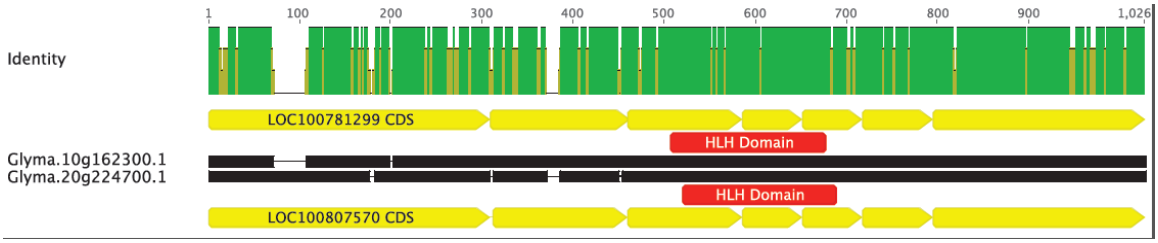

Figure 2. Alignment showing the exon and domain structure of a pair of genes identified by the domain centered pipeline.

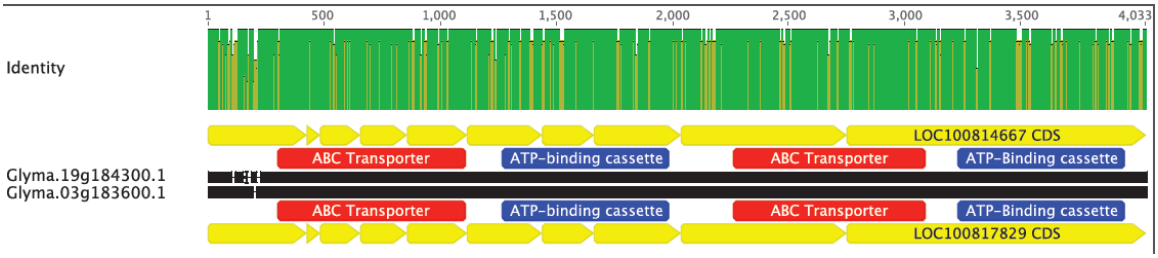

Figure 3. Alignment showing the exon and domain structure of a pair of genes identified by the domain centered pipeline.

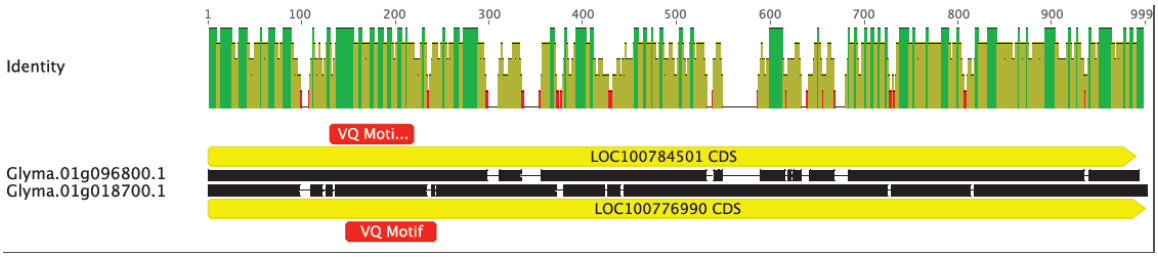

Figure 4. Alignment showing the exon and domain structure of a pair of genes identified by the domain centered pipeline.

CHAPTER 4: SEQUENCE SIMULATION AND DN/DS ERROR CHARACTERIZATION

4.1 Introduction

There are many different methods and pipelines for quantifying the synonymous and nonsynonymous distances between two related sequences. These pipelines and methods have been established and widely used for more than a decade now, and all of these methods suffer from a variety of sources of error. These sources of error are due to both computational issues derived from improper parameter settings, as well as errors in estimation due to the limitations of estimation methods to quantify unobservable mutations. In the case of computational errors, there are a set of risks inherent in each step. For instance, in the initial search step to identify duplicate sequences, through BLAST or other search algorithms, if the search parameters are too stringent, some duplicate genes may not be found (Altschul et al.). However, if the search parameters are not stringent enough, some sequences will be identified that are not proper matches, and the pipeline will proceed with those sequences. These sequences will likely skew the final results and will affect the quality of the final distribution of dS and dN values. Spurious matches are a particular problem in second pipeline step: sequence alignment. While strong, true matches are similar enough that default parameters are sufficient to produce a good quality alignment, sequences that should not have been matched initially will often produce poor quality, gap-filled alignments. In theory, these sorts of problems can be solved by repeated parameter tuning, and while that can be

relatively effective, it is difficult for one significant reason: the true dN and dS is unknowable for real sequences.

In order to address this problem we have designed a sequence simulation program based off the HKY substitution model (Hasegawa, Kishino, and Yano). While sequence mutation programs based off of substitution models are ubiquitous in Bioinformatics software packages, these programs operate in such a way that makes unsuitable for use as simulators to investigate errors in dN and dS estimation. These programs operate as black boxes, taking sequences and mutating them, but do not record each substitution as it happens. As such, the true dN and dS cannot be known and any advantage of having a simulated sequence for investigating error is lost.

Because our research goals demand that we know the true dN and dS, we implemented our own sequence mutation program designed to simulate duplicate sequences undergoing evolution to identify sources of error in PAML-based methods of dN and dS estimation (Z. Yang). The program was designed with the goal of having it be as biologically-informed as possible while still being applicable to our goals. As such, we chose to use the HKY substitution model. Although most of the dN and dS estimation methods implemented in PAML estimate evolutionary distances using substitution models more closely related to K2P and JC69, the Yang and Neilson method, which is the only method implemented in PAML that provides standard errors for dN and dS, uses a substitution model most closely related to HKY. Although we were initially concerned by the risk of overfitting our simulated data by using the same substitution model that the estimates were derived from, we decided that ultimately having the mutation model and estimation model be as closely related as possible would help glean

more information from the results, in addition to HKY being a more biologically true model.

Nucleotide substitution models have grown in complexity since the concept was introduced in the Jukes-Cantor model (Jukes and Cantor). This initial model, more commonly known as the JC model, is the simplest implementation of a nucleotide substitution model as all substitutions occur at a single rate, α. This model was expanded on in the Kimura 2-parameter model, which introduce a second rate β, to differentiate substitutions between purines and pyrimidines(Kimura, "A Simple Method for Estimating Evolutionary Rates of Base Substitutions through Comparative Studies of Nucleotide Sequences"). The next step up in model complexity is the model chosen for our simulation program, the Hasegawa, Kishino, and Yano model or HKY85. This model is a combination of the K2P model and the model by Felsenstein 81 model which introduced the concept of unequal base frequencies (Felsenstein). These additional four parameters correspond to the rate of all substitutions leading to the formation of a given base. An extension of the HKY model is the TN93 model by Tamura and Nei, which further differentiates the α parameter, allowing for unequal transitions rights between purine to purine substitutions and pyrimidine to pyrimidine substitutions (Tamura and Nei).

Given that nucleotide substitution models are one of the backbones on which model phylogenic techniques are built, it is no surprise that many programs that simulate sequence mutations exist. One of these programs, *evolver*, is a part of the PAML package that is used by our pipelines to determine evolutionary distances. *Evolver* is noteworthy because its implementation includes a number of nucleotide, codon, and

amino acid substitution models as options, as well as the option to build trees based on the results. The sequences generated by *evolver* can be used in *codeml* or *yn00*, and thus estimates of dN and dS can be generated for a simulated data set. However useful this would be, it would be infinitely more desirable to know the true dN and dS as well as an estimate. Since these are simulated sequences and each substitution can be recorded as it occurs, this is a reasonable expectation of the program yet the value is not provided.

The specific needs of this program were fairly straight forward. The main goal was to create something that, when implemented, could recreate dN and dS distributions similar to those seen in previously investigated species. For example, we had previously observed a bimodal distribution in soybean indicative of two genome wide duplication events. To accomplish this, the program includes a time parameter which allows the user to build out longer dS distributions using constant rate parameters. For ease of use, the program will also generate identical sequences, so that sequences with dS = 0 can also be simulated (as duplication is an ongoing process and this is observed in real datasets). Another important goal was that the sequences generated be biologically relevant. To accomplish this goal, in addition to mutating the sequences with a complex substitution model, rather than having each sequence be built randomly we have incorporated a codon usage bias table, generated from Wang and Roossinck, such that the sequence composition could more accurately portray real sequences (Wang and Roossinck). These facets of the program will allow us to test and measure what contribution of each step in a dN/dS discovery pipeline has on the overall error of the evolutionary distance measure. We can then leverage that information to both improve those steps, such as standardizing dS error cut offs and performing parameter tuning.

4.2 Materials and Methods

4.2.1 HKY-based Mutation Simulations

The HKY substitution model was implemented in Python. The code itself is comprised of two main functions, a control function, and the HKY model function itself. The HKY function takes 8 parameters, which are given by the main control function and can be specified by the user; default values are provided to the HKY function if not explicitly given by the user when the code is run. These parameters are the alpha and beta rate parameters, which control the transition and transversion mutation rates. Also required are the four nucleotide base frequencies, or the average frequency of each nucleotide in the complete system. These two rate parameters and four base frequencies are used to generate a substitution probability matrix, computed for a single time step. The final parameters are the sequence to be mutated and a time parameter which loops the substitution over a user-defined number of time steps. The sequence to the acted on by the HKY method is generated randomly by another method of the program, using a random length and a codon usage bias file which can be user generated.

After the values for the substitution probability matrix are instantiated, a variable is needed to determine the number of substitutions that happen in a single time step. To determine this number, the HKY program uses a series of random variables generated from a random distribution, and adds these variables together until such time that the variables increase past 1, or the number of substitutions in one time step. These random variables are generated from an exponential distribution, calculated using the rate parameters and base frequencies. This distribution is appropriate to the needs of the HKY model, as higher rates will generate lower values, and allow for more substitutions

in a given time step. If the program passes this check, a random nucleotide from the length of the sequence is chosen for mutation. Random values from an exponential distribution are calculated using the substitution probabilities of each outcome. As higher probabilities will generate lower values, the lowest value from amongst these outcome values is chosen, and the substitution is made. From here, the program calls another method, which reviews the change based on the original sequence, and determines whether or not that change was synonymous or nonsynonymous. Based on the return of this method, the HKY method will either add 1 to a count value for each type of change. Counting each mutation as categorizing it as it occurs allows us to preserve a true dN and dS. If a time value other than 1 was specified, this process will repeat itself until the number of specified time steps, in single time increments, is satisfied.

At the end of this process, the counts of each synonymous and nonsynonymous change are used to calculate the true dN and dS. Because codon changes have occurred, the number of sites in the parent and mutated sequence are different, and thus the mean number of sites between the two sequences are used. Both sequences and the dN and dS values are reported to the control method, and the HKY method is repeated for a new sequence. This process will generate one pair of sequences for each iteration as defined by the user.

Pseudocode:

Main()

Read in user defined parameters, substitute default values if not specified

For the number of pairs defined in iterations parameter

Generate a random sequence -> genseq()

- *Define a sequence length 150-450 nucleotides long*
- *Populate sequence using codon usage bias table (user defined or supplied)*
- *Return sequence*

  Begin HKY substitution method -> hky()
- *Instantiate substitution probability values using rate and base frequency values*
- *Duplicate the sequence variable*
- *Define first rate value using an exponential random variable*
- *While rate value is less than 1:*
    - *Randomly choose a nucleotide within the sequence*
    - *Determine what substitution will occur based on probability values*
    - *Make the substitution in the duplicate sequence*
    - *Using positional information, determine whether or not the substitution was synonymous or nonsynonymous, and record that -> findcodon(), newcodon()*
        - *If substitution led to a premature stop codon, revert the change*
    - *Calculate new rate value, add to previous value*
- *Calculate the number of synonymous and nonsynymous sites in parent and child sequences*
- *Calculate dN and dS*
- *Return both sequences, dN and dS*

  Write sequences out in FASTA format, putting dN and dS in the FASTA header

  Write dS value to dS value file

4.2.2 Investigation of e-value Cutoffs.

To test the effect of parameter tuning, specifically e-value cutoffs during the initial BLAST pairing step, three runs of the pipeline described in Chapter 3 were run using the current soybean CDS sequences. The results from these runs were visualized using R.

4.3 Results and Discussion

4.3.1 Validation of Simulated Sequences

One of the defining and necessary requirements of this program is the ability to recreate data that is an approximation of dS distributions of real systems. The ability of this program to do so was assessed by attempting to recreate the shape of previously described bimodal dS distribution from *Glycine max* (Figure 1)**.** The program described here was able to recreate these dS distributions using a simple set of parameters, a=.5, b=.5. The codon usage table utilized by the program for these testing runs was the Arabidopsis usage table which comes supplied with the program, and therefore was, along with the base frequency parameters used, not appropriate for each system tested. However, for the purposes of testing the ability to recreate shapes of distributions associated with different species, the risk this fault posed was negligible.

In addition to replicating the distribution patterns of different species, the other most critical aspect in the validation of this program is the correspondence between the known dS value generated by the program, and the estimated value obtained by running the sequence through PAML. Although some amount of variation in the estimates is tolerable, given the nature of estimated values, it was critically important that we observe higher correspondence between values at lower dS ranges. Higher disparities at higher

dS values is tolerable and expected, given that a dS of 3 equates to 3 substitutions per synonymous site. At such a value, the number of observable changes is greatly outweighed by unobservable changes, and the estimate of dS by PAML is greatly affected. Table 1 shows a selection of gene pairs derived from the simulator program, the true values of dS and dN, and the estimates of these values from PAML including the standard error. This effect is also observable in Figure 1. From these pairs, we can observe a high correspondence between low dS values as the estimates of those values. At high dS values, we see estimates that are more different than the true values. At a certain point, roughly dS=3, the estimates of begin to be far lower than the actual dS values. We hypothesize that this is caused by a limitation in the mutation program which does not allow for in the inclusion of indels (insertions and deletions), which would almost certainly be present in real gene pairs that distantly related, although we have not been able to exclude the possibility that this is an error in the PAML estimate as well.

4.3.2 Error in dS Measurements

In order to access the magnitude of error in PAML measurements of dS, a simulated set of sequences resembling the distribution seen in legumes was generated and run through PAML programs *codeml* and *yn00*. The results comparing the true dS values, known through the simulator program, and the estimates of those values by the PAML package are seen in Figure 2. We observe that PAML overestimates the true dS value. We investigated this further by binning the dS values in bins of size .12, and taking the mean value of the reported error for dS values in each bin. Figure 3 shows the relationship between dS and its error, a line with slope=1 is included for comparison. We

observe that the error for dS increases as dS increases, and does so at an increasing rate. At certain point, at approximately dS=3, the error values exceed the estimate of dS.

The distinct advantage of having created an analysis pipeline, such as the one described in Chapter 3 of this manuscript, as well as the sequence simulation software described here is the ability to "know" the correct answer, and leverage this knowledge to investigate different sources of error within the analysis pipeline itself. The validation of the pipeline in Chapter 3 is to some extent a fool's errand, as with real data it is effectively impossible to ever know the correct dS and dN for a given gene pair. While it is possible to compare the results of one dS and dN estimation pipeline against another and say one is validated by the other, it would be presumptive to claim either is the correct estimation as the true dN and dS are unknowable. However, the addition of the sequence simulation program described here allows us to not only compare the true dS, the estimate of dS, and the standard error of the estimate, but to also see how the relationship between these three values change in response to changes in the parameters used during each step of the analysis pipeline. This investigation can identify the largest sources of error within each step of the pipeline, and can direct parameter tuning efforts and scripts that can be built back into the pipeline to improve the results and user experience.

