Web-based Computational Tools for Exploration, Analysis, and Visualization of Genomic Data

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Ming Chen

December 2018

Copyright © 2018 by Ming Chen

All rights reserved.

Dedicated to my beloved wife

Rong Wang

ACKNOWLEDGEMENTS

This work was carried out during 2015.08 ~ 2018.12 at the Department of Genome Science and Technology and the Department of Entomology and Plant Pathology, University of Tennessee, Knoxville, TN.

I would like to express my deepest appreciation to my mentor Dr. Margaret E. Staton. Without her consistent support, guidance and trust throughout my dissertation research, this work would not have been possible. Thanks to her for being extremely supportive since the first day I joined the lab, for providing me useful and practical instructions and helping me overcome obstacles and challenges one after another, for leading me to the right paths when I felt confused and indecisive, and for giving me the freedom to develop skills needed to achieve my career goals.

I would like to extend my sincere thanks to my committee members Dr. Jill L. Wegrzyn, Dr. Russell L. Zaretzki, Dr. Michael A. Langston and Dr. Daniel A. Jacobson for their guidance, discussions and constructive criticism. My special thanks should go to Dr. John M. Zobel and Dr. Russell Zaretzki for their assistance in my Master's program training in Statistics. I am also extremely grateful to Dr. Albrecht von Arnim for his support and mentorship, to Ms. Terrie Yeatts for her tremendous help as the GST program coordinator.

I would like to thank my current and past lab mates: Dr. Bradford Condon, Abdullah Almsaeed, Matt Huff, Fang Liu, Jiali Yu, Dr. Miriam Payá, Nathan Henry, and Thomas Lane, for sharing their knowledge and experiences, and for creating a peaceful and joyful office environment to work and study in.

Finally, I would like to express my utmost gratitude and indebtedness to my wife, my parents, and my parents-in-law for their support, love and sacrifice, and for being with me all the time.

ABSTRACT

The advent of high throughput technologies not only brought huge amounts of biological data but also many versatile bioinformatics tools, as well as highly sophisticated and complex analysis workflows (Ortuño and Rojas, 2016). However, the majority of biological scientists are not bioinformaticians. There is a gap between the average biologist and what is needed in terms of bioinformatics and statistics knowledge, and programming skills for tackling biological questions with those already huge but still growing amount of biological data. The aim of this study is to develop a set of web-based computational tools to enable biological researchers to access and acquire data from Tripal databases more efficiently, as well as gain insights from data with easy-to-use and powerful bioinformatics tools. The research goal was achieved by improving and extending the biological database construction toolkit, Tripal, and the web-based computational platform, Galaxy. First, I developed two Tripal extension modules: one for storing and visualizing high throughput differential expression experiments data, including gene expression data, NCBI Biosample metadata, and analysis methodology data; and the other for site-wide and customizable search. Second, I created an R Markdown based framework for Galaxy tool development. The new framework provides a generalized tool wrapper and enables the generation of output in more intuitive and standardized HTML reports. Finally, I developed the *Galaxy Tool Generator* to improve the efficiency of developing Galaxy tools. In the last chapter, I presented a straightforward but comprehensive use case to demonstrate how these computational tools can facilitate scientific research.

TABLE OF CONTENTS

CHAPTER I. INTRODUCTION 1

1.1 BIOLOGY IN THE ERA OF BIG DATA 1

1.2 THE CHALLENGES OF BIG BIOLOGICAL DATA 2

1.3 TRIPAL: CONSTRUCTING BIOLOGICAL DATABASES 3

1.4 HARDWOOD GENOMICS DATABASE: A TRIPAL-BASED DATABASE 4

1.5 GALAXY: TURNING DATA INTO INFORMATION AND KNOWLEDGE 8

1.6 INTEGRATION OF TRIPAL AND GALAXY 9

1.7 OVERVIEW OF RESEARCH GOALS 11

1.7.1 STORAGE, DISPLAY AND VISUALIZATION OF GENE EXPRESSION ANALYSIS DATA 11

1.7.2 THE NEED FOR EFFICIENT SEARCH IN TRIPAL DATABASES 12

1.7.3 DATA ANALYSIS TRANSPARENCY AND RESULTS REPORTING IN GALAXY TOOLS 12

1.7.4 EFFICIENCY IN GALAXY TOOL DEVELOPMENT 13

1.8 OVERVIEW OF TECHNOLOGIES 14

1.8.1 DRUPAL AND TRIPAL SOFTWARE 14

1.8.2 ELASTICSEARCH 14

1.8.3 R MARKDOWN 15

1.8.4 DRUPAL WEBFORM MODULE 16

1.8.5 DOCKER 16

1.9 APPENDIX 18

CHAPTER II. NEW EXTENSION SOFTWARE MODULES TO ENHANCE SEARCHING AND DISPLAY OF TRANSCRIPTOME DATA IN TRIPAL DATABASES....24

2.1 ABSTRACT....25

2.2 INTRODUCTION....26

2.3 TRIPAL ELASTICSEARCH MODULE....29

2.4 TRIPAL ANALYSIS EXPRESSION MODULE....35

2.5 CONCLUSIONS....40

2.6 ACKNOWLEDGEMENTS....41

2.7 FUNDING....41

2.8 APPENDIX....42

CHAPTER III. AURORA GALAXY TOOLS: USING R MARKDOWN AS A FRAMEWORK FOR GALAXY TOOL DEVELOPMENT....52

3.1 ABSTRACT....53

3.2 BACKGROUND....54

3.3 METHODS....56

3.4 RESULTS....61

3.5 USE CASES....64

3.6 DISCUSSION....66

3.7 ACKNOWLEDGEMENTS & FUNDING....68

3.8 APPENDIX....70

CHAPTER IV. GALAXY TOOL GENERATOR (GTG): A WEB APPLICATION FOR DEVELOPING GALAXY TOOLS....79

4.1 ABSTRACT....80

4.2 INTRODUCTION **81**

4.3 IMPLEMENTATION **82**

4.4 DISCUSSION **85**

4.5 ACKNOWLEDGEMENTS & FUNDING **86**

4.6 APPENDIX **87**

CHAPTER V. USE CASE: GENE FAMILIES INVOLVED IN O_3 STRESS RESPONSE FROM THREE PHYLOGENETICALLY DIVERSE SPECIES OF HARDWOOD TREES **88**

5.1 INTRODUCTION **89**

5.2 DATASETS **91**

5.3 DISCOVERING AND DOWNLOADING DATA WITH TRIPAL ELASTICSEARCH AND TRIPAL ANALYSIS EXPRESSION **92**

5.4 DATA ANALYSIS AND VISUALIZATION WITH TRIPAL GALAXY MODULE AND AURORA GALAXY TOOLS **94**

5.4.1 ANALYSIS WORKFLOW SUMMARY 94

5.4.2 ANALYSIS IMPLEMENTATION 96

5.4.3 RESULTS AND DISCUSSION 98

5.5 AURORA GALAXY TOOL DEVELOPMENT WITH GTG **102**

5.5.1 LAUNCH GTG WEB APPLICATION 102

5.5.2 CREATE GALAXY TOOL XML FILE 103

5.5.3 ADD SHELL SCRIPTS TO THE COMMAND SECTION 104

5.5.4 ADD ANALYSIS LOGIC 105

5.5.5 PUBLISH A DRAFT TOOL TO TEST TOOLSHED FOR ITERATIVE DEVELOPMENT 105

5.5.6 SYNC TOOL UPDATES IN GTG TO GALAXY INSTANCE 106

5.5.7 SHUT DOWN AND REMOVE GTG 107

5.5 APPENDIX 108

CHAPTER VI. CONCLUSIONS 126

REFERENCES 128

VITA 145

LIST OF TABLES

Table 2.1. Data types supported by Tripal. 42

Table 3.1 File compositions and functionalities of standard Galaxy tools and Aurora Galaxy tools. 70

Table 3.2 Template files and functionalities for developing Aurora Galaxy tools. 71

Table 3.3 Aurora tools for RNA-Seq differential expression analysis. 72

LIST OF FIGURES

Figure 1.2 Galaxy user interface and graphical workflow editor. 20

Figure 1.3 Tripal Galaxy Module as a Tripal gateway to Galaxy. 21

Figure 1.4 R Markdown document (A) and R Markdown rendered as a HTML file (B). 22

Figure 1.5 Drupal webform interface (A) and example form (B)..... 23

Figure 2.1. The Tripal Elasticsearch module depends on a number of other software packages. 43

Figure 2.2. The Tripal Elasticsearch module administrative pages provide centralized control of data indexing and construction of search interfaces. 44

Figure 2.3. The Tripal Elasticsearch module builds content blocks for search forms. 46

Figure 2.4. Flow of data in the Tripal Analysis Expression module. 48

Figure 2.5. Administration interface of the Expression module. 49

Figure 2.6. User interface of Tripal Analysis Expression module. 50

Figure 3.1 Interactive file tree display of all tool outputs from the aurora_fastqc tool. 73

Figure 3.2 Single HTML report from aurora_skewer tool. 74

Figure 3.3 Screenshots of reports *from aurora_fastqc*. 75

Figure 3.4 Screenshots of reports from *aurora_star* tool. 76

Figure 3.5 Screenshots of reports from *aurora_htseq*. 77

Figure 3.6. Screenshots of reports from *aurora_deseq2*..... 78

Figure 4.1. Interfaces and structure of the Galaxy Tool Generator application. 87

Table 5.1 Shared protein families of differentially expressed genes under different O_3 exposure across three hardwood tree species 108

Figure 5.1 Collecting data from HGD website. 109

Figure 5.2 Expression data format. 110

Figure 5.3 Treatment data format. 111

Figure 5.4 Analysis workflow and final report. 112

Figure 5.5 the view of deseq2-hmmscan-pfam and hmmscan-pfam-merger workflows in Galaxy's graphic workflow editor. 113

Figure 5.6 Steps of deseq2-hmmscan-pfam workflow in HGD website. 114

Figure 5.7 Analysis results page of the *deseq2-hmmscan-pfam* workflow. 115

Figure 5.8 screenshots of aurora_deseq2 analysis report 116

Figure 5.9 Steps of hmmscan-pfam-merger workflow. 117

Figure 5.10 Analysis results of *hmmscan-pfam-merger* workflow. 118

Figure 5.11 Interfaces of GTG and connected Galaxy instance. 119

Figure 5.12 Interface of Create Galaxy Tool XML. 120

Figure 5.13 Web form components interface. 121

Figure 5.14 Adding a requirement component for the tool dependency HMMER. 122

Figure 5.15 Interface to add XML file to the galaxy_tool_repository folder. 123

Figure 5.16 Interface to add ToolShed/Test ToolShed API keys for publishing Galaxy tools. . 124

Figure 5.17 Interface for publishing Galaxy tools. 125

CHAPTER I. INTRODUCTION

1.1 Biology in the era of big data

The rapid growth of data over the last decade in biology, especially in genomics, has brought us into the era of big data. The total number of whole genome sequences records in the United States National Center for Biotechnology Information (NCBI) is up to 722,438,528 (accessed by Oct 2018). The Sequence Read Archive (SRA) maintained by the NCBI is one of the largest repositories in the world for hosting raw sequencing reads used in most published studies. By 2018 over 10 petabases of raw sequence data had been deposited to SRA. The archived sequence data in SRA only represents a small fraction of the total produced, as most of it is not yet stored in SRA or any of its counterparts (Stephens *et al*., 2015). The amount of genetic sequencing data stored at European Bioinformatics Institute (EBI) takes less than one year to double in size. The majority of data in both NCBI and EBI is publicly available, giving biological researchers easy access to these petabytes of data (Marx, 2013a).

This explosion in biological data is largely due to continued advances in high throughput structural and functional genomic technologies. It has become increasingly affordable and faster to obtain large quantities of data with increasing accuracy (Stephens *et al*., 2015; Diehn *et al*., 2003; O'Driscoll *et al*., 2013; Li and Chen, 2014). In the Human Genome Project (Collins *et al*., 2003), experts from 20 institutes spent 13 years and over $3 billion to identify the entire genome of approximately 3 billion nucleotides. By 2014, the cost of sequencing a human genome dropped to $1000 and could be done within a few days (Li and Chen, 2014). Nowadays, it will only take

less time to sequence a human genome at smaller cost. Before 2008, how DNA sequencing costs reduced as time could be exceedingly well explained by the Moore's Law. The Moore's Law was coined in 1965 by Intel co-founder Gordon Moore to describe the trend that the number of transistors that can be placed on an integrated circuit board doubles every year (Brock and Moore, 2006). Since the formulation of this empirical rule, Moore's Law has been a golden guide for technology development, including sequencing technologies. However, the advent of 'second generation' DNA sequencing technologies in 2008 broke the rule, resulting a dramatic reduction in sequencing cost but increase in sequencing capacity (Wetterstrand, 2016) (Figure 1.1). The third "generation sequencing" technologies (Sanders *et al*., 2017) sustain the rapid falling of sequencing costs and the explosive growth in sequencing data

1.2 The challenges of big biological data

Big data is much more than simply a matter of size (Bizer *et al*., 2012). In addition to the property of being large in volume, big biological data is also defined by the high velocity of data generation, the huge variety of data sources and formats, and enormous variability in data context (Katal *et al*., 2013; May, 2014; Khan *et al*., 2018). The four V's (volume, velocity, variety, variability) are signature properties of big data and the complexity caused by their combinations pose increasing challenges in data engineering and semantics (Bizer *et al*., 2012). Inevitably, finding solutions to manage data at unimaginable scales and to efficiently extract information from data has become more and more difficult for biologists, many of whom have never been trained in data or computational science. There is an ongoing need to fill this gap by building robust cyberinfrastructure that connects scientists, software tools, computers, storage systems and data through intuitive online interfaces.

1.3 Tripal: constructing biological databases

Tripal (Ficklin *et al*., 2011; Sanderson *et al*., 2013) is a set of software modules that provides a simplified solution for the construction of informative, research-oriented websites for the biological scientific community. It is built as a suite of Drupal (VanDyk and Westgate, 2007) modules that incorporate the community standard GMOD Chado (Mungall *et al*., 2007) schema into a Drupal-based web database. Chado is an open source database schema that was first designed for the FlyBase (FlyBase Consortium, 2003), and since its initial release, has been developed and adopted by many biological online databases. Currently, there are over 130 Chado tables which support the storage of a variety of types of biological data such as genomic features, sequence data, expression data, phenotypic data, genomic libraries, stocks, publications and much more. Drupal, the underlying technology, is a popular open source Content Management System (CMS). Drupal empowers admins and users to add content to websites as well as edit or delete content without programming. Drupal's functionality can be extended by installing any of the thousands of modules developed by the Drupal developer community. A Drupal API is also available to build custom modules and websites.

With the advances in DNA sequencing technologies, it is possible for single research groups to generate a large amount of genomic data very rapidly and at a substantially small cost. Research communities are facing increasing challenges of creating and maintaining databases with large quantities of biological data. While the online database is a perfect way for sharing and managing 'big data', most biologists don't have the expertise or necessary resources to build such an online database. Tripal uses the open-source, biologically-oriented database Chado for data storage and can be installed in the Drupal system like other Drupal modules to construct research-oriented biological sites without the need of programming. Tripal has been adopted by a variety of

plant and animal websites (Table 1.1). Researchers with programming expertise can write custom Tripal modules with the Tripal API to extend its functionality, or modify Tripal's data templates to create customized biological websites. Currently Tripal has 30 extensions, and is still undergoing active development from over 10 groups (Tripal 3, 2018).

Tripal, Chado, and many other interoperable software tools are members of the Generic Model Organism Databases project (GMOD, http://gmod.org/). GMOD is a collection of database schemas and open software tools for genetic and genomic data storage, manipulation, visualization, analysis as well as web-based database construction. In early 2000s, NIH, NSF and USDA-ARS funded GMOD to satisfy these purposes for biological research communities, and it has since continued to expand and adjust to meet the evolving needs of scientists.

1.4 Hardwood Genomics Database: a Tripal-based database

Hardwood tree species have great economic and ecological significance. As a source of timber, they are widely used for construction and other purposes like furniture, flooring, musical instruments, tools, and numerous other specialty wood products (Pijut *et al*., 2011; Panayotou and Ashton, 1992; Seth, 2003). Hardwood tree species are a source of sustenance by providing specialty fruit and nut crops, such as apples, citrus, almonds, peaches, walnuts, and more. They are also an important source of medical compounds, and serve as fuel (Omar *et al*., 2000; Seth, 2003). Our urban environments are improved by hardwood trees because they can eliminate pollutants, moderate temperature and noise, and function in recreation and ornamentation (Seth, 2003; Pijut *et al*., 2011). Hardwood trees play a significant role in environmental stabilization and restoration by controlling soil erosion and desertification, and by serving as riparian buffers, windbreaks and watershed protection (Seth, 2003; Allen *et al*., 2001; Jacobs, 2006; Skousen *et al*.,

2009). Organisms at different level of food chain use hardwood trees as habitats, shelters, nesting sites and nutrients (Leopold *et al*., 1998; DeGraaf, 2002; MacGowan, 2003; Seth, 2003; Pijut *et al*., 2011), and they also contribute to nutrient cycling and distribution (Pastor and Bockheim, 1984; Harmon *et al*., 1986).

These invaluable tree species, however, are increasingly facing threats from invasive pests and pathogens as well as climate change (Pimentel *et al*., 2000; Wingfield *et al*., 2010; Loo, 2009; Lovett *et al*., 2006). Approximately 50,000 foreign species have been introduced into the United States (Pimentel *et al*., 2000, 2005), and among these introduced foreign species, insects and pathogens pose the greatest threats to the forest ecosystem (Loo, 2009). Non-native invasive insects and pathogens often cause mortality of hosts and affect forest composition, productivity and nutrient cycling (Moser *et al*., 2009; Gandhi and Herms, 2010). The emerald ash borer (EAB) larvae feed on the bark of ash trees and destroy the phloem tissue. Once the EAB populations are established, all ash trees in that region are likely to become infected and die (McCullough, 2013). Pathogens may fundamentally change an ecosystem by eliminating the dominant tree population (Lovett *et al*., 2006). The pathogenic fungus *Cryphonectria parasitica*, which is the main cause of chestnut blight, led to the widespread death of the American chestnut a century ago (Anagnostakis, 2012).

Compared to the impacts from invasive insects and pathogens, the related impacts from climate change will be damaging through global and species-nonspecific disturbance via inducing drought and heat (Hanson and Weltzin, 2000; Allen *et al*., 2010). The study by Zhao & Steven (Zhao and Running, 2010) indicated that the warming climate resulted in a significant reduction in the global terrestrial Net Primary Production (NPP, the net carbon fixed by a given plant community or ecosystem (Gough, 2012)) from 2000 to 2009. Climate change may also influence

the forest composition (Wang et al., 2016), distribution (Sykes and Prentice, 1996; McKenney *et al*., 2007), timber products (Bateman and Lovett, 2000) and the atmospheric CO_2 level, and therefore the physiological responses of hardwood trees (Ceulemans and Mousseau, 1994). Forests are likely to be exposed to more frequent and severe disturbance in the near future (Dale *et al*., 2016).

In order to maintain and enhance the economic and ecological value of hardwood trees, researchers are actively seeking to identify individuals that are resilient to biotic and abiotic threats or have higher productivity (Schmidtling *et al*., 2004; Neale and Kremer, 2011). Tree improvement or breeding is the repeated cycling of selecting the most desirable trees, breeding or propagating those selected trees, and testing the resulting progeny (Neale and Kremer, 2011). Genetic principles are applied to help manage natural tree populations of high quality, screen trees of desired phenotypes/genotypes for breeding, and, more recently, to genetically modify trees to generate more adaptive, disease or pest resistant trees (Pijut *et al*., 2011). The growing availability of genomics data for hardwood tree species has opened the doors to additional exciting avenues of research and application. For example, profiles of genetic variation of tree populations can be estimated to determine which populations have higher adaptive genetic potential and then used to optimize a species conservation strategy (Neale and Kremer, 2011). Marker-assisted selection (MAS) approaches, already common in other agricultural plant species, can be applied to reduce the breeding cycle time (Collard and Mackill, 2008; Neale and Kremer, 2011). Landscape genomics studies are being used to explore the geographic pattern of genome-wide genetic variation of tree species and therefore to assess a species' ability to respond to predicted environmental change (Sork *et al*., 2013). Association genetic studies are being used to dissect complex traits and identify the individual genetic components that are responsible for phenotypes

of interest (Harper *et al*., 2016; Ćalić *et al*., 2017). Genetic markers that are associated with ecological and economic traits can be used to monitor forest health (Rabbani *et al*., 2003; Harper *et al*., 2016) or for breeding selection (Crossa *et al*., 2010; Stejskal *et al*., 2018).

The successful application of the powerful genetic approaches described above to hardwood trees improvement and breeding largely depends on the availability of high-quality genetic and genomic resources such as genetic markers (e.g., Single Nucleotide Polymorphisms(SNPs), Simple Sequence Repeats (SSRs)), reference genetic and physical maps, genome and transcriptome sequences, and associated gene expression profiles (Pijut *et al*., 2011; Neale and Kremer, 2011). Fortunately, advances in DNA sequencing technologies have made it affordable for single research groups to generate large amounts of valuable genetic resources in a very short time (Reuter *et al*., 2015). Because of these advances, a new challenge for researchers is finding, analyzing and using these existing genetic resources.

The Hardwood Genomics Database (HGD, (https://www.hardwoodgenomics.org/) is a Tripal-based database, hosting genetic and genomic data from 32 economically and phylogenetically important hardwood species. The database provides transcriptomic and genomic data, SSRs markers, genetic and physical maps from those species, a set of BLAST tools, as well as JBrowse (Skinner *et al*., 2009), a genome browser tool. All data in the HGD database are fully accessible to all biological researchers. Associated publications and data provenance are also included in HGD. HGD will be used as an example Tripal database to demonstrate how web-based computational tools have been developed in my studies and could be used to facilitate research by providing data exploration, visualization and analysis solutions.

1.5 Galaxy: turning data into information and knowledge

Although Tripal eases the process of data collection and organization by constructing biological databases efficiently, creating these databases is only one of the very first steps of the data-to-knowledge transformation process. Storing data in a Tripal database helps to make data more discoverable and useable, but researchers rely on additional computational methods and analysis tools to transform data into scientific results and knowledge. With the price drop of high throughput instruments, biologists turn more often to big data to explore the biological systems (Marx, 2013b). However, without training in bioinformatics, manipulating and analyzing a large amount of biological data is not a simple task, and "big data" to biologists can mean more challenges than opportunities.

Galaxy is an open-source, web-based platform that enables researchers without programming expertise to perform computational analyses through web interfaces (Giardine *et al*., 2005; Afgan *et al*., 2016a). Galaxy tools are designed to remove programming from the computational analysis process by collecting user inputs through web interfaces and automatically generating data analysis scripts that are executed by Galaxy on a single server or a large supercomputing cluster. The Galaxy platform has a large set of tools available from the open source Galaxy Toolshed (Blankenberg *et al*., 2014a). The Galaxy ToolShed functions as an application store which hosts over 6,300 galaxy tools (https://toolshed.g2.bx.psu.edu/, accessed by Sep 11, 2018) developed by the whole Galaxy community.

Galaxy also allows for creation of workflows through a graphic user interfaces (GUIs). Workflows for multi-step computational analysis can be easily created by dragging, dropping and connecting a variety of tools in an intuitive web browser-based interface (Afgan *et al*., 2016b; Goecks *et al*., 2010). Workflows and data analysis histories can be easily shared among galaxy

instances, resulting high research reproducibility (Afgan *et al*., 2016c). All these properties make Galaxy a good fit as a data analysis engine for Tripal sites and work by a Tripal developer team is ongoing to implement this.

The Galaxy platform is one of the widely used platforms that enables performing complicated bioinformatics analysis through web interfaces. Other common platforms include the Discovery Environment of CyVerse (formerly known as iPlant Collaborative) (Goff *et al*., 2011) and KBase (Arkin *et al*., 2016). While both the Discovery Environment of CyVerse and KBase have only one instance running, there are ~100 public Galaxy instances being maintained by institutes all over the world (Galaxy Project Stats). Galaxy has a very large user base and a growing number of public instances. During the last five years, the number of registered users on the main Galaxy platform (https://usegalaxy.org/) has increased from ~20,000 to ~120,000 (Galaxy Project Stats). There are over 90 public Galaxy instances maintained by institutes around the world (Galaxy Project Stats).

1.6 Integration of Tripal and Galaxy

The rapid growth in data has led to significant gaps between the potential value of data and the actual knowledge extracted from them (Cao, 2016). These gaps increase further due to the separation of the data from the analytic tools. Many biologists lack the expertise to install and manage tools to perform computationally-intensive data analysis, especially when those tools require use of a Linux high performance computing (HPC) environment and command line knowledge (Afgan *et al*., 2016a).

The integration of Galaxy platform with Tripal databases would enable the coupling of tremendous amounts of data with hundreds of data analysis tools. Associating databases with

appropriate computational tools can save the step of transferring data out of the databases for analysis. For example, with integrated sequence mapping and assembly tools, you can run mapping and assembly jobs without downloading the reference genomes and raw reads to your local computer, which usually are very large datasets. This allows for scientists to focus on the data analysis processes, instead of spending time on tool installation and administration. This is especially important to biologists when administration and deployment of some tools requires a high level of computational resources and expertise in computer science.

To achieve the goal of integration of Galaxy and Tripal systems, researchers from the Tripal core team have developed the *blend4php* library (Wytko *et al*., 2017) and the Tripal Galaxy module. Galaxy currently provides web services for remote construction and execution of workflows without use of the Galaxy web interfaces. The *blend4php* library provides an PHP interface to the Galaxy web services APIs so that Tripal can use it to interact with Galaxy. The open source Tripal Galaxy module is a Tripal extension module that uses this API to allow invocation of Galaxy workflows through a Tripal website, such as HGD.

Figure 1.3 outlines how Tripal and Galaxy work together to enable HGD users to efficiently perform workflows through a Tripal site. When an HGD user visits the database, the user can collect data from the database or upload local data. After necessary input data has been collected, the user can invoke a workflow from in a connected Galaxy instance through the Tripal Galaxy module. Once the workflow execution completes in Galaxy, the HGD user will be able to view the analysis results on the HGD website and download the results to their local computer.

1.7 Overview of research goals

1.7.1 Storage, display and visualization of gene expression analysis data

RNA-Seq technology has a wide range of application scenarios and become an increasing choice for researchers to study transcriptome. Since the introduction of high throughput RNA sequencing, RNA-Seq have been applied to many aspects of transcriptome studies. RNA-Seq can be combined with genome-wide identification of transcription start sites to identify potential riboswitches (Rosinski-Chupin *et al*., 2014). Mapping RNA-Seq reads to a reference genome enables identification of variants at single nucleotide resolution and therefore detection of allele-specific expression (Pirinen *et al*., 2015). RNA-Seq reads can be used to identify fusion genes. Discordant read pairs that are mapped to two independent genes indicate locations of fusion genes (Li *et al*., 2011). The broad application of RNA-Seq techniques, continued technical improvements, and dramatic decrease in sequencing cost together have led to a sharp increase in transcriptome data in the past decade. In order to make full use of RNA-Seq and transcriptomics data, Tripal have extended functionality in recent years to provide support for storage, display, and visualization of genes, assembled transcript sequences and their functional annotation.

In addition to the series of important applications mentioned above, RNA-Seq is used to profile gene expression and identify differentially expressed genes across different biological conditions, such as different tissues, disease states, and developmental stages (Soneson and Delorenzi, 2013; Anders and Huber, 2010). The count of RNA-Seq reads that are mapped to a given gene is used as a direct representation of the gene expression level. Statistical methods are applied to normalize across libraries and estimate the significance of the read count differences across biological samples (Soneson and Delorenzi, 2013; Li *et al*., 2015; Kharchenko *et al*., 2014;

Bullard *et al*., 2010). Although Tripal core and existing extension modules have provided support for many kinds of biological data types (see table 2.1), a robust solution for displaying and visualizing gene expression data, and sample metadata is still lacking in Tripal.

1.7.2 The need for efficient search in Tripal databases

Without search engines, the value of information stored in the internet would be largely undiscoverable due to its enormous size and heterogeneous nature. The rapid growth of data brings more information as well as more challenges to access useful information. Online biological databases have the same problem: freely available data does not necessarily mean accessible data unless scientists can efficiently identify and access the data they really need.

Efficient search relies on indexing, a method of pre-sorting, parsing and storing a very large amount of data in a format that enables very fast lookup and retrieval. Drupal offers its own indexing and searching system, however, it only indexes data inside its own database. This means information that has been placed in Chado, the standard biological database, is not searchable. For websites functioning as portals to a Chado database, Drupal's search system is also not meant for massive scientific databases, leading to poor performance. To access any data in a Drupal/Tripal based website, we need a new module that can index any Drupal or Chado table in an efficient manner, and provide flexible, customizable interfaces for searching the data.

1.7.3 Data analysis transparency and results reporting in Galaxy tools

Any command line tool can be "wrapped" for use in Galaxy tool by writing an Extensible Markup Language (XML) specification file, which is used by Galaxy to build the web interface layer. The web interface accepts user inputs, translates them into the properly formatted command line application, and runs the command as a shell script. The user controls the input values and

executes analytics through web interfaces without writing the actual shell scripts. When the analysis jobs are complete, results are returned to the user via web pages and downloadable file links.

While the web interface layer removes programming obstacles from data analysis and simplifies the process for non-programming users, it also hides the analysis logic from them. The scripts used by the tool and how they are executed is usually invisible to end users. Exposing the data analysis logic to users increases the transparency and reproducibility of the process. Additionally, most Galaxy tools generate outputs in a variety of formats and render them individually to users. The ability to build a more intuitive, self-explanatory, and well-organized analysis report would be beneficial to users, as an addition to the raw text files or other output files rendered by the software.

1.7.4 Efficiency in Galaxy tool development

Galaxy uses an XML based development framework to define Galaxy tools. Currently there are over 70 XML components available to define all kinds of fields in the web interface of a tool. Additionally, each component has many attributes that are used to specify the properties of the field. When developing a Galaxy tool, developers need to reference the XML definition documentation many times to obtain the valid components and their attributes. It would be more efficient to develop Galaxy tools if there existed an application to provide predefined XML components as a selectable list and allow specifying values of component attributes through web forms. In this situation, tool developers would not need to worry about syntax errors when developing Galaxy tools. This would also allow bioinformaticians who are not familiar with the XML syntax, and have minimum knowledge of Galaxy tool development, to contribute tools.

1.8 Overview of technologies

The research approaches described in subsequent chapters build on many different web technologies, the most important of which are briefly introduced below.

1.8.1 Drupal and Tripal software

I utilize the Drupal and Tripal APIs to extend Tripal to support storage, display, and visualization of data and information from expression experiments, as well as provide better search functionality in Tripal databases. The developer's guide for both Drupal and Tripal can be retrieved from the official project websites: https://www.drupal.org/ (Drupal) and http://tripal.info/ (Tripal).

1.8.2 Elasticsearch

Elasticsearch is a distributed, full-text search engine with HTTP web interfaces and schema-free JSON documents. It is written in Java and was first released by Shay Banon in 2010 (Divya and Goyal, 2013; Banon, 2011). Elasticsearch clients are available for many programming languages, including PHP, the language of Drupal and Tripal.

Elasticsearch is a popular open source search solution for very large datasets. It offers a fast and incisive search against large volumes of data and is broadly distributable and highly scalable (Divya and Goyal, 2013). An Elasticsearch cluster can consist of nodes that are hosted at different servers, and each node can be easily added or removed to adjust the performance of the cluster. Data stored in one Elasticsearch cluster can be communicated to other clusters with RESTful APIs. These features have led to broad adoption by industrial companies and government institutes. For example, the telecommunications company Sprint uses Elasticsearch to power the search for retail operation insights, and the U.S. Geological Survey uses the real-time search functionality of Elasticsearch combined with analysis of hundreds of millions of 'earthquake'

tweets to detect earthquakes (Langseth et al., 2014). Additionally, there are also many successful use cases of Elasticsearch in education and research: researchers from Yale University use Elasticsearch to power their research database and help clinicians and researchers to identify novel causes of cancer and potential therapeutic targets (www.elastic.co/blog, accessed by Oct 31, 2018). Xin et al. (2016) built high-performance web services with Elasticsearch for querying gene and variant annotation (Xin *et al*., 2016).

1.8.3 R Markdown

R Markdown is a file format that is used by the R package *rmarkdown* to generate dynamic documents, presentations and reports (Xie, 2015). An R Markdown document is written in a lightweight markup language called *markdown* (Gruber, 2004) and contains chunks of embedded R code (Team, 2015) (Figure 1.4 A). When an R Markdown document is rendered with the *rmarkdown* package, the markdown plain text is converted to a structurally valid HTML, PDF or MS Word document. The chunks of embedded code in the document will be executed and the results of the code will be appended to the document next to the code chunk (Figure 1.4 (B)) by default. The *rmarkdown* package provides many functions and parameters that allow users to specify whether they want the code to be evaluated, where the code results will be placed in the document, and the format and style of the document.

