An analysis pipeline for the processing, annotation, and dissemination of Expressed Sequence Tags.

Joseph P. Morris
University of Louisville

AN ANALYSIS PIPELINE FOR THE PROCESSING, ANNOTATION, AND DISSEMINATION OF EXPRESSED SEQUENCE TAGS

By

Joseph P. Morris
B.S., University of Louisville, 2006

A Thesis
Submitted to the Faculty of the
University of Louisville
J. B. Speed School of Engineering
as Partial Fulfillment of the Requirements
for the Professional Degree

MASTER OF ENGINEERING

Department of Computer Engineering and Computer Science

May 2009

AN ANALYSIS PIPELINE FOR THE PROCESSING, ANNOTATION, AND DISSEMINATION OF EXPRESSED SEQUENCE TAGS

Submitted by:___________________________________
Joseph P. Morris

A Thesis Approved on

____
(Date)

by the Following Reading and Examination Committee:

____
Eric Rouchka, Thesis Director

____
Ming Ouyang

____
Gail DePuy

____
David Schultz

ABSTRACT

Due to the complex nature of interactions at the genomic level as well as the large number of proteins present in an organism, understanding the functions of various genes that are expressed is essential. Creating an analysis pipeline for Expressed Sequence Tags (ESTs) is one way to accomplish this, allowing a researcher to quickly take a set of sequences, perform all necessary analysis operations, and publish the data in a database with a graphical user interface (GUI). This pipeline falls into several steps. First, the data must be preprocessed to remove any extraneous sequence data, low-complexity regions, and regions that repeat throughout the genome. Next, it is necessary to combine a large number of ESTs into larger sequences that better describe the underlying mRNA. After larger contiguous sequences have been constructed, putative functions can be assigned to each sequence, whether part of a larger grouping or a singleton.

An application of this pipeline using 3906 ESTs generated from trichome tissue of *Pelargonium* x*hotorum* (commonly, the geranium plant) resulted in 425 contiguous sequences using the CAP3 program. These sequences, along with the 2208 sequences that are not a part of a contig, were then BLASTed against the non-redundant protein database to assign putative functions to each sequence. Finally, BLAST2GO was run on these BLAST results in order to assign a GO (Gene Ontology) to each sequence. These annotations were then added to the database for later investigation by researchers.

In order to aid researchers in the further analysis of the annotated sequences, a mySQL database was used for data storage and a GUI was developed using Java and Java

Server Pages. In addition, an applet for viewing the Sanger trace files for each sequence is included to further aid the researcher in determining the validity of the data.

ACKNOWLEDGMENTS

First, I would like to thank my parents, James and Mary Morris, as well as the rest of my family, for supporting me through the time it took for me to complete this project. It was only with their encouragement and support that I was able to accomplish everything that I have accomplished.

Next, I would like to thank the members of my committee, Dr. Eric Rouchka, Dr. David Schultz, Dr. Ming Ouyang, and Dr. Gail DuPuy for ther time, advice, and guidance. I would especially like to thank Dr. Eric Rouchka for his patience and invaluable advice that helped to make this a reality.

I would like to thank the rest of the faculty, staff, and students of the University of Louisville for all the help I received, including Dr. Ted Kalbfleisch, Dr. Elizabeth Cha, Christy Bogard, Dr. Tim Hardin, Nate Johnson, and everyone else that has given me advice along the way. I would also like to thank Steven Yelton and Chris Keen at Mission Data for giving me lots of good ideas about web design and architecture.

Finally, I would like to thank my friends who gave me encouragement and support through this difficult process, including Cassandra Hobbic, Matt Bates, Michal Kruger, Jack Hazlett, Jamie Fries, Jon Rohner, Joe Evans, and John Finkbone. This includes the times they were a distraction from the work at hand. You know who you are and the times of which I am speaking. I thank you for it.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

In the analysis of Expressed Sequence Tags (ESTs), there are numerous techniques and tools available to aid the bioinformatician. Due to the nature of ESTs, it is important to preprocess the sequences, create consensus sequences of the ESTs, then attempt to annotate the sequences to glean useful information from the set. In addition, it is helpful to organize the sequences so that the information can be made available for use by other scientists.

There are a number of issues related to the analysis, storage, and retrieval of ESTs. Because of the varied needs of scientists, it is necessary to have a comprehensive system in order to assist with a wide range of interests. In addition, varying organisms may have different needs for specific annotations or additional analyses that will aid the researcher. Finally, the system must be organized in a way that it acts as an aid to research, not as a hindrance.

Analysis of ESTs is, by no means, standardized. The fact that ESTs themselves are only partially composed of usable sequences makes some sort of trimming of the sequences a necessary first step. After that, because each EST is a small section of genetic sequence, larger contiguous sequences must be constructed from the trimmed sequences. Finally, the larger sequences and those that do not fit into larger sequences can be annotated with some putative function.

Storage of ESTs into a database and the data’s subsequent retrieval must be done with two considerations in mind. First, the data must be stored so that the raw data can be extracted from the database, if needed for additional or alternate analysis. Second, the analysis that accompanies the data must be organized so that it can be easily searched and downloaded for the researcher's use. These two principles allow for a system that is suited both from an information technology view as well as for the end user, the researcher.

Several programs are readily available that can aid in this task. Among them, VecScreen and RepeatMasker can be used to eliminate unwanted vector contamination and repeat regions, respectively. CAP3 (Huang & Madan, 1999) is a program that constructs consensus sequences out of a number of ESTs. With this data in hand, BLASTing against the non-redundant database at NCBI can help to assign putative functions. Further analysis and grouping can be done with BLAST2GO (Conesa & Gotz, 2008), a program that assigns potential Gene Ontology (Ashburner et al., 2000) names to sequences using existing BLAST data.

In addition to the analysis pipeline, a structure has been created to store the data for ease of access. This has been designed so allow for several factors. First, the data must be easy to access, both for further analysis as well as for the researcher to add and update entries. Next, the structure must be flexible enough to be reused with other data. Finally, the data must be secure so that the researcher can allow outside users to only see data that has been deemed accurate and trustworthy.

The storage and access to the data was accomplished using open source, freely available software. In this case, a mySQL (DuBois, 2005) database was created to store the data, satisfying the goal of ease of access. MySQL can be queried for results with minimal computer training, is widely documented, and can be easily integrated into most programming languages. In addition, a web interface was created that allows a user without database knowledge to access the data for analysis as well as administrative purposes, such as altering the data as necessary. These web pages were generated using JSP, or Java Server Pages, and servlets. Run on an Apache Tomcat servlet engine, this group of technologies allow for dynamic web development within the Java programming language. Also, with the necessary security already present in the database software, a researcher can feel safe knowing that the data cannot be compromised and can only be accessed by qualified individuals.

It is these three areas that are the primary focus of this project: the preparation and analysis of the data, the design of a flexible database, and the design of web pages to disseminate the data in a secure and user-friendly manner. By successfully applying this approach to the set of ESTs used in this project, it is hoped that the work can be generalized to allow other researches the ability to use this system for other data sets.

II. BACKGROUND

A. Biological Background

DNA is the basis of all biological life, and therefore, a brief review of molecular biology is important for understanding the work of this project. First, a discussion of the "Central Dogma of Biology" will give context to the biological data present in this system. Next, a discussion of ESTs, how they are generated, and their importance in analysis of protein expression will help to show the meaning of the data from a biological standpoint. Finally, a discussion of the specific tissue and organism used for this project will aid in showing the importance of this data and why biologists are interested in it in the first place.

1. Central Dogma of Molecular Biology

The "Central Dogma of Molecular Biology" (Crick, 1970) is a term used to describe the general path that cells use to create working proteins from DNA. Using DNA as a template, the cell goes through several steps to eventually produce the proteins needed for every aspect of a cell's existence. The steps generally taken by the cell are replication, or the copying of DNA needed for cell division; transcription, or the creation of a template of RNA molecules from existing DNA; and translation, or the transition of the RNA from a string of nucleotides to a strand of amino acids, which fold into a protein. Each of these steps will now be looked at in greater detail.

DNA is a made of two strands of nucleotides, connected in a double helix, with the two strands running in opposite directions. The nucleotides are adenine, cytosine, guanine, and thymine, and are commonly represented by the letters A, C, G, and T. Each nucleotide is made up of a sugar backbone which connects the nucleotide to its neighbor, and a base, which is unique for each type of nucleotide. The bases connect the two strands by loose bonds, called hydrogen bonds, such that adenines always connect to thymines and cytosines always connect to guanines. This makes the two stands of a DNA molecule reverse complements of each other, meaning the two strands run in opposite directions and have complementary bases to one another.

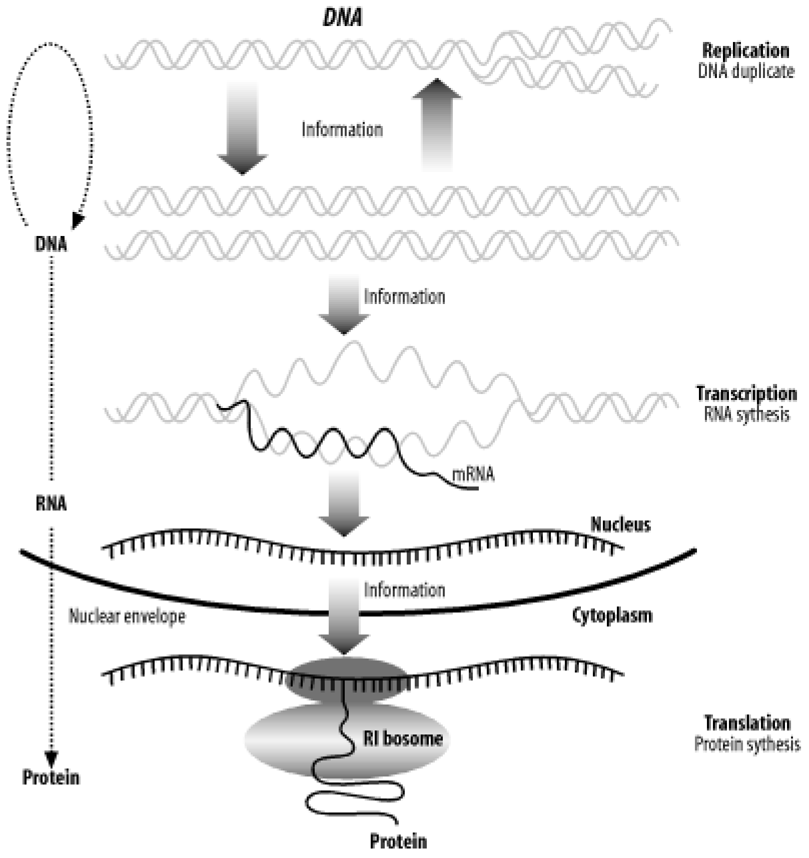


FIGURE 1: THE CENTRAL DOGMA OF MOLECULAR BIOLOGY

(Ussery, 2003)

Through various complex mechanism within cells, the cell determines that particular proteins must be produced in order for proper cell functioning. How each protein is created is encoded within that cell's DNA but it must be extracted first. A template is created from the section of the DNA that encodes for the needed protein. This process is known as transcription. A strand of messenger RNA, or mRNA, is created. mRNA, like DNA, is made up of nucleotides stranded together (with the exception being uracil is used instead of thymine), in the reverse complement of one of the DNA strands. Unlike DNA, RNA is single-stranded. This mRNA is then processed and sent outside the nucleus to be translated into a protein.

Translation is the process by which a cell converts the template made of mRNA into a working protein. This is done at the ribosomes within the cell and involves reading the mRNA, three bases at a time, and creating a chain of amino acids that correspond to each of the three-nucleotide group (called a codon). From this translation, the strand of amino acids folds into a protein. This protein is the product that is needed by the cell to perform the task that prompted its initial creation.

The Central Dogma has been found to be somewhat simplistic in describing all molecular biology behavior (1970). In retroviruses, single stranded RNA can be transcribed into double stranded DNA, the reverse of the way transcription occurs in the Central Dogma. In addition, it has been discovered that some RNA, known as small interfering RNA (SIRNA), act in a regulatory capacity to prevent the expression of other genes (Hamilton & Baulcombe, 1999). These exceptions are important to note because, since ESTs are representative of RNA, it is possible that the EST does not necessarily represent part of an mRNA that would ultimately result in a functional protein. If the

EST was generated from mRNA that functions as a retrovirus or SIRNA, there is no protein product.

2. Expressed Sequence Tags

Now that the general mechanism for protein creation has been established, it is important to understand what ESTs are and how they can be used to gain insight into this process. Expressed Sequence Tags, or ESTs, are short sequences of nucleotides that are copies of parts of mRNA from a cell (Adams et al., 1991). They are rarely complete copies due to restrictions in the technology and are often of relatively low quality. However, they do represent parts of the DNA that have been transcribed, potentially to become proteins. The “expressed” part of their name derives from this: these are parts of the DNA the cell has found a need to express.

The process for generating ESTs is somewhat complex, but necessary to capture the information contained in the fragile and easily degraded mRNA. Because mRNA is unstable outside of the nucleus, the first step is making a template of the mRNA fragments that can undergo subsequent treatments. These template strands are called complimentary DNA, or cDNA. The cDNA fragments are then cloned.

The next step in the processing of ESTs is to clone the cDNA fragments into amounts of the sequences that can be more easily read by a sequencing machine. The process involves inserting the cDNA segments into an organism that rapidly replicates its DNA, usually bacteria such as *E. coli*. The circular DNA strand inserted into the bacterium cell is replicated quickly. The resulting cells, which have replicated the cDNA segments, can then have the segments removed for sequencing.

Removal of the section of the DNA of interest requires cutting the circular DNA at points outside of the cDNA section, known as restriction sites. Because each end of the cDNA segments had additional known segments added, through biochemical processes, those sections can be sought out and the DNA can be cleaved at those sites. This leaves the segments of interest plus some of the cloning vector, separate from the rest of the DNA. Figure 2 shows an overview of this process.

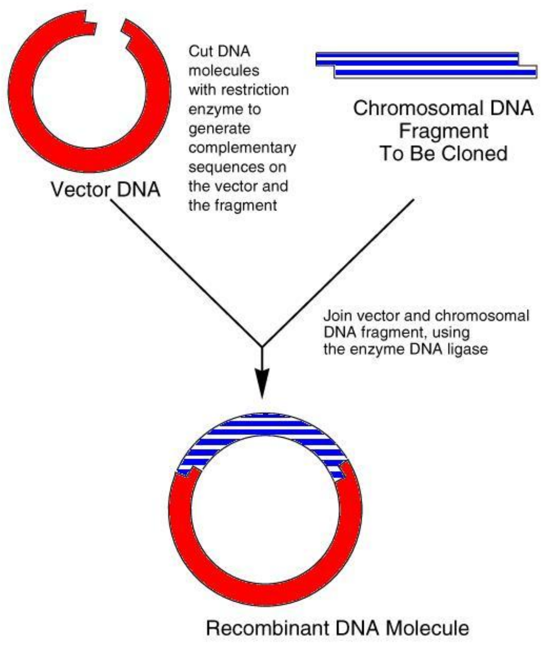

Vector DNA
Cut DNA molecules with restriction enzyme to generate complementary sequences on the vector and the fragment
Chromosomal DNA Fragment To Be Cloned
Join vector and chromosomal DNA fragment, using the enzyme DNA ligase
Recombinant DNA Molecule

FIGURE 2: GENERATION OF ESTs

3. <u>Sanger Sequencing</u>

Once the cDNA sequences are separated from the rest of the vector plasmid DNA in the bacteria, the sequencing can begin. Sequencing begins replicating the cDNA with nucleotides that are marked with fluorescent markers that allow subsequent identification. After the sequences have been marked, a machine able to read the fluorescent markers is employed and the result is a trace file. This trace file can then be used to determine the underlying sequence.

In Sanger sequencing (Sanger, Nicklen, & Coulson, 1977), the sequence to be sequenced, known as the template, is mixed with a combination of regular deoxynucleotides and deoxynucleotides, which are nucleotides that terminate replication. These are labeled with four different fluorescent markers. DNA polymerase is added, allowing for replication of the template.