For instance, the BLAST pair search step is an ideal candidate to explore how this process can work. Using soybean as an example, we repeated the BLAST search step for different e-values, starting at E-50 up to E-200, going in increments of -10, and limit the pipeline to only report the number of matches each of these values would return. For reference, the current version of the soybean genome has approximately 55,000 genes,

the majority of which are duplicated in two or more copies. Using this knowledge, we can guess a fair expectation of the number of pairs the BLAST step would generate; it would not be fair to expect the user to know this information though. At E-50 however, BLAST reports over 1.4 million pairs. By far, most of these matches are low identity, low coverage matches that are not indicative of true duplicate genes. Increasing the stringency of the e-value cutoff reduces the number of returned pairs in an inverse exponential fashion, as demonstrated in Figure 3. The plateau of this curve occurs at E-180, with ~174,000 pairs. At this point, the only remaining matches within the results are those matches that have 100% coverage and nearly or exactly 100% identify. While it might be tempting to assume that one should only keep the best matches possible, doing so can have a profound effect on the final results of the pipeline. Again, using soybean as an example, we can expect from previous research that the final dS distribution should be bimodal with a peak at approximately dS=0.4 and another, smaller peak at dS=0.8. However, if we eliminate all but the best matches, we hypothesized that this could also eliminate the slightly dissimilar pairs that account for the second peak within the soybean results. If that did occur, without the prior knowledge that the peak exists, we would be forced to conclude soybean only had one whole genome duplication event, a conclusion that is false. A subset of the full spectrum of the BLAST cutoffs and the end effect those cutoff values have on the final dS distribution, most specifically the secondary peak at approximately dS =0.8, can be observed in Figure 4-6. These figures show that at a relative low e-value, both peaks are present but the secondary peak is not as apparent and the density of the first peak is low. When the e-value is increased to 1e-100 in Figure 5, we observe that the density of the primary peak has risen, and the secondary peak is also

more noticeable with a higher density. What appears to be most different is the dS values greater than 1, which have decreased in density significantly. When the e-value is raised further to 1e-175, we see that both peaks are even more apparent than in Figure 5. This is in sharp contrast to our hypothesis that the resolution of the secondary peak would be less clear under very stringent e-values. Although this is initially a very positive reinforcement of the idea of using highly stringent BLAST parameters, the 1e-175 run also introduced a tertiary peak at approximately dS=1.6 that should not exist. So while we in fact showed the opposite of our hypothesis, that a highly stringent e-value would cause us to lose resolution of the secondary peak, the 1e-175 run in Figure 6 showed a spurious peak which would still lead an unseasoned researcher to draw the wrong conclusion from his or her data. Perhaps the most compelling aspect of these results are the graphs of dS against its standard error. In the e=1e-50 data set, the maximal dS standard errors exceed a value of 40 despite a dS cutoff of 5, meaning there are estimates in this dataset with error values more than 8 times the estimate itself. However, when the e-value cutoff is increased to 1e-175, we are eliminating these pairs with the correspondingly high standard error values. The maximum standard error in the 1e-175 run is <10, only double the highest allowed estimates of dS (Figure 7). While this is obviously still a very high error value, it is a significant improvement in the quality and accuracy of the estimates compared to the 1e-50 data set. This result highlights the unequivocal need for appropriately stringent parameters, especially in the gene pairing step. When combined with the results demonstrated in Figure 3, we can conclude that the likely optimal e-value cutoff for soybean and related species would be approximately 1e-100.

4.4 Conclusions

The creation of the gene duplicate simulator is a powerful tool to augment the Pipelines described in Chapter 3. The simulator allows us to generate gene duplicate pairs that are best-case approximates of actual genes that have diverged by using both real world codon usage bias and a relatively complex nucleotide substitution model. The simulator produces gene pairs with known dS and dN values, and can be run multiple times in succession to effectively fill out a distribution that is equivalent in shape and density to a real genome. Leveraging the known dS values of the simulator, we have investigated the true effects of error in the estimates of dS, and can observe how the effects of error skew distributions of dS and dN. We have also used the well-characterized dS distribution of soybean to show how stringency of e-value cutoff parameters in the initial steps of the pipeline can have a drastic effect on the final results and conclusions drawn from those results. These results demonstrate the need for a rigorous, automated parameter tuning and error characterization, and provide a strong initial framework for what that should look like. The user will be best served with an automated implementation of that eschews the default parameter design of other pipeline implementations, as these default parameters are rarely if ever appropriate for all cases.

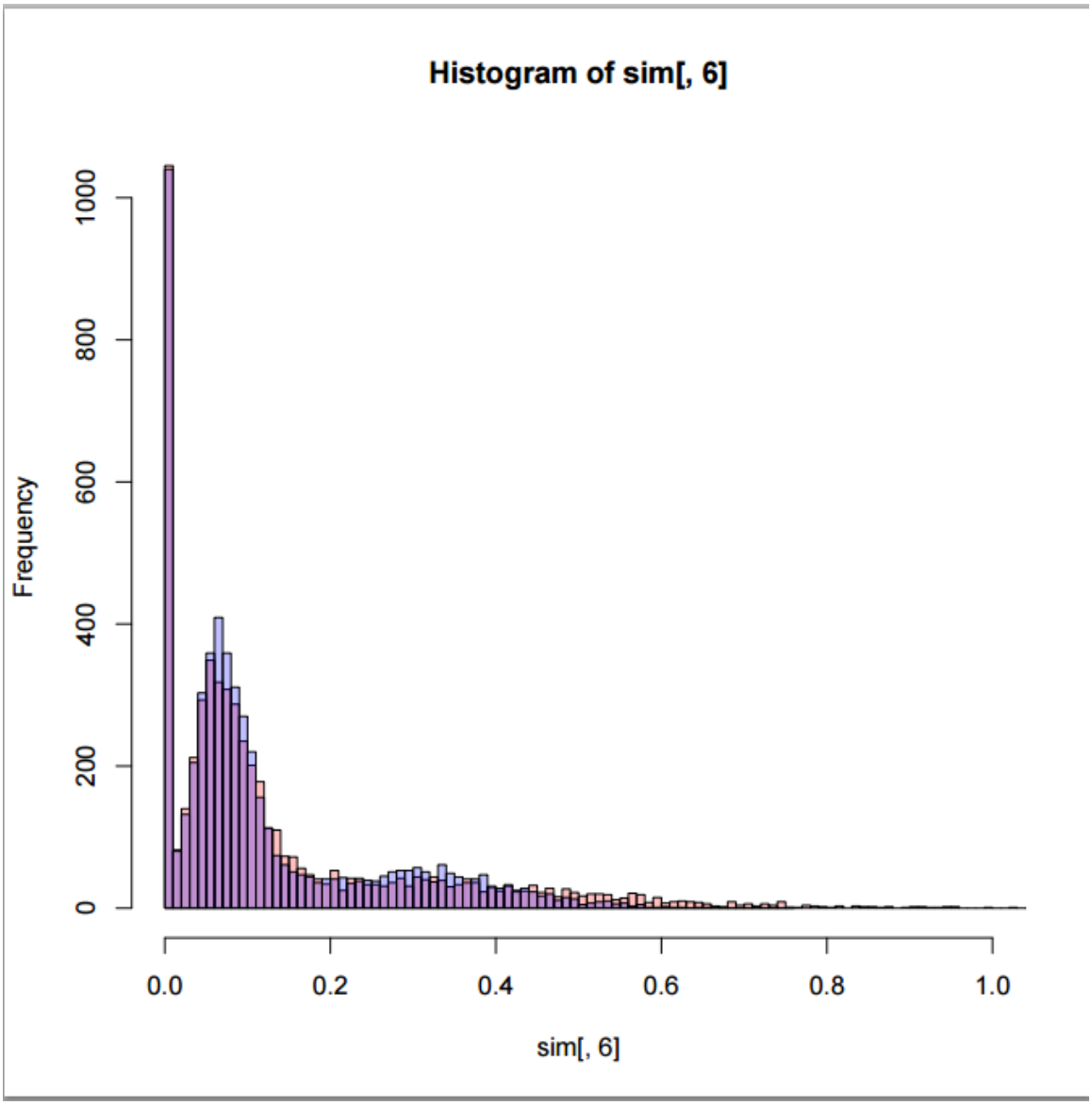

Figure 1. Histogram generated in R of the true dS value (blue) compared with the PAML estimated dS value (red). The high density of the true dS at lower values indicates that PAML overestimates the true dS.

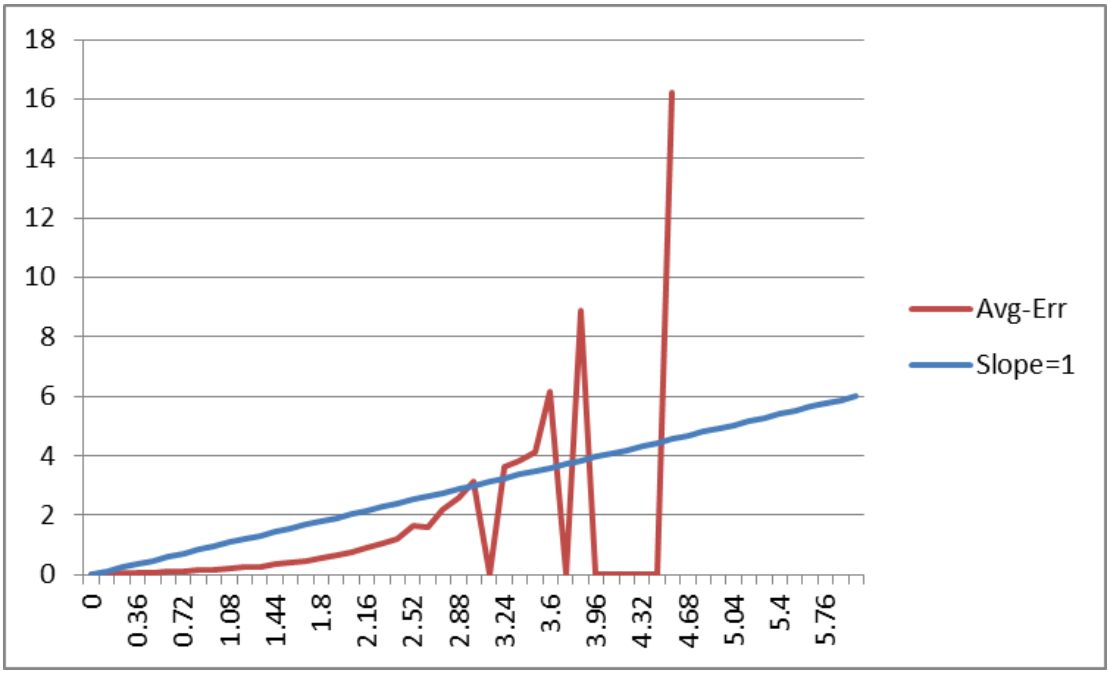

Figure 2. A graph showing the mean standard error of binned dS values in a simulated set of sequences.

file_origin	true_dn	true_ds	ds	ds_se
pair29939	0.9921	4.9379	1.3271	0.3936
pair9729	0.9348	4.9315	1.1688	0.312
pair9800	0.9386	4.9284	1.2844	0.3469
pair2888	0.4186	2.0478	0.9024	0.1751
pair3300	0.4594	2.0477	1.015	0.1992
pair2688	0.4225	2.0466	0.8148	0.146
pair15721	0.0938	0.4201	0.3909	0.0739
pair17446	0.0962	0.4201	0.4456	0.0842
pair17962	0.0918	0.4201	0.2631	0.0564
pair19644	0.0826	0.4201	0.3512	0.0725
pair137	0.0204	0.0658	0.0615	0.0254
pair275	0.0192	0.0654	0.0541	0.0273
pair76	0.0287	0.0637	0.053	0.0309
pair19	0.022	0.059	0.0584	0.0264

Table 1. A selection of simulated gene pairs and the result dS values from PAML.

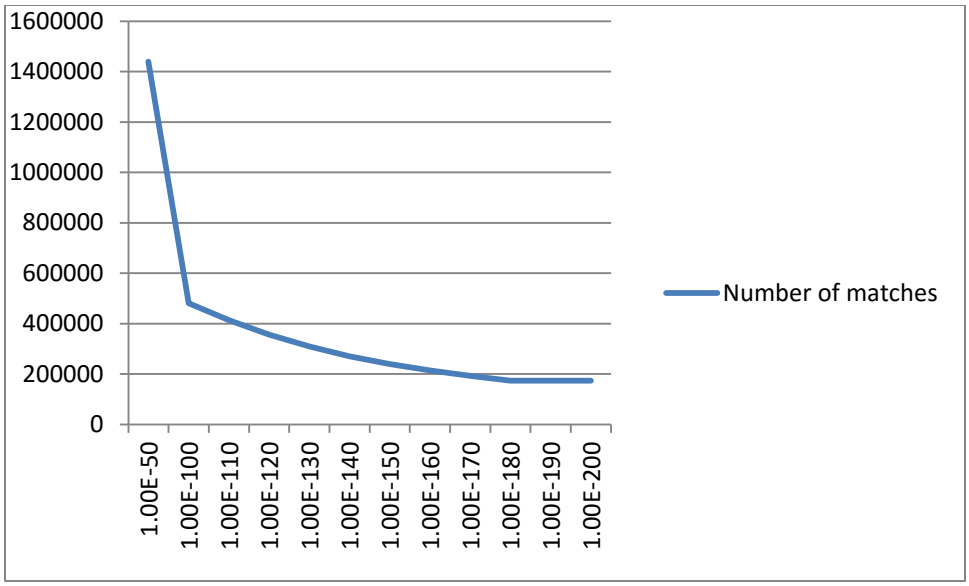

Figure 3. Number of BLAST matches results from different e-value cutoffs using current G. max gene models.

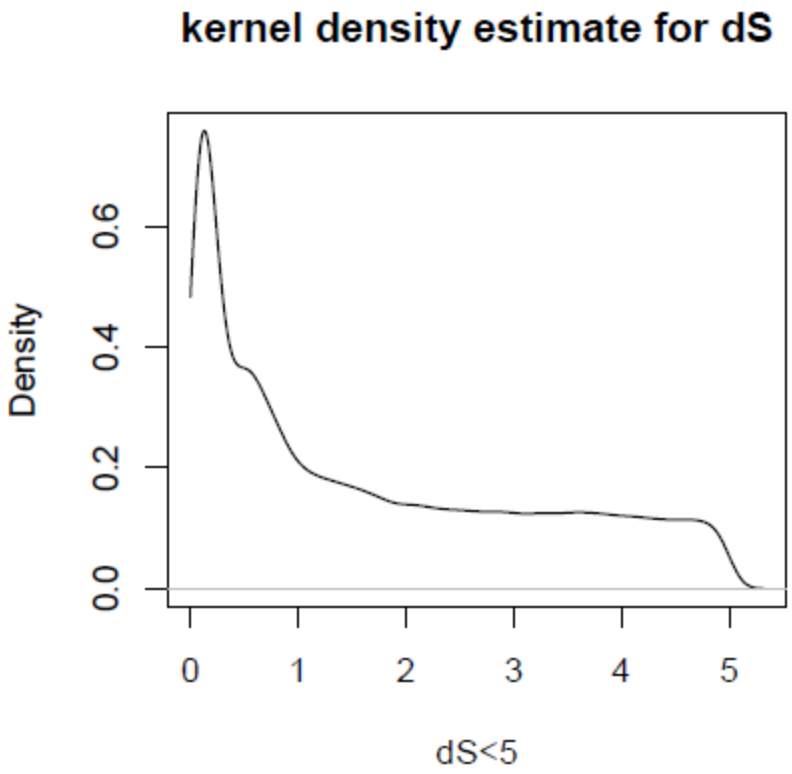

Figure 4. The density plot of dS for G. max CDS using a BLAST e-value cutoff parameter of 1e-50

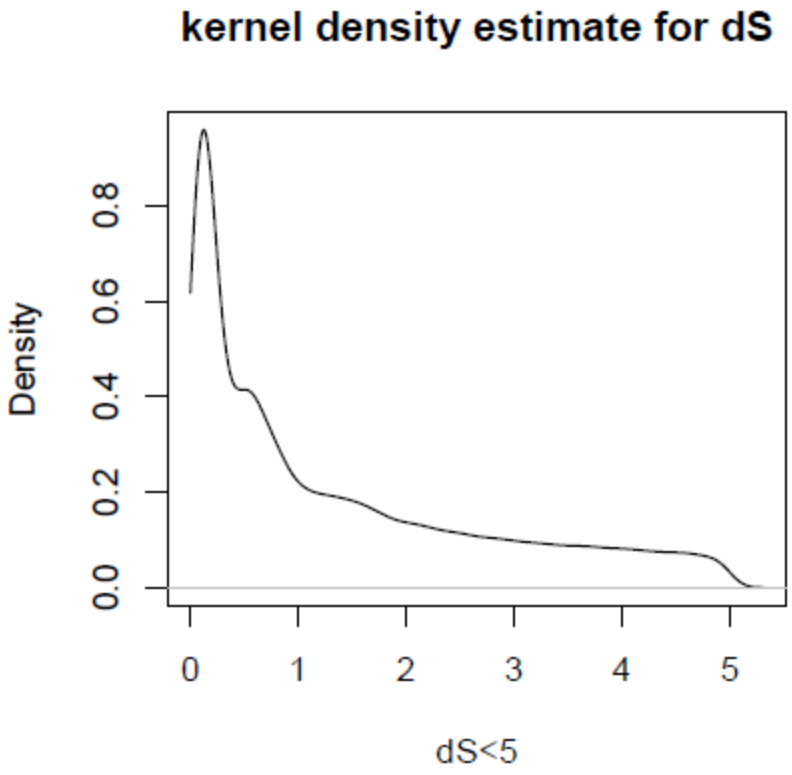

Figure 6. The density plot of dS for G. max CDS using a BLAST e-value cutoff parameter of 1e-100