R Markdown has been used as a technique to generate data analysis reports and tutorial-like documents, especially documents involving computational code, because embedded code can be executed and displayed along with results in the rendered document. For example, Blattmann et al. (2016) generated instruction documents with R Markdown to explain how to use their R package SWATH2stats. Callahan et al. (2016) used R Markdown to develop a reproducible research workflow for analyzing personalized human microbiome data. In addition to R code, code

written in many other languages, including Bash, Python, SQL, Javascript and more, can be embedded and executed in an R Markdown document. The features of increasing analysis transparency, increasing reproducibility, and summarizing results in a single document make R Markdown a perfect solution for generating analysis reports for Galaxy tools.

1.8.4 Drupal webform module

The Drupal webform module is a Drupal extension for making forms and surveys through web interfaces. Without this module, website developers would need to use the Drupal form APIs to generate HTML content which is then rendered as form elements. The webform module provides a set of predefined components that match with form elements. Website developers select webform components (Figure 1.5(A)) to build the form elements (Figure 1.5(B)). The forms are highly customizable, including the ability for developers to drag-and-drop webform components to arrange their structure and location. Webform APIs are also available for developers to create new webform components and therefore new, custom form elements.

1.8.5 Docker

Docker provides a solution to the problem of how to reliably run software across multiple computing environments. It is a powerful system for building "containers", which are virtualized computing systems that carry their own software packages and configurations. Docker is one among a number of container technologies, which recently found niches in bioinformatics analyses (Di Tommaso *et al*., 2015; Schulz *et al*., 2016). With container technologies, bioinformatics applications and pipelines can be wrapped into an isolated, self-contained environment, and distributed to and run on independent operating systems. This process eliminates the steps of installation, configuration and deployment of bioinformatics tools and enables researchers to focus

on the tool application. A well-maintained Galaxy Docker container is available (Grüning, 2018). I utilized the docker container technology and this Galaxy Docker container to build an interactive, easy-to-use development environment for Galaxy tool development.

1.9 Appendix

Table 1.1. A list of currently known Tripal databases from http://tripal.info.

Databases	Stored Data
Banana Genome Hub (http://banana-genome-hub.southgreen.fr/)	Musa Genomes, Genetic Markers, Primers
Cacao Genome Database (http://www.cacaogenomedb.org/)	Genomics, Genetics and Breeding resource for Cacao
Citrus Genome Database (https://www.citrusgenomedb.org/)	Resources for citrus genomics, genetics, breeding and disease research
Cool Season Food Legume Database (https://www.coolseasonfoodlegume.org/)	Genomic, genetic and breeding resources for pea, lentil, chickpea and Faba bean crop
CottenGen (https://www.cottongen.org/)	Genomic, genetic and breeding resources for Cotten
GeneNet Engine (http://sysbio.genome.clemson.edu/)	A network biology resource for genotype-phenotype relationship and biomarker discovery
Genome Database for Rosaceae (https://www.rosaceae.org/)	Genomic and genetic resources for Rosaceae
Genome Database for Vaccinium (https://www.vaccinium.org/)	Genomic, genetic and breeding resources for blueberry, cranberry and other Vaccinium sp.
Hardwood Genomics Database (http://www.hardwoodgenomics.org/)	Genomics and genetics resources for hardwood tree species
Planosphere (https://planosphere.stowers.org/)	Schmidtea mediterranea molecular staging resources and atlas
KnowPulse (http://knowpulse2.usask.ca)	Pulse crop breeding and genetics resources
Legume Information System (https://legumeinfo.org/)	Genetic, genomic and trait data across legume species
Musa Germplasm Information System (https://www.crop-diversity.org/mgis/)	Genomic, genetic and breeding resources for Musa
TreeGenes (https://treegenesdb.org/)	Genetic, phenotypic and environment data for forest trees
PeanutBase (https://peanutbase.org/)	Genetic and genomic resources for peanut

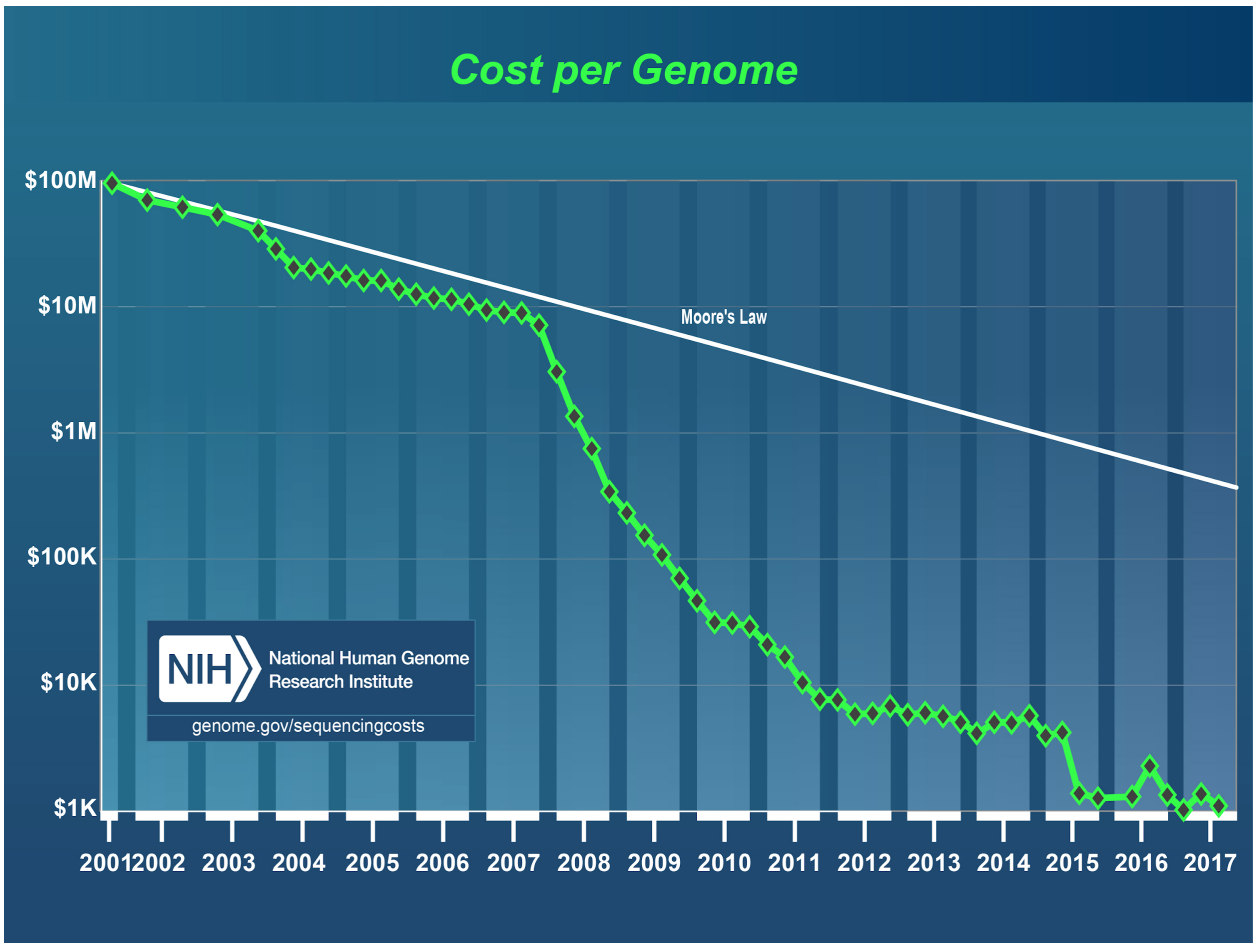


Figure 1.1 Relationship between sequencing costs and time (Wetterstrand, 2016).

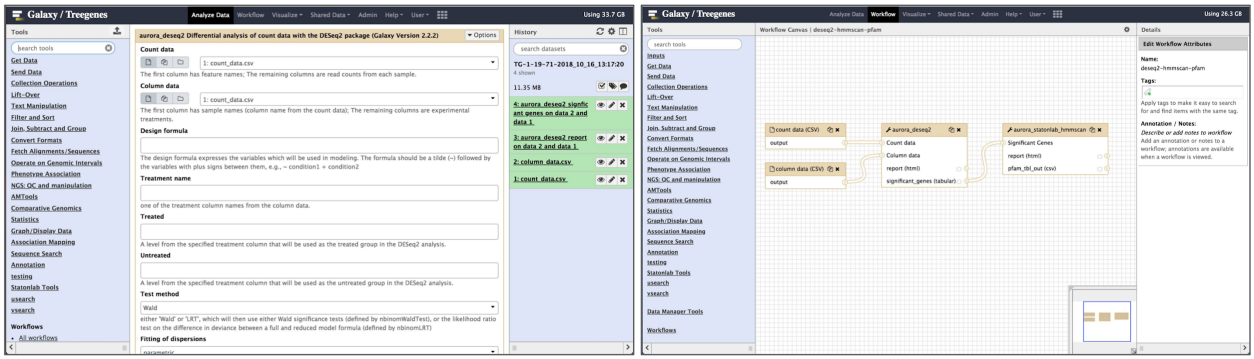


Figure 1.2 Galaxy user interface and graphical workflow editor.

(A) Galaxy analysis interface consists of the tool menu (left pane), tool interface (center pane), and history interface (right pane). (B) Galaxy's graphical workflow editor.

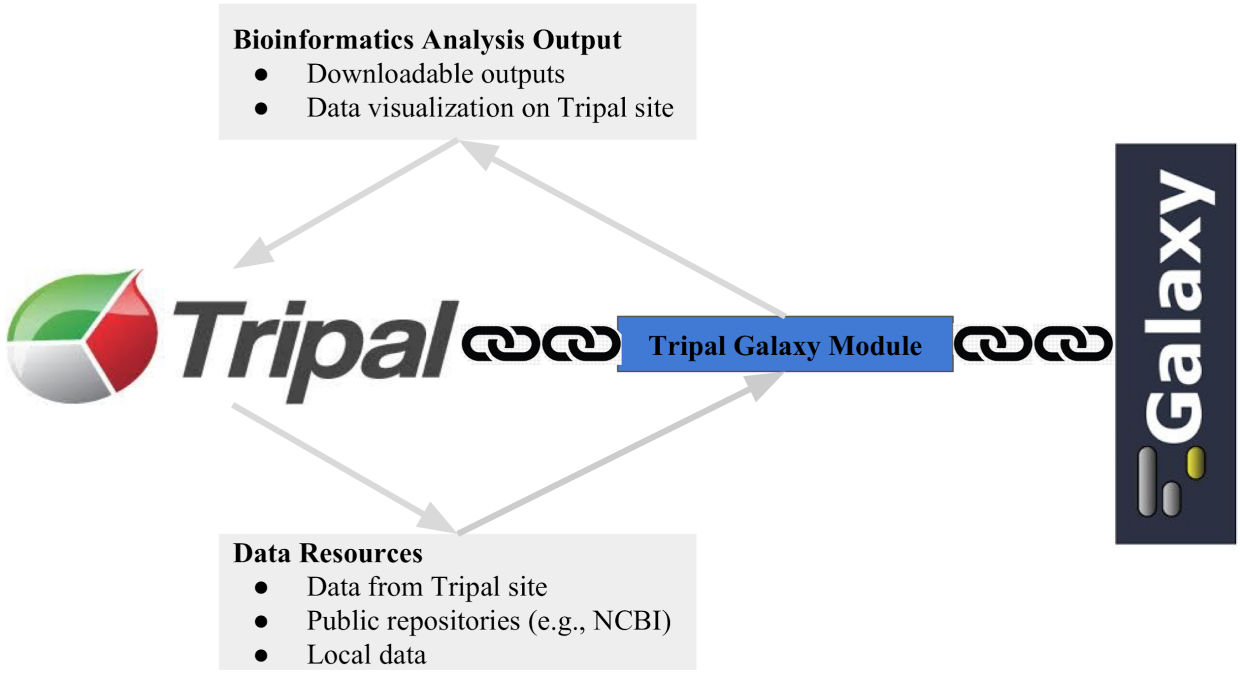


Figure 1.3 Tripal Galaxy Module as a Tripal gateway to Galaxy.

A

```
---
output: html_document
---

This is an R Markdown document. Markdown is a
simple formatting syntax for authoring HTML, P
DF, and MS Word documents. For more details on
using R Markdown see .

When you click the **Knit** button a document
 will be generated that includes both content
 as well as the output of any embedded R code
 chunks within the document. You can embed an
 R code chunk like this:

```{r}
summary(cars)
```

You can also embed plots, for example:

```{r, echo=FALSE}
plot(cars)
```

Note that the `echo = FALSE` parameter was add
ed to the code chunk to prevent printing of th
e R code that generated the plot.
```