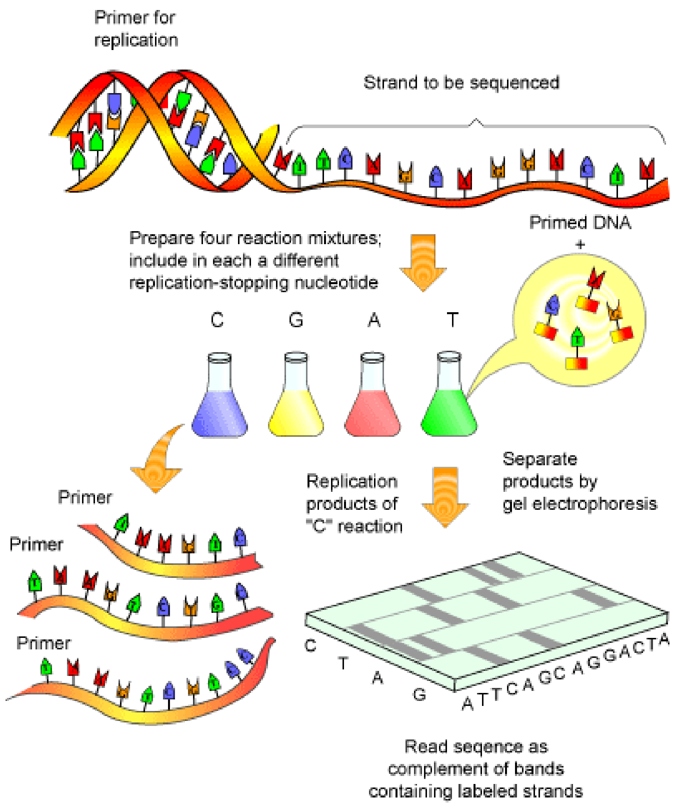

Primer for replication
Strand to be sequenced
Prepare four reaction mixtures; include in each a different replication-stopping nucleotide
Primed DNA +
C
G
A
T
Primer
Primer
Primer
Replication products of "C" reaction
Separate products by gel electrophoresis
C
T
A
G
ATTCAGCAGGACTA
Read seqence as complement of bands containing labeled strands

FIGURE 3: SANGER SEQUNCING (Kae, 2003)

The reaction results in fragments starting at the 5’ end of the template. Each of the fragments is terminated by a marked nucleotide. The segments are separated by size using gel electrophoresis. Because each of the different nucleotides is marked with a dye that fluoresces at a different wavelength, the underlying template can be determined by reading the signal from the markers from one end to the other. This results in a file that can be viewed in a trace viewer, as seen in Figure 4.

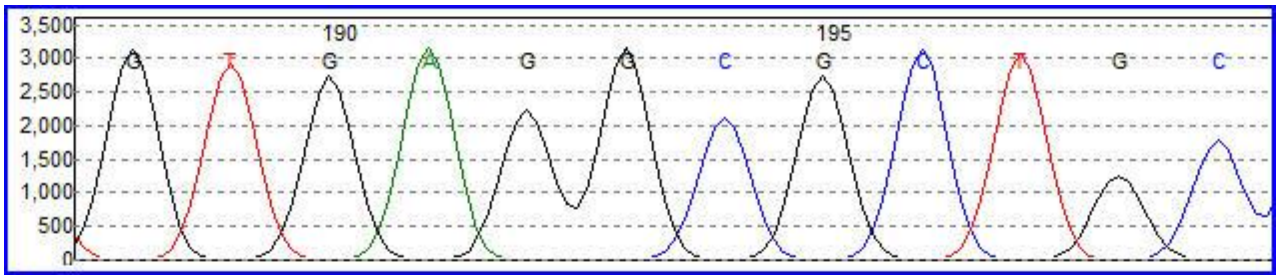


FIGURE 4: SANGER TRACE FILE (http://en.wikipedia.org/wiki/DNA_sequencing)

4. <u>Trichomes</u>

It is also important to understand the tissues that are being sequenced for this particular project and why scientists would be interested in the genes of this particular tissue. Because genes are differentially expressed in distinct tissues, both the species and the specific tissue are important for an understanding of the ultimate goal of this research. So, the question remains: what is a trichome and why is it significant.

Trichomes are outgrowths on surface of a plant. Most commonly, these are manifested as a hair which is present on the stalk or leaf surface of the plant. They aid the plant in several ways, most commonly as a defense from herbivorous pests and offer protection from frost and water evaporation at the surface. Besides the protection of the plant, however, trichomes have been found to secrete substances that are can be used in beneficial ways with other organisms.

There has been significant research into the substances secreted by trichomes. The resulting metabolic products have been found to be useful as food additives, natural pesticides, and pharmaceuticals (Schilmiller, Last, & Pichersky, 2008). In addition, because of the ease of access to these organs, further understanding of the metabolic pathways of the rest of the plant has been achieved.

The trichomes of geraniums (*Pelargonium* x*hortorum*) have been found to contain several compounds important in potential uses as natural pesticides because of the resistance of the plant to particular types of mites. The research is not just limited to beneficial effects on plants, however. Compounds from trichomes are used for the treatment of malaria and are showing promise for use in treating some cancers.

B. Technology Background

There are numerous tools and databases currently available to aid in the analysis of biological data. Among these are programs for aligning sequences, BLAST (Altschul, Gish, Miller, Myers, & Lipman, 1990), T-Coffee (Notredame & Suhre, 2004), and ClustalW (Chenna et al., 2003), and databases for comparing sequences, such as the dbEST (http://www.ncbi.nlm.nih.gov/dbEST/), the non-redundant protein database at NCBI (www.ncbi.nlm.nih.gov/), and PubMed (http://www.ncbi.nlm.nih.gov/pubmed/). Within this set, it is the job of the bioinformatician to determine which tools to use as well as how to use them in order to gain insight into the raw biological data. The next section will discuss several tools that were used, a brief explanation of how they work, and how they helped in the final analysis of the ESTs.

1. BLAST

One of the great tools that has been created for the analysis of any DNA-based set of data has been BLAST, or basic local alignment search tool. BLAST allows a researcher to input one or more biological sequences and, in a relatively short amount of time, determine where in a database of various sequences, a similar sequence occurs. Housed at the National Center for Biotechnology Information (NCBI), BLAST has access to the sequences that have been submitted by various researchers, many with an annotation describing a function or putative function. By doing a similarity search, a researcher can get a better understanding of a sequence's function by looking at its similarity to other annotated sequences. BLASTing the ESTs against the non-redundant protein database was one of the early steps of the analysis process. Though it will be discussed in greater detail later, it is important to understand that it is by using BLAST that any potential function can be assigned to the sequences.

BLAST comes in a number of forms and can be run using a number of parameters to assure the BLAST search is optimal for the circumstance. Among the various forms of BLAST are megablast, used for searching highly similar searches, and bl2seq, used for searching two sequences against each other (as opposed to against a database). In addition, BLAST can search various combinations of amino acid and protein sequences, such as nucleotide against a nucleotide database (blastn), protein against a protein database (blastp), and nucleotide against a protein database (blastx).

2. Phred

Taking trace files, which are of an analog nature, and translating them into a discrete DNA sequence is the job of programs such as phred (Ewing, Hillier, Wendl, & Green, 1998). Phred is an analysis tool that takes trace files, analyzes their content, and generates a file of data that quantifies the underlying sequence, the confidence the program has for each base call, and the position in the file where each peak resides. It is the link between the biology lab and computer analysis. For each base call, a quality score is generated, which is based on the likelihood of an incorrect call. A quality score of 10 represents a 90% confidence, 20 represents a 99%, 30 for 99.9%, etc. Looking at the quality scores of a trace file helps determine how much of the resulting sequence is accurate.

Phred files contain a good amount of information about the trace that is invaluable in the subsequent displaying of the traces. First, it contains the sequence as determined by phred. This is generated by the program analyzing the four signal channels and determining what base is most likely at each point.

For each base in the sequence, phred adds two additional pieces of information. First, it gives a quality score for how likely the call is correct, as detailed above. The second piece of information is the location of the peak in the trace that corresponds to the base. This is important in displaying the traces since it allows lining up the sequence with the peaks.

3. CAP3

Another difficult task is to align the thousands of ESTs into overlapping sequences. As previously mentioned, ESTs are not complete copies of the expressed mRNA sequences, but only partial copies. Due to sequencing technology constraints, ESTs are only 500 to 800 bases long, whereas mRNA sequences can be thousands of bases long. In addition, since much of the sequencing is random, repeats of the same expressed regions can exist. Therefore, all of these separate, smaller sequences must be aligned into larger contiguous sequences where appropriate. CAP3 (Huang & Madan, 1999) is a program that allows for this alignment.

Developed at Michigan Technology Institute by Huang and Madan, CAP3 uses the information directly out of the EST files to form contiguous sequences from the fragments. Using the quality of the sequence reads as well as the sequences themselves, CAP3 does multiple alignments to generate larger sequences. It is these larger sequences, as well as the single sequences which do not align with others, that are BLASTed.

The CAP3 program works in three distinct phases. First, the quality scores, as calculated by phred, are analyzed so that lower quality scores at the ends of the sequences are not taken into effect in the alignments. Next, sequences are connected by finding the overlapping regions between ESTs. Finally, consensus sequences are created with quality scores based on choosing the highest quality score of the smaller reads that make up the consensus sequence (Madan, 1999).

4. GO Ontologies

Another helpful database to use for classifying sequences is the GO database (http://www.geneontology.org/). GO, or gene ontologies, are standardized descriptions that are used to classify a gene based on a broad category: biological process, cellular location, or molecular function. The GO terms are built in a tree-like fashion, with some terms deriving from others. A single gene can be associated with multiple GO terms. By adding GO terms to a set of sequences, a description of the gene can be created that is more standardized than the descriptions given in BLAST.

The GO annotations are organized to give a simplified and uniform description of a gene without losing the complex behavior and interactions that a gene may possess. Each GO term is a number that can be used as a key to the GO database where more information about that family of genes can be found. Each term is first categorized by its function as a cellular component, a molecular function, or a biological process. In addition, each GO term describes the annotation as a specific description of another term, an “is a” relationship, or a smaller part of a larger term, a “part of” relationship.

However, the difficulty is in mapping the sequences to GO terms. BLAST2GO is a program and database developed to allow for the mapping from a set of BLASTed sequences to potential GO terms. By inputting a file of BLAST results, BLAST2GO uses its database to assign a putative set of GO terms, if possible. By assigning GO terms, a number of sequences can have additional, standardized annotations for further classification and analysis.

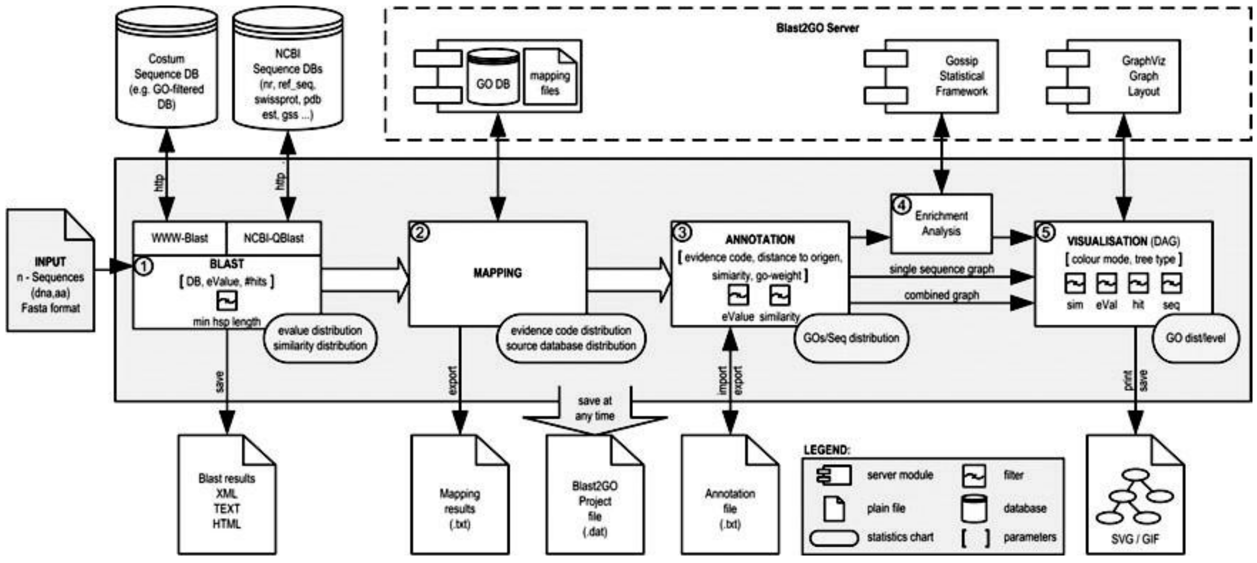

Costum Sequence DB (e.g. GO-filtered DB)
NCBI Sequence DBs (nr, ref_seq, swissprot, pdb est, gss ...)
GO DB
mapping files
Blast2GO Server
Gossip Statistical Framework
GraphViz Graph Layout
http
http
INPUT n - Sequences (dna,aa) Fasta format
WWW-Blast
NCBI-QBlast
1 BLAST [DB, eValue, #hits]
min hsp length
evalue distribution similarity distribution
2 MAPPING
evidence code distribution source database distribution
3 ANNOTATION [evidence code, distance to origen, simiarity, go-weight]
eValue similarity
GOs/Seq distribution
4 Enrichment Analysis
single sequence graph
combined graph
5 VISUALISATION (DAG) [colour mode, tree type]
sim eVal hit seq
GO dist/level
save
export
save at any time
import export
print save
Blast results XML TEXT HTML
Mapping results (.txt)
Blast2GO Project file (.dat)
Annotation file (.txt)
LEGEND:
server module
filter
plain file
database
statistics chart
[ ] parameters
SVG / GIF

FIGURE 5: BLAST2GO SCHEMA (Conesa et al., 2005)

III. METHODOLGY

This project breaks into three main areas of development. First, an analysis pipeline was constructed in order to go from unprocessed ESTs to a set of organized, annotated sequences. Second, the infrastructure was put into place in order to accommodate further analysis of the sequences by researchers through a database. Finally, a web interface was created to allow the researcher without database training to be able to interrogate the data.

A. Analysis Pipeline

The generation of ESTs is not precise. Because they can be generated quickly and relatively inexpensively, the resulting data is sometimes unusable. This can be because of sequencing errors, ESTs that are not long enough to be of use, or regions of low complexity. Therefore, preprocessing is necessary in order to obtain data that is usable and free of unwanted subsequences, such as repeat regions. First, due to the technology used for the sequencing, sequencing vectors are present in each sequence that must be removed. Next, there are areas of the sequencing where extremely low quality reads lead to unreliable sequences. Finally, regions of sequences containing known repeat regions must be masked to avoid incorrect alignments. All of these must be taken into account before any detailed analysis of the sequences can take place.

The raw data is comprised of several parts. First, there is the trace file, which is the raw data generated from the experiment. This is a binary file that can be used to generate a graph of the trace. Next, there are files containing the sequence and its quality scores. Finally, for this set of ESTs, there is a file containing the annotations that were assigned to the sequences by hand.

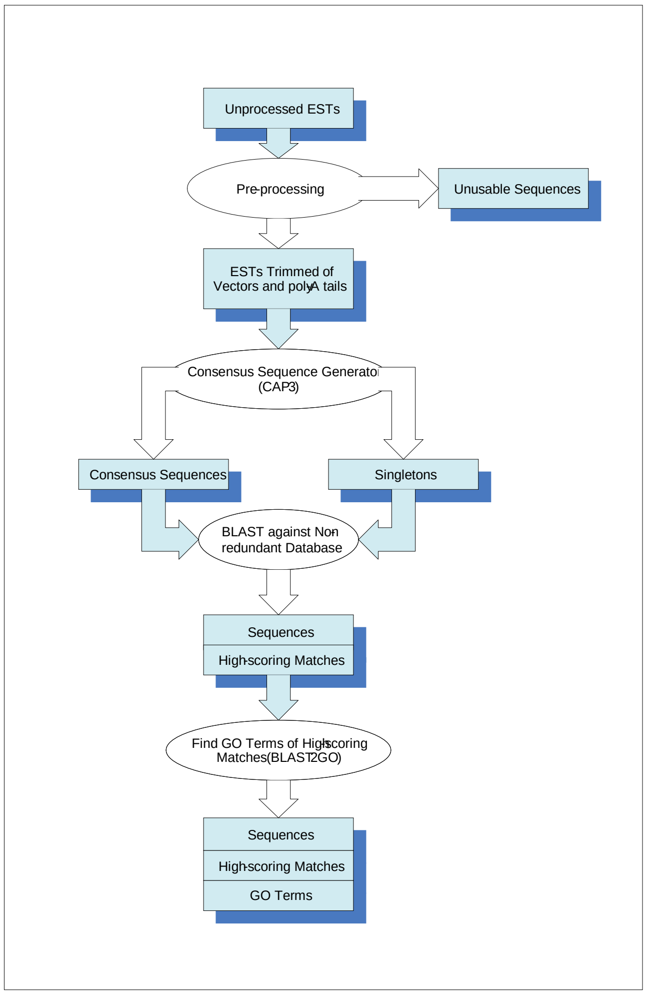

Unprocessed ESTs
Pre-processing
Unusable Sequences
ESTs Trimmed of Vectors and polyA tails
Consensus Sequence Generator (CAP3)
Consensus Sequences
Singletons
BLAST against Non redundant Database
Sequences
High-scoring Matches
Find GO Terms of High-scoring Matches(BLAST2GO)
Sequences
High-scoring Matches
GO Terms

FIGURE 6: ANALYSIS PIPELINE DATAFLOW

The technology for sequencing DNA using the Sanger method has a drawback that must first be compensated for before the actual sequence can be determined. The beginning and end of each sequence will have sections of low quality scores which must be removed because the data is too unreliable to be used as viable sequence. Therefore, the first step in the analysis process is the removal of the ends where the quality score is too low.