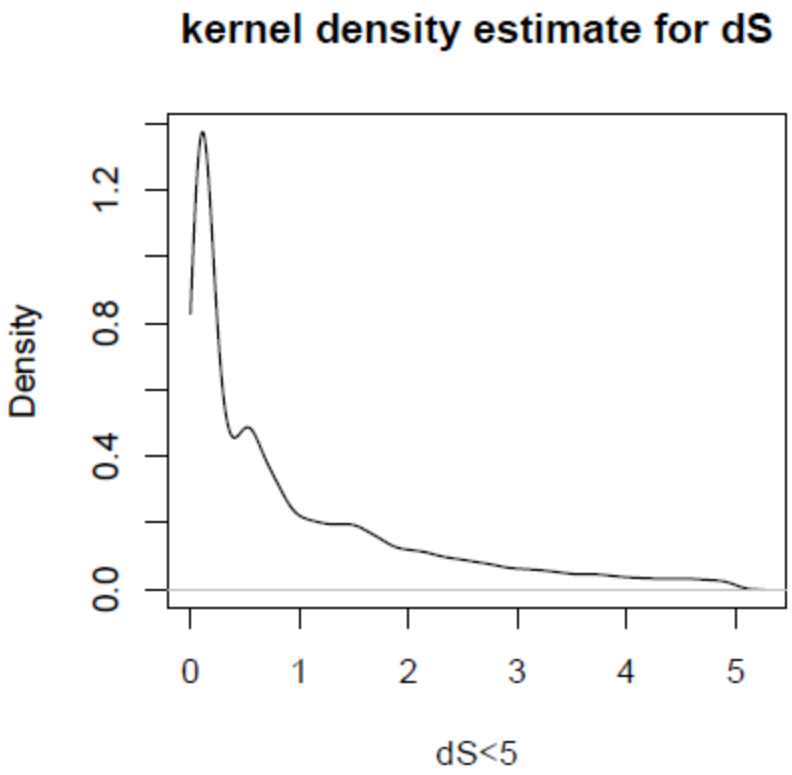

Figure 5. The density plot of dS for G. max CDS using a BLAST e-value cutoff parameter of 1e-175

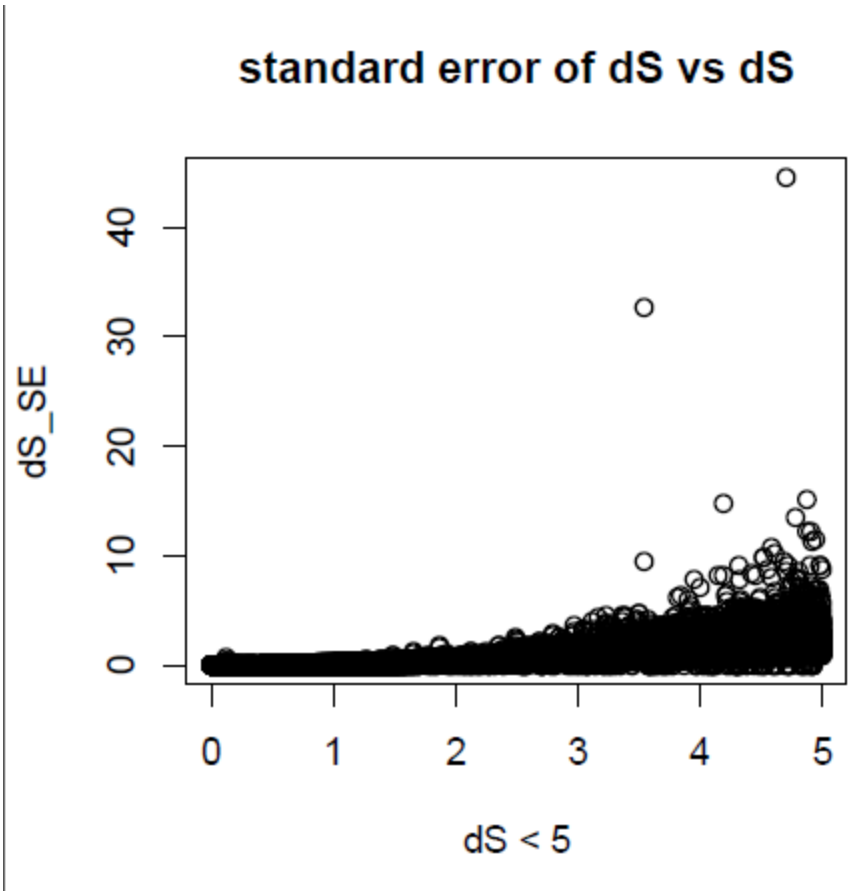

Figure 6. R plot of estimates of dS against their standard errors generated from the dS estimate pipeline using a BLAST cutoff parameter of 1e-50.

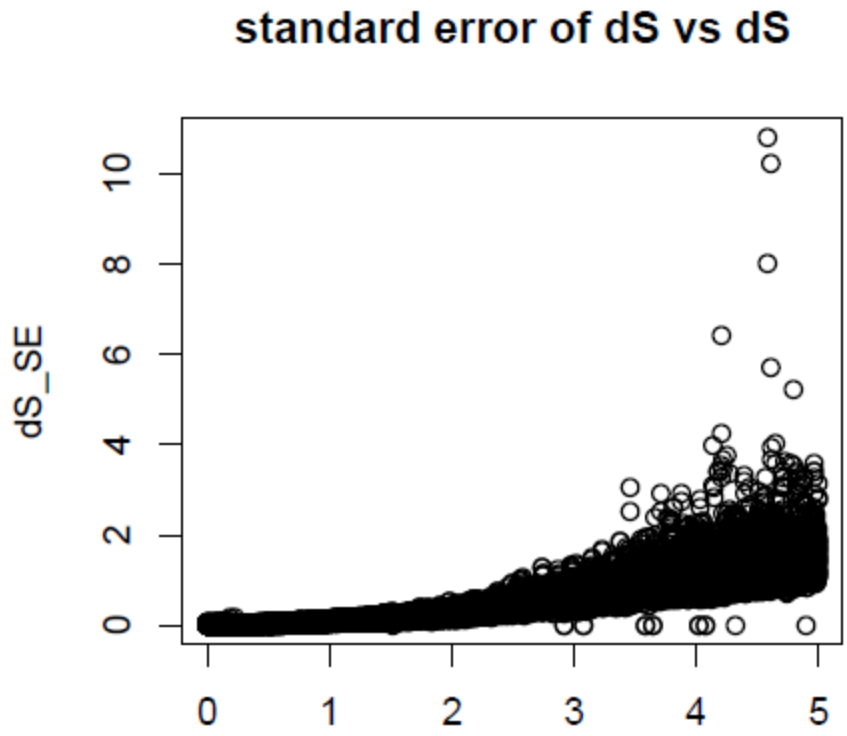

Figure 7. R plot of estimates of dS against their standard errors generated from the dS estimate pipeline using a BLAST cutoff parameter of 1e-175.

REFERENCES

Adams, Keith L., Ryan Percifield, and Jonathan F. Wendel. “Organ-Specific Silencing of Duplicated Genes in a Newly Synthesized Cotton Allotetraploid.” *Genetics* 168.4 (2004): 2217–2226. Print.

Ainsworth, Elizabeth A. et al. “The Effects of Tropospheric Ozone on Net Primary Productivity and Implications for Climate Change.” *Annual Review of Plant Biology* 63.1 (2012): 637–661. *Annual Reviews*. Web.

---. “The Effects of Tropospheric Ozone on Net Primary Productivity and Implications for Climate Change*.” *Annual Review of Plant Biology* 63.1 (2012): 637–661. *Annual Reviews*. Web.

Altschul, Stephen F. et al. “Basic Local Alignment Search Tool.” *Journal of Molecular Biology* 215.3 (1990): 403–410. *ScienceDirect*. Web.

Anderson, H. R., R. G. Derwent, and Mr J. Stedman. “4.7 Air Pollution and Climate Change.” n. pag. Print.

Aw, Jeremy, and Michael J. Kleeman. “Evaluating the First-Order Effect of Intraannual Temperature Variability on Urban Air Pollution.” *Journal of Geophysical Research: Atmospheres (1984–2012)* 108.D12 (2003): n. pag. *Google Scholar*. Web. 27 Aug. 2015.

Bell, Michelle L. et al. “Climate Change, Ambient Ozone, and Health in 50 US Cities.” *Climatic Change* 82.1-2 (2007): 61–76. Print.

Blank, L. W. “A New Type of Forest Decline in Germany.” *Nature* 314 (1985): 311–314. Print.

Booker, Fitzgerald et al. “The Ozone Component of Global Change: Potential Effects on Agricultural and Horticultural Plant Yield, Product Quality and Interactions with Invasive Species.” *Journal of Integrative Plant Biology* 51.4 (2009): 337–351. *Wiley Online Library*. Web.

Bou Jaoudé, Maher et al. “Analysis of the Ozone Effect on Soybean in the Mediterranean Region: II. The Consequences on Growth, Yield and Water Use Efficiency.” *European Journal of Agronomy* 28.4 (2008): 519–525. *ScienceDirect*. Web.

Burkey, Kent O., and Thomas E. Carter Jr. “Foliar Resistance to Ozone Injury in the Genetic Base of U.S. and Canadian Soybean and Prediction of Resistance in Descendent Cultivars Using Coefficient of Parentage.” *Field Crops Research* 111.3 (2009): 207–217. *ScienceDirect*. Web.

Chaudhary, Bhupendra et al. “Reciprocal Silencing, Transcriptional Bias and Functional Divergence of Homeologs in Polyploid Cotton (gossypium).” *Genetics* 182.2 (2009): 503–517. *PubMed*. Web.

Chen, Zhong, and Daniel R. Gallie. “Increasing Tolerance to Ozone by Elevating Foliar Ascorbic Acid Confers Greater Protection against Ozone than Increasing Avoidance.” *Plant Physiology* 138.3 (2005): 1673–1689. Print.

Conklin, P. L., and C. Barth. “Ascorbic Acid, a Familiar Small Molecule Intertwined in the Response of Plants to Ozone, Pathogens, and the Onset of Senescence.” *Plant, Cell & Environment* 27.8 (2004): 959–970. Print.

Crisosto, C. H. et al. “Postharvest Performance Evaluation of Plum (Prunus Salicina Lindel.,Casselman’) Fruit Grown under Three Ozone Concentrations.” *Journal of the American Society for Horticultural Science* 118.4 (1993): 497–502. Print.

Duarte, Jill M. et al. “Expression Pattern Shifts Following Duplication Indicative of Subfunctionalization and Neofunctionalization in Regulatory Genes of Arabidopsis.” *Molecular biology and evolution* 23.2 (2006): 469–478. Print.

Du, Zhou et al. “agriGO: A GO Analysis Toolkit for the Agricultural Community.” *Nucleic acids research* 38.suppl 2 (2010): W64–W70. Print.

Ebi, Kristie L., and Glenn McGregor. “Climate Change, Tropospheric Ozone and Particulate Matter, and Health Impacts.” *Environ Health Perspect* 116.11 (2008): 1449–1455. Print.

Edgar, Robert C. “MUSCLE: Multiple Sequence Alignment with High Accuracy and High Throughput.” *Nucleic acids research* 32.5 (2004): 1792–1797. Print.

Elstner, E. F., W. Osswald, and R. J. Youngman. “Basic Mechanisms of Pigment Bleaching and Loss of Structural Resistance in Spruce (Picea Abies) Needles: Advances in Phytomedical Diagnostics.” *Experientia* 41.5 (1985): 591–597. *link.springer.com*. Web.

EPA, U. *Air Quality Criteria for Ozone and Related Photochemical Oxidants*. EPA/600/R-05/004aF-cF, 2006. Print.

Ernst, D. et al. “Ozone-Induced Changes of mRNA Levels of β-1, 3-Glucanase, Chitinase and ‘pathogenesis-Related’protein 1b in Tobacco Plants.” *Plant molecular biology* 20.4 (1992): 673–682. Print.

Felsenstein, Joseph. “Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach.” *Journal of Molecular Evolution* 17.6 (1981): 368–376. *link.springer.com*. Web.

Fiscus, Edwin L., Fitzgerald L. Booker, and Kent O. Burkey. "Crop Responses to Ozone: Uptake, Modes of Action, Carbon Assimilation and Partitioning." *Plant, Cell & Environment* 28.8 (2005): 997–1011. Print.

Fishman, Jack et al. "An Investigation of Widespread Ozone Damage to the Soybean Crop in the Upper Midwest Determined from Ground-Based and Satellite Measurements." *Atmospheric Environment* 44.18 (2010): 2248–2256. *ScienceDirect*. Web.

Forkel, Renate, and Richard Knoche. "Regional Climate Change and Its Impact on Photooxidant Concentrations in Southern Germany: Simulations with a Coupled Regional Climate-Chemistry Model." *Journal of Geophysical Research: Atmospheres (1984–2012)* 111.D12 (2006): n. pag. *Google Scholar*. Web. 27 Aug. 2015.

Frei, Michael, Juan Pariasca Tanaka, and Matthias Wissuwa. "Genotypic Variation in Tolerance to Elevated Ozone in Rice: Dissection of Distinct Genetic Factors Linked to Tolerance Mechanisms." *Journal of experimental botany* 59.13 (2008): 3741–3752. Print.

Galant, Ashley et al. "From Climate Change to Molecular Response: Redox Proteomics of Ozone-Induced Responses in Soybean." *New Phytologist* 194.1 (2012): 220–229. *Wiley Online Library*. Web.

Garber, Manuel et al. "Computational Methods for Transcriptome Annotation and Quantification Using RNA-Seq." *Nature methods* 8.6 (2011): 469–477. Print.

Goff, L., C. Trapnell, and D. Kelley. "cummeRbund: Analysis, Exploration, Manipulation and Visualization of Cufflinks High-Throughput Sequencing Data." *R package version* 1.0 (2011): n. pag. Print.

Grant, David et al. "SoyBase, the USDA-ARS Soybean Genetics and Genomics Database." *Nucleic acids research* 38.suppl 1 (2010): D843–D846. Print.

Haldane, J. B. S. "The Part Played by Recurrent Mutation in Evolution." *American Naturalist* (1933): 5–19. Print.

Handberg, Kurt, and Jens Stougaard. "Lotus Japonicus, an Autogamous, Diploid Legume Species for Classical and Molecular Genetics." *The Plant Journal* 2.4 (1992): 487–496. Print.

Hasegawa, Masami, Hirohisa Kishino, and Taka-aki Yano. "Dating of the Human-Ape Splitting by a Molecular Clock of Mitochondrial DNA." *Journal of Molecular Evolution* 22.2 (1985): 160–174. *link.springer.com*. Web.

Hayes, F. et al. "Meta-Analysis of the Relative Sensitivity of Semi-Natural Vegetation Species to Ozone." *Environmental Pollution* 146.3 (2007): 754–762. Print.

Heagle, A. S., J. E. Miller, and W. A. Pursley. “Influence of Ozone Stress on Soybean Response to Carbon Dioxide Enrichment: III. Yield and Seed Quality.” *Crop Science* 38.1 (1998): 128. *CrossRef*. Web.

Heck, W. W., and R. Philbeck. “B. and Dunning JA (1978) A Continuous Stirred Tank Reactor (CSTR) System for Exposing Plants to Gaseous Air Contaminants. Principles, Specifications, Construction, and Operation. US Dept. Agr.” *ARS-S-181* n. pag. Print.