B

R Markdown

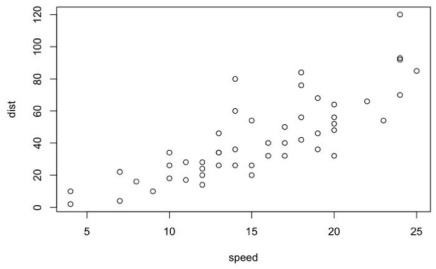
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2
##  1st Qu.:12.0   1st Qu.: 26
##  Median :15.0   Median : 36
##  Mean   :15.4   Mean   : 43
##  3rd Qu.:19.0   3rd Qu.: 56
##  Max.   :25.0   Max.   :120
```

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Figure 1.4 R Markdown document (A) and R Markdown rendered as a HTML file (B).

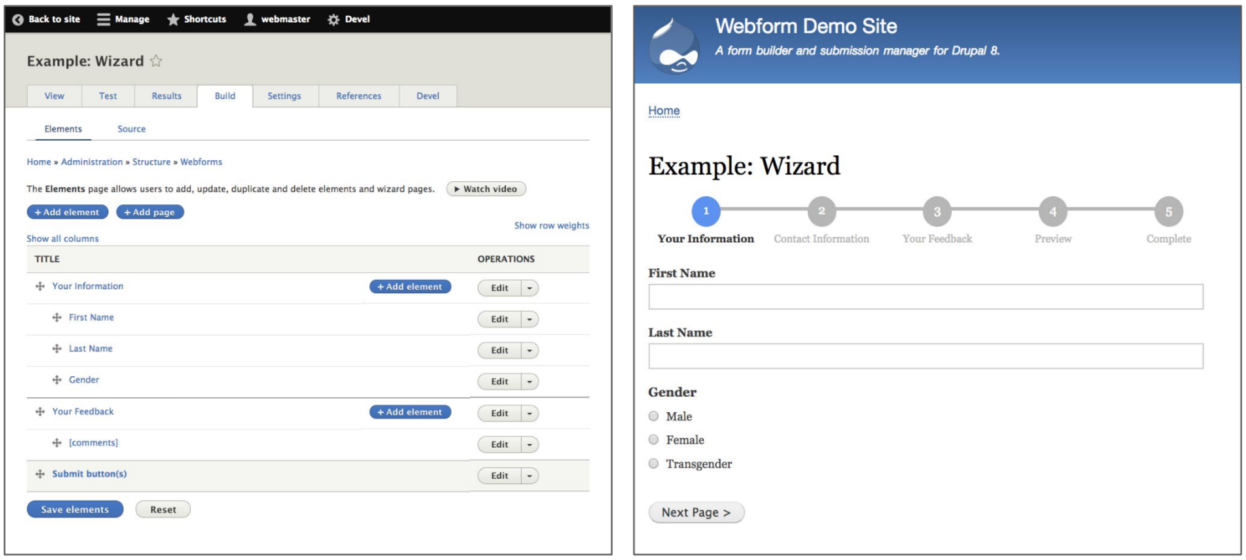


Figure 1.5 Drupal webform interface (A) and example form (B).

CHAPTER II. NEW EXTENSION SOFTWARE MODULES TO ENHANCE SEARCHING AND DISPLAY OF TRANSCRIPTOME DATA IN TRIPAL DATABASES

A version of this chapter was originally published by Chen et al. in *Database*:

Chen, M., Henry, N., Almsaeed, A., Zhou, X., Wegrzyn, J., Ficklin, S. and Staton, M., 2017. **New extension software modules to enhance searching and display of transcriptome data in Tripal databases**. Database, 2017.

2.1 Abstract

Tripal is an open source software package for developing biological databases with a focus on genetic and genomic data. It consists of a set of core modules that deliver essential functions for loading and displaying data records and associated attributes including organisms, sequence features, and genetic markers. Beyond the core modules, community members are encouraged to contribute extension modules to build on the Tripal core and to customize Tripal for individual community needs. To expand the utility of the Tripal software system, particularly for RNA-Seq data, we developed two new extension modules. Tripal Elasticsearch enables fast, scalable searching of the entire content of a Tripal site as well as the construction of customized advanced searches of specific data types. We demonstrate the use of this module for searching assembled transcripts by functional annotation. A second module, Tripal Analysis Expression, houses and displays records from gene expression assays such as RNA sequencing. This includes biological source materials (biomaterials), gene expression values, and protocols used to generate the data. In the case of an RNA-Seq experiment, this would reflect the individual organisms and tissues used to produce sequencing libraries, the normalized gene expression values derived from the RNA-Seq data analysis and a description of the software or code used to generate the expression values. The module will load data from common flat file formats including standard NCBI Biosample XML. Data loading, display options and other configurations can be controlled by

authorized users in the Drupal administrative backend. Both modules are open source, include usage documentation, and can be found in the Tripal organization's GitHub repository.

Software download URL:

Tripal Elasticsearch module: https://github.com/tripal/tripal_elasticsearch

Tripal Analysis Expression module: https://github.com/tripal/tripal_analysis_expression

2.2 Introduction

Tripal is an open source toolkit for construction of online genome databases (Ficklin *et al*., 2011). Tripal is built on the Drupal content management system (www.drupal.org) and stores data in the recommended standardized biological database schema, Chado(Mungall *et al*., 2007). Both Tripal and Chado are members of the Generic Model Organism Database (GMOD) collection of open source and interoperable software tools (http://gmod.org). Tripal consists of a core set of modules that encompass a variety of common biological and genetic data types, such as organisms, sequence features and genetic markers which include attributes such as genus and species, nucleotide residues and marker locations, respectively. The software has been deployed for numerous genetic and biological data websites, with over 15 active sites listed on the Tripal information web page (http://tripal.info/sites_using_tripal). The adoption of this standardized web database software platform has numerous advantages, including leveraging code developed across different groups, standardization of data storage formats, and an active mailing list and developer community for support.

The Tripal community encourages individual developers to extend and customize the software and provides helpful tools for this, including an application programming interface (API) and flexible data display templates. By leveraging these features, user-contributed extension

modules provide additional functionality, including the upload and display of new data types such as transcriptome assemblies by Tripal Analysis Unigene (https://github.com/tripal/tripal_analysis_unigene), BLAST sequence similarity results by Tripal Analysis BLAST (https://github.com/tripal/tripal_analysis_blast), and Gene Ontology (GO) annotations through Tripal Analysis GO (https://github.com/tripal/tripal_analysis_go). The Tripal.info website provides a system for developers to register their extension modules, with descriptions of functionality, development status (from in-development to ready-for-use), compatible versions of the Tripal core, and compatible versions of the Chado (http://tripal.info/extensions). Developers are also able to join the Tripal organization on GitHub (https://github.com/tripal) and contribute their modules in the centralized Tripal repository, thus encouraging discoverability, collaboration and communication among developer groups.

Despite the rich functionality already available, including storage of gene or assembled transcript sequences and their functional annotation, Tripal does not yet have robust search, storage or display of expression information or the biological samples queried in expression studies. As next generation sequencing instruments with high-throughput and low cost became available in 2005 and widely adopted by 2009, RNA sequencing (RNA-Seq) has become the most common method for assaying gene expression (Lister *et al*., 2009; Wilhelm and Josette-Renée, 2009; Todd *et al*., 2016). This is reflected by the rising percentage of RNA records in public databases. In February of 2017, the NCBI sequence read archive (SRA) contains 518,625 records for RNA sequencing runs from Eukaryotes, representing 46% of the total Eukaryote sequence records.

A gene expression experiment seeks to quantify changes in transcript abundance across different samples; this may take the form of a broad survey (gene atlas) across many tissues, time points or developmental stages, or as a targeted profiling of changes induced by an experimental

treatment, often abiotic or biotic stress (Martin *et al*., 2013; Wang *et al*., 2009; Marguerat and Bähler, 2009). To increase the reusability of data and the comparison of data across different experiments, community databases need to capture as much metadata as possible about these biomaterials(Wegrzyn *et al*., 2012; Wilkinson *et al*., 2016). Once RNA is extracted and sequencing performed, a standard bioinformatics analysis incorporates sequence quality control steps such as adapter removal, *de novo* assembly if a reference genome is not available, mapping of reads to the reference assembly or genome, quantification of reads per gene or transcript, and normalization of data (Oshlack *et al*., 2010; Conesa *et al*., 2016). These normalized values are comparable across biomaterials and often visualized as a heatmap for interpretation. Numerous software tools are available for each step of the analysis, and the choice of tool software, version and user-specified parameters influences final results(Fonseca *et al*., 2014; Vijay *et al*., 2013). Preserving methodological details is critical for appropriate data interpretation and to enable reuse, reproducibility, and comparison among datasets.

For a community genome database, users can mine gene expression data to identify candidate genes or transcripts for further study based on information about expression patterns across different tissues and conditions. To further this goal, we developed two new publicly available Tripal extension modules. The Elasticsearch module enables more efficient and flexible searching powered by the Elasticsearch software (https://www.elastic.co). With a broad site-wide search, users can enter any keyword, from a software tool name to a transcript name to a metabolic function, to find results across the entire Tripal site. Further, database developers may build customized search interfaces for specific data types with advanced filtering options. A second Tripal module, the Tripal Analysis Expression module, has been developed to store and display gene expression levels across multiple tissues and conditions derived from sequencing

transcriptome data. For each sequencing sample, a biomaterial record is created and all attributes such as individual, tissue and experimental condition are added as ontology-tagged, searchable fields. Normalized expression values for each transcript or gene from individual biomaterial are stored. The resulting user interface of a bar chart or heatmap provides an intuitive visual representation of how genes or transcripts are expressed in an organism, providing important clues about biological function. Together, the modules provide significant additional functionality for users to explore RNA-Seq data, transcriptome assemblies, and transcript expression levels.

The code for both modules is available via GitHub. They are currently in use at the Hardwood Genomics Web (http://hardwoodgenomics.org), a database that houses genetic and genomic data from hardwood tree species. Both modules are build in PHP and work with Drupal 7.x as an extension to Tripal version 2.x. Linux is the recommended operating system for Drupal installations. The modules currently do not support Drupal version (8), as the core Tripal modules have not yet transitioned, however, Drupal 7 is in long term support mode by the Drupal community. The modules are compatible with Chado versions from 1.2 to 1.31. Source code and developer guides are open source and made available through GitHub under the GNU General Public License 3. We welcome developer and user feedback, including bug reports and questions, through the GitHub issues queue.

2.3 Tripal Elasticsearch Module

2.3.1 Overview

The Drupal content management system, which forms the base for all Tripal sites, provides its own built-in search. However, this system suffers from several disadvantages that render it either slow or unusable for some Tripal databases. First, the Drupal search system only indexes

the HTML-rendered text content for Drupal nodes; in Drupal, nodes are individual pieces of content rendered as a web page. This is a major impediment to intelligent searching of a Tripal site, where the biological data that is stored in the Chado database schema is ignored unless it is part of a node. Even when data is displayed as part of a node, the biological context of the data may be lost. The native Drupal search also does not offer fuzzy or partial word matching. Finally, the indexing procedure for searching is very inefficient for sites with hundreds of thousands of records, leading to very slow indexing and searching for Tripal sites dealing with very large numbers of records.

In order to provide site-wide searching functionality and flexible, efficient access to the data hosted in Chado, we developed the Tripal Elasticsearch module to integrate the Elasticsearch engine with Tripal sites. The Tripal Elasticsearch module leverages this search engine to provide both generic site-wide searching and the ability to build customized search interfaces for specific Chado database tables (Figure 2.1). Written in Java, Elasticsearch is open source software that provides a powerful, fast, and feature-rich search engine. Elasticsearch clients are available in a variety of languages, including PHP, the language of Drupal and Tripal (https://github.com/elastic/elasticsearch-php). Elasticsearch is designed to be used in distributed environments and thus is highly scalable, a critical concern with the quickly growing amount of available genomic data for many organisms. Elasticsearch indexes data in schema-free JSON documents that are stored on the web server and enable very fast look-up of search terms. The index files can be split into pieces, called shards, across many servers, and support is provided to search and index multiple source databases. The Tripal Elasticsearch module opts to use the built-in query_string query method provided by Elasticsearch, which provides common search methods

such as AND/OR/NOT operators, wildcard (*) search, phrase search with quotes, and +/- (include/exclude) operators.

The Tripal Elasticsearch module utilizes the Elasticsearch-PHP client to interact with the Elasticsearch cluster. For site-wide indexing, the full Hyper Text Markup Language (HTML) contents of all pages on the site are indexed. This is accomplished by first querying the Drupal database for all node IDs, as all individual web pages are stored as nodes in Drupal 7. The module uses these IDs to query the site and extract the HTML contents for each page. The HTML strings are processed and, using the Elasticsearch-PHP client, turned into Elasticsearch indexes. For table-specific indexing, the process is slightly different. Instead of querying HTML, the module extracts data content from the specific Chado database tables and fields specified by the administrative user, and then uses the Elasticsearch-PHP client to create an Elasticsearch index specific to this dataset. A customized form and content search block are automatically created for this index, with the administrator able to customize the form fields, the placement of the content block on the website, and the URLs to link the results to appropriate pages.

2.3.2 Administration interface

The Elasticsearch software can be installed on the same server as the Tripal site or on a remote host or cloud host. After this installation, the remaining setup can be controlled entirely through the graphical user interface (GUI) administrative backend provided by the Tripal Elasticsearch module. The module can be downloaded and enabled through Drupal in the same way as other Tripal extension modules. Next, the site administrator can connect the module to the local or remote Elasticsearch cluster through an administrative page. After a successful connection, health information of the cluster will be displayed (Figure 2A).

Next, site administrators can select the type of database content they would like to index and create cron jobs that will create those indexes. Administrators can also delete the indexes. One option is to create a site-wide index, where the module extracts the HTML source code of all published Drupal nodes, excluding sequence strings, and then indexes these html strings using the Elasticsearch engine. By doing this, all published web pages become indexed and searchable, regardless of content type or location of the underlying data in the database. The module also implements Drupal insert, delete, and update function hooks so that any modification to site content will be detected by the module and corresponding modifications to the Elasticsearch indexes will be made. The required modifications - adding, removing or altering individual page indexes - are submitted as jobs to the cron queue.

As an alternative to the site-wide search, the administrative page pre-populates a drop-down menu box with all tables from both the Drupal and Chado database schemas, allowing the administrator to select one for individual indexing (Figure 2B). The module will automatically detect which tables have been previously indexed and warn the user to prevent multiple indexes of the same data. To further refine administrator control of indexing, individual fields from a table may be included or excluded from the index. This individual table indexing is useful to design advanced search forms for subsets of data in the database or to incorporate searching of content in Chado that is not displayed in Drupal nodes.

Both site-wide and individual table indexing can be time consuming for large datasets, so the Tripal Elasticsearch module enables concurrent indexing jobs. By leveraging the cron queue functionality in Drupal (https://api.drupal.org/api/drupal/modules!system!system.api.php/function/hook_cron_queue_info/7.x), administrative users can opt to use up to ten simultaneous queues for the indexing process.

Tripal Elasticsearch module also requires the Drupal module Queue UI, which can be used to monitor and manage cron queues (https://www.drupal.org/project/queue_ui). Indexing jobs are then divided and assigned evenly across the specified number of cron queues, and the queues are managed independently and concurrently by the cron daemon. Using more queues can largely decrease the time of indexing, but it also uses more memory and processors. The number of queues should be selected judiciously based on available server resources. One successful strategy to prevent interference with the responsiveness of a live production site is to run the indexing on a development server, then move the Elasticsearch index files to the primary web server.

After a table is selected and indexed, the administrator can build a search block to display to site users (Figure 2C). A Drupal block is a flexible unit of content that can be configured to display on all site pages or only on certain pages, and it can be placed in specific page locations such as a sidebar or header. Next, the administrator must enter information about how individual search result columns should link to content (Figure 2D). The search results themselves are available as tokens for building a URL to appropriately link each search result. For example, a gene search may return a table with the gene name and its organism. Each of these fields may be linked to the most appropriate place; i.e. the gene name to its feature page and the organism name to the organism page. Once the block and results links are built, the final step is to customize the search form if needed. The administrative user can select which of the indexed fields to expose as search boxes, the user input type (enter free text or select options from a dropdown), the field label, and the order of the fields in the display (Figure 2E).

2.3.3 User interface

The Tripal Elasticsearch module provides a simple input box for the generic site-wide search (Figure 3A) and the administrator-customized form for a table specific search (Figure 3B) .

While the Elasticsearch software provides over a hundred searching methods, the Tripal Elasticsearch module uses a comprehensive search method called *query_string* as its default search method. With this search method, users can build advanced queries, for example by enclosing exact search phrases with quotes, by prefixing words with '+' or '-' to require or exclude from results, and by using *AND*, *OR,* and *NOT* operators. For better performance only the first 100 search results are returned by the generic site-wide search by default, but this can be altered by the site administrator. Search results are displayed with a web page title link and a portion of web page content with keywords in bold and italics, to provide context for the matching keyword. For a table-specific data search, 1000 search results are returned by default. Search results are displayed in a paginated table and can be sorted by clicking the table headers. A download button is provided to download all search results as a comma-separated value (csv) file.

Like the search forms which are embedded in a Drupal block, search results from individual tables are also displayed in independent Drupal blocks. This makes it possible for the Tripal Elasticsearch module to display aggregated data, and to functional as a faster alternative to Drupal Views, a widely-used core Drupal module. Administrative users can build a custom data view by using the same procedure for building a custom search block for desired data and then disabling the search form blocks. The populated result block will remain and may be display on any page in the site.

2.3.4 Example of customized transcript search in HGD using materialized views

As a highly normalized schema, Chado often stores related data across many different tables. To speed page rendering, administrators often use materialized views. Materialized views are tables built to hold the results of a common database query, often a query that joins multiple tables or filters data. By storing the query results in a materialized view, the results do not have to

be rebuilt for new page loads and can be quickly returned. The Tripal core includes a module for administrators to create, populate and update materialized views. The Tripal Elasticsearch module only indexes and creates a search block from a single table, so a materialized view is useful for building an advanced search interface that searches content normally distributed across multiple tables. For example, a site with RNA-Seq data may provide a search form that incorporates searching by transcript unique name, transcript functional annotation, and organism. In this example (Figure 3B), the materialized view is created to aggregate data from three database tables: *chado.organism*, *chado.blast_hit_data*, and *chado.feature*. The Tripal Elasticsearch module can be used to index the new materialized view and build a search form with the requisite custom fields. This transcript search example could easily be extended to include other types of functional annotation, such as KEGG results, gene ontology terms, phenotype associations and more. Once the custom materialized view is indexed by the module, it can be deleted from the database to reduce redundancy and minimize filesystem storage space.

2.4 Tripal Analysis Expression Module

2.4.1 Overview of functionality

Tripal has existing support for aspects of RNA sequencing experiments, including assembled transcripts and associated *in silico* functional annotation from BLAST (Altschul *et al*., 1997), InterProScan (Jones *et al*., 2014) and KEGG (Kanehisa and Goto, 2000) analysis. The Tripal Analysis Expression module expands this support by enabling storage and display of a gene expression experiment, including a description of the experiment, the biological samples (biomaterials), and the normalized gene expression values (Table 1). The module is able to accept either RNA-Seq or microarray expression assays (Figure 4). To build a gene expression analysis

record, an administrator must first add biomaterial records. At a minimum, the biomaterial record must hold a name, a description, and a provider contact, and it must be linked to an organism. However, this content type is flexible and expandable in order to store the many different possible types of metadata that may be associated with a biomaterial. Customizable property fields associated with a controlled vocabulary may also be added to fully describe a record (Figure 5A). This could include information such as tissue, treatment, life stage, or geographic location. The list of properties may be expanded to suit the needs of each experiment by importing new controlled vocabularies or adding new custom controlled vocabulary terms. Both options are available through the Tripal core. The module also provides for database cross referencing, where records link out to other databases such as NCBI's Biosample database(Barrett *et al*., 2012). The biomaterial content type may be used in a Tripal site independently without adding information about an expression experiment. It could be useful for describing biomaterials for other related-omics experiments such as genotyping, DNA sequencing, proteomics or metabolomics.

After biomaterials are established, a full gene expression experiment may be described, starting by creating a new expression analysis record. The Tripal core already provides a generic analysis content type to represents a bioinformatic analysis of a dataset. An analysis record is useful for linking all data in the database back to the relevant materials and methods used to generate the data. To customize this for a gene expression experiment, the Tripal Analysis Expression module extends the analysis content type and adds fields for laboratory and data analysis protocols. If a microarray was used, there is an option to specify the array platform and design. It also enables the upload of the normalized gene expression data associated with the gene expression analysis record. In the database, this links each expression value to an individual feature (a gene, a transcript or other sequence) and to a biomaterial.

2.4.2 Administration interface

After installing the Tripal Analysis Expression module, administrators or curators can utilize web forms to create new biomaterials and expression analysis content. For large sets of data, bulk import is also available. For biomaterials the provided data loader can parse BioSample records downloaded from NCBI in XML format(Barrett *et al*., 2012) or derive data from a tab-delimited text file (Figure 5B). The data is stored into the Chado biomaterial tables (biomaterial, biomaterial_dbxref, biomaterialprop, etc). Another data loader provides the ability to enter expression values into the Chado schema from common expression data formats (Figure 5C). The first option is column format, where a folder with individual files is provided with each file corresponding to a biomaterial and containing one gene per line, followed by a space or tab and the numerical expression value. A second option is to provide a large tab-separated value file with gene names beginning each line and columns of values corresponding to biomaterials. Examples of both file formats are provided within the module documentation. The module assumes that expression values are normalized prior to entry and no statistical manipulation of the data is performed. In an effort to maintain compatibility with current and past databases with the Chado schema, the Tripal Analysis Expression module does not create any new database tables. Instead the module utilizes existing Chado tables from the MAGE (microarray gene expression), organism, contact, sequence, and companalysis table groups(Mungall *et al*., 2007).

The Tripal Analysis Expression module provides full administrative functionality for each content type. The site administrator can sync, delete, and alter settings for each content type provided by the module. The Tripal Analysis Expression module is completely separate from the Tripal Elasticsearch module; some sites may choose to use one but not the other. To account for

this possibility, standard Drupal views-based searches are automatically provided for biomaterials and gene expression analysis records.

2.4.3 User interface

Users can access biomaterials associated with an organism directly from the organism page in a Tripal site. The Tripal Analysis Expression module interacts with the organism module using the Drupal hook system and uses the standard Tripal page template to add a link to biomaterials on the organism page sidebar. When a user clicks on this side menu link, a list of biomaterials is displayed, with multiple pages if the list is longer than ten. Clicking through to an individual biomaterial provides information including all associated metadata properties and external database references. All page templates are fully customizable and site developers may override defaults to alter menu items, link locations, paging options, etc. Further, if a site administrator wishes to create a page dedicated to exploring all biomaterial records outside of the organism page, a dedicated page may be created with Drupal Views or Tripal Elasticsearch.

To explore gene expression values, the Tripal Analysis Expression module also hooks into the Tripal Feature module. For each feature with expression data, a link labeled “Expression” is added to the side menu on the feature page. Clicking on this link generates a one dimensional heatmap figure using the JavaScript D3 library (https://d3js.org/). The user has the option of altering the heatmap figure by removing biomaterial libraries that do not express the feature or sorting the biomaterials by expression value. The expression figure may be displayed as a bar chart (Figure 6A) or as a one-dimensional heat map (Figure 6B). The biomaterial names displayed on the x-axis are usually short identifiers, so if the user hovers their mouse pointer over a biomaterial, a pop-up display with the full biomaterial description appears.

The module also provides a tool for users to enter a list of feature names (Figure 6C) and display their expression data as a two-dimensional heatmap (Figure 6D). This tool comes embedded in a Drupal content block that may be placed on any page or embedded in its own page. Users can directly enter a list of feature unique names or IDs of interest, or obtain features via transcript or feature search interfaces that are built by applying the Tripal Elasticsearch module. The heatmap is created with the Plotly javascript graphing library (https://plot.ly/javascript/) which provides interactive tools enabling the user to pan, zoom, and download the image inside the browser window. Users may perform further image exploration by selecting the "save and edit plot in cloud", which loads their current image and its underlying data into the public Plotly cloud-based workspace tool (https://plot.ly/create/) with many additional data visualization and filtering options.

2.4.4 Example of exploring differential expression with the Tripal Elasticsearch and Tripal Analysis Expression modules

Both modules are currently installed on the Hardwood Genomics Web, a site housing forest tree transcriptome data. Using this database, we can illustrate how the modules can be used together to explore the expression patterns of a group of transcripts. For example, a user may want to explore expression patterns in transcripts originating from heat shock gene family members in *Fraxinus pennsylvanica* (green ash). This species has a published transcriptome with 107,611 transcripts derived from *de novo* assembly of RNA sequence data from 55 biomaterials, including several stress treatments such as heat, cold, drought, mechanical wounding, and varying levels of O_3 concentration(Lane *et al*., 2016). Heat shock proteins were initially discovered as an induced response to heat stress in Drosophila(Ritossa, 1962) and plant homologs have also been observed increasing in expression under heat stress as well as other types of stress including cold and

wounding(De Maio, 1999; Swindell *et al*., 2007). To explore if heat shock proteins are induced across the various stressed biomaterials in *F. pennsylvanica*, a user might first use the transcript search interface with the phrase "heat shock" in the BLAST match description (Figure 3B). The organism may be filtered by selecting "*Fraxinus pennsylvanica*". As of May 2017, this yields 302 putative heat shock transcripts. For comparison purposes, reference transcripts likely to be constitutively expressed across all tissues may also be examined. Ubiquitin-conjugating enzyme E2 is often used in RT-qPCR (quantitative reverse transcription PCR) and other assays as a constitutive expression control gene(Czechowski *et al*., 2005). A search for gene features that contain BLAST matches to "Ubiquitin-conjugating enzyme E2" (including quotes in the search) yields 123 transcripts from *F. pennsylvanica* as of May 2017. To generate a small exploratory heatmap, we selected 8 transcripts randomly from each set of search results and compared them using the expression visualization heatmap tool (Figure 6D). From the heatmap, we can see that 6 out of 8 heat-shock transcripts have increased expression under several stress conditions, notably heat treatments. The 8 ubiquitin transcripts display more consistent expression across all biomaterials. This example demonstrates the types of data exploration enabled by a combined use of the Tripal Elasticsearch module and the Tripal Analysis Expression module.

2.5 Conclusions

The modular nature of the Tripal software enables new functionality to be easily built and offered as optional add-ons for the Tripal community. The installation of a new module or set of modules can immediately provide the infrastructure for a new genomic content types and provide custom data discovery and visualization interfaces. Here we described two new extension modules inspired by the need to provide robust transcriptome data access in a Tripal website. First, the

Tripal Elasticsearch module services the findable principle of data infrastructure by providing a major advance over prior search solutions. Second, the Tripal Analysis Expression module provides an efficient way for Tripal databases to store and offer public access to gene expression experiment data. This enables exploration of data through visualization of gene expression levels for individual genes and the ability for users to build their own expression heatmaps from their genes of interest, encouraging reuse of existing experimental data. Together, the modules provide significant additional Tripal functionality for users to explore RNA-Seq data, transcriptome assemblies, and transcript expression levels.

2.6 Acknowledgements

We acknowledge the entire Tripal software developers community, particularly Stephen Ficklin (Washington State University) and Lacey-Anne Sanderson (University of Saskatchewan), for providing the core Tripal infrastructure upon which these extension modules build. We also acknowledge Anthony Bretaudeau (Symbiose Project Team, INRIA/Irisa) for his code contributions via pull request to the Tripal Elasticsearch module.

2.7 Funding

This work was supported by the National Science Foundation [#1443040 to M.S., J.W and S.F. and #1444573 to M.S. and J.W.].

Conflict of interest. None declared.

2.8 Appendix

Table 2.1. Data types supported by Tripal.

Many different types of data may be produced related to an RNA sequencing experiment, and in many cases, there already exists a supporting Tripal module that accepts standard data file formats and encourages use of appropriate controlled vocabularies.

Type of Data	Tripal Module	Upload File Format	Controlled Vocabulary	Corresponding NCBI Database
RNA Sequence read	Feature (core)	Fasta	Sequence Ontology	Sequence Read Archive
Assembled Transcript Sequence	Feature (core) and Analysis Unigene (extension)	Fasta	Sequence Ontology	Transcriptome Shotgun Assembly
Gene Sequence	Feature (core)	Fasta	Sequence Ontology	Gene
BLAST result	BLAST (extension)	XML	Gene Ontology	NA
KEGG result	KEGG (extension)	Tab-delimited	Gene Ontology	NA
Biomaterial	Analysis Expression (extension)	XML or tab-delimited	Species-dependent [a]	BioSample
Gene Expression Values	Analysis Expression (extension)	Matrix or column files	NA	Gene Expression Omnibus
Gene Expression Experiment Methods	Analysis Expression (extension)	Descriptive text	NA	BioProject

Data may be source from or uploaded to a corresponding database in NCBI.

[a]Biomaterials may be associated with multiple controlled vocabularies, often species-specific. For example, plant samples may be described by anatomical structure and development stage with the Plant Ontology (Avraham *et al*., 2008), by phenotype with the Plant Trait Ontology or by stress treatment with the Plant Stress Ontology. All of these ontologies are available through Planteome (http://planteome.org/).

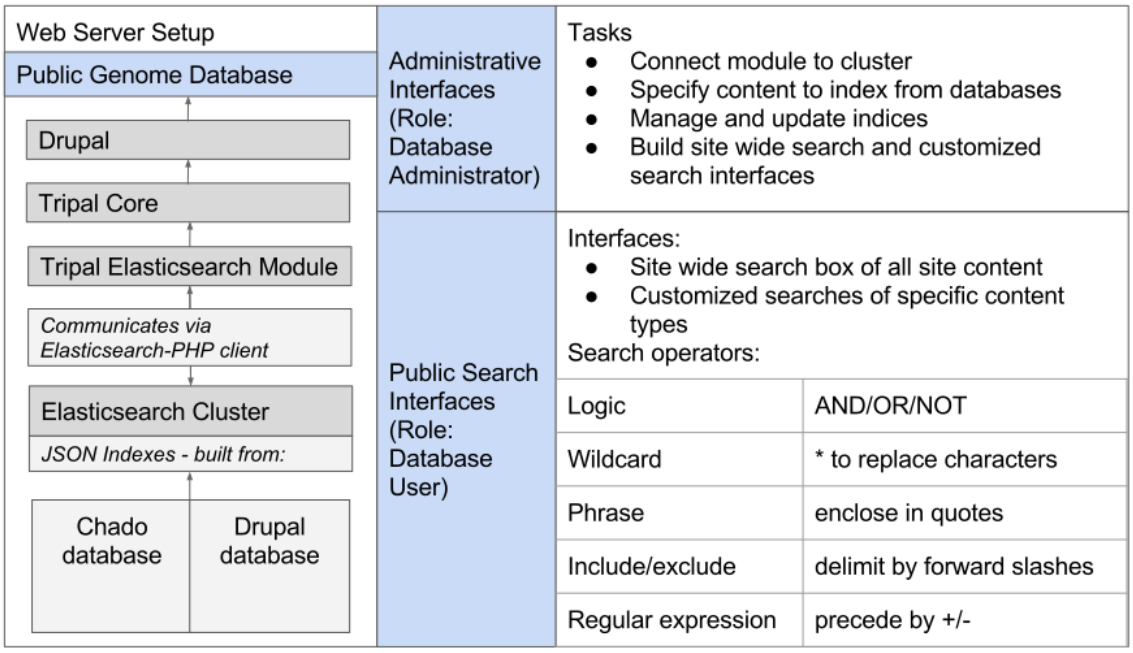


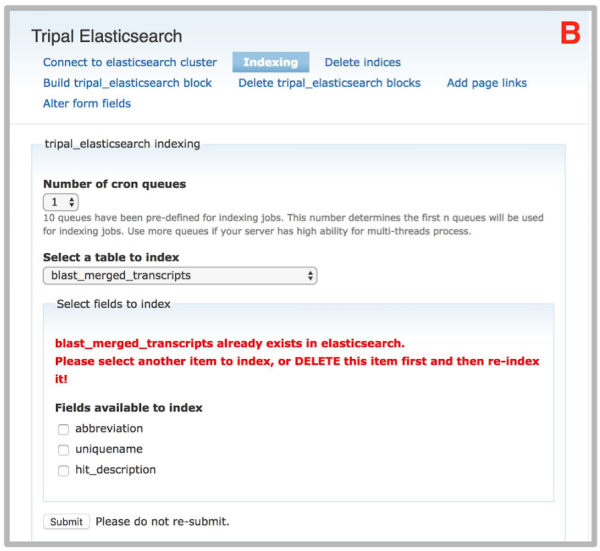
Figure 2.1. The Tripal Elasticsearch module depends on a number of other software packages.

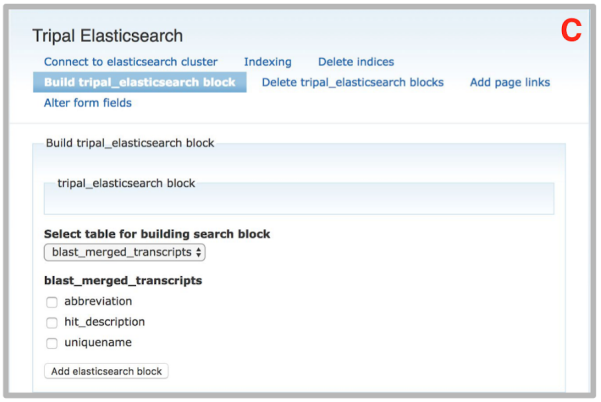
It works within the Drupal content management system and requires the core Tripal modules. To build indexes and run efficient searchs, it relies on an Elasticsearch cluster, which it communicates with via the Elasticsearch-PHP client. After installation of the software, all functionality can be controlled from web interfaces. The Drupal administrative user can configure and set up site-wide and customized search interfaces. Once those are active, all website users can perform fast, efficient searching with flexible search operators.

Figure 2.2. The Tripal Elasticsearch module administrative pages provide centralized control of data indexing and construction of search interfaces.

(A) The module first needs to be connected to a running Elasticsearch cluster. (B) The user may opt to index the entire site by selecting "index_website" or select individual tables and fields from the database. The index jobs are automatically launched via cron. (C) Based on available search indexes, a search form can be built and placed in a Drupal content block for display to users. (D) For each search block, the search results are linked to site content by constructing custom URLs (universal resource locator). The administrative page gives an example of how to use search results as tokens in the construction of URLs. (E) The search form can be customized for the best final user experience.


A
Tripal Elasticsearch
Connect to elasticsearch cluster
Indexing
Delete indices
Build tripal_elasticsearch block
Delete tripal_elasticsearch blocks
Add page links
Alter form fields
Elasticsearch cluster health information:
epoch: 1485706182
timestamp: 11:09:42
cluster: HGD-cluster-1
status: green
node.total: 2
node.data: 2
shards: 15
pri: 15
relo: 0
init: 0
unassign: 0
pending_tasks: 0
max_task_wait_time: -
active_shards_percent: 100.0%
Any host from an elasticsearch cluster
127.0.0.1: 9200
Submit


B
Tripal Elasticsearch
Connect to elasticsearch cluster
Indexing
Delete indices
Build tripal_elasticsearch block
Delete tripal_elasticsearch blocks
Add page links
Alter form fields
tripal_elasticsearch indexing
Number of cron queues
1
10 queues have been pre-defined for indexing jobs. This number determines the first n queues will be used for indexing jobs. Use more queues if your server has high ability for multi-threads process.
Select a table to index
blast_merged_transcripts
Select fields to index
blast_merged_transcripts already exists in elasticsearch.
Please select another item to index, or DELETE this item first and then re-index it!
Fields available to index
abbreviation
uniquename
hit_description
Submit
Please do not re-submit.


C
Tripal Elasticsearch
Connect to elasticsearch cluster
Indexing
Delete indices
Build tripal_elasticsearch block
Delete tripal_elasticsearch blocks
Add page links
Alter form fields
Build tripal_elasticsearch block
tripal_elasticsearch block
Select table for building search block
blast_merged_transcripts
blast_merged_transcripts
abbreviation
hit_description
uniquename
Add elasticsearch block


D
Tripal Elasticsearch
Connect to elasticsearch cluster
Indexing
Delete indices
Build tripal_elasticsearch block
Delete tripal_elasticsearch blocks
Add page links
Alter form fields
Example: add page links to search results
Add page links to search results
This page allows you to add dynamic URL to search results. Here dynamic URL means that the URL dependens on the seach result values. For example, the search form for organism has four fields: abbreviation, common_name, genus, species. We want to link some search results to corresponding oranism pages. Assuming that all organism pages have the same pattern: organism/genus/species (all URLs have the same part "organism/" but are different at "genus" and "species"),and the different parts in URLs only depend on the values of search results, in this case, we can link search results to their corresponding organism pages by adding dynamic links to the fields.
A dynamic link is created by replacing variable values in URL with corresponding field names within "[]".
The example below shows how to add links to search results in fields abbreviation and species.
organism/[genus]/[species]
abbreviation
common_name
genus
organism/[genus]/[species]
species
Add page links to search results
blast_merged_transcripts
abbreviation
hit_description
uniquename
Add links

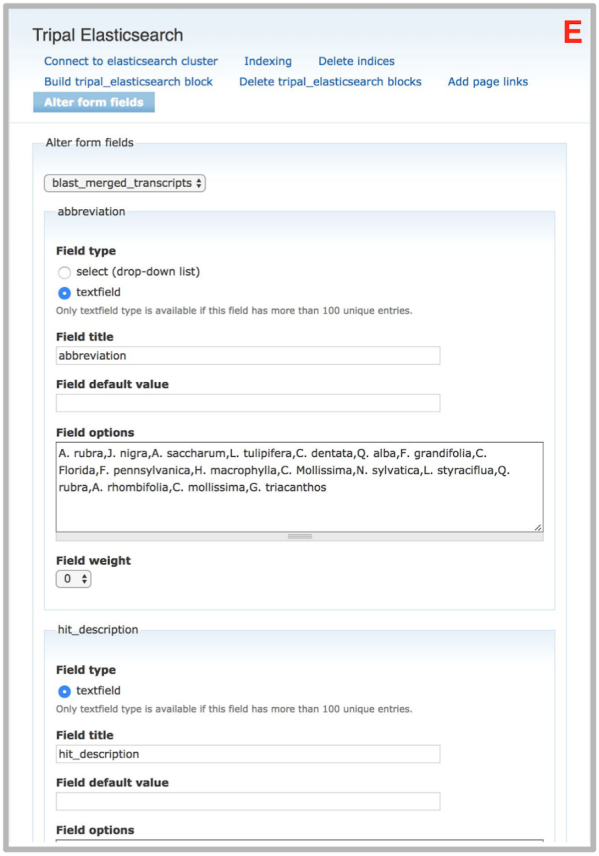
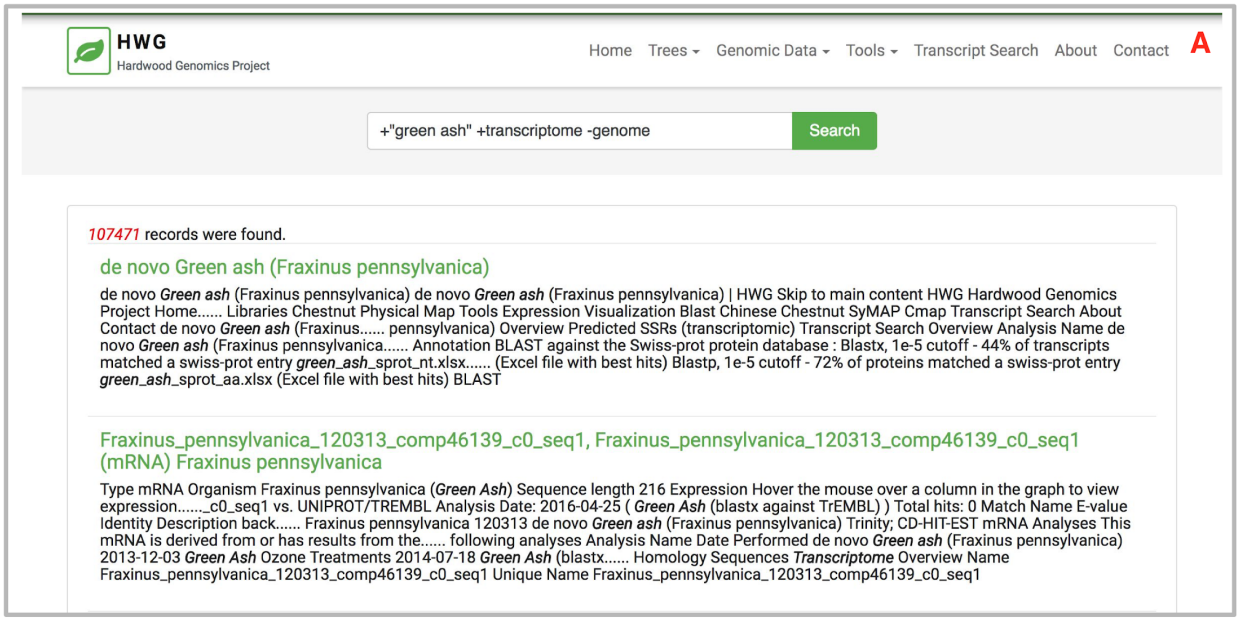
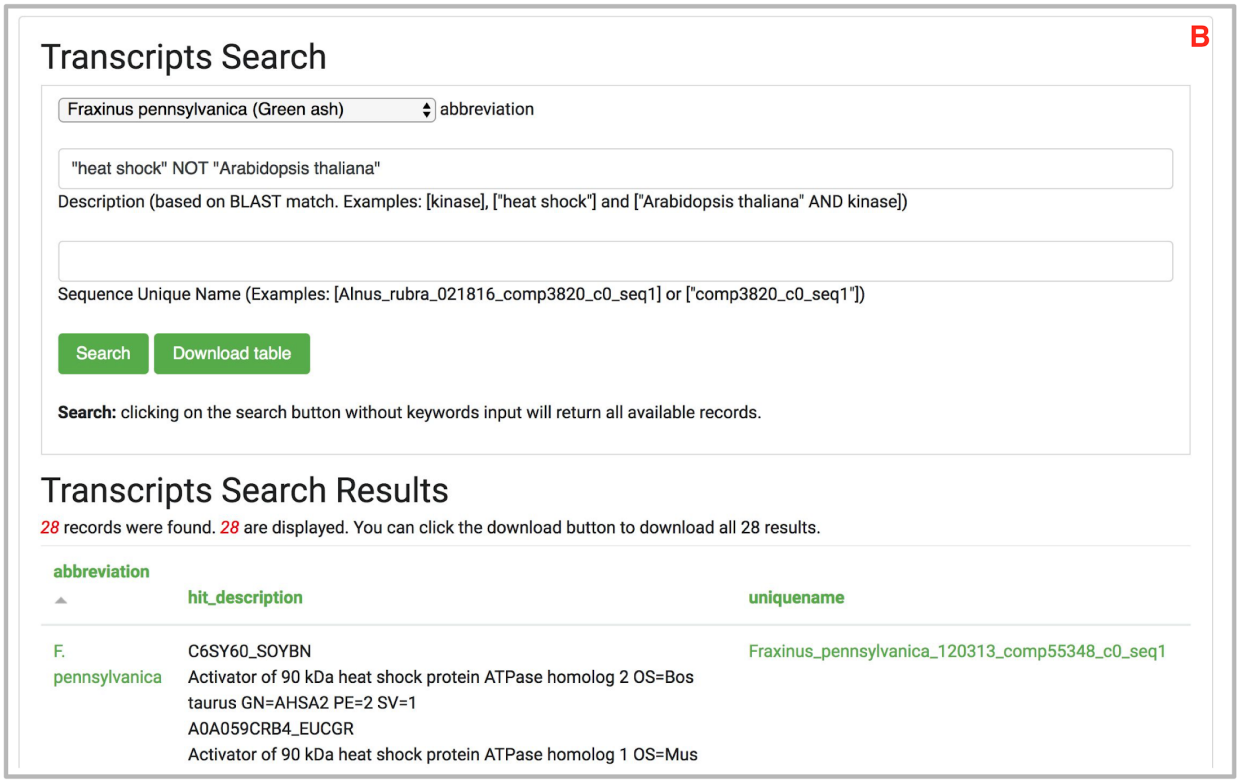

E
Tripal Elasticsearch
Connect to elasticsearch cluster
Indexing
Delete indices
Build tripal_elasticsearch block
Delete tripal_elasticsearch blocks
Add page links
Alter form fields
Alter form fields
blast_merged_transcripts
abbreviation
Field type
select (drop-down list)
textfield
Only textfield type is available if this field has more than 100 unique entries.
Field title
abbreviation
Field default value
Field options
