STUDY OF THE RELATIONSHIP BETWEEN *Mus musculus* PROTEIN SEQUENCES AND THEIR BIOLOGICAL FUNCTIONS

A Thesis

Presented to

The Graduate Faculty of The University of Akron

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Pawan Seth

May, 2007

STUDY OF THE RELATIONSHIP BETWEEN *Mus musculus* PROTEIN SEQUENCES AND THEIR BIOLOGICAL FUNCTIONS

Pawan Seth

Thesis

Approved:	Accepted:
Advisor Dr. Zhong-Hui Duan	Dean of the College Dr. Ronald F. Levant
Committee Member Dr. Chien-Chung Chan	Dean of the Graduate School Dr. George R. Newkome
Committee Member Dr. Xuan-Hien Dang	Date
Committee Member Dr. Yingcai Xiao	
Department Chair Dr. Wolfgang Pelz	

ABSTRACT

The central challenge in post-genomic era is the characterization of biological functions of newly discovered proteins. Sequence similarity based approaches infer protein functions based upon the homology between proteins. In this thesis, we present the similarity relationship between protein sequences and functions for mouse proteome in the context of gene ontology slim. The similarity between protein sequences is computed using a novel measure based upon the local BLAST alignment scores. The similarity between protein functions is characterized using the three gene ontology categories. In the study, the ontology categories are represented using a general tree structure. Three ontology trees are constructed using the definitions provided in gene ontology slim. The mouse protein sequences are then mapped onto the trees. We present the sequence similarity distributions at different levels of GO tree. The similarities of protein sequences across gene ontology levels and traversing branches are studied. The posterior probabilities for correct predictions are calculated to study the mathematical underpinnings in evaluating the similarities between the protein sequences. Our results indicate that proteins with similar amino acid sequences have similar biological functions. Although the similarity distribution in each functional group across GO levels varies from one functional group to another, the comparison between distributions of parent and child groups reveals the strong relationship between sequence and function similarity. We conclude that sequence similarity approach can function as a key measure in the prediction of biological functions of unknown proteins. Our results suggest that the posterior probability of a correct prediction could also serve as one of the key measures for protein function prediction.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my advisor, Dr. Zhong-Hui Duan, for her constant encouragement and invaluable guidance during this study. I am grateful to her for offering me an opportunity to do my thesis under her. I am very impressed by her kindness and personality. This thesis and my study in Computer Science Department would not have been possible without her help and support.

I would also like to acknowledge the help of Computer Science Department for offering me an assistantship. I would also like to acknowledge the help from Dr. Wolfgang Pelz, Dr. Yingcai Xiao, Dr. Timothy W. O'Neil, Dr. Xuan-Hien Dang, Dr. Chien-Chung Chan, Dr. K.J. Liszka and Ms. Peggy Speck for their constant assistance.

I would like to dedicate this thesis to my family. Without their encouragement, love and support, I do not think I can finish this degree, this thesis and the study at the University of Akron. I am forever indebted to them, for the sacrifices they make to help me to achieve this success.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The accrual of sequence data including genomic sequences, transcripts, expression data [1] is primarily due to the effort started by U.S. Human Genome Project in 1990 [2]. The rapid advancements in the technology have accelerated the current speed of sequencing resulting in the accumulation of large amounts of information. This has created a bottleneck for a large number of genes which still remain uncharacterized i.e. they have no structural or functional notation [3].

The major problem that has baffled biologists in the post-genomic biology is the functional assignment of proteins: A large percentage of Open Reading Frames (ORFs) have unknown functions which unless resolved will not help biologists comprehend the capabilities of an organism [4]. The challenge is to use bioinformatics to help abridge the gap between the amount of sequence data and the functional annotation. Comparative sequence analysis tools are used for the detection of functional regions in genomic sequences.

1.1 Comparative Methods

The Comparative methods have become an important tool to study the protein sequences. Proteins are composed of amino acids which can be aligned and compared to other protein sequence(s) [5].

The computational tools based on sequence homology -- BLAST, PSI BLAST, are widely used for the functional annotations of genes in newly sequenced genomes [6]. In sequence similarity-approach the functions of a query protein are deduced from those of homologous proteins of known functions obtained from database searches. The sequence similarity approaches for these proteins are based on the assumption that they are functionally linked. The hypothesis is that the evolution of proteins with similar functions occurs in a correlated fashion and therefore the homology is present in the same subset of organisms [7]. There are varieties of sequence similarity algorithms that can find the regions of similarity between protein sequences.

1.1.1 Smith-Waterman Algorithm

Smith-Waterman is one of the most popular local sequence alignment schemes to determine the similarities between the regions of the query sequence and a sequence database (proteins or nucleotides). In 1981 Temple Smith and Michael Waterman proposed this algorithm [8] based on dynamic programming technique which is guaranteed to find an optimal local alignment between two sequences corresponding to the scoring system being implemented (Substitution Matrix or Gaps Scoring). It identifies the maximal homologous sequences among the protein sequences being compared. These protein sequences can be of any length, at any location. The amino acid chains (in case of proteins) or nucleotides are taken as a string and character by character comparison is done. Relative weights are assigned to these character-to-character comparisons. If an exact match is found ("hit") or if a substitution is done a positive weight is assigned to that comparison or else if an insertion or deletion operation is performed a negative weight is assigned to the comparison. These scores are arranged in the weight matrices where they may be added together and the highest scoring alignment is reported.

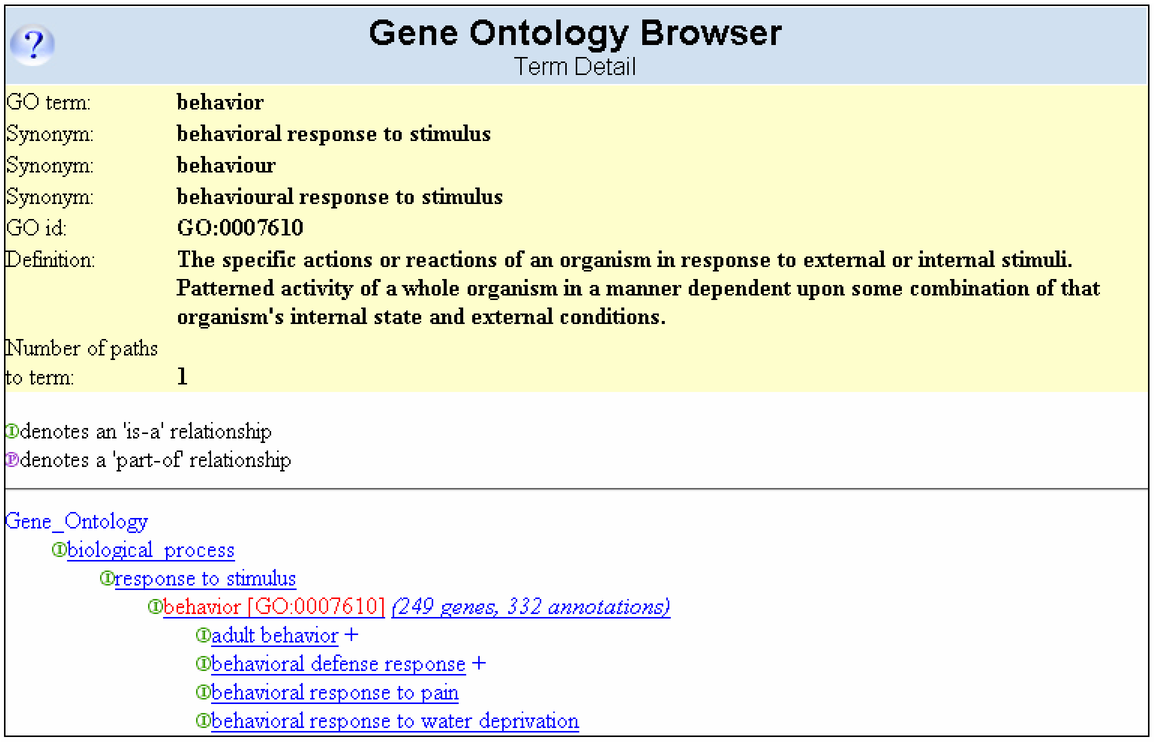
1.1.2. Basic Local Alignment Search Tool

BLAST, a heuristic search algorithm, approximates the Smith-Waterman algorithm is used to compare amino acid sequences of different proteins or the nucleotides of Deoxyribonucleic acid (DNA) sequences [9, 10].

The BLAST then compares a query sequence (protein or nucleotide) and a sequence database (protein database or nucleotide database) and identifies the database sequences that resemble the query sequence above a certain threshold. The main idea behind BLAST's operation is that given a pair of sequences, algorithm will try to match small fixed length W between the query and sequences in database and will try to extend this length in both directions. Using this way it identifies regions of local alignment in the query sequence similar to subsequences in database and label them as High Scoring Pairs (H.S.P.) [10]. These regions of high sequence similarity are assigned some scores based on the scoring system used and statistically significant alignments are displayed to the user. These alignments can further be studied and with the help of statistical concepts and inferences can be drawn.

1.2 Gene Ontology

The genetic information of a cell is carried by Deoxyribonucleic Acid (DNA) and it consists of thousands of genes. Genes are the working subunits of DNA and encode instructions on how to make proteins [11]. The Gene Ontology (GO) provides a controlled vocabulary to describe gene and gene products in an organism. The three organizing principles of GO are - biological process, cellular component and molecular function [12]. A gene or a gene product may be associated with one or more cellular processes; active in biological process and perform molecular function. A cellular component is a part of the cell, either an anatomical structure or a gene product. A biological process refers to events attained by a single unit or assembly of molecular functions. Molecular function describes the activities occurring at the molecular level. The terms in these ontology are organized in a Directed Acyclic Graph (DAG) and linked by two relationships, 'is a' and 'part of'. DAG is also referred to as a rooted tree (tree with a root). Gene Ontology Browser can be used to describe this tree like structure.


Gene Ontology Browser
Term Detail

GO term:	**behavior**
Synonym:	**behavioral response to stimulus**
Synonym:	**behaviour**
Synonym:	**behavioural response to stimulus**
GO id:	**GO:0007610**
Definition:	**The specific actions or reactions of an organism in response to external or internal stimuli. Patterned activity of a whole organism in a manner dependent upon some combination of that organism's internal state and external conditions.**
Number of paths to term:	**1**

Ⓘdenotes an 'is-a' relationship
Ⓟdenotes a 'part-of' relationship

```
Gene_Ontology
    Ⓘbiological_process
        Ⓘresponse to stimulus
            Ⓘbehavior [GO:0007610] (249 genes, 332 annotations)
                Ⓘadult behavior +
                Ⓘbehavioral defense response +
                Ⓘbehavioral response to pain
                Ⓘbehavioral response to water deprivation
```

Figure 1.1. View of GO:0007610 using Gene Ontology Browser

For example GO:0007610 represent the behavioral response to stimulus, assigned to biological process and textual definition for this GO terms is *"The specific actions or reactions of an organism in response to external or internal stimuli. Patterned activity of a whole organism in a manner dependent upon some combination of that organism's internal state and external conditions."* GO slim is a cut down vocabulary provided by GO ontologies. GO slim contains a subset of terms in the whole GO [13]. GO slims are created by users according to their needs and provides a brief overview of ontology content without going into specific fine grained specifications.