Hellgren, Lars I. et al. “In Situ Leaf Lipid Metabolism in Garden Pea (pisum Sativum L.) Exposed to Moderately Enhanced Level of Ozone.” *Journal of experimental botany* 46.2 (1995): 221–230. Print.

Hughes, Austin L., and Masatoshi Nei. “Pattern of Nucleotide Substitution at Major Histocompatibility Complex Class I Loci Reveals Overdominant Selection.” *Nature* 335.6186 (1988): 167–170. Print.

Hwang, R., M. J. Burer, and M. Bell. “Smog in the Forecast: Global Warming, Ozone Pollution and Health in California.” *San Francisco: National Resources Defense Council* (2004): n. pag. Print.

Innan, Hideki, and Fyodor Kondrashov. “The Evolution of Gene Duplications: Classifying and Distinguishing between Models.” *Nature Reviews Genetics* 11.2 (2010): 97–108. Print.

Joshi, Trupti et al. “Soybean Knowledge Base (SoyKB): A Web Resource for Soybean Translational Genomics.” *BMC genomics* 13.Suppl 1 (2012): S15. Print.

Jukes, Thomas H., and Charles R. Cantor. “Evolution of Protein Molecules.” *Mammalian protein metabolism* 3 (1969): 21–132. Print.

Kanofsky, J. R., and P. Sima. “Singlet Oxygen Production from the Reactions of Ozone with Biological Molecules.” *Journal of Biological Chemistry* 266.14 (1991): 9039–9042. Print.

Keen, Noel T., and O. C. Taylor. “Ozone Injury in Soybeans Isoflavonoid Accumulation Is Related to Necrosis.” *Plant physiology* 55.4 (1975): 731–733. Print.

Kettunen, R., K. Overmyer, and J. Kangasjärvi. “The Role of Ethylene in the Formation of Cell Damage during Ozone Stress.” *Biology and Biotechnology of the Plant Hormone Ethylene II*. Springer, 1999. 299–305. Print.

Kimura, Motoo. “A Simple Method for Estimating Evolutionary Rates of Base Substitutions through Comparative Studies of Nucleotide Sequences.” *Journal of Molecular Evolution* 16.2 (1980): 111–120. *link.springer.com*. Web.

---. *The Neutral Theory of Molecular Evolution*. Cambridge University Press, 1984. *Google Scholar*. Web. 27 Aug. 2015.

Kleinjan, Dirk A. et al. “Subfunctionalization of Duplicated Zebrafish pax6 Genes by Cis-Regulatory Divergence.” *PLoS Genet* 4.2 (2008): e29. Print.

Knowlton, Kim et al. “Assessing Ozone-Related Health Impacts under a Changing Climate.” *Environmental Health Perspectives* (2004): 1557–1563. Print.

Kubo, Akihiro et al. “Expression of Arabidopsis Cytosolic Ascorbate Peroxidase Gene in Response to Ozone or Sulfur Dioxide.” *Plant molecular biology* 29.3 (1995): 479–489. Print.

Kumar, Sudhir et al. “MEGA: A Biologist-Centric Software for Evolutionary Analysis of DNA and Protein Sequences.” *Briefings in bioinformatics* 9.4 (2008): 299–306. Print.

Kunst, Ljerka, and Lacey Samuels. “Plant Cuticles Shine: Advances in Wax Biosynthesis and Export.” *Current Opinion in Plant Biology* 12.6 (2009): 721–727. *ScienceDirect*. Web.

Langmead, Ben et al. “Ultrafast and Memory-Efficient Alignment of Short DNA Sequences to the Human Genome.” *Genome biol* 10.3 (2009): R25. Print.

Le Provost, Grégoire et al. “Soil Water Stress Affects Both Cuticular Wax Content and Cuticle-Related Gene Expression in Young Saplings of Maritime Pine (Pinus Pinaster Ait).” *BMC plant biology* 13.1 (2013): 95. Print.

Liu, Baohui et al. “Genetic Redundancy in Soybean Photoresponses Associated with Duplication of the Phytochrome A Gene.” *Genetics* 180.2 (2008): 995–1007. Print.

Li, Wen-Hsiung. “Unbiased Estimation of the Rates of Synonymous and Nonsynonymous Substitution.” *Journal of molecular evolution* 36.1 (1993): 96–99. Print.

Li, Wen-Hsiung, Chung-I. Wu, and Chi-Cheng Luo. “A New Method for Estimating Synonymous and Nonsynonymous Rates of Nucleotide Substitution Considering the Relative Likelihood of Nucleotide and Codon Changes.” *Molecular biology and evolution* 2.2 (1985): 150–174. Print.

Lushbough, Carol M., Douglas M. Jennewein, and Volker P. Brendel. “The BioExtract Server: A Web-Based Bioinformatic Workflow Platform.” *Nucleic acids research* 39.suppl 2 (2011): W528–W532. Print.

Lynch, Michael, and John S. Conery. “The Evolutionary Fate and Consequences of Duplicate Genes.” *Science* 290.5494 (2000): 1151–1155. Print.

Maccarrone, Mauro, Gerrit A. Veldink, and Johannes FG Vliegenthart. “Thermal Injury and Ozone Stress Affect Soybean Lipoxygenases Expression.” *FEBS letters* 309.3 (1992): 225–230. Print.

Marchler-Bauer, Aron et al. “CDD: A Conserved Domain Database for the Functional Annotation of Proteins.” *Nucleic acids research* 39.suppl 1 (2011): D225–D229. Print.

Marioni, John C. et al. “RNA-Seq: An Assessment of Technical Reproducibility and Comparison with Gene Expression Arrays.” *Genome research* 18.9 (2008): 1509–1517. Print.

Mauzerall, Denise L., and Xiaoping Wang. “PROTECTING AGRICULTURAL CROPS FROM THE EFFECTS OF TROPOSPHERIC OZONE EXPOSURE: Reconciling Science and Standard Setting in the United States, Europe, and Asia.” *Annual Review of Energy and the Environment* 26.1 (2001): 237–268. *Annual Reviews*. Web.

McGettigan, Paul A. “Transcriptomics in the RNA-Seq Era.” *Current opinion in chemical biology* 17.1 (2013): 4–11. Print.

Mills, G. et al. “A Synthesis of AOT40-Based Response Functions and Critical Levels of Ozone for Agricultural and Horticultural Crops.” *Atmospheric Environment* 41.12 (2007): 2630–2643. *ScienceDirect*. Web.

Mittler, Ron. “Oxidative Stress, Antioxidants and Stress Tolerance.” *Trends in plant science* 7.9 (2002): 405–410. Print.

Morgan, Patrick B. et al. “Season-Long Elevation of Ozone Concentration to Projected 2050 Levels under Fully Open-Air Conditions Substantially Decreases the Growth and Production of Soybean.” *New Phytologist* 170.2 (2006): 333–343. *Wiley Online Library*. Web.

Nadeau, Joseph H., and David Sankoff. “Comparable Rates of Gene Loss and Functional Divergence after Genome Duplications Early in Vertebrate Evolution.” *Genetics* 147.3 (1997): 1259–1266. Print.

Nei, Masatoshi, and Takashi Gojobori. “Simple Methods for Estimating the Numbers of Synonymous and Nonsynonymous Nucleotide Substitutions.” *Molecular biology and evolution* 3.5 (1986): 418–426. Print.

Norton, J. S., A. J. Charig, and DEMORANV IE. “EFFECT OF OZONE ON STORAGE OF CRANBERRIES.” *Proceedings of the American Society for Horticultural Science*. Vol. 93. AMER SOC HORTICULTURAL SCIENCE 701 NORTH SAINT ASAPH STREET, ALEXANDRIA, VA 22314-1998, 1968. 792. Print.

Ohno, Susumu, and others. *Evolution by Gene Duplication.* London: George Alien & Unwin Ltd. Berlin, Heidelberg and New York: Springer-Verlag., 1970. Print.

Organization, World Health, and others. “Air Quality Guidelines for Europe.” (2000): n. pag. *Google Scholar*. Web. 27 Aug. 2015.

Oshlack, Alicia, Matthew J. Wakefield, and others. “Transcript Length Bias in RNA-Seq Data Confounds Systems Biology.” *Biol Direct* 4.1 (2009): 14. Print.

Overmyer, Kirk, Mikael Brosché, and Jaakko Kangasjärvi. “Reactive Oxygen Species and Hormonal Control of Cell Death.” *Trends in plant science* 8.7 (2003): 335–342. Print.

Patz, Jonathan A. et al. “Impact of Regional Climate Change on Human Health.” *Nature* 438.7066 (2005): 310–317. Print.

Pepke, Shirley, Barbara Wold, and Ali Mortazavi. “Computation for ChIP-Seq and RNA-Seq Studies.” *Nature methods* 6 (2009): S22–S32. Print.

Pond, Sergei L. Kosakovsky, and Spencer V. Muse. “HyPhy: Hypothesis Testing Using Phylogenies.” *Statistical Methods in Molecular Evolution*. Springer, 2005. 125–181. *Google Scholar*. Web. 28 Aug. 2015.

Prather, M. et al. *Atmospheric Chemistry and Greenhouse Gases, Climate Change 2001: The Scientific Basis, Contribution of Working Group I to the Third Assessment Report of the Intergovernmental Panel on Climate Change JT Houghton, et Al., 239–287*. Cambridge Univ. Press, New York, 2001. Print.

Purcell, L.C., and Specht, J.S. *Soybeans: Improvement, Production and Uses.* Ed. H. R. Boerma and J. E. Specht. 3rd ed. N.p., 2004. Print.

Pye, John M. “Impact of Ozone on the Growth and Yield of Trees: A Review.” *Journal of environmental quality* 17.3 (1988): 347–360. Print.

Qian, Wenfeng et al. “Maintenance of Duplicate Genes and Their Functional Redundancy by Reduced Expression.” *Trends in Genetics* 26.10 (2010): 425–430. Print.

Quail, Michael A. et al. “A Tale of Three next Generation Sequencing Platforms: Comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq Sequencers.” *BMC genomics* 13.1 (2012): 341. Print.

Rao, Mulpuri V., and Keith R. Davis. “Ozone-Induced Cell Death Occurs via Two Distinct Mechanisms in Arabidopsis: The Role of Salicylic Acid.” *The Plant Journal* 17.6 (1999): 603–614. Print.

Rao, Mulpuri V., Hyung-Il Lee, and Keith R. Davis. “Ozone-Induced Ethylene Production Is Dependent on Salicylic Acid, and Both Salicylic Acid and Ethylene Act in Concert to Regulate Ozone-Induced Cell Death.” *The Plant Journal: For Cell and Molecular Biology* 32.4 (2002): 447–456. Print.

Reddy, G. et al. “Changes in Ethylene and Polyamines in Relation to mRNA Levels of the Large and Small Subunits of Ribulose Bisphosphate Carboxylase/oxygenase

in Ozone-Stressed Potato Foliage." *Plant, Cell & Environment* 16.7 (1993): 819–826. Print.

Reich, Peter B. "Quantifying Plant Response to Ozone: A Unifying Theory." *Tree physiology* 3.1 (1987): 63–91. Print.

Reich, Peter B., Anna W. Schoettle, and Robert G. Amundson. "Effects of Low Concentrations of O3, Leaf Age and Water Stress on Leaf Diffusive Conductance and Water Use Efficiency in Soybean." *Physiologia Plantarum* 63.1 (1985): 58–64. *Wiley Online Library*. Web.

Rice, Peter et al. "EMBOSS: The European Molecular Biology Open Software Suite." *Trends in genetics* 16.6 (2000): 276–277. Print.

Rogers, H.H. et al. "Measuring Air Pollutant Uptake by Plants: A Direct Kinetic Technique." *Journal of the Air Pollution Control Association* 27.12 (1977): 1192–1197. *Taylor and Francis+NEJM*. Web.

Roulin, Anne et al. "The Fate of Duplicated Genes in a Polyploid Plant Genome." *The Plant Journal* 73.1 (2013): 143–153. Print.

Runeckles, V. C., and B. I. Chevone. "Crop Responses to Ozone." *Surface level ozone exposures and their effects on vegetation* (1992): 189–270. Print.

Schmid, Karl, and Ziheng Yang. "The Trouble with Sliding Windows and the Selective Pressure in BRCA1." *PLoS One* 3.11 (2008): e3746. Print.

Schmutz, Jeremy et al. "Genome Sequence of the Palaeopolyploid Soybean." *nature* 463.7278 (2010): 178–183. Print.

Seinfeld, John H., and Spyros N. Pandis. *Atmospheric Chemistry and Physics: From Air Pollution to Climate Change*. John Wiley & Sons, 2012. *Google Scholar*. Web. 27 Aug. 2015.

Sheehan, Moira J. et al. "Subfunctionalization of PhyB1 and PhyB2 in the Control of Seedling and Mature Plant Traits in Maize." *The Plant Journal* 49.2 (2007): 338–353. Print.

Shen, Yanting et al. "Global Dissection of Alternative Splicing in Paleopolyploid Soybean." *The Plant Cell Online* 26.3 (2014): 996–1008. *www.plantcell.org*. Web.

Shepherd, Tom, and D. Wynne Griffiths. "The Effects of Stress on Plant Cuticular Waxes." *New Phytologist* 171.3 (2006): 469–499. Print.

Singh, E., S. Tiwari, and M. Agrawal. "Effects of Elevated Ozone on Photosynthesis and Stomatal Conductance of Two Soybean Varieties: A Case Study to Assess

Impacts of One Component of Predicted Global Climate Change." *Plant Biology* 11.s1 (2009): 101–108. Print.

Singh, Guriqbal. *The Soybean: Botany, Production and Uses*. CABI, 2010. Print.

Steinway, Steven N. et al. "JCoDA: A Tool for Detecting Evolutionary Selection." *BMC bioinformatics* 11.1 (2010): 284. Print.

Tamura, K., and M. Nei. "Estimation of the Number of Nucleotide Substitutions in the Control Region of Mitochondrial DNA in Humans and Chimpanzees." *Molecular Biology and Evolution* 10.3 (1993): 512–526. Print.

Tong, Daniel et al. "The Use of Air Quality Forecasts to Assess Impacts of Air Pollution on Crops: Methodology and Case Study." *Atmospheric Environment* 41.38 (2007): 8772–8784. *ScienceDirect*. Web.

Trapnell, Cole et al. "Differential Gene and Transcript Expression Analysis of RNA-Seq Experiments with TopHat and Cufflinks." *Nature protocols* 7.3 (2012): 562–578. Print.

Trapnell, Cole, Lior Pachter, and Steven L. Salzberg. "TopHat: Discovering Splice Junctions with RNA-Seq." *Bioinformatics* 25.9 (2009): 1105–1111. Print.

Vahala, Jorma et al. "Differential Effects of Elevated Ozone on Two Hybrid Aspen Genotypes Predisposed to Chronic Ozone Fumigation. Role of Ethylene and Salicylic Acid." *Plant Physiology* 132.1 (2003): 196–205. *www.plantphysiol.org*. Web.

Varshney, Rajeev K. et al. "Draft Genome Sequence of Pigeonpea (Cajanus Cajan), an Orphan Legume Crop of Resource-Poor Farmers." *Nature biotechnology* 30.1 (2012): 83–89. Print.

Vingarzan, Roxanne. "A Review of Surface Ozone Background Levels and Trends." *Atmospheric Environment* 38.21 (2004): 3431–3442. Print.

Wagner, Andreas. "The Fate of Duplicated Genes: Loss or New Function?" *BioEssays* 20.10 (1998): 785–788. Print.

Walsh, J. Bruce. "How Often Do Duplicated Genes Evolve New Functions?" *Genetics* 139.1 (1995): 421–428. Print.

Wang, Liangjiang, and Marilyn J. Roossinck. "Comparative Analysis of Expressed Sequences Reveals a Conserved Pattern of Optimal Codon Usage in Plants." *Plant Molecular Biology* 61.4-5 (2006): 699–710. *link.springer.com*. Web.

Wang, Liguo, Shengqin Wang, and Wei Li. "RSeQC: Quality Control of RNA-Seq Experiments." *Bioinformatics* 28.16 (2012): 2184–2185. Print.

Wang, Xiaoping, and Denise L. Mauzerall. “Characterizing Distributions of Surface Ozone and Its Impact on Grain Production in China, Japan and South Korea: 1990 and 2020.” *Atmospheric Environment* 38.26 (2004): 4383–4402. *ScienceDirect*. Web.