The process for quality trimming the sequences is straightforward. The phd file contains the base calls and the resulting quality score for each call. For the start and end of each sequence, the bases were deemed of low quality until a window of 10 bases with a quality score of 20 or above was found. Because a score of 20 means that phred is 99% confident with the called base, 10 bases in a row of scores at or above this threshold generally means the rest of the sequence is acceptable. This analysis was done from both ends to remove all base calls of low quality resulting from technology drawbacks of sequencing.

After quality trimming, the sequencing vectors had to be removed from each sequence. As previously mentioned, the process of generating ESTs leaves small, identical sequences at the beginning and end of the resulting data. These are the result of the sequencing vector used in the process of generating the sequences. However, because these vectors are not actually present in the mRNA that has been sequenced, they must first be removed in order for any sort of alignment to occur. Not doing so would lead to all sequences lining up along these identical end sequences.

To perform the vector removal, several options were weighed. First, NCBI has a tool called VecScreen (2005) which, using a database of some known vectors, scans a sequence and removes any vector contamination. There are databases containing many known vectors, such as UniVec (2005) and EMVEC (2009), which can be BLASTed against for potential vector matches.

For this set of sequences, the decision was made to manually remove the vectors through string manipulation rather than through a tool or database. This was because it was known at the onset that all of the sequences would be contaminated with the same vector and that vector, the blueScript vector, was known beforehand. Therefore, it was an easier task to simply BLAST that sequence against each sequence for matches and remove any part of the beginning or end of the sequence with a significant BLAST match. The FASTA file containing the vectors is listed in Appendix VI.

The next task in the pre-processing step involved the removal of low complexity parts of the sequences. First, Repeat Masker (http://www.repeatmasker.org/) was run on the sequences. Because there are regions of DNA that are often repeated, causing difficulties in sequence alignments, these must be masked from their current nucleotide to "N", the symbol for any nucleotide. This is to help prevent two regions which contain the same repeated region, but are not the same region, to be inadvertently aligned to each other.

Because ESTs are generated from expressed mRNA, it is also important to note that some of the sequences will contain poly-A tails. Poly-A tails are a result of the post-transcription processing of mRNA that takes place in the cell. A number of adenine nucleotides are added to the end of each mRNA strand for protection against degradation of the strand in the cell. Because these are not present in the actual DNA, but added after transcription, it is possible that they were transcribed into the ESTs. Any ESTs containing a long sequence of adenines at the end of a sequence were also trimmed of these nucleotides.

The next step in the analysis pipeline is running CAP3 on the remaining sequences in order to construct consensus sequences, or contigs, from the separate ESTs. Because ESTs are not representative of complete transcribed mRNA strands, any cases where the same expressed mRNA strand was sequenced more than once can be combined. This gives longer and more complete representations of existing mRNA as well as eliminates redundant sequence portions.

After a set of contiguous and individual sequences has been generated, a first attempt at at assigning function to the sequences can be executed. The result of CAP3 is two types of sequences: contigs made of two or more sequences and those sequences that are made of only one sequences, called singletons. This step involves BLASTing each sequence against the non-redundant database at NCI using blastx. Blastx translates the nucleotide sequence into an amino acid sequence in order to search against a protein database. The non-redundant database, or nr, contains all submitted proteins with an annotation for most. By BLASTing each sequence against this database, similarities between these proteins and the generated ESTs can be linked.

There are issues with using this methodology to assign function to the ESTs. First, there is no way of knowing how the annotations were assigned to each protein in the nr database. These proteins were annotated by the submitters who may have determined the annotations using experimental or computational means. These annotations are often based on previous entries in the nr database. Thus, previous errors are often reiterated in subsequent annotations. In addition, because the annotations are not standardized, various wording can be used, making it difficult to glean similar meanings from proteins that are in fact similar.

The final step in the analysis pipeline is to analyze the high-scoring matches that result from BLAST. For each sequence, zero or more matches result. For those without matches, no further analysis can take place. In addition, those without significant matches cannot be further analyzed. However, for those with one or more significant matches, BLAST2GO (Conesa et al., 2005) is used to attempt to assign GO terms.

The final result of the analysis pipeline is the existance of a set of sequences that have GO terms associated with them. In addition, there is a set that did not have significant enough BLAST hits to have GO assignments. Since intial analysis of the ESTs is complete, the next step of the project was to develop a database to store the data.

B. Database Design

The schema was developed for storing the ESTs, corresponding data, and analysis results are shown in Figure 7. An explanation of each table follows.

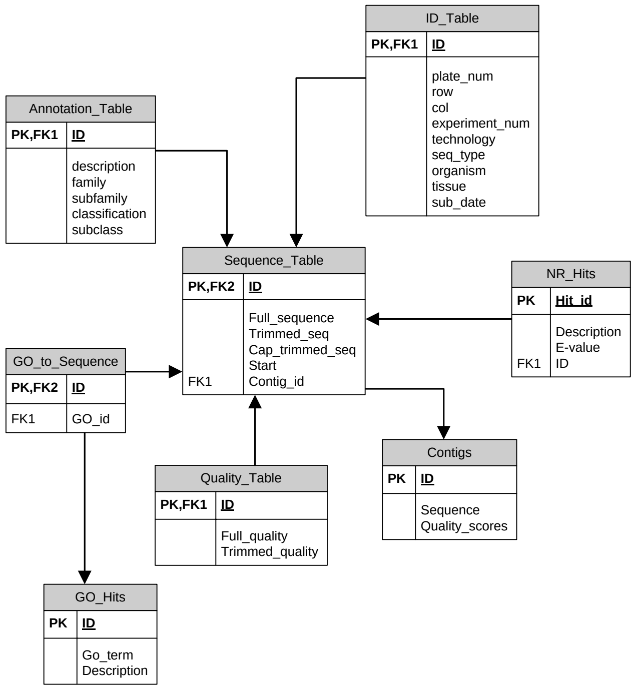


FIGURE 7: DATABASE STRUCTURE

Sequence_table: This contains information about the sequences themselves. It contains the original sequence, the sequence as trimmed by vector removal, the sequence as trimmed by quality score by CAP3, and a reference to the contigs table if the sequence is part of a larger contig. In addition, if part of a contig, there is a field which notes where in the contig the sequence begins.

ID_table: This table contains the information originally present with each sequence. It details the experimental details of the sequence, such as its placement on a plate, the organism the sequence came from, and the experiment that generated the sequence.

Quality_table: This table contains information from the phred file about the quality scores of each base call.

Annotation_Table: This table contains information about the annotations given to the ESTs initially through previous analysis. Because these sequences were initially annotated by hand, this gives a good comparison for checking the results of the analysis.

Contigs: This table holds the results of CAP3, including the sequence and quality scores determined by the program.

NR_Hits: This table contains information from the BLAST hits. For each sequence, all BLAST hits are stored. The actual information contained in the hit, such as the alignment, are stored in files on the server instead of the database.

GO_Hits: This is a table containing the GO IDs and descriptions of all GO terms.

GO_to_Sequence: This table is a link between sequences and the GO terms. The relationship between GO terms and sequences is many to many, necessitating this table.

C. Interface Design

Having a comprehensive and useful interface to allow researchers the ability to query data in a meaningful way is difficult. It is important to be able to show the data and its aggregate annotations while still allowing the researcher the ability to conduct additional analysis on the data in either a raw or formatted form. In addition, enough information about the annotations must be present in order to allow the researcher the ability to accept or reject the annotations, based on a threshold of desired accuracy of the results. These were all design considerations that were taken into account when developing a web interface.

It was decided that Java would be the primary language for development. This is for several reasons. First, web development with Java is well-supported and a broad knowledge base exists as a useful resource. Next, through the Java Data Bridge Connection, or JDBC, querying a database within Java is simple and effective. Finally, the use of Java Server Pages, or JSPs, allow for effective dynamic web pages that are supported across all platforms and web browsers.

Another reason why Java was chosen was because of an applet that was taken from the NCBI website and extended for use in this project. The NCBI Trace Viewer was originally designed as a trace viewer to either be run at the command line or embedded in a web page. This has been expanded in ways that improve its performance and eliminate some of the drawbacks of the initial product. Being an applet, it was written in Java and requires a Java-based web project for its implementation.

JavaServer Pages, or JSPs, are made up of two elements. The main part of the page consists of traditional HTML and follows the tag rules of HTML. In addition, however, there are JSP tags, which are denoted by containing a percentage sign, as in <%=variable%>. Anything contained inside of these tags can be Java code. At the time of compilation, JSPs are converted to servlets, which can then produce a web page by replacing anything in the tags with data. This allows for web pages that vary in content based on Java constructs, such as the data contained in data structures and variables, as well as taking advantage of control loops, such as "for" and "while" loops. Data is passed to and retrieved from these tags through the session, where variable values can be stored.

Because there can be a considerable amount of processing involved within the Java part of the JSPs, each of the JSPs is backed by a Java class which does all computations, leaving the JSPs free of lengthy tags of Java interspersed with HTML. All database calls, data manipulation, and organization is left to the background class, leaving only retrieval of data from the session as the task of the tags within the JSP. This makes the code for the page more readable and easier to eventually add Cascading Style Sheets, or CSS, for formatting. It is based on the Resource-Oriented Programming principles.

ROP is based on the REST architecture. REST, or Representational State Transfer, first detailed by Roy Thomas Fielding (Fielding, 2000), is a set of design principles that describe a series of rules and organizational paradigms for the ordering, manipulation, and transfer of data. ROP takes these principles and applies them to create a RESTful web architecture.

The core of a ROP-organized web program is the Resource Servlet. The Resource Servlet, which is inherited by all subsequent servlets, is in charge of directing data to the correct method within the servlet. The URI contains the information about what servlet is being called, what method within that servlet, and any data that is required to retrieve information from the database or manipulate information. The method within the servlet finishes by storing information in the session and rendering a new JSP page.

For example, there is a page which conducts BLAST searches against the tags in the trichome database. The JSP page, *blast.jsp*, contains a box for entering a sequence. When the user submits the sequence by hitting the button, the sequence is sent to the class *Blast.java* using the URI "http://server/TriPages/blast/search?sequence=AAACCGTT". The Resource Servlet tokenizes the URI and separates the name of the class that is being called, in this case Blast, from the method called, in this case "search".   It is in this method where BLAST is run and the results are saved to the session in a variable called *results* and the *results.jsp* is rendered. The results page can then retrieve the contents of the results variable from the session and include them when rendering the page. Examples of the source files are included in Appendix I.

There are several reasons why this architecture improves the organization of the project. As stated, it removes much of the Java from the JSP, allowing easier reading of the code. Next, because only small amounts of data are sent to the servlets, the methods can be easily reused based on what parameters are sent in the URI. Finally, it allows for a class encapsulate a number of methods that deal with related data, but have different tasks.

The following specifications were used in developing the TriPages web browser:

1. The original data must be stored along with information about trimming done to remove vectors, poly-A tails, low quality and repeat sections, as well as any trimming done by CAP3.
2. The trace files must be able to be viewed from an applet-based viewer.
3. The annotations of the data must be easily accessible and searchable to allow the researcher to mine information from the database.
4. The data must be searchable using BLAST.
5. An administrative researcher must be able to alter or remove data and annotations that are found to be inaccurate.

1. Data Storage

Several steps in the analysis pipeline trim the underlying sequences. The sequencing vector are removed, the poly-A tails are eliminated, and the sequences are trimmed for quality by CAP3. This leads to sequences which are thought to be the most meaningful subsequences of the originals. Because a researcher may have less-stringent thresholds for using the ESTs, the data was stored so that the original sequences as well as the trimmed sequences can be retrieved.

Another important factor is that the ESTs were constructed into contiguous sequences by the CAP3 program. Although this is what was needed for this project, another researcher may want to look at the ESTs separately, without the use of CAP3. Therefore, all ESTs were stored in their original form, along with information about how to reconstruct them into the CAP3 contigs.

2. Applet

The ability to view the trace files and to investigate the signals for accuracy is key in several areas of analysis in bioinformatics. The trace file is a binary representation of an analog process, the process of reading signal strengths of a sequence, which is turned into a binary file for further analysis. The phred base-calling program, in most cases, does a reasonable job of detecting the underlying sequence, but occasionally incorrectly calls bases due to experimental error, the existence of polymorphisms on the sequence, or an insertion or deletion of a base on a sequence. Any of these can lead to results that may need to be verified by looking at the trace.

The applet used for this project was originally developed by NCBI. It was designed to retrieve trace files either from a specified file or from the NCBI Trace Archive. In order to meet the needs of this project, it was altered in several significant ways. The result is a more robust applet that allows for the viewing of trace files as well as other significant information about the sequences.

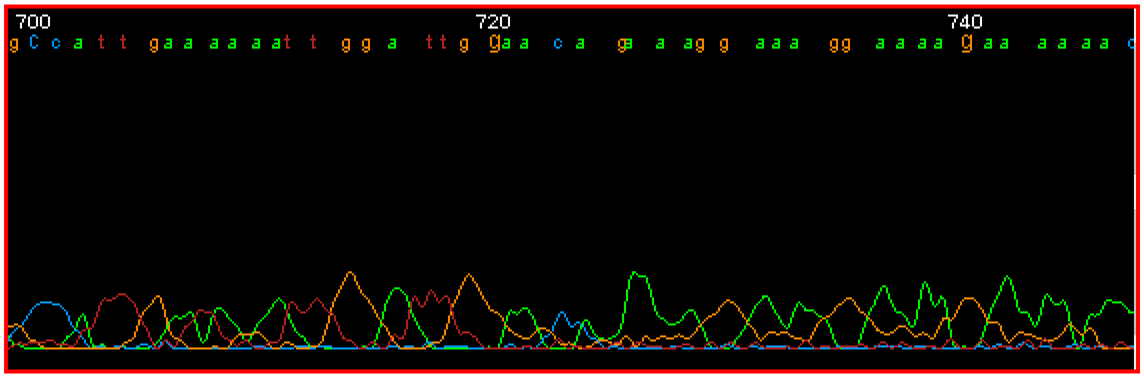


FIGURE 8: LOW-QUALITY TRACE DATA

Initially, a problem was encountered with the formatting of the trace files due to the sequencer that was used. With this change in formatting, it was impossible for the trace viewer to read the called sequence from the file. Therefore, an alternate form was used for the trace viewer. When the applet is called, several actions are taken. First, the trace file is located. Next, phred is run on the trace file in order to get the called bases and locations directly from the phred file. This information is then used in conjunction with the data from the trace file to render the trace and bases.

3. Searchability

A vital use of the database and its web portal is the ability to mine the data in meaningful ways. To facilitate this, several ways to search the data are built into the interface. Among the ways that the data could be used, BLASTing against the data is essential to allow users to compare the sequences in this database with sequences of their own. Comparison of new sequences with those in the database allows the user to view similar sequences and annotations to gain more understanding about their own sequences.

In addition to searching using BLAST, the interface allows for searching the sequences based on other factors. Besides searching by sequence name, the sequences have several ways to search by annotations. Each sequence has a family, putative function based on BLAST searches, and GO annotations based on BLAST2GO. These can searched through a number of interfaces.

4. Administrative Access

There are several places in the process where errors can be introduced. Thus, it is necessary to alter the data when these errors are discovered. Whether in sequencing errors, data trimming, or annotation discrepancies, it is important to allow an administrative researcher the ability to alter or remove data. This is accomplished using an administrative tool.

