POGs: An Online Resource For Plant Comparative Genomics

by

MICHAEL A TOMCAL

A THESIS

Presented to the Department of Biology
and the Robert D. Clark Honors College
in partial fulfillment of the requirements for the degree of
Bachelor of Science

June 2014

An Abstract of the Thesis of

Michael A Tomcal for the degree of Bachelor of Science
in the Department of Biology to be taken June 2014

Title: POGs: An Online Resource For Plant Comparative Genomics

Approved: 

Alice Barkan

The Putative Orthologous Groups database (POGs – http://pogs.uoregon.edu) presents scientists with an online web application for understanding gene function through the phylogenetic relationships between the genomes of *Arabidopsis*, Maize, Rice, and Poplar Tree. We discuss all matters needed for understanding the Biology and Information Technology behind the POGs database such as the Central Dogma, Phylogenetics, Plant Cell anatomy, Whole Genome Sequencing, Programming languages, Web Applications, Database servers, and more. Lastly, we discuss a real world case study on how plant geneticists use POGs everyday. Ultimately, POGs "fills a sorely needed gap in this post genomic era for the plant community."

Acknowledgements

I am hugely indebted to Alice Barkan, Nicholas Stiffler and the members of the Barkan Lab for the POGs project, its journal publication and the publication of my thesis. After working on this project for two years, I am continually humbled by the patient teaching, friendship, and support from all these extraordinary individuals. I would also like to thank John Conery for introducing me to Alice Barkan and the POGs project. Lastly, I also wish to thank the faculty of the Clark Honors College for handing me the tools necessary for this level of scholarship.

Table of Contents

List of Figures

Introduction

The World Wide Web represents a universal platform for disseminating information about all topics in the human intellectual and cultural experience. From the invention of the internet in the 1980s, and the subsequent boom of “dot coms” in the 1990s individuals built online websites talking about their passions, business, culture, and scientific research. Webmasters from this age took haphazard approaches toward mixing text and images without thinking about how users read and understand the “wall of data” from these websites.

In the 21st century, the persons creating web sites figured out new ways to display data and information using creative visualizations, scientifically validated color and design schemes, and ways to display larger amounts of data than before. The late 90s and early 2000s saw the birth of large web applications that function quite similar to programs users open from a “Start” menu or a “Dock”, allowing users to shop, bank, search for information, and do many other useful tasks. However, unlike these desktop applications, a “web application” requires only an Internet browser, which is not reliant on any particular operating system.

Scientists also have realized the usefulness a web application provides for communicating with colleagues and allowing them to search and study experimental results in a clear and organized fashion. In particular, Biologists have entered a new age with the advent of gene sequencing. Gene sequencing consists of reading the DNA “letters” from any cell-based life form and turning them into a digital form on the computer. The four letters are used repeatedly in a seemingly endless fashion for up to billions of letters for a given species’ genome (the complete set of DNA in each of its

cells). Given Biology's digital genetic revolution, a new kind of biologist, the Bioinformatician, researches from a computer rather than in the lab. The Bioinformatician attempts to make sense from the chaos of novel DNA sequences that are being produced rapidly with modern DNA sequencing technologies. The web application represents one tool in the Bioinformatician's skillset for publishing their findings and simplifying access for non-computational biologists.

In regards to the subject matter at hand, the Putative Orthologous Groups database[1] (POGs) represents a prime example of a web application that provides ready access to genetic information from four plant species, Maize (corn), Arabidopsis (a member of the mustard family), Poplar tree, and Rice. These four species were chosen because they are widely used in plant genetic research (maize, rice, Arabidopsis), they are of agronomic importance (maize rice, poplar), they have fully sequenced genomes, and they represent considerable phylogenetic diversity. This database serves as a valuable resource for plant geneticists around the globe. From here, we will discuss the basics needed in biology and computer science required to build the POGs database application, the steps taken to gather the data and program the application, and lastly a case study on how its used by scientists everyday.

Biology Background

In order to understand a molecular biologist's perspective on POGs, one must delve into the topics such as the central dogma, phylogenetics, and plant cell biology.

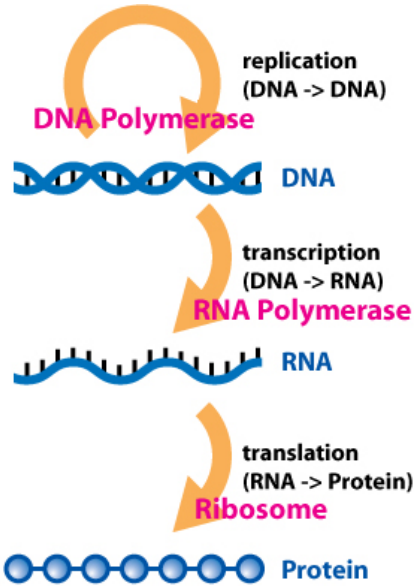
Firstly, the gene simply represents a unit of heritable genomic sequence that confers a blueprint for a functional molecule in a living cell. Heritability means the ability to pass on the gene from a parent cell to its offspring. A genome represents the set of all DNA contained in a cell from a given species. A DNA sequence refers to the sequence of letters or "nucleotides" contained in the DNA; these nucleotides are abbreviated as 'A', 'T', 'G', or 'C'. Each nucleotide letter has a "complement" that it pairs with. For example, 'A' pairs with 'T' and 'G' pairs with 'C'. The two DNA strands in the classic "double helix" pair through interactions of complementary base pairs. When cell machinery copies a gene, a sequence 'ATGC' becomes 'TACG' as it copies the complement of the original sequence.

Most genes encode one or more proteins that function as machinery in a cell. The basic anatomy of the gene includes a coding region that ultimately encodes a protein, and a flanking "regulatory" region that determines when the gene will be expressed and how much gene product will be produced.

One must consider the journey a gene takes from the double helix to the protein. Biologists refer to this as the "Central Dogma" as it describes how all-living organisms "express" genetic information and transmit this information from one generation to the next. While the details and nuances of this process vary, organisms carry out the following sequence of events: DNA transcribes to RNA and RNA translates to amino acid protein sequence. The RNA polymerase enzyme carries out the transcription of a

gene into messenger RNA (Figure 1). Messenger RNA (mRNA) carries the blueprint for the synthesis of a protein during a process known as “translation” which is carried out by structures called ribosomes (Figure 1). Proteins are polymers of “ amino acids”, and 20 different types of amino acids occur naturally in living things. The sequence of nucleotides in the RNA determines the sequence of amino acids in the protein, which in turn determines the shape and function of that protein. Proteins and their components function in all aspects of cell dynamics such as harvesting energy, replicating the cell, or communicating with other proteins in other cells.

Figure 1. Central Dogma



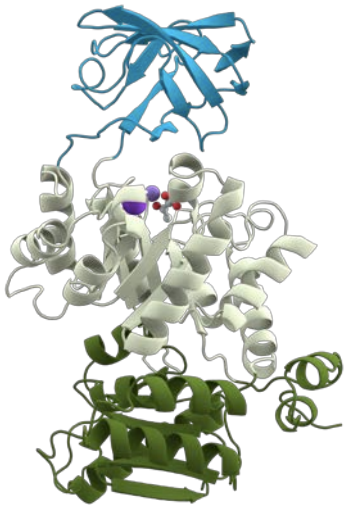
DNA polymerase protein replicates DNA during cell procreation. RNA polymerase protein transcribes a gene on DNA to single-stranded RNA. Lastly, the ribosome translates the RNA transcript to a chain of amino acids (a protein).[2]

Additionally, many biologists have differing opinions on a specific definition of gene as new studies every day modify our understanding of how genes operate. For every subject in biology, answers usually contain the words ‘it depends’ as there exist many exceptions to every rule. According to Gerstein et al., a gene “is a union of

genomic sequences encoding a coherent set of potentially overlapping products"[3]. This definition takes into account the discontinuous nature of a gene from actual coding sequence to elements with no apparent coding function. Also, the authors describe how one sequence can code for different proteins depending on where the reading of the sequence starts[3]. Hence, the authors include "overlapping" in their definition for a gene as many protein products can come from one coding region.

Furthermore, scientists also study the standalone building blocks of proteins called "domains". Protein amino acid sequences fold into multiple shapes depending on the chemical interactions between the amino acids. These shapes may be globular, helical, or sheet-like structures. Specifically, protein domains exist as independently folding units within a protein that have a characteristic shape, and this shape often associates with a characteristic biochemical activity. As evolution favors recycling useful domains, many genes contain "conserved" sequences that fold into similar domains (often with slight modifications). These sub-protein domains mostly serve similar functions in the context of different proteins. Washers, screws, gears and nails serve as a rough analogy to protein domains as they represent reusable parts in man-made structures. Figure 2 represents the Pyruvate kinase protein with each of its domains colored as a visual example.

Figure 2. Pyruvate kinase

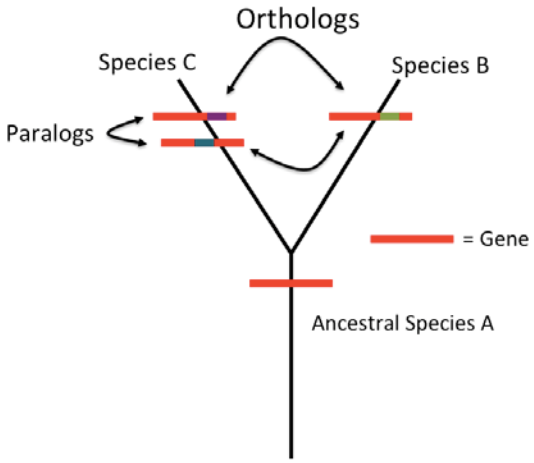


The protein pyruvate kinase has each of its domains colored [4].