West, J. Jason, Sophie Szopa, and Didier A. Hauglustaine. “Human Mortality Effects of Future Concentrations of Tropospheric Ozone.” *Comptes Rendus Geoscience* 339.11 (2007): 775–783. Print.

Wittig, Victoria E. et al. “Quantifying the Impact of Current and Future Tropospheric Ozone on Tree Biomass, Growth, Physiology and Biochemistry: A Quantitative Meta-Analysis.” *Global Change Biology* 15.2 (2009): 396–424. Print.

Yang, Da-Hai et al. “Silencing Nicotiana Attenuata Calcium-Dependent Protein Kinases, CDPK4 and CDPK5, Strongly up-Regulates Wound-and Herbivory-Induced Jasmonic Acid Accumulations.” *Plant physiology* 159.4 (2012): 1591–1607. Print.

Yang, Z. “PAML: A Program Package for Phylogenetic Analysis by Maximum Likelihood.” *Computer applications in the biosciences: CABIOS* 13.5 (1997): 555–556. Print.

Yang, Ziheng. *Computational Molecular Evolution*. Vol. 21. Oxford University Press Oxford, 2006. *Google Scholar*. Web. 28 Aug. 2015.

Yang, Ziheng, and Joseph P. Bielawski. “Statistical Methods for Detecting Molecular Adaptation.” *Trends in Ecology & Evolution* 15.12 (2000): 496–503. *ScienceDirect*. Web.

Yang, Ziheng, and Rasmus Nielsen. “Estimating Synonymous and Nonsynonymous Substitution Rates under Realistic Evolutionary Models.” *Molecular biology and evolution* 17.1 (2000): 32–43. Print.

Yoshida, Seiji et al. “Ethylene and Salicylic Acid Control Glutathione Biosynthesis in Ozone-Exposed Arabidopsis Thaliana.” *Physiologia Plantarum* 136.3 (2009): 284–298. *Wiley Online Library*. Web.

Young, Nevin D. et al. “The Medicago Genome Provides Insight into the Evolution of Rhizobial Symbioses.” *Nature* 480.7378 (2011): 520–524. Print.

Zhao, Tian Hong et al. “Effects of Reactive Oxygen Species Metabolic System on Soybean (Glycine Max) under Exogenous Chitosan to Ozone Stress.” *Bulletin of environmental contamination and toxicology* 85.1 (2010): 59–63. Print.

APPENDIX A: ADDITION DATA AND CODE

A.1: Ozone

A.1.1 Wax biosynthesis genes

Supplementary Table 1. Glyma identifier for genes implicated in the wax biosynthesis pathway and subsequent expression data

Gene	T1-v1	T1-v2	T1-fc	T1-sig	T2-v1	T2-v2	T2-fc	T2-sig
Glyma.09G226200	0.788152	0.218539	-1.85058	no	0.411709	0.335942	-0.293414	no
Glyma.11G185200	0.630978	0.0110401	-5.83677	no	0.336494	0.015133	-4.47481	no
Glyma.05G152500	11.9788	46.585	1.95939	no	11.2293	50.8866	2.18002	no
Glyma.02G086100	1.86468	8.31019	2.15595	no	1.90316	20.2206	3.40936	no
Glyma.07G000300	17.9188	36.4446	1.02423	no	19.9494	41.5757	1.0594	no
Glyma.04G057800	96.4737	60.5939	-0.670962	no	147.108	36.1393	-2.02523	no
Glyma.14G074300	4.5606	4.11235	-0.149262	no	2.52689	5.18577	1.0372	no
Glyma.07G116200	0	0	0	no	0.0262647	0	#NAME?	no
Glyma.06G058500	23.3559	17.938	-0.380764	no	24.8295	7.70838	-1.68755	no
Glyma.15G048400	0.0296936	0.00724567	-2.03496	no	0.0167917	0.00921544	-0.865622	no
Glyma.01G225200	39.3581	64.6087	0.715068	no	55.1016	95.7602	0.797332	no
Glyma.08G110000	2.15258	1.8752	-0.199026	no	1.82724	5.63594	1.62499	no
Glyma.04G149300	0.566039	0.530661	-0.0931106	no	0.729061	2.89915	1.99152	no
Glyma.02G266400	0.311012	0	#NAME?	no	0.442042	0	#NAME?	no
Glyma.18G086700	0	0	0	no	0	0	0	no
Glyma.02G169100	0	0	0	no	0	0.0123902	inf	no
Glyma.12G183300	0.0290734	0.0129305	-1.16892	no	0.0410413	0	#NAME?	no
Glyma.15G114700	0.987458	0.307265	-1.68424	no	0.740501	1.43706	0.956552	no
Glyma.02G296000	3.89505	0.976855	-1.99543	no	2.07776	2.5017	0.267883	no
Glyma.20G115500	57.8633	33.7585	-0.7774	no	75.4998	8.75867	-3.10769	no
Glyma.05G216600	1.22036	2.41371	0.983946	no	1.57242	2.43306	0.629782	no
Glyma.06G097300	12.673	13.3862	0.0789852	no	9.51037	14.0994	0.568064	no
Glyma.06G161700	4.93811	7.94513	0.686111	no	5.2856	11.3687	1.10493	no
Glyma.01G028900	20.7373	19.7609	-0.0695727	no	23.8688	30.9758	0.376013	no
Glyma.10G092900	0	0	0	no	0	0	0	no
Glyma.08G323400	0	0	0	no	0	0	0	no
Glyma.15G008600	0	0	0	no	0	0.0156029	inf	no
Glyma.11G185300	13.0503	5.13028	-1.34698	no	14.8684	4.55385	-1.70709	no
Glyma.05G011100	8.26667	8.15927	-0.0188676	no	10.475	13.0313	0.315031	no
Glyma.09G196400	1.18194	2.37854	1.00892	no	3.78815	8.52727	1.17059	no
Glyma.03G101800	0.0098907	0	#NAME?	no	0.0192072	0	#NAME?	no
Glyma.08G261100	2.54078	0.663111	-1.93795	no	2.09017	0.328037	-2.67169	no
Glyma.05G003200	37.9284	69.3635	0.870898	no	66.109	34.7226	-0.928972	no
Glyma.07G031200	0	0	0	no	0	0	0	no
Glyma.10G010800	35.6515	22.0334	-0.694271	no	54.2776	19.9844	-1.44148	no
Glyma.03G221400	1.75305	0.218989	-3.00094	no	0.536501	0.0653988	-3.03624	no
Glyma.13G317700	6.42643	10.0792	0.649297	no	23.7519	23.3407	-0.0251909	no
Glyma.02G273300	19.0428	16.449	-0.211241	no	23.6122	15.1374	-0.641416	no
Glyma.11G017900	16.0476	19.0712	0.249034	no	23.5408	6.3284	-1.89525	no
Glyma.07G171400	0.25899	0.028762	-3.17066	no	0.187585	0.192655	0.0384733	no
Glyma.19G046000	65.8345	4.00298	-4.0397	yes	64.5362	0.103597	-9.28299	no
Glyma.11G185000	0.0542959	0	#NAME?	no	0	0.0229428	inf	no
Glyma.19G089500	0	0	0	no	0.0408226	0.235506	2.52832	no
Glyma.04G147500	0.58229	0.228289	-1.35087	no	0.0932097	0.0605553	-0.622226	no
Glyma.19G003600	0	0	0	no	0	0	0	no
Glyma.17G251000	5.81136	7.04446	0.277612	no	5.65652	10.0246	0.825558	no

Supplementary Table 1 (Continued). Glyma identifier for genes implicated in the wax biosynthesis pathway and subsequent expression data

Glyma.17G069100	292.089	146.309	-0.997388	no	318.409	84.9566	-1.90608	no
Glyma.07G121500	0.871749	0.191053	-2.18994	no	0.405507	0.145988	-1.47388	no
Glyma.06G221800	0.128093	0.0180547	-2.82675	no	0.0130467	0	#NAME?	no
Glyma.03G101300	0	0	0	no	0	0	0	no
Glyma.02G103800	14.4577	18.3853	0.346711	no	21.16	24.5207	0.212657	no
Glyma.18G181200	5.86705	8.77921	0.581456	no	9.45405	11.7071	0.308378	no
Glyma.07G132300	12.1631	20.0368	0.72014	no	17.2235	27.7258	0.686849	no
Glyma.11G245600	15.6356	8.22298	-0.927098	no	12.7835	9.1001	-0.490324	no
Glyma.09G010200	2.078	0.484806	-2.09972	no	0.735861	1.07686	0.54932	no
Glyma.03G101700	0.445335	0.0136062	-5.03255	no	0.351344	0.132943	-1.40208	no
Glyma.13G091200	126.176	51.3397	-1.29729	no	82.6494	22.3002	-1.88994	no
Glyma.05G087900	0	0	0	no	0	0	0	no
Glyma.18G257900	0.0407187	0	#NAME?	no	0.0717013	0.252714	1.81743	no
Glyma.10G241700	0.0077048	0.0381307	2.30712	no	0.00785736	0.165263	4.39458	no
Glyma.10G274400	168.38	101.762	-0.726522	no	169.062	64.0343	-1.40064	no
Glyma.18G089000	0	0	0	no	0	0	0	no
Glyma.17G118700	20.7403	12.0291	-0.785914	no	19.8858	13.51	-0.557707	no
Glyma.05G003000	0.105228	0	#NAME?	no	0.0116993	0	#NAME?	no
Glyma.07G215200	16.6541	18.5433	0.155022	no	20.9605	24.0082	0.195857	no
Glyma.12G087400	2.55591	0.791666	-1.69087	no	0.46804	1.10291	1.23661	no
Glyma.03G092900	0	0	0	no	0	0	0	no
Glyma.20G007900	40.4809	28.2604	-0.518459	no	31.6964	35.4055	0.159653	no
Glyma.18G244400	10.4858	14.4227	0.459907	no	13.314	14.624	0.135396	no
Glyma.14G043500	17.8235	9.15573	-0.961034	no	18.8096	12.0062	-0.647689	no
Glyma.11G015300	5.66709	4.46488	-0.343986	no	6.27657	5.97088	-0.0720334	no
Glyma.05G198500	7.58482	9.67487	0.351128	no	8.28256	10.0046	0.272517	no
Glyma.08G323100	0.596644	0.171642	-1.79747	no	0.531171	0.325848	-0.704977	no
Glyma.13G166700	0.242491	0.0723107	-1.74565	no	0.0894808	0.0207583	-2.10789	no
Glyma.13G317600	29.0508	2.7075	-3.42354	yes	109.296	2.27869	-5.58388	yes
Glyma.08G211600	0	0	0	no	0	0	0	no
Glyma.11G122500	1.01335	1.05399	0.0567334	no	0.740614	0.650547	-0.187068	no
Glyma.08G159900	1.10702	0.100697	-3.45859	no	0.391604	2.22575	2.50682	no
Glyma.11G185100	0.983639	0.0383857	-4.67949	no	0.417708	0	#NAME?	yes
Glyma.12G047400	8.66275	1.95303	-2.14911	no	5.66565	0.665266	-3.09024	no
Glyma.19G218300	1.44446	0.241334	-2.58143	no	0.806344	0.439241	-0.876383	no
Glyma.15G267000	7.06709	5.43216	-0.379591	no	4.93741	7.12798	0.52974	no
Glyma.03G260300	20.0064	6.30525	-1.66584	no	16.0936	3.37744	-2.25249	no
Glyma.07G114200	0.0114054	0	#NAME?	no	0	0	0	no
Glyma.02G214900	15.3004	16.012	0.0655897	no	15.6459	13.7749	-0.183739	no
Glyma.03G101200	1.88542	0.174744	-3.43158	no	0.18741	0.301955	0.688139	no
Glyma.12G146300	0	0	0	no	0	0	0	no
Glyma.19G003400	0.440398	0	#NAME?	no	0.289189	0	#NAME?	no
Glyma.04G203600	4.12488	8.64749	1.06793	no	4.32074	16.1127	1.89885	no
Glyma.05G003100	0	0.0257294	inf	no	0	0.079215	inf	no
Glyma.07G161900	8.9556	6.28839	-0.510099	no	8.6942	2.33054	-1.89939	no
Glyma.08G323600	1.56365	0.389552	-2.00503	no	2.14879	0.714239	-1.58905	no
Glyma.03G102000	0	0	0	no	0	0	0	no
Glyma.01G227700	3.9092	2.99101	-0.386239	no	3.60991	3.58621	-0.00950172	no
Glyma.02G001500	71.9438	17.4349	-2.04489	no	72.3817	11.8448	-2.61137	no
Glyma.12G183400	11.6502	10.247	-0.185152	no	57.3996	2.05818	-4.8016	yes
Glyma.17G114500	0.296453	0.132127	-1.16588	no	0.137844	0.181403	0.396167	no
Glyma.18G011600	42.9665	15.0631	-1.51219	no	29.7857	14.4824	-1.04033	no
Glyma.02G036300	25.7471	30.3403	0.236823	no	29.8307	59.2463	0.989927	no
Glyma.08G184600	9.08177	9.1657	0.0132705	no	14.0996	9.20523	-0.615128	no

Supplementary Table 1 (Continued). Glyma identifier for genes implicated in the wax biosynthesis pathway and subsequent expression data

Gene	T2-sig	T3-v1	T3-v2	T3-fc	T3-sig	T4-v1	T4v2	T4-fc	T4-sig
Glyma.09G226200	no	0.231161	0.760158	1.7174	no	0.578537	1.00305	0.793907	no
Glyma.11G185200	no	0.0727803	0.515336	2.82389	no	0.224062	2.94875	3.71813	no
Glyma.05G152500	no	48.8621	24.1635	-1.01589	no	22.8996	7.0962	-1.6902	no
Glyma.02G086100	no	13.3309	6.04285	-1.14147	no	11.6073	4.09306	-1.50378	no
Glyma.07G000300	no	28.9172	13.4712	-1.10205	no	21.4894	19.9941	-0.104055	no
Glyma.04G057800	no	67.8472	91.4187	0.430199	no	97.2125	206.012	1.08352	no
Glyma.14G074300	no	4.53112	4.44831	-0.0266115	no	3.12233	3.24843	0.0571165	no
Glyma.07G116200	no	0.109123	0	#NAME?	no	0.0218717	0.0741451	1.76129	no
Glyma.06G058500	no	16.4259	16.0062	-0.0373369	no	19.431	41.0771	1.07998	no
Glyma.15G048400	no	0.0466285	0.0136172	-1.77578	no	0.0659489	0.0175806	-1.90737	no
Glyma.01G225200	no	61.635	56.0878	-0.136064	no	67.1893	42.595	-0.657549	no
Glyma.08G110000	no	7.10996	8.02563	0.174774	no	2.81687	0.822743	-1.77558	no
Glyma.04G149300	no	5.51508	1.07815	-2.35483	no	6.52996	0.677908	-3.26791	no
Glyma.02G266400	no	0	0.051717	inf	no	0	0.274172	inf	no
Glyma.18G086700	no	0	0	0	no	0	0	0	no
Glyma.02G169100	no	0	0	0	no	0	0.00293332	inf	no
Glyma.12G183300	no	0	0.0251866	inf	no	0	0.00375851	inf	no
Glyma.15G114700	no	1.4606	0.792891	-0.88137	no	0.787118	0.605858	-0.3776	no
Glyma.02G296000	no	2.57236	4.58174	0.832805	no	2.70399	5.30032	0.970991	no
Glyma.20G115500	no	16.1402	57.2153	1.82574	no	18.6996	93.7717	2.32615	no
Glyma.05G216600	no	3.28196	5.81442	0.825076	no	2.75119	1.54911	-0.828614	no
Glyma.06G097300	no	10.5605	3.98619	-1.4056	no	5.52222	3.74502	-0.560277	no
Glyma.06G161700	no	9.00379	8.19808	-0.135247	no	7.26818	5.77321	-0.332221	no
Glyma.01G028900	no	30.3844	43.9571	0.532765	no	31.8391	25.3837	-0.326896	no
Glyma.10G092900	no	0	0	0	no	0	0.0251885	inf	no
Glyma.08G323400	no	0	0	0	no	0	0	0	no
Glyma.15G008600	no	0	0	0	no	0	0.00757357	inf	no
Glyma.11G185300	no	22.6647	44.2501	0.965236	no	10.8021	4.91238	-1.13681	no
Glyma.05G011100	no	11.9488	11.1095	-0.105067	no	11.634	11.0937	-0.0686024	no
Glyma.09G196400	no	6.01028	1.62679	-1.88541	no	6.92879	4.46448	-0.634112	no
Glyma.03G101800	no	0	0.0119634	inf	no	0	0.0409843	inf	no
Glyma.08G261100	no	0.297865	0.425843	0.515661	no	0.643116	2.16849	1.75354	no
Glyma.05G003200	no	45.1426	32.6861	-0.46581	no	53.4298	40.1544	-0.412088	no
Glyma.07G031200	no	0	0	0	no	0	0	0	no
Glyma.10G010800	no	15.3677	21.822	0.505884	no	38.0933	94.6737	1.31343	no
Glyma.03G221400	no	0.321591	0.403456	0.327184	no	0.210097	0.697134	1.73038	no
Glyma.13G317700	no	23.3046	9.70871	-1.26327	no	17.4123	1.7803	-3.28992	no
Glyma.02G273300	no	17.648	18.6874	0.0825612	no	16.4888	18.7025	0.181745	no
Glyma.11G017900	no	11.8534	9.86581	-0.264796	no	11.6862	26.5471	1.18375	no
Glyma.07G171400	no	0.201044	0.842958	2.06795	no	0.150377	0.187906	0.321429	no
Glyma.19G046000	no	2.21831	24.6034	3.47132	no	14.7996	152.889	3.36886	yes
Glyma.11G185000	no	0.356291	0.0687144	-2.37437	no	0	0.0238366	inf	no
Glyma.19G089500	no	0	0	0	no	0.0790107	0	#NAME?	no
Glyma.04G147500	no	0.165455	0.306591	0.889874	no	0.113845	0.689819	2.59915	no
Glyma.19G003600	no	0	0	0	no	0	0	0	no
Glyma.17G251000	no	11.895	10.7803	-0.141959	no	12.5604	7.21488	-0.799836	no
Glyma.17G069100	no	103.656	114.308	0.141116	no	117.298	252.284	1.10487	no
Glyma.07G121500	no	0.134188	1.4366	3.42033	no	0.0335995	1.07478	4.99945	no
Glyma.06G221800	no	0	0.0465422	inf	no	0	0.041775	inf	no
Glyma.03G101300	no	0	0.0349102	inf	no	0	0.011644	inf	no