The most important part of the process of altering data is to retain information about the initial data so that accidental or malicious changes can be restored to their initial form. This is accomplished using audit tables within the database. Any time data is changed in the database, a trigger within the database records the ID of the record that was altered, the name of the user that changed the data, the date of the change, and the original data. This is enough information to reconstruct data if necessary as well as determine who and when the data was altered. The trigger is fired on updates and deletes.

IV. RESULTS

A. Analysis Pipeline

For the implementation of this project, 3906 sequences were used for analysis. These sequences range from 70 to 1433 bases in length and each was accompanied by the following information:

1. qual file: This is a file containing the quality score for each of the bases as determined by the sequencing machine. An example of a quality file can be found in Appendix II.
2. scf file: This is a binary file of the sampling of the raw data from the sequencing machine. It is the file that is read in order to generate the trace file in the trace viewing applet of the database browser. These files are approximately 100 kb in size.
3. seq file: This is a file that contains the sequence in FASTA format. An example of a seq file is contained in Appendix III.

In addition to the files generated by the sequencing machine, the following additional information is attached to each sequence:

1. Annotation: Each sequence has been previously annotated manually by a researcher and information about the family, classification, subclass, and a description is contained in the file contained in Appendix IV.
2. phd file: Phred was run on each of the scf files. Because phred acts as an independent base-caller, the resulting phd file contains information about the

underlying sequence, quality scores, and the location of each of the peaks in the trace file. The sequence and quality scores are ignored because this information is already present in other files. The location of the peaks is used for the trace viewer applet. An example of a phred file is contained in Appendix IV.

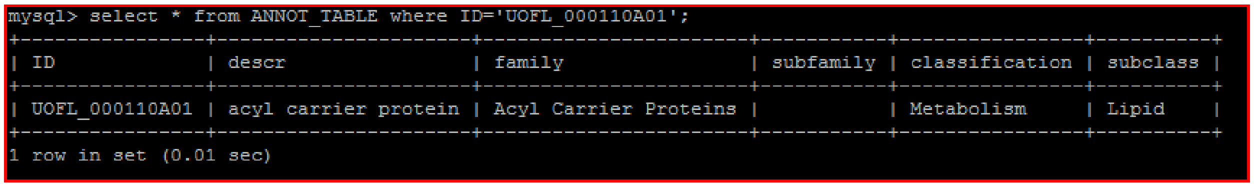
The next step was to trim the sequences of sequencing vectors. These are the vector fragments present due to the sequencing process. The process for removing the vectors is as follows:

1. Create a BLASTable database containing the vector sequences.
2. BLAST each of the EST sequences against the vector sequences.
3. Analyze the BLAST results, noting the positions where the vector matches either the beginning or end of the EST sequence.

Next, any poly-A tails were removed from the sequences. Because mRNA is processed in the cell by adding a series of adenines to the sequence to prevent degradation outside of the nucleus, this part of the sequence is not present in the genome and will not be translated into a subsequent protein. Therefore, it should not be included in the trimmed sequences. This is accomplished by removing any multi-adenine oligonucleotide sequence on the ends of the EST sequences.

After the trimming of the sequences was completed, CAP3 was run on the trimmed sequences to generate contiguous sequences. The resulting file contains information about each sequence, which contig the sequence belongs to, and the starting position of the sequence within the contig. This led to the compilation of 425 contigs, accounting for 1,698 of the sequences, leaving 2,208 single sequences.

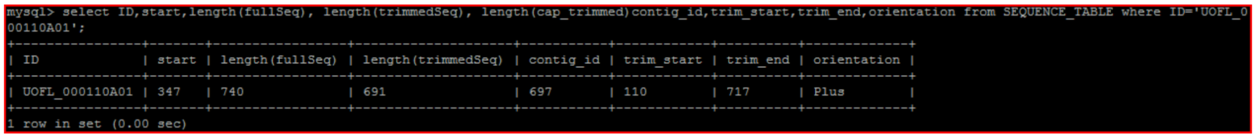
Once the information about the sequence annotations, trimming positions, and contig composition was compiled, the information was added to the database. This is enough information to populate the ANNOT_TABLE and SEQUENCE_TABLE. Example of a query run to display annotation information about a sequence is shown in Figure 9.



```
mysql> select * from ANNOT_TABLE where ID='UOFL_000110A01';
+----------------+----------------------+-----------------------+-----------+----------------+----------+
| ID             | descr                | family                | subfamily | classification | subclass |
+----------------+----------------------+-----------------------+-----------+----------------+----------+
| UOFL_000110A01 | acyl carrier protein | Acyl Carrier Proteins |           | Metabolism     | Lipid    |
+----------------+----------------------+-----------------------+-----------+----------------+----------+
1 row in set (0.01 sec)
```

FIGURE 9: ANNOT_TABLE QUERY

The query to get an entire record from the SEQUENCE_TABLE results in a large and poorly formatted result due to the large sequences, so a query was run to display the information in this table and the lengths of the sequences, shown in Figure 10.

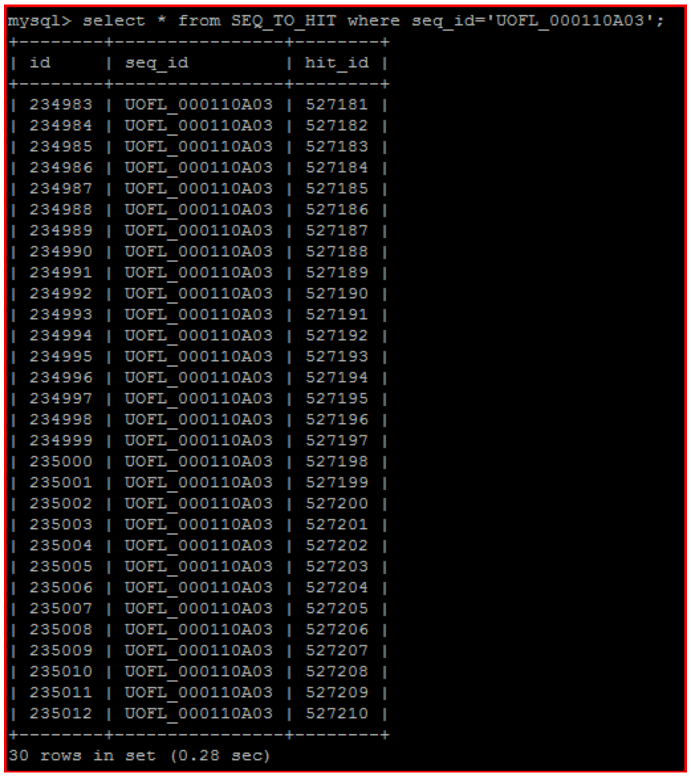


```
mysql> select ID,start,length(fullSeq), length(trimmedSeq), length(cap_trimmed)contig_id,trim_start,trim_end,orientation from SEQUENCE_TABLE where ID='UOFL_0
00110A01';
+----------------+-------+-----------------+--------------------+-----------+------------+----------+-------------+
| ID             | start | length(fullSeq) | length(trimmedSeq) | contig_id | trim_start | trim_end | orientation |
+----------------+-------+-----------------+--------------------+-----------+------------+----------+-------------+
| UOFL_000110A01 | 347   | 740             | 691                | 697       | 110        | 717      | Plus        |
+----------------+-------+-----------------+--------------------+-----------+------------+----------+-------------+
1 row in set (0.00 sec)
```

FIGURE 10: SEQUENCE_TABLE QUERY

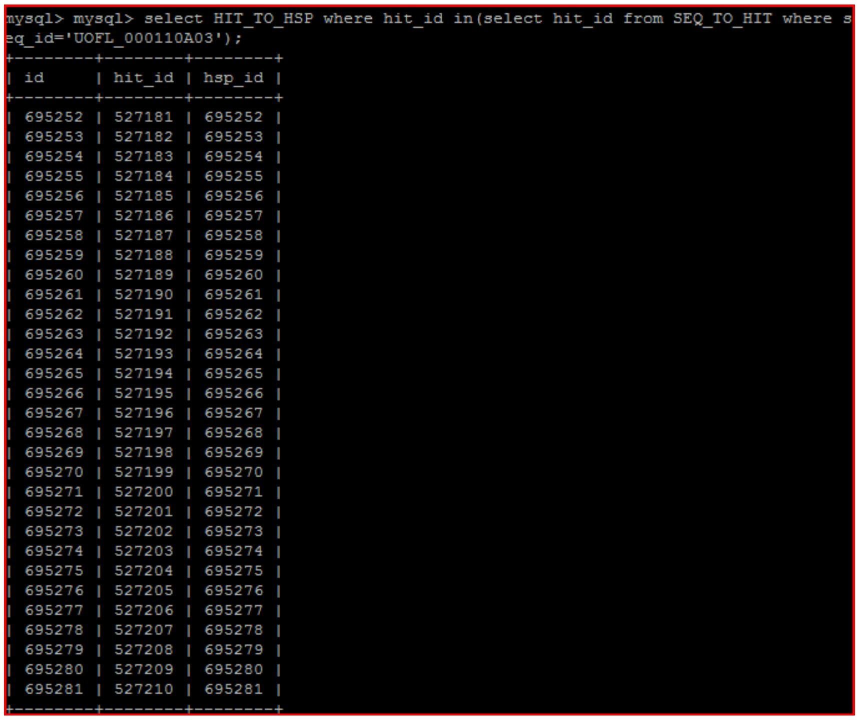
A FASTA file containing all of the individual sequences as well as the contig sequences was compiled. This file was BLASTed against the non-redundant database and the file was analyzed. The results were then used to populate the NCBI_HITS table, the HSP_TAB, and the linking tables that associated sequences with hits and hits with high scoring pairs. Queries to view the BLAST results for a sequence are displayed in Figures 11, 12, and 13.

The BLAST results were used for the subsequent task of using BLAST2GO. BLAST2GO runs in a user interface that takes the BLAST results as input and determines, based on the proteins that match each sequence and the GO annotation of the matching protein, what GO IDs should be attached to the sequence. The information from the GO database was then entered into the Trichome database. At the time this was done, 27,010 GO terms were present in the GO database. These were stored in the Trichome database along with the information about the “is a” and “part of” relationships. 11,963 terms were assigned to the sequences in the database. These mapped to 1,913 of the sequences. Not all of the sequences obtained GO annotations due to the fact that some of the sequences did not have significant BLAST hits and some of the matching proteins were not present in the GO database.



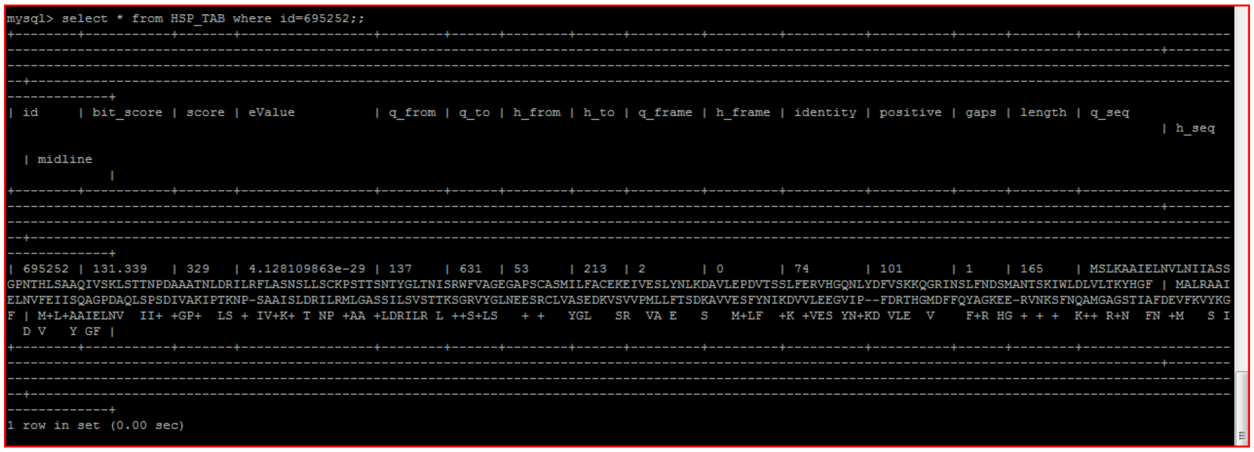
```
mysql> select * from SEQ_TO_HIT where seq_id='UOFL_000110A03';
+--------+----------------+--------+
| id     | seq_id         | hit_id |
+--------+----------------+--------+
| 234983 | UOFL_000110A03 | 527181 |
| 234984 | UOFL_000110A03 | 527182 |
| 234985 | UOFL_000110A03 | 527183 |
| 234986 | UOFL_000110A03 | 527184 |
| 234987 | UOFL_000110A03 | 527185 |
| 234988 | UOFL_000110A03 | 527186 |
| 234989 | UOFL_000110A03 | 527187 |
| 234990 | UOFL_000110A03 | 527188 |
| 234991 | UOFL_000110A03 | 527189 |
| 234992 | UOFL_000110A03 | 527190 |
| 234993 | UOFL_000110A03 | 527191 |
| 234994 | UOFL_000110A03 | 527192 |
| 234995 | UOFL_000110A03 | 527193 |
| 234996 | UOFL_000110A03 | 527194 |
| 234997 | UOFL_000110A03 | 527195 |
| 234998 | UOFL_000110A03 | 527196 |
| 234999 | UOFL_000110A03 | 527197 |
| 235000 | UOFL_000110A03 | 527198 |
| 235001 | UOFL_000110A03 | 527199 |
| 235002 | UOFL_000110A03 | 527200 |
| 235003 | UOFL_000110A03 | 527201 |
| 235004 | UOFL_000110A03 | 527202 |
| 235005 | UOFL_000110A03 | 527203 |
| 235006 | UOFL_000110A03 | 527204 |
| 235007 | UOFL_000110A03 | 527205 |
| 235008 | UOFL_000110A03 | 527206 |
| 235009 | UOFL_000110A03 | 527207 |
| 235010 | UOFL_000110A03 | 527208 |
| 235011 | UOFL_000110A03 | 527209 |
| 235012 | UOFL_000110A03 | 527210 |
+--------+----------------+--------+
30 rows in set (0.28 sec)
```

FIGURE 11: SEQ_TO_HIT QUERY



```
mysql> mysql> select HIT_TO_HSP where hit_id in(select hit_id from SEQ_TO_HIT where s
eq_id='UOFL_000110A03');
+--------+--------+--------+
| id     | hit_id | hsp_id |
+--------+--------+--------+
| 695252 | 527181 | 695252 |
| 695253 | 527182 | 695253 |
| 695254 | 527183 | 695254 |
| 695255 | 527184 | 695255 |
| 695256 | 527185 | 695256 |
| 695257 | 527186 | 695257 |
| 695258 | 527187 | 695258 |
| 695259 | 527188 | 695259 |
| 695260 | 527189 | 695260 |
| 695261 | 527190 | 695261 |
| 695262 | 527191 | 695262 |
| 695263 | 527192 | 695263 |
| 695264 | 527193 | 695264 |
| 695265 | 527194 | 695265 |
| 695266 | 527195 | 695266 |
| 695267 | 527196 | 695267 |
| 695268 | 527197 | 695268 |
| 695269 | 527198 | 695269 |
| 695270 | 527199 | 695270 |
| 695271 | 527200 | 695271 |
| 695272 | 527201 | 695272 |
| 695273 | 527202 | 695273 |
| 695274 | 527203 | 695274 |
| 695275 | 527204 | 695275 |
| 695276 | 527205 | 695276 |
| 695277 | 527206 | 695277 |
| 695278 | 527207 | 695278 |
| 695279 | 527208 | 695279 |
| 695280 | 527209 | 695280 |
| 695281 | 527210 | 695281 |
+--------+--------+--------+
```