Long ago, several hundred domains evolved and show up repeatedly in the context of many proteins and organisms. For example, fruit flies (*Drosophilia melanogaster*), worms (*C. elegans*), and yeast (*S. cerevisiae*) share 744 common protein domains, each found inside many different proteins[5]. Evolution favors the reuse of efficient components as evolution finds it "easier" to tinker than invent new ones.

The study of comparing genomes, genes and their protein products in different organisms falls under the field of comparative genomics. Comparative genomics uses phylogenetic theory and its terminology to visually represent relationships between DNA and protein sequences from different species and their common ancestors. A node with two branches demonstrates a trivial phylogenetic tree (Figure 3).

Figure 3. A Phylogeny and its terminology.



This diagram provides an example of a molecular gene tree. The node joining the branches of the tree represents the common ancestral species. The two branches represent the two diverged species. In this example, the figure demonstrates the genetic history of a single gene. In Species B, a single gene copy exists that was directly inherited from the ancestor, Species A. The green band represents a part of a sequence that mutated since Species A. Likewise, Species C inherited the same gene from Species A. The ancestral gene together with its direct descendents are called "orthologs". Species C contains a second copy of this gene due to a duplication event occurring after the divergence of Species B and C. These duplicated genes represent "paralogs": genes that share a common ancestor but coexist in a single species.

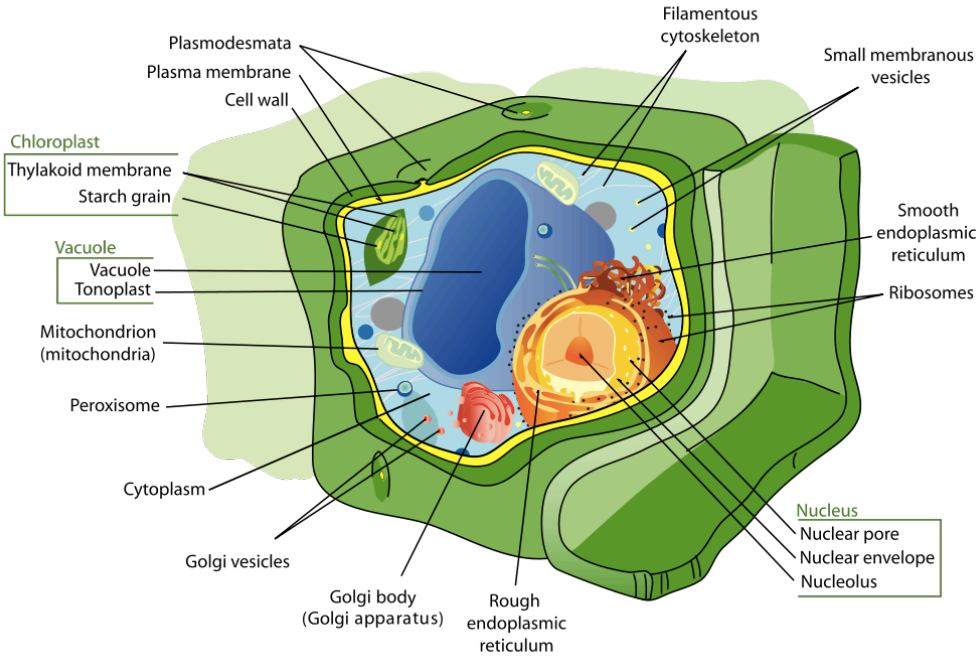
The node represents the point where the two daughter species shared a common ancestor in the evolutionary past. From that point in time, the species accumulated mutations, resulting in two distinct species with changes to their genomes over time. Each species shares common functional genes, each with genetic similarities to their common ancestor. Scientists refer to the ancestral gene and its direct descendants as "orthologs". Additional copies of a gene may be formed within a species' genome due to a duplication event . Scientists refer to these in-species gene duplications as "paralogs." The term "homologous" refers to orthologous and paralogous genes or proteins which, share common genetic ancestry. The POGs database organizes all of the

genes in the four represented species into “putative orthologous groups” (POGs), based on this type of logic.

The POGs database annotates each POG with various types of information that provide insights into the functions of the proteins in that group. To appreciate these features, we’ll need to discuss the basics of cell organization. The POGs database focuses on plants, which have certain characteristic features. Figure 4 describes the layout of a plant cell with its organelles or cell parts. For the purposes of POGs, the chloroplast, nucleus, endoplasmic reticulum, and mitochondrion characterize the most important parts of the cell. The proteins found in these organelles contain special sequences of amino acids called “targeting” sequences, which tell the cellular machinery to localize the proteins to the appropriate organelle.

In particular, plant molecular biologists focus on the chloroplast as it coordinates the processes that captures the energy of sunlight and converts it into chemical energy that fuels life on earth – hence the process called “photosynthesis”. According to the endosymbiotic theory, chloroplasts existed independently as a photosynthetic bacterium until a key event in evolutionary history caused the chloroplast to merge with a larger parent cell symbiotically[6]. Today, the chloroplast retains a small genome of its own, a remnant of the genome in its bacterial ancestor. These genes in the chloroplast coordinate with thousands of genes in the nucleus to produce a photosynthetically competent chloroplast.

Figure 4. The Plant Cell



This plant cell diagram[7] contains the features of most eukaryotic cells except that Plant cells have cell walls versus phospholipid membranes. Plant cells also contain chloroplasts in addition to mitochondrion.

Intriguingly, plant survival depends on both the chloroplast and the nucleus for coordinating photosynthesis. Scientists spend time trying to determine how these genes from both locations coordinate their activities.

Additionally, the mitochondrion represents the second organelle acquired through an endosymbiotic-merging event[6]. The mitochondrion represents another energy powerhouse which, carries out respiration, the process of converting sugars into chemical energy, used for cellular processes. Plant, animal, and fungal cells contain mitochondria, obtaining process sugars originally from photosynthesis, although sometimes indirectly via consumption of other organisms.

Thirdly, the endoplasmic reticulum (ER) provides the site for further protein processing. Proteins targeted to the ER ultimately become secreted as the ER packages and processes proteins for export outside the cell.

Lastly, the nucleus provides enclosure for chromosomes. RNA polymerase transcribes DNA to messenger RNA for delivery to the cytoplasm and onwards to the ER for translation. The nucleus contains thousands of genes used for chloroplast and mitochondrial functions due, in part, to the transfer of genetic material between the locations over evolutionary time[6]. Expression of the small chloroplast and mitochondrial genomes relies on many genes and their protein products originating in the nucleus[8]. For example, the ATAB2 protein from the nucleus functions to translate the *psaB* mRNA, critical for photosynthetic harvesting[9]. Understanding the relationship between nucleus and chloroplast represents the primary reason we developed POGs, but also helps in studying other aspects of plant biology.

Incidentally, transposable elements mark another interesting feature of genomes in all organisms, but especially in plants and animals. Transposable elements colloquially referred to as “jumping genes” code for proteins that pick up its parent gene and move it elsewhere in the genome. Sometimes these genes insert themselves in important genes required for organism function, and thereby disrupt normal growth and development. For example, *crp1* –a nuclear-encoded RNA processing gene required for expression of two chloroplast genes – fails to work on insertion of a transposon into its gene[10]. When the chloroplast fails to work completely or partially, the corn plant appears pale as a result of chloroplast/photosynthetic defects. Many researchers utilize mutations of this type to track down undiscovered genes in the nucleus that affect the

chloroplast as they may visually observe plants, likely to have defective chloroplasts due to mutations in genes needed for chloroplast functions[8].

We now turn to understanding the problems posed by the revolution in DNA sequencing technologies. In the early 2000s, a combined effort between the National Institute of Health and private industry developed high-speed next generation sequencing. This opened the doors to sequencing complete genomes for many different organisms. Since this seminal breakthrough, scientists may easily obtain genome sequences. However, understanding how each gene works in each sequenced genome poses a challenging problem. As such, scientists focus on a relatively few species to study gene functions in depth – using organisms easy for lab work. Scientists then use phylogenetic theory to cross-reference highly documented species to understand genes in a less well-understood species. Orthologous gene annotation across these genomes provides one of the most useful tools in a geneticists toolbox. As orthologous gene function remains conserved between species, scientists may extrapolate the function from one gene in one genome to another gene in a different genome.

POGs serves as an online gateway for inferring function between highly documented genes to less documented genes between plant species. By understanding biological concepts such as the central dogma, protein domains, phylogenetics, plant cell anatomy and whole genomic sequencing the user may appreciate what POGs offers to the field of plant genomics.

Information Technology Background

Biological concepts play a motivational role in the conception and purpose for POGs. In understanding the components and engineering involved in the construction of POGs, we must consider background in programming languages, web development, databases and bioinformatic tools.

Firstly, POGs uses Ruby, PHP, and Javascript to power its capabilities. These three languages represent members of scripting programming languages. In many ways, interpretive scripting languages represent a secondary tier in the programming languages' chain of being. In order to build a scripting language, a programmer uses compiled programming languages to construct the scripting language.