A. rubra,J. nigra,A. saccharum,L. tulipifera,C. dentata,Q. alba,F. grandifolia,C. Florida,F. pennsylvanica,H. macrophylla,C. Mollissima,N. sylvatica,L. styraciflua,Q. rubra,A. rhombifolia,C. mollissima,G. triacanthos
Field weight
0
hit_description
Field type
textfield
Only textfield type is available if this field has more than 100 unique entries.
Field title
hit_description
Field default value
Field options

Figure 2.3. The Tripal Elasticsearch module builds content blocks for search forms.

Using the Drupal block system, a search block may be displayed on every page, on its own dedicated page or on a specific page or pages across the site. Screenshots are taken from the Hardwood Genomics Web where the Tripal Elasticsearch module is in production use. (A) The generic site-wide search block includes a single, global search field and search button. (B) A table-specific search form provides additional search fields for more advanced filtering of results. Here, transcript results are quickly returned from a database containing over 500,000 transcript records and more than 13,600,000 BLAST hits.


HWG
Hardwood Genomics Project
Home
Trees
Genomic Data
Tools
Transcript Search
About
Contact
A
+"green ash" +transcriptome -genome
Search
107471 records were found.
de novo Green ash (Fraxinus pennsylvanica)
de novo Green ash (Fraxinus pennsylvanica) de novo Green ash (Fraxinus pennsylvanica) | HWG Skip to main content HWG Hardwood Genomics Project Home..... Libraries Chestnut Physical Map Tools Expression Visualization Blast Chinese Chestnut SyMAP Cmap Transcript Search About Contact de novo Green ash (Fraxinus..... pennsylvanica) Overview Predicted SSRs (transcriptomic) Transcript Search Overview Analysis Name de novo Green ash (Fraxinus pennsylvanica..... Annotation BLAST against the Swiss-prot protein database : Blastx, 1e-5 cutoff - 44% of transcripts matched a swiss-prot entry green_ash_sprot_nt.xlsx..... (Excel file with best hits) Blastp, 1e-5 cutoff - 72% of proteins matched a swiss-prot entry green_ash_sprot_aa.xlsx (Excel file with best hits) BLAST
Fraxinus_pennsylvanica_120313_comp46139_c0_seq1, Fraxinus_pennsylvanica_120313_comp46139_c0_seq1 (mRNA) Fraxinus pennsylvanica
Type mRNA Organism Fraxinus pennsylvanica (Green Ash) Sequence length 216 Expression Hover the mouse over a column in the graph to view expression....._c0_seq1 vs. UNIPROT/TREMBL Analysis Date: 2016-04-25 (Green Ash (blastx against TrEMBL)) Total hits: 0 Match Name E-value Identity Description back..... Fraxinus pennsylvanica 120313 de novo Green ash (Fraxinus pennsylvanica) Trinity; CD-HIT-EST mRNA Analyses This mRNA is derived from or has results from the..... following analyses Analysis Name Date Performed de novo Green ash (Fraxinus pennsylvanica) 2013-12-03 Green Ash Ozone Treatments 2014-07-18 Green Ash (blastx..... Homology Sequences Transcriptome Overview Name Fraxinus_pennsylvanica_120313_comp46139_c0_seq1 Unique Name Fraxinus_pennsylvanica_120313_comp46139_c0_seq1


B
Transcripts Search
Fraxinus pennsylvanica (Green ash)
abbreviation
"heat shock" NOT "Arabidopsis thaliana"
Description (based on BLAST match. Examples: [kinase], ["heat shock"] and ["Arabidopsis thaliana" AND kinase])
Sequence Unique Name (Examples: [Alnus_rubra_021816_comp3820_c0_seq1] or ["comp3820_c0_seq1"])
Search
Download table
Search: clicking on the search button without keywords input will return all available records.
Transcripts Search Results
28 records were found. 28 are displayed. You can click the download button to download all 28 results.
abbreviation
hit_description
uniquename
F. pennsylvanica
C6SY60_SOYBN
Activator of 90 kDa heat shock protein ATPase homolog 2 OS=Bos taurus GN=AHSA2 PE=2 SV=1
A0A059CRB4_EUCGR
Activator of 90 kDa heat shock protein ATPase homolog 1 OS=Mus
Fraxinus_pennsylvanica_120313_comp55348_c0_seq1

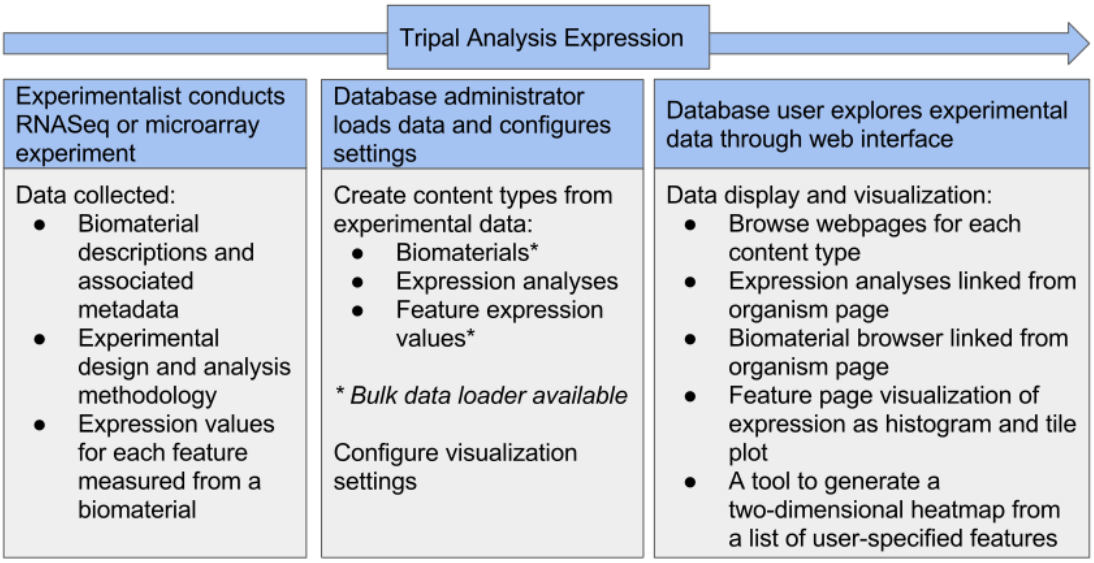


Figure 2.4. Flow of data in the Tripal Analysis Expression module.

Either RNA-Seq or microarray experimental data can be utilized and needs to include biomaterials, design and analysis methodology and expression values. From within a web interface, a site administrator can add and configure this content, yielding a number of visual interfaces for users to explore the data.

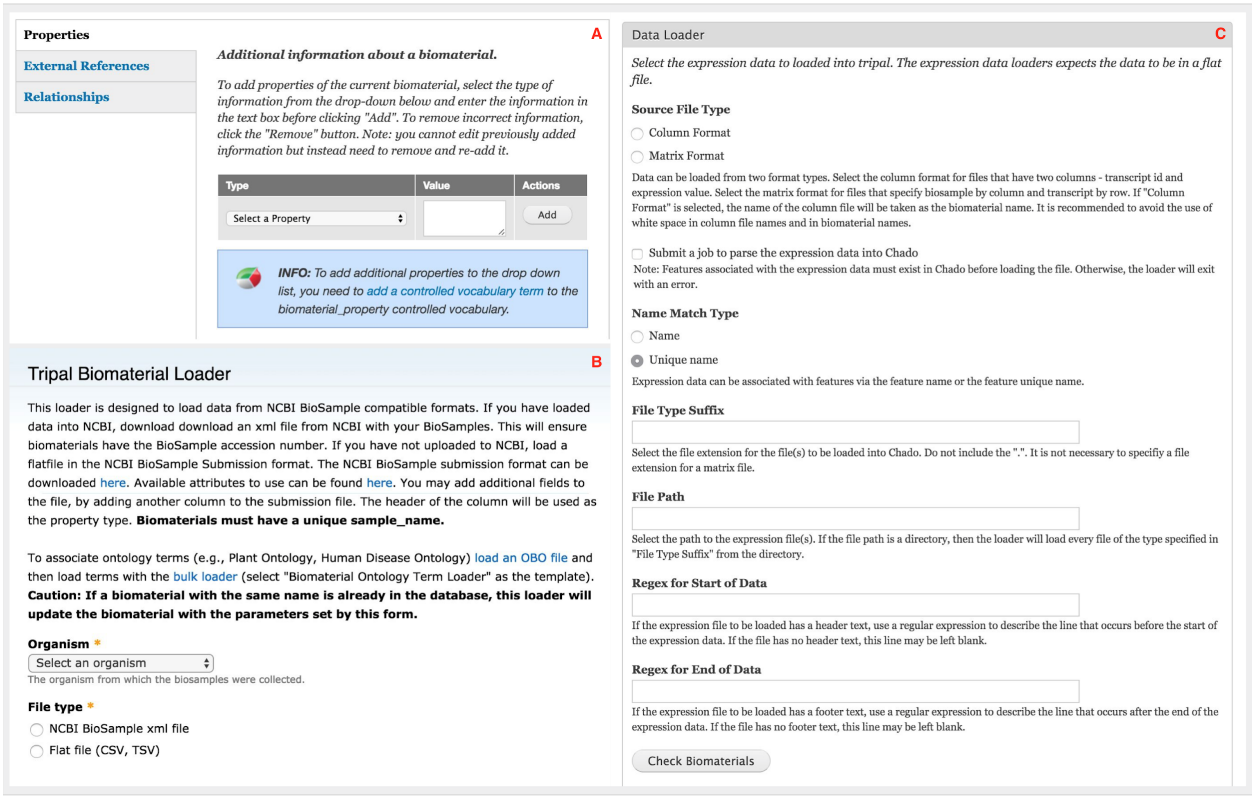
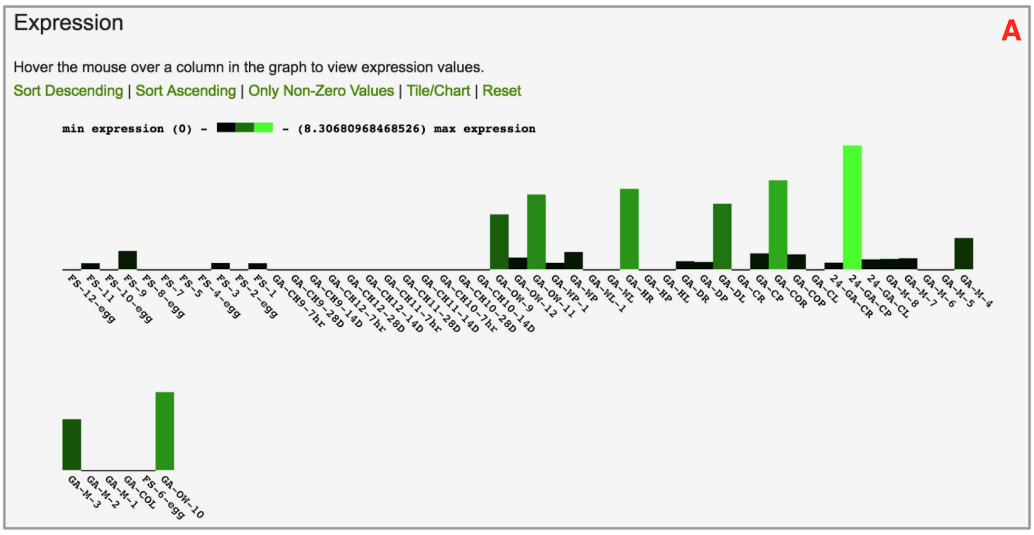


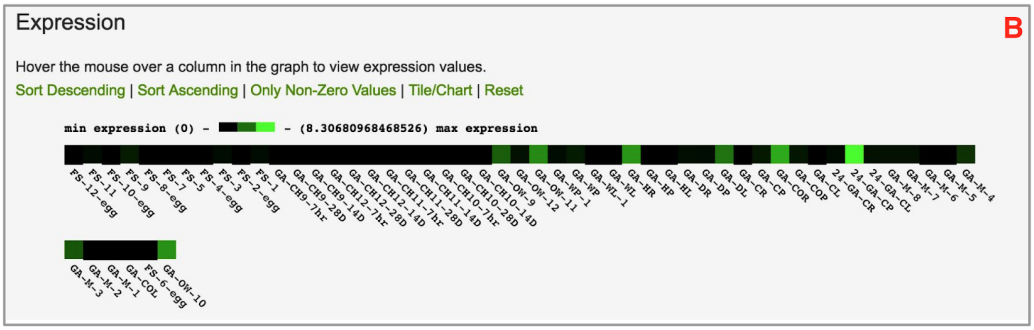
Figure 2.5. Administration interface of the Expression module.

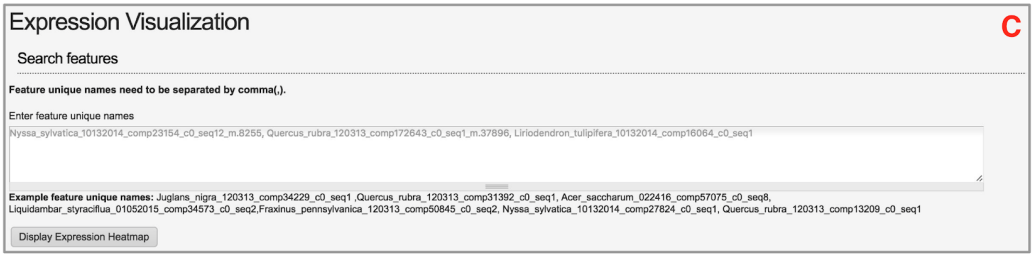
(A) Administrators can add customizable property fields to biomaterial record to provide metadata about experimental conditions, tissue, developmental stage, etc. These fields must be registered as a part of a controlled vocabulary through the core Tripal Controlled Vocabulary module. (B) The module is able to bulk load biomaterials from a CSV file or an NCBI BioSample XML formatted file. (C) The module can load expression data in column or matrix formats. These formats are described at the top of the page. The interface also allows the administrative user to specify regular expressions to exclude any header or footer text in the file that should be ignored such as column headings.

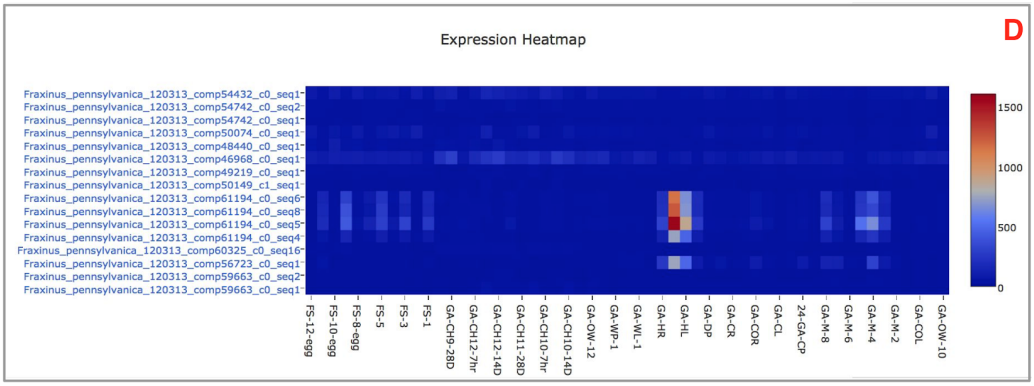
Figure 2.6. User interface of Tripal Analysis Expression module.

(A) The feature page initially displays expression values as a bar chart. (B) Users may toggle the the barchart to a one-dimensional heatmap. (C) An interactive tool allows users to paste in a list of feature ids of interest. (D) The tool will create a two dimensional heatmap of the feature expression values across all available biomaterials. This heatmap displays ubiquitin transcripts (top eight rows) vs. heat-shock transcripts (bottom eight rows) from Fraxinus pennsylvanica (green ash). Hovering over the boxes with red and orange color in the online heatmap reveals that they are derived from petioles and leaves exposed to heat stress (40 °C for 24 hours).


Expression
A
Hover the mouse over a column in the graph to view expression values.
Sort Descending | Sort Ascending | Only Non-Zero Values | Tile/Chart | Reset
min expression (0) - - (8.30680968468526) max expression


Expression
B
Hover the mouse over a column in the graph to view expression values.
Sort Descending | Sort Ascending | Only Non-Zero Values | Tile/Chart | Reset
min expression (0) - - (8.30680968468526) max expression


Expression Visualization
C
Search features
Feature unique names need to be separated by comma(,).
Enter feature unique names
Nyssa_sylvatica_10132014_comp23154_c0_seq12_m.8255, Quercus_rubra_120313_comp172643_c0_seq1_m.37896, Liriodendron_tulipifera_10132014_comp16064_c0_seq1
Example feature unique names: Juglans_nigra_120313_comp34229_c0_seq1 ,Quercus_rubra_120313_comp31392_c0_seq1, Acer_saccharum_022416_comp57075_c0_seq8,
Liquidambar_styraciflua_01052015_comp34573_c0_seq2,Fraxinus_pennsylvanica_120313_comp50845_c0_seq2, Nyssa_sylvatica_10132014_comp27824_c0_seq1, Quercus_rubra_120313_comp13209_c0_seq1
Display Expression Heatmap


Expression Heatmap
D
Fraxinus_pennsylvanica_120313_comp54432_c0_seq1
Fraxinus_pennsylvanica_120313_comp54742_c0_seq2
Fraxinus_pennsylvanica_120313_comp54742_c0_seq1
Fraxinus_pennsylvanica_120313_comp50074_c0_seq1
Fraxinus_pennsylvanica_120313_comp48440_c0_seq1
Fraxinus_pennsylvanica_120313_comp46968_c0_seq1
Fraxinus_pennsylvanica_120313_comp49219_c0_seq1
Fraxinus_pennsylvanica_120313_comp50149_c1_seq1
Fraxinus_pennsylvanica_120313_comp61194_c0_seq6
Fraxinus_pennsylvanica_120313_comp61194_c0_seq8
Fraxinus_pennsylvanica_120313_comp61194_c0_seq5
Fraxinus_pennsylvanica_120313_comp61194_c0_seq4
Fraxinus_pennsylvanica_120313_comp60325_c0_seq16
Fraxinus_pennsylvanica_120313_comp56723_c0_seq1
Fraxinus_pennsylvanica_120313_comp59663_c0_seq2
Fraxinus_pennsylvanica_120313_comp59663_c0_seq1
FS-12-egg
FS-10-egg
FS-8-egg
FS-5
FS-3
FS-1
GA-CH9-28D
GA-CH12-7hr
GA-CH12-14D
GA-CH11-28D
GA-CH10-7hr
GA-CH10-14D
GA-OW-12
GA-WP-1
GA-WL-1
GA-HR
GA-HL
GA-DP
GA-CR
GA-COR
GA-CL
24-GA-CP
GA-M-8
GA-M-6
GA-M-4
GA-M-2
GA-COL
GA-OW-10
1500
1000
500
0

CHAPTER III. AURORA GALAXY TOOLS: USING R MARKDOWN AS A FRAMEWORK FOR GALAXY TOOL DEVELOPMENT

A version of this chapter will be submitted to the journal of *BMC Bioinformatics* by Chen et al.:

Chen, M., Condon, B., Almsaeed, A., Wytko, C., Hough, H., Main, M., Ficklin, S., Staton, M. 2018. **Aurora Galaxy Tools: Using R Markdown as a framework for Galaxy tool development**.

3.1 Abstract

Background

Galaxy is an open, web-based platform which enables computational analysis for biological data through web interfaces without programming. The power and flexibility of Galaxy relies on the thousands of open source tools from the Galaxy ToolShed. Users can drag, drop and connect tools to build Galaxy workflows to perform a complete data analysis process. However, tools generate analysis results and output files in a variety of formats, often without any way for a user to visualize and interact with the results in an intuitive manner. Here I provide a Galaxy tool development framework which uses an R Markdown template to wrap command line tools and/or scripts from different programming languages.

Results

To enable Galaxy developers to create a consistent and user-friendly output report for each tool, I have developed a new tool wrapping framework that creates a web-browsable R Markdown document as output. To demonstrate this technique as a wrapper for many different types of command line tools and scripts from different programming languages, I have developed a set of new Galaxy tools using our R Markdown Galaxy approach called *Aurora Galaxy Tool*s, all of which are available from the Galaxy ToolShed. To enable users to easily explore and test the

reports in a Galaxy environment. I also developed a Galaxy Docker image that includes these tools. The source code for the tools and the Docker image is available on GitHub (https://github.com/statonlab/aurora-galaxy-tools). Here, I present an RNA-Seq data analysis, from raw sequencing data to gene expression as a use case. This demonstration of the Aurora Galaxy tools yields a reproducible workflow with HTML-based interfaces that encourages the user to explore the data output after each step of the pipeline. Many of the reports are interactive, allowing the user to manipulate visualizations and access all output files through a clickable file tree.

Conclusion

While many biologists are able to run sophisticated bioinformatic analysis using Galaxy, it is not always clear how well they understand the role of each tool in a workflow or whether the output is sensible. *Aurora Galaxy Tools* aims to engage the user in the output of each step of the workflow by providing easy-to-navigate summary reports and intuitive visualizations. Code is displayed alongside analysis results, making each tool's role in the analysis process more transparent to users and reproducible. I therefore suggest R Markdown as a general framework for Galaxy tool development to provide better display and visualization of results for users.

Keywords

R Markdown - Galaxy Project - Data visualization - Framework - Dynamic document - Galaxy Toolshed

3.2 Background

Galaxy is an open-source, web-based platform which enables biological researchers without programming expertise to perform computational analyses through a web interface

(Giardine *et al*., 2005; Afgan *et al*., 2016a). This intuitive and flexible method of data analysis has attracted many bioinformaticians and researchers to join the Galaxy developer community and contribute to Galaxy improvement and expansion. One of the community based efforts is to grow the Galaxy ToolShed (Blankenberg *et al*., 2014b). The ToolShed is an application store which currently hosts 5,649 bioinformatic tools (https://Tool Shed.g2.bx.psu.edu/, accessed on 05-01-2018). To add a tool to the ToolShed and make it available to any Galaxy instance, a definition file must be created. This XML file defines how the tool is called on the command line, the input and output files, the parameters that can be modified by the user, and help text. These tools provide the ability to perform comprehensive and complicated data analyses through web interfaces and form the fundamental work units of Galaxy. However, most Galaxy tools lack the ability to present analysis results in a well organized, integrated, and web friendly report. Rather, they simply generate the same outputs as the command line program, which are various files in different formats, mostly text-based. To our knowledge, there is only one Galaxy tool called iReport (Hiltemann *et al*., 2014) that aims at providing a generalized solution for analysis reporting in Galaxy. This tool builds HTML from outputs collected from other tools, and it only accepts a few types of content. MultiQC (Ewels *et al*., 2016) is another tool that aggregates bioinformatic analyses into a single HTML report. However, both MultiQC and iReport are stand-alone software packages that are not widely applicable to all tools in the Galaxy ToolShed.

One solution for building a flexible output report is R Markdown (Allaire *et al*., 2015), a plain text formatting syntax that includes both source code and annotations to stylize text. R Markdown files can be rendered by R to generate dynamic HTML documents (Xie, 2015). When rendered, code chunks in the file are executed and non-code text is interpreted in markdown language. The commands, analysis results and graphics are embedded into a rich document with

formatting and figures rendered as HTML (Baumer *et al*., 2014). This is very useful for building integrated reports that display the analysis results as well as the data analysis logic. R Markdown supports not only R scripts but scripts written in a variety of other languages, including Python, JavaScript, Bash, C++ and many others (Team, 2015). Since Galaxy tools are actually command line applications with an additional web interface layer; they can be developed using an R Markdown framework. The R Markdown solution for Galaxy tool development will not only enable a general format for results reporting, but also improve data visualization and presentation by enabling developers to add new code logic to generate meaningful user interfaces for results. I have developed a set of R Markdown based Galaxy tools as examples of this framework, referred to as *Aurora Galaxy Tools*. *Aurora Galaxy tools* enable data analysis and summarize results in HTML format for better visualization and display.

3.3 Methods

R Markdown Galaxy tools generate HTML outputs which can display code-embedded analysis results. During the development of the R Markdown document, I can add text to the report to describe what the command is doing and how to interpret the results. The HTML format also makes it straightforward to present interactive plots, which can be easily created using existing R HTML widgets (htmlwidgets for R) without expertise in Javascript programming.

Structure of an Aurora Galaxy tool To develop an *Aurora Galaxy* tool, three types of files are required. Table 3.1 shows the comparison of file compositions and functions between a regular Galaxy tool and an Aurora Galaxy tool. Like regular Galaxy tools, an XML tool configuration file is required and serves to define the command line execution, the tool dependencies, the parameters that can be altered by the user, the input and output files, and any help text to guide users in usage.

In the case of an R Markdown Galaxy tool, the command line does not point to the software itself but to the second file, an R script that acts as a bridge from the XML tool definition file to the R Markdown. Finally, the R Markdown file includes the code to be executed to run the underlying tool and serves as the template for generating the final HTML report.

The R script and the R Markdown files are the two new files that differ from the standard Galaxy tool wrapper. The R script defines variables to store user input values, output file paths, and any other extra command line arguments from the XML tool definition file. These variables can then be used in the R Markdown file for data analysis. The R script also defines environment variables to store the same input values, output file paths, and the extra command line arguments so that these values can be easily used with other programming languages in the R Markdown file. This will make integrating different programming scripts in the same R Markdown file or using these scripts directly in the tool much easier. This is because different programming languages can more easily use values stored in an environment variable than values stored in an R variable. The R Markdown files are then rendered through the R script file by calling the *render* or *render_site* function from the R package *rmarkdown* (Allaire *et al*., 2016). Data analysis code in the R Markdown files gets executed during the rendering process and a final report is generated as a tool output.

Two versions of Aurora Galaxy Tools R Markdown files can be rendered to generate not only a single HTML report, but also a static website with multiple interlinked HTML pages. When a tool has many different types of output or many different visualizations, a static website may be a better option for providing a more organized analysis report. For each output file from a given Galaxy tool, Galaxy allows a tool to create a directory that is associated with this output file to store extra files. This associated directory can be utilized to hold all the components of a static

website and develop tools that render a website to report analysis results. Any page from the website can be used as the tool's output report. End users will be then able to access the entire website within Galaxy. Although any page can be used as the output report, it is recommended to always use the *index.html* page, which is traditionally the name of the home page for a website and a required page for any R Markdown rendered website (Allaire *et al*., 2016).

Steps to developing an Aurora Galaxy Tool I have developed a set of template files to help developers to quickly develop new *Aurora Galaxy Tools*. All template files (Table 3.2) are available from the GitHub repository. Most of those template files are common to both single HTML report and website report types of Aurora tools. In this section, I will demonstrate how to develop an *Aurora Galaxy Tool* and explain the main functionalities of each template file.

Step 1: Define the web interface

First, a developer needs to create a Galaxy tool XML file which defines the web interfaces including inputs and outputs, as well as command line execution. The *rmarkdown_report.xml* file can be used as a template. The command line execution portion in this file creates an outputs directory and three environment variables which store paths to the tool installation directory, the output report and the outputs directory. Developers should extend the command line execution section by defining command line arguments that will be passed to the *rmarkdown_report_render.R* file.

Step 2: Collect inputs for data analysis

After the developer has defined the tool web interface, they can use the *rmarkdown_report_render.R* to collect inputs for the data analysis. This R script file functions as a command line application and bridges the tool definition XML file and the data analysis R Markdown files (see details in next step). This file reads the option specifications from the

command-line-arguments.csv file and for each command line argument that is passed from the *rmarkdown_report.xml*, stores the value in both an R variable and an Unix environment variable. Developers don't need to alter the rmarkdown_report_render.R file. They add option specifications to the *command-line-arguments.csv* file and these option specifications will be automatically imported by the *rmarkdown_report_render.R* file as command line arguments and R variables. The *command-line-arguments.csv* file uses a tab-delimited table with four columns to store specifications of command line options defined in the command line execution of the *rmarkdown_report.xml* file. The first and last columns store short option flags and variable names. The second column specifies if this option is required or not. The third column specifies the data type that this option accepts. Users can refer to the documentation of the R package *getopt* for more details (from Trevor L Davis and Zenka., 2018). After all command line arguments are collected from the *rmarkdown_report.xml,* the *rmarkdown_report_render.R* template renders R Markdown files to perform the actual data analysis.

Step 3: Build data analysis logic

After the developer has defined the tool web interface through the *rmarkdown_report.xml* file and collected and converted all necessary user inputs into to R variables through the rmarkdown_report_render.R file, they can start adding the actual data analysis logic to the rmarkdown_report.Rmd template file. Developers should add any descriptive text and data analysis code to the end of this file to extend the report template. The existing content in this template file provides necessary CSS and Javascript code for building an interactive file tree of the outputs directory in the final report by using the jsTree (Bozhanov) Javascript library. For website report tools, the same CSS and Javascript content is moved to the *index.Rmd* file. The path to the tool outputs directory is assigned to an environment variable named

REPORT_FILES_PATH in the rmarkdown_report.xml file. When developing the data analysis logic, tool developers should consider moving any output files to the outputs directory if they want those files to be accessible to users. In website report tools, the analysis logic in *rmarkdown_report.Rmd* would be split into multiple R Markdown files. Each R Markdown file will be rendered to generate a separate HTML page. The *_site.yml* file defines the website layout and structure and is specific to website report tools. The website layout and theme is highly customizable and completely determined by tool developers. I provided *_site.yml* template file as a minimal example mainly for demonstration purpose. Developers can refer to Xie et al. (2017) (Xie *et al*., 2017) for many more possibilities.

Although R Markdown allows executing bash code chunks directly, I recommend building separate shell scripts to be called from the R Markdown files. This helps to organize the various pieces of code and makes debugging or altering tool commands easier in the future. Therefore, I created the *build-and-run-job-scripts.sh* template file for any computational analysis that are executed in bash script. However, this template file is optional, and tool developers can have their own customized shell scripts or choose to not use any shell script files.

Step 4: Expose outputs to Galaxy history

Although *Aurora Galaxy Tools* use a clickable file tree to expose all output files in the outputs directory to users, these files cannot be used directly as inputs for other Galaxy tools. To make an output file available to other Galaxy tools, I need to define an output component in the *rmarkdown_report.xml* file, and then attach an output file to that component. Developers can expose any output files in the tool outputs directory as Galaxy outputs by extending this template and defining more outputs in the *rmarkdown_report.xml*. Similar to the *build-and-run-job-*

scripts.sh file, this is not required to be a separate file, but the template is added for the purpose of organizing code and providing easy maintenance and updates.

3.4 Results

Resources Availability

the source code for all *Aurora Galaxy Tools* can be found in the Github repository: https://github.com/statonlab/aurora-galaxy-tools. A website is also available for demonstration of Aurora Galaxy tool reports: https://statonlab.github.io/aurora-galaxy-tools/.

Feature highlights compared to regular Galaxy tools

1. Access to all output files with a clickable file tree navigation

Users can navigate the outputs directory and view or download any output files by exploring the clickable file tree (Figure 3.1). For the reports from website aurora tools, a blue bar on the top provides an additional menu navigation system for users to explore results.

Although all *Aurora Galaxy Tools* provide an interactive file tree which allows users to access any outputs generated by the tool, these files cannot be directly used as inputs by other tools. However, tool developers can add extra output components to the *rmarkdown_report.xml* and selectively expose outputs to the Galaxy history. For example, the Aurora STAR tools provide access to all output files to users through the clickable file tree, but they also expose reads alignment in BAM file format to the Galaxy history. These BAM files are used as inputs for the Aurora HTSeq tools to generate read counts in CSV file format. Read counts are then used by Aurora DESeq2 tools to perform differential gene expression analysis.

2. More informative and transparent in analysis

Since *Aurora Galaxy Tools* use R Markdown documents to report analysis results, tool developers can easily add descriptive contents to the report to make it a comprehensive tutorial to users, instead of simply rendering analysis results to them. For example, details can be added to explain what steps have been taken to get the results and how to explain the results, or add extra further analysis steps to the raw outputs from the bioinformatics tools that are being wrapped. For example, I add extra summary tables and interactive plots to help users to understand the raw outputs from the FastQC tool. The outputs from the regular FastQC Galaxy tool includes a .html html and a .txt file. Example outputs can be found at https://statonlab.github.io/aurora-galaxy-tools/aurora_fastqc/read_1_fastqc/fastqc_report.html and https://statonlab.github.io/aurora-galaxy-tools/aurora_fastqc/read_1_fastqc/fastqc_data.txt. Both these outputs are included in the Aurora FastQC tool report as interlinks. An example Aurora FastQC tool report can be found at https://statonlab.github.io/aurora-galaxy-tools/aurora_fastqc/rmarkdown_report.html.

In R Markdown reports, analysis code can be displayed along with the results, which makes the tool and analysis procedure less like a "black-box". Developers can choose to display analysis code by default, or make it optional for the user to determine if they want it included in the report.

3. Easy integration of interactive data visualization

Without knowing any Javascript, tool developers can easily implement interactive data visualization with many existing R and Python libraries to make the results exploration procedure more user-friendly. Although "raw" outputs from many bioinformatics tools are static, I could replace them with interactive reports in our Galaxy tools. For example, I used ggplot2 (Wickham, 2016) and plotly (Carson Sievert *et al*., 2016) together in the Aurora FastQC and Aurora DESeq2 tools to create interactive plots (see the *use cases* section).

4. Easy integration of analyses from different tools

It would be easier for users to make sense out of the results from multiple different but closely related tools if they could view the results in a single, well-organized reports. However, Galaxy currently does not have a generic way to summarise analysis results from different tools in a single report. By using R Markdown as a development framework, all *Aurora Galaxy Tools* provide a single report, either a single HTML or a single website report, to render all analysis results to users. With this feature, tool developers can wrap multiple different tools into a single Aurora Galaxy tool and render analysis results from different tools with a single report. For example, the report from the Aurora STAR tool includes analysis results from STAR, Samtools view, and samtools flagstat which covers genome indexing, mapping, SAM to BAM file format conversion and BAM file evaluation (see the *use cases* section).

The RNA-Seq analysis pipeline includes the following steps: 1) RNA sequences are trimmed with the *aurora_skewer*/*aurora_skewer_site* tools. 2) *aurora_fastqc*/*aurora_fastqc_site* are used to evaluate the reads quality before and after trimming. 3) The trimmed reads are mapped to a reference genome with the *aurora_star*/*aurora_star_site* tools. 4) The aligned reads are then processed with the *aurora_htseq*/*aurora_htseq_site* tools to obtain read counts data. 5) *aurora_deseq2*/*aurora_deseq2_site* tools are used to analyze counts data and detect significantly differentially expressed genes. As the goal is to illustrate the idea and advantages of *Aurora Galaxy Tools*, a small public RNA dataset was selected for demonstration purposes only. Reads files can be downloaded from Batut et. al (2017) (Batut *et al*., 2017). Reference genome and annotation files can be downloaded from NCBI Assembly database (accession ID GCF_000001215). Count and column datasets used for the deseq2 tool demonstration can be downloaded from the Github repository.

In this section, I will demonstrate the interactive reports generated by each tool from an RNA-Seq differential expression analysis pipeline. Both reports from the single HTML report tools and the website report tools have exactly the same content, except that the website version has a menu navigation bar on the top. Therefore, I only provide screenshots from single HTML report tools. However, both single HTML reports and website reports from each tool can be found in the online example available at (https://statonlab.github.io/aurora-galaxy-tools/).

3.5 Use cases

To demonstrate the advantages of wrapping Galaxy tools with the R Markdown framework and how to use the template files, I have developed a list of *Aurora Galaxy Tools* comprising a simplified RNA-Seq differential expression analysis pipeline (Table 3.3). I wrapped each tool in two different versions: a single HTML report version and a website report version. Both reports from the single HTML report tools and the website report tools have exactly the same content, except that the website version has a menu navigation bar on the top. The two versions are technically separate tools on the Galaxy ToolShed.

Aurora Skewer tools The first step in our pipeline was to trim adapters and low quality ends from a set of RNA reads. This was done with the *aurora_skewer*/*aurora_skewer_site* tools, which are wrappers of the software skewer (Jiang *et al*., 2014). Figure 3.2 shows the screenshots from the output reports generated by aurora_skewer. The report displays the skewer trimming code and provide access to the trimmed reads files, trimming log file, and other outputs with an interactive file tree. In addition to the report, the tool also expose the trimmed read files in the analysis history pane so that these files can be used by other tools.

Aurora FastQC tools After trimming, the quality of the reads was evaluated with the *aurora_fastqc/aurora_fastqc_site* tools, which are wrappers of the short read evaluation software FastQC (Andrews and Others, 2010). Our FastQC Galaxy tools generate better reports than the existing regular Galaxy tool *fastqc*. Unlike the regular FastQC Galaxy tool which simply outputs the raw HTML reports from the FastQC software, our tools display FastQC analysis code and visualize results with interactive plots implemented with ggplot2 (Wickham, 2016) and plotly (Carson Sievert *et al*., 2016). The raw HTML reports are also available and easily accessible from the interactive file tree in our Aurora Galaxy fastqc tools (Figure 3.3).

A quality control tool provides a good example of the extra value *Aurora Galaxy Tools* provides a Galaxy user. Without *Aurora Galaxy Tools*, the user will only be vaguely aware that a QC step was performed. Perhaps they will download the raw text files and make a note of the results. The enhanced *Aurora Galaxy Tools* reports bring this information front and center, with human readable tables and interactive visualizations. Users can explore this information and verify the analysis is working as expected before moving on to the next step, all without leaving Galaxy.

Aurora STAR tools After trimming the reads, I aligned them to a reference genome with the *aurora_star/aurora_star_site* tools, which are wrappers of the software STAR (Dobin *et al*., 2013). With the Aurora STAR tools, users can generate reports that display the analysis code for each step of the STAR tool mapping procedure, from genome indexing to mapping (Figure 3.4). At the end of the mapping process, a mapping evaluation summary is available. The summary is calculated with the samtools (Li *et al*., 2009) flagstat tool. This highlights one of the features of building an aurora tool, that an additional software package or script can be called to generate additional output such as descriptive summary statistics or visualizations.

Aurora HTSeq tools The output BAM files from the aurora_star/aurora_star_site tools can be used as input to the *aurora_htseq/aurora_htseq_site* tools to estimate the number of reads mapped to specific genes. *aurora_htseq/aurora_htseq_site* are wrappers of the HTSeq (Anders *et al*., 2015) count tool. An extra script is also embedded in this tool to merge the standard output files from htseq-count (Anders *et al*., 2015) into an output that can be directly used by the downstream tool Aurora DESeq2. Specifically, the software *htseq-count* generates a separate output file for each sample. The Aurora HTSeq tool combines these into a single tab-delimited matrix file, which is a common file format for expression data. This new output file is also exposed to the analysis history pane where it can be used for a number of downstream differential expression analysis tools. Figure 3.5 shows screenshots of example report from *aurora_htseq*.

Aurora DESeq2 tools The *aurora_deseq2/aurora_deseq2_site* are wrappers of custom R scripts that perform differential expression analysis with the DESeq2 R package (Love *et al*., 2013). The counts file from the Aurora HTSeq tools is used as input to the Aurora DESeq2 tools. The Aurora DESeq2 tools generates a differential expression analysis report as well as a CSV file that includes all significant genes. Figure 3.6 shows screenshots of example report from *aurora_deseq2*.

3.6 Discussion

Galaxy has a broad impact on biological research and has been increasingly adopted by researchers to perform a variety of bioinformatics analysis (Levin-Reisman *et al*., 2017; Marques *et al*., 2017; Rees *et al*., 2017). A single Galaxy instance may host hundreds of bioinformatics tools that allow users to perform a wide variety of computational analyses through web interfaces. Galaxy dramatically lowers the bar for performing bioinformatic analyses (Grüning *et al*., 2017)

since it does not require users to know command line programming or to have access to high performance computing services. There are over 100,000 registered users on the Main Galaxy server and over 1,500 new user registrations per month. As of March 13, 2018, there were about 100 public Galaxy servers (Galaxy Project Stats).

Despite the widespread adoption of Galaxy, a universal solution for the myriad file output formats and result types does not exist. As a result, the intermediate steps of a galaxy workflow go largely unexamined. Standardizing analysis reports across tools would improve the interpretability of results for end users. In this work, I proposed an R Markdown based framework for Galaxy tools development. Tools developed under this framework may generate well organized, generalized reports in HTML format, either a single HTML report or a static R Markdown website. R Markdown is one of the most popular frameworks for reproducible research (Gandrud, 2016; Baumer *et al*., 2014) and is an easy-to-learn skill for Galaxy tool developers. To adopt this framework, Galaxy tool developers simply need to move all data analysis and visualization logic of a tool to the R Markdown files and let the framework to generate the final reports.

In addition to standardizing analysis reports across tools, our R Markdown development framework adds extra values to tool and tool development in several other aspects. For example, it enables tool developers to display code along with results by default, or make it optional for the user to determine if they want it included in the report (Xie, 2015). Displaying the code that was executed to the user has a number of advantages. First, it reduces the "black box" problem with Galaxy tools, by revealing exactly how data is being handled. For users interested in how the tool is being run by the wrapper and willing to read the manual of command line options, they can decipher exactly which parameters are being used and how. Also, for users either already familiar

with the command line or learning the command line, it can help them to learn more about the software execution and even reproduce the results outside of Galaxy. Another advantage of using R Markdown is that many HTML widgets become immediately available to tool developers to provide better data visualization. These HTML widgets are R bindings to Javascript libraries (Vaidyanathan *et al*., 2014) such as Plotly (C. Sievert *et al*., 2016), Highcharts (Highsoft, 2014) and D3 (Meeks, 2017). Developers who are familiar with R programming but not with Javascript can still develop tools with efficient visualization outputs. Furthermore, by adding extra details and instructions to the computational logic of a tool in the R Markdown files, tool developers can create a tool that generate not a analysis reports but a comprehensive analysis tutorials, which could be very useful to non-advanced users.

R Markdown Galaxy tools have the potential to generates interactive documents through integrating Shiny application (Chang *et al*., 2015) into the R Markdown documents. Shiny applications are built straight from R with the Shiny R package and can be nested into the R Markdown document as a web application (Allaire *et al*., 2015). With this functionality, users can further explore their data based on analysis results. The success of integrating Docker image and running RStudio and Jupyter from within Galaxy makes it possible to run Shiny applications within Galaxy (Grüning *et al*., 2017). In future work, methods of developing Shiny embedded R Markdown Galaxy tools will be explored.

3.7 Acknowledgements & Funding

This work was supported by the National Science Foundation under grant #1443040. This work used the Extreme Science and Engineering Discovery Environment (XSEDE) Jetstream

Cloud, which is supported by National Science Foundation grant number ACI-1548562 and specifically by the XSEDE allocation to MS #TG-MCB160182.

3.8 Appendix

Table 3.1 File compositions and functionalities of standard Galaxy tools and *Aurora Galaxy Tools*.

	Files	Functions
Standard Galaxy tools	.xml	Required file. Defines user interface, command line execution, install tool dependencies
	Tool-specific files	These files are tool-specific (e.g., test data files, deseq2.R from the deseq2 tool)
Aurora Galaxy Tools	.xml	Required file. Defines user interface, command line execution, install tool dependencies. The command line execution calls the R script (see below).
	Tool-specific files	These files are tool-specific (e.g., the binary file twoBitToFa from the *aurora_fasta_importer* tool)
	R script file (.R)	Required file. Executed by the tool XML file. This file collects user inputs and other command line arguments and passes them to the R Markdown (.Rmd) files. Finally, it renders all R Markdown files.
	R Markdown files (.Rmd)	One or more required files. These files execute all data analysis code and serve as templates for generating the final HTML reports.

Table 3.2 Template files and functionalities for developing *Aurora Galaxy Tools*.

Files	Single HTML report	Website report
rmarkdown_report.xml	✓	✓
rmarkdown_report_render.R	✓	✓
rmarkdown_report.Rmd	✓	
index.Rmd		✓
_site.yml		✓
build-and-run-job-scripts.sh	✓	✓
command-line-arguments.csv	✓	✓
expose-outputs-to-galaxy-history.sh	✓	✓
vakata-jstree-3.3.5.zip [a]	✓	✓

[a] This is a compressed file of the jQuery plugin *jsTree* developed by Ivan Bozhanov (Bozhanov).

Table 3.3 Aurora tools for RNA-Seq differential expression analysis.

Tool name [a] in Galaxy test tool shed	Tool description	Tool References
aurora_skewer	Wrapper of skewer: a fast and accurate adapter trimmer for next-generation sequencing paired-end reads.	Jiang, Hongshan, et al. (Jiang *et al*., 2014)