A wide variety of ontology based searches have been designed to annotate sequences on a large scale. Vinagayam [14] used support vector machines for the assignment of molecular function GO terms to uncharacterized cDNA sequences and to define a confidence value for each prediction. cDNA sequences were annotated to GO and these sequences were then used to train a Support Vector Machine (SVM) classifier. The nucleotide sequences were searched against GO-mapped protein databases and significant hits were recorded. Each GO-term obtained was either labeled as correct (+1) or incorrect (-1) by comparing it with original annotation. BLAST results were associated as "features" with these samples. The classifier was trained with this data to predict the function of unknown sequences. This automated annotation system resulted in the large scale cDNA functional assignment, to achieve a high-level of prediction accuracy without any manual intervention. Zehetner [15] worked on the OntoBlast to predict the potential functions for an unknown sequence by presenting a weighted list of ontology entries associated with similar sequences from completely sequenced genomes identified in BLAST search. It then finds information regarding the potential functions. The functional annotation of the sequences provides an insight to the processes in which a gene may be involved . Xie et al's [16] GO engine combines homology search with text mining. Schug [17] developed rule-based systems based on the intersection of GO terms that contain protein domain at different similarity levels. The appeal of these approaches is that they can directly assign a biological meaning to an uncharacterized protein sequence. However, matching sequences do not always infer similar functions [4].

1.3 Chromosome (*Mus Musculus*)

In this thesis, we investigated the degree of overall similarity of protein sequences from Chromosome 1 (Mouse) in each functional group defined by GO terms. Mouse (Mus musculus) is a common rodent, closely related to the rat. The mouse has been a major organism, for research purposes to study basic biology, on which extensive works have been done to sequence its genome. The genome of Mus musculus was the second mammalian genome to be sequenced whose complete draft entered the public nucleotide sequence repositories in 2002. It has 19 chromosome pairs, 1 X and 1 Y chromosomes which can be viewed with the help of Ensembl tool [18].

Ensembl project came into being with the collaborative efforts from EMBL - European Bioinformatics Institute (EBI) and the Wellcome Trust Sanger Institute (WTSI). The main task was to develop a software system which could produce and maintain an automatic annotation on selected eukaryotic genomes.

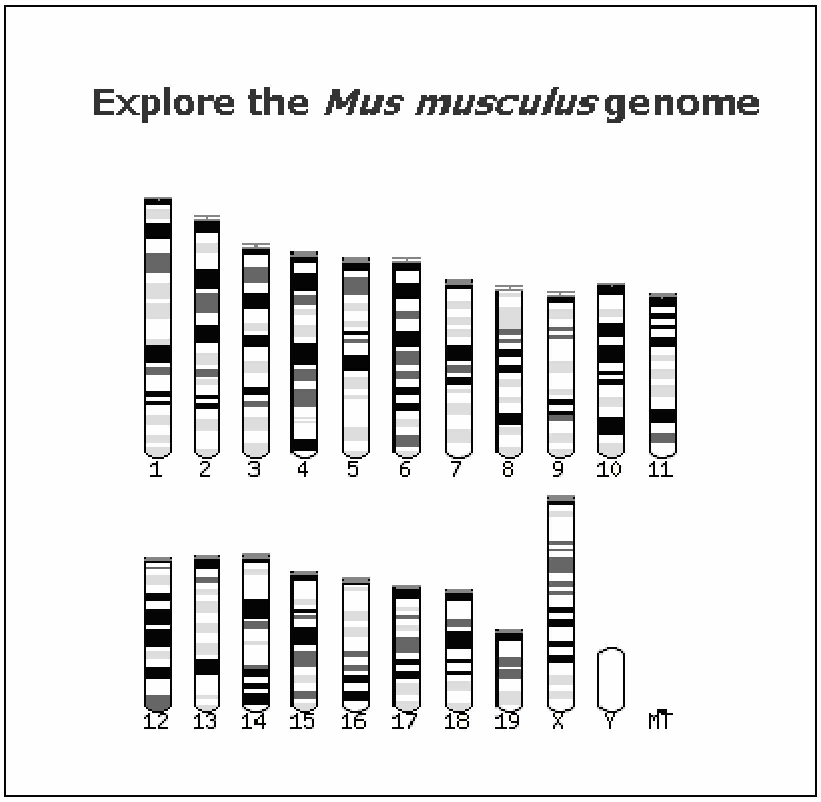


Figure 1.2 Exploring the *Mus Musculus* genome using Ensembl site tool

1.4 Overview of Thesis Work

In this thesis, we investigated the mathematical underpinnings of an automated sequence annotation approach based on sequence similarity and gene ontology. In the Chapter I we revised the basic concepts of biology and bio-informatics relevant to the area of this research study. In Chapter II - materials and methods, we studied the degree of similarity of protein sequences in each functional group defined by a GO term, using the protein sequences from chromosome 1 of *Mus Musculus*. The dataset (protein sequences for chromosome 1 for *Mus Musculus*) was downloaded from European Bioinformatics Institute (EBI) website [20], gene ontology file from gene ontology consortium [12] and alignment tools from National Center for Biotechnology Information

(NCBI) [9]. In chapter III - results and discussion, PERL scripts were processed and parsed to get the distribution of similar pairs for the three ontologies - namely biological process, molecular function and cellular component. We studied the degree of similarity of protein sequences in each functional group defined by a GO term, using the protein sequences from chromosome 1 of mouse. We explored the structures of the three ontologies - biological, cellular and molecular category and re-evaluate the hypothetical assumption - similar biological sequences implies similar functions. We used a novel measure of overall similarity between protein sequences based on the results of local BLAST alignments [19].

We also examined the effects of the levels of GO terms on the degree of similarity and also discussed the sequence similarity distribution at different levels of GO tree. Similarity distributions of sequence pairs were also analyzed for each of molecular function, biological process and cellular component ontologies branch-wise. To analyze and predict the plausible potential relationships of similar sequences we computed the posterior probability of the hypothesis - probabilities of the A and B having similar functions after it is known that both A and B have similar sequences.

CHAPTER II

MATERIALS AND METHODS

This chapter addresses the strategies and operations used and implemented in our studies. Mouse (Mus musculus) has been an important organism in biology and medicine for research purposes. Sequence similarity approach, in particular Smith-Waterman algorithm was proposed by Temple Smith and Michael Waterman in 1981. A more faster and popular algorithm which approximates Smith-Waterman is Basic Local Alignment Search Tool (BLAST) was developed by Stephen Altschul, Warren Gish, David Lipman, which primarily compares biological sequence information.

2.1. Dataset (Chromosome 1 of *Mus Musculus*)

The protein sequences for first chromosome of mouse (*Mus Musculus*) were downloaded from the (EBI - UNIPROT format) [20] in May, 2006. Each line of an experiment entry in the file begins with a two character line code (identifier) which suggests the type of information contained in the line. The identifiers and the information they suggest are shown in the Table 2.1.

Table 2.1 Information contained in UniProt flat file [20]

Code	Meaning	Description
ID	Identification	Contains identifying information and characteristics of the sequence.

Table 2.1 Information contained in UniProt flat file [20]

DT	Date	When the entry was created, or when the sequence or annotation was modified.
DE	Description	The gene(s) that code for the protein.
GN	Gene name(s)	The organism from which the sequence is derived.
OS	Organism species	If the sequence is non-chromosomal in origin.
OG	Organelle	The taxonomic class to which the organism belongs.
OC	Organism classification	The NCBI TaxID for the OC line.
OX	Taxonomy cross-reference(s)	The sequential number of the literature citation within the entry.
RN	Reference number	Bibliographic cross-reference, such as PubMed ID.
RX	Reference cross-reference(s)	Authors of the citation.
RA	Reference authors	Title of the citation.
RT	Reference title	Source of the citation, such as journal, book, or unpublished data.
RL	Reference location	Free text notes about the protein.
CC	Comments	Pointers to sources or related information for the entry.
DR	Database cross-references	Annotation of specific residues of the sequence.
FT	Feature table	Marks the beginning of the sequence and provides summary data.
SQ	Sequence header	The sequence itself.
(no code)	Sequence data	End of entry.
//	Termination line	

UniProt dataset was picked up [20] for this thesis as it comes along with a lot of information related to any particular protein other than the amino acid sequences comprising it.

Table 2.2 List of unique proteins for each chromosome pair (*Mus Musculus*)

Genome component	Length (bp)	Number of unique proteins

Table 2.2 List of unique proteins for each chromosome pair (*Mus Musculus*)

Chromosome 1	197069962	1870
Chromosome 2	181976762	3709
Chromosome 3	159872112	1547
Chromosome 4	155029701	2811
Chromosome 5	152003063	1869
Chromosome 6	149525685	1728
Chromosome 7	145134094	2583
Chromosome 8	132085098	1565
Chromosome 9	124000669	1780
Chromosome 10	129959148	1450
Chromosome 11	121798632	3367
Chromosome 12	120463159	1088
Chromosome 13	120614378	1179
Chromosome 14	123978870	1177
Chromosome 15	103492577	1178
Chromosome 16	98252459	1017
Chromosome 17	95177420	1580
Chromosome 18	90736837	773
Chromosome 19	61321190	1063
Chromosome X	165556469	1378
Chromosome Y	16029404	38

UniProt sets for 19 chromosome pairs, 1 X and 1 Y pair were taken and base pairs (bp) per chromosome and number of unique proteins in each of them were listed in Table 2.2 above. Two nucleotides on opposite complementary DNA or RNA strands that are connected via hydrogen bonds are called base pairs. Chromosome 1 has the largest length (bp) so it picked up for this thesis. There were 1870 protein sequences in the first chromosome.

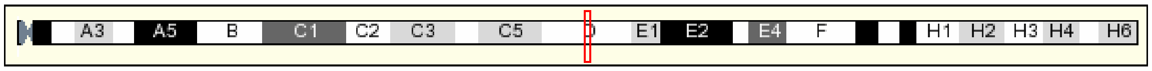


Figure 2.1 Chromosome 1 using Ensembl site tool

2.2. Sequence Similarity Approach

There are varieties of sequence similarity tools that align the amino acid sequence pairs (from two different proteins) and find the regions of high similarity scores between them. Local or global alignments are the two main approaches to compute the regions of high similarity between sequence pairs. Global alignment approach aligns the entire amino acid sequences between the pairs. By contrast, local alignment scheme identifies the similar regions within the long sequences thus increasing the chances of getting more number of similar regions as compared to former method which tries to globally optimize the entire sequence over the other [21].

Smith-Waterman is one of the most popular local sequence alignment schemes to determine the similarities between the regions of the query sequence and a sequence database (proteins or nucleotides).

This algorithm is based on the dynamic programming approach, which finds the solutions to the smaller chunks of a problem and combines them on the whole to find a complete optimal solution to the problem. It recursively performs the local alignment comparison on the segments of all possible paths and picks up the one which has the maximum similarity score as an optimal solution until a threshold has been reached. Based on the above calculations, character-to-character comparison is done and scores or weights are assigned to each comparison. It's positive for exact matches/substitutions, and negative for insertions/deletions. A weight matrix is build, scores are added and highest scoring alignment is reported.

This technique is more sensitive and superior as compared to BLAST and FASTA as it does pair wise comparisons which results in covering large number of possibilities but the time taken to run this algorithm is higher as compared to the other two. This explains the popularity of the BLAST algorithm.

For example, there are two nucleotide sequences $A = a_1\ a_2\ a_3 \ldots a_n$ and $B = b_1\ b_2\ b_3 \ldots b_m$. $s(a, b)$ denotes the similarity between sequence elements a and b. W_k denotes the deletions of length k. A matrix H to find pairs of segments with high degrees of similarity is set up

$$H_{k0} = H_{0l} = 0 \text{ for } 0 \leq k \leq n \text{ and } 0 \leq l \leq m$$

H_{ij} is the maximum similarity of two segments ending in a_i and b_j respectively is calculated from the equation [22]

$$H_{ij} = \max\{H_{i-1, j-1} + s(a_i, b_j), \max\{H_{i-k, j} - W_k\}, \max\{H_{i, j-1} - W_l\}, 0\} \qquad \text{(Eq. 2.1)}$$

Where, $1 \leq i \leq n$ and $1 \leq j \leq m$

The calculation of H_{ij} from equation 2.1 considers the following possibilities for ending segments at any a_i and b_j.