Compiled programming languages translate "human readable" syntax directly into machine code for computer hardware to use. Compiled languages require a greater attention to detail for controlling memory storage and declaring "type" for variables. Like in algebra, variables represent numbers but in programming languages variables may represent characters, integers, decimal values and more. When using compiled languages, one must declare in a program the type of a value the variable contains otherwise the program fails to work. Controlling random access memory also remains a major factor in some compiled languages, as many computers have a limited amount of memory for a program to store information in when running. If one does not manually delete variables containing large amounts of information after use, the program may crash without fulfilling its function.

Hence, interpretive scripting languages such as Ruby, PHP, and Javascript take away the tedious tasks of specifying exactly what a variable must contain and manually

manage the memory space. Scripting languages have an interpreter, which takes code and makes some assumptions about the variables and translates it into machine language. However, the program's speed slows down in comparison to a compiled program, as assumptions may prove costly as the interpreter may fail to create "optimal" machine code. For the purposes of creating POGs and processing complex amounts of genetic information, scripting languages overall provide a quicker way to develop code, outweighing the marginal speed gains by using a more detailed language such as C.

To the point, Matsumoto Yukihiro wrote Ruby in the C programming language to build a purely object-oriented language with many powerful features drawn from older scripting languages such as Python or Perl[11]. For the purposes of POGs, Ruby provided easy syntax and library support for processing many text files containing genetic information as well as great support for connecting to databases.

Secondly, Rasmus Lerdorf wrote PHP or "Personal Home Page Tools" in C to target building websites and connecting to databases[12]. PHP resembles C in many ways but interprets type, manages memory, and comes with functions specific to creating web documents. After its birth in 1994, PHP has risen to become one of the most popular and favored languages for web programming[12]. For this reason, POGs relies on PHP for bridging the database to the web pages.

Lastly, Sun Microsystems (now Oracle) and Netscape Inc. wrote Javascript (ECMAscript) in 1995 for scripting the behavior of the web browser[13]. Browsers such as Firefox, Chrome, and Internet Explorer all implement Javascript interpreters in their software. When a browser loads a page, any included Javascript program executes and

provides users with an enhanced user experience. This enhanced experience could include live search results, drag and drop, sorting, animations and much more. Javascript also allows for loading data after page load, a useful optimization for the user experience. POGs utilizes many tools built in Javascript for optimizing loading speed of the data provided by the POGs database.

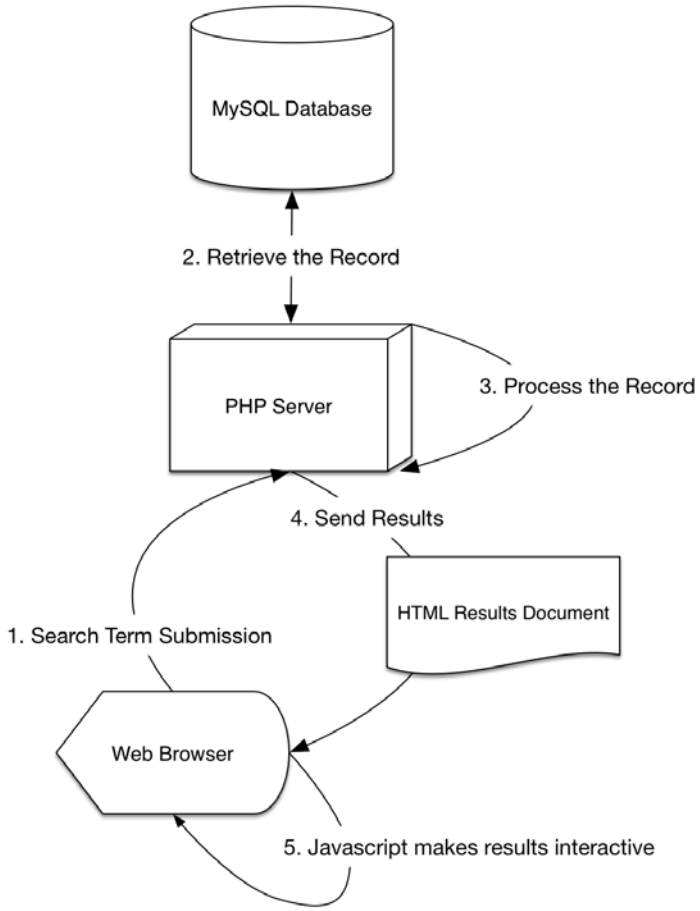
Moving on, the web development field of software engineering relies on languages that render Hypertext Markup Language (HTML) text documents for display in a web browser. PHP and Javascript in POGs form the web development languages used to achieve this. PHP uses HTML as a "templating language" as it provides styles and formatting for web pages. A developer may insert PHP variables inside HTML and the PHP fills those variables in when a browser requests the page. This marks the base case for web application programming.

Most complex web applications rely on a chain of server processes to store, collect, and display the information on the browser. In reality, the web development story becomes quite complex when the web application takes a variety of inputs from user (e.g. results of a web search) and connects the user with the relevant information. Figure 5 describes a generic process for how a web application takes a search term and returns results to the user.

The database provides a "file cabinet" for storing, organizing, and retrieving data efficiently. Databases may handle gigabytes to even terabytes of stored data and keeping storage and retrieval efficient remains a challenge to database designers. In the case of POGs, the MySQL database server[14] manages the tables storing information about all of the genes in each of the organisms represented. Each table contains a row

similar to a single row in a spreadsheet with names columns and each cell contains characters, numbers, dates, and many other data types.

Figure 5. Basic Search Engine Application



The web browser retrieves user input on a search term and sends it to the PHP server. The PHP server application requests and receives the records from the MySQL database. The server processes the record returns an HTML document with the results for the user to view in the browser. Javascript program creates interactive elements for sorting, clicking, visualizing, and manipulating retrieved data.

Next, the PHP application server relies on its own internal architecture to handle information flow through. These advanced applications rely on advanced object-oriented programming and design patterns. As with architecture, constructing a large building requires a larger frameworks, tools, and scaffolding. Incidentally, web

development tools reuse architecture terminology for the tools built on top of web programming languages.

For example, POGs uses a framework called FuelPHP[15], which provides prewritten tools for working with the database, handling routing for the different pages on a web application, managing security for logged in users, and many other generic tools. The “Model View Controller” (MVC) organization strategy or “design pattern” for the framework helps keep the code maintainable and readable when hundreds of lines of code turn into thousands of line of code. The MVC pattern provides a convention many industry practitioners found as best practice since the 1980s[16]. The Model domain provides a programming object, which serves as a direct liaison for each database table and provides functions for users to call and retrieve data. The View in web programming represents the pre-formatted HTML document with placeholder variables for data retrieved by the Controller. The Controller bridges the Model and Views by retrieving database information, processing the data, formatting the View with the data and sending it to browser. By organizing code into files and folders based on this pattern, developers may follow the logic of the application efficiently.

Lastly, the browser-side code written in Javascript creates an interactive experience with the data returned by the PHP server. Once the HTML document loads on the browser, the Javascript code runs and provides interactive features such as the ability to filter results, choose various options with buttons, and more. Javascript also can call the PHP server itself to retrieve more data without reloading the page. Some web applications use spinners to indicate this action when loading more data for the user. The AngularJS library written by Google[17] extends the functionality of HTML

with Javascript itself to provide interactive elements and methods for contacting the PHP server without reloading the page. This allows for optimizations through retrieving only the data without asking the server for images and formatting markup on every reload. When displaying large amounts of images and data, these optimizations allow for a seamless experience as the server may easily slow down when many browsers connect to the website.

Returning back to the biology, the BLAST[18] and MUSCLE[19] sequence alignment algorithms play a large role in the data components of POGs. Once scientists started sequencing genomes on a large scale in the early 2000s, they needed to develop methods for sequence comparison to reveal the differences and similarities among various DNA sequences and among various protein sequences. Hypothetically, the degree of sequence difference between orthologous genes in Species B and Species C reflects the amount of time since those two species diverged (Figure 3). Nonetheless, orthologs can be predicted as the genes that share the most similarity between two organisms. Algorithms have been created to optimize the "alignment" among different proteins or DNA sequences. These can help determine which, gene or protein sequences should belong in one orthologous group.

BLAST[18] in particular provides a quick way to search for and measure the sequence similarity of related genes or proteins. BLAST takes a sequence "query" and genomes to search through as its inputs. Figure 6 shows the output of one BLAST alignment search. BLAST breaks down the query sequence into short 'word' segments and finds the top matches possible for each word[20]. Next, BLAST takes the top matches

for each segment and searches its database for optimal matches for each consecutive word[20].

Another variant sequence alignment algorithm, MUSCLE[19] generates an alignment for a couple sequences at a time, unlike BLAST[18], which can search for similarities across large numbers of sequences. MUSCLE[1,19] uses sequence alignments to output phylogenetic trees (a graphical representation of inferred ancestral relationships) in particular which, POGs uses heavily. The textual output for a MUSCLE alignment resembles the BLAST output unless the user requests a phylogenetic tree (output shown in the next section). MUSCLE[19] uses a scoring matrix table for each letter in the sequences and fills in the score for each match or mismatch. Later, the algorithm finds the best possible score from the matrix for each aligned letter and outputs the result to the user.