aurora_fastqc	Wrapper of FastQC : evaluate short reads with FastQC software on a single or a paired of untrimmed and trimmed reads files	Andrews, Simon. (Andrews and Others, 2010)
aurora_star	Wrapper of STAR: ultrafast universal RNA-seq aligner	Dobin, Alexander, et al. (Dobin *et al*., 2013)
aurora_htseq	Wrapper of HTSeq-count: counting reads in features	Anders, Simon, Paul Theodor Pyl, and Wolfgang Huber. (Anders *et al*., 2015)
aurora_deseq2	Wrapper of DESeq2: Differential expression analysis with R DESeq2 package	Love, Michael, Simon Anders, and Wolfgang Huber. (M. I. Love *et al*., 2014)

[a] This table only display tool names for single HTML report tools. For website report tools, the corresponding name would be *aurora_tool_site*. For example, the website report version for aurora_skewer is aurora_skewer_site.

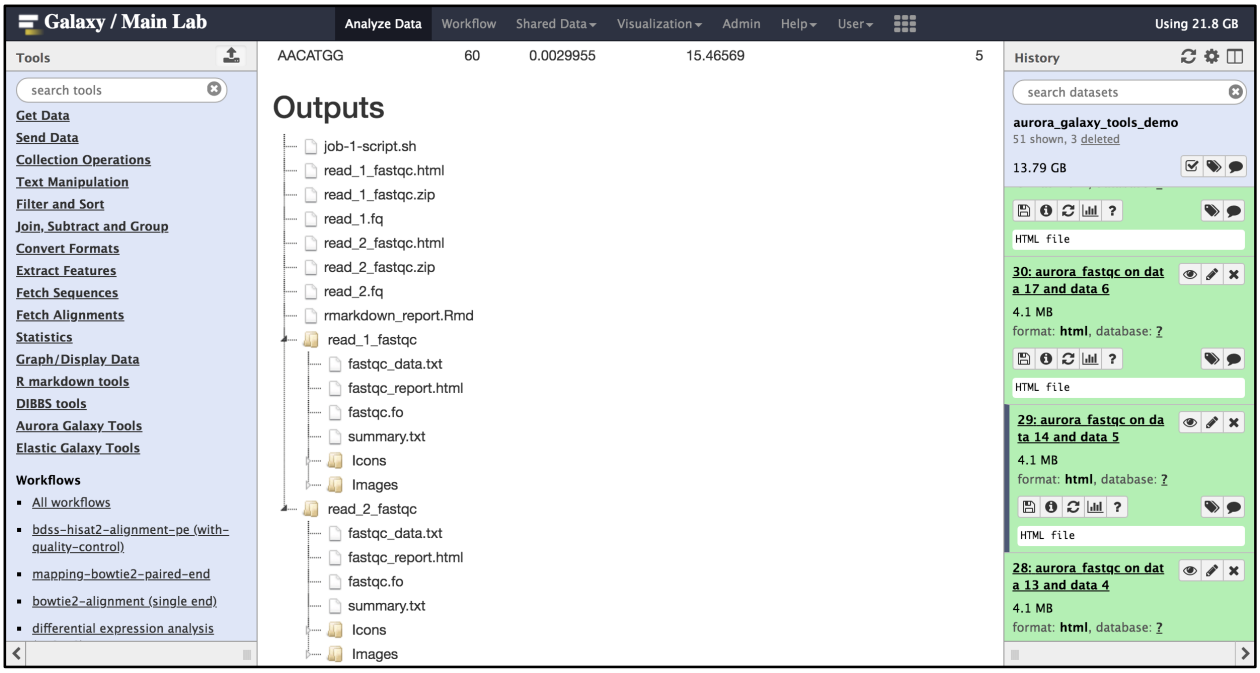


Figure 3.1 Interactive file tree display of all tool outputs from the *aurora_fastqc* tool.

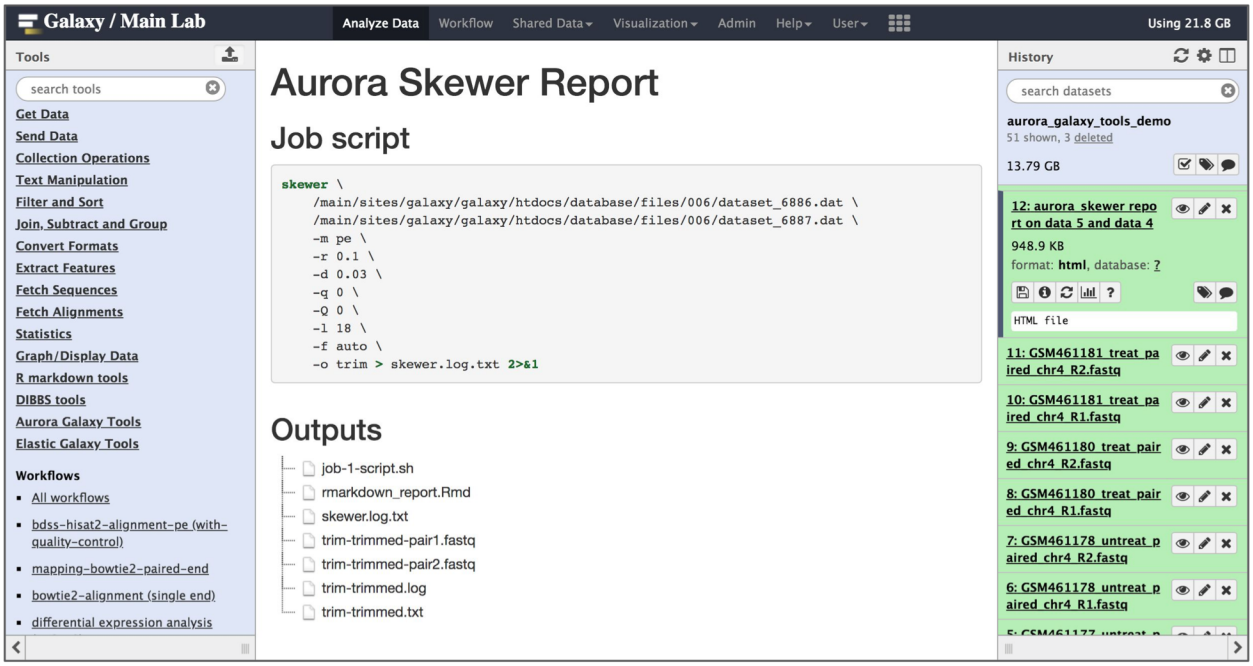


Figure 3.2 Single HTML report from *aurora_skewer* tool.

The report displays the skewer command executed and a clickable file tree for all outputs.

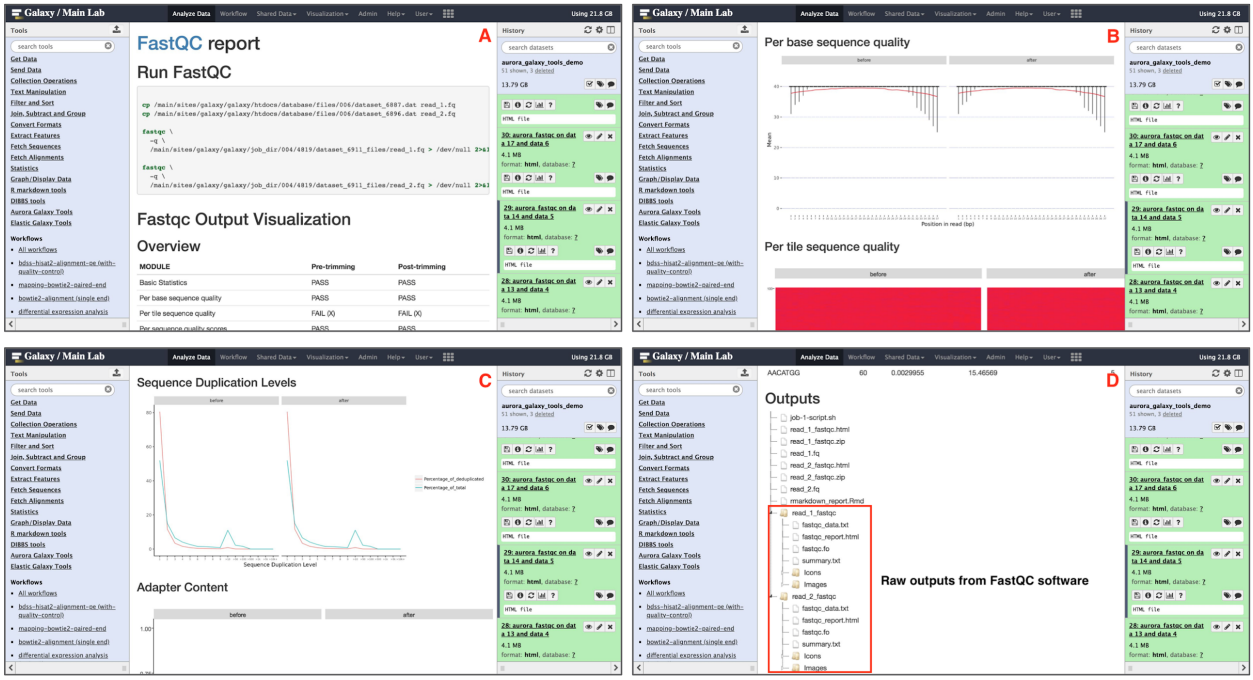


Figure 3.3 Screenshots of reports from *aurora_fastqc*.

(A) FastQC job script. (B & C) Visualization of 'per base sequence quality' and 'sequence duplication levels' with plots implemented with ggplot2 (Wickham, 2016) and plotly (Carson Sievert *et al*., 2016). (D) A clickable file tree for all outputs. Raw outputs from the FastQC software are also available.

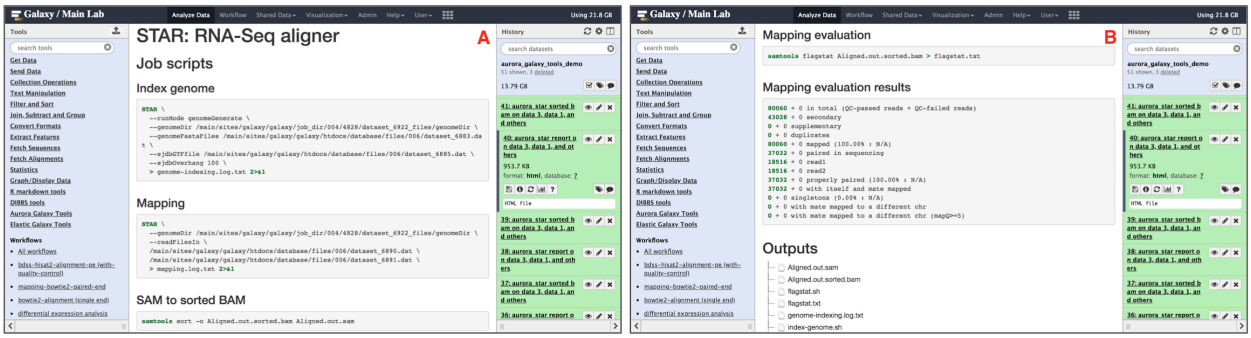


Figure 3.4 Screenshots of reports from *aurora_star* tool.

(A) scripts for genome index and mapping with STAR software. (B) mapping evaluation results obtained from samtools flagstat (Li *et al*., 2009).

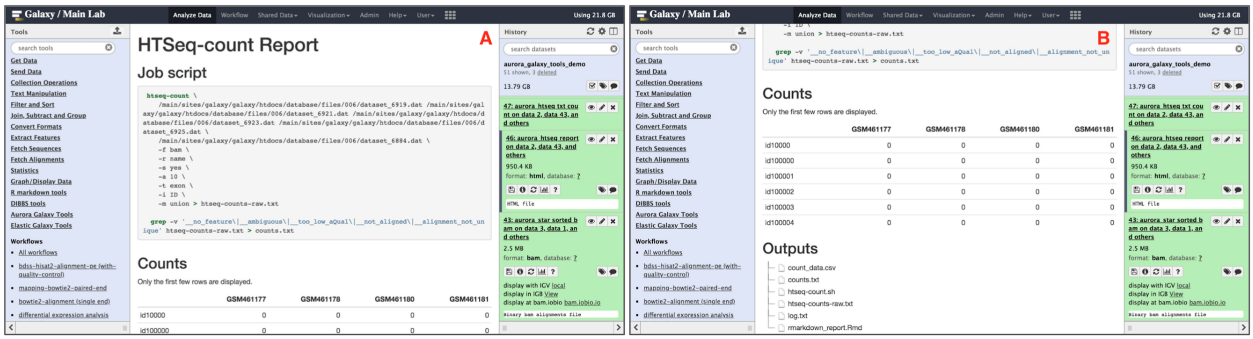


Figure 3.5 Screenshots of reports from *aurora_htseq*.

(A) htseq-count job script. (B) A table to show first few rows of the count data and a file tree for all outputs.



Figure 3.6 Screenshots of reports from *aurora_deseq2*.

(A) stdout of DESeq2 analysis. (B) an interactive Mean Average plot (MA-plot) implemented with ggplot2 (Wickham, 2016) and plotly (Carson Sievert *et al*., 2016). (C) heatmap of count matrix for top significant genes. (D) PCA plot and a clickable file tree for all outputs.

CHAPTER IV. GALAXY TOOL GENERATOR (GTG): A WEB APPLICATION FOR DEVELOPING GALAXY TOOLS

A version of this chapter will be submitted to the journal of *Bioinformatics* by Chen et al.:

Chen, M., Condon, B., Staton, M., 2018. **Galaxy Tool Generator (GTG): a web application for developing Galaxy tools**.

4.1 Abstract

Summary: With thousands of open source informatics tools and associated workflows available, Galaxy has become a powerful web-based analytic platform for high throughput genomics data. Developing a Galaxy tool can be a time-consuming process and usually requires expertise in multiple programming languages, as well as familiarity with the XML-based development framework. To lower the bar of entry for new Galaxy tool developers, I have developed a web application, Galaxy Tool Generator (GTG), which enables wrapping Galaxy tools, testing tools interactively and publishing tools to the Galaxy ToolShed and Test ToolShed through web interfaces. Developers can build tool XML files by adding components selected from a predefined drop-down components list, restructure the XML content by dragging and dropping components, and duplicate or remove components by clicking buttons.

Availability and implementation:

- https://github.com/MingChen0919/gtgdocker
- https://github.com/MingChen0919/galaxy_tool_generator_ui
- https://github.com/MingChen0919/galaxy_tool_generator

4.2 Introduction

Continued advancements in high throughput sequencing technologies has led to exponential growth of data in genome and transcriptome studies as well as expansion of the biological applications of those data (Marx, 2013b). Analyzing massive datasets and interpreting the results is a non-trivial task. It requires expertise in bioinformatics, which has become a key bottleneck preventing efficient use and interpretation of next generation sequencing (NGS) data (Li *et al*., 2017). However, this situation has been alleviated by many web-based bioinformatics platforms, such as Galaxy (Afgan *et al*., 2016a), CyVerse (Goff *et al*., 2011), KBase (Arkin *et al*., 2016) and EDGE (Li *et al*., 2017). These platforms enable researchers to perform complex bioinformatics analyses without the need for programming or establishing their own computational infrastructure. Among those platforms, Galaxy is one of the most well-known and has a large number of users (Galaxy Project Stats, accessed by Jan 20, 2018). Its success relies on the thousands of Galaxy tools contributed by the community and found in the Galaxy ToolShed, which serves as the "App Store" of Galaxy (Blankenberg *et al*., 2014c).

Here, I introduce the Galaxy Tool Generator (GTG) web application which assists in developing Galaxy tools. A Galaxy tool is an XML file that wraps a command line program and specifies how to build tool input and output interfaces, how to generate the commands to run the software with the specified inputs and parameters, how to install the tool and its dependencies, and many other aspects. These components are interpreted by Galaxy to build a web-based application for the specified data analysis. To successfully build a Galaxy tool, developers need to be very familiar with each XML tag, its associated attributes, and what information it defines. For example, the XML tag "param" defines an input component. The attribute "type" has the value "data" which

specifies that this input field will accept files as input. The rest of the attributes specify other properties of this input component (Figure 4.1A).

GTG simplifies and speeds the development of Galaxy tools by providing an editable web form to construct the tool XML. Developers can interactively work with the XML elements as easy to edit components and view them in an intuitive hierarchical structure in order to quickly build a complete Galaxy tool XML file (Figure 4.1 B,C). The form includes all valid tags and their particular attributes and allows the user to interactively add elements, edit element attributes, move elements in the hierarchy, clone an existing element to a new element, and delete elements form. Information about which attributes are required or optional for each tag is also available, and the form enforces proper hierarchy of elements. Using this tool, developers at different levels of familiarity with Galaxy tool development will be able to build and publish new tools to the Galaxy Toolshed.

After developing the XML tool file, multiple rounds of testing are needed to ensure it works as expected with different inputs. GTG also adds efficiency to this step of tool development by deploying a local Galaxy instance and enabling the tool developer to quickly test tools during the development process. Finally, GTG incorporates a feature of the Planemo tool (Chilton *et al*.), an easy mechanism to publish the finished tool to the Galaxy Toolshed. This final step completes the development process and encourages all developers to share their tools with the wider Galaxy community.

4.3 Implementation

GTG consists of two Drupal modules: *Galaxy Tool Generator* and *Galaxy Tool Generator UI*. The Galaxy Tool Generator module is the core of GTG and builds Drupal (VanDyk and

Westgate, 2007) webform components for each Galaxy tool XML tag. This enables the tool XML files to be rendered as an easy to understand Drupal webform. Users can create and edit XML files by adding webform components from a drop-down list and then structure the XML file content by dragging and dropping those webform components (Figure 4.1 C). Each web form component has a descriptive title to indicate its role in building a tool and a type column to specify its placement in the hierarchical structure within the XML file. The Galaxy Tool Generator UI module implements the user interfaces for interacting with XML components to build XML files, for gathering created XML files to build a Galaxy tool repository and for publishing the tool repository to the Galaxy Toolshed (Blankenberg *et al*., 2014c).

GTG provides three options to start building an XML file. First, a developer can start from scratch. This option allows users to build an XML file by selecting them from a drop down individually. A developer could also choose to upload an existing XML file as a starting point. GTG will automatically convert all XML components to their corresponding webform components and display them in correct hierarchical structure. GTG also provides a generic template, and with this option, users can start developing a tool with a predefined set of commonly used XML components.

GTG includes a Docker image which launches a Drupal website with the two GTG modules installed and enabled. The Docker image also has the planemo tool installed, which is a command line tool for Galaxy tool development (Chilton *et al*.). Planemo assists in Galaxy tool development by initializing a basic XML template and allows developers to validate XML components before publishing tools to Galaxy ToolShed. However, GTG has additional features that significantly advance the ease of tool development. For example, GTG provides XML templates and also allows users to upload any existing XML as a template; this is particularly

convenient when wrapping tools with similar input/output or when updating a tool to a new version that requires only minor changes to elements. In GTG, there is no need to verify if an XML component is valid, since all XML components are predefined and provided by GTG instead of by developers. However, Planemo does provide an easy way to publish Galaxy tools to the Galaxy ToolShed, and this feature has been adopted in the GTG application.

A GTG application can be easily launched with the shell script *launch_dev_env.sh,* available in the Github repository. In addition to launching a GTG application, the *launch_dev_env.sh* launches a Galaxy instance using the *docker-galaxy-stable (Grüning, 2018)* image created by Björn Grüning. Through a series of folders mapped between GTG and the docker container, tools being built with GTG can be quickly tested in the local Galaxy instance, providing a faster, more agile, and completely local development environment. Specifically, to connect GTG and Galaxy, launch_dev_env.sh creates a *galaxy_tool_repository* directory in the current directory in the host machine and maps this directory to the Docker container. When users select created XML files to build a tool repository on GTG, these XML files will automatically be available to the host machine. The script also creates a *shed_tools* and *database* directory in the current working directory in the host machine and maps them to the Galaxy instance docker container. The shed_tools will map all installed tools from the Galaxy container to the host machine. The *database* directory will map all job working details (e.g., job scripts, logs, stderr/stdout, outputs, etc.) to the host machine when a job is running in the Galaxy instance.

The Galaxy instance and the GTG application together can be used as an interactive development environment for wrapping Galaxy tools. The best practice is to first use GTG to quickly develop a draft working version of a tool and then publish the tool to the Galaxy Test ToolShed (https://testtoolshed.g2.bx.psu.edu/), which is similar to Galaxy ToolShed but for tool

developing and testing purposes. Once the tool is available from the Galaxy Test ToolShed, a developer can easily install it into their local Galaxy instance to view and test. GTG allows users to link the tool being edited in GTG with the tool installed in the Galaxy instance. Going to the extra effort to deploy and then install the tool from the Test ToolShed, is an important step to test the automated installation. When the developer edits the XML file by adding, deleting or updating XML components through GTG, the changes will automatically be integrated into the Galaxy instance to enable immediate testing. When the tool is complete, it can be published to the Galaxy ToolShed for other users to discover and install.

4.4 Discussion

GTG could significantly lower the bar of entry to developing Galaxy tools by allowing the construction and publication of Galaxy tools for new tool developers with little XML experience and also speed the development time for experienced Galaxy tool developers. All improvements in the speed and ease of development is a benefit to the Galaxy community, as the platform's appeal is dependent on a comprehensive, diverse and well tested set of bioinformatic tools. GTG addresses the full Galaxy tool development cycle in three steps. First, an intuitive Graphical User Interface (GUI) provides a web-based form to represent the Galaxy tool XML components. This enables developers to fill in XML tag-specific attribute values without the need to refer to the Galaxy tool XML definition documentation and automatically enforces proper element hierarchy, reducing difficult to debug errors. Next, GTG enables incremental tool development by integrating a local, easy to set up testing system. This allows developers to immediately see if tools work and iteratively add features. Finally, by integrating with the existing Galaxy tool planemo, developers can publish finished tools to the Galaxy ToolShed.

4.5 Acknowledgements & Funding

This work was supported by the National Science Foundation under grant #1443040. This work used the Extreme Science and Engineering Discovery Environment (XSEDE) Jetstream Cloud, which is supported by National Science Foundation grant number ACI-1548562 and specifically by the XSEDE allocation to MS #TG-MCB160182.

4.6 Appendix

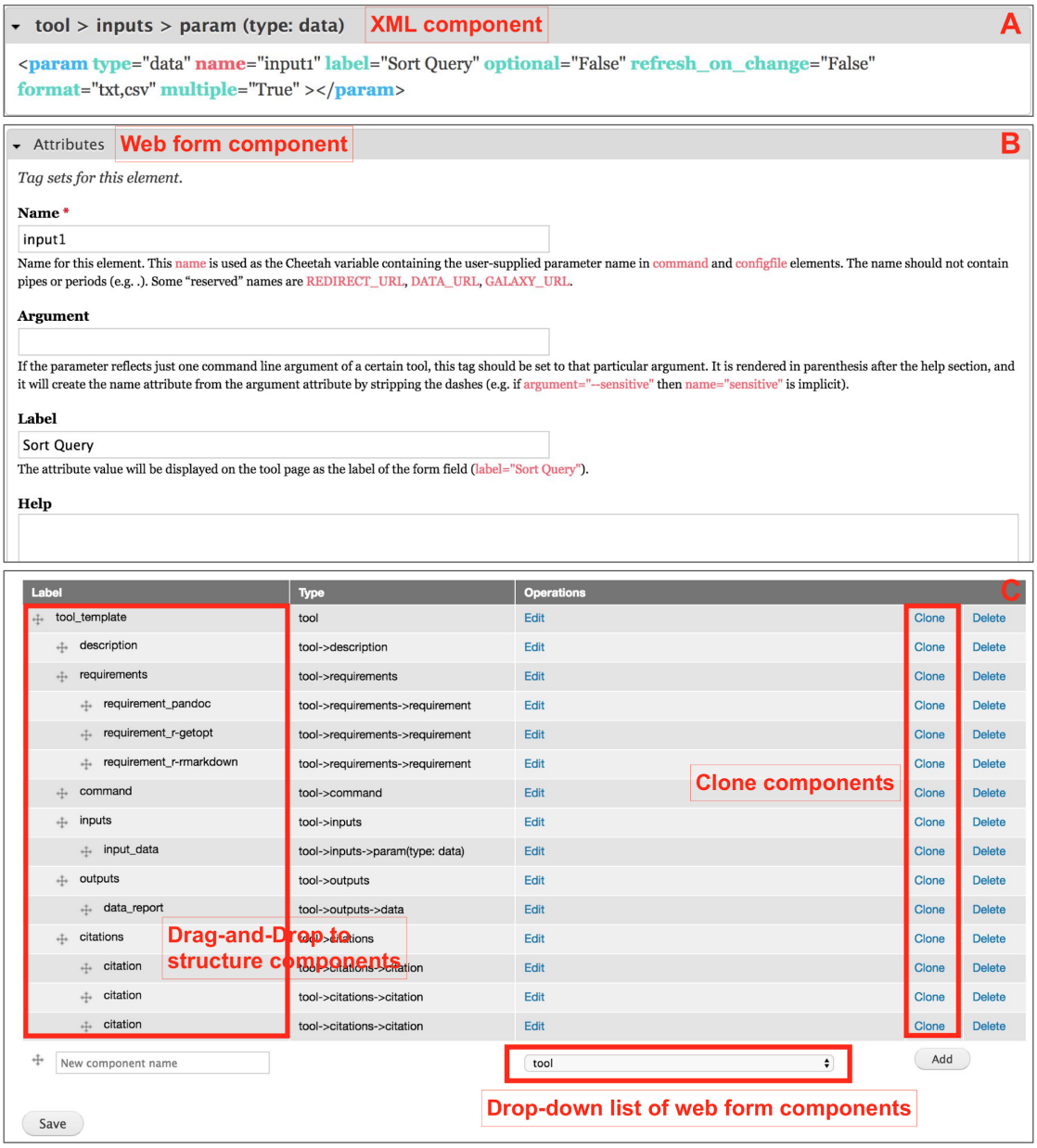


Figure 4.1. Interfaces and structure of the Galaxy Tool Generator application.

(A) an example of an XML component. (B) An example of web form interface for building and editing an XML component. (C) The interface for building a complete Galaxy XML file. A component and all its child components can be duplicated or removed by simply clicking the "clone" or "delete" button, respectively.

CHAPTER V. USE CASE: GENE FAMILIES INVOLVED IN O_3 STRESS RESPONSE FROM THREE PHYLOGENETICALLY DIVERSE SPECIES OF HARDWOOD TREES

5.1 Introduction

The following use case demonstrates how the tools and approaches developed in previous chapters help explore, analyze, and visualize data in Tripal databases. With both the data and analytical tool resources available in a community database, our next step would be to generate new biological hypotheses by analyzing the data with the tools. For this use case, I will be using public data from HGD to compare the gene expression differences resulting from abiotic stress in the form of ozone (O_3) exposure on three different hardwood tree species. Specifically, I will examine gene expression data from: green ash *(Fraxinus pennsylvanica*), northern red oak (*Quercus rubra*), and black walnut (*Juglans nigra*). The three organisms were profiled using RNA-Seq in identical exposure experiments, consisting of seedlings grown at four O_3 exposure levels (ranging from ambient to 225ppb) with RNA profiled from leaf samples taken at 7 hours, 14 days and 28 days.

The physiological reaction of trees to near-surface atmospheric O_3 levels is of particular interest in forestry research, as atmospheric O_3 levels are already increasing and expected to continue an upward trend based on current climate change models (Fann *et al*., 2015; Yue *et al*., 2015; Zhang and Wang, 2016). Ozone is a harmful air pollutant that substantially changes plant function including damaging and altering plant metabolism (Caregnato *et al*., 2015; Jolivet *et al*., 2016; Tiwari and Agrawal, 2018). The potential productivity loss and alterations in carbon sequestration in forests due to increasing O_3 is of major concern but has proved difficult to predict (Wang *et al*., 2016; Ollinger *et al*., 1997). At the individual plant level, high O_3 levels induce cellular damage, reduce photosynthesis, modify carbon allocation, and increase respiration, all of which contribute to visible leaf injury and growth reduction. However, different tree species and genotypes respond variably to O_3 exposure (Zak *et al*., 2007; Matyssek *et al*., 2010), and little is

currently understood about how this response is mediated at the genetic or gene expression level. The availability of O_3 response data from phylogenetically diverse tree species is an interesting opportunity to explore how this abiotic stress response is shared and how it differs among forest tree species at the molecular level.

To acquire the data and perform a comparative gene expression analysis, this use case will employ Tripal modules, Galaxy Aurora tools, and the GTG web application, all within the scope of HGD. First, the Tripal Elasticsearch module is used to quickly find the data, which has been stored in the site using the Tripal Analysis Expression module. After downloading the expression counts data and Biosample data, Galaxy workflows that include Aurora tools enable differential expression analyses and report significantly differentially expressed genes under different durations of O_3 exposure. Next, Aurora tools enable the extraction of the protein sequences of those significant genes and searches of the Pfam database (Bateman *et al*., 2004)with HMMER (Eddy, 1992). The final output is a set of differentially expressed genes from all three species that share the same protein families. This meta-analysis could provide initial evidence to spur a more in depth study or provide a means to filter or confirm a set of candidate genes for a scientist interested in plant O_3 response.

Tools needed in this use case will be developed with the GTG web application using the *Aurora Galaxy Tool* framework. Galaxy workflows will be added to the HGD website with the Tripal Galaxy module. Since each tool will be wrapped as an Aurora Galaxy tool which generates an HTML report, analysis results will be able to be viewed directly within the HGD website. The genes obtained in the final results report will be linked back to the individual feature pages of HGD website so that users can easily explore other related information. The final report also links the target names and accession IDs of protein families to the EMBL-EBI Protein Family database.

This use case gives an example of how Tripal Elasticsearch, *Aurora Galaxy Tool* and GTG can quickly extend the functionalities of HGD and facilitate exploratory research in hardwood tree species.

5.2 Datasets

HGD contains public data from RNA-Seq experiments for green ash, northern red oak and black walnut, all of which were produced as part of the "Comparative genomics of environmental stress responses in North American hardwoods" NSF Plant Genome Research Program (Award #1025974, PI John Carlson, Pennsylvania State University). Full experimental methods and analysis of differentially expressed genes is available for the green ash data (Lane *et al*., 2016). Manuscripts are under preparation for northern red oak and black walnut but are not overlapping in scope with this use case. Raw data including Biosample and sequencing platform details are available from NCBI (Wheeler *et al*., 2001): green ash (SRX858416-SRX858431), northern red oak (SRX858564-SRX858610), and black walnut (SRX858655-SRX858666). Briefly, seedlings were acclimated in continuously stirred tank reactors and then exposed to one of four O_3 treatment levels: ambient (~10ppb), 80ppb, 125ppb or 225ppb. Leaves were sampled from six seedlings in each treatment group at 7hr, 14d and 28d. For green ash and black walnut, RNA samples from the six biological replicates were pooled and sequenced on an Illumina MiSeq. For northern red oak, individual biological replicates were sequenced on an Illumina HiSeq 2000.

None of the three species have available fully sequenced genomes for use as a reference. HGD provides access to assembled transcriptomes, which were created from larger sets of RNA-Seq reads that included the O_3 treatment experiments examined here. These putative unique transcripts are referred to as "genes" in this chapter for brevity. Additional informatics analysis of

the transcriptomes available from HDG includes: functional annotation using BLAST (Altschul *et al*., 1990) and InterProScan (Jones *et al*., 2014), amino acid sequences from ORF prediction, and gene expression information (normalized and raw) generated by mapping reads back to the transcripts. Informatics protocols are described online and follow the methods of Lane et al., 2016.

5.3 Discovering and downloading data with Tripal Elasticsearch and Tripal Analysis Expression

Although users can find all necessary input data for this use case in the HGD website by clicking through the menu navigation, this generally requires that a user to already know what data they need and where it is located in the site. The search functionality provided by the Tripal Elasticsearch module makes it easier and faster to find the data, especially if the user is unsure where it might be or if it exists. Below is the list of datasets I need for the use case for each species:

- RNA-Seq Sample data - the treatment information for each sample. Each row is a sample and each column is a set of treatment levels for a given treatment (control or O_3 exposed; exposure duration: 7 hours, 14 days, or 28 days).
- RNA-Seq Count data - the number of sequence reads mapped to each transcript for each sample. Each row is a gene and each column is a sample.
- Predicted protein sequences from each transcript

Frequent HGD database users may be able to easily find the expression and protein sequences data, while users who are not familiar with the HGD database can use the full site search with simple and intuitive keywords to discover information related to their interests. The count data can be obtained from species-specific O_3 experiment pages; the treatment and protein sequences data can be obtained from species-specific transcriptome assembly pages (Figure 5.1).

I was able to locate these web pages with these keywords: “green ash expression ozone”, “green ash ozone treatment”, “fraxinus ozone”, “black walnut expression zone”, “black walnut ozone treatment”, “juglans ozone”, “green ash protein sequences ozone”, and “black walnut protein sequences ozone”. Notably, the text on the HGD pages of interest use the word “ozone” but do not use the abbreviation “O_3”. Site wide searches with phrases such as “Quercus O3” or “walnut O3” do not return ozone relevant results. This highlights a potential for improvement in search results by incorporating scientific synonym substitutions in the search engine that would automatically search for “ozone” if the user provides the keyword “O_3”.

The species specific O_3 experiment pages include experimental methods, sequencing statistics, links to raw reads in NCBI, and a link to download the raw gene read-mapping count data as a CSV file (Figure 5.2). This file is formatted with a transcript on each row and individual samples as columns. Numbers are raw counts of reads from each sample mapped uniquely to each reference transcript. Importantly for the analysis pipeline, the first row contains sample (library) IDs that need to exactly match the sample IDs in the treatment data (Figure 5.3). The treatment data is available on the same page under the ‘Description’ pane section, however, the data is not stored in a downloadable CSV file. For this use case, users need to manually create a CSV file in their local computer to store the treatment data. The description for the file format is part of the Galaxy workflow, enabling users to read the detailed description about each treatment can be found under the ‘Biosamples’ pane section and create the properly formatted file for their experimental design. The predicted protein sequences data is available and can be downloaded as a FASTA file. In this use case, the user can provide the URL of the FASTA file to the Galaxy pipeline to perform the data analysis.

5.4 Data analysis and visualization with Tripal Galaxy module and Aurora Galaxy Tools

5.4.1 Analysis workflow summary

After collecting all the necessary datasets, they are submitted to Galaxy workflows. All workflows use *Aurora Galaxy Tool* which generate HTML reports and enable results to be viewed directly from within the HGD website. HGD has a central Galaxy tool page that lists the available Galaxy workflows, any of which can be used by a user with an account. User accounts are free and easy to request through the site. For this use case, both previously existing and new workflows will be used. Only admin users can create workflows and add them to the site, however, the HGD development team accepts requests for new workflows. I will discuss how workflows are created and added to the HGD website in the next sections, including how *Aurora Galaxy Tools* may be developed with the GTG web application. This section focuses on how the web-based analysis tools can help Tripal database end users to perform bioinformatic data analysis and visualize the results.

The data analysis goal is to identify differentially expressed genes under conditions of different O_3 exposure durations in green ash, black walnut and northern red oak, and to find if there are any protein families that occur in the three lists. This is a basic exploratory biological question, with a comparative genomics approach, to identify the protein families involved in O_3 stress response across all organisms of interest. To achieve this goal, the transcript expression values will be statistically evaluated using DESeq2, a differential expression package that uses a model based on the negative binomial distribution and is designed for high throughput RNA sequencing data (Love *et al*., 2013). The results are a set of significant genes (padj < 0.1) for each organism. I

extracted the protein sequences for the identified significant genes from the FASTA file and compared them to the profiles of the Pfam database (Bateman *et al*., 2004) with the HMMER tool (Eddy, 1992) to identify their protein families. Finally, I compare the results from the three organisms, yielding a set of protein families that have at least one significant gene from each organism. Figure 5.4 illustrates the analysis workflow of the entire use case. The final result is an HTML table with a row for each protein family. Columns include the accession of the protein family followed by significant genes from green ash, black walnut and northern red oak that belong to that family. Each item in the table is a URL link to the corresponding web resource.

When implementing the analysis workflow above, two Galaxy workflows are involved: 1) the *deseq2-hmmscan-pfam* workflow which performs the differential gene expression analysis and the hmmscan search analysis; 2) the *hmmscan-pfam-merger* workflow which merges the results from all *deseq2-hmmscan-pfam* analyses. Specifically, I will run the *deseq2-hmmscan-pfam* workflow separately for green ash, black walnut and northern red oak data and then merge the results to generate the final report. The *Aurora Galaxy Tools* involved in these use cases are:

- aurora_deseq2 (Galaxy Version 2.2.2): a wrapper of the bioconductor DESeq2 package (version 1.16.1) (M. Love *et al*., 2014)
- aurora_statonlab_hmmscan (Galaxy Version 1.0.0): a wrapper of HMMER3 (version 3.2.1) (Eddy, 2010)
- aurora_statonlab_hmmscan_merger (Galaxy Version 1.0.0): a wrapper of an R script that finds common protein families in the result sets

5.4.2 Analysis implementation

Create Galaxy workflows in HGD-connected Galaxy server The *deseq2-hmmscan-pfam* and the *hmmscan-pfam-merger* workflows were built in Galaxy's graphic workflow editor (Figure 5.5). The graphic workflow editor provides intuitive interface that allows users to drag and connect tools with inputs and outputs. The aurora_deseq2, aurora_statonlab_hmmscan, and aurora_statonlab_hmmscan_merger tools in the two workflows are the key components. All those tools are Aurora Galaxy tools and wrapped with the GTG web application. The tool development details will be discussed in the next section. I created the workflows in the Galaxy server (http://treegenes.cam.uchc.edu/galaxy) maintained by the Plant Computational Genomics Lab at University of Connecticut. The Galaxy server is connected to the HGD database with the Tripal Galaxy module.

Run deseq2-hmmscan-pfram workflow Input to this workflow requires the expression and treatment data collected from the HGD website. The HGD web interface, powered by Tripal Galaxy, guides the user through the 5 steps of the workflow (Figure 5.6):

- Step 1: An instruction step which has information about what tools are used in this workflow, including versions
- Step 2: A prompt to upload expression data or select a previously uploaded file for use
- Step 3: A prompt to upload treatment data or select a previously uploaded file for use.
- Step 4: A form for the user to specify parameters for DESeq2 including the statistical model details. The DESeq2 tool will perform the differential transcript expression analysis.
- Step 5: A form for the user to specify the parameters for the hmmscan software, including e-value threshold and score threshold. This step will use the aurora_statonlab_hmmscan

tool to query protein sequences from identified significant genes against the Pfam database. At each step, the workflow has instructions on what data is needed and how to use the tool. Users are able to preview their input before they submit the workflow in a final step. Once the workflow is submitted, Tripal Galaxy transfers the data from HGD to the connected Galaxy server and invokes the workflow execution there.

Results of deseq2-hmmscan-pfam workflow HGD users can view the status of all submitted workflows by clicking "My Account -> Analyses". Once the status of workflow becomes 'completed', users will be able to view the analysis results. Depending on the number and types of tools in the workflows and the amount of data submitted, the time to completion can be variable. Users can opt to be notified via email when the workflow is complete. The link to the workflow results includes the output information from each workflow step, presented along a sidebar for navigation (Figure 5.7). Item 1 and 2 are the uploaded expression and treatment datasets, respectively. Item 3 ("aurora_deseq2 report on data 2 and data 1 (ok)") provides a link to the HTML report from the aurora_deseq2 tool (Figure 5.8). Item 4 is a CSV file consisting of all significant genes and associated statistical test results. Item 5 is an HTML report from the aurora_statonlab_hmmscan tool. Item 6 is the Pfam database search results in CSV format. This file should be downloaded to the local computer for use in the *hmmscan-pfam-merger* workflow. For this use case, I ran the workflow for the datasets of each the three tree species and collected three datasets of significantly differentially expressed transcripts assigned to Pfam database entries.

Run hmmscan-pfam-merger workflow This workflow merges the hmmscan Pfam database search results and returns a table in an HTML report. Figure 5.9 shows the 3 steps of the workflow in the HGD website:

- Step 1: An instruction step which has information about what tools are used in this workflow.
- Step2: A prompt to upload two or more sets of the Pfam database search results collected from the *deseq2-hmmscan-pfam* workflows.
- Step 3: A description of the tool that will merge the results and generate final analysis report (no user action required).

Analysis results of hmmscan-pfam-merger workflow Each row consists of feature IDs of significant genes from all organisms (green ash, black walnut and northern red oak in this case), target name and accession ID of returned protein family from the hmmscan Pfam database search. Only the records that have at least one significant gene from each organism are kept. It is possible that multiple genes from a single organism have a common protein family hit, and that one single gene has multiple different protein family hits. The target names and accession IDs are linked to the EMBL-EBI Protein Family database. The feature IDs are linked back to the feature page in HGD.

5.4.3 Results and discussion

O_3 at ground-level is one of the most widespread air pollutants and is the most damaging to plants (Ozone effects on plants (USDA, ARS)). O_3 enters plant leaves through stomata and oxidizes plant tissue during respiration, leading to plant leaf damage and reduced plant survival (Ozone effects on plants (USDA, ARS)). This use case demonstrates how the web-based computational tools introduced in Chapter II-IV could help identify candidate genes from hardwood tree species that respond in expression to increased O_3 exposure.

The differential gene expression analysis identified 26, 2095, and 11571 significant genes in black walnut, green ash, and northern red oak, respectively (Table 5.1). This wide range of responsive transcripts could be due to experimental or physiological factors. From the experimental design, the much higher sequencing depth and the sequencing of individual biological replicates for the northern red oak samples lends significantly more statistical power to detect differential expression. Both black walnut and green ash were sequenced at a much lower depth and the replicates were pooled for sequencing. More than 90% of genes have an average read count lower than 5 across all 12 samples in black walnut and green ash, leading to a lack of enough data for many genes to accurately detect differential expression. In contrast, only 68% of northern red oak genes have an average read count of less than 5. This is reflective of the reality of mining data from public sources as opposed to designing new experiments to be conducted; often the experimental techniques across datasets are different and both data analysis and interpretation of the results must carefully consider those differences. No physiological data has been provided from the O_3 exposure experiments conducted to generate this data, but the wide range in number of genes responding, even between black walnut and green ash, suggests follow up experiments profiling O_3 exposure tolerance in these species may be warranted.

By scanning the protein sequences of those significant genes against the Pfam database with HMMSCAN tool, 14 common protein families were identified across the three species. Each shared protein family included at least one differentially expressed gene in each tree species. Genes may belong to multiple protein families, with the results representing 6 unique genes out of 14 protein family matches in black walnut; 62 unique genes out of 144 protein family matches in green ash; and 204 unique genes out of 544 protein family matches in northern red oak (Table 5.1).

The set of leucine-rich repeat (LRR) families (LRR_1, LRR_4, LRR_6, and LRR_8) are notable both for their shared leucine-rich repeat domain and the large number of putative genes responding in the green ash and northern red oak experiments. The LRR protein structural motif is composed of repeating 20~30 hydrophobic, leucine-rich stretches. Prior research suggests that proteins with LRR domains are involved in plant stress response, and they have specifically been found to respond to elevated O_3. In an examination of O_3 susceptible and tolerant poplar hybrid genotypes, semi-quantitative RT-PCR confirmed up regulation of Fs23A-LRP, a gene coding for a leucine-rich repeats protein, in both tested genotypes after O_3 exposure (Rizzo *et al*., 2007). Leisner et al. found that among the 25 genes with the greatest log fold change in response to elevated O_3 in soybean flower tissue, 5 genes encode Leucine-rich repeat receptor-like protein kinase (Leisner *et al*., 2014). Additionally, LRR is also involved in human's response to O_3 exposure. A recent study published by Speen et al. (2018) showed that O_3-derived Oxysterols form adducts with NLRP2 (NACHT, LRR and PYD domains-containing protein 2) in the leucine-rich repeat region, potentially contributing to inflammation in airway epithelial cells.

As a strong pro-oxidant, increased O_3 exposure leads to extra reactive oxygen species that exceed the scavenging capacity by antioxidants, disrupting the pro-oxidant/antioxidant balance and causing oxidative stress to organisms (Demidchik 2015; Birben et al. 2012). Across many plant species, molecular profiling of O_3 stress response has indicated a strong oxidation-reduction response in cells in an attempt to mitigate damage directly from oxidation or from reactive oxygen species. This is supported in hardwood tree species by the presence of genes from the adh_short and adh_short_C2 pfam groups. adh_short is a large family of that consists mostly of oxidoreductases, while adh_short_C2 is a more specific class of reductase enzymes that catalyze the last step of fatty acid biosynthesis. These genes provide a basis to spur additional research into

how trees tightly regulate the balance between pro-oxidants and antioxidants to maintain vital cellular and biochemical functions under redox stress (Kohen and Nyska, 2002; Pilch *et al*., 2014; Kisaoglu *et al*., 2013).

The RuBisCo_small (ribulose bisphosphate carboxylase, small chain) and RbcS (ribulose-1,5-bisphosphate carboxylase small subunit) Pfam results both indicate potential impacts of O_3 exposure on photosynthesis and carbon fixation. This alteration of regulation of Rubisco is a widely observed molecular response of ozone exposure in other trees, including *Pinus halepensis* (Pelloux *et al*., 2001), *Poplar* spp. (Brendley and Pell, 1998; Landry and Pell, 1993; Bagard *et al*., 2008), and *Betula pendula* (Pääkkönen *et al*., 1996), as well as other non-woody plants such as *Zea mays* (Louis Leitao *et al*., 2007; L. Leitao *et al*., 2007), *Arabidopsis thaliana (Brendley and Pell, 1998; Rao et al., 1995)*, and *Phaseolus vulgaris* (Leitao *et al*., 2003).

Significant further work to understand the results from this use case remains to be done, particularly characterizing and comparing the individual differentially expressed genes in each protein family and in understanding the protein family functions not described above. Further, the small number of significant genes in black walnut resulted in a limited number of shared protein families being identified. Additional analysis incorporating protein families with differentially expressed genes from only two of the three species could increase the number of detected protein families. Effort in exploring the individual genes in each protein family and identifying orthology would be needed to confidently point to a broadly conserved evolutionary response to O_3 exposure across hardwood trees. This additional analysis is outside the scope of this dissertation. However, this use case demonstrates the types of comparative analysis enabled through public community databases and web-based analysis tools.

5.5 *Aurora Galaxy Tool* Development with GTG

In the previous sections, I have discussed how HGD database users can find data with search functionalities implemented by the Tripal Elasticsearch module, download gene expression experimental data aggregated by the Tripal Analysis Expression module, integrate collected data into workflows and run workflows with the Tripal Galaxy module, and explore analysis results in an HTML report within the HGD website with the help of *Aurora Galaxy Tools*. In this section, I will discuss how HGD admin users or maintainers can utilize the GTG web application to develop *Aurora Galaxy Tools* and significantly extend the functionalities of the database. Although the three Aurora Galaxy tools used in this case were created with GTG, I will use the aurora_statonlab_hmmscan tool as an example, as the development process for all *Aurora Galaxy Tools* is similar.

5.5.1 Launch GTG web application

A shell script named *launch_dev_env.sh* is available from the GitHub repository of GTG to quickly start the web application (https://github.com/MingChen0919/gtgdocker). The only dependency is a local installation of Docker. Users can download a copy of the *launch_dev_env.sh* file to their local computer and run this one line command below to launch GTG:

```
sh launch_dev_env.sh
```

This command line starts GTG, launches a connected Galaxy instance, and creates a *gtg_dev_dir* folder as a workspace within the current working directory. Within this workspace are three folders that are automatically mounted to folders in the GTG Docker container. The functionalities of these folders have been explained in Chapter IV. Users can access GTG and the

connected Galaxy instance through the browser at URLs http://127.0.0.1:8080 and http://127.0.0.1:8081, respectively (Figure 5.11).

```
. gtg_dev_dir
 ├──── database
 ├──── galaxy_tool_repository
└── shed_tools
```

5.5.2 Create Galaxy tool XML file

Initialize a new tool XML file Clicking the "Create Tool XML" tab will initialize a new tool XML file. GTG offers the option to use a template when building a new tool, including a template specifically for an *Aurora Galaxy Tool*. To use this feature, I entered "rmarkdown_report.xml" into the XML file name field (Figure 5.12), selected "Aurora Galaxy Tool", and then clicked "Save". GTG uses the integrated template to create a web form which consists of a list of predefined web form components.

Add web form components Once an initial XML file has been generated, components can be added and edited through the web form components interface to build up the XML file. Users can manipulate the components through the web form components interface (Figure 5.13). For example, a "tool->requirements->requirement" component can be added to the XML file to define tool dependencies for the aurora_statonlab_hmmscan. The aurora_statonlab_hmmscan tool depends on the HMMER (Eddy, 1992) tool. To add HMMER as a dependency, I selected "tool->requirements->requirement" and then clicked "Add". After that, I was directed to the component edit interface which allows me to specify a package name and version. This package must be a valid package from the bioconda repository (Grüning *et al*., 2018). The package name

for the HMMER tool in bioconda is “hmm”, and the version 3.2.1 is used in aurora_statonlab_hmmscan.

In addition to the “hmm” tool, the aurora_statonlab_hmmscan also depends on other tools, including “seqkit” (0.8.1) (Shen *et al*., 2016) “r-tidyverse” (1.1.1) (Wickham, 2017). After a component has been added, it can be dragged and placed in the correct location. For example, the “tool->requirements->requirement” component should be placed under the “tool->requirements” components. I used the same process as above to add other components into the requirements, inputs, outputs, and citations sections.

5.5.3 Add shell scripts to the command section

The command section specifies the shell script to use to execute the job. A shell script can be added, edited and updated directly through the “tool->command” component edit form. However, it is often easier and more convenient to open the rmarkdown_report.sh file within the galaxy_tool_repository directory and edit the shell script there. The content of rmarkdown_report.sh will automatically be inserted into the command section by GTG when the tool XML page is being viewed. GTG recognizes the .sh file that has the same base name as the Galaxy tool XML file and plugs in the content of the .sh file into the command section. This feature is available for developing any tools, not just *Aurora Galaxy Tools*. For this use case, the .sh file should be named as “rmarkdown_report.sh”, as the XML file is named as “rmarkdown_report.xml”.

5.5.4 Add analysis logic

After completing the web interface layer of a Galaxy tool (creating the tool XML file), analysis logic can be created. For *Aurora Galaxy Tools*, the shell script in the command section of the XML file does not contain any analysis logic. The only purpose of it is to collect user inputs and parameter values through command line arguments. Arguments and parameters added to the getopt.csv file (equivalent to command-line-arguments.csv in Chapter III) can be recognized by the rmarkdown_report.Rmd file to create data analysis logic. Details about how to use getopt.csv (equivalent of command-line-arguments.csv in Chapter III), galaxy_outputs.sh (equivalent of expose-outputs-to-galaxy-history.sh in Chapter III), build-and-run-job-scripts.sh, rmarkdown_report_render.R and rmarkdown_report.Rmd can be found in Chapter III.

5.5.5 Publish a draft tool to Test ToolShed for iterative development

Instead of completing the tool development in GTG and then publishing it to the ToolShed, it is recommended to first publish a draft tool to the Test ToolShed and install the preliminary version for iterative development. This is an important step to test automated installation, as the normal path to tool installation for Galaxy instances is from the ToolShed. The draft tool does not have to be fully functional but should have a full fledged requirements section. This is because Galaxy requires a well established requirements section to install all required tool dependencies, which are necessary for testing the tool during the iterative development. Installing tools into Galaxy requires privileged permissions. The credential below can be used to login into the connected Galaxy instance launched by GTG to install the draft tool for interactive tool testing:

Username: admin@galaxy.org
Password: admin

Below are the steps to publish the draft tool to Test ToolShed:

- Click “Build Tool Repository”, select “rmarkdown_report.xml” and then click “Update XMLs in galaxy_tool_directory folder” to add the XML file to the galaxy_tool_directory folder (Figure 5.15).
- Click “API Keys” to add the Test ToolShed credential (Figure 5.16).
- Click “Publish tool repository” and fill the form and then click “Publish to Test Tool Shed” (Figure 5.17).

5.5.6 Sync tool updates in GTG to Galaxy instance

When installing a tool from the ToolShed or Test ToolShed into the Galaxy instance, the tool repository will appear in the shed_tools directory within the workspace folder (gtg_dev_dir). The tool repository within GTG can be linked to the tool repository in the Galaxy instance so that any updates in GTG can be synced to the Galaxy instance and become testable immediately. Below is how to link the tool repository:

- Enter the path to a tool repository in the *shed_tools* folder into “Sync to Galaxy” field in Figure 5.15. The path is a relative path to the *shed_tools* folder.
- Click “Update” to complete the linking.

To sync tool updates in GTG, I selected “Update XMLs in galaxy_tool_repository folder”. Then the updates can be tested in Galaxy. When a fully functional tool is ready after many rounds of iteration and testing, the ToolShed API key can be added through the interface in Figure 5.16 to publish the tool to ToolShed for other people to use.

5.5.7 Shut down and remove GTG

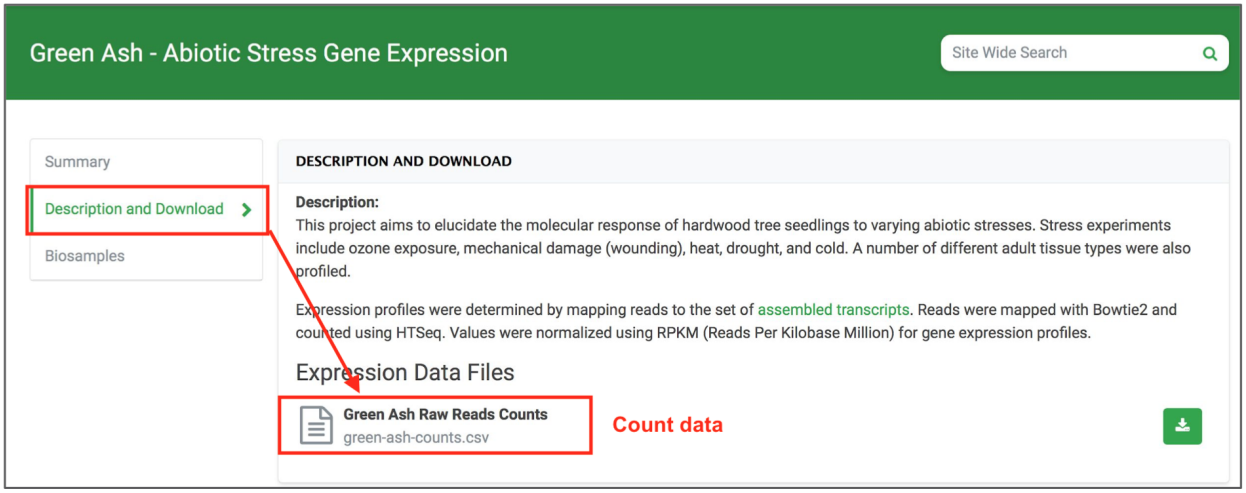
Shutting down and removing GTG is as simple as running a one line command shown below. The command will stop and remove the containers running GTG and Galaxy instance. The workspace folder gtg_dev_dir can be removed directly from your system if you no longer need the source code of your Galaxy tool repository.