1) If a_i and b_j are associated, then new score is the previous score plus the similarity scores for the two residues.

$$H_{i-1, j-1} + s(a_i, b_j)$$

2) If a_i is at the end of a deletion of length k, the similarity is

$$H_{i-1, j} - W_K$$

3) If b_j is at the end of a deletion of length l, the similarity is

$$H_{i-1, j} - W_l$$

4) Finally, a zero is included to prevent calculated negative similarity, indicating that no similarity up to a_i and $b_{j.}$

Noticeably, we are transforming one string into another string by performing certain operations on the individual characters that make up that string. So similarity between two strings can also be defined as "the value of alignment between the two strings that maximizes the total alignment value (highest score)"

Here's an example to show the implementation of the Smith Waterman algorithm more clearly [23]. Suppose there are two nucleotide sequences which are to be compared against each other

Sequence 1: CAGCCUCGCUUAG

Sequence 2: AAUGCCAUUGACGG

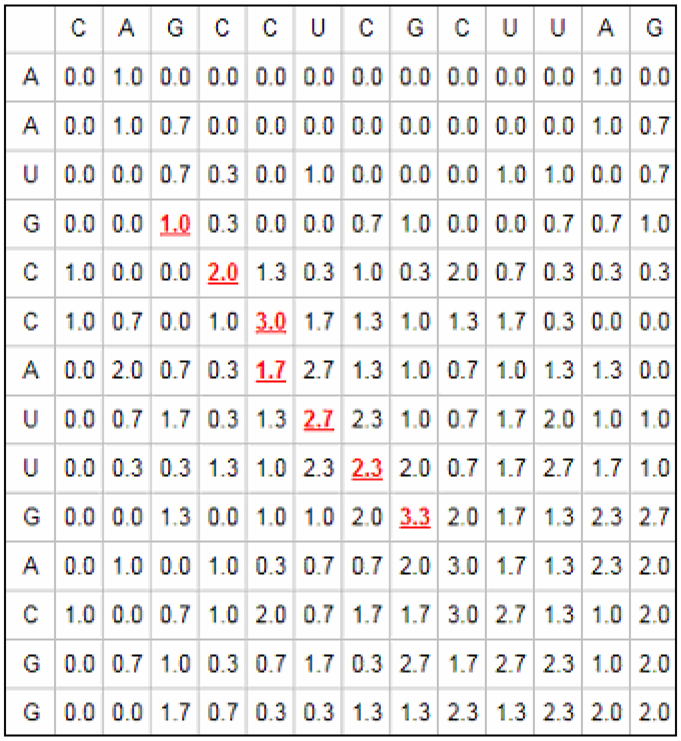
Scores are derived from a simple similarity matrix, values chosen are:

$\rightarrow$ Match = +1

$\rightarrow$ Mismatch = $-\frac{1}{3}$

$\rightarrow$ Gap = $-1+\frac{1}{3} \times k$ (k = extent of gap, number of residues included in the gap)

A similarity matrix is build up with all cell values = 0 and to ensure that a new alignment path can start at any point the scores are not allowed to fall below 0. Values are updated in the cell based on the value of the cell plus the highest value in sub row, sub column or direct diagonal while keeping the gap penalties in account. These values can rise, fall or stay same. The value in any cell is the highest score for an alignment of any length ending at that cell.



	C	A	G	C	C	U	C	G	C	U	U	A	G
A	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
A	0.0	1.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.7
U	0.0	0.0	0.7	0.3	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.7
G	0.0	0.0	1.0	0.3	0.0	0.0	0.7	1.0	0.0	0.0	0.7	0.7	1.0
C	1.0	0.0	0.0	2.0	1.3	0.3	1.0	0.3	2.0	0.7	0.3	0.3	0.3
C	1.0	0.7	0.0	1.0	3.0	1.7	1.3	1.0	1.3	1.7	0.3	0.0	0.0
A	0.0	2.0	0.7	0.3	1.7	2.7	1.3	1.0	0.7	1.0	1.3	1.3	0.0
U	0.0	0.7	1.7	0.3	1.3	2.7	2.3	1.0	0.7	1.7	2.0	1.0	1.0
U	0.0	0.3	0.3	1.3	1.0	2.3	2.3	2.0	0.7	1.7	2.7	1.7	1.0
G	0.0	0.0	1.3	0.0	1.0	1.0	2.0	3.3	2.0	1.7	1.3	2.3	2.7
A	0.0	1.0	0.0	1.0	0.3	0.7	0.7	2.0	3.0	1.7	1.3	2.3	2.0
C	1.0	0.0	0.7	1.0	2.0	0.7	1.7	1.7	3.0	2.7	1.3	1.0	2.0
G	0.0	0.7	1.0	0.3	0.7	1.7	0.3	2.7	1.7	2.7	2.3	1.0	2.0
G	0.0	0.0	1.7	0.7	0.3	0.3	1.3	1.3	2.3	1.3	2.3	2.0	2.0

Figure 2.2 Matrix H_{ij} generated after applying the algorithm [23]

In the above example the alignment is obtained contains both a mismatch and an internal deletion.

G-C-C-A-U-U-G

G-C-C-*-U-C-G

However, the Smith-Waterman algorithm is fairly demanding of time and memory resources: in order to align two sequences of lengths m and n, O (mn) time and space are required. In the next section we will be discussing about another comparison algorithm popularly known as BLAST.

2.3 Basic Local Alignment Search Tool Algorithm

Basic Local Alignment Search Tool (BLAST), an approximation of Smith-Waterman algorithm searches for high scoring sequence alignments between the query sequence and the database of sequences. BLAST works in three major steps [24, 25, 26,]:

1) Compile list of high-scoring strings (words) - BLAST filters out low complexity regions from the query sequence and compiles a list of high-scoring words which consists of all words with '*w*' characters that scores at least '*T*' with some word in the query sequence. BLAST uses a scoring matrix (described below - BLOSUM 62 is by default for amino acids) to determine all matching words with high scores. A Low complexity and small threshold score may result in reporting of large number of statistical significant but biologically un-interesting results. The values above a certain threshold are taken. There can be a tradeoff between speed and sensitivity at this stage: higher threshold gives greater speed but might miss biologically significant results [27].
2) Search for hits - In the second step BLAST searches through the target sequence database for exact matches to the word list generated either using a hash table or finite state machine. Finite state machines are used are used to calculate state transition table that tells what state to go is based on the next character in the sequence. If a match is found, it is used to seed a possible alignment between the query and the database sequences.
3) Extend seeds to obtain segment pairs - In third step, BLAST method tries to extend the alignment from these matching words in both directions as long as score increases. The resulting segment pairs are called High Scoring Pair (H.S.P).

BLAST determines whether each score found by one of the above methods is greater in value than a given cutoff score S [27, 28]. The maximal scoring pairs, or MSPs, from the entire database are identified and listed. Consequently, BLAST finds out the statistical significance of each score, initially, by calculating the probability that two random sequences, one the length of the query sequence and the other the length of the database could produce the calculated score. When the expectation value for a given database sequence is satisfied a match is reported. Typically the expect value is between 0.1 and 0.001.

BLAST search of the sequence database may result in many alignments and it becomes hard to distinguish between significant alignments and potential random matches. BLAST provides information about: raw scores, bit scores and E values. The raw score for a local sequence alignment is the sum of the individual scores making up the MSP. Because of differences between scoring matrices, raw scores are not necessarily comparable. Bit scores, however, can be compared, since they take into account the scale or log base of the scoring matrix λ) and the scale of the search space size (K), and can be expressed as:

$$S' = \frac{\lambda S - \ln K}{\ln 2}$$

The expectation, or *E* value, corresponding to a given bit score is $E = m \times n \times 2^{-S}$, where n is the length of the query sequence and m is the length of the database sequence. Given that the score of the best local alignment (MSP score) is the maximum of scores of many independent alignments, the probability of observing a score S greater than or equal to certain given threshold when comparing two random sequences is given by extreme value distribution. For certain conditions, this can be rearranged to express the probability that a pair wise alignment with score S could have been obtained by chance. Poisson distribution can be used to find out the probability of observing a particular score in a database of sequences. Expectation value for the Poisson distribution is given by E = $Kmne^{-\lambda S}$ [27, 28] and describes the probability that a score as high as the one observed between two sequences will be found purely by chance. E- values provide an estimate of the number of alignments one would expect to find with a score greater than or equal to that of the observed alignment in a search against a random database of the same composition. An E value greater than 1 indicates that the alignment probably has occurred by chance, and that the query sequence has been aligned to a sequence in the database to which it is not related. E values less than 0.01 are typically taken to represent biological significance [29, 30].

2.3.1. Scoring Matrices

BLAST tool conducts a local similarity search between a target query sequence and a sequence database. It assigns a weight to all relative relationships between different amino acids in a protein sequence based on a match or a mismatch in the form of a scoring matrix. A two dimensional matrix is used to model a match or a mismatch between all pairs of amino acids. The two most used and popular matrices are the Block Substitution Matrix (BLOSUM) [Henikoff and Henikoff, 1992] [31] and Point Accepted Mutation (PAM) [Dayhoff and Schwartz, 1978 [32]. The BLOSUM matrix assigns a probability score, 'P' for each position in an alignment based on the frequency with which the substitution occurs within conserved blocks of related proteins. PAM is based on the

Markov model where change of amino acid at a particular site is assumed to be independent of previous mutation.

Blocks Substitution Matrix (BLOSUM)

It was developed by Heinkoff and Heinkoff and is based on the extraction of conserved ungapped segments called "blocks" from a set of locally aligned protein sequences. Local alignments can be represented as ungapped blocks with each row a different protein segment and each column an aligned residue position.

A blocks database contains numerous aligned ungapped segments corresponding to highly conserved regions of proteins. They are used to search for differences among sequences of the much conserved regions of a protein family i.e. BLOcks SUbstitution Matrix (BLOSUM) [33]. After all the sequences were collected in the blocks database, then for each one the sum of the number of amino acids in each site is collected to get a frequency table of how often different pairs of amino acids are found together in these conserved regions. For example, BLOSUM62 can be used to represent a block which has more than 62% identity in the gapped sequence alignment. BLOSUM 62 is the default matrix for the BLAST program.

All the sequences of amino acids are collected in the BLOCK database and then for each one the number of amino acids in each site is summed up to get a frequency table (q_{ij} , i, j =1, …, 20) which represents the number of times different pairs of amino acids pairs are found together in these conserved regions. Hence the observed frequency of occurrence of one amino acid is [34]

$$p_{ij} = q_{ii} + \sum_{i \neq j} q_{ij} \times \frac{1}{2}$$

p_{ij} represents the expected probability of occurrence of the ith residue in an (i, j) pair.

Frequency of the given pairs can be given as:

$$e_{ij} = p_i^2 \text{ , } if \ i = j \ (like\ comparisons)$$

and

$$e_{ij} = 2 \times p_i \times p_j, \ if \ i \neq j \ (unlike\ comparisons)$$

Where p_i, p_j represents the number of times the residue i, j was observed in the column respectively.

The odds matrix is given by $s_{ij} = 2 \times \log_2 (q_{ij} \div e_{ij})$ after taking the logarithm of the odd matrix. Where, $s_{ij} = 0$ represents that there are no differences between the observed and expected number of pairs of amino acids. $s_{ij} < 0$ represents if the observed number of pairs of amino acids are less than the expected and if the observed is greater than the expected then $s_{ij} > 0$.

Scores are populated in the form of two dimensional matrixes where the relative similarity and dissimilarity between the pairs of amino acids in the query sequence and a sequence database are reported on the basis of percentage of similarity of the amino acids in the groups. For example, BLOSUM62 matrix is calculated from the protein blocks only if the two sequences are more than 62% identical. The standard substitution matrix for BLOSUM62 contains the score for all possible exchanges of one amino acid with another [35] is show below.