Overall, the information technology brings the data to life by creating efficient data 'plumbing' and enriching the user experience through interactive visualization. Each language chosen to assist in the construction of POGs represents a few of many such tools available but these allowed the author to overcome multiple challenges during development. The bioinformatics tools (e.g. sequence alignment tools) allow for scientists using POGs to search and compare gene sequences on a large scale. Arguably, sequence alignment algorithms represent the top most used tools in Bioinformatics and genetics research.

Figure 6. BLAST Search Output

```
> AT5G44740.2 | Symbols: POLH | Y-family DNA polymerase H | chr5:18047903-18051779
REVERSE LENGTH=672
Length=672

 Score =  352 bits (823),  Expect = 1e-96, Method: Compositional matrix adjust.
 Identities = 199/484 (41%), Positives = 226/484 (47%), Gaps = 38/484 (8%)

Query  7    RVVALVDMDCFFVQVEQRQNPHLRNKPCAVVQYKSWKGGGIIAVSYEARAFGVTRSMWAD  66
            RV+A VDMDCF+VQVEQR  P LR  P AVVQY  W GGG IAVSYEAR  GV RSM  D
Sbjct  12   RVIAHVDMDCFYVQVEQRKQPELRGLPSAVVQYNEWQGGGLIAVSYEARKCGVKRSMRGD  71

Query  67   DAKKLCPDLLLAQVRESRGKANLTKYREASVEVMEIMSRFAVIERASIDEAYVDLTSAVQ  126
            +AK  CP   L QV   RGKA+L  YR A  EV  I+      ERASIDE Y DLT A +
Sbjct  72   EAKAACPQIQLVQVPVARGKADLNLYRSAGSEVVSILAKSGKCERASIDEVYLDLTDAAE  131

Query  127  ERLQKLQGQPISADLLPSTYIEGLPQGPTTAEETVQKEGMRKQGLFQWLDSLQIDNLTSP  186
              L      P S +L           G    +    KE  R      W      ++
Sbjct  132  SMLA--DAPPESLELIDEEVLKSHILGMNREDGDDFKESVRN-----WICR---EDADRR  181

Query  187  DLQLTVGAVIVEEMRAAIERETGFQCSAGISHNKVLAKLACGLNKPNRQTLVSHGSVPQL  246
            D  L  G +IV E+R  +  ET F CSAGI HNK LAKLA G+NKP  QT V    V +L
Sbjct  182  DKLLSCGIIIVAELRKQVLKETEFTCSAGIAHNKMLAKLASGMNKPAQQTVVPYAAVQEL  241

Query  247  FSQMPIRKIRSLGGKLGASVIEILGIEYMGELTQFTESQLQSHFGEKNGSWLYAMCRGIE  306
             S +PI K   LGGKLG S    LG++  G+L QF E  LQ H+G   G WL    RGI
Sbjct  242  LSSLPIKKMKQLGGKLGTSLQTDLGVDTVGDLLQFSETKLQEHYGVNTGTWLWNIARGIS  301

Query  307  HDPVKPRQLPKTIGCSKNFPGKTALATREQVQWWLLQLAQELEERLTKDRNDNDRVATQL  366
             + V  R LPK  G  K FPG  AL     VQ WL QL +EL ERL  D   N R+A  L
Sbjct  302  GEEVQGRLLPKSHGSGKTFPGPRALKSLSTVQHWLNQLSEELSERLGSDLEQNKRIASTL  361

Query  367  VV------SIRVQGDKRLSSLRRCCALTRYDAHKMSHDAFTVIKNC-----NTSGIQTE-  414
                    S      K   S    C + RY   K   DAF              GI  +
Sbjct  362  TLHASAFRSKDSDSHKKFPS--KSCPM-RYGVTKIQEDAFNLFQAALREYMGSFGIKPQG  418
```

BLAST[18] output from POGs[1] showing one sequence match result. Using human DNA polymerase, the search result found a match to a plant DNA polymerase protein product. The output displays a “Query” sequence line and a “Subject” sequence line. The “Query” sequence would be a line from the Human DNA polymerase sequence and the “Subject” would be the plant DNA polymerase sequence result.

A Tour of POGs

POGs exists to catalog the proteins encoded in the genomes of four model plant species, Maize, Arabidopsis, Poplar tree, and Rice, into one centralized database. POGs takes advantage of other algorithms, OrthoMCL[21] and Ensemble[22], that predict orthology relationships among all genes/proteins in specified sets of species. These algorithms group all proteins/genes in these species into Putative Orthologous Groups (POGs), and these groups form the core of the POGs database. .

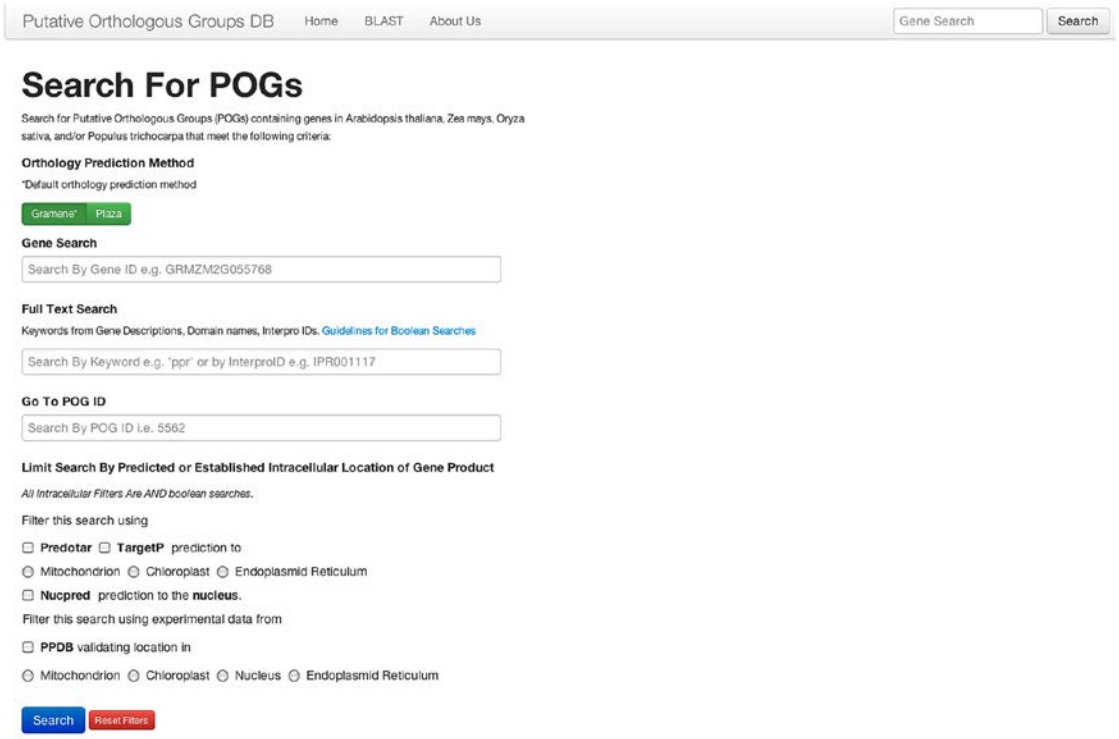
Arabidopsis thaliana represents the first plant genome sequenced and the most intensively studied plant. Consequently, the "annotations" for its genes (information about the sequence, expression and function) tend to be more complete than gene annotations in other plants. Oftentimes, a gene in Maize may be better understood than its ortholog in *Arabidopsis*. By grouping information about orthologs into a single web page, the combined information from these can help scientists infer the function of unknown genes. By searching with an unknown protein, DNA, or RNA sequence using POGs' BLAST tool, one may determine the functionality and purpose for a given sequence as it may have similar function with its genetic cousins in these four species. This cycle of searching with an unknown sequence, retrieving an orthologous group, and studying the group's functional annotations denotes the purpose for POGs.

With this purpose in mind, each section in POGs serves to provide scientists with a powerful search engine, useful information about each orthologous group, and linking to further resources on each group.

The primary landing page on visiting pogs.uoregon.edu gives users a search page based on ortholog prediction method, gene catalogue identification, genetic

descriptions, and cell location of a gene's protein product (Figure 7). POGs features two separate orthology prediction methods, Gramene[2,23] or PLAZA[3,24] for determining whether a group of similar genes belong in a single orthologous group. There exist many theoretical methods for determining the 'optimal' prediction, however each algorithm contains its quirks. By featuring multiple methods, POGs lets the scientists explore multiple predictions and evaluate which is likely to be most accurate for their gene of interest.

Figure 7. Search Page



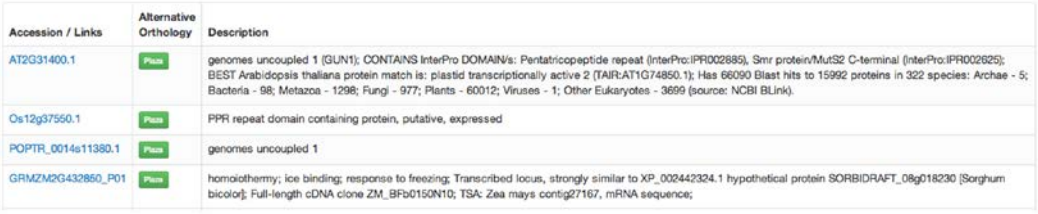
The POGs[1,3] search page located at pogs.uoregon.edu.