FIGURE 12: HIT_TO_HSP QUERY



```
mysql> select * from HSP_TAB where id=695252;;
+--------+-----------+-------+-----------------+--------+------+--------+------+---------+---------+----------+----------+------+--------+------------------
----------------------------------------------------------------------------------------------------------------------------------------------+--------
----------------------------------------------------------------------------------------------------------------------------------------------------------
--+-------------------------------------------------------------------------------------------------------------------------------------------------------
------------+
| id     | bit_score | score | eValue          | q_from | q_to | h_from | h_to | q_frame | h_frame | identity | positive | gaps | length | q_seq
                                                                                                                                              | h_seq

  | midline
           |
+--------+-----------+-------+-----------------+--------+------+--------+------+---------+---------+----------+----------+------+--------+------------------
----------------------------------------------------------------------------------------------------------------------------------------------+--------
----------------------------------------------------------------------------------------------------------------------------------------------------------
--+-------------------------------------------------------------------------------------------------------------------------------------------------------
------------+
| 695252 | 131.339   | 329   | 4.128109863e-29 | 137    | 631  | 53     | 213  | 2       | 0       | 74       | 101      | 1    | 165    | MSLKAAIELNVLNIIASS
GPNTHLSAAQIVSKLSTTNPDAAATNLDRILRFLASNSLLSCKPSTTSNTYGLTNISRWFVAGEGAPSCASMILFACEKEIVESLYNLKDAVLEPDVTSSLFERVHGQNLYDFVSKKQGRINSLFNDSMANTSKIWLDLVLTKYHGF | MALRAAI
ELNVFEIISQAGPDAQLSPSDIVAKIPTKNP-SAAISLDRILRMLGASSILSVSTTKSGRVYGLNEESRCLVASEDKVSVVPMLLFTSDKAVVESFYNIKDVVLEEGVIP--FDRTHGMDFFQYAGKEE-RVNKSFNQAMGAGSTIAFDEVFKVYKG
F | M+L+AAIELNV  II+ +GP+  LS + IV+K+ T NP +AA +LDRILR L ++S+LS   + +   YGL   SR  VA E   S   M+LF  +K +VES YN+KD VLE  V    F+R HG + + +  K++ R+N  FN +M   S I
  D V   Y GF |
+--------+-----------+-------+-----------------+--------+------+--------+------+---------+---------+----------+----------+------+--------+------------------
----------------------------------------------------------------------------------------------------------------------------------------------+--------
----------------------------------------------------------------------------------------------------------------------------------------------------------
--+-------------------------------------------------------------------------------------------------------------------------------------------------------
------------+
1 row in set (0.00 sec)
```

FIGURE 13: HSP_TAB QUERY

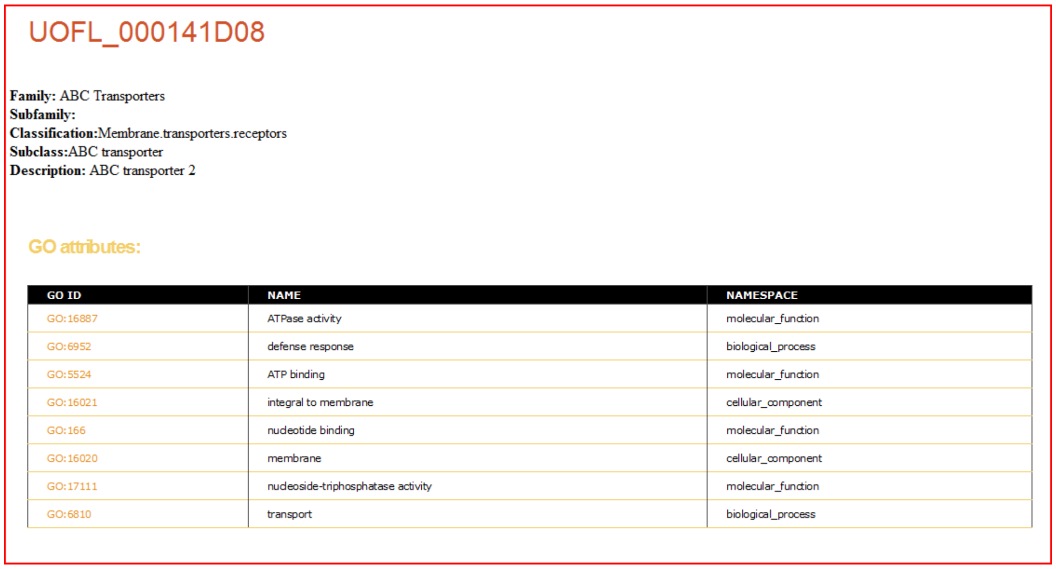
B. Database Browser

The database browser is divided into five pathways to get ultimately reach the sequence data. Sequence data can be found on each of the following:

- Name: Either the name of the sequence or a contig. The name can be searched for by the entire name or a substring within the name. For example, searching for "Contig1" will return a list of contigs including "Contig1", "Contig10", and "Contig11".

- Keyword: This searches the description in the SEQUENCE_TABLE for matching substrings based on the inputted string.
- BLAST: By inputting a sequence or sequence file, BLAST is run against all of the sequences in the Trichome database and the results are displayed.
- GO ID: The GO terms can be searched, either by GO ID or keyword in the description, and the information about the matching GO terms is displayed. This includes any sequences annotated with that GO ID.

The result of any of these searches leads to the sequence page, which displays the pertinent information about the sequence. This page can be displayed in two modes, regular or administrative. In regular mode, the mode for most users to view the page, all data is static and cannot be changed. In administrative mode, however, all fields are editable and all BLAST hits and GO terms can be removed as annotations for that sequence. Examples of the sequence display, in both regular and administrative mode, are shown in Figures 14 and 15. In addition, examples of all pages can be found in Appendix VI.

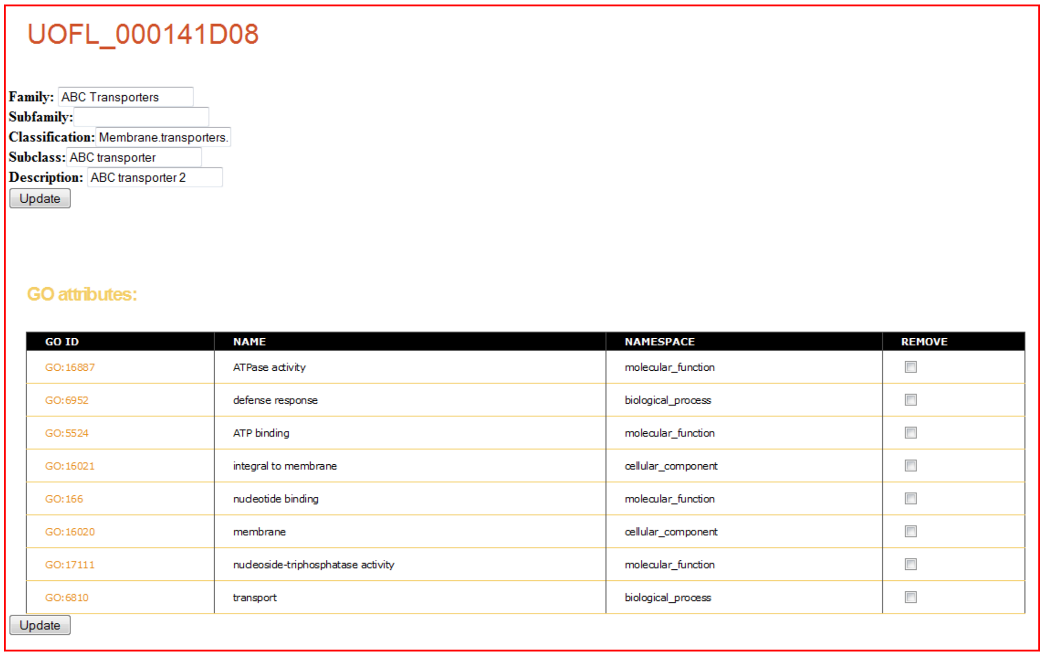

UOFL_000141D08

Family: ABC Transporters
Subfamily:
Classification: Membrane.transporters.receptors
Subclass:ABC transporter
Description: ABC transporter 2

GO attributes:

GO ID	NAME	NAMESPACE
GO:16887	ATPase activity	molecular_function
GO:6952	defense response	biological_process
GO:5524	ATP binding	molecular_function
GO:16021	integral to membrane	cellular_component
GO:166	nucleotide binding	molecular_function
GO:16020	membrane	cellular_component
GO:17111	nucleoside-triphosphatase activity	molecular_function
GO:6810	transport	biological_process

FIGURE 14: NORMAL SEQUENCE VIEW


UOFL_000141D08

Family: ABC Transporters
Subfamily:
Classification: Membrane.transporters.
Subclass: ABC transporter
Description: ABC transporter 2
Update

GO attributes:

GO ID	NAME	NAMESPACE	REMOVE
GO:16887	ATPase activity	molecular_function	☐
GO:6952	defense response	biological_process	☐
GO:5524	ATP binding	molecular_function	☐
GO:16021	integral to membrane	cellular_component	☐
GO:166	nucleotide binding	molecular_function	☐
GO:16020	membrane	cellular_component	☐
GO:17111	nucleoside-triphosphatase activity	molecular_function	☐
GO:6810	transport	biological_process	☐

Update

FIGURE 15: ADMIN SEQUENCE VIEW

V. FUTURE DIRECTION

There are a number of potential future extensions of this project. First, additional information about *P.* x*hotorum* or other organisms can be added to the database. Second, systematic issues with the data and its analysis can be improved to allow for additional information in the sequence browser, notably in the applet. Third, the various processes that were used for this analysis can be incorporated into a user interface to allow for greater ease of adding additional data. Finally, an additional administrative functionality can added to submit the data to NCBI for incorporation into dbEST.

The amount of EST data that is not currently incorporated into a database is vast. For any experiment that generates EST data, thousands of sequences are generated and need analysis, especially in a case such as *P.* x*hotorum* where the whole genome has not been sequenced. In addition to being a resource for the researchers at the University of Louisville, additional annotated plant ESTs from institutions such as the Noble Foundation (http://www.noble.org/) can add to the breadth of the database.

In the initial implementation of the database browser, it was intended to add notations in the applet to show where sequences have been trimmed. This created a systematic problem that was not realized until the end of the project. The sequences, as used throughout the project, were obtained from the sequencing machine used to generate the trace files. However, the positions in the applet where these bases occur were obtained from phred, which does an independent base calling. In most cases, the two sequences were identical. However, in cases where the two sequences were discordant, it was impossible to determine where in the trace file the trimming occurred. Therefore, it is suggested that the initial step in analysis should be to run phred on all trace files and use the results as analysis. Using two different base callers, as was done here, results in data that does not always agree.

This project includes several steps that were run independently of each other. For development purposes, this was acceptable as each step required a trial and error approach to find the best solution. However, in order to be of the greatest use, packaging all processes into one application would improve the usability for the researcher. By incorporating all steps into one larger interface and allowing the researcher to specify the location of all input files, the data could be analyzed, annotation, and added to the database.

Finally, this database is only meant for temporary storage of EST data. It is assumed that the eventual destination of most data is for public dissemination in a database such as dbEST at NCBI. The guidelines for submission to dbEST require specific formatting and a system such as the one described here would aid the researcher in ultimately adding data to a publicly available database.

REFERENCES

Central dogma reversed (1970). *Nature, 226*, 1198-1199.

VecScreen (2005). NCBI [On-line]. Available: http://www.ncbi.nlm.nih.gov/VecScreen/VecScreen_docs.html

EMVEC (2009). EMBL-EBI [On-line]. Available: http://www.ebi.ac.uk/Tools/blastall/vectors.html

Adams, M. D., Kelley, J. M., Gocayne, J. D., Dubnick, M., Polymeropoulos, M. H., Xiao, H. et al. (1991). Complementary DNA sequencing: expressed sequence tags and human genome project. *Science, 252*, 1651-1656.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. *J.Mol.Biol., 215*, 403-410.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M. et al. (2000). Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat.Genet., 25*, 25-29.

Chenna, R., Sugawara, H., Koike, T., Lopez, R., Gibson, T. J., Higgins, D. G. et al. (2003). Multiple sequence alignment with the Clustal series of programs. *Nucleic Acids Res., 31*, 3497-3500.

Conesa, A. & Gotz, S. (2008). Blast2GO: A Comprehensive Suite for Functional Analysis in Plant Genomics. *Int.J.Plant Genomics, 2008,* 619832.

Conesa, A., Gotz, S., Garcia-Gomez, J. M., Terol, J., Talon, M., & Robles, M. (2005). Blast2GO: a universal tool for annotation, visualization and analysis in functional genomics research. *Bioinformatics., 21,* 3674-3676.

Crick, F. (1970). Central dogma of molecular biology. *Nature, 227,* 561-563.

DuBois, P. (2005). *MySQL*

the definitive guide to using, programming, and administering MySQL 4.1 and 5.0. (3rd ed ed.) Indianapolis, Ind: Sams Pub.

Ewing, B., Hillier, L., Wendl, M. C., & Green, P. (1998). Base-calling of automated sequencer traces using phred. I. Accuracy assessment. *Genome Res., 8,* 175-185.

Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures.* University of California, Irvine.

Hamilton, A. J. & Baulcombe, D. C. (1999). A species of small antisense RNA in posttranscriptional gene silencing in plants. *Science, 286,* 950-952.

Huang, X. & Madan, A. (1999). CAP3: A DNA sequence assembly program. *Genome Res., 9,* 868-877.

Kae, H. (2003). GENOME PROJECTS: UNCOVERING THE BLUEPRINTS OF BIOLOGY. scq.ubc.ca [On-line]. Available: http://www.scq.ubc.ca/genome-projects-uncovering-the-blueprints-of-biology/

Notredame, C. & Suhre, K. (2004). Computing multiple sequence/structure alignments with the T-coffee package. *Curr.Protoc.Bioinformatics., Chapter 3,* Unit3.

Sanger, F., Nicklen, S., & Coulson, A. R. (1977). DNA sequencing with chain-terminating inhibitors. *Proc.Natl.Acad.Sci.U.S.A, 74,* 5463-5467.

Schilmiller, A. L., Last, R. L., & Pichersky, E. (2008). Harnessing plant trichome biochemistry for the production of useful compounds. *Plant J., 54,* 702-711.

Ussery, D. (2003). Bioinformatics Lekture Notes. David Ussery [On-line]. Available: http://protfun.com/staff/dave/MScourse/Lekt_11Feb2003a.html

APPENDIX I: BLAST CODE EXAMPLE

Blast.java

```
package web;

import java.io.FileNotFoundException;
import java.sql.ResultSet;
import edu.louisville.bioinformatics.utilities.ExternalCmdExec;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.Random;
import javax.servlet.ServletException;
import javax.servlet.ServletInputStream;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import objects.BlastParser;
import objects.BlastResults;
import objects.FileUploadBean;
import objects.Hit;
import objects.Iteration;
import org.xml.sax.SAXException;

public class Blast extends ResourceServlet {

    public void index(HttpServletRequest request, HttpServletResponse response) throws
IOException, InterruptedException, SAXException, ServletException {
        Random r = new Random();
        long id = r.nextLong();
        String filename = "query.fas";
        if (request.getParameter("sequence") != null) {
            createBlastFile(request.getParameter("sequence"), id);
        } else {
            filename = processRequest(request, response, id);
        }
        System.out.println("BLASTing file");
        blastFile(id, filename);
        System.out.println("BLASTed");
        BlastResults results = getBlastResults(id);
        System.out.println("Got results");
        request.setAttribute("results", results);
        render("blast_results.jsp", request, response);
    }

    public void createBlastFile(String sequence, long connectionId) throws IOException {
        BufferedWriter out = new BufferedWriter(new FileWriter(new File("/tmp/EST/" +
connectionId + "query.fas")));
        out.write(sequence);
        out.close();
    }

    protected String processRequest(HttpServletRequest request,
            HttpServletResponse response, long id)
            throws ServletException, IOException {
        FileUploadBean upLoader = new FileUploadBean();
        upLoader.setSavePath("/tmp/EST/" + id);
        upLoader.doUpload(request);
        return (upLoader.getFilename());
    }

    public void database(HttpServletRequest request, HttpServletResponse response)
throws SQLException, IOException, ServletException {
        Connection conn = (Connection) request.getSession().getAttribute("connection");

        String sequence = request.getParameter("sequenceId");
        String hitId = request.getParameter("hitId");
        BlastResults blastResults = getDatabaseBlastResults(conn, sequence, hitId);
        request.setAttribute("results", blastResults);
        render("blast_results.jsp",request,response);
    }

    public void blastFile(long connectionId, String filename) throws IOException,
InterruptedException {
        String infile = "/tmp/EST/" + connectionId + filename;
        String cmd = "megablast -F F -r 1 -q -2 -m 7" +
                " -i " + infile + " -d /tmp/EST/formattedDatabase -o " + "/tmp/EST/" +
connectionId + "results.txt";
        String[] execCmd = {"/bin/sh", "-c", cmd};
        System.out.println(cmd);
        ExternalCmdExec externalCmdExec = new ExternalCmdExec(execCmd, new
String[0], "stdOut.txt", "stdErr.txt");
        int exitCode = externalCmdExec.exec();
    }
    public BlastResults getBlastResults(long connectionId) throws SAXException,
IOException, SAXException {
        BlastParser parser = new BlastParser("/tmp/EST/" + connectionId + "results.txt");
        BlastResults results = parser.getResults();
        return results;
    }

    public BlastResults getDatabaseBlastResults(Connection conn, String sequenceId,
String hitId) throws SQLException {

        String defQuery = "Select * from NCBI_HITS where db_id=" + hitId;
        String hitDef = null;
        System.out.println(defQuery);
        try {
            java.sql.Statement stmt = conn.createStatement();
            ResultSet result = stmt.executeQuery(defQuery);
            while (result.next()) {
                hitDef = result.getString("hit_def");
            }
            result.close();
            stmt.close();

        } catch (SQLException ex) {
            ex.printStackTrace();
        }

        String query = "select * from HSP_TAB ht, HIT_TO_HSP hh where
ht.id=hh.hsp_id and hit_id=" + hitId;
        System.out.println(query);

        BlastResults returnResults = new BlastResults();
        try {
            java.sql.Statement stmt = conn.createStatement();
            ResultSet result = stmt.executeQuery(query);
            while (result.next()) {
                Hit h = new Hit();

                h.setAlignmentLength(result.getInt("length"));
                h.setHitDef(hitDef);
                h.setHitFrame(result.getInt("h_frame"));
                h.setHitFrom(result.getInt("h_from"));
                h.setHitLength(result.getInt("length"));
                h.setHitSeq(result.getString("h_seq"));
                h.setHitTo(result.getInt("h_to"));
                h.setIdentity(result.getInt("identity"));
                h.setMidline(result.getString("midline"));
                h.setPositive(result.getInt("positive"));
                h.setQueryFrame(result.getInt("q_frame"));
                h.setQueryFrom(result.getInt("q_from"));
                h.setQuerySeq(result.getString("q_seq"));
                h.setQueryTo(result.getInt("q_to"));
                h.setScore(result.getInt("score"));
                h.setEValue(result.getString("eValue"));
                Iteration newIteration = new Iteration();
                newIteration.setQueryDef(sequenceId);
                newIteration.addHit(h);
                returnResults.addIteration(newIteration);

            }
            result.close();
            stmt.close();

        } catch (SQLException ex) {
            ex.printStackTrace();
        }


        return returnResults;
    }
}
```