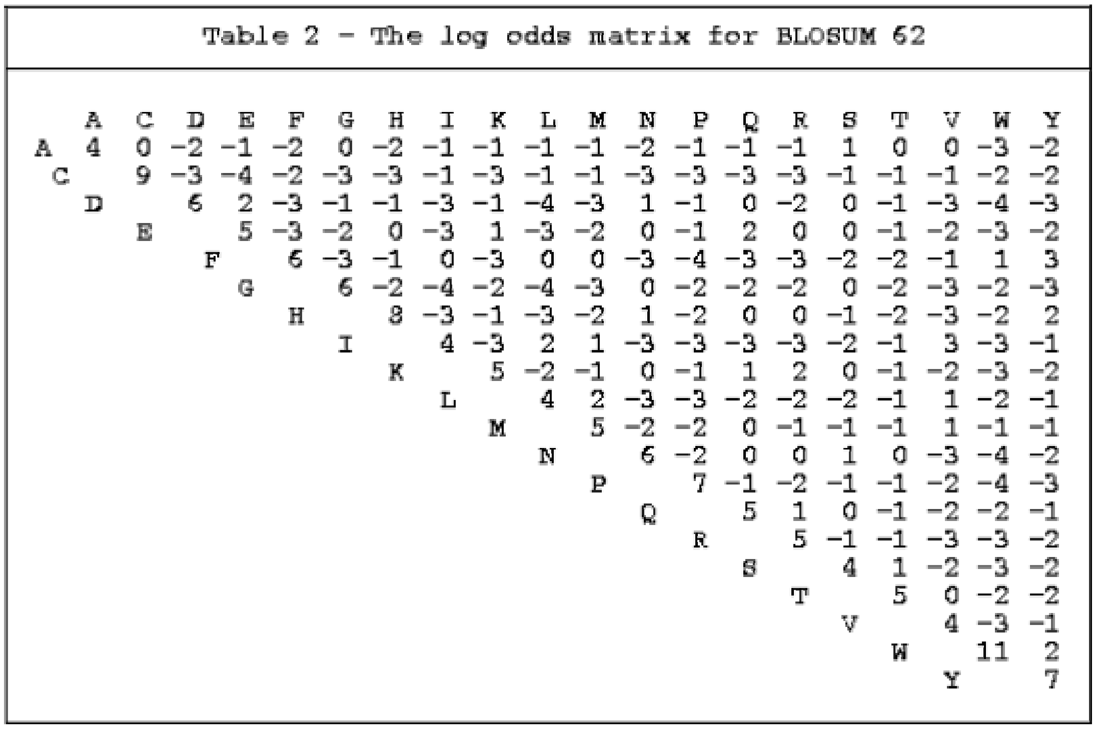


Table 2 - The log odds matrix for BLOSUM 62

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
A	4	0	-2	-1	-2	0	-2	-1	-1	-1	-1	-2	-1	-1	-1	1	0	0	-3	-2
C		9	-3	-4	-2	-3	-3	-1	-3	-1	-1	-3	-3	-3	-3	-1	-1	-1	-2	-2
D			6	2	-3	-1	-1	-3	-1	-4	-3	1	-1	0	-2	0	-1	-3	-4	-3
E				5	-3	-2	0	-3	1	-3	-2	0	-1	2	0	0	-1	-2	-3	-2
F					6	-3	-1	0	-3	0	0	-3	-4	-3	-3	-2	-2	-1	1	3
G						6	-2	-4	-2	-4	-3	0	-2	-2	-2	0	-2	-3	-2	-3
H							8	-3	-1	-3	-2	1	-2	0	0	-1	-2	-3	-2	2
I								4	-3	2	1	-3	-3	-3	-3	-2	-1	3	-3	-1
K									5	-2	-1	0	-1	1	2	0	-1	-2	-3	-2
L										4	2	-3	-3	-2	-2	-2	-1	1	-2	-1
M											5	-2	-2	0	-1	-1	-1	1	-1	-1
N												6	-2	0	0	1	0	-3	-4	-2
P													7	-1	-2	-1	-1	-2	-4	-3
Q														5	1	0	-1	-2	-2	-1
R															5	-1	-1	-3	-3	-2
S																4	1	-2	-3	-2
T																	5	0	-2	-2
V																		4	-3	-1
W																			11	2
Y																				7

Figure 2.3 Standard substitution matrix for BLOSUM62

There are many levels to score proteins which are less divergent as compared to others which are more divergent. For distant related protein sequences BLOSUM45 can be used. For closely related sequences BLOSUM80 matrices can be used. BLOSUM50, BLOSUM62 and BLOSUM80 are few of the different levels of the BLOSUM matrix scoring system that can be implemented to assign different weights to similarity between two sequences. BLOSUM62 is the default for BLOSUM and studies have shown it to be the best for detecting weak protein similarities [36]. So for this thesis BLOSUM62 scoring system was chosen as a substitution matrix for sequence alignment of proteins.

2.3.2. Bl2seq

BL2seq works on the BLAST algorithm and performs a comparison between the two sequences using either the blastn or blastp program [37, 38, 39]. Both sequences must be either nucleotides or proteins. Input to the bl2seq is two sequences files (either nucleotides or proteins) which are in the FASTA format. Typically the command to run Bl2seq from the command line is as follows:

```
bl2seq -p -i -j -o
```

Table 2.3 bl2seq options (cited from NIH website)

Option	Definition	Type	Default
-i	First sequence.	[File In]	
-j	Second sequence.	[File In]	
-o	Alignment output file.	[File Out]	stdout
-p	Program name: *blastp*, *blastn*, *blastx*, *tblastn*, *tblastx*. For *blastx*, the first sequence should be nucleotide; for *tblastn*, the 2nd sequence should be nucleotide.	[String]	

Bl2seq Results

Statistically significant alignments are stored and result values are reported in following way [40]:

- Score: It is calculated by summing the scores for each aligned position of the amino acid and deducting the penalty of the gaps.
- Expect: It estimates the statistical significance of the match, specifying the number of matches within a given score that are expected in a search of a database of this size absolutely by chance.

- Positives: The number and fraction of residues for which the alignment scores have positive values.
- Identities: Number and percentage of exact residue matches.

  Gaps: Positions at which a letter is paired with a null are called gaps.

However, to encompass the overall similarity of two protein sequences in terms of functional, structural or evolutionary relationships is not obvious. For example, proteins from the same domain with a certain similarity score might be more dissimilar as compared to proteins from two different domains (based on the similarity scores comparison). In this study, we employed the alignment technique to blast two protein sequences so to obtain the similar regions with a certain statistical significant score, also known as regions of optimal sequence alignments. If there are n regions having scores $\{R_1, \ldots\ldots, R_n\}$ of statistical significance, we use the following score to compute the overall similarity of two sequences.

$$S = -\sum_i \ln p_i \qquad \text{(Eq. 2.2)}$$

Where S is overall similarity score, p_i is the probability of finding high-scoring segment pair with a local alignment score of at least S_i i.e. $p_i = 1 - e^{-Ei}$ and E_i is expected number of H.S.P.'s of score at least S_i. Assuming that the H.S.P.'s are independent of each other, the p-value can be given as $p = e^{-S}$, probability of finding a pair of protein sequences with a list of scores at least $\{R_1, \ldots\ldots, R_n\}$. We use *p*-values and E-values to represent the significance of the alignment between a pair of protein sequences. *p*-values and E-values are same when they are small. For convenience we will use p-values for our study. bl2seq (v. 2.2.14) alignment tool was downloaded from NCBI website and implemented with a BLOSUM62 scoring system with all default parameters. PERL scripts were written to parse the protein sequences and blast them against the protein sequences from chromosome 1. Results were stored in the text files which were easy to work with.

The GO terms for which the proteins from chromosome 1 were annotated and also which have their definitions in GOSlim were stored in a tree like structure for each of the three gene ontologies.

2.4. Gene Ontology

Gene ontology consortium started the gene ontology project which provides a controlled vocabulary for the consistent description of gene and gene product attributes for any organism. It encompasses broadly three roles:

- First, the development and maintenance of the structured vocabularies (ontologies) themselves.
- The annotation of the gene products which includes the association between the ontologies and the genes and gene products.
- Development of tools that can help in the making, maintaining and using ontologies.

The Gene Ontology (GO) project describes the gene products in terms of three structured controlled vocabularies (ontologies) namely biological process, cellular component and molecular function. The GO terms are the building blocks for GO and are represented by an unique alphanumerical identifier in the form 'GO:XXXXXXX', a term name, synonym, and a definition. GO terms are then classified into one of the three ontologies and structured as a DAG. Each GO term has got a definition, association with one of the three ontologies, along with a relationship identifier which describes the term's relationship (parent-child) with other GO terms. The consortium updates the ontology frequently on the monthly basis [41]. If it is decided by the consortium that a GO term is not appropriate then it is marked as obsolete. GO terms can also be represented using the GOSlim which are the cut down version and subset of the gene ontology. The GOSlim depicts a broader overview of the content of ontology without going into details of specific grained terms [42]. GO slims are created by users according to their experimental needs, purposes. The terms are represented as nodes and arcs the different relationships. These relationships can be used to draw tree like structure where child nodes can be derived from the root node or they are the part of their parent node. GOSlim has a tag-value format to represent the definitions of the GO definitions file. The tag "id:" denotes a unique GO id assigned to a term. This GO term is then recognized by this name only. "name:" tag denotes the respectively ontology that a GO id belongs to [42]. It can be any of the three ontologies. "def:" tag gives a formal definition for that GO id. The tag "subset" describes the gene ontology of the organism from where this GO id is taken. GO terms are classified according to their levels which corresponds to the depth it has in the Gene Ontology tree and their defined functions. This tree like structure is actually an acyclic diagraph which represents the parent-child relationship between various GO terms. GOSlim definition file was downloaded from [42] (format-version: 1.0 dated: 21:12:2006 19:30). PERL scripts then parsed out to annotate protein sequences for biological process, molecular function and cellular component ontologies.

2.5. Perl

There are many languages like Java, C, FORTRAN, MATLAB etc. which can be used to write bioinformatics applications. In our thesis we used Practical Extraction and Report Language (PERL) [43] because of the following reasons.

- Protein sequences and other biological data is stored in enormous databases and text files. PERL with its high capability of recognizing string patterns simplifies the processing and analysis.
- It takes far less programming time to extract data with PERL than with C or with Java.
- It is an excellent scripting language for text analysis. The built-in operators make the searching, replacing and pattern matching effortless.
- PERL is easy to install and requires very less space to install the libraries.
- PERL has the portability of an interpreted language while achieving nearly the speed of a compiled language.
- “Techniques" such as "fast CGI", keeps the frequently accessed CGI script in memory for repetitive execution [44]. This avoids this startup latency, except on the very first execution of a script.

CHAPTER III

RESULTS AND DISCUSSIONS

The UNIPROT protein sequences data set for *M. Musculus* (chromosome 1) was downloaded from EBI website [20]. There were 1870 protein sequences contained in it. The protein sequences were annotated to GO terms for each biological process, molecular function and cellular component ontology. In total there were 130 GO terms defined for the three ontologies in GO slim file downloaded from GO consortium website.

GOSlim has 52 GO terms definitions in total for biological process ontology, 41 GO terms definitions for molecular function ontology, 37 GO terms definitions for cellular component ontology. The next task was to calculate the actual number of GO terms from these 130 GO terms for which protein sequences (1870 from chromosome 1) were annotated (Table 3.1). There were 449 protein sequences (24.01 % of protein sequences of chromosome 1) annotated with 29 molecular functions terms, 398 protein sequences (21.28 % of protein sequences of chromosome 1) were annotated for 21 cellular component terms and 191 protein sequences (10.21 % of protein sequences of chromosome 1) annotated for 26 biological process terms.