A search by accession or gene ID (a unique identifier for each gene in each species) allows scientists already familiar with a gene to lookup a group quickly. In fact, most POGs users prefer searching by this method. POGs also provides full-text search of the functional annotations for each gene catalogued among the four species. If a user

types “DNA polymerase” inside “Full Text Search”, the results will display groups annotated with the word, “DNA polymerase.”

Lastly, the “Limit Search By … Intracellular Location” helps scientists filter results based on where protein products operate inside the plant cell. If one wishes to find all proteins with “PPR domains” that localize to chloroplasts, one would type “PPR” in “Full Text Search”, select location prediction method, Predotar[5,25] or TargetP[6,26] and select “Chloroplast”. These location prediction methods use an algorithm to examine the beginning of a protein sequence to predict where the protein moves to in the cell. Nucpred[7,27] provides a method for determining whether a protein functions in the nucleus. The PPDB[6,28] database provides experimentally validated locations for proteins in plants. This gives greater confidence than an algorithm as an experiment has verified the location. Overall, these algorithms when used together provide a good indication of actual location.

Figure 8. Group annotation section



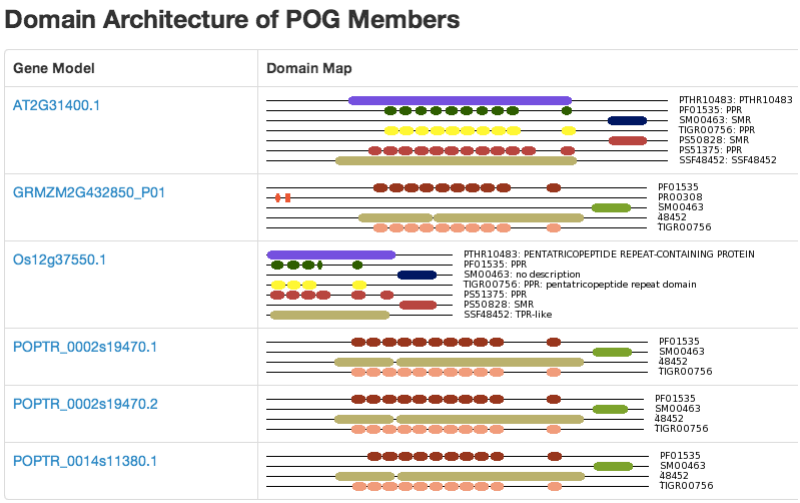
Accession / Links	Alternative Orthology	Description
AT2G31400.1	Plaza	genomes uncoupled 1 (GUN1); CONTAINS InterPro DOMAIN/s: Pentatricopeptide repeat (InterPro:IPR002885), Smr protein/MutS2 C-terminal (InterPro:IPR002625); BEST Arabidopsis thaliana protein match is: plastid transcriptionally active 2 (TAIR:AT1G74850.1); Has 66090 Blast hits to 15992 proteins in 322 species: Archae - 5; Bacteria - 98; Metazoa - 1298; Fungi - 977; Plants - 60012; Viruses - 1; Other Eukaryotes - 3699 (source: NCBI BLink).
Os12g37550.1	Plaza	PPR repeat domain containing protein, putative, expressed
POPTR_0014s11380.1	Plaza	genomes uncoupled 1
GRMZM2G432850_P01	Plaza	homoiothermy; ice binding; response to freezing; Transcribed locus, strongly similar to XP_002442324.1 hypothetical protein SORBIDRAFT_08g018230 [Sorghum bicolor]; Full-length cDNA clone ZM_BFb0150N10; TSA: Zea mays contig27167, mRNA sequence;

The figure displays the members of the orthologous group and their annotation in POGs (http://pogs.uoregon.edu/#/pog/9879)[1,6].

Once the server sends back the search results, a list of POGs ID numbers show up with their descriptions. The user clicks on these ID numbers to view the orthologous group profile page. The first section (Figure 8) features a table of the POG member’s gene IDs plus their descriptive annotations. These annotations may provide further insight into the protein’s function. Oftentimes, one gene in one species contains more

information than its ortholog another species. POGs simplifies the process of inferring genetic functions by taking advantage of its well-known orthologs.

Figure 9. Protein Domain Visualization section



The figure displays the protein domains predicted in each orthologous gene. Each ellipse of one color represents a predicted protein domain. Each row or line represents a different algorithm for predicting protein domains (http://pogs.uoregon.edu/#/pog/9879)[1,8].

The following section named, “Domain Architecture of POG Members” shows domain visualizations of each protein product for this group (Figure 9). Interpro[29] supplies the information about protein architecture for this section. Interpro contains a large collection of algorithms for finding conserved protein domains. From these results, we create graphical representations of these domain matches. Each ellipse represents a given protein domain predicted by various algorithms from Interpro dedicated to this purpose (i.e. Pfam, Interpro, Smart). If a user clicks on the ellipse, the protein domain page from the source opens and displays more information about the domain. Each line with ellipses represents a separate algorithm used for making domain predictions, accounting for the different sized ellipses on each line of the visualization.

Figure 10. Experimentally Verified Locations

Experimentally Verified Intracellular Location

Accession	Annotation	Experimentally Verified Location	Reference
AT2G31400.1	genomes uncoupled 1	plastid	Source Data From PPDB
GRMZM2G432850_P01		plastid	Source Data From PPDB

This table contains information about any genes that the Plant Proteome Database (PPDB)[9,28] verifies localizes to a cellular organelle.

The "Experimentally Verified Intracellular Location" table (Figure 10) uses annotations for genes that the Plant Proteome Database (PPDB)[10,28] experimentally verified belongs in a certain organelle in the plant cell.

Figure 11. Predicted Intracellular Locations

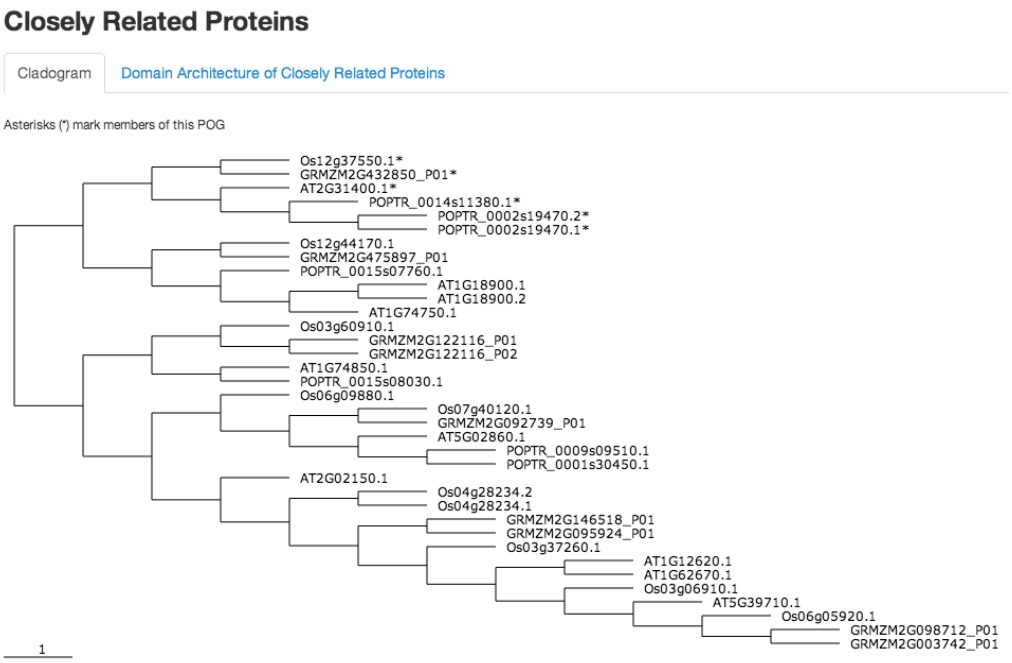
Predicted Intracellular Location

Accession	Prediction Tool	Chloroplast	Mitochondria	Signal Peptide	None / Other	Nucleus	Prediction
AT2G31400.1	Predotar	0.560	0.010	0.000	0.430	–	Chloroplast
	TargetP	0.940	0.087	0.001	0.051	–	Chloroplast
	Nucpred	–	–	–	–	0.63	No Prediction
Os12g37550.1	Predotar	0.010	0.010	0.000	0.980	–	No Prediction
	TargetP	0.082	0.122	0.098	0.905	–	No Prediction
	Nucpred	–	–	–	–	0.23	No Prediction
POPTR_0014s11380.1	Predotar	0.170	0.010	0.000	0.830	–	No Prediction
	TargetP	0.730	0.052	0.001	0.567		Chloroplast

The figure shows the predicted locations by the Predotar[8,25], TargetP[11,26], and Nucpred[12,27] algorithms on protein localization in the plant cell.

The last table (Figure 11), "Predicted Intracellular Location" shows the statistical percentage on the algorithm's certainty for each intracellular location, and a "bottom line" prediction when it passes a certain threshold (usually 80%).