Supplementary Table 1 (Continued). Glyma identifier for genes implicated in the wax biosynthesis pathway and subsequent expression data

Glyma.02G103800	no	19.3979	18.8183	-0.0437698	no	17.5022	15.2453	-0.199173	no
Glyma.18G181200	no	9.16293	9.4885	0.0503706	no	8.15185	7.9715	-0.0322753	no
Glyma.07G132300	no	20.1032	19.2559	-0.0621277	no	18.7005	14.9478	-0.323148	no
Glyma.11G245600	no	9.7447	14.2227	0.545502	no	10.0426	19.5372	0.960085	no
Glyma.09G010200	no	0.812214	1.23703	0.60695	no	1.23195	1.06517	-0.209863	no
Glyma.03G101700	no	0.185901	3.6525	4.29628	no	0.149167	2.44173	4.0329	no
Glyma.13G091200	no	33.6889	36.8927	0.13106	no	41.263	130.491	1.66103	no
Glyma.05G087900	no	0	0	0	no	0	0.0424749	inf	no
Glyma.18G257900	no	0.0772833	0.0226254	-1.77221	no	0.0520055	0.0186579	-1.47888	no
Glyma.10G241700	no	0.107916	0.0770008	-0.486968	no	0.0659122	0.0204412	-1.68906	no
Glyma.10G274400	no	109.983	112.594	0.0338495	no	108.354	206.761	0.932213	no
Glyma.18G089000	no	0	0	0	no	0	0	0	no
Glyma.17G118700	no	16.5767	24.6965	0.575151	no	16.1642	30.8546	0.932677	no
Glyma.05G003000	no	0	0.0141783	inf	no	0	0.0819996	inf	no
Glyma.07G215200	no	20.8411	21.5923	0.051085	no	20.3889	16.7518	-0.283469	no
Glyma.12G087400	no	2.05862	1.26808	-0.699028	no	1.256	2.13869	0.767897	no
Glyma.03G092900	no	0	0	0	no	0	0	0	no
Glyma.20G007900	no	24.3975	23.5965	-0.0481623	no	19.9082	41.2223	1.05006	no
Glyma.18G244400	no	13.1445	12.6345	-0.0570816	no	15.2208	10.378	-0.552513	no
Glyma.14G043500	no	11.7734	17.1943	0.546395	no	14.9228	25.9373	0.797509	no
Glyma.11G015300	no	6.18437	5.06645	-0.287651	no	5.84628	5.0556	-0.209635	no
Glyma.05G198500	no	9.9625	7.55755	-0.398588	no	10.4242	10.473	0.00673931	no
Glyma.08G323100	no	0.499567	1.20189	1.26655	no	0.287949	0.349089	0.277784	no
Glyma.13G166700	no	0.0917775	0.255486	1.47703	no	0.0126608	0.0845616	2.73963	no
Glyma.13G317600	yes	6.65922	32.7481	2.29799	no	17.5224	138.591	2.98356	yes
Glyma.08G211600	no	0	0	0	no	0	0	0	no
Glyma.11G122500	no	0.94506	1.68976	0.838339	no	0.701723	0.749915	0.0958267	no
Glyma.08G159900	no	1.66111	4.08282	1.29741	no	2.52359	0.426722	-2.56411	no
Glyma.11G185100	yes	0	0.486367	inf	no	0.0177866	8.64383	8.92473	no
Glyma.12G047400	no	2.02957	5.45934	1.42755	no	3.94695	15.6981	1.99179	no
Glyma.19G218300	no	0.44285	2.53178	2.51527	no	0.519065	0.672151	0.372871	no
Glyma.15G267000	no	7.42043	6.57621	-0.174248	no	10.0445	4.7171	-1.09044	no
Glyma.03G260300	no	7.17899	7.99893	0.156026	no	7.68016	15.5814	1.02061	no
Glyma.07G114200	no	0.0132072	0.0714568	2.43575	no	0.372082	0.0175685	-4.40455	no
Glyma.02G214900	no	14.4002	11.0548	-0.381424	no	13.0782	16.2599	0.314151	no
Glyma.03G101200	no	0.610269	0.452765	-0.430684	no	0.16585	0.759974	2.19607	no
Glyma.12G146300	no	0	0	0	no	0	0	0	no
Glyma.19G003400	no	0	0.313601	inf	no	0.0381659	0.0209097	-0.868112	no
Glyma.04G203600	no	10.0538	6.18268	-0.701435	no	7.91566	5.11927	-0.62877	no
Glyma.05G003100	no	0.0570763	0.0255787	-1.15795	no	0.00898351	0.00387836	-1.21183	no
Glyma.07G161900	no	5.14147	7.66513	0.576128	no	5.00377	17.1357	1.77592	no
Glyma.08G323600	no	0.734341	1.64975	1.16772	no	0.622146	3.51345	2.49756	no
Glyma.03G102000	no	0	0	0	no	0	0	0	no
Glyma.01G227700	no	3.7559	2.9029	-0.371663	no	3.73172	3.67722	-0.0212261	no
Glyma.02G001500	no	19.7909	36.3311	0.87637	no	36.2989	147.279	2.02056	no
Glyma.12G183400	yes	9.91663	21.4151	1.11071	no	32.8863	80.8361	1.29751	no
Glyma.17G114500	no	0.143048	0.231883	0.696892	no	0.107923	0.232864	1.10949	no
Glyma.18G011600	no	19.3656	27.4841	0.505098	no	22.2188	45.1954	1.02439	no
Glyma.02G036300	no	51.6355	77.854	0.592407	no	50.2711	31.2798	-0.684496	no
Glyma.08G184600	no	9.65126	5.72961	-0.752279	no	9.70249	11.2708	0.216161	no

A.1.2 RNA-seq pattern comparison script (compare_patterns.py)

```
#usr/bin
import sys
import os

def patterns(default=1):


	question="Please select a comparison:\n 1:FL-FH   2:FL-ML   3:ML-MH   4:FH-MH\n"

	prompt=" [1,2,3,4] "
	valid = {"1":['402_404_diff','406_408_diff','412_410_diff','414_416_diff'],
			"2":['402_403_diff','406_407_diff','412_413_diff','414_415_diff'],
			"3":['403_405_diff','407_409_diff','413_411_diff','415_417_diff'],
	"4":['FH-MH1nn','FH-MH2nn','FH-MH3nn','FH-MH4nn']}


	sys.stdout.write(question + prompt)
	choice = raw_input().lower()
	if default is not None and choice == '':
		return valid[default]
	elif choice in valid:
		### Open output file for specified choice
		if choice=='1':
			fout = open('FL-FH_timing_analysis.txt','w')

		elif choice=='2':
			fout = open('FL-ML_timing_analysis.txt','w')
		elif choice=='3':
			fout = open('ML-MH_timing_analysis.txt','w')
		elif choice=='4':
			fout = open('FH-MH_timing_analysis.txt','w')
			fout1 = open('FH-MH_p1.txt','w')
			fout2 = open('FH-MH_p2.txt','w')
			fout3 = open('FH-MH_p3.txt','w')
			fout4 = open('FH-MH_p4.txt','w')
			fout5 = open('FH-MH_p5.txt','w')
			fout6 = open('FH-MH_p6.txt','w')

		fout.write("#Pattern codes:\n# 1: Starts with 0 expression in both samples, faster response in sample 2\n")
		fout.write("# 2: Starts with 0 expression in both samples, faster response in sample 1\n")
```

```
		fout.write("# 3: Starts with expression in both samples, sample 1 turns off while sample 2 remains expressed\n")
		fout.write("# 4: Starts with expression in both samples, sample 2 turns off while sample 1 remains expressed\n")
		fout.write("# 5: Sample 2 has no expression, high expression in sample 1. Sample 2 begins to be expressed\n")
		fout.write("# 6: Sample 1 has no expression, high expression in sample 2. Sample 1 begins to be expressed\n")

		### Open gene_exp.diff in directories for chosen comparison
		genes = {}
		dir = valid[choice]
		for file in dir:
			fh = open(file + '/gene_exp.diff','r')
			for line in fh.readlines():
				if line.startswith('test_id'):
					continue
				### Read line, looking for 'yes' to denote significant differential expression
				### as well as fold change value relative to 0
				line=line.rstrip('\n').split('\t')
				name = line[2]
				if line[13]=='yes':

					if float(line[7])==0 and float(line[8])!=0:
						fc = 'P'
						#return line
					elif float(line[7])!=0 and float(line[8])==0:
						fc = 'N'
						#return line


					else:
						if float(line[9])>0:
							fc = 'p'
						else:
							fc = 'n'
					### Add relevant results to dictionary
					if genes.has_key(name):
						genes[name].append((file,fc))
					else:
						genes[name] = [(file,fc)]
				elif float(line[7])>0 and float(line[8])>0:
					if genes.has_key(name):
						genes[name].append((file,'B'))
					else:
```

```
                                        genes[name] = [(file,'B')]
                                else:
                                        if genes.has_key(name):
                                                genes[name].append((file,"U"))
                                        else:
                                                genes[name] = [(file,"U")]

                        fh.close()
                return_lst = []
                for gene in genes.keys():
                        #eliminate entries with a common pattern through all samples in
series
                        if genes[gene][0][1]==genes[gene][1][1] and
genes[gene][0][1]==genes[gene][2][1] and genes[gene][0][1]==genes[gene][3][1]:
                                continue
                        else:
                                lst = genes[gene]
                                pattern = lst[0][1]+lst[1][1]+lst[2][1]+lst[3][1]
                                return_lst.append(pattern)

                                if pattern[0]=='U':
                                        valid1 =
["UUUP","UUPP","UPPP","UPPp","UPpp","UPpB","UPBB","UppB","UpBB"]
                                        valid2 =
["UUUN","UUNN","UNNN","UNNn","UNnn","UNnB","UNBB","UnnB","UnBB"]
                                        if pattern in valid1:
                                                fout.write(gene + "\t1\n")
                                                if ',' in gene:
                                                        gene = gene.split(',')
                                                        for g in gene:
                                                                fout1.write(g + '\n')
                                                else:
                                                        fout1.write(gene + '\n')

                                        elif pattern in valid2:
                                                fout.write(gene + "\t2\n")
                                                if ',' in gene:
                                                        gene = gene.split(',')
                                                        for g in gene:
                                                                fout2.write(g + '\n')
                                                else:
                                                        fout2.write(gene + '\n')

                                elif pattern[0]=='B':
                                        valid =
["BBBp","BBpp","Bppp","BppP","BpPP","BpPU","BpUU","BPPU","BPUU"]
```

```
                                        valid2 =
["BBBn","BBnn","Bnnn","BnnN","BnNN","BnNU","BnUU","BNNU","BNUU"]
                                        if pattern in valid:
                                                fout.write(gene + "\t3\n")
                                                if ',' in gene:
                                                        gene = gene.split(',')
                                                        for g in gene:
                                                                fout3.write(g + '\n')
                                                else:
                                                        fout3.write(gene + '\n')

                                        elif pattern in valid2:
                                                fout.write(gene + "\t4\n")
                                                if ',' in gene:
                                                        gene = gene.split(',')
                                                        for g in gene:
                                                                fout4.write(g + '\n')
                                                else:
                                                        fout4.write(gene + '\n')

                                elif pattern[0]=='N':
                                        valid =
["NNNn","NNnn","Nnnn","NnnB","NnBB","NBBB"]
                                        if pattern in valid:
                                                fout.write(gene + "\t5\n")
                                                if ',' in gene:
                                                        gene = gene.split(',')
                                                        for g in gene:
                                                                fout5.write(g + '\n')
                                                else:
                                                        fout5.write(gene + '\n')

                                elif pattern[0]=='P':
                                        ##print pattern
                                        valid2 =
["PPPp","PPpp","Pppp","PppB","PpBB","PBBB"]
                                        if pattern in valid2:
                                                fout.write(gene + "\t6\n")
                                                if ',' in gene:
                                                        gene = gene.split(',')
                                                        for g in gene:
                                                                fout6.write(g + '\n')
                                                else:
                                                        fout6.write(gene + '\n')
```

```
else:
        sys.stdout.write("Not valid int! Try again\n\n")
```

A.1.3 Pulling Wax Genes Script (waxid.py)

```
#/usr/bin



def findWaxGenes():
        file_lst = ['FH-MH1','FH-MH2','FH-MH3','FH-MH4']


        gene_lst = []
        genes = open('wax_atl_genes.txt','r')
        table = open('wax_manual_results2.table','w')
        d = {}


        for line in genes.readlines():
                line = line.rstrip('\n')
                gene_lst.append(line)


        for file in file_lst:
                fh = open(file + "/gene_exp.diff",'r')


                for line in fh.readlines():
                        line = line.rstrip('\n').split('\t')


                        if line[2] in gene_lst:
                                gene = line[2]
                                v1 = line[7]
```

```
                    v2 = line[8]
                    fc = line[9]
                    sig = line[13]

                    if d.has_key(gene):
                        d[gene].append((v1,v2,fc,sig))
                    else:
                        d[gene] = [(v1,v2,fc,sig)]
                else:
                    continue
    table.write('Gene\tT1-v1\tT1-v2\tT1-fc\tT1-sig\tT2-v1\tT2-v2\tT2-fc\tT2-sig\tT3-v1\tT3-v2\tT3-fc\tT3-sig\tT4-v1\tT4v2\tT4-fc\tT4-sig\n')

    for key in d.keys():
        table.write(key + '\t')
        for result in d[key]:
            for val in result:
                table.write(val + '\t')
        table.write('\n')
```