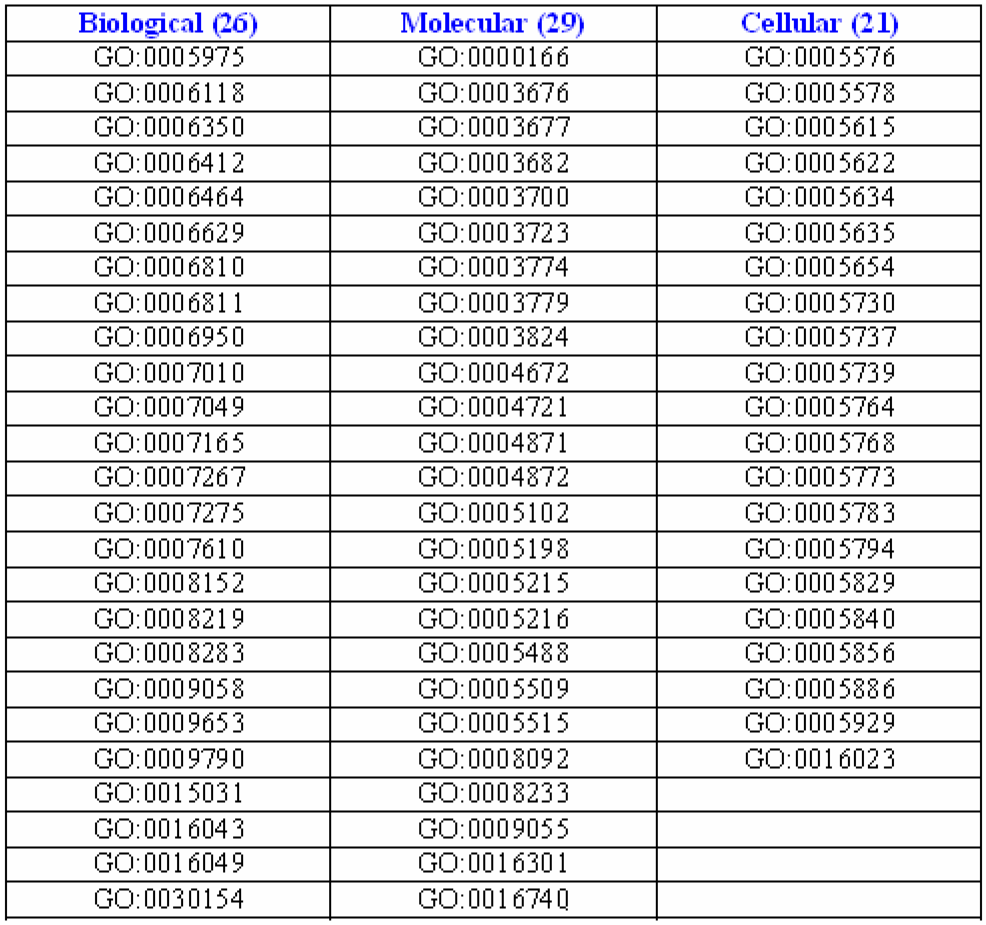
Tables 3.1 Annotated protein sequences distribution for GO slim

	Protein sequences annotated to GOSlim tree		Ontologies associated with protein sequences
	(Number)	(Percentage)	
Molecular function	449	449/1870 = 24.01 %	29

Tables 3.1 Annotated protein sequences distribution for GO slim

Cellular component	398	398/1870 = 21.28 %	21
Biological process	191	191/1870 = 10.21 %	26

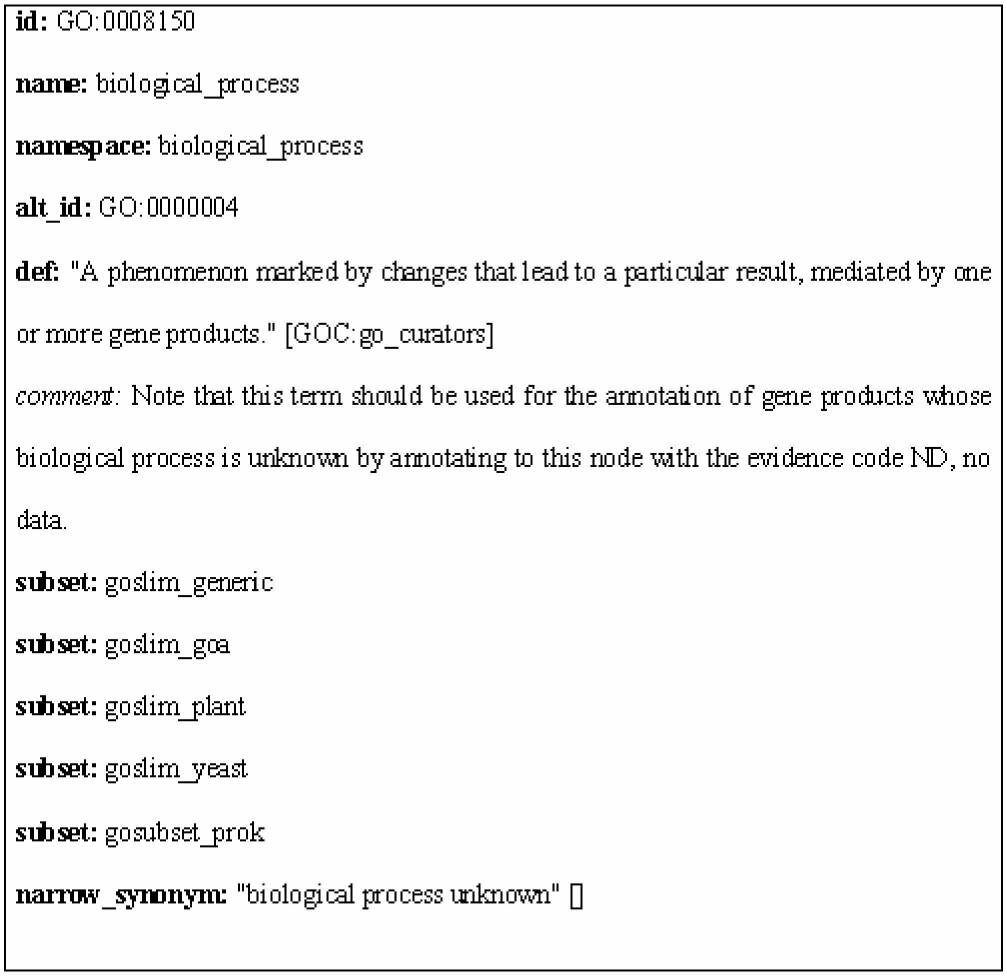
Table 3.2: GO terms for three ontologies for which protein sequences were annotated



Biological (26)	Molecular (29)	Cellular (21)
GO:0005975	GO:0000166	GO:0005576
GO:0006118	GO:0003676	GO:0005578
GO:0006350	GO:0003677	GO:0005615
GO:0006412	GO:0003682	GO:0005622
GO:0006464	GO:0003700	GO:0005634
GO:0006629	GO:0003723	GO:0005635
GO:0006810	GO:0003774	GO:0005654
GO:0006811	GO:0003779	GO:0005730
GO:0006950	GO:0003824	GO:0005737
GO:0007010	GO:0004672	GO:0005739
GO:0007049	GO:0004721	GO:0005764
GO:0007165	GO:0004871	GO:0005768
GO:0007267	GO:0004872	GO:0005773
GO:0007275	GO:0005102	GO:0005783
GO:0007610	GO:0005198	GO:0005794
GO:0008152	GO:0005215	GO:0005829
GO:0008219	GO:0005216	GO:0005840
GO:0008283	GO:0005488	GO:0005856
GO:0009058	GO:0005509	GO:0005886
GO:0009653	GO:0005515	GO:0005929
GO:0009790	GO:0008092	GO:0016023
GO:0015031	GO:0008233	
GO:0016043	GO:0009055	
GO:0016049	GO:0016301	
GO:0030154	GO:0016740	

From the above Table 3.2 we can see that there are 76 GO terms (29 + 21 + 26) being actually used from 130 (total number of GO terms in GOSlim). However, the pre-requisite for the construction of a GO tree is that for any ontology corresponding to a child GO node we must have definitions for the parent GO node.

For example: In Biological process there are proteins annotated for GO:0007582 which is a child of GO:0008150 as can be seen from "*is_a*" identifier in GO:0007582 definition (Table 3.1) given below. However there are no proteins annotated directly to GO:0008150 although its definition is given in GOSlim file. So to build a GO tree for biological process we have to include definitions both for the GO:0008150 and GO:0007582. However this assumption is not valid vice versa. The definition for GO:0008150 can be viewed from Figure 3.1.



id: GO:0008150

name: biological_process

namespace: biological_process

alt_id: GO:0000004

def: "A phenomenon marked by changes that lead to a particular result, mediated by one or more gene products." [GOC:go_curators]

comment: Note that this term should be used for the annotation of gene products whose biological process is unknown by annotating to this node with the evidence code ND, no data.

subset: goslim_generic

subset: goslim_goa

subset: goslim_plant

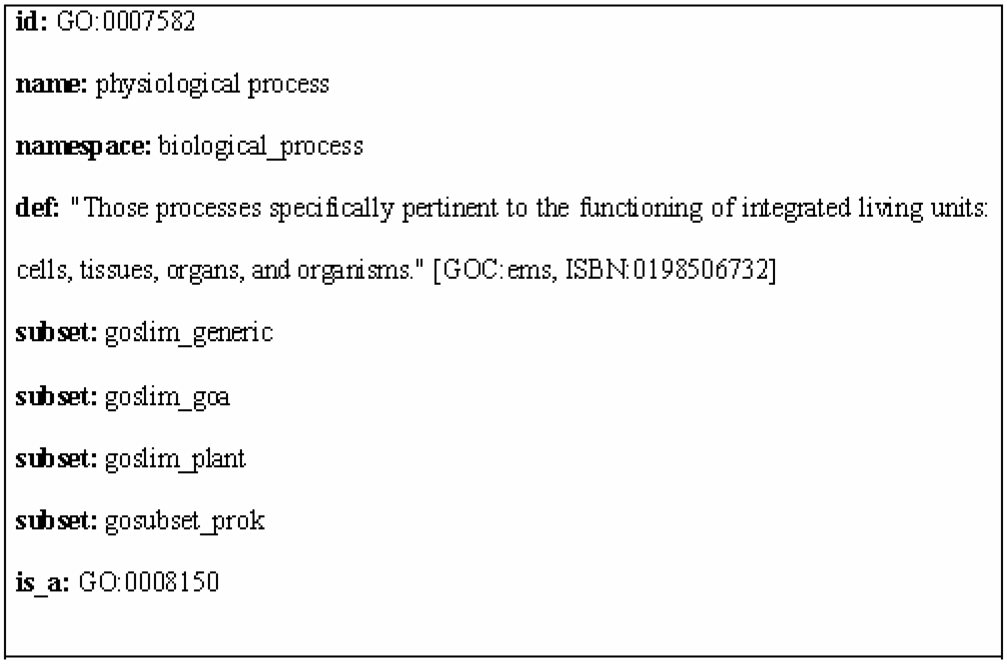
subset: goslim_yeast

subset: gosubset_prok

narrow_synonym: "biological process unknown" []

Figure 3.1 Definition for GO:0008150 in GO slim

GO:0008150 is the root node defined for the biological process as can be seen from its “name:” identifier. GO:0007582 is a child of GO:0008150 and represents the physiological process branch.