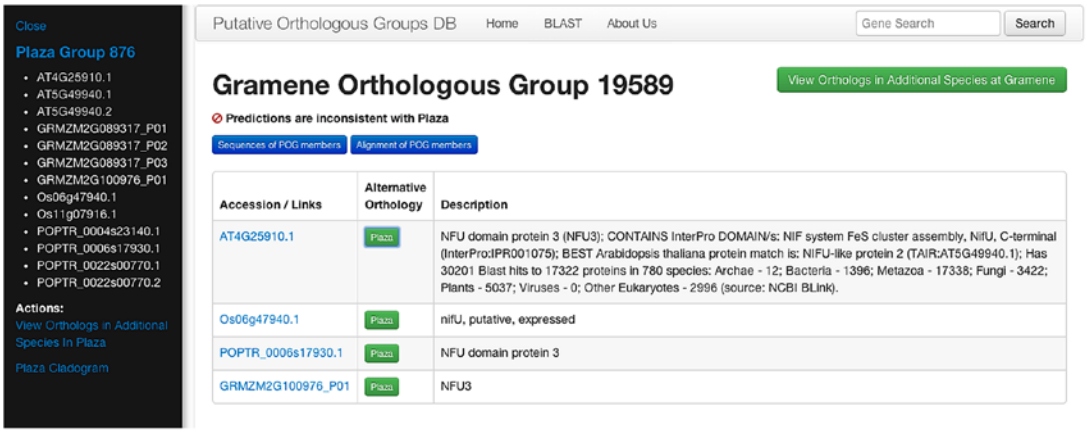
Figure 12. Closely Related Proteins Tree



The figure shows the gene tree for the current orthologous group but also includes proteins matched by BLAST that fall within a certain sequence similarity value. The asterisks indicate the current orthologous group members.

The “Closely Related Proteins” tree (Figure 12) provides with a view of the phylogenetic relationships among POGs members and other closely related proteins. The asterisks denote the orthologous group members. These member branches should cluster together without any non-members in between as an orthologous group contains the closest relatives to an ancestral species gene. Otherwise, the algorithm incorrectly predicted the orthologous group. The tree uses a BLAST[12,18] search, querying each member sequence for additional similar proteins. MUSCLE[13,19] takes the resulting sequences from BLAST and creates a tree. The Javascript library, jsPhyloSVG[14,30], draws the tree in the browser on page load, allowing uses to click similar proteins to visit their orthologous group pages.

Figure 13. PLAZA prediction sidebar



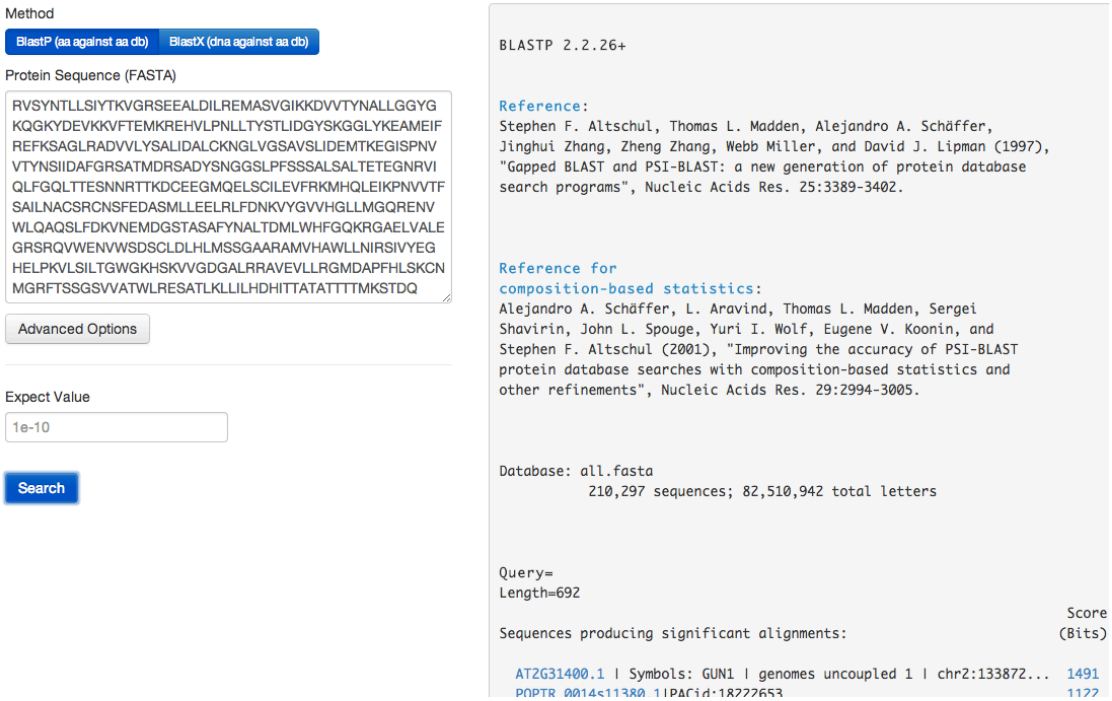
The PLAZA orthologous group sidebar displays a list of orthologous group members. By clicking the "Plaza" button in the "Alternative Orthology" column, one may open the sidebar view.

Because errors often arise in large-scale predictions for orthologous groups, POGs[1,15] provides an additional orthology prediction method (OrthoMCL[21], the algorithm used in the PLAZA[16,24] project). By clicking the green "Plaza" button, the sidebar slides over and one may compare the predictions from the Ensemble pipeline used at Gramene[17,23] (default) and the Plaza[18,24] orthologous members. Although the two methods usually agree, sometimes they disagree as they use slightly different algorithms. A helpful indicator underneath the "Gramene Orthologous Group" heading denotes whether the prediction methods remain consistent or different. One may click the "Plaza Cladogram" link in the sidebar to retrieve a "Closely Related Protein" tree for its member predictions for scientists to visually validate its results.

Lastly, scientists may search POGs by protein or DNA sequence by clicking the "BLAST" menu item at the top of the screen. For example, a scientist may have sequenced a novel plant gene or protein product and wishes to find out how it functions in the plant cell. By pasting the sequence into the sequence search box (Figure 14), the

scientist receives the top protein matches with links to their orthologous group page. From there, the annotations may provide useful insights into how the mystery protein may function.

Figure 14. BLAST search page



A search query using a protein sequence pasted into the search box yields a list of closest matching proteins with links to orthologous group pages.

To summarize, POGs offers many tools and search options for inferring function of poorly understood proteins. By offering extensive gene annotations, domain structures in each protein sequence, intracellular localization, closely related protein trees, alternative orthology predictions, and BLAST[18,19] searching, POGs[1,18] provides an invaluable service to biologists in this field of research.

The Construction of POGs

The development of the Putative Orthologous Groups database presents a unique challenge to the developer in the methods for gathering gigabytes of genome annotation into a database, running Bioinformatic algorithms on the data, the construction of the PHP server and the development of the AngularJS browser client.

Before beginning the creation process, we downloaded genomic data from each plant's genome project. These projects offer text file downloads filled with sequence letters and their annotations free of charge for our academic purposes. We consulted, Gramene[23], The Arabidopsis Information Resource (TAIR)[31], MaizeSequence.org, and Phytozome for providing these text files on each of our four species. We also downloaded open source software such as Predotar[25], TargetP[26], Nucpred[27], BLAST[18] and others for providing the useful information featured on POGs. By inputting our sequences into each individual tool we received output data for storing inside our database.

The first step in creating an online database search engine requires use of a database server such as MySQL[14,19]. Many types of data such as annotations, protein domains, and localization require its own virtual "file cabinet" on the server neatly organized and indexed for future lookup. In this case, a data table in MySQL stores records in rows similar to that in a spreadsheet. Figure 15 provides a diagram containing the different tables and data they store in each.

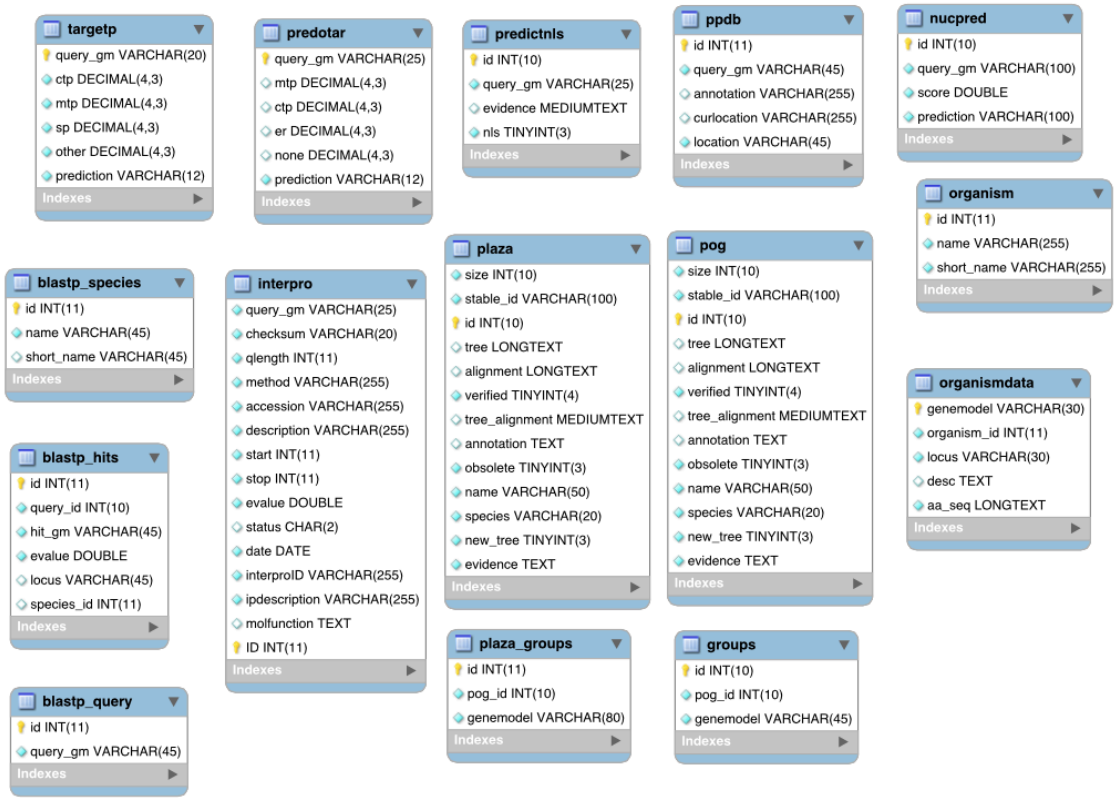
Filling each table with information usually required its own program written in the Ruby[11,18] programming language. For example, all the genes featured in POGs live in a single table "organismdata" with the column names, "genemodel", "organism_id",