```
docker stop GTG gtg_galaxy
```

5.5 Appendix

Table 5.1 Shared protein families of differentially expressed genes under different O_3 exposure across three hardwood tree species

Protein family	Number of Significant Genes in Each Protein Family (Total number of differential genes)		
	Black Walnut (26)	**Green Ash (2095)**	**Northern Red Oak (11571)**
Adh_short: the short-chain dehydrogenases/reductase family (SDR).	1	10	30
adh_short_C2: Enoyl-(Acyl carrier protein) reductase	1	8	20
CENP-T_C: Centromere kinetochore component CENP-T histone fold	1	1	2
Epimerase: NAD dependent epimerase/dehydratase family	1	13	24
Glyco_hydro_18: Glycoside hydrolase family 18	1	1	2
KR: Kringle domain	1	9	18
LRR_1: Leucine-rich repeat	1	17	101
LRR_4: Leucine-rich repeat	1	21	116
LRR_6: Leucine-rich repeat	1	15	78
LRR_8: Leucine-rich repeat	1	19	114
P450: Cytochrome p450	1	23	33
RbcS: Ribulose-1,5-bisphosphate carboxylase small subunit	1	2	2
RuBisCo_small: Ribulose bisphosphate carboxylase, small chain	1	2	1
THF_DHG_CYH_C: Tetrahydrofolate dehydrogenase/cyclohydrolase, NAD(P)-binding domain	1	3	3
Unique Genes/Total Hits	**6/14**	**62/144**	**204/544**





Green Ash ozone 12 - 28 days, 225ppb	GA-CH12-28D_12	Illumina MiSeq	159,444	48,152,088
Green Ash ozone 11 - 28 days, 125ppb	GA-CH11-28D_11	Illumina MiSeq	258,322	78,013,244
Green Ash ozone 10 - 28 days, 80ppb	GA-CH10-28D_10	Illumina MiSeq	243,475	73,529,450
Green Ash ozone 9 - 28 days, control	GA-CH9-28D_9	Illumina MiSeq	220,432	66,570,464
Green Ash ozone 8 - 14 days, 225ppb	GA-CH12-14D_8	Illumina MiSeq	241,871	73,045,042
Green Ash ozone 7 - 14 days, 125ppb	GA-CH11-14D_7	Illumina MiSeq	253,817	76,652,734
Green Ash ozone 6 - 14 days, 80ppb	GA-CH10-14D_6	Illumina MiSeq	213,649	64,521,998
Green Ash ozone 5 - 14 days, control	GA-CH9-14D_5	Illumina MiSeq	225,466	68,090,732
Green Ash ozone 4 - 7 hrs, 225ppb	GA-CH12-7hr_4	Illumina MiSeq	250,059	75,517,818
Green Ash ozone 3 - 7 hrs, 125ppb	GA-CH11-7hr_3	Illumina MiSeq	247,864	74,854,928
Green Ash ozone 2 - 7 hrs, 80ppb	GA-CH10-7hr_2	Illumina MiSeq	228,559	69,024,818
Green Ash ozone 1 - 7 hrs, control	GA-CH9-7hr_1	Illumina MiSeq	209,532	63,278,664

Figure 5.1 Collecting data from HGD website.

The green ash Ozone Treatments and Transcriptome Assembly web pages are the main resources for information and downloads, including expression data, sequence library descriptions and amino acid sequences.

gene_ID	GA10	GA11	GA12	GA1	GA2	GA3	GA4	GA5	GA6	GA7	GA8	GA9
Fraxinus_pennsylvanica_120313_comp100024_c0_seq1	0	0	0	0	0	1	0	0	1	0	0	0
Fraxinus_pennsylvanica_120313_comp100050_c0_seq1	0	0	0	0	0	0	0	0	0	0	0	0
Fraxinus_pennsylvanica_120313_comp10005_c0_seq1	0	0	0	0	0	0	0	0	0	0	0	0
Fraxinus_pennsylvanica_120313_comp100063_c0_seq1	0	0	0	1	1	0	1	1	0	1	0	0
Fraxinus_pennsylvanica_120313_comp10006_c0_seq1	0	1	0	0	0	0	0	0	0	0	0	0
Fraxinus_pennsylvanica_120313_comp10007_c0_seq1	0	0	0	0	0	0	0	0	0	1	0	0
Fraxinus_pennsylvanica_120313_comp100093_c0_seq1	0	6	5	1	1	0	0	0	3	4	6	0
Fraxinus_pennsylvanica_120313_comp1000_c0_seq1	0	0	0	0	0	0	0	0	0	0	0	0
Fraxinus_pennsylvanica_120313_comp10011_c0_seq1	0	0	0	0	0	0	0	0	1	0	0	0

Figure 5.2 Expression data format.

Expression data is provided as a downloadable CSV file with each unique transcript on a row and the twelve RNA-Seq libraries as columns. Numbers represent unique mapped reads.

sample_id	time	ppb
GA1	7 hours	control
GA2	7 hours	80ppb
GA3	7 hours	125ppb
GA4	7 hours	225ppb
GA5	14 days	control
GA6	14 days	80ppb
GA7	14 days	125ppb
GA8	14 days	225ppb
GA9	28 days	control
GA10	28 days	80ppb
GA11	28 days	125ppb
GA12	28 days	225ppb

Figure 5.3 Treatment data format.

For green ash, twelve RNA-Seq libraries were produced from the O_3 experiment, with four treatment levels and three time points. For the workflow, the user must create this matrix from the detailed descriptions of the samples on HGD.



Figure 5.4 Analysis workflow and final report.

(A) The analysis workflow starts with raw data (green: expression count data; blue: treatment data; orange: protein sequence data) from HGD (boxes) and progresses through multiple stages of analysis (ovals) that yield a final set of interesting candidate genes for further research. (B) The final report, viewed from a web browser at HGD.

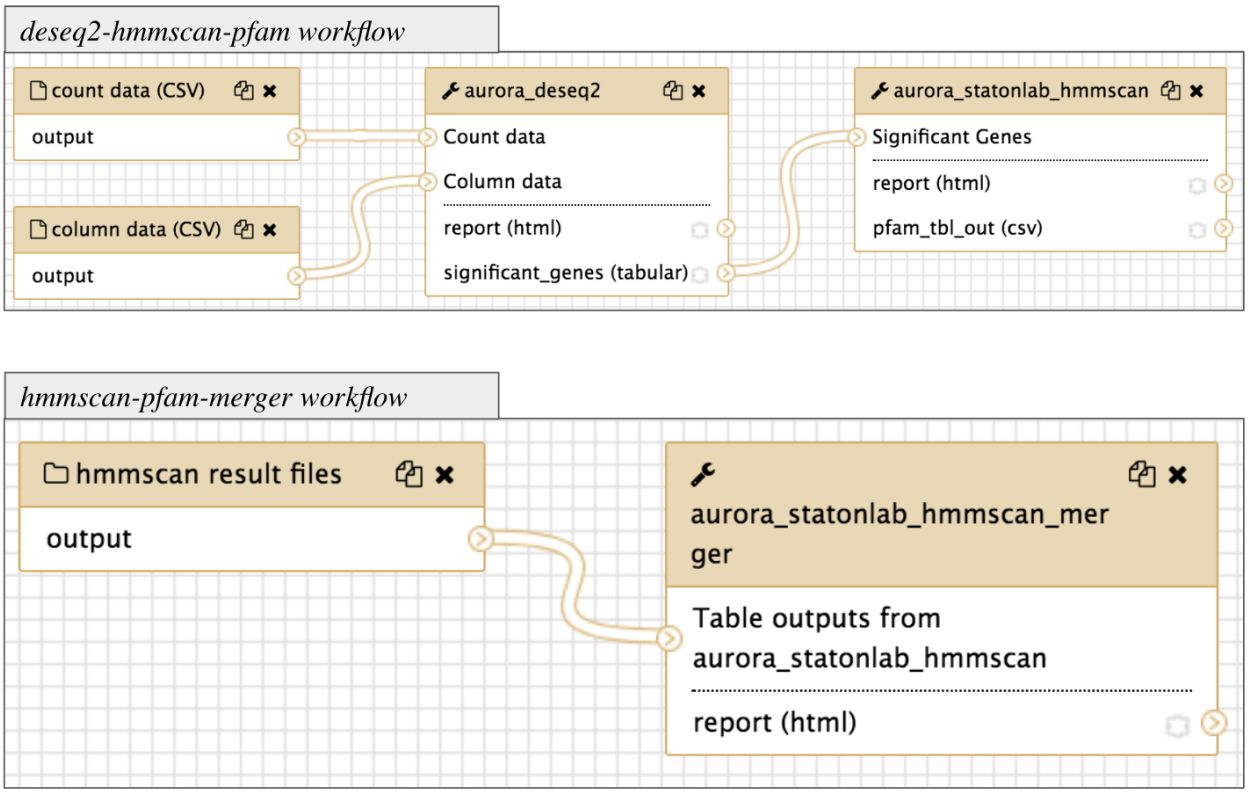


Figure 5.5 the view of deseq2-hmmscan-pfam and hmmscan-pfam-merger workflows in Galaxy's graphic workflow editor.

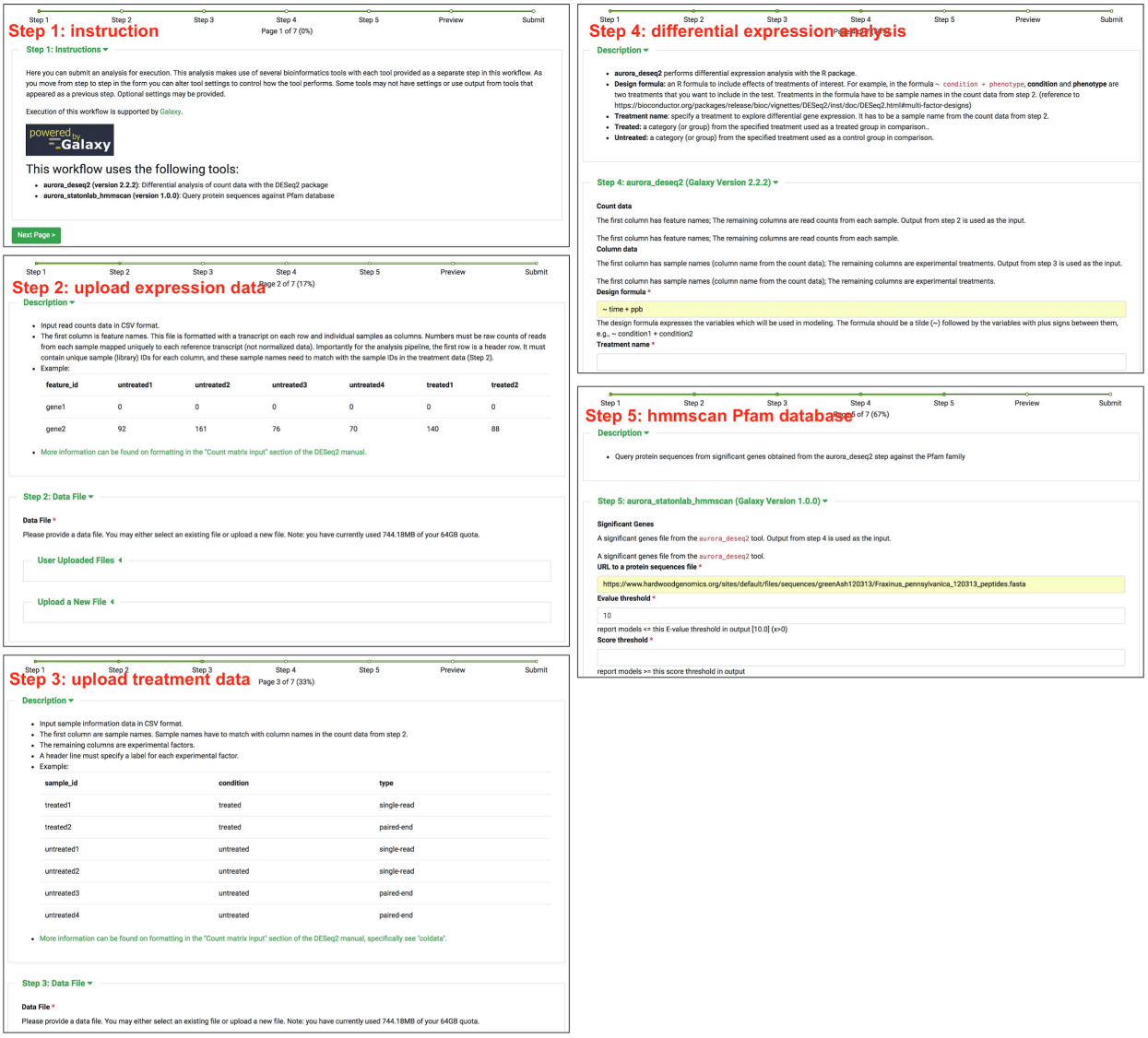


Figure 5.6 Steps of deseq2-hmmscan-pfam workflow in HGD website.

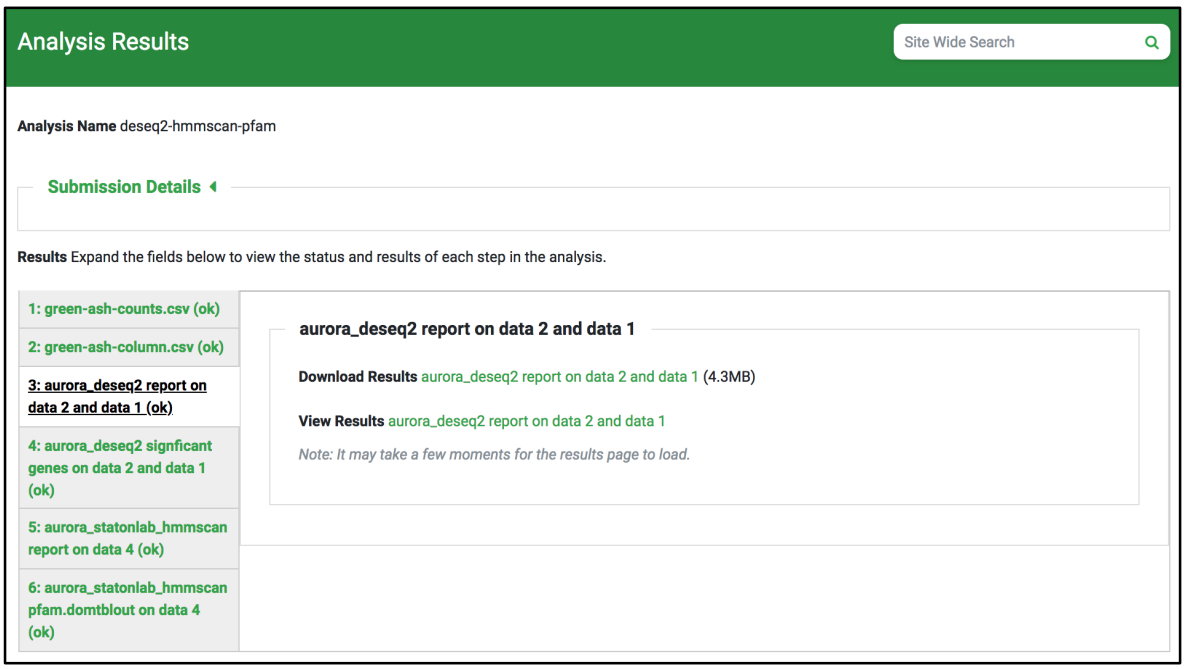


Figure 5.7 Analysis results page of the *deseq2-hmmscan-pfam* workflow.

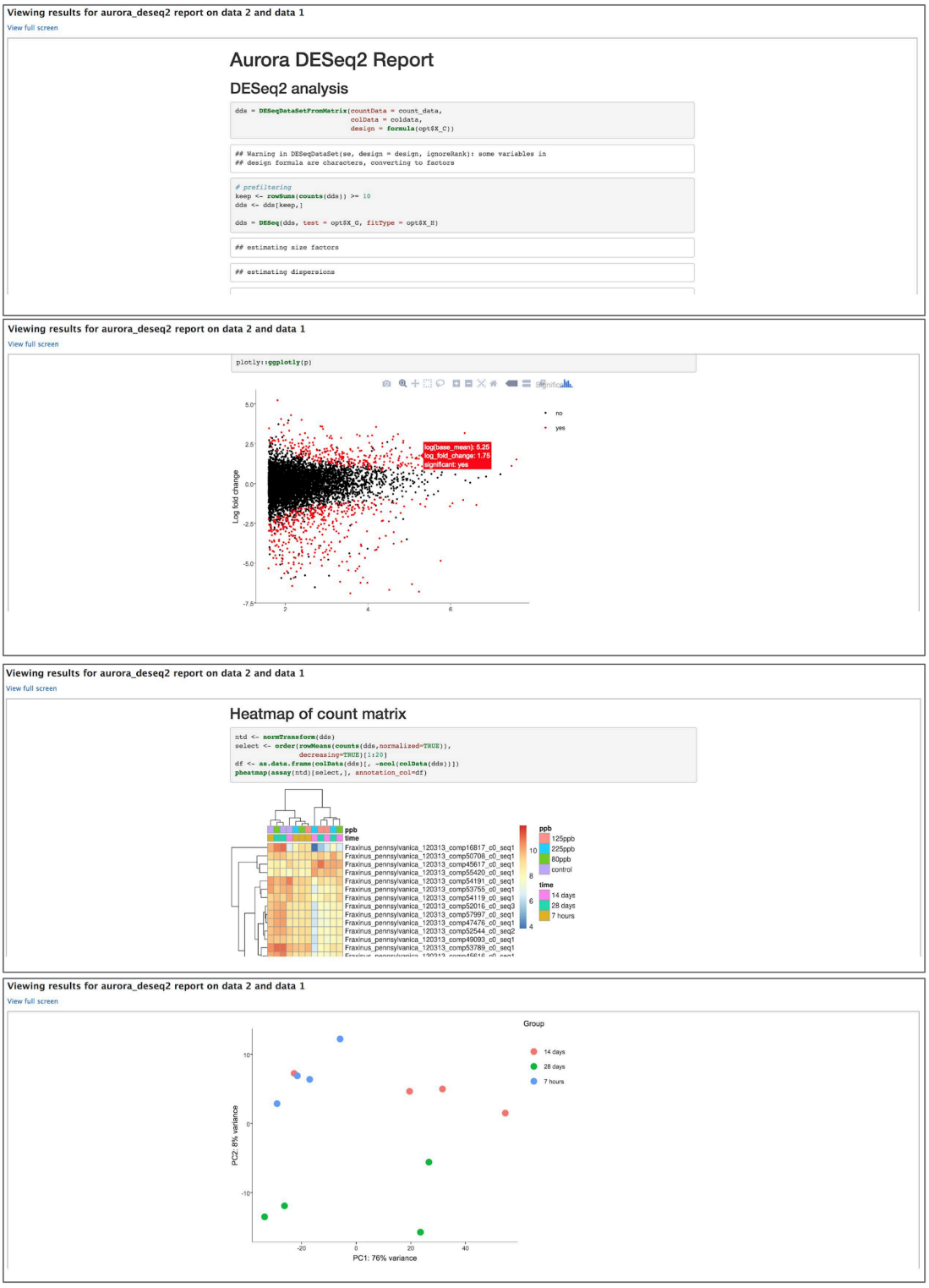


Figure 5.8 screenshots of aurora_deseq2 analysis report

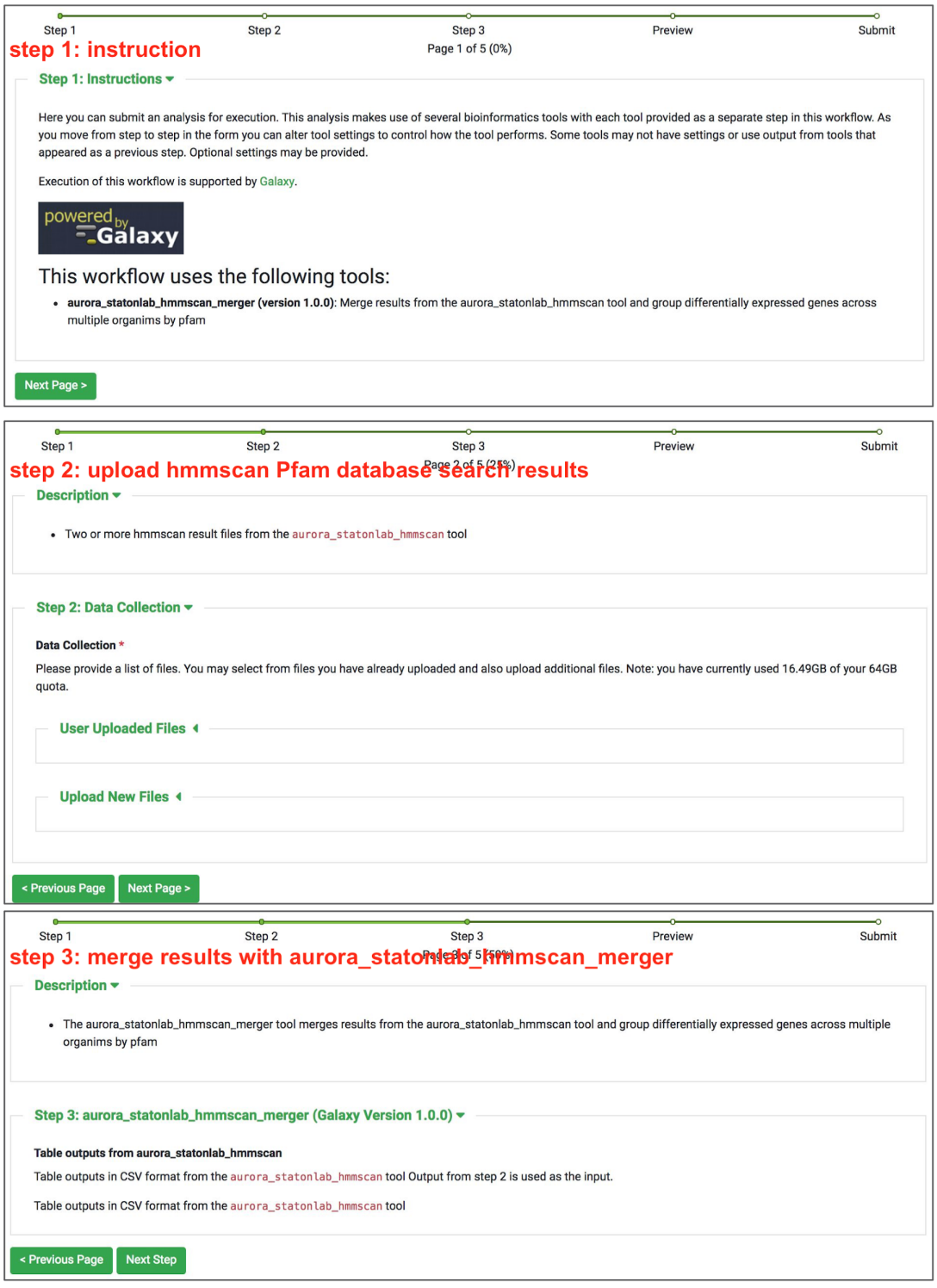


Figure 5.9 Steps of hmmscan-pfam-merger workflow.

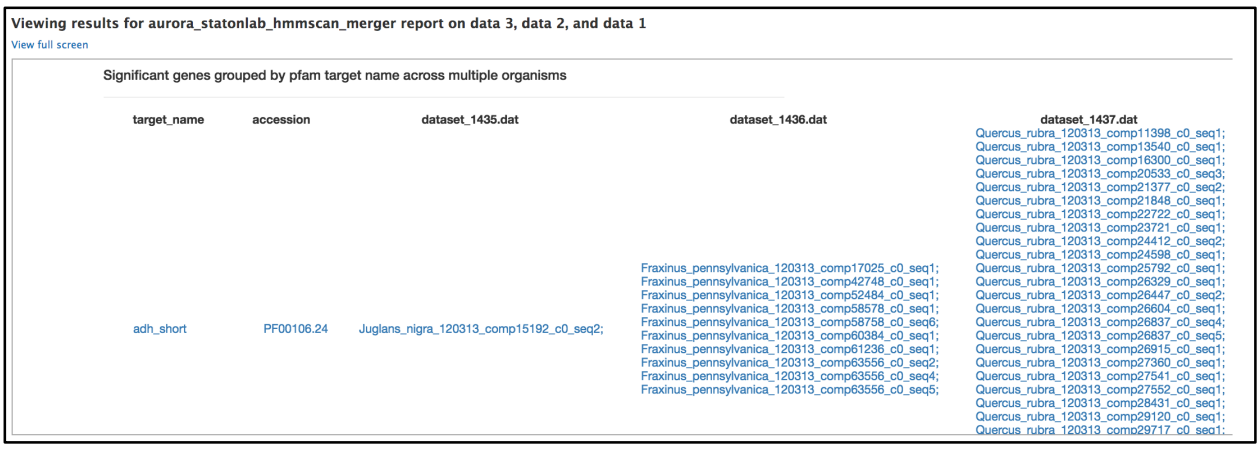


Figure 5.10 Analysis results of *hmmscan-pfam-merger* workflow.

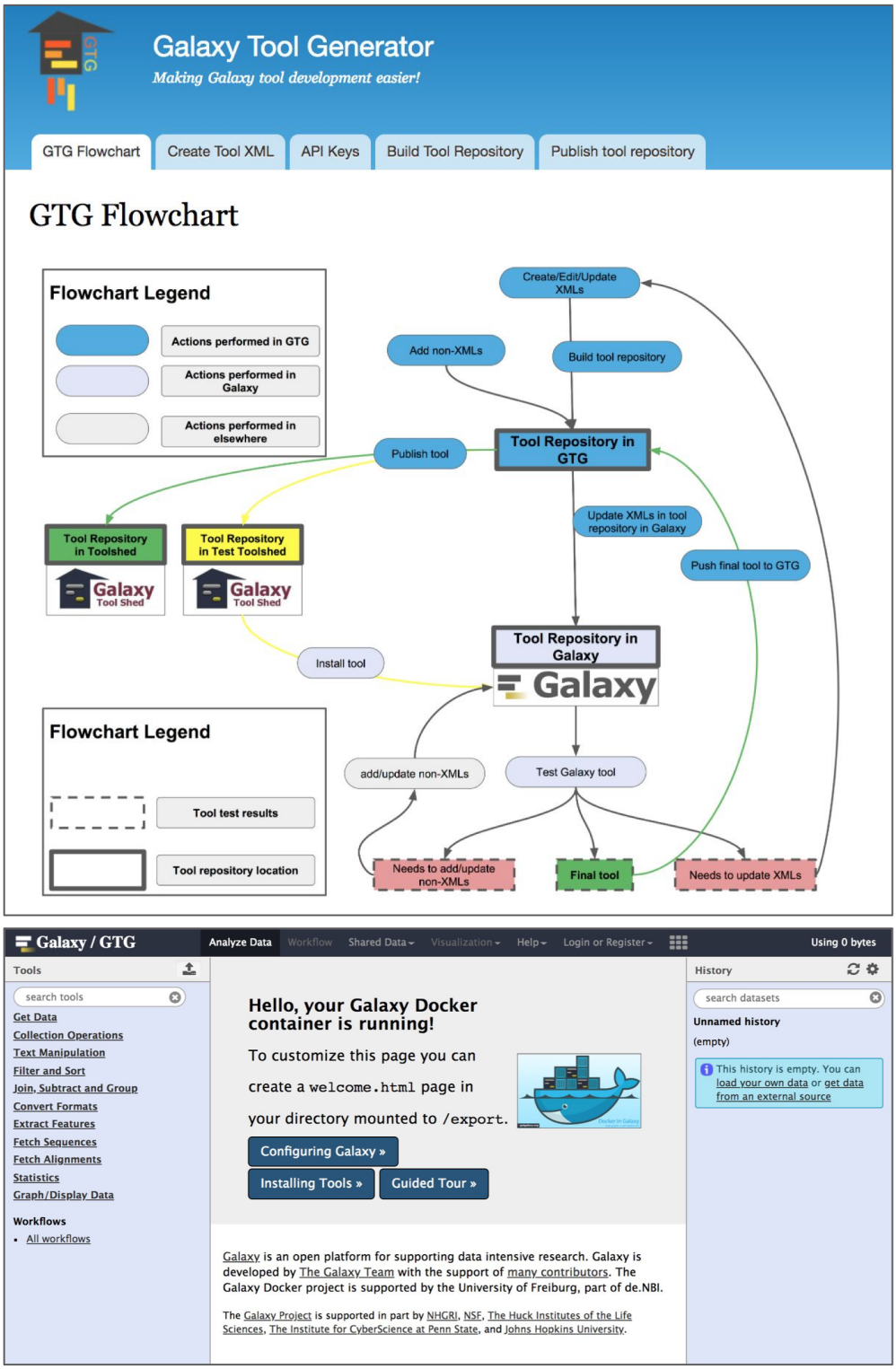


Figure 5.11 Interfaces of GTG and connected Galaxy instance.

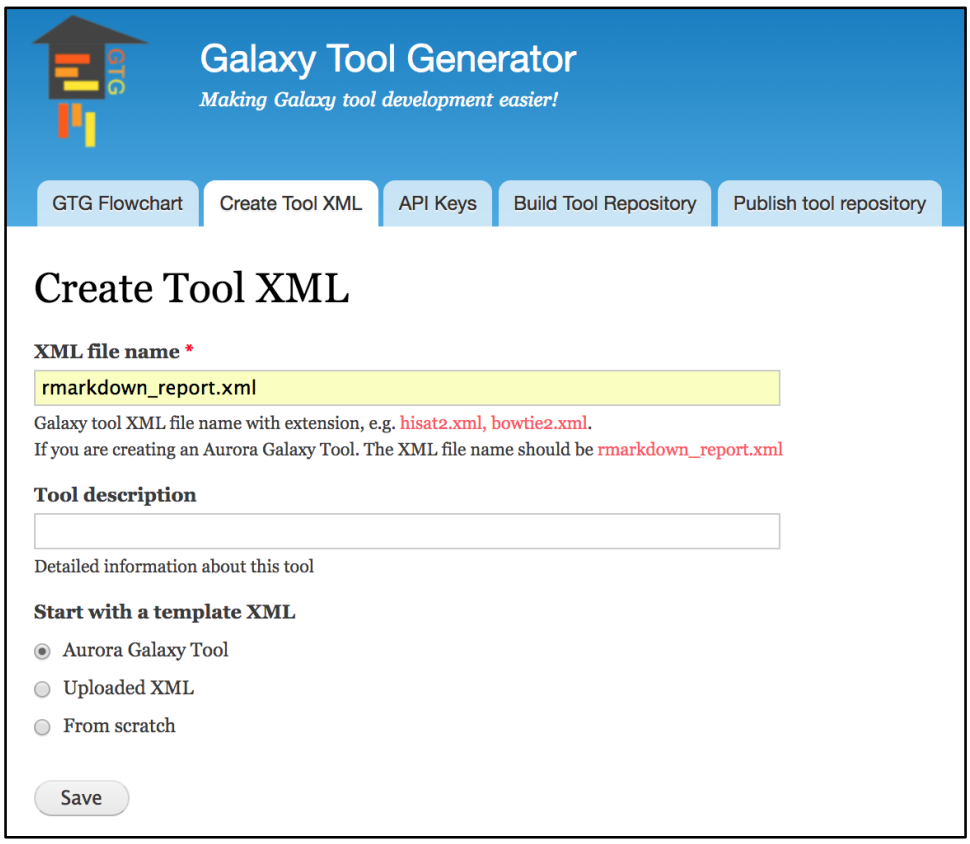


Figure 5.12 Interface of Create Galaxy Tool XML.

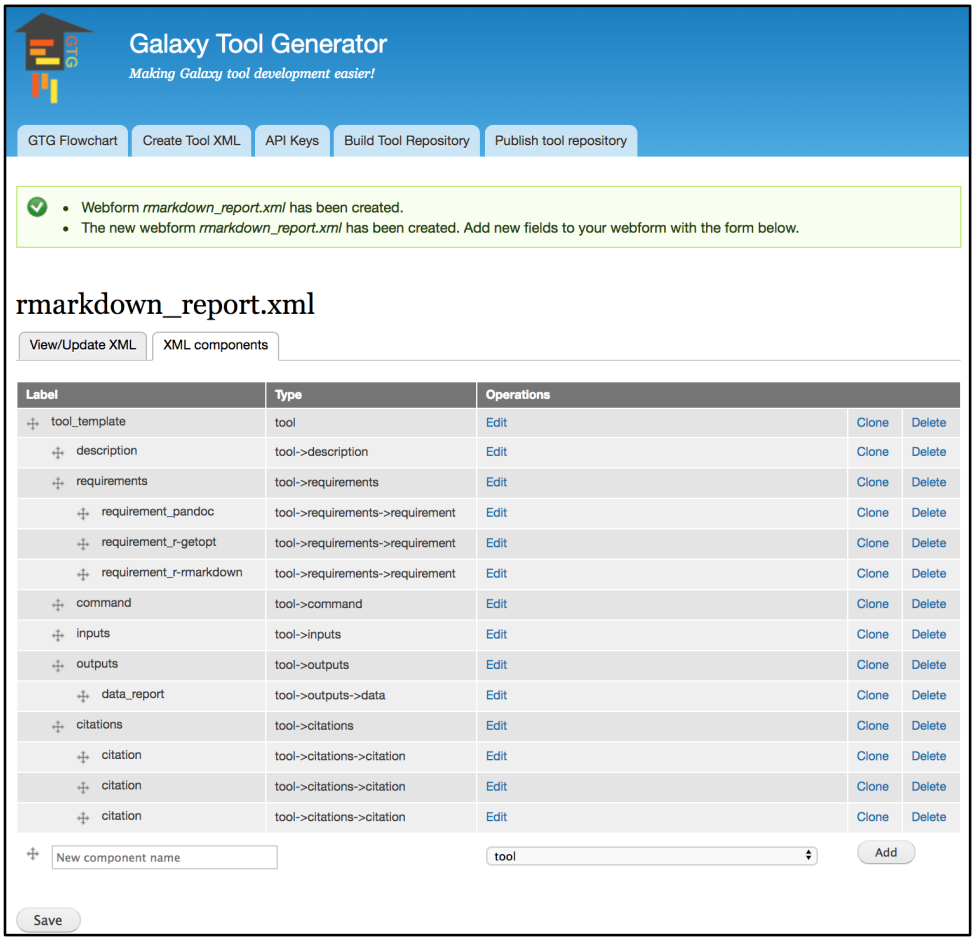


Figure 5.13 Web form components interface.

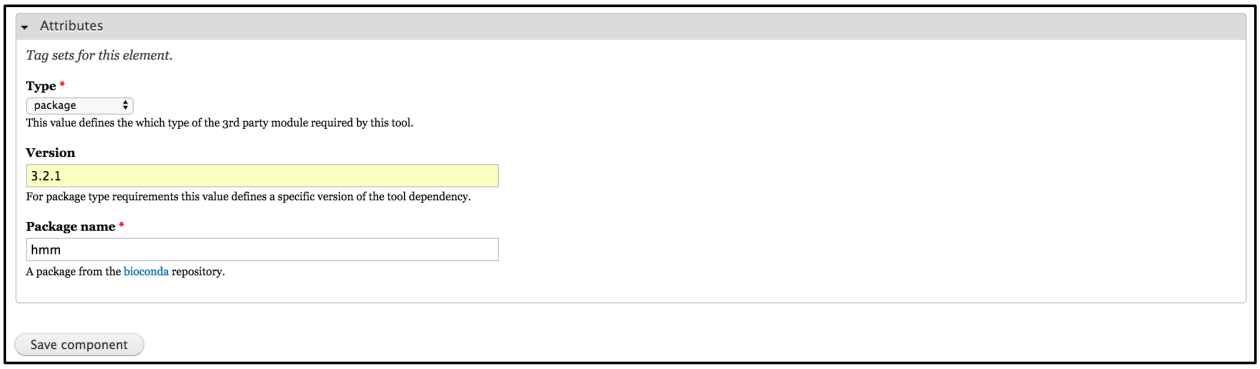


Figure 5.14 Adding a requirement component for the tool dependency HMMER.

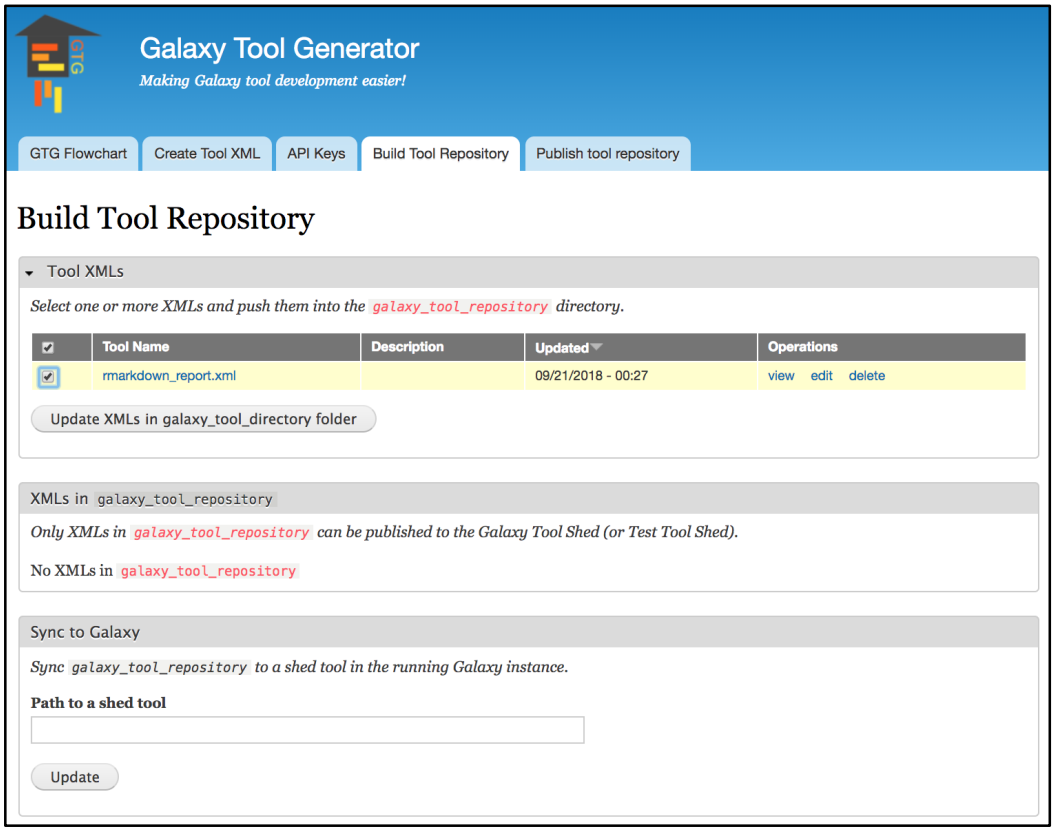


Figure 5.15 Interface to add XML file to the galaxy_tool_repository folder.

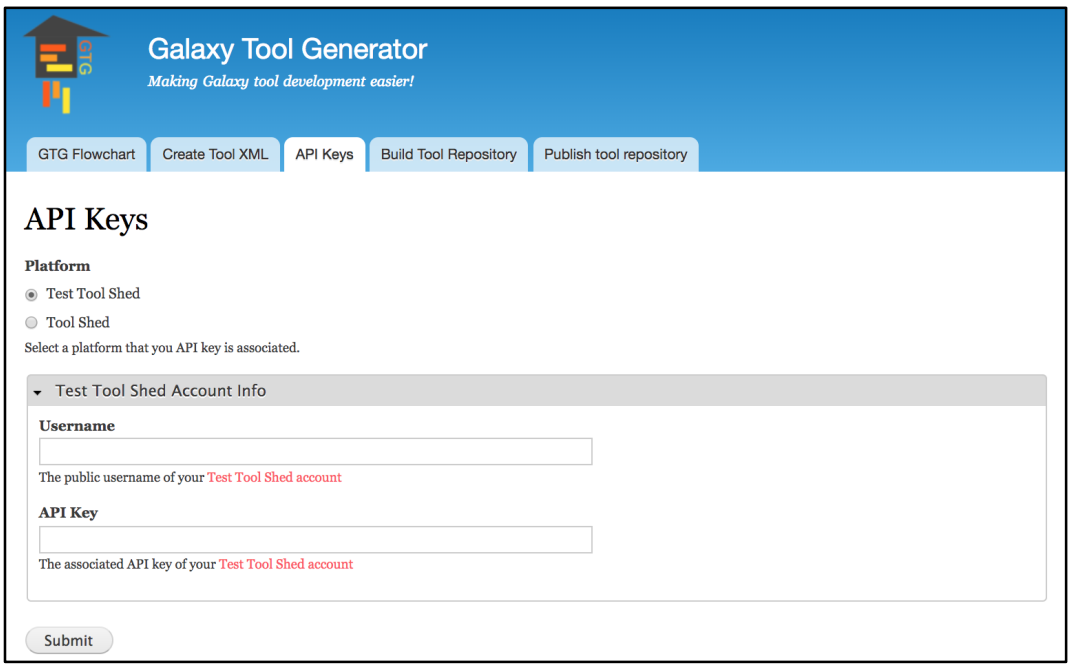


Figure 5.16 Interface to add ToolShed/Test ToolShed API keys for publishing Galaxy tools.

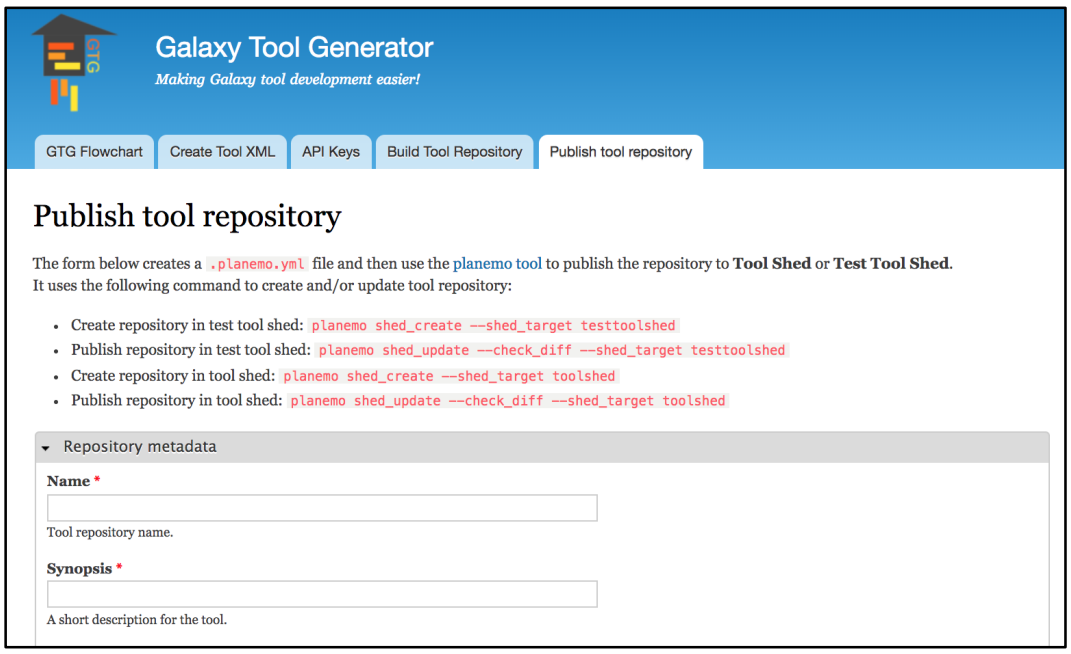


Figure 5.17 Interface for publishing Galaxy tools.

CHAPTER VI. CONCLUSIONS

Being brought into the era of big biological data by continuously rapid development in high throughput technologies, biological researchers are encouraged to make sense out of their study systems with data at larger scales, higher dimensions, and greater diversity than ever before. While there is a growing emphasis on developing computational skills including programming for scientists, there still exists a gap between the tool boxes of average biologists and what is needed for them to tackle scientific questions based on big data. The web-based computational tools presented in this study fill this gap from several aspects.

In Chapter II, the Tripal Elasticsearch module bridges the powerful text search engine Elasticsearch to Tripal-based genomics databases which enables database users to efficiently locate and gather genomic and transcriptomic data to analyze. The Tripal Analysis Expression module stores and communicates gene expression experiment data and enable database users to interactively visualize expression levels from single or a set of features across multiple treatments. A combined use of these two module could further facilitate transcriptome analyses for researchers, as demonstrated in the case study in Chapter II.

In Chapter III, the *Aurora Galaxy Tools* standardize data analysis reports across Galaxy tools with HTML documents. This new R Markdown based framework allows Galaxy tools to be wrapped to display analysis logic in the analysis report, leading to more transparent data analysis processes to end users. The clickable file tree generated by each *Aurora Galaxy Tool* gives users full access to all tool outputs with an intuitive navigation tree. Development templates are available

to help Galaxy tool developers transition to the *Aurora Galaxy Tool* framework.

In Chapter IV, the GTG web application was designed not for database users but for Galaxy tool developers. With GTG, developers can build web-based tools more easily and efficiently, and therefore benefit more database users in their data analysis and visualization.

This study was concluded with a comprehensive use case which includes the application of all tools and approaches developed. The use case successfully demonstrated the use of the tools on a practical biological problem, the response of hardwood trees to O_3 exposure, and provided some implications for future work on the tools presented here. For example, data indexed by the Tripal Elasticsearch module is stored in an Elasticsearch cluster, and therefore can be extracted with existing R or Python Elasticsearch interfaces. Search results returned by the Tripal Elasticsearch module are presented on web pages and are not as easy to manipulate as common data files like csv or txt. To avoid this situation, however, I could develop Galaxy tools that utilize Elasticsearch interfaces to directly extract data from Tripal databases to Galaxy for data analysis. Another promising direction is to develop *Aurora Galaxy Tools* that are highly customized for a specific Tripal database. Because of the GTG web application, developing a Galaxy tool becomes much faster and easier, removing barriers to developing and maintaining more tools. As shown in the use case presented in Chapter V, the final report generated by the workflow provided URL links from the identified genes back to their feature pages in HGD website. This report won't work if the tools and workflows are used in other Tripal databases, but it integrates with the HGD database seamlessly. I could tailor the tool development even further so that it accepts URLs and unprocessed data types from HGD and returns results that can be integrated back to the database. These features would largely extend the functionality of HGD.

REFERENCES

Afgan,E. *et al*. (2016) The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. *Nucleic Acids Res*., **44**, W3–W10.

Allaire,J. *et al*. (2016) Rmarkdown: dynamic documents for R, 2015. *R package version 0. 9*, **6**.

Allaire,J.J. *et al*. (2015) rmarkdown: Dynamic Documents for R. *R package version 0. 5*.

Allen,C.D. *et al*. (2010) A global overview of drought and heat-induced tree mortality reveals emerging climate change risks for forests. *For. Ecol. Manage*., **259**, 660–684.

Allen,J.A. *et al*. (2001) A guide to bottomland hardwood restoration GEOLOGICAL SURVEY RESTON VA.

Altschul,S.F. *et al*. (1990) Basic local alignment search tool. *J. Mol. Biol*., **215**, 403–410.

Altschul,S.F. *et al*. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*., **25**, 3389–3402.

Anagnostakis,S.L. (2012) Chestnut Breeding in the United States for Disease and Insect Resistance. *Plant Dis*., **96**, 1392–1403.

Anders,S. *et al*. (2015) HTSeq--a Python framework to work with high-throughput sequencing data. *Bioinformatics*, **31**, 166–169.

Anders,S. and Huber,W. (2010) Differential expression analysis for sequence count data Genome Biology, 11. *R106*.

Andrews,S. and Others (2010) FastQC: a quality control tool for high throughput sequence data.

Arkin,A.P. *et al*. (2016) The DOE Systems Biology Knowledgebase (KBase). *bioRxiv*, 096354.

Avraham,S. *et al*. (2008) The Plant Ontology Database: a community resource for plant structure and developmental stages controlled vocabulary and annotations. *Nucleic Acids Res*., **36**, D449–54.

Bagard,M. *et al*. (2008) Ozone-induced changes in photosynthesis and photorespiration of hybrid

poplar in relation to the developmental stage of the leaves. *Physiol. Plant.*, **134**, 559–574.

Banon,S. (2011) Elasticsearch: An open source, distributed, RESTful search engine.

Barrett,T. *et al.* (2012) BioProject and BioSample databases at NCBI: facilitating capture and organization of metadata. *Nucleic Acids Res.*, **40**, D57–63.

Bateman,A. *et al.* (2004) The Pfam protein families database. *Nucleic Acids Res.*, **32**, D138–41.

Bateman,I.J. and Lovett,A.A. (2000) Estimating and valuing the carbon sequestered in softwood and hardwood trees, timber products and forest soils in Wales. *J. Environ. Manage.*, **60**, 301–323.

Batut,B. *et al.* (2017) Reference-based RNA-seq data analysis (training data).

Baumer,B. *et al.* (2014) R Markdown: Integrating A Reproducible Analysis Tool into Introductory Statistics. *arXiv [stat.OT]*.

Bizer,C. *et al.* (2012) The Meaningful Use of Big Data: Four Perspectives -- Four Challenges. *SIGMOD Rec.*, **40**, 56–60.

Blankenberg,D. *et al.* (2014a) Dissemination of scientific software with Galaxy ToolShed. *Genome Biol.*, **15**, 403.

Blankenberg,D. *et al.* (2014b) Dissemination of scientific software with Galaxy ToolShed. *Genome Biol.*, **15**, 403.

Blankenberg,D. *et al.* (2014c) Dissemination of scientific software with Galaxy ToolShed. *Genome Biol.*, **15**, 403.

Blattmann,P. *et al.* (2016) SWATH2stats: An R/Bioconductor Package to Process and Convert Quantitative SWATH-MS Proteomics Data for Downstream Analysis Tools. *PLoS One*, **11**, e0153160.

Bozhanov,I. jsTree: a jQuery plugin for providing interactive file trees.

Brendley,B.W. and Pell,E.J. (1998) Ozone-induced changes in biosynthesis of Rubisco and associated compensation to stress in foliage of hybrid poplar. *Tree Physiol.*, **18**, 81–90.

Brock,D.C. and Moore,G.E. (2006) Understanding Moore's Law: Four Decades of Innovation Chemical Heritage Foundation.

Bullard,J.H. *et al.* (2010) Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments. *BMC Bioinformatics*, **11**, 94.

Ćalić,I. *et al.* (2017) Genome-wide association study identifies a major gene for beech bark disease resistance in American beech (Fagus grandifolia Ehrh.). *BMC Genomics*, **18**, 547.

Callahan,B. *et al.* (2016) REPRODUCIBLE RESEARCH WORKFLOW IN R FOR THE ANALYSIS OF PERSONALIZED HUMAN MICROBIOME DATA. *Pac. Symp. Biocomput.*, **21**, 183–194.

Cao,L. (2016) Data science: Nature and pitfalls. *IEEE Intell. Syst.*, 66–75.

Caregnato,F.F. *et al.* (2015) The impact of ozone pollution on plant defence metabolism: detrimental effects on yield and quality of agricultural crops. In, Chakraborty,U. and Chakraborty,B. (eds), *Abiotic stresses in crop plants*. CABI, Wallingford, pp. 112–135.

Ceulemans,R. and Mousseau,M. (1994) Tansley Review No. 71 Effects of elevated atmospheric CO2on woody plants. *New Phytol.*, **127**, 425–446.

Chang,W. *et al.* (2015) Shiny: web application framework for R. *R package version 0. 11*, **1**.

Chilton,J. *et al.* Galaxy Planemo 0.35. 0 [Data set]. Zenodo. 2016. *Data Source*, **1**, 2.

Collard,B.C.Y. and Mackill,D.J. (2008) Marker-assisted selection: an approach for precision plant breeding in the twenty-first century. *Philos. Trans. R. Soc. Lond. B Biol. Sci.*, **363**, 557–572.

Collins,F.S. *et al.* (2003) The Human Genome Project: lessons from large-scale biology. *Science*,

300, 286–290.

Conesa,A. *et al*. (2016) A survey of best practices for RNA-seq data analysis. *Genome Biol*., **17**, 13.

Crossa,J. *et al*. (2010) Prediction of genetic values of quantitative traits in plant breeding using pedigree and molecular markers. *Genetics*, **186**, 713–724.

Czechowski,T. *et al*. (2005) Genome-wide identification and testing of superior reference genes for transcript normalization in Arabidopsis. *Plant Physiol*., **139**, 5–17.

Dale,V.H. *et al*. (2016) Climate Change and the Future of Natural Disturbances in the Central Hardwood Region. In, Greenberg,C.H. and Collins,B.S. (eds), *Natural Disturbances and Historic Range of Variation: Type, Frequency, Severity, and Post-disturbance Structure in Central Hardwood Forests USA*. Springer International Publishing, Cham, pp. 355–369.

DeGraaf,R.M. (2002) Trees, Shrubs, and Vines for Attracting Birds UPNE.

De Maio,A. (1999) Heat shock proteins: facts, thoughts, and dreams. *Shock*, **11**, 1–12.

Diehn,M. *et al*. (2003) SOURCE: a unified genomic resource of functional annotations, ontologies, and gene expression data. *Nucleic Acids Res*., **31**, 219–223.

Di Tommaso,P. *et al*. (2015) The impact of Docker containers on the performance of genomic pipelines. *PeerJ*, **3**, e1273.

Divya,M.S. and Goyal,S.K. (2013) ElasticSearch: An advanced and quick search technique to handle voluminous data. *Compusoft*, **2**, 171.

Dobin,A. *et al*. (2013) STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, **29**, 15–21.

Eddy,S. (2010) HMMER3: a new generation of sequence homology search software. *URL: http://hmmer. janelia. Org*.

Eddy,S. (1992) HMMER user's guide. *Department of Genetics, Washington University School of*

Medicine, **2**.

Ewels,P. *et al*. (2016) MultiQC: summarize analysis results for multiple tools and samples in a single report. *Bioinformatics*, **32**, 3047–3048.

Fann,N. *et al*. (2015) The geographic distribution and economic value of climate change-related ozone health impacts in the United States in 2030. *J. Air Waste Manag. Assoc.*, **65**, 570–580.

Feng,Y. *et al*. (2015) Nanopore-based fourth-generation DNA sequencing technology. *Genomics Proteomics Bioinformatics*, **13**, 4–16.

Ficklin,S.P. *et al*. (2011) Tripal: a construction toolkit for online genome databases. *Database* , **2011**, bar044.

FlyBase Consortium (2003) The FlyBase database of the Drosophila genome projects and community literature. *Nucleic Acids Res.*, **31**, 172–175.

Fonseca,N.A. *et al*. (2014) RNA-Seq gene profiling--a systematic empirical comparison. *PLoS One*, **9**, e107026.

Galaxy Project Stats *Galaxy Project Stats*.

Gandhi,K.J.K. and Herms,D.A. (2010) Direct and indirect effects of alien insect herbivores on ecological processes and interactions in forests of eastern North America. *Biol. Invasions*, **12**, 389–405.

Gandrud,C. (2016) Reproducible research with R and R studio Chapman and Hall/CRC.

Giardine,B. *et al*. (2005) Galaxy: a platform for interactive large-scale genome analysis. *Genome Res.*, **15**, 1451–1455.

Goecks,J. *et al*. (2010) Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol.*, **11**,

R86.

Goff,S.A. *et al*. (2011) The iPlant Collaborative: Cyberinfrastructure for Plant Biology. *Front. Plant Sci*., **2**, 34.

Gough,C.M. (2012) Terrestrial primary production: fuel for life. *Nature Education Knowledge*, **3**, 1–7.