A.2 dN/dS Pipeline Scripts

A.2.1 Step 3 Scripts - readBlast.py

```
#/usr/bin

import os

import sys

import tarfile


def __init__(self):

    pass


def main():

        infile = sys.argv[1]


        seqfile = sys.argv[2]



        evalue = float(sys.argv[3])


        # Need to read in a specific output directory to use for output files


        output_folder = str(sys.argv[4])


        seqs = readSeq(seqfile)


        readBlast(infile,seqs,evalue,output_folder)
```

```
def readSeq(seqfile):

    fh = open(seqfile,'r')
    d = {}
    seq = ''
    gene = ''
    for line in fh.readlines():
        if line.startswith('>'):
            if len(seq) > 0:
                d[gene] = seq
                seq = ''
                gene = line.rstrip('\n').lstrip('>').split(None, 1)[0]
                print gene
            else:
                gene = line.rstrip('\n').lstrip('>').split(None, 1)[0]
                print gene
        else:
            seq = seq + line.rstrip('\n')

    d[gene] = seq
    return d


def readBlast(blastfile,seqs,evalue,output_folder):

    fh = open(blastfile)
```

```
        fout = open('tracking_genepairs.txt','w')
        filewritecount = 0
        filewritefolder = "input"
        foldernum = 1
#make that first folder
        os.makedirs(os.path.join(output_folder,str(filewritefolder) + str(foldernum)))

        d = {}
        target = ''
        score = evalue
        lst = []
        for line in fh.readlines():

                if line.startswith('#'):

                        continue

                else:
                        line = line.split('\t')


                        target = line[0]
                        result = line[1]
                        if target == result:
                                continue
```

```
                    if d.has_key(target):
                            pass
                    else:
                            d[target] = []


                    if score > float(line[10]):
                            pass
                    else:
                            continue
                    if result in d[target]:
                            pass
                    else:
                            d[target].append(result)


                            if target in lst:
                                    pass
                            else:
                                    lst.append(target)
                            if result in lst:
                                    pass
                            else:
                                    lst.append(result)
count = 1
for targets in d.keys():
        for item in d[targets]:
                fout.write(targets + '\t' + item + '\n')
```

```
                    if filewritecount >= 10000:
                        tar_up_stuff(output_folder,filewritefolder, foldernum)
                        filewritecount = 0
                        foldernum = foldernum + 1
                        os.makedirs(str(filewritefolder) + str(foldernum))

                    filewritecount = filewritecount + 1
                    filename = 'fasta_pair' + str(count) + '.fasta'
                    foldername = str(filewritefolder) + str(foldernum)
                    completename = os.path.join(output_folder,
os.path.join(foldername,filename))
                    fout2 = open(completename,'w')
                    fout2.write('>' + targets + '\n' + seqs[targets] + '\n' + '>' + item + '\n' +
seqs[item] +'\n')
                    fout2.close()
                    count = count + 1

        tar_up_stuff(output_folder, filewritefolder, foldernum)




def tar_up_stuff(output_folder, folder, num):
        tf = tarfile.open(os.path.join(str(output_folder),str(folder)+str(num)+'.tar'), 'w')
        folder_to_add = os.path.join(str(output_folder), str(folder) + str(num))
        internal_dir = str(folder) + str(num)
```

```
            tf.add(folder_to_add, arcname=internal_dir)
            tf.close()
            os.system("rm -rf %s" % os.path.join(output_folder,str(folder)+str(num)))


if __name__ == "__main__":

    main()
```

A.2.2 Step 4 scripts

A.2.2.1 do_muscle.sh

```
#!/bin/bash
muscle="$PWD/bin/muscle3.8.31_i86linux64"

inputfile="$1"
outputfolder="$2"

filename=$(basename "$inputfile")
extension="${filename##*.}"
foldername="${filename%.*}"

if [[ ! -f "$filename" ]] && [[ "$extension" == "tar" ]]; then
    echo "$filename is not a tar file"
    exit 1
fi

time tar -C $outputfolder -xf $filename
rm $filename

inputdir=${outputfolder}/${foldername}

if [ ! -d "$inputdir" ]; then
            echo "$inputdir is not a directory"
```

```
        exit 1
fi

echo "processing input directory $inputdir"

cd $inputdir

#make the directory
inputbase=$(basename "${inputdir}")
mkdir ../output/${inputbase}

for file in `find . -maxdepth 1 -type f -name "*.fasta"`; do
        $muscle -quiet -loga ../muscle.log -in $file -out
../output/${inputbase}/${file}_MUSCLE.out
done

cd ..
#clean up inputs
rm -rf $foldername

cd output/
#tar up output files
time tar cf ${inputbase}.tar ${inputbase}
#remove that old folder
rm -rf $inputbase
```

A.2.3 Step 5 Scripts – build_pickle.py and convert.py

```
#/usr/bin


try:
   import cPickle as pickle
except:
   import pickle
import sys
import re


def __init__(self):
        pass


def main():
        cds = sys.argv[1 ]
        build(cds)



def build(cds):



        d = {}
        fh = open(cds,'r')
        name = ''
        seq = ''
        for line in fh.readlines():
```

```
        if line.startswith('>'):


                if len(name)>0:
                        ns = ''
                        lst = []
                        for i in range(0,len(seq)):
                                ns = ns + seq[i]
                                if len(ns) == 3:
                                        lst.append((ns))
                                        ns = ''
                        d[name] =  lst
                        seq = ''
                        name = ''
                        ns = ''
                else:
                        pass


                name = line.lstrip('>').rstrip('\n')
                name = name.split()[0]


        else:
                seq = seq+line.rstrip('\n')
lst = []
ns = ''
for i in range(0,len(seq)):
                                ns = ns + seq[i]
```

```
                                        if len(ns) == 3:
                                                lst.append((ns))
                                                ns = ''
        d[name] = lst



        pickle.dump(d, open('cds.p', 'wb'))




def fixname(name_to_fix):
        m = re.search('[_][1-6]$', name_to_fix)
        return re.sub(m.group(0), '', name_to_fix)




if __name__ == '__main__':
        main()



#!/usr/bin/python

import fnmatch
try:
   import cPickle as pickle
```

```
except:
    import pickle
#import cProfile
import os
import re
import shutil
import sys
import tarfile



def __init__(self):
    pass



def main():
    list_of_tars = sys.argv[1]
    output_folder = sys.argv[2]

    d = pickle.load(open('cds.p', 'rb'))

    tars = []
    for line in open(list_of_tars, 'r').readlines():
        tars.append(line.rstrip('\n'))

    filtered_tars = fnmatch.filter(tars, "*.tar")
```

```
    ensure_dir(output_folder);

    for input_tar in filtered_tars:
        if tarfile.is_tarfile(input_tar):
            print "Current tar: ",  input_tar
            cur_tar = tarfile.open(input_tar)
            folder_name = input_tar.rstrip('.tar')
            folder_name = os.path.basename(folder_name)
            ensure_dir(os.path.join(output_folder, folder_name))
            for member in cur_tar.getmembers():
                if member.isreg():
                    f = cur_tar.extractfile(member)
                    convert(f, os.path.join(output_folder, member.name), d)
                    f.close()

            cur_tar.close()
            os.chdir(output_folder)
            tar_out = tarfile.open("".join((folder_name,".tar")),'w')
            tar_out.add(folder_name)
            tar_out.close()
            shutil.rmtree(folder_name)
            os.chdir("..")


def convert(alignment, output, d):
    count = 1
```

```
#fh = open(alignment, 'r')
fh = alignment
name1 = ''
name2 = ''
seq1 = ''
seq2 = ''
name = ''
seq = ''
for line in fh.readlines():

    if line.startswith('>'):
        if len(name) > 0:
            name1 = name
            seq1 = seq
            name = ''
            seq = ''

        name = line.lstrip('>').rstrip('\n').split()[0]

        name = fixname(name)
    else:
        seq = line.rstrip('\n')
name2 = name
seq2 = seq

if len(seq1) != len(seq2):
```

```
    print len(seq1)
    print len(seq2)
    print name1
    return 'error in ' + alignment
else:
    pass

nseq1 = d[name1]

nseq2 = d[name2]

count1 = 0
count2 = 0
new1 = ''
new2 = ''

for i in range(0, len(seq1)):
    if seq1[i] == '-':
        new1 = new1 + '---'
    else:
        new1 = new1 + nseq1[count1]
        count1 = count1 + 1
    if seq2[i] == '-':
        new2 = new2 + '---'
    else:
```

```
            new2 = new2 + nseq2[count2]
            count2 = count2 + 1

    fout = open(output + '.convert', 'w')
    fout.write('>' + name1 + '\n')
    fout.write(new1 + '\n')
    fout.write('>' + name2 + '\n')
    fout.write(new2 + '\n')


def ensure_dir(f):
    if not os.path.exists(f):
        os.makedirs(f)


def fixname(name_to_fix):
    m = re.search('[_][1-6]$', name_to_fix)
    return re.sub(m.group(0), '', name_to_fix)


if __name__ == '__main__':
    main()
```

A.2.4 Step 7 Scripts – assembleResults.py

```
#!/usr/bin/python

import os

import sys



def __init__(self):

    pass



def main():

        #getting the directory of files in as arg1

        assembleResults(sys.argv[1])


def assembleResults(input_folder):


        #dir_lst = os.listdir(os.getcwd())

        dir_lst = []

        for root, dirs, files in os.walk(input_folder, topdown=False):

                for name in files:

                        dir_lst.append(os.path.join(root, name))


        fout = open('assembled_results.txt','w')

        fsum = open('summary_log.txt','w')

        nan_count = 0

        nan_list = ''
```

```
zero_count = 0

total = 0

none_count = 0

none_lst = ''


fout.write('Gene1\tGene2\tdN\tdN_SE\tdS\tdS_SE\tdN/dS\n')


for file in dir_lst:

        if os.path.getsize(file) == 0:

                continue


        if file.endswith('.mlc'):

                total = total + 1

                pass

        else:

                continue


        fh = open(file,'r')


        for line in fh.readlines():

                dn = ''

                ds = ''

                if line.startswith('#1'):

                        line = line.split(':')[1].rstrip('\n')

                        gene1 = line.lstrip(' ')
```

```
        elif line.startswith('#2'):
                line = line.split(':')[1].rstrip('\n')
                gene2 = line.lstrip(' ')



        elif line.startswith('dN'):
                line = line.split('=')
                dn = line[1].split('+-')[0].lstrip(' ')
                dn_se = line[1].split('+-')[1].split('  ')[0].lstrip(' ')
                ds = line[2].split('+-')[0].lstrip(' ')
                ds_se = line[2].split('+-')[1].lstrip(' ').split(' ')[0]


        else:
                continue


if dn == '':
        none_count = none_count + 1
        none_lst = none_lst + file + '\n'
        continue
```

```
                dn = float(dn)
                ds = float(ds)
                print "%s: dn=%s, ds=%s\n" % (file, dn, ds)
                #if dn == 0 and ds == 0:


                if ds == 0:
                        zero_count = zero_count + 1
                else:
                        dn_ds = dn / ds
                        dn_ds = str(dn_ds)
                dn = str(dn)
                ds = str(ds)
                if ds == '-nan' or ds == 'nan' or dn_ds == '-nan' or dn_ds == 'nan':
                        nan_count = nan_count + 1
                        nan_list = nan_list + file + '\n'
                        continue
                fout.write(gene1 + '\t' + gene2 + '\t' + dn + '\t' + dn_se + '\t' + ds + '\t' + ds_se + '\t'
+ dn_ds + '\n')
                fh.close()


        fout.close()
        fsum.write(str(total) + ' PAML reports\n')
        fsum.write(str(zero_count) + ' reports with dS = 0\n')
        fsum.write(str(nan_count) + ' reports with dS = nan\n')
        fsum.write(str(none_count)+ ' reports containing no results\n')
        fsum.write('Report files containing "nan"\n')
        fsum.write(nan_list)
```

```
        fsum.close()



if __name__ == "__main__":

    main()
```

A.2.5 Additional Scripts – do-codeml.sh and do-yn.sh

```
#!/bin/bash

#CODEML=$PWD/bin/codeml

CODEML="../../bin/codeml"


inputfolder="$1"

outputfolder="$2"


if [ ! -d "$inputfolder" ]; then

   echo "$inputfolder is not a directory"

   exit 1

fi


echo "processing input directory $inputfolder"


cd $inputfolder


for file in `find . -maxdepth 1 -type f -name "*_MUSCLE.out.convert"`; do

   ctlfile="${file}.ctl"

   cat <<EOF>>$ctlfile

seqfile = ${file}

outfile = ../../${outputfolder}/${file}.mlc

noisy = 0

verbose = 0

runmode = -2

seqtype = 1
```

```
CodonFreq = 2
*ndata = 10
clock = 0
aaDist = 0
aaRatefile = wag.dat
model = 0
NSsites = 0
icode = 0
Mgene = 0
fix_kappa = 0
kappa = 2
fix_omega = 0
omega = .8
fix_alpha = 1
alpha = 0
Malpha = 0
ncatG = 8
getSE = 1
RateAncestor = 1
Small_Diff = .5e-6
cleandata = 1
*fix_blength = 0
method = 0
EOF
   $CODEML $ctlfile
Done
```

```
#!/bin/bash
#CODEML=$PWD/bin/codeml
CODEML="../../bin/codeml"

inputfolder="$1"
outputfolder="$2"

if [ ! -d "$inputfolder" ]; then
   echo "$inputfolder is not a directory"
   exit 1
fi

echo "processing input directory $inputfolder"

cd $inputfolder

for file in `find . -maxdepth 1 -type f -name "*_MUSCLE.out.convert"`; do
   ctlfile="${file}.ctl"
   cat <<EOF>>$ctlfile
seqfile = ${file}
outfile = ../../${outputfolder}/${file}.mlc
noisy = 0
verbose = 0
runmode = -2
seqtype = 1
CodonFreq = 2
```

```
*ndata = 10
clock = 0
aaDist = 0
aaRatefile = wag.dat
model = 0
NSsites = 0
icode = 0
Mgene = 0
fix_kappa = 0
kappa = 2
fix_omega = 0
omega = .8
fix_alpha = 1
alpha = 0
Malpha = 0
ncatG = 8
getSE = 1
RateAncestor = 1
Small_Diff = .5e-6
cleandata = 1
*fix_blength = 0
method = 0
EOF
   $CODEML $ctlfile
done
```