"locus", "desc", and "aa_seq". We retrieved text files we downloaded containing the gene id "genemodel", descriptions "desc", and protein sequence "aa_seq". Each text file contained rows of values, typically separated by tabs, and oftentimes very large on the megabytes to gigabytes scale. These files required an efficient program to read each column for these text files, format them correctly, and insert them into the "organismdata" table. Every gene in a particular species receives a numerical "organism_id" to match to a plant species name in a separate table.

Each genome project's text files containing genomic information varied significantly, resulting in many different programs required to read them correctly. This stage in the project represented the second most time consuming portion for the project as we needed to "data mine" various text files for insertion into each of the tables listed in Figure 15.

Running bioinformatics algorithms such as protein location predictors served as a prerequisite for insertion into a table. A Ruby program would run a tool such as TargetP[18,26] to read each protein sequence loaded into POGs from all four genomes, output a location where the protein localized to, and insert into its own database table. I devised various scripts for the BLAST data, sequence alignments, orthology prediction data, protein domain information, and others for these purposes.

Figure 15. MySQL Table Structure



Each table stores data from any information provided on the POGs website. Each table contains different field “columns” specifying the type of data stored there for future lookup.

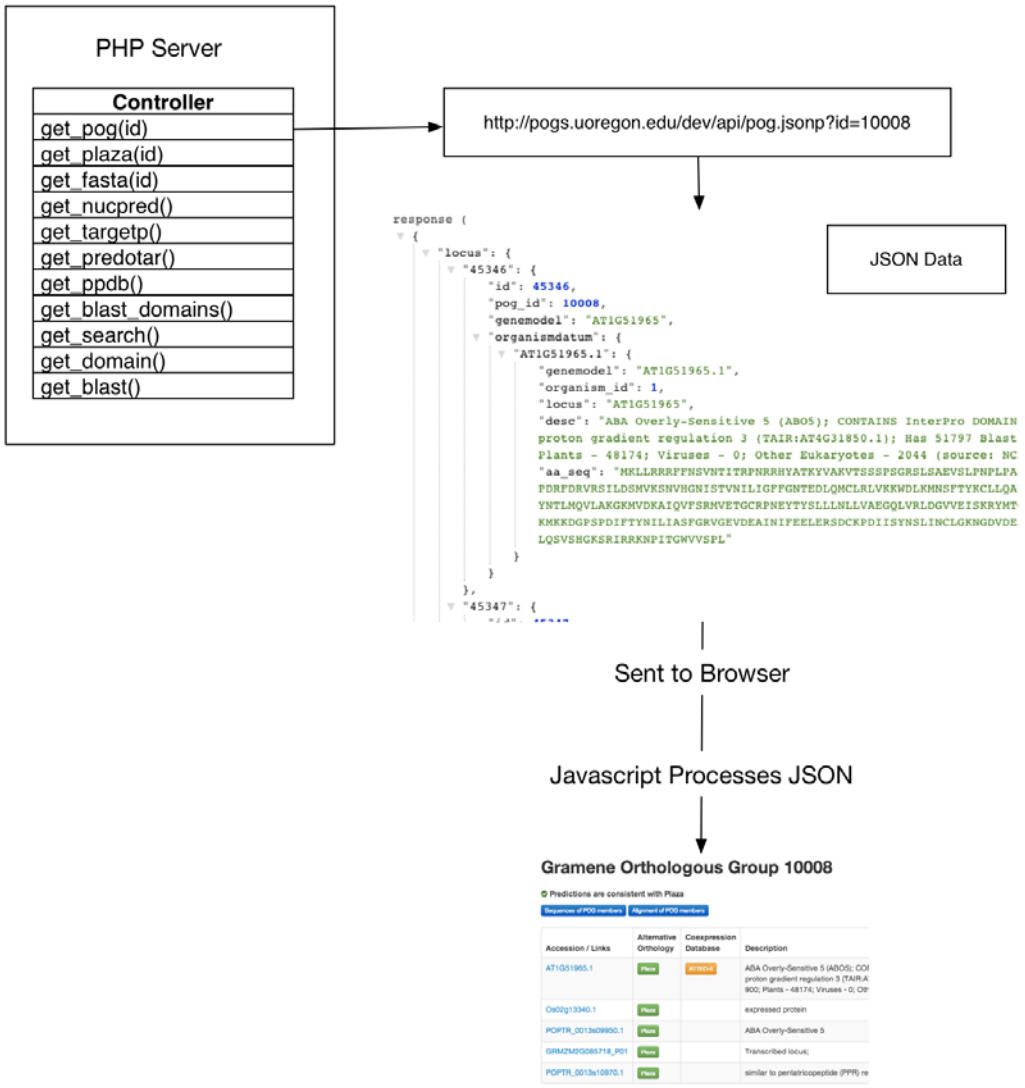
After finishing the data mining step, I built a PHP application server for reading the data from the database and transferring the data to the browser (See Figure 5). FuelPHP[1,15] provided tools in the Model layer for connecting to the database and retrieving data as needed. The Controller layer in the server provided “routes” or website addresses for calling certain pieces of data from the database when needed by the user (Figure 16). When the browser requests these route addresses, the PHP server hands the data for processing by the browser. Instead of the simplified model in Figure 5, POGs uses Javascript Object Notation (JSON) to send data to the browser rather than

HTML. The browser runs a Javascript program when it receives the data and creates the page the user sees on page load. By using this method, the load speed increases as the browser receives data without unnecessary graphical formatting.

The receiving Javascript program uses AngularJS[17] to create the variables inside an HTML document, filling in data from the JSON server response. AngularJS provides tools contacting the PHP server without a browser refresh allowing for a quick response time to any user response such as searching or clicking. By using AngularJS to handle the HTML documents in the browser rather than waiting for the server to render the HTML document, the data flow comes faster and more efficiently. As the POGs database currently contains at least 3 gigabytes of textual data, easing the load of the server prevents long waiting periods for the user to receive the data.

Ultimately, constructing POGs not only represents a project involving biological data processing but also careful planning and design for the technology appropriate for a data intensive web application. By collating and mining data in an organized fashion, a web application provides a powerful tool for displaying relevant insights for scientists to benefit from.

Figure 16. POGs Data Flow Model



Model demonstrates the flow of data from the PHP controller outputting JSON text documents instead of HTML. The Browser receives the JSON data and Javascript creates the graphical markup required to see the orthologous group page.

Case Study: Track down the Gene

Transposable elements help scientists to study genes because they can interrupt a gene leading to a change in organism's properties (a "phenotype"), and the phenotype can be used to infer the functions of the disrupted gene. For example, these "jumping genes" may insert themselves into crucial genes located inside the nucleus required for chloroplast operation. If these insertions disrupt the photosynthetic dynamics, the plant survives only briefly, outwardly displaying its ailment with pale color pigmentation (Figure 17).

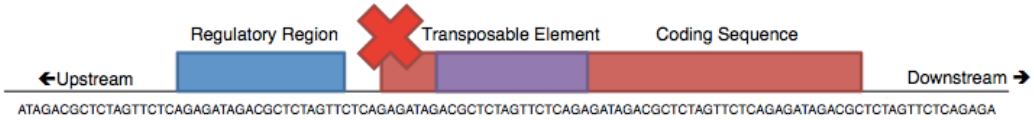
Figure 17. Healthy to sickly maize seedlings

This figure shows healthy maize seedlings on the left and unhealthy on the right. A major disruption in a gene may leave a seedling completely unable to photosynthesis, displaying a white pale color as seen on the right.