Gruber,J. (2004) Markdown language.

Grüning,B. *et al*. (2018) Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nat. Methods*, **15**, 475–476.

Grüning,B. (2018) Galaxy Docker Image.

Grüning,B.A. *et al*. (2017) Jupyter and Galaxy: Easing entry barriers into complex data analyses for biomedical researchers. *PLoS Comput. Biol*., **13**, e1005425.

Hanson,P.J. and Weltzin,J.F. (2000) Drought disturbance from climate change: response of United States forests. *Sci. Total Environ*., **262**, 205–220.

Harmon,M.E. *et al*. (1986) Ecology of Coarse Woody Debris in Temperate Ecosystems. In, MacFadyen,A. and Ford,E.D. (eds), *Advances in Ecological Research*. Academic Press, pp. 133–302.

Harper,A.L. *et al*. (2016) Molecular markers for tolerance of European ash (Fraxinus excelsior) to dieback disease identified using Associative Transcriptomics. *Sci. Rep*., **6**, 19335.

Highsoft,A.S. (2014) Highcharts-Interactive JavaScript charts for your web projects. 2014.

Hiltemann,S. *et al*. (2014) iReport: a generalised Galaxy solution for integrated experimental reporting. *Gigascience*, **3**, 19.

htmlwidgets for R

Jacobs,D.F. (2006) Importance of hardwood tree plantings. *UDSA Forest Service, Northern*

Research Station. *FNR-219*.

Jiang,H. *et al*. (2014) Skewer: a fast and accurate adapter trimmer for next-generation sequencing paired-end reads. *BMC Bioinformatics*, **15**, 182.

Jolivet,Y. *et al*. (2016) Deciphering the ozone-induced changes in cellular processes: a prerequisite for ozone risk assessment at the tree and forest levels. *Ann*. *For*. *Sci*., **73**, 923–943.

Jones,P. *et al*. (2014) InterProScan 5: genome-scale protein function classification. *Bioinformatics*, **30**, 1236–1240.

Kanehisa,M. and Goto,S. (2000) KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res*., **28**, 27–30.

Katal,A. *et al*. (2013) Big data: Issues, challenges, tools and Good practices. In, *2013 Sixth International Conference on Contemporary Computing (IC3)*., pp. 404–409.

Khan,N. *et al*. (2018) The 10 Vs, Issues and Challenges of Big Data. In, *Proceedings of the 2018 International Conference on Big Data and Education*. ACM, pp. 52–56.

Kharchenko,P.V. *et al*. (2014) Bayesian approach to single-cell differential expression analysis. *Nat*. *Methods*, **11**, 740–742.

Kisaoglu,A. *et al*. (2013) Tissue damage and oxidant/antioxidant balance. *Eurasian J Med*, **45**, 47–49.

Kohen,R. and Nyska,A. (2002) Oxidation of biological systems: oxidative stress phenomena, antioxidants, redox reactions, and methods for their quantification. *Toxicol*. *Pathol*., **30**, 620–650.

Landry,L.G. and Pell,E.J. (1993) Modification of Rubisco and Altered Proteolytic Activity in O3-Stressed Hybrid Poplar (Populus maximowizii x trichocarpa). *Plant Physiol*., **101**,

1355–1362.

Lane,T. *et al*. (2016) The green ash transcriptome and identification of genes responding to abiotic and biotic stresses. *BMC Genomics*, **17**, 702.

Langseth et al. (2015) Community for Data Integration 2015 annual report. *US Geological Survey*, 2015

Leisner,C.P. *et al*. (2014) Distinct transcriptional profiles of ozone stress in soybean (Glycine max) flowers and pods. *BMC Plant Biol*., **14**, 335.

Leitao,L. *et al*. (2007) Changes in PEP carboxylase, rubisco and rubisco activase mRNA levels from maize (Zea mays) exposed to a chronic ozone stress. *Biol. Res*., **40**, 137–153.

Leitao,L. *et al*. (2007) The impact of ozone on juvenile maize (Zea mays L.) plant photosynthesis: effects on vegetative biomass, pigmentation, and carboxylases (PEPc and Rubisco). *Plant Biol*. , **9**, 478–488.

Leitao,L. *et al*. (2003) Time-course of Rubisco oxidation in beans (Phaseolus vulgaris L.) subjected to a long-term ozone stress. *Plant Sci*., **165**, 613–620.

Leopold,D.J. *et al*. (1998) Trees of the central hardwood forests of North America. An identification and cultivation guide Timber Press.

Levin-Reisman,I. *et al*. (2017) Antibiotic tolerance facilitates the evolution of resistance. *Science*, **355**, 826–830.

Li,H. *et al*. (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.

Li,P. *et al*. (2015) Comparing the normalization methods for the differential analysis of Illumina high-throughput RNA-Seq data. *BMC Bioinformatics*, **16**, 347.

Li,P.-E. *et al*. (2017) Enabling the democratization of the genomics revolution with a fully

integrated web-based bioinformatics platform. *Nucleic Acids Res*., **45**, 67–80.

Lister,R. *et al*. (2009) Next is now: new technologies for sequencing of genomes, transcriptomes, and beyond. *Curr. Opin. Plant Biol*., **12**, 107–118.

Li,Y. *et al*. (2011) FusionHunter: identifying fusion transcripts in cancer using paired-end RNA-seq. *Bioinformatics*, **27**, 1708–1710.

Li,Y. and Chen,L. (2014) Big biological data: challenges and opportunities. *Genomics Proteomics Bioinformatics*, **12**, 187–189.

Loo,J.A. (2009) Ecological impacts of non-indigenous invasive fungi as forest pathogens. In, Langor,D.W. and Sweeney,J. (eds), *Ecological Impacts of Non-Native Invertebrates and Fungi on Terrestrial Ecosystems*. Springer Netherlands, Dordrecht, pp. 81–96.

Love,M. *et al*. (2014) Differential analysis of count data--the DESeq2 package. *Genome Biol*., **15**, 10–1186.

Love,M. *et al*. (2013) Differential analysis of RNA-Seq data at the gene level using the DESeq2 package.

Love,M.I. *et al*. (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biol*., **15**, 550.

Lovett,G.M. *et al*. (2006) Forest Ecosystem Responses to Exotic Pests and Pathogens in Eastern North America. *Bioscience*, **56**, 395–405.

MacGowan,B.J. (2003) Designing hardwood tree plantings for wildlife Hardwood Tree Improvement and Regeneration Center.

Marguerat,S. and Bähler,J. (2009) RNA-seq: from technology to biology. *Cell. Mol. Life Sci*., **67**, 569–579.

Marques,F.Z. *et al*. (2017) High-Fiber Diet and Acetate Supplementation Change the Gut

Microbiota and Prevent the Development of Hypertension and Heart Failure in Hypertensive Mice. *Circulation*, **135**, 964–977.

Martin,L.B.B. *et al*. (2013) Catalyzing plant science research with RNA-seq. *Front. Plant Sci.*, **4**, 66.

Marx,V. (2013a) Biology: The big challenges of big data. *Nature*, **498**, 255–260.

Marx,V. (2013b) Biology: The big challenges of big data. *Nature*, **498**, 255–260.

Matyssek,R. *et al*. (2010) Advances in understanding ozone impact on forest trees: messages from novel phytotron and free-air fumigation studies. *Environ. Pollut.*, **158**, 1990–2006.

May,M. (2014) Big biological impacts from big data. *Science*.

McCullough,D.G. (2013) Will we kiss our ash goodbye. *American Forests 188*, 16–22.

McKenney,D.W. *et al*. (2007) Potential Impacts of Climate Change on the Distribution of North American Trees. *Bioscience*, **57**, 939–948.

Meeks,E. (2017) D3.js in Action: Data Visualization With Javascript Manning Publications Company.

Moser,W.K. *et al*. (2009) Impacts of Nonnative Invasive Species on US Forests and Recommendations for Policy and Management. *J. For.*, **107**, 320–327.

Mungall,C.J. *et al*. (2007) A Chado case study: an ontology-based modular schema for representing genome-associated biological information. *Bioinformatics*, **23**, i337–46.

Neale,D.B. and Kremer,A. (2011) Forest tree genomics: growing resources and applications. *Nat. Rev. Genet.*, **12**, 111–122.

O'Driscoll,A. *et al*. (2013) 'Big data', Hadoop and cloud computing in genomics. *J. Biomed. Inform.*, **46**, 774–781.

Ollinger,S.V. *et al*. (1997) Simulating Ozone Effects on Forest Productivity: Interactions among

Leaf-, Canopy-, and Stand-Level Processes. *Ecol. Appl.*, **7**, 1237.

Omar,S. *et al*. (2000) Antimicrobial activity of extracts of eastern North American hardwood trees and relation to traditional medicine. *J. Ethnopharmacol.*, **73**, 161–170.

Ortuño,F. and Rojas,I. (2016) Bioinformatics and Biomedical Engineering: 4th International Conference, IWBBIO 2016, Granada, Spain, April 20-22, 2016, Proceedings Springer.

Oshlack,A. *et al*. (2010) From RNA-seq reads to differential expression results. *Genome Biol.*, **11**, 220.

Ozone effects on plants (USDA, ARS) *Ozone effects on plants (USDA, ARS)*.

Pääkkönen,E. *et al*. (1996) Growth responses and related biochemical and ultrastructural changes of the photosynthetic apparatus in birch (Betula pendula) saplings exposed to low concentrations of ozone. *Tree Physiol.*, **16**, 597–605.

Panayotou,T. and Ashton,P. (1992) Not by Timber Alone: Economics And Ecology For Sustaining Tropical Forests Island Press.

Pandit,S. *et al*. (2013) Genome-wide analysis reveals SR protein cooperation and competition in regulated splicing. *Mol. Cell*, **50**, 223–235.

Pastor,J. and Bockheim,J.G. (1984) Distribution and Cycling of Nutrients in an Aspen-Mixed-Hardwood-Spodosol Ecosystem in Northern Wisconsin. *Ecology*, **65**, 339–353.

Pelloux,J. *et al*. (2001) Changes in Rubisco and Rubisco activase gene expression and polypeptide content inPinus halepensisM. subjected to ozone and drought. *Plant Cell Environ.*, **24**, 123–131.

Pijut,P.M. *et al*. (2011) Biotechnological efforts for preserving and enhancing temperate hardwood tree biodiversity, health, and productivity. *In Vitro Cellular & Developmental Biology - Plant*, **47**, 123–147.

Pilch,W. *et al*. (2014) Disturbances in pro-oxidant-antioxidant balance after passive body overheating and after exercise in elevated ambient temperatures in athletes and untrained men. *PLoS One*, **9**, e85320.

Pimentel,D. *et al*. (2000) Environmental and Economic Costs of Nonindigenous Species in the United States. *Bioscience*, **50**, 53–65.

Pimentel,D. *et al*. (2005) Environmental, Energetic, and Economic Comparisons of Organic and Conventional Farming Systems. *Bioscience*, **55**, 573–582.

Pirinen,M. *et al*. (2015) Assessing allele-specific expression across multiple tissues from RNA-seq read data. *Bioinformatics*, **31**, 2497–2504.

Pollack,A. (2011) DNA sequencing caught in deluge of data. *NY Times* , **30**.

Rabbani,M.A. *et al*. (2003) Monitoring expression profiles of rice genes under cold, drought, and high-salinity stresses and abscisic acid application using cDNA microarray and RNA gel-blot analyses. *Plant Physiol*., **133**, 1755–1767.

Rao,M.V. *et al*. (1995) DIFFERENTIAL RESPONSE OF PHOTOSYNTHETIC PIGMENTS, RUBISCO ACTIVITY and RUBISCO PROTEIN OF Arabidopsis thaliana EXPOSED TO UVB and OZONE. *Photochem. Photobiol*., **62**, 727–735.

Rees,H.A. *et al*. (2017) Improving the DNA specificity and applicability of base editing through protein engineering and protein delivery. *Nat. Commun*., **8**, 15790.

Reuter,J.A. *et al*. (2015) High-throughput sequencing technologies. *Mol. Cell*, **58**, 586–597.

Ritossa,F. (1962) A new puffing pattern induced by temperature shock and DNP in drosophila. *Experientia*, **18**, 571–573.

Rizzo,M. *et al*. (2007) Identification of differentially expressed genes induced by ozone stress in sensitive and tolerant poplar hybrids. *J. Plant Physiol*., **164**, 945–949.

Rosinski-Chupin,I. *et al*. (2014) Riboswitch discovery by combining RNA-seq and genome-wide identification of transcriptional start sites. *Methods Enzymol*., **549**, 3–27.

Sanderson,L.-A. *et al*. (2013) Tripal v1.1: a standards-based toolkit for construction of online genetic and genomic databases. *Database* , **2013**, bat075.

Sanders,S. *et al*. (2017) Third Generation Sequencing.

Schmidtling,R.C. *et al*. (2004) The role of genetics and tree improvement in southern forest productivity. *In: Gen. Tech. Rep. SRS--75. Asheville, NC: US Department of Agriculture, Forest Service, Southern Research Station. Chapter 10. p. 97-108*.

Schulz,W.L. *et al*. (2016) Use of application containers and workflows for genomic data analysis. *J. Pathol. Inform*., **7**, 53.

Seth,M.K. (2003) Trees and their economic importance. *Bot. Rev*., **69**, 321–376.

Shen,W. *et al*. (2016) SeqKit: A Cross-Platform and Ultrafast Toolkit for FASTA/Q File Manipulation. *PLoS One*, **11**, e0163962.

Sievert,C. *et al*. (2016) plotly: Create Interactive Web Graphics via 'plotly. js'. *R package version*, **3**.

Sievert,C. *et al*. (2016) plotly: Create interactive web graphics via Plotly's JavaScript graphing library [Software].

Skinner,M.E. *et al*. (2009) JBrowse: a next-generation genome browser. *Genome Res*., **19**, 1630–1638.

Skousen,J. *et al*. (2009) Hardwood tree survival in heavy ground cover on reclaimed land in West Virginia: mowing and ripping effects. *J. Environ. Qual*., **38**, 1400–1409.

Soneson,C. and Delorenzi,M. (2013) A comparison of methods for differential expression analysis of RNA-seq data. *BMC Bioinformatics*.

Sork,V.L. *et al*. (2013) Putting the landscape into the genomics of trees: approaches for understanding local adaptation and population responses to changing climate. *Tree Genet. Genomes*, **9**, 901–911.

Speen,A. *et al*. (2018) Ozone-Derived Oxysterols Form Adducts with NLRP2, Potentially Contributing to Inflammation in Airway Epithelial Cells. In, *B24. AN UPDATE ON INDOOR AND OUTDOOR AIR POLLUTION*, American Thoracic Society International Conference Abstracts. American Thoracic Society, pp. A2825–A2825.

Stejskal,J. *et al*. (2018) Effect of genomic prediction on response to selection in forest tree breeding. *Tree Genet. Genomes*, **14**, 74.

Stephens,Z.D. *et al*. (2015) Big Data: Astronomical or Genomical? *PLoS Biol.*, **13**, e1002195.

Swindell,W.R. *et al*. (2007) Transcriptional profiling of Arabidopsis heat shock proteins and transcription factors reveals extensive overlap between heat and non-heat stress response pathways. *BMC Genomics*, **8**, 125.

Sykes,M.T. and Prentice,I.C. (1996) Climate change, tree species distributions and forest dynamics: A case study in the mixed conifer/northern hardwoods zone of northern Europe. *Clim. Change*, **34**, 161–177.

Team,R. (2015) RStudio: integrated development for R. *RStudio, Inc. , Boston, MA URL http://www. rstudio. com*.

Tiwari,S. and Agrawal,M. (2018) Effect of Ozone on Physiological and Biochemical Processes of Plants. In, Tiwari,S. and Agrawal,M. (eds), *Tropospheric Ozone and its Impacts on Crop Plants: A Threat to Future Global Food Security*. Springer International Publishing, Cham, pp. 65–113.

Todd,E.V. *et al*. (2016) The power and promise of RNA-seq in ecology and evolution. *Mol.*

Ecol., **25**, 1224–1241.

from Trevor L Davis,A.D.C. and Zenka.,R. (2018) getopt: C-Like 'getopt' Behavior.

Tripal 3 (2018) *Tripal 3*.

Vaidyanathan,R. *et al*. (2014) htmlwidgets: HTML Widgets for R. *R package version 0. 3*, **2**.

VanDyk,J.K. and Westgate,M. (2007) Pro Drupal Development.

Vijay,N. *et al*. (2013) Challenges and strategies in transcriptome assembly and differential gene expression quantification. A comprehensive in silico assessment of RNA-seq experiments. *Mol. Ecol*., **22**, 620–634.

Wang,B. *et al*. (2016) Forests and ozone: productivity, carbon storage, and feedbacks. *Sci. Rep*., **6**, 22133.

Wang,Z. *et al*. (2009) RNA-Seq: a revolutionary tool for transcriptomics. *Nat. Rev. Genet*., **10**, 57–63.

Wegrzyn,J.L. *et al*. (2012) Uniform standards for genome databases in forest and fruit trees. *Tree Genet. Genomes*, **8**, 549–557.

Wetterstrand,K.A. (2016) DNA sequencing costs: data from the NHGRI Genome sequencing program (GSP). 2013. *URL http://www. genome. gov/sequencingcosts*.

Wheeler,D.L. *et al*. (2001) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*., **29**, 11–16.

Wickham,H. (2016) ggplot2: Elegant Graphics for Data Analysis Springer.

Wickham,H. (2017) Tidyverse: Easily install and load'tidyverse'packages. *R package version*, **1**.

Wilhelm,B.T. and Josette-Renée,L. (2009) RNA-Seq—quantitative measurement of expression through massively parallel RNA-sequencing. *Methods*, **48**, 249–257.

Wilkinson,M.D. *et al*. (2016) The FAIR Guiding Principles for scientific data management and

stewardship. *Sci Data*, **3**, 160018.

Wingfield,M.J. *et al*. (2010) Novel associations between pathogens, insects and tree species threaten world forests. *N. Z. J. For. Sci.*, **40**, S95–S103.

Wytko,C. *et al*. (2017) blend4php: a PHP API for galaxy. *Database* , **2017**.

Xie,Y. *et al*. (2017) blogdown: Creating Websites with R Markdown CRC Press.

Xie,Y. (2015) Dynamic Documents with R and knitr, Second Edition CRC Press.

Xin,J. *et al*. (2016) High-performance web services for querying gene and variant annotation. *Genome Biol.*, **17**, 91.

Yue,X. *et al*. (2015) Impact of 2050 climate change on North American wildfire: consequences for ozone air quality. *Atmos. Chem. Phys.*, **15**, 10033–10055.

Zak,D.R. *et al*. (2007) Belowground competition and the response of developing forest communities to atmospheric CO2and O3. *Glob. Chang. Biol.*, **13**, 2230–2238.

Zhang,Y. and Wang,Y. (2016) Climate-driven ground-level ozone extreme in the fall over the Southeast United States. *Proc. Natl. Acad. Sci. U. S. A.*, **113**, 10025–10030.

Zhao,M. and Running,S.W. (2010) Drought-induced reduction in global terrestrial net primary production from 2000 through 2009. *Science*, **329**, 940–943.

VITA

Ming Chen was born in Daye, a county-level city in eastern Hubei province, China. He obtained his Bachelor's degree in aquaculture at the Yangtze University and Master's degree in Zoology at the Institute of Hydrobiology, Chinese Academy of Sciences. Before he came to Knoxville, he spent three years at Waco, Texas and realized that the PhD program at Baylor University was not a good fit for him. Then he applied to the Graduate School of Genome Science and Technology at University of Tennessee Knoxville and was accepted by the program. During his time pursuing a Doctorate degree at UTK, he also completed a Master's degree in Statistics.