id: GO:0007582

name: physiological process

namespace: biological_process

def: "Those processes specifically pertinent to the functioning of integrated living units: cells, tissues, organs, and organisms." [GOC:ems, ISBN:0198506732]

subset: goslim_generic

subset: goslim_goa

subset: goslim_plant

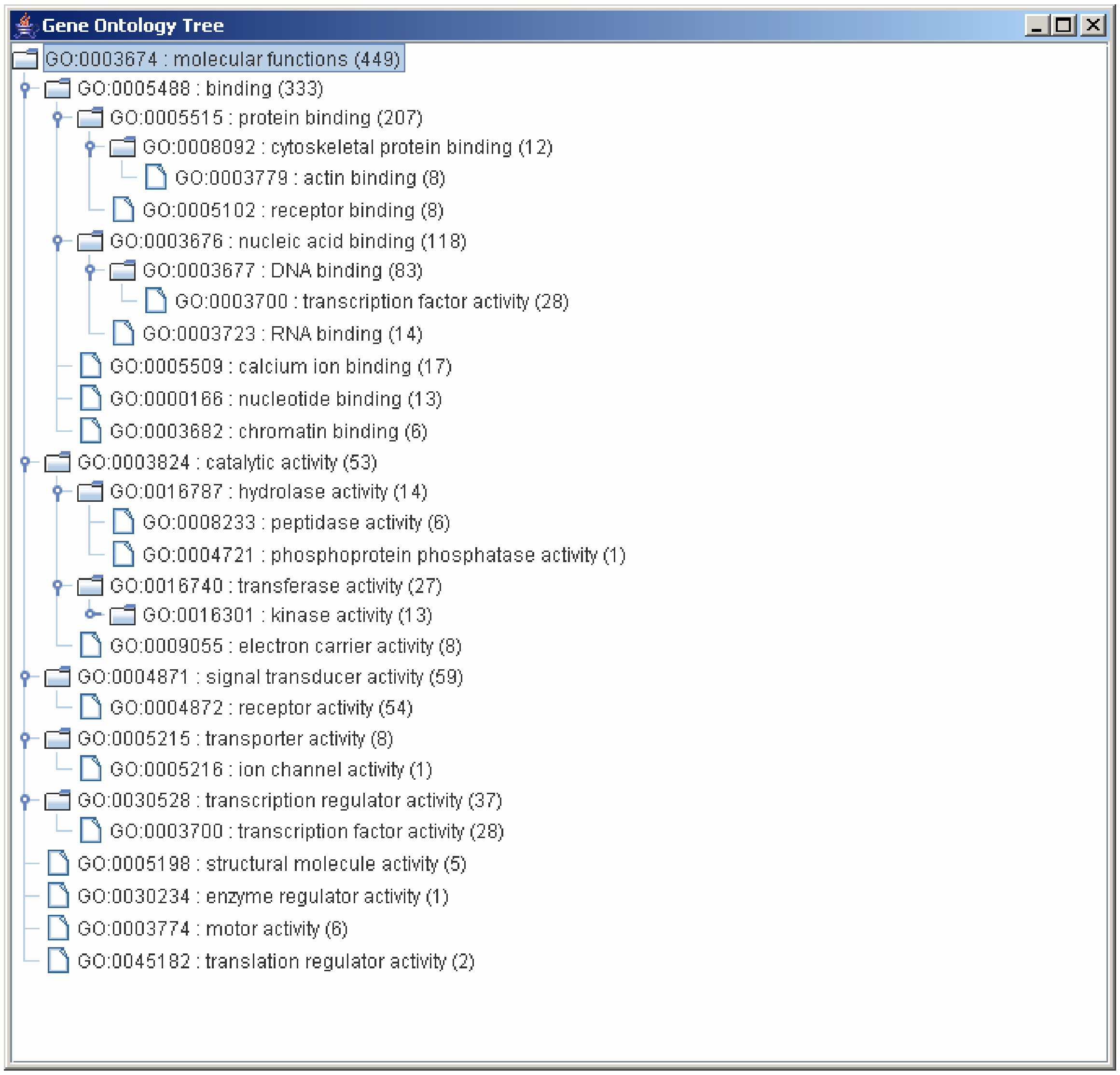
subset: gosubset_prok

is_a: GO:0008150

Figure 3.2 Definition for GO:0007582 in GO slim

All parent-child GO groups relationships were identified for biological process, molecular function and cellular component ontogloies for which protein sequences (from chromosome 1) were annotated. GO trees corresponding to each of these three ontologies were built. GO terms were classified according to their levels which corresponds to the depth it has in the Gene Ontology tree. GO terms are defined along with their functions, parent (if any).

More specifically ontology can be viewed in a simple tree like structure separately for each of the ontologies. The depth of the tree represents the GO level. As seen from Figure 3.3 (next page) molecular function ontology tree have 31 nodes and 5 GO levels with the root node being GO: 0003674. In the molecular function, the GO: ID – 0003674, has 9 child nodes. It represents the elemental activities, such as catalysis or binding and describing the actions of a gene product at the molecular level.

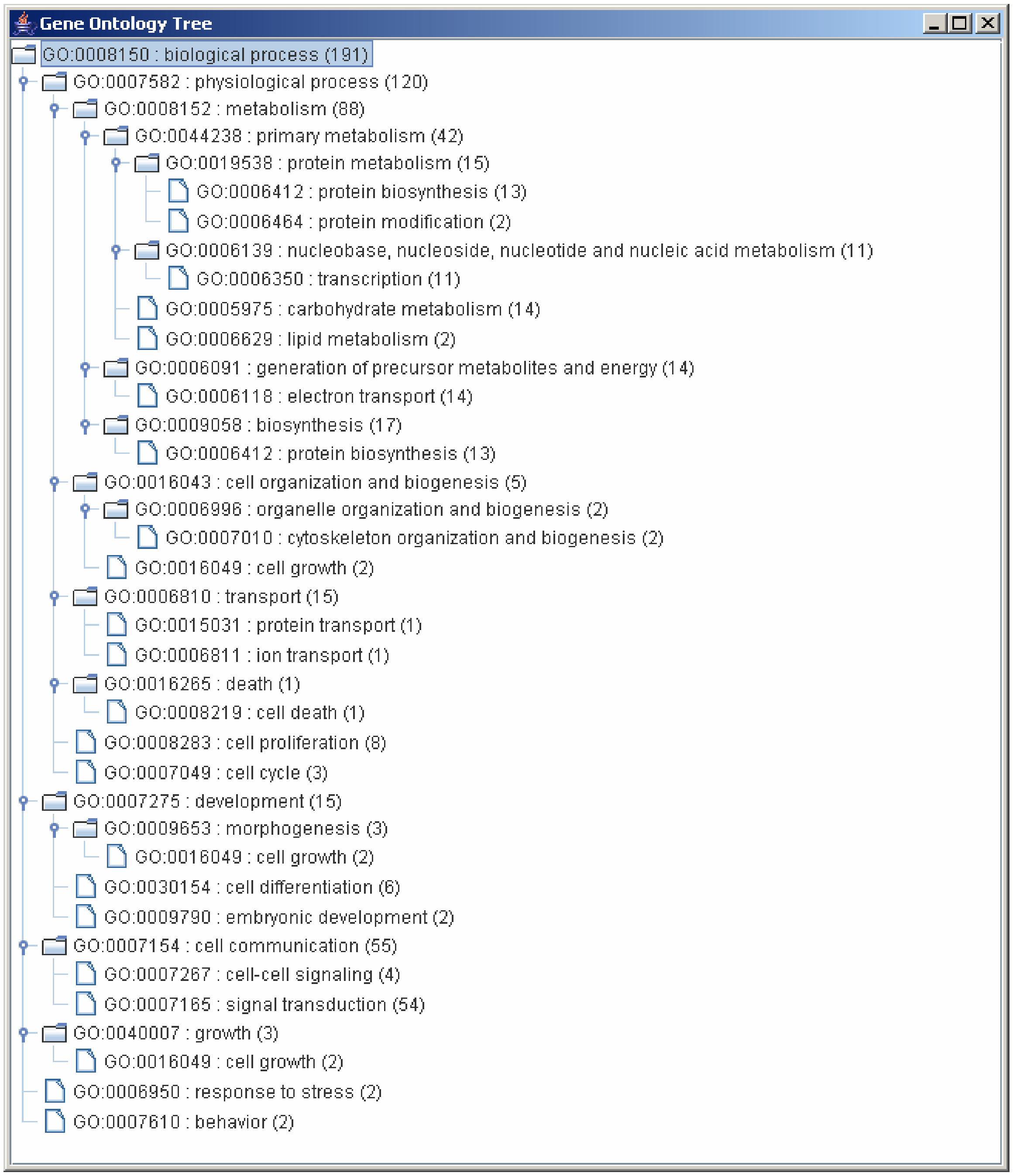


```
GO:0003674 : molecular functions (449)
  GO:0005488 : binding (333)
    GO:0005515 : protein binding (207)
      GO:0008092 : cytoskeletal protein binding (12)
        GO:0003779 : actin binding (8)
      GO:0005102 : receptor binding (8)
    GO:0003676 : nucleic acid binding (118)
      GO:0003677 : DNA binding (83)
        GO:0003700 : transcription factor activity (28)
      GO:0003723 : RNA binding (14)
    GO:0005509 : calcium ion binding (17)
    GO:0000166 : nucleotide binding (13)
    GO:0003682 : chromatin binding (6)
  GO:0003824 : catalytic activity (53)
    GO:0016787 : hydrolase activity (14)
      GO:0008233 : peptidase activity (6)
      GO:0004721 : phosphoprotein phosphatase activity (1)
    GO:0016740 : transferase activity (27)
      GO:0016301 : kinase activity (13)
    GO:0009055 : electron carrier activity (8)
  GO:0004871 : signal transducer activity (59)
    GO:0004872 : receptor activity (54)
  GO:0005215 : transporter activity (8)
    GO:0005216 : ion channel activity (1)
  GO:0030528 : transcription regulator activity (37)
    GO:0003700 : transcription factor activity (28)
  GO:0005198 : structural molecule activity (5)
  GO:0030234 : enzyme regulator activity (1)
  GO:0003774 : motor activity (6)
  GO:0045182 : translation regulator activity (2)
```

Figure 3.3 GO tree (GO slim) for molecular function

At level 3 there are maximum GO: ID's –11, which indicates that this level has the maximum number of functionalities for Molecular function ontology. A given gene product may exhibit one or more molecular functions.

The Biological function Gene Ontology tree, as seen from Figure 3.4, has 6 levels and a total of 38 GO terms. The root node is at GO: 0008150. GO: 0008150 denote a phenomenon which is marked by changes that lead to a particular result, mediated by