Researchers in the Barkan lab identify transposable element insertions that disrupt chloroplast function, as a means toward understanding the nuclear genes involved in the biogenesis of the chloroplast. For that purpose, they developed techniques for sequencing the transposable elements and their surrounding gene locations. Figure 18 demonstrates an invaded gene disrupting its coding sequence. Bioinformatic analysis provides a list of invaded genes and their gene IDs, only one or

two of which represents the true mutant disrupting the chloroplast's photosynthesis. POGs[1] helps filter this list by providing annotations on whether a) the gene product localizes to the chloroplast and b) does the annotation data suggest involvement in chloroplast metabolism. Using logical deduction, one may find the potential gene disruption responsible for the phenotype. Afterwards, lab scientists use this candidate gene in their lab studies to experimentally verify the candidate gene invasion causes the phenotype.

Figure 18. Gene invasion by transposable element



The transposable element inserts itself into the coding sequence preventing the gene from working properly when transcribed into RNA and then translated into a protein.

For example, a researcher discovers a pale corn seedling plant. The researcher sequences all of the transposable elements and their surrounding genes in a set of plants with this same mutation. This process returns a list of gene ID, found in all of the mutant plants; for example GRMZM2G049495, GRMZM2G007729 and GRMZM2G455587.

Next, the researcher browses to http://pogs.uoregon.edu and searches each gene ID using the "Gene Search" field. The annotations for the first gene, GRMZM2G049495 states that it functions as an "acyl-CoA-binding domain-containing protein 6" with a localization to the Endoplasmic Reticulum. As the annotations on the orthologous groups page fail to indicate a connection to the chloroplast, this gene most likely does not cause the pale green mutation. The next gene, GRMZM2G007729,

involves a “low molecular weight heat shock protein precursor” with localization to the mitochondria. This indicates another unlikely candidate[1]. Lastly, GRMZM2G455587, involves “MAP kinase activity; ATP binding; protein amino acid phosphorylation;”[1]. However, its ortholog, AT1G15510.1 in *Arabidopsis thaliana* works as an “EARLY CHLOROPLAST BIOGENESIS2” protein, a very positive indication for a connection between the gene and chloroplast formation. The protein localizes to the chloroplast as well according to the PPDB table. This evidence suggests that the mutation of the gene GRMZM2G455587 by a transposable element causes the mutant phenotype.

The researcher with this knowledge returns to the lab to verify it with experimentation whether this candidate truly causes the issues in this plant.

While simplified, this case study illustrates a daily activity for molecular plant researchers in trying to further understand the relationship between nuclear genes and the chloroplast. Other plant processes follow the same logic – e.g. the development of the corn’s “ear”, or the spacing of leaves along the plant. Without POGs[1], this task would require browsing many different websites containing the individual tools to look for gene annotations, localization, and others. In a vibrant field such as plant genetics, tools like these help lead to quicker breakthroughs important to agricultural practices and even human medicine.

Conclusion

Web applications represent the future for conveying large amounts of data. The browser represents a powerful tool with features rivaling that of desktop application development. In many ways, the Internet has evolved into the universal operating system available on all platforms and devices. As software engineering moves away from the desktop world into the cloud, the flow of information will increase as barriers to accessing data decreases.

Indeed, biological web applications present the future for how biologists use information technology to further their research. POGs[1] effectively provides ready access to a large body of data for inferring the function of novel genetic and protein sequences. The architecture behind POGs[1] represents the cutting edge in current web development practice, furthering emphasizing the importance of Javascript's role in a responsive interface for accessing large amounts of data. Current biological databases use the old model, involving server-side rendering of HTML data rather than sending only pure data to a standalone Javascript program, handling the HTML browser interface. Not only does POGs provide a useful role in a biological sense but also may serve as model for future biological web applications.

We published a paper on the POGs[1] database last fall titled, "POGs2: A Web Portal to Facilitate Cross-Species Inferences About Protein Architecture and Function in Plants" and it received very positive comments during the review process. For example, our reviewer mentioned, "The POGs2 database fills a sorely needed gap in this post genomic era for the plant community."

As of the current writing, the POGs received a citation from a paper involving plant genetics after publication for only 5 months. We expect that this will become a popular tool among plant geneticists, as its availability becomes more widely known.

Bibliography

1. Tomcal, M., Stiffler, N. & Barkan, A. POGs2: A Web Portal to Facilitate Cross-Species Inferences About Protein Architecture and Function in Plants. *PLoS ONE* **8,** e82569 (2013).
2. Horspool, D. File:Central Dogma of Molecular Biochemistry with Enzymes.jpg - Wikipedia, the free encyclopedia. *en.wikipedia.org* (2008). at <http://en.wikipedia.org/wiki/File:Central_Dogma_of_Molecular_Biochemistry_with_Enzymes.jpg>
3. Gerstein, M. B. *et al.* What is a gene, post-ENCODE? History and updated definition. *Genome Research* **17,** 669–681 (2007).
4. Splettstoesser, T. File:Pyruvate kinase protein domains.png - Wikipedia, the free encyclopedia. *en.wikipedia.org* (2012). at <http://en.wikipedia.org/wiki/File:Pyruvate_kinase_protein_domains.png>
5. Rubin, G. M. *et al.* Comparative Genomics of the Eukaryotes. *jstor.org*
6. Dyall, S. D. Ancient Invasions: From Endosymbionts to Organelles. *Science* **304,** 253–257 (2004).
7. Ruiz, M. File:Plant cell structure svg.svg - Wikipedia, the free encyclopedia. *en.wikipedia.org* (2006). at <http://en.wikipedia.org/wiki/File:Plant_cell_structure_svg.svg>
8. Barkan, A. & GOLDSCHMIDTCLERMONT, M. Participation of nuclear genes in chloroplast gene expression. *Biochimie* **82,** 559–572 (2000).
9. Barneche, F., Winter, V., Crèvecoeur, M. & Rochaix, J.-D. ATAB2 is a novel factor in the signalling pathway of light-controlled synthesis of photosystem proteins. *EMBO J* **25,** 5907–5918 (2006).
10. A Barkan, M. W. M. N. D. J. A nuclear mutation in maize blocks the processing and translation of several chloroplast mRNAs and provides evidence for the differential translation of alternative mRNA forms. *EMBO J* **13,** 3170 (1994).
11. Ruby Programming Language. *ruby-lang.org* at <https://www.ruby-lang.org/en/>
12. PHP: History of PHP - Manual. *us3.php.net* at <http://us3.php.net/manual/en/history.php.php>
13. NETSCAPE AND SUN ANNOUNCE JAVASCRIPT, THE OPEN, CROSS-PLATFORM OBJECT SCRIPTING LANGUAGE FOR ENTERPRISE NETWORKS AND THE INTERNET. *web.archive.org* (1995). at <https://web.archive.org/web/20070916144913/http://wp.netscape.com/newsref/pr/newsrelease67.html>
14. MySQL :: MySQL Documentation: MySQL Reference Manuals. *dev.mysql.com* at <http://dev.mysql.com/doc/>
15. FuelPHP » A simple, flexible, community driven PHP5.3 framework. *fuelphp.com* at <http://fuelphp.com/>
16. Krasner, G. & Pope, S. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. *itu.dk* (1988). at <http://www.itu.dk/courses/VOP/E2005/VOP2005E/8_mvc_krasner_and_pope.pdf>
17. AngularJS — Superheroic JavaScript MVW Framework. *angularjs.org* at <https://angularjs.org/>

18. Altschul, S. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25,** 3389–3402 (1997).
19. Edgar, R. C. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.* **32,** 1792–1797 (2004).
20. Pevsner, J. *Bioinformatics and Functional Genomics*. (Wiley-Blackwell, 2009).
21. Li, L. OrthoMCL: Identification of Ortholog Groups for Eukaryotic Genomes. *Genome Research* **13,** 2178–2189 (2003).
22. Vilella, A. J. *et al.* EnsemblCompara GeneTrees: Complete, duplication-aware phylogenetic trees in vertebrates. *Genome Research* **19,** 327–335 (2008).
23. Jaiswal, P. Gramene database: a hub for comparative plant genomics. *Methods Mol. Biol.* **678,** 247–275 (2011).
24. Van Bel, M. *et al.* Dissecting plant genomes with the PLAZA comparative genomics platform. *Plant Physiol.* **158,** 590–600 (2012).
25. Small, I., Peeters, N., Legeai, F. & Lurin, C. Predotar: A tool for rapidly screening proteomes forN-terminal targeting sequences. *PROTEOMICS* **4,** 1581–1590 (2004).
26. Emanuelsson, O., Nielsen, H., Brunak, S. & Heijne, von, G. Predicting subcellular localization of proteins based on their N-terminal amino acid sequence. *J. Mol. Biol.* **300,** 1005–1016 (2000).
27. Brameier, M., Krings, A. & MacCallum, R. M. NucPred Predicting nuclear localization of proteins. *Bioinformatics* **23,** 1159–1160 (2007).
28. Sun, Q. *et al.* PPDB, the Plant Proteomics Database at Cornell. *Nucleic Acids Res.* **37,** D969–D974 (2009).
29. Hunter, S. *et al.* InterPro in 2011: new developments in the family and domain prediction database. *Nucleic Acids Res.* **40,** D306–12 (2012).
30. Smits, S. A. & Ouverney, C. C. jsPhyloSVG: a javascript library for visualizing interactive and vector-based phylogenetic trees on the web. *PLoS ONE* **5,** e12267 (2010).
31. Lamesch, P. *et al.* The Arabidopsis Information Resource (TAIR): improved gene annotation and new tools. *Nucleic Acids Res.* **40,** D1202–10 (2012).