blast.jsp

```
<%@ include file="_top.jsp" %>

<h2>BLAST the Trichome database</h2>

<h3>Enter a sequence:</h3>
<form id='frm'
      action="<%=request.getContextPath()%>/blast/">
    <textarea cols="40" rows="5" name="sequence">
    </textarea>
    <input type="submit"/>
    <input name="type" type="hidden" value="blast"/>
</form>

<h3>Enter a FASTA file:</h3>
<form id='frm'
      action="<%=request.getContextPath()%>/blast/"
      enctype="multipart/form-data"
      method="post">
    <INPUT TYPE="file" NAME="fastaFile"/>
    <INPUT TYPE="submit" VALUE="Upload File">

</form>
<%@ include file="_bottom.jsp" %>
```

blast_results.jsp

```
<%@ include file="_top.jsp" %>
<%@include file="_blast_results.jsp" %>
<%@ include file="_bottom.jsp" %>
```

_blast_results.jsp

```
<%@ page import="objects.*" %>
<%@ page import="java.util.StringTokenizer" %>


 <%
        BlastResults blastResults=(BlastResults)request.getAttribute("results");
%>

    <%
        Iteration[] iterations = blastResults.getIterations();

        for (int i = 0; i < iterations.length; i++) {
            Hit[] hits = iterations[i].getHits();
    %>


    <br>
    <br>
    <%

        for (int j = 0; j < hits.length; j++) {
            String queryDef = hits[j].getHitDef();
            String link;
            if (queryDef.startsWith("Contig")) {
                StringTokenizer tokenizer = new StringTokenizer(queryDef, "|");
                link = request.getContextPath()+"/sequence/contig/"+tokenizer.nextToken();
            } else if(queryDef.startsWith("UOFL")){
                StringTokenizer tokenizer = new StringTokenizer(queryDef, "|");
                link = request.getContextPath()+"/sequence/index/" + tokenizer.nextToken();
            }else{
              StringTokenizer tokenizer = new StringTokenizer(queryDef, "|");
              tokenizer.nextToken();
              link="http://www.ncbi.nlm.nih.gov/protein/"+tokenizer.nextToken();
            }

        %>
        <a href=<%=link%> target="_blank"><%=hits[j].getHitDef()%></a>
        <br>
        <%
            }
          }
        %>
        <font face="Courier">
          <% for (int i = 0; i < iterations.length; i++) {
            Hit[] hits = iterations[i].getHits();
          %>
          Results for Query Sequence: <%=iterations[i].getQueryDef()%><br>
          <br>
          <%for (int j = 0; j < hits.length; j++) {%>
          Hit ID:<%=hits[j].getHitDef()%><br>
          Score:<%=hits[j].getScore()%><br>
          Expect:<%=hits[j].getEValue()%><br>
          Identities:<%=hits[j].getIdentity()%>/<%=hits[j].getAlignmentLength()%><br>
          <br>
          <br>
          <table class="sortable" border="0">
            <%
      String querySeq = hits[j].getQuerySeq();
      String hitSeq = hits[j].getHitSeq();
      String midline = hits[j].getMidline();
      for (int k = 0; k < querySeq.length(); k += 60) {
        int startPos = k;
        int endPos = k + 60;
        if (endPos > querySeq.length()) {
          endPos = querySeq.length();
        }

            %>

            <tr><td>Query</td><td>  <%=hits[j].getQueryFrom() + startPos%>
    </td><td><%=querySeq.substring(startPos,
    endPos)%>&nbsp;&nbsp;<%=hits[j].getQueryFrom() + endPos - 1%></td></tr>
            <tr><td><td/><td>  <%=midline.substring(startPos, endPos)%></td></tr>
```

```
		<tr><td>Sbjct</td><td>   <%=hits[j].getHitFrom() + startPos%>
</td><td><%=hitSeq.substring(startPos,
endPos)%>&nbsp;&nbsp;<%=hits[j].getHitFrom() + endPos - 1%></td></tr>
		<tr><td/><td/><td/></tr>
		<tr><td/><td/><td/></tr>
		<tr><td/><td/><td/></tr>

		<%}%>
	    </table>
	    <%}
	    }%>
	</font>
```

APPENDIX II: QUALITY SCORES

```
>Plate1_A01.qual
31 35 39 39 33 33 26 28 28 28 30 30 36 36 36 57 42 42 39 34 34 37 42 42 39 39 39 43 43
36 36 32 30 36 36 43 43 43 43 48 48 46 32 29 12 14 14 14 27 30 42 46 39 37 37 39 39 41
44 57 44 47 47 57 57 57 54 54 54 44 32 27 20 27 27 27 27 28 36 40 46 47 50 50 57 57 47
57 57 50 44 43 32 32 44 44 50 43 35 32 32 43 36 43 37 35 35 44 32 57 37 37 35 39 39 48
42 42 31 35 35 35 35 35 36 42 42 37 37 32 32 47 43 44 32 32 39 35 35 44 44 57 57 57 46
37 37 43 48 57 68 68 68 68 68 68 68 68 54 54 48 48 48 48 48 57 57 46 42 39 35 22 24 24
16 22 22 28 28 32 32 30 32 32 39 37 28 33 28 32 33 34 28 24 18 18 24 26 28 28 28 32 37
37 46 46 43 43 39 35 35 32 37 37 32 32 20 15 15 19 21 28 30 29 24 31 31 31 33 30 30 32
32 32 36 36 28 22 24 27 24 30 39 30 41 20 24 20 32 32 43 36 36 30 32 24 24 22 25 32 32
43 57 47 54 47 57 44 43 32 29 28 30 26 17 15 15 21 16 28 21 22 18 20 28 32 28 18 18 10
10 15 20 25 25 19 22 22 24 12 12 12 20 20 21 28 28 20 19 12 14 14 19 30 33 37 24 24 19
15 11 11 14 14 21 21 19 19 18 18 12 16 12 29 29 28 17 13 10 10 10 10 10 8 13 12 9 9 12
11 16 16 16 17 19 14 10 10 10 19 19 22 22 20 15 15 14 17 17 10 10 10 13 13 19 23 25 18
15 13 12 9 9 14 19 12 10 15 11 17 17 11 8 8 8 8 8 10 10 9 9 9 12 20 23 23 23 25 15 12 7
7 7 7 10 12 8 8 8 8 8 13 13 13 12 9 9 9 12 12 18 22 25 22 18 15 16 13 16 13 10 8 8 9 11 9
8 8 6 6 6 8 8 8 8 8 15 15 20 17 14 10 10 10 9 12 11 12 9 11 11 11 10 12 12 9 9 8 8 8 8 8 8
9 9 9 8 8 8 8 8 8 10 10 14 10 9 8 11 9 9 9 9 13 13 13 15 12 10 10 11 16 17 11 10 9 9 10
10 9 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 10 8 8 8 8 9 9 9 9 10 10 9 8 8 10 10 8 8 10 10 10
10 10 12 10 10 10 8 8 8 8 8 8 10 10 8 8 8 8 8 9 10 10 10 8 8 10 7 9 12 12 9 9 9 12 13 10
10 11 11 10 10 10 10 10 10 10 10 8 9 9 8 9 9 10 10 10 10 15 15 8 8 8 8 9 10 8 8 8 8 8 8 8
8 9 10 9 9 9 9 9 8 8 8 10 10 10 11 6 8 6 6 13 15 9 13 13 13 12 9 8 8 8 8 7 9 11 9 9 9 9 9 9
10 10 10 10 17 13 10 10 8 8 8 6 6 6 6 8 7 6 10 10 10 10 6 6 6 6 8 8 8 8 6 6 8 8 8 11 9 8 7
6 8 8 8 8 8 8 8 8 10 10 10 10 10 10 11 12 11 11 11
```

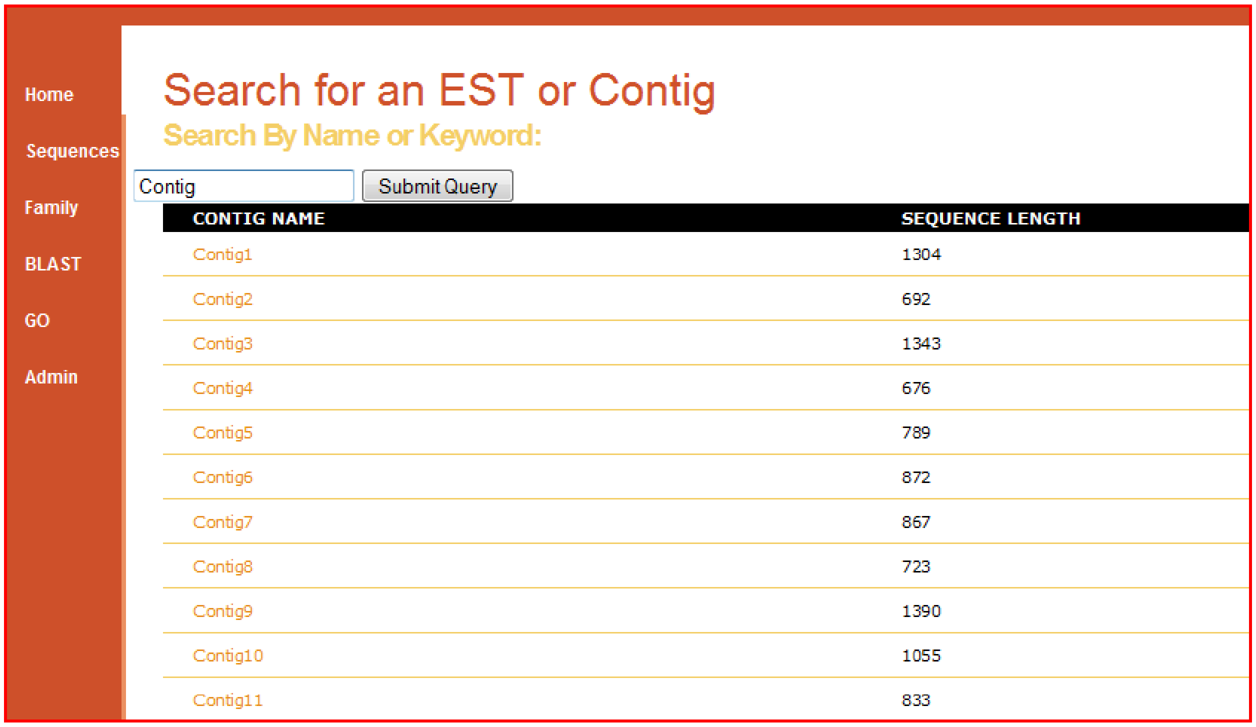
APPENDIX III: SEQUENCE FILE

```
 >Plate1_A01.seq (Untrimmed)   Agencourt Bioscience
Corporation    ABI
gaggagactgctgacccttgagacagagagacagagagagagataatgttgataacggtc
agagacccatcttcttgcttgagcttcaactccaactccagacacagacacatacacaga
cacagaccctccacctccatttgcagagccaccttttctcttccgaggaaacccaacacc
accaagccccgggaccttgccttttcctctaacaccaacaccagtaatgagagctccgtg
tttgatccattgggtatcaatccagatttatcttctagcttaaatggtacttgggaaagt
cttcaaggattgttttctcaaacctttgagagttcctcaggcccaaaaaaagagagaggg
tcttctgctcgcggggttgcacctgctattgaggatagtttaattgattttaggatttt
tttaaaggccaattaccaggaaaattcctccagctcctgggggtttttggccttatctct
actgcaaaatatataccttccgggtggggtaaatagggaggctattgttcgcattctgga
taaaactgtttttgatcccttgggttccttttctgggggagcattggtagctggggaat
tctccttgtattgccccttcataaggccaattggtttcaacttctgcaccatttttccca
attgtagacttcttaaaaatatgcgcaccagaaaagatatctctttatcaaaaagctcat
tggttgcaata
```

APPENDIX IV: PHRED FILE

```
BEGIN_SEQUENCE Plate1_A01.scf

BEGIN_COMMENT

CHROMAT_FILE: Plate1_A01.scf
ABI_THUMBPRINT: 0
PHRED_VERSION: 0.020425.c
CALL_METHOD: phred
QUALITY_LEVELS: 99
TIME: Sun Apr  5 19:56:26 2009
TRACE_ARRAY_MIN_INDEX: 0
TRACE_ARRAY_MAX_INDEX: 8716
TRIM: 0 336 0.0500
TRACE_PEAK_AREA_RATIO: 0.1435
CHEM: term
DYE: big

END_COMMENT

BEGIN_DNA
g 24 10
a 25 20
g 40 33
g 40 46
a 29 57
g 29 70
a 22 80
c 27 88
t 26 96
g 26 115
c 26 125
t 32 135
g 35 152
a 35 162
c 35 172
c 42 184
c 35 195
t 37 206
t 35 218
g 35 234
a 33 247
g 34 259
a 35 269
………….
```