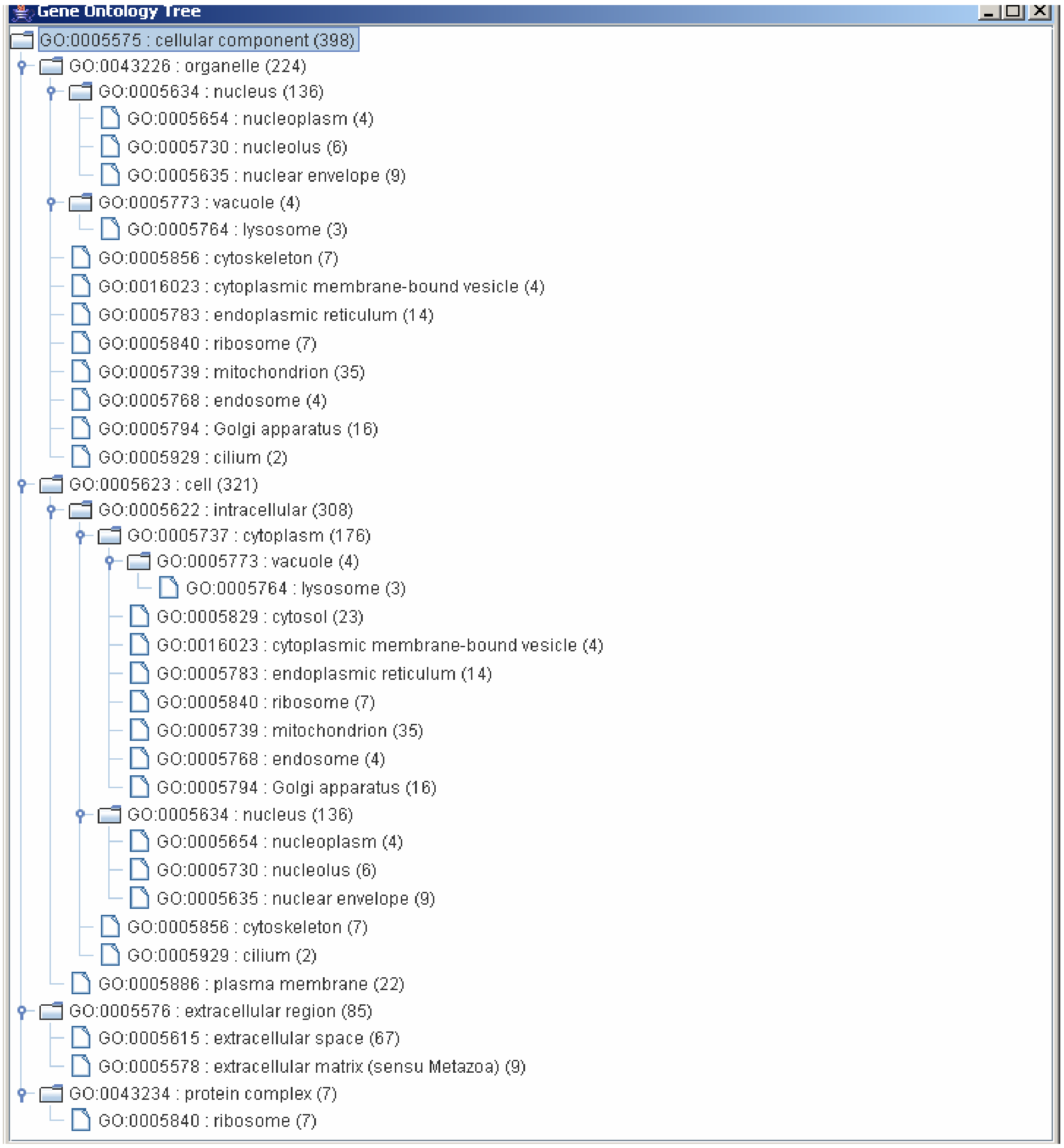
Gene Ontology Tree

```
GO:0008150 : biological process (191)
  GO:0007582 : physiological process (120)
    GO:0008152 : metabolism (88)
      GO:0044238 : primary metabolism (42)
        GO:0019538 : protein metabolism (15)
          GO:0006412 : protein biosynthesis (13)
          GO:0006464 : protein modification (2)
        GO:0006139 : nucleobase, nucleoside, nucleotide and nucleic acid metabolism (11)
          GO:0006350 : transcription (11)
        GO:0005975 : carbohydrate metabolism (14)
        GO:0006629 : lipid metabolism (2)
      GO:0006091 : generation of precursor metabolites and energy (14)
        GO:0006118 : electron transport (14)
      GO:0009058 : biosynthesis (17)
        GO:0006412 : protein biosynthesis (13)
    GO:0016043 : cell organization and biogenesis (5)
      GO:0006996 : organelle organization and biogenesis (2)
        GO:0007010 : cytoskeleton organization and biogenesis (2)
      GO:0016049 : cell growth (2)
    GO:0006810 : transport (15)
      GO:0015031 : protein transport (1)
      GO:0006811 : ion transport (1)
    GO:0016265 : death (1)
      GO:0008219 : cell death (1)
    GO:0008283 : cell proliferation (8)
    GO:0007049 : cell cycle (3)
  GO:0007275 : development (15)
    GO:0009653 : morphogenesis (3)
      GO:0016049 : cell growth (2)
    GO:0030154 : cell differentiation (6)
    GO:0009790 : embryonic development (2)
  GO:0007154 : cell communication (55)
    GO:0007267 : cell-cell signaling (4)
    GO:0007165 : signal transduction (54)
  GO:0040007 : growth (3)
    GO:0016049 : cell growth (2)
  GO:0006950 : response to stress (2)
  GO:0007610 : behavior (2)
```

Figure 3.4 GO tree (GO slim) tree for biological process

one or more gene products. The first level has 1 node, second has 6 nodes, third level has 12 nodes, fourth level has 9 nodes, fifth level has 7 nodes and sixth level has 3 nodes respectively. This means that Biological function ontology has the maximum number of nodes at the third level.

Similarly for cellular component ontology GO tree (Figure 3.5) we see that there



```
Gene Ontology Tree
GO:0005575 : cellular component (398)
  GO:0043226 : organelle (224)
    GO:0005634 : nucleus (136)
      GO:0005654 : nucleoplasm (4)
      GO:0005730 : nucleolus (6)
      GO:0005635 : nuclear envelope (9)
    GO:0005773 : vacuole (4)
      GO:0005764 : lysosome (3)
    GO:0005856 : cytoskeleton (7)
    GO:0016023 : cytoplasmic membrane-bound vesicle (4)
    GO:0005783 : endoplasmic reticulum (14)
    GO:0005840 : ribosome (7)
    GO:0005739 : mitochondrion (35)
    GO:0005768 : endosome (4)
    GO:0005794 : Golgi apparatus (16)
    GO:0005929 : cilium (2)
  GO:0005623 : cell (321)
    GO:0005622 : intracellular (308)
      GO:0005737 : cytoplasm (176)
        GO:0005773 : vacuole (4)
          GO:0005764 : lysosome (3)
        GO:0005829 : cytosol (23)
        GO:0016023 : cytoplasmic membrane-bound vesicle (4)
        GO:0005783 : endoplasmic reticulum (14)
        GO:0005840 : ribosome (7)
        GO:0005739 : mitochondrion (35)
        GO:0005768 : endosome (4)
        GO:0005794 : Golgi apparatus (16)
      GO:0005634 : nucleus (136)
        GO:0005654 : nucleoplasm (4)
        GO:0005730 : nucleolus (6)
        GO:0005635 : nuclear envelope (9)
      GO:0005856 : cytoskeleton (7)
      GO:0005929 : cilium (2)
    GO:0005886 : plasma membrane (22)
  GO:0005576 : extracellular region (85)
    GO:0005615 : extracellular space (67)
    GO:0005578 : extracellular matrix (sensu Metazoa) (9)
  GO:0043234 : protein complex (7)
    GO:0005840 : ribosome (7)
```

Figure 3.5 GOSlim tree for cellular component

are 40 nodes (GO terms) and 6 GO levels. The root node40 nodes (GO terms) and 6 GO levels. The root node is at GO:0005575; 11 nodes at level 5; 8 nodes at level 4; 15 nodes at level 3 and 4 nodes at level 2.

GO: 0043226 lies at the second level have the highest number of child nodes (10). It has got the highest number of subclasses. It represents an organized structure of distinctive morphology and function. This includes the nucleus, mitochondria, plastids, vacuoles, vesicles, ribosomes and the cytoskeleton. It excludes the plasma membrane. For the cellular second level the maximum number of nodes (GO terms) are at the third level.

The local sequence alignments were performed for all-to-all pair-wise annotated proteins using alignment tool downloaded from NCBI for blasting two sequences. The *p*-values were calculated to determine the overall similarity of two protein sequences on (Eq. 2.1). The *p*-value distributions for the protein sequence pairs over certain intervals ($p \in [0, 1]$) are shown in Table 3.3.

Table 3.3 *p*-value distribution for annotated protein sequence pairs

		Percentage of pairs		
p-value range	Percentage of pairs	Biological process	Molecular function	Cellular component
$[1, 10^{-1})$	89.2158	88.2447	89.2290	89.4219
$[10^{-1}, 10^{-2})$	6.9106	6.0292	7.0285	6.963
$[10^{-2}, 10^{-3})$	1.7165	1.3998	1.7658	1.7265
$[10^{-3}, 10^{-4})$	0.5629	0.4574	0.5538	0.5987
$[10^{-4}, 10^{-5})$	0.2134	0.1433	0.1998	0.2468
$[10^{-5}, 10^{-10})$	0.2974	0.2315	0.2645	0.3544
$[10^{-10}, 10^{-15})$	0.0824	0.0276	0.0686	0.1127
$[10^{-15}, 10^{-20})$	0.0450	0.0441	0.0308	0.0633
$[10^{-20}, 10^{-50})$	0.2589	0.7165	0.1969	0.2329

Table 3.3 *p*-value distribution for annotated protein sequence pairs

$[10^{-50}, 10^{-100})$	0.4618	2.1383	0.4514	0.0899
$[10^{-100}, 0]$	0.2352	0.5676	0.2108	0.1899

In Table 3.3, the second column represents the *p*-value of protein sequence pairs from chromosome 1; third column depicts the *p*-value distribution of protein sequence pairs annotated for biological process, fourth column for molecular function, and fifth column for cellular component. We can clearly see that *p*-value distribution of protein sequence pairs for the three ontologies (third, fourth and firth columns) is quite similar with the *p*-value distribution of total number of annotated proteins (second column), indicating that protein sequences annotated for the ontologies is a representative sample set of sequences from chromosome 1. More than 88% of the sequence-pairs lies in the first interval indicating that majority of the pairs are not similar. Only about 5.7% of the sequence pairs have p-values less than 0.01.

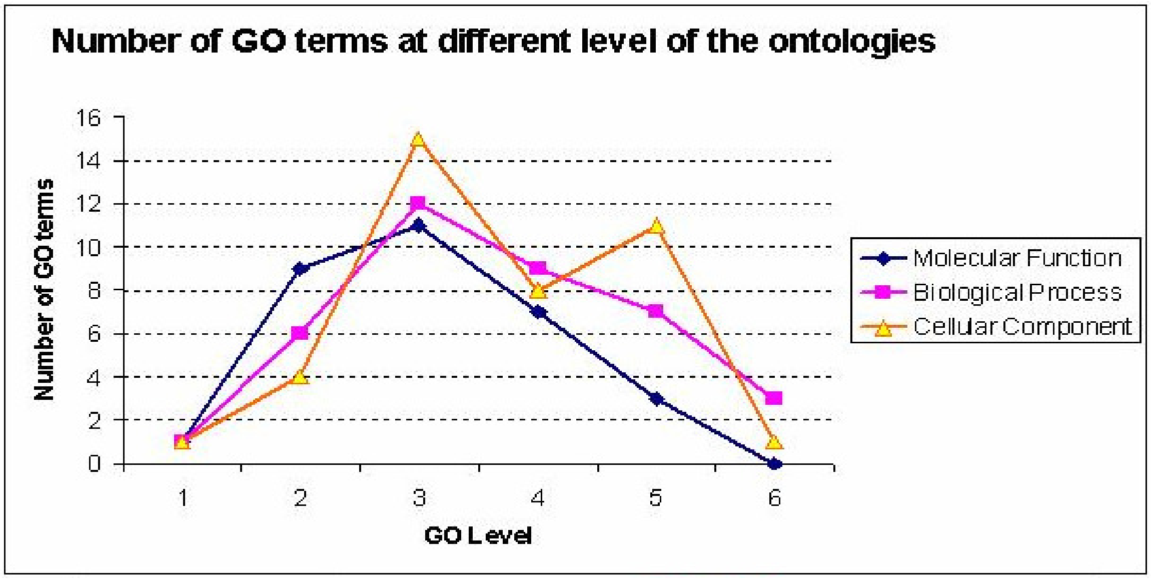


Figure 3.6 Number of GO groups at different levels of ontologies

The distribution of sizes of GO groups across GO levels were also studied (Figure 3.6). We see clearly that the GO groups in all the three ontologies namely; molecular function, cellular component and biological process populate the third level of the ontologies. Molecular level has the maximum GO groups - 11 at the third level, the biological process has the maximum GO groups - 12 at the third level and the cellular process has the maximum GO groups -15 at the third level.

Figure 3.7 (below) depicts the distribution of number of proteins at different levels of ontologies. We see that in all three GO categories the average size of the GO groups decreases in most of the cases as their level increases.