A.3 Simulator code

A.3.1 defaultCodonTable.txt

GCA 1.11

GCC 0.63

GCG 0.54

GCT 1.72

AGA 2.14

AGG 1.22

CGA 0.73

CGC 0.42

CGG 0.55

CGT 0.93

AAC 0.94

AAT 1.06

GAC 0.63

GAT 1.37

TGC 0.82

TGT 1.18

CAA 1.14

CAG 0.86

GAA 1.06

GAG 0.94

GGA 1.47

GGC 0.57

GGG 0.64

GGT 1.32

CAC 0.76

CAT 1.24

ATA 0.74

ATC 1.02

ATT 1.24

CTA 0.65

CTC 0.99

CTG 0.65

CTT 1.52

TTA 0.85

TTG 1.34

AAA 0.99

AAG 1.01

TTC 0.94

TTT 1.06

CCA 1.34

CCC 0.46

CCG 0.68

CCT 1.51

AGC 0.77

AGT 0.96

TCA 1.24

TCC 0.75

TCG 0.6

TCT	1.67
ACA	1.26
ACC	0.8
ACG	0.59
ACT	1.35
TAC	0.93
TAT	1.07
GTA	0.62
GTC	0.75
GTG	1.01
GTT	1.61
ATG	1
TGG	1

A.3.2 hky.py

```
#!/usr/bin/python


import random

from random import randint

from math import *

import sys

from random import uniform

import argparse


def __init__(self):

    pass




def main():


    parser = argparse.ArgumentParser(description='Generates sequences for dN/dS testing using the HKY model of nucleotide substitution.')

    parser.add_argument('-o', action='store', dest='outfile', help='Name of results file in FASTA format.  Will default to hky_mut_results.fasta if not specified.', type=str, default='hky_mut_results.fasta')

    parser.add_argument('-d', action='store', dest='dsfile', help='Name of results file of dS values derived from simulated sequences.  Will default to dS_for_hist.txt if not specified.', type=str, default='dS_for_hist.txt')

    parser.add_argument('-a', action='store', dest='alpha_val', help='Set the value for alpha, default = 0.4',default=.4,type=float)
```

```
	parser.add_argument('-b', action='store', dest='beta_val', help='Set the value for beta, default = .02', default=.2, type=float)

	parser.add_argument('-T', action='store', dest='T_val', help='Set the base frequency for T, default = 0.25', default=.25, type=float)

	parser.add_argument('-A', action='store', dest='A_val', help='Set the base frequency for A, default = 0.25', default=.25, type=float)

	parser.add_argument('-G', action='store', dest='G_val', help='Set the base frequency for G, default = 0.25', default=.25, type=float)

	parser.add_argument('-C', action='store', dest='C_val', help='Set the base frequency for C, default = 0.25', default=.25, type=float)

	parser.add_argument('--codon', action='store', dest='codonTable', help='File specifying non-default codon usage information in tab delimited format.  See defaultCodonTable.txt for format specifications.  Default codon usage from A. thaliana, adapted from Wang and Roossinck 2006.', type=str, default='defaultCodonTable.txt')

	parser.add_argument('-i', action='store', dest='iterations', help='The number of iterations to perform.  Will generate 1 pair of genes per iteration.  default = 10000', default=10000, type=int)

	parser.add_argument('-t', action='store', dest='time', help='The number of time steps.  default = 1', default=1, type=int)

	args = parser.parse_args()


	if args.T_val + args.A_val + args.C_val + args.G_val != 1:

		print "Error:  Specified base frequencies did not total 1.  Please fix"

		return 0

	else:

		pass



	print 'outfile:\t', args.outfile
```

```
print 'alpha:\t', args.alpha_val

print 'beta:\t', args.beta_val

print 'T freq:\t', args.T_val

print 'A freq:\t', args.A_val

print 'C freq:\t', args.C_val

print 'G freq:\t', args.G_val

print 'codon table:\t', args.codonTable

print 'iterations:\t', args.iterations

print 'time steps:\t', args.time



d = codonUsageTable(args.codonTable)

fout = open(args.outfile,'w')

fout2 = open(args.dsfile,'w')

i = 1

if args.time > 0:


        while i <= args.iterations:

                seq = genseq(random.randrange(450,1350,3),d)

                result =
hky(args.alpha_val,args.beta_val,args.T_val,args.C_val,args.A_val,args.G_val,seq,args.ti
me)

                fout.write('>seq'+str(i)+"A_dS=" + result[2] + '_dN=' + result[3]
+'\n')

                fout2.write(result[2] + '\n')

                fout.write(result[0] + '\n')
```

```
                    fout.write('>seq'+str(i)+"B_dS=" + result[2] + '_dN=' + result[3]
+'\n')
                    fout.write(result[1] + '\n')
                    i = i + 1
        else:
            while i <= args.iterations:
                    seq = genseq(random.randrange(150,450,3),d)
                    seq2 = ''
                    for nu in seq:
                        seq2 = seq2+nu
                    fout.write('>seq'+str(i)+"A_dS=0" + '_dN=0'  +'\n')
                    fout2.write('0\n')
                    fout.write(seq2 + '\n')
                    fout.write('>seq'+str(i)+"B_dS=0" + '_dN=0' +'\n')
                    fout.write(seq2+'\n')
                    i = i + 1
```

```
def genseq(n,d):

    codons = codonUsageTable()
    max = codons['max_val']
    seq = 'ATG'
    for i in range(0,n/3):
        choice = uniform(0,max)
        #print choice
        for key in codons.keys():
            if key == 'max_val':
                continue
            lst = codons[key]

            if choice >= lst[0] and choice < lst[1]:
                seq = seq + key
                #print lst
                #print key
            else:
                continue

    #print seq
    new_seq = []
    for codon in seq:
        for nucl in codon:
            new_seq.append(nucl)
```

```
        return new_seq


s = {'ATT':2.0/3,
'TTT':1.0/3,'TTC':1.0/3,'TTA':2.0/3,'TTG':2.0/3,'TCT':1.0,'TCC':1.0,'TCA':1.0,'TCG':1.0,'
TAT':1.0,'TAC':1.0,'TGT':1.0/2,'TGC':1.0/2,'TGG':0.0,

'CTT':1.0,'CTC':1.0,'CTA':4.0/3,'CTG':4.0/3,'CCT':1.0,'CCC':1.0,'CCA':1.0,'CCG':1.0,'C
AT':1.0/3,'CAC':1.0/3,'CAA':1.0/3,'CAG':1.0/3,'CGT':1.0,'CGC':1.0,'CGA':8.0/5,'CGG':4.
0/3,

'AAT':2.0/3,'ATC':2.0/3,'ATA':2.0/3,'ATG':0.0,'ACT':1.0,'ACC':1.0,'ACA':1.0,'ACG':1.0,
'AAT':1.0/3,'AAC':1.0/3,'AAA':1.0/3,'AAG':1.0/3,'AGT':1.0/3,'AGC':1.0/3,'AGA':4.0/5,'
AGG':2.0/3,

'GTT':1.0,'GTC':1.0,'GTA':1.0,'GTG':1.0,'GCT':1.0,'GCC':1.0,'GCA':1.0,'GCG':1.0,'GAT'
:1.0/3,'GAC':1.0/3,'GAA':1.0/3,'GAG':1.0/3,'GGT':1.0,'GGC':1.0,'GGA':1.0,'GGG':1.0,'T
AA':0.0,'TGA':0.0,'TAG':0.0}



aa = ['TTT','TTC','TTA','TTG','TCT','TCC','TCA','TCG','TAT','TAC','TGT','TGC','TGG',

'CTT','CTC','CTA','CTG','CCT','CCC','CCA','CCG','CAT','CAC','CAA','CAG','CGT','CG
C','CGA','CGG',

'AAT','ATC','ATA','ATG','ACT','ACC','ACA','ACG','AAT','AAC','AAA','AAG','AGT','A
GC','AGA','AGG',

'GTT','GTC','GTA','GTG','GCT','GCC','GCA','GCG','GAT','GAC','GAA','GAG','GGT','G
GC','GGA','GGG']

d = {'Met':['ATG'], 'Phe':['TTT', 'TTC'], 'Leu':['TTA', 'TTG', 'CTT', 'CTC', 'CTA', 'CTG'],
'Cys':['TGT', 'TGC'], 'Tyr':['TAC', 'TAT'], 'Trp':['TGG'], 'Pro':['CCT', 'CCC', 'CCA',
'CCG'], 'His':['CAT', 'CAC'],
```

```
'Gln':['CAA', 'CAG'], 'Arg':['CGT', 'CGC', 'CGA', 'CGG', 'AGA', 'AGG'], 'Ile':['ATT',
'ATC', 'ATA'], 'Thr':['ACT', 'ACC', 'ACA', 'ACG'],

'Asn':['AAT', 'AAC'], 'Lys':['AAA', 'AAG'], 'Ser':['AGT', 'AGC', 'TCT', 'TCC', 'TCA',
'TCG'], 'Val':['GTT', 'GTC', 'GTA', 'GTG'],

'Ala':['GCT', 'GCC', 'GCA', 'GCG'], 'Asp':['GAT', 'GAC'], 'Glu':['GAA', 'GAG'],
'Gly':['GGT', 'GGC', 'GGA', 'GGG'], '*':['TAA','TAG','TGA']}



def hky(a,b,T,C,A,G,seq,time):

        # a = alpha

        # T,C,G,A are nucleotide rates

        # b is beta

        syn_ch = 0

        nsyn_ch = 0

        m=len(seq)

        list = seq

        count = 0


        list2 = []

        for i in range(0,len(list)):

                list2.append(list[i])

        #print "Beginning rate calculations"

        #Rate calculations

        T1 hx = T + ((T*(A+G))/(T+C))*exp(-b*1) + (C/ (T+C)) *exp((-(T+C)
*a+(A+G) *b)*1)
```

```
        C1ix = C + ((C*(A+G))/(T+C))*exp(-b*1) - (C/ (T+C)) *exp((-(T+C) *a+(A+G)
*b)*1)

        T2jx = T + ((T*(A+G))/(T+C))*exp(-b*1) - (T/ (T+C)) *exp((-(T+C) *a+(A+G)
*b)*1)

        C2kx = C + ((C*(A+G))/(T+C))*exp(-b*1) + (T/ (T+C)) *exp((-(T+C) *a+(A+G)
*b)*1)

        A1lx = A + ((A*(T+C))/(A+G))*exp(-b*1) + (G/ (A+G)) *exp((-(A+G)
*a+(T+C) *b)*1)

        G1mx = G + ((G*(T+C))/(A+G))*exp(-b*1) - (G/ (A+G)) *exp((-(A+G)
*a+(T+C) *b)*1)

        A2nx = A + ((A*(T+C))/(A+G))*exp(-b*1) - (A/ (A+G)) *exp((-(A+G)
*a+(T+C) *b)*1)

        G2ox = G + ((G*(T+C))/(A+G))*exp(-b*1) + (A/ (A+G)) *exp((-(A+G)
*a+(T+C) *b)*1)

        #print "\nRate calculations continuing"

        A0px = A * (1 - (exp(-b*1)))

        G0qx = G * (1 - (exp(-b*1)))

        T0rx = T * (1 - (exp(-b*1)))

        C0sx = C * (1 - (exp(-b*1)))

        T1xhs = hx

        C1xi = ix

        T2xjs = jx

        A1xls = lx

        G1xms = mx

        A2xns = nx

        G2xos = ox

        A0xps = px

        G0xqs = qx
```

```
T0xrs = rx

C0xss = sx

TT = C1x + A0x + G0x

TC = T2x + A0x + G0x

TA = T0x + C0x + G1x

TG = T0x + C0x + A2x

#print "\n Rate calculations complete"

x = ((list.count('T')) * TT) + ((list.count('C')) * TC) + ((list.count('A')) * TA) + ((list.count('G')) * TG)

#print "caculating t"


for iii in range(0,time):


    t = random.expovariate(x)

    while t < 1:


        p =randint(0,m-1)

        B = list2[p]

        if B == 'A':


            f = random.expovariate(T0x)

            g = random.expovariate(G1x)

            c = random.expovariate(C0x)

            if f < g and f < c:

                list2[p] = 'T'
```

```
            dx = newCodon(list,list2,p)
            if dx == 'stop':

                list2[p] = 'A'
            elif dx == 'dS':

                syn_ch = syn_ch+1

            elif dx == 'dN':

                nsyn_ch = nsyn_ch + 1


    else:
            if g < f and g < c:
                list2[p] = 'G'
                dx = newCodon(list,list2,p)
                if dx == 'stop':
                        list2[p] = 'A'
                elif dx == 'dS':
                        syn_ch = syn_ch+1
                elif dx == 'dN':
                        nsyn_ch = nsyn_ch + 1
            else:
```

```
                    if c < f and c < g:
                        list2[p] = 'C'
                        dx = newCodon(list,list2,p)
                        if dx == 'stop':
                            list2[p] = 'A'
                        elif dx == 'dS':
                            syn_ch = syn_ch+1
                        elif dx == 'dN':
                            nsyn_ch = nsyn_ch + 1
else:
    if B == 'C':
        f = random.expovariate(T2x)
        a = random.expovariate(A0x)
        g = random.expovariate(G0x)
        if f < a and f < g:
            list2[p] = 'T'
            dx = newCodon(list,list2,p)
            if dx == 'stop':
                list2[p] = 'C'
            elif dx == 'dS':
                syn_ch = syn_ch+1
            elif dx == 'dN':
                nsyn_ch = nsyn_ch + 1
        else:
            if a < f and a < g:
```

```
                    list2[p] = 'A'
                    dx = newCodon(list,list2,p)
                    if dx == 'stop':
                        list2[p] = 'C'
                    elif dx == 'dS':
                        syn_ch = syn_ch+1
                    elif dx == 'dN':
                        nsyn_ch = nsyn_ch + 1
                else:
                    if g < f and g < a:
                        list2[p] = 'G'
                        dx = newCodon(list,list2,p)
                        if dx == 'stop':
                            list2[p] = 'C'
                        elif dx == 'dS':
                            syn_ch = syn_ch+1
                        elif dx == 'dN':
                            nsyn_ch = nsyn_ch +
1


            else:
                if B == 'T':
                    c = random.expovariate(C1x)
                    a = random.expovariate(A0x)
                    g = random.expovariate(G0x)
```

```
                if c < a and c < g:
                        list2[p] = 'C'
                        dx = newCodon(list,list2,p)
                        if dx == 'stop':
                                list2[p] = 'T'
                        elif dx == 'dS':
                                syn_ch = syn_ch+1
                        elif dx == 'dN':
                                nsyn_ch = nsyn_ch + 1
                else:
                        if a < c and a < g:
                                list2[p] = 'A'
                                dx = newCodon(list,list2,p)
                                if dx == 'stop':
                                        list2[p] = 'T'
                                elif dx == 'dS':
                                        syn_ch = syn_ch+1
                                elif dx == 'dN':
                                        nsyn_ch = nsyn_ch +
1
                        else:
                                if g < a and g < c:
                                        list2[p] = 'G'
                                        dx =
newCodon(list,list2,p)
                                        if dx == 'stop':
```

```
                                        list2[p] = 'T'
                                    elif dx == 'dS':
                                        syn_ch =
syn_ch+1
                                    elif dx == 'dN':
                                        nsyn_ch =
nsyn_ch + 1
                            else:
                                if B=='G':
                                    f = random.expovariate(T0x)
                                    a = random.expovariate(A2x)
                                    c = random.expovariate(C0x)
                                    if f < a and f < c:
                                        list2[p] = 'T'
                                        dx = newCodon(list,list2,p)
                                        if dx == 'stop':
                                            list2[p] = 'G'
                                        elif dx == 'dS':
                                            syn_ch = syn_ch+1
                                        elif dx == 'dN':
                                            nsyn_ch = nsyn_ch +
1
                                    else:
                                        if a < f and a < c:
                                            list2[p] = 'A'
                                            dx =
newCodon(list,list2,p)
```

```
                                        if dx == 'stop':
                                            list2[p] = 'G'
                                        elif dx == 'dS':
                                            syn_ch =
syn_ch+1
                                        elif dx == 'dN':
                                            nsyn_ch =
nsyn_ch + 1
                                else:
                                        if c < a and c < f:
                                            list2[p] = 'C'
                                            dx =
newCodon(list,list2,p)
                                            if dx == 'stop':
                                                list2[p]
= 'G'
                                            elif dx == 'dS':
                                                syn_ch
= syn_ch+1
                                            elif dx ==
'dN':
            nsyn_ch = nsyn_ch + 1

                        x = ((list2.count('T')) * TT) + ((list2.count('C')) * TC) +
((list2.count('A')) * TA) + ((list2.count('G')) * TG)

                        t = t + random.expovariate(x)
```

```
#print list
#print list2
seq2 = ''
seq1 = ''
#print seq1
#print seq2
for nu in list:
        seq1 = seq1 + nu
for nu in list2:
        seq2 = seq2 + nu



synS = synSites(seq1,seq2)
NsynS = (len(seq1)*3) - synS

#print '# of syn changes'
#print syn_ch
#print '# of non syn changes'
#print nsyn_ch
dS = str(syn_ch / synS)[0:6]
dN = str(nsyn_ch / NsynS)[0:6]
return (seq1,seq2,dS,dN)
```

```
def newCodon(list,list2,p):
        #print p
        oldcodon = findCodon(list,p)
        #print oldcodon
        newcodon = findCodon(list2,p)
        #print newcodon
        old_aa = ''
        new_aa = ''
        stops = ['TAA','TAG','TGA']
        for key in d.keys():
                if oldcodon in d[key]:
                        old_aa = key
                if newcodon in d[key]:
                        new_aa = key
        #print old_aa
        #print new_aa
        if new_aa in stops:

                return 'stop'
        if new_aa == '*':

                return 'stop'
        if new_aa == '':
```

```
            return 'stop'
        if old_aa == new_aa:
            return 'dS'
        else:
            return 'dN'



def findCodon(lst,p):
    codon = ''
    br = 0
    for i in range(0,len(lst)):
        if i == p:
            br = 1
        codon = codon + lst[i]
        if len(codon) % 3 == 0:
            if br == 1:
                return codon
                break
            codon = ''



def synSites(seq1,seq2):
```

```
        total = 0

        for i in range(0,len(seq1),3):
                codon1 = seq1[i:i+3]
                codon2 = seq2[i:i+3]

                synsite1 = s[codon1]
                synsite2 = s[codon2]
                #print synsite1
                #print synsite2
                if synsite1 == synsite2:
                        total = total + synsite1
                        #print total
                else:
                        total = total + ((synsite1+synsite2)/2)
                        #print total

        return total



def codonUsageTable(file='defaultCodonTable.txt'):
```

```
        fh = open(file,'r')
        d = {}
        count = 0.0
        for line in fh.readlines():
                line = line.rstrip('\n').split('\t')
                key = line[0]

                val = float(line[1])
                d[key] = (count,count+val)
                count = count + val

        d['max_val'] = count
        return d



if __name__ == "__main__":

    main()
```