APPENDIX V: SCREEN SHOTS



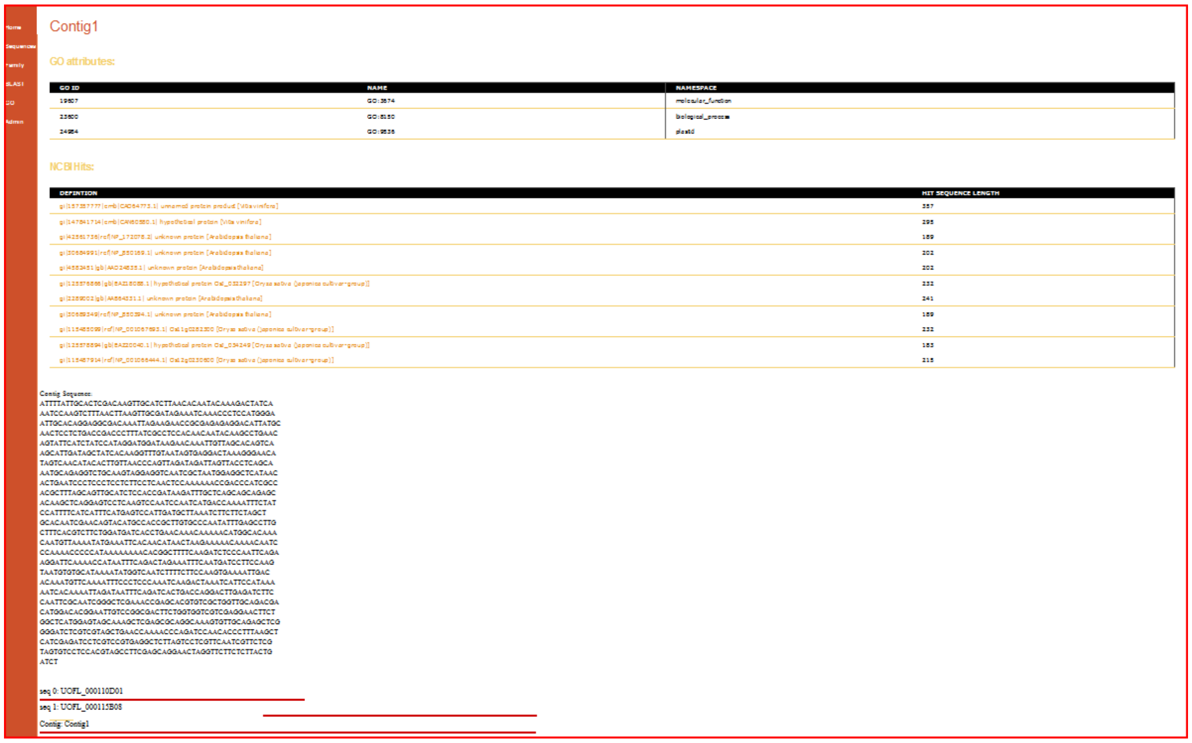
Search for an EST or Contig

Search By Name or Keyword:

Contig

CONTIG NAME	SEQUENCE LENGTH
Contig1	1304
Contig2	692
Contig3	1343
Contig4	676
Contig5	789
Contig6	872
Contig7	867
Contig8	723
Contig9	1390
Contig10	1055
Contig11	833

NAME.JSP



Contig1

GO attributes:

GO ID	NAME	NAMESPACE
19607	GO:3674	molecular_function
23600	GO:8150	biological_process
24984	GO:9536	plastid

NCBI Hits:

DEFINTION	HIT SEQUENCE LENGTH
gi|157357777|emb|CAO64773.1| unnamed protein product [Vitis vinifera]	357
gi|147841714|emb|CAN60580.1| hypothetical protein [Vitis vinifera]	295
gi|42561736|ref|NP_172078.2| unknown protein [Arabidopsis thaliana]	189
gi|30684991|ref|NP_850169.1| unknown protein [Arabidopsis thaliana]	202
gi|4582451|gb|AAD24835.1| unknown protein [Arabidopsis thaliana]	202
gi|125576866|gb|EAZ18088.1| hypothetical protein OsJ_032297 [Oryza sativa (japonica cultivar-group)]	232
gi|2289002|gb|AAB64331.1| unknown protein [Arabidopsis thaliana]	241
gi|30689349|ref|NP_850394.1| unknown protein [Arabidopsis thaliana]	189
gi|115485099|ref|NP_001067693.1| Os11g0282300 [Oryza sativa (japonica cultivar-group)]	232
gi|125578894|gb|EAZ20040.1| hypothetical protein OsJ_034249 [Oryza sativa (japonica cultivar-group)]	183
gi|115487914|ref|NP_001066444.1| Os12g0230600 [Oryza sativa (japonica cultivar-group)]	215

Contig Sequence:

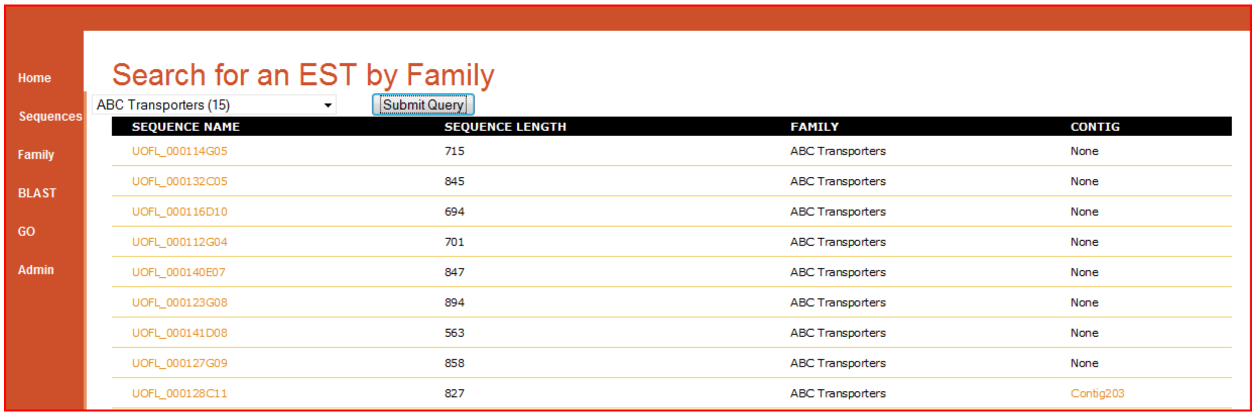
```
ATTTTATTGCACTCGACAAGTTGCATCTTAACACAATACAAAGACTATCA
AATCCAAGTCTTTAACTTAAGTTGCGATAGAAATCAAACCCTCCATGGGA
ATTGCACAGGAGGCGACAAATTAGAAGAACCGCGAGAGAGGACATTATGC
AACTCCTCTGACCGACCCTTTATCGCCTCCACAACAATACAAGCCTGAAC
AGTATTCATCTATCCATAGGATGGATAAGAACAAATTGTTAGCACAGTCA
AGCATTGATAGCTATCACAAGGTTTGTAATAGTGAGGACTAAAGGGAACA
TAGTCAACATACACTTGTTAACCCAGTTAGATAGATTAGTTACCTCAGCA
AATGCAGAGGTCTGCAAGTAGGAGGTCAATCGCTAATGGAGGCTCATAAC
ACTGAATCCCTCCCTCCTCTTCCTCAACTCCAAAAAACCGACCCATCGCC
ACGCTTTAGCAGTTGCATCTCCACCGATAAGATTTGCTCAGCAGCAGAGC
ACAAGCTCAGGAGTCCTCAAGTCCAATCCAATCATGACCAAAATTTCTAT
CCATTTTCATCATTTCATGAGTCCATTGATGCTTAAATCTTCTTCTAGCT
GCACAATCGAACAGTACATGCCACCGCTTGTGCCCAATATTTGAGCCTTG
CTTTCACGTCTTCTGGATGATCACCTGAACAAACAAAAACATGGCACAAA
CAATGTTAAAATATGAAATTCACAACATAACTAAGAAAAACAAAACAATC
CCAAAACCCCCATAAAAAAAACACGGCTTTTCAAGATCTCCCAATTCAGA
AGGATTCAAAACCATAATTTCAGACTAGAAATTTCAATGATCCTTCCAAG
TAATGTGTGCATAAAATATGGTCAATCTTTTCTTCCAAGTGAAAATTGAC
ACAAATGTTCAAAATTTCCCTCCCAAATCAAGACTAAATCATTCCATAAA
AATCACAAAATTAGATAATTTCAGATCACTGACCAGGACTTGAGATCTTC
CAATTCGCAATCGGGCTCGAAACCGAGCACGTGTCGCTGGTTGCAGACGA
CATGGACACGGAATTGTCCGGCGACTTCTGGTGGTCGTCGAGGAACTTCT
GGCTCATGGAGTAGCAAAGCTCGAGCGCAGGCAAAGTGTTGCAGAGCTCG
GGGATCTCGTCGTAGCTGAACCAAAACCCAGATCCAACACCCTTTAAGCT
CATCGAGATCCTCGTCCGTGAGGCTCTTAGTCCTCGTTCAATCGTTCTCG
TAGTGTCCTCCACGTAGCCTTCGAGCAGGAACTAGGTTCTTCTCTTACTG
ATCT
```

seq 0: UOFL_000110D01

seq 1: UOFL_000115B08

Contig: Contig1

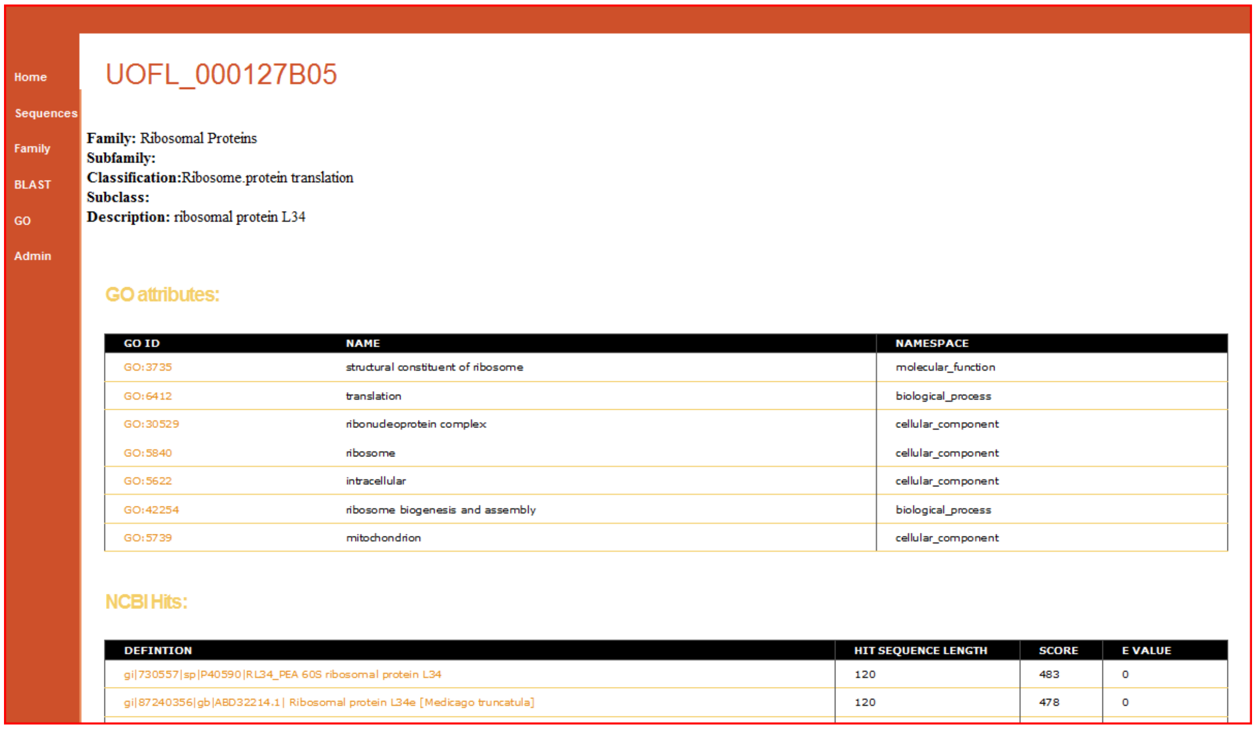
CONTIG.JSP



Search for an EST by Family

SEQUENCE NAME	SEQUENCE LENGTH	FAMILY	CONTIG
UOFL_000114G05	715	ABC Transporters	None
UOFL_000132C05	845	ABC Transporters	None
UOFL_000116D10	694	ABC Transporters	None
UOFL_000112G04	701	ABC Transporters	None
UOFL_000140E07	847	ABC Transporters	None
UOFL_000123G08	894	ABC Transporters	None
UOFL_000141D08	563	ABC Transporters	None
UOFL_000127G09	858	ABC Transporters	None
UOFL_000128C11	827	ABC Transporters	Contig203

FAMILY.JSP



UOFL_000127B05

Family: Ribosomal Proteins
Subfamily:
Classification: Ribosome.protein translation
Subclass:
Description: ribosomal protein L34

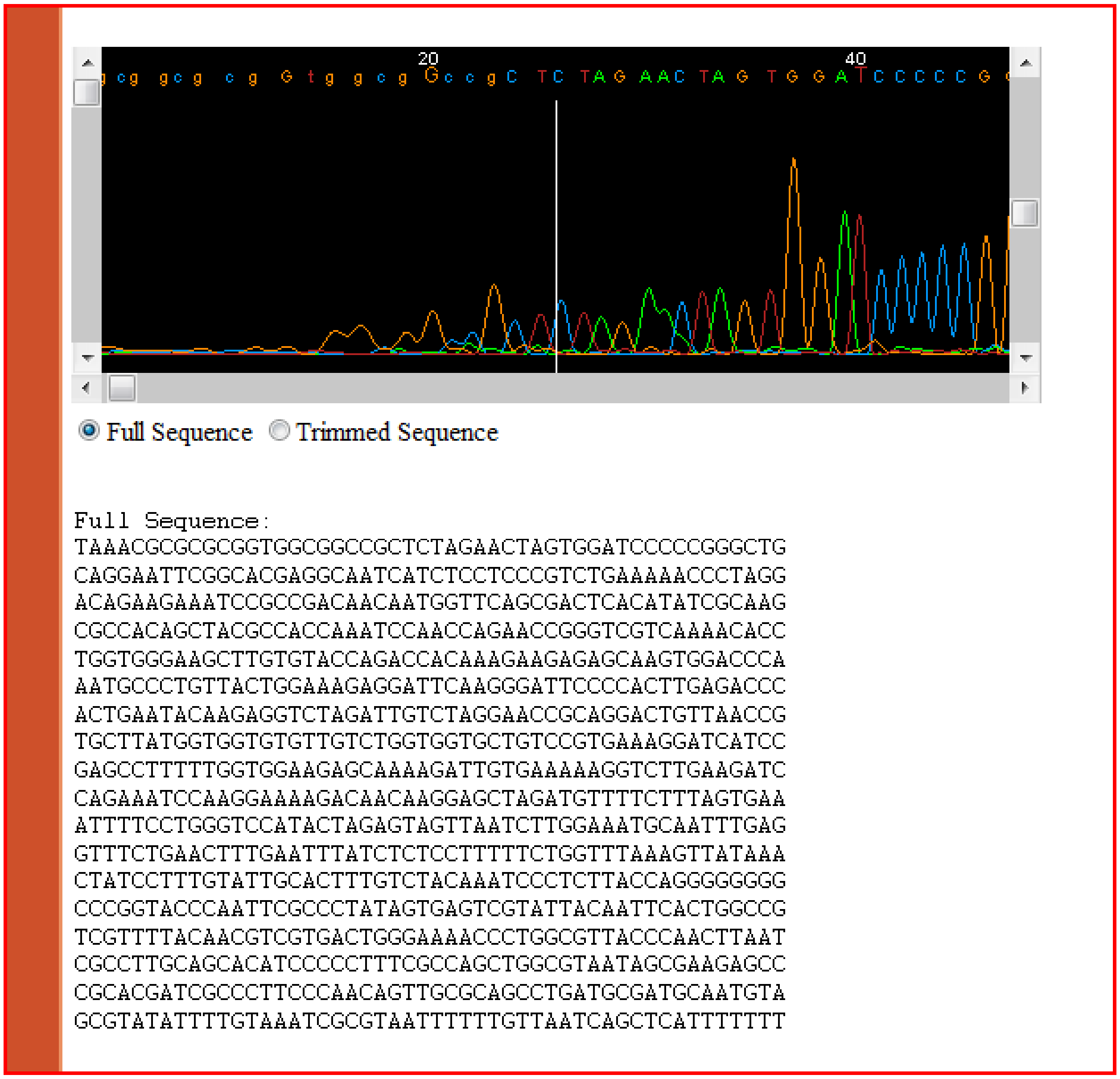
GO attributes:

GO ID	NAME	NAMESPACE
GO:3735	structural constituent of ribosome	molecular_function
GO:6412	translation	biological_process
GO:30529	ribonucleoprotein complex	cellular_component
GO:5840	ribosome	cellular_component
GO:5622	intracellular	cellular_component
GO:42254	ribosome biogenesis and assembly	biological_process
GO:5739	mitochondrion	cellular_component

NCBI Hits:

DEFINTION	HIT SEQUENCE LENGTH	SCORE	E VALUE
gi|730557|sp|P40590|RL34_PEA 60S ribosomal protein L34	120	483	0
gi|87240356|gb|ABD32214.1| Ribosomal protein L34e [Medicago truncatula]	120	478	0

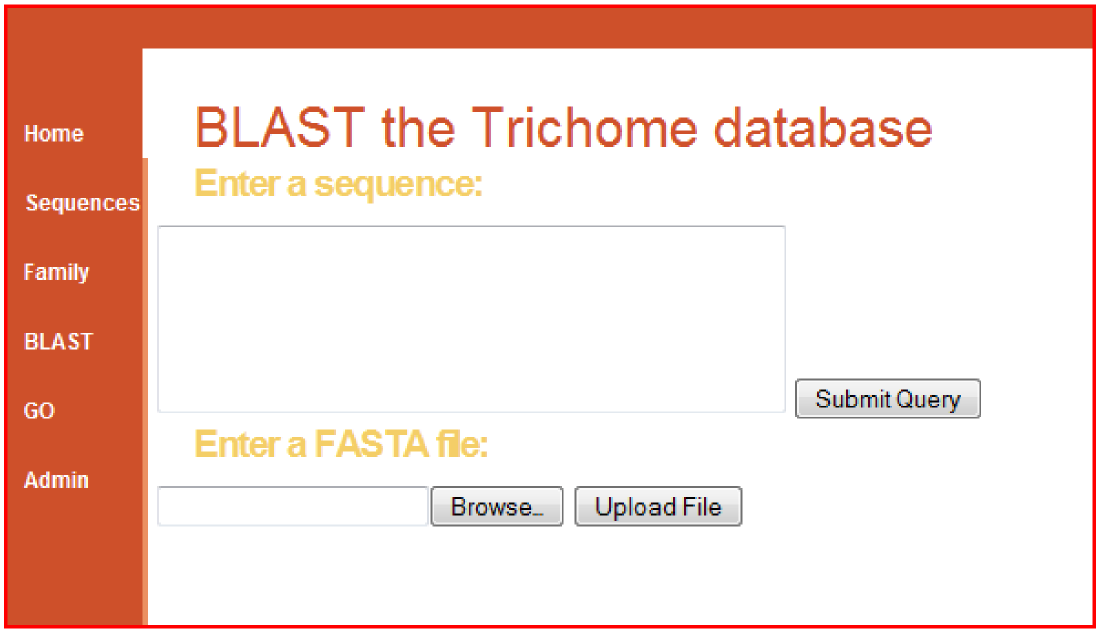
SEQUENCE.JSP (TOP)



[✓ Full Sequence] Trimmed Sequence

```
Full Sequence:
TAAACGCGCGCGGTGGCGGCCGCTCTAGAACTAGTGGATCCCCCGGGCTG
CAGGAATTCGGCACGAGGCAATCATCTCCTCCCGTCTGAAAAACCCTAGG
ACAGAAGAAATCCGCCGACAACAATGGTTCAGCGACTCACATATCGCAAG
CGCCACAGCTACGCCACCAAATCCAACCAGAACCGGGTCGTCAAAACACC
TGGTGGGAAGCTTGTGTACCAGACCACAAAGAAGAGAGCAAGTGGACCCA
AATGCCCTGTTACTGGAAAGAGGATTCAAGGGATTCCCCACTTGAGACCC
ACTGAATACAAGAGGTCTAGATTGTCTAGGAACCGCAGGACTGTTAACCG
TGCTTATGGTGGTGTGTTGTCTGGTGGTGCTGTCCGTGAAAGGATCATCC
GAGCCTTTTTGGTGGAAGAGCAAAAGATTGTGAAAAAGGTCTTGAAGATC
CAGAAATCCAAGGAAAAGACAACAAGGAGCTAGATGTTTTCTTTAGTGAA
ATTTTCCTGGGTCCATACTAGAGTAGTTAATCTTGGAAATGCAATTTGAG
GTTTCTGAACTTTGAATTTATCTCTCCTTTTTCTGGTTTAAAGTTATAAA
CTATCCTTTGTATTGCACTTTGTCTACAAATCCCTCTTACCAGGGGGGGG
CCCGGTACCCAATTCGCCCTATAGTGAGTCGTATTACAATTCACTGGCCG
TCGTTTTACAACGTCGTGACTGGGAAAACCCTGGCGTTACCCAACTTAAT
CGCCTTGCAGCACATCCCCCTTTCGCCAGCTGGCGTAATAGCGAAGAGCC
CGCACGATCGCCCTTCCCAACAGTTGCGCAGCCTGATGCGATGCAATGTA
GCGTATATTTTGTAAATCGCGTAATTTTTGTTAATCAGCTCATTTTTT
```

SEQUENCE.JSP (BOTTOM)

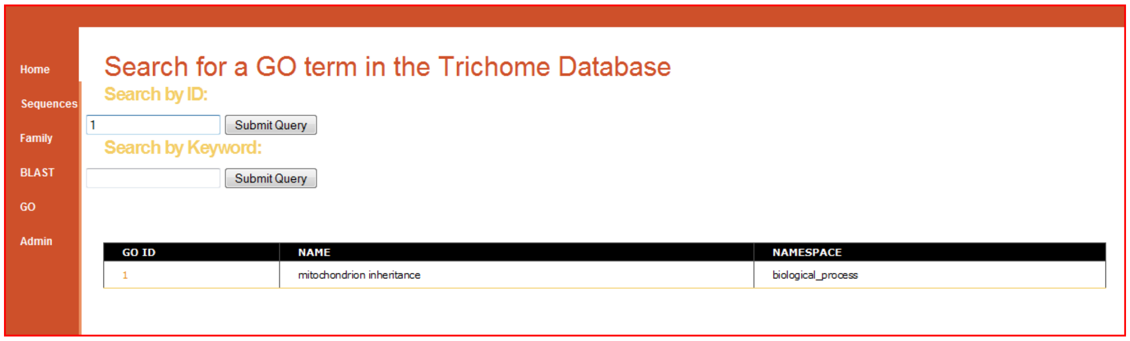


BLAST the Trichome database

Enter a sequence:

Enter a FASTA file:

BLAST.JSP



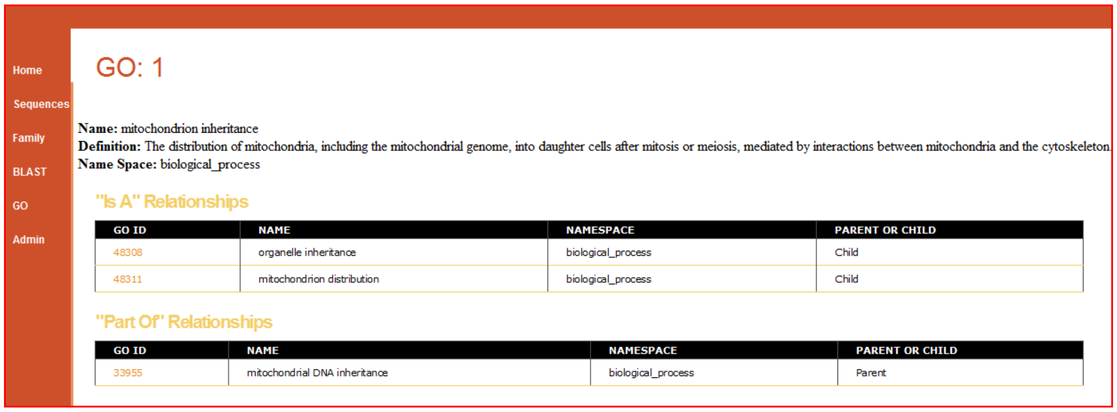
Search for a GO term in the Trichome Database

Search by ID: 1

Search by Keyword:

GO ID	NAME	NAMESPACE
1	mitochondrion inheritance	biological_process

GO_SEARCH.JSP



GO: 1

Name: mitochondrion inheritance
Definition: The distribution of mitochondria, including the mitochondrial genome, into daughter cells after mitosis or meiosis, mediated by interactions between mitochondria and the cytoskeleton.
Name Space: biological_process

"Is A" Relationships

GO ID	NAME	NAMESPACE	PARENT OR CHILD
48308	organelle inheritance	biological_process	Child
48311	mitochondrion distribution	biological_process	Child

"Part Of" Relationships

GO ID	NAME	NAMESPACE	PARENT OR CHILD
33955	mitochondrial DNA inheritance	biological_process	Parent

GO_RESULTS.JSP

APPENDIX VI: REMOVED VECTORS

```
>pBlueScript Vector
TTGTAAAACGACGGCCAGTGAATTGTAATACGACTCACTATAGGGCGAATTGGGTACCGGGCCCCCCCTCGAGGT
CGACGGTATCGATAAGCTTGATATCGAATTCCTGCAGCCCGGGGGATCCACTAGTTCTAGAGCGGCCGCCACCGC
GGTGGAGCTCCAGCTTTTGTTCCCTTTAGTGAGGGTTAATTTCGAGCTTGGCGTAATCATGGTCATAGCTGTTTC
>5' Adapter
AATTCGGCACGAGG
>3' Adapter
GCCGTGCTCC
>Strascript RT
GAGAGAGAGAGAGAGAGAGAACTAGTCTCGAGTTTTTTTTTTTTTTTTT
>gi|58066|emb|X52324.1| pBluescript SK(-) vector DNA, phagemid excised from
lambda ZAP
CACCTGACGCGCCCTGTAGCGGCGCATTAAGCGCGGCGGGTGTGGTGGTTACGCGCAGCGTGACCGCTAC
ACTTGCCAGCGCCCTAGCGCCCGCTCCTTTCGCTTTCTTCCCTTCCTTTCTCGCCACGTTCGCCGGCTTT
CCCCGTCAAGCTCTAAATCGGGGGCTCCCTTTAGGGTTCCGATTTAGTGCTTTACGGCACCTCGACCCCA
AAAAACTTGATTAGGGTGATGGTTCACGTAGTGGGCCATCGCCCTGATAGACGGTTTTTCGCCCTTTGAC
GTTGGAGTCCACGTTCTTTAATAGTGGACTCTTGTTCCAAACTGGAACAACACTCAACCCTATCTCGGTC
TATTCTTTTGATTTATAAGGGATTTTGCCGATTTCGGCCTATTGGTTAAAAAATGAGCTGATTTAACAAA
AATTTAACGCGAATTTTAACAAAATATTAACGCTTACAATTTCCATTCGCCATTCAGGCTGCGCAACTGT
TGGGAAGGGCGATCGGTGCGGGCCTCTTCGCTATTACGCCAGCTGGCGAAAGGGGGATGTGCTGCAAGGC
GATTAAGTTGGGTAACGCCAGGGTTTTCCCAGTCACGACGTTGTAAAACGACGGCCAGTGAATTGTAATA
CGACTCACTATAGGGCGAATTGGGTACCGGGCCCCCCCTCGAGGTCGACGGTATCGATAAGCTTGATATC
GAATTCCTGCAGCCCGGGGGATCCACTAGTTCTAGAGCGGCCGCCACCGCGGTGGAGCTCCAGCTTTTGT
TCCCTTTAGTGAGGGTTAATTTCGAGCTTGGCGTAATCATGGTCATAGCTGTTTCCTGTGTGAAATTGTT
ATCCGCTCACAATTCCACACAACATACGAGCCGGAAGCATAAAGTGTAAAGCCTGGGGTGCCTAATGAGT
GAGCTAACTCACATTAATTGCGTTGCGCTCACTGCCCGCTTTCCAGTCGGGAAACCTGTCGTGCCAGCTG
CATTAATGAATCGGCCAACGCGCGGGGAGAGGCGGTTTGCGTATTGGGCGCTCTTCCGCTTCCTCGCTCA
CTGACTCGCTGCGCTCGGTCGTTCGGCTGCGGCGAGCGGTATCAGCTCACTCAAAGGCGGTAATACGGTT
ATCCACAGAATCAGGGGATAACGCAGGAAAGAACATGTGAGCAAAAGGCCAGCAAAAGGCCAGGAACCGT
AAAAAGGCCGCGTTGCTGGCGTTTTTCCATAGGCTCCGCCCCCCTGACGAGCATCACAAAAATCGACGCT
CAAGTCAGAGGTGGCGAAACCCGACAGGACTATAAAGATACCAGGCGTTTCCCCCTGGAAGCTCCCTCGT
GCGCTCTCCTGTTCCGACCCTGCCGCTTACCGGATACCTGTCCGCCTTTCTCCCTTCGGGAAGCGTGGCG
CTTTCTCATAGCTCACGCTGTAGGTATCTCAGTTCGGTGTAGGTCGTTCGCTCCAAGCTGGGCTGTGTGC
ACGAACCCCCCGTTCAGCCCGACCGCTGCGCCTTATCCGGTAACTATCGTCTTGAGTCCAACCCGGTAAG
ACACGACTTATCGCCACTGGCAGCAGCCACTGGTAACAGGATTAGCAGAGCGAGGTATGTAGGCGGTGCT
ACAGAGTTCTTGAAGTGGTGGCCTAACTACGGCTACACTAGAAGGACAGTATTTGGTATCTGCGCTCTGC
TGAAGCCAGTTACCTTCGGAAAAAGAGTTGGTAGCTCTTGATCCGGCAAACAAACCACCGCTGGTAGCGG
TGGTTTTTTTGTTTGCAAGCAGCAGATTACGCGCAGAAAAAAAGGATCTCAAGAAGATCCTTTGATCTTT
TCTACGGGGTCTGACGCTCAGTGGAACGAAAACTCACGTTAAGGGATTTTGGTCATGAGATTATCAAAAA
GGATCTTCACCTAGATCCTTTTAAATTAAAAATGAAGTTTTAAATCAATCTAAAGTATATATGAGTAAAC
TTGGTCTGACAGTTACCAATGCTTAATCAGTGAGGCACCTATCTCAGCGATCTGTCTATTTCGTTCATCC
ATAGTTGCCTGACTCCCCGTCGTGTAGATAACTACGATACGGGAGGGCTTACCATCTGGCCCCAGTGCTG
CAATGATACCGCGAGACCCACGCTCACCGGCTCCAGATTTATCAGCAATAAACCAGCCAGCCGGAAGGGC
CGAGCGCAGAAGTGGTCCTGCAACTTTATCCGCCTCCATCCAGTCTATTAATTGTTGCCGGGAAGCTAGA
GTAAGTAGTTCGCCAGTTAATAGTTTGCGCAACGTTGTTGCCATTGCTACAGGCATCGTGGTGTCACGCT
CGTCGTTTGGTATGGCTTCATTCAGCTCCGGTTCCCAACGATCAAGGCGAGTTACATGATCCCCCATGTT
GTGCAAAAAAGCGGTTAGCTCCTTCGGTCCTCCGATCGTTGTCAGAAGTAAGTTGGCCGCAGTGTTATCA
CTCATGGTTATGGCAGCACTGCATAATTCTCTTACTGTCATGCCATCCGTAAGATGCTTTTCTGTGACTG
GTGAGTACTCAACCAAGTCATTCTGAGAATAGTGTATGCGGCGACCGAGTTGCTCTTGCCCGGCGTCAAT
ACGGGATAATACCGCGCCACATAGCAGAACTTTAAAAGTGCTCATCATTGGAAAACGTTCTTCGGGGCGA
AAACTCTCAAGGATCTTACCGCTGTTGAGATCCAGTTCGATGTAACCCACTCGTGCACCCAACTGATCTT
CAGCATCTTTTACTTTCACCAGCGTTTCTGGGTGAGCAAAAACAGGAAGGCAAAATGCCGCAAAAAAGGG
AATAAGGGCGACACGGAAATGTTGAATACTCATACTCTTCCTTTTTCAATATTATTGAAGCATTTATCAG
GGTTATTGTCTCATGAGCGGATACATATTTGAATGTATTTAGAAAAATAAACAAATAGGGGTTCCGCGCA
CATTTCCCCGAAAAGTGC
```

CURRICULUM VITA

Joseph Patrick Morris

EDUCATION

- ***University of Louisville, Louisville, Ky***

  M.Eng. in Computer Engineering and Computer Science Complete May 2009

- ***University of Louisville, Louisville, KY***

  B.S. in Computer Engineering and Computer Science 2006

RELATED EXPERIENCE

- **May 2008 – Present** *Department of Biochemistry and Molecular Biology, University of Louisville*
  Programmer Analyst

  Further development and documentation of existing computing infrastructure for the storage and dissemination of single nucleotide polymorphisms in cattle populations.

- **Jan 2008 - May 2008** *PGxl Labs, University of Louisville*
  Information Technology Consultant

  Development of pharmacogenetic diagnostic tool as well as general maintenance and consultation of web-based computing needs.

- **Aug 2007 – Dec 2008** *Center for Genetics and Molecular Medicine, University of Louisville*
  Researcher

  Development of a web services-based laboratory information management system for use with microarray data.

- **Jun 2006-Aug 2007** *Center for Genetics and Molecular Medicine, University of Louisville*
  Researcher

  Development of web services and database for the dissemination of single nucleotide polymorphisms in cattle.