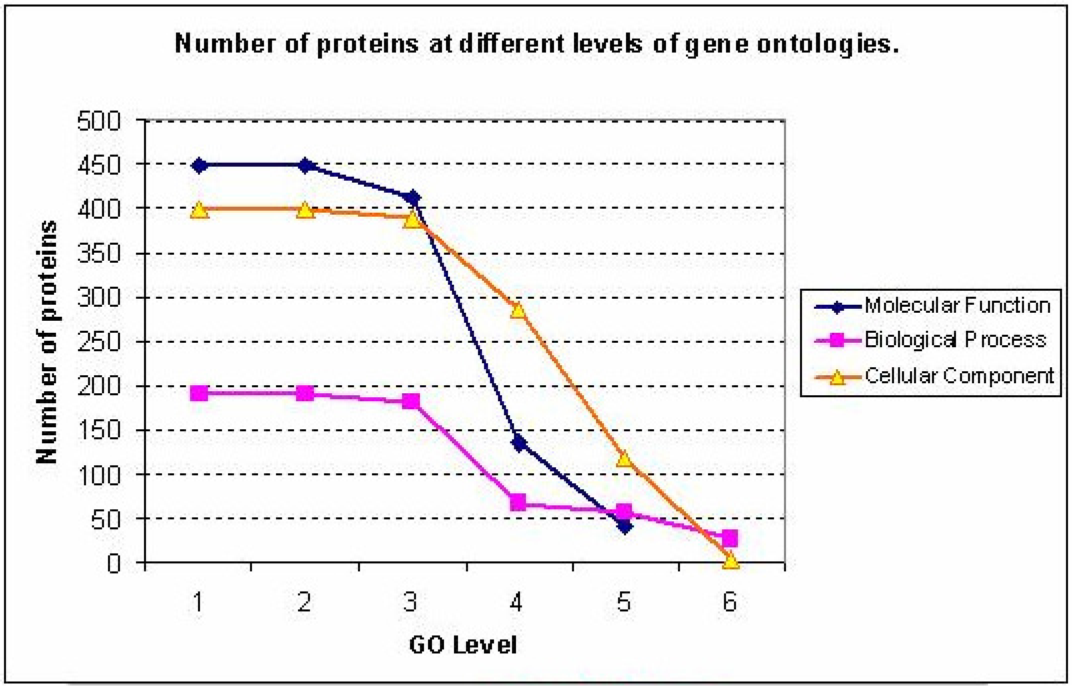


Figure 3.7 Number of proteins across different GO levels

The numbers of proteins for molecular process, biological function and cellular component decreases down. The first level represents the root node, and the subsequent nodes represent the child nodes. The highest numbers of proteins, 449 are in the molecular process, followed by cellular component which has 398 proteins and 191 being in the biological function.

Table 3.4 depicts the *p*-value distribution of protein sequence pairs for annotated for molecular function ontology GO terms at different levels.

Table 3.4 *p*-value distribution of sequence pairs annotated for molecular function

	GO levels				
p value range	1	2	3	4	5
log p <= 0	100.0000	100.0000	100.0000	100.0000	100.0000
log p <= -1	10.77096	13.5106	14.94208	15.5121	16.6271
log p <= -2	3.742444	5.2973	6.703111	6.6558	7.8385
log p <= -3	1.976615	3.0396	4.418847	3.9391	3.8005
log p <= -4	1.422805	2.2932	3.689965	3.1513	3.3254
log p <= -5	1.222956	2.0055	3.400364	2.7710	3.0879
log p <= -10	0.958479	1.6061	2.905766	2.2005	2.8504
log p <= -15	0.889874	1.4995	2.756085	2.1462	2.6128
log p <= -20	0.859052	1.4453	2.804894	2.0103	2.1378
log p <= -50	0.662186	1.1221	2.102043	1.5213	0.9501
log p <= -100	0.210786	0.3588	0.657295	1.1953	0.7126

The curve in the Figure 3.8 (next page) represents the percentages of sequence pairs less than or equal to certain *p*-value plotted across different GO levels. We see clearly that, majority of the sequence pairs, 90 % of them, are considered non-similar across all the GO levels, have at least *p*-values greater than 0.1. However, the percentage of similar sequence pairs does increase steadily with their GO levels. For level 1 around 3.7 % sequence pairs have *p*-values $\leq 10^{-2}$ which increments with the GO level. Level 5 has the highest percentage of similar pairs for molecular function ontology. At this level around 8 % of the sequence pairs have similarity *p*-value less than or equal to 10^{-3}.

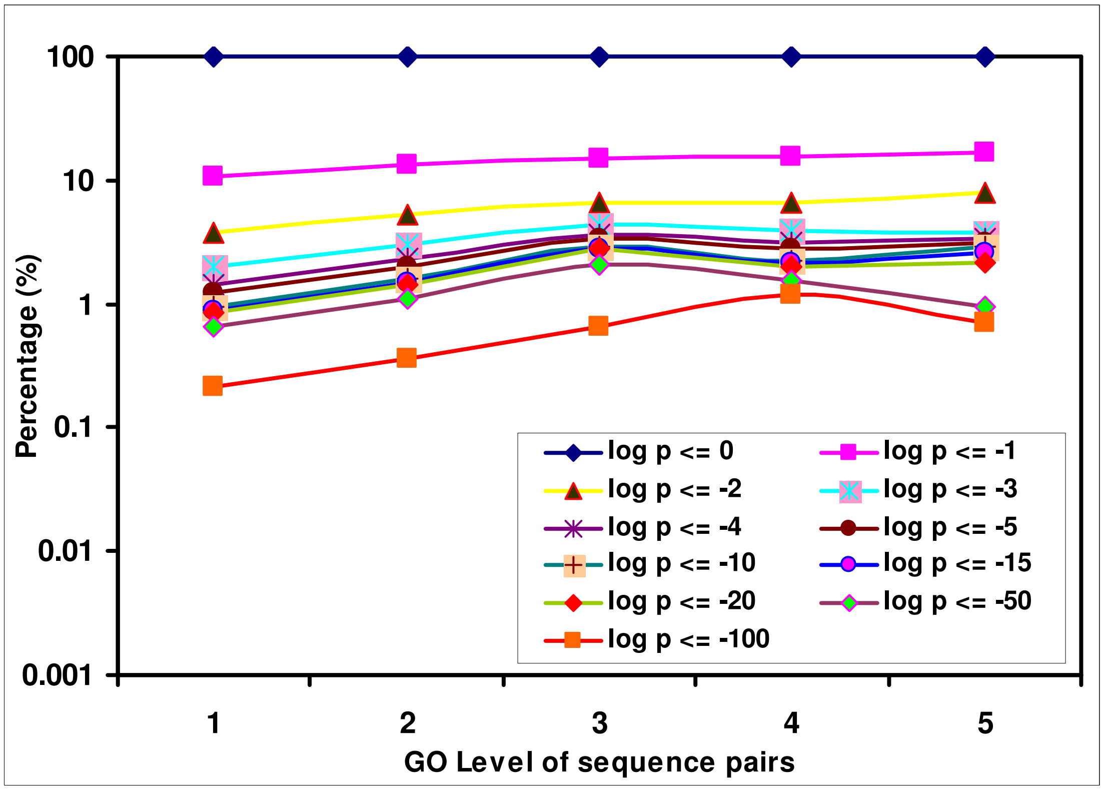


Figure 3.8 *p*-value distribution of sequence pairs annotated for molecular function

In particular, the percentage of pairs with high similarity scores ($p \leq 10^{-10}$) has an increase from level 1 to level 3. The percentage increase is not monotonic for level 4 and 5. At these levels a downward slope is observed, may be due to the nature of the ontology graph in which fewer GO terms are on levels higher than 3 but overall there is a trend of percentage increase with the level.

Table 3.5 *p*-value distribution of sequence pairs annotated for biological process

	GO levels					
P value range	1	2	3	4	5	6
log p <= 0	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000
log p <= -1	11.7553	16.6457	20.6776	16.8246	13.3822	13.0435
log p <= -2	5.7261	10.1088	14.6198	12.7962	7.6994	10.4348
log p <= -3	4.3263	8.4717	12.9442	11.1374	5.8662	8.6957
log p <= -4	3.8688	7.8077	12.1525	10.4265	4.9496	8.6957
log p <= -5	3.7255	7.5673	11.8210	10.1896	4.5830	8.6957

Table 3.5 *p*-value distribution of sequence pairs annotated for biological process

log p <= -10	3.4941	7.1208	11.3239	9.4787	3.8497	6.9565
log p <= -15	3.4665	7.0979	11.3055	9.4787	3.8497	6.9565
log p <= -20	3.4224	7.0063	11.1766	9.2417	3.7580	6.9565
log p <= -50	2.7060	5.5867	8.9486	7.3460	2.9789	4.3478
log p <= -100	0.5676	1.1677	1.8413	5.6872	2.3831	2.6087

Table 3.5 above depicts the *p*-value distributions of sequence pairs annotated for biological process. In particular, the percentage of pairs with high similarity scores ($p \leq 10^{-10}$) increases from 3.5 % for level 1 to 7 % for level 5. There is an increase of 3.49 % of the sequence pairs having *p*-value $\leq 10^{-15}$ from level 1 to level 5. There is a monotonous increase in the percentage of sequence pairs from level 1 to level 3. Level 3 has the highest percentage of similar pairs for biological process ontology.

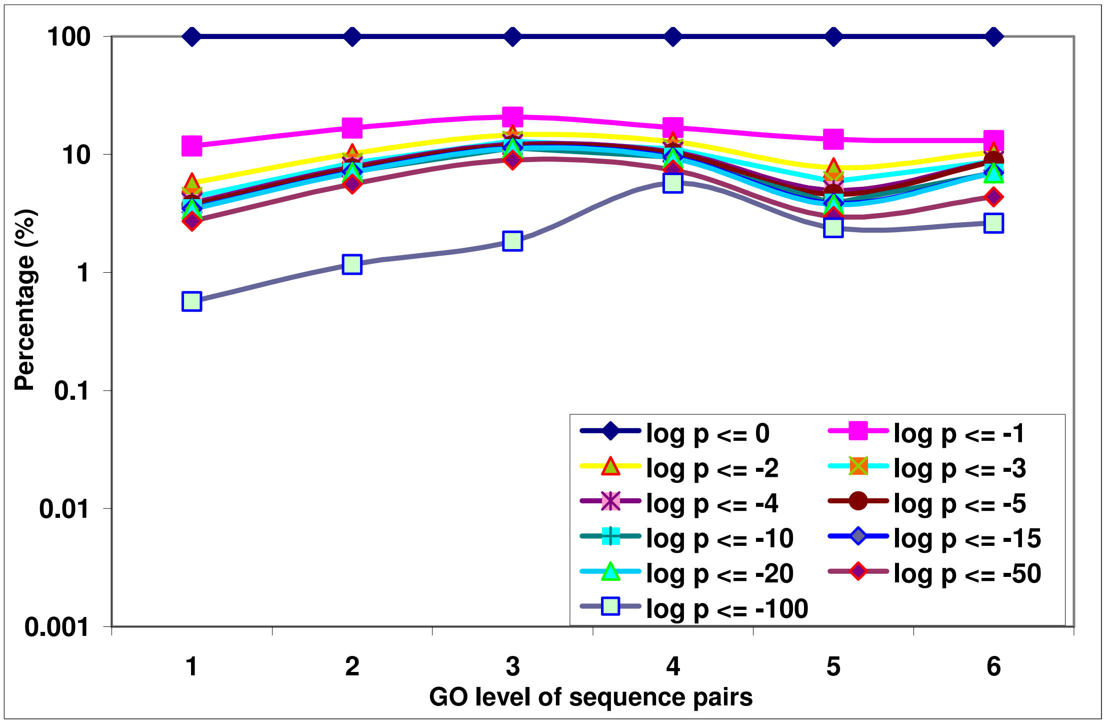


Figure 3.9 *p*-value distribution of sequence pairs annotated for biological process

At this level over 13 % of the sequence pairs have similarity *p*-values less than or equal to 10^{-3} while for level 1 there are over 4.3 % sequences. Figure 3.9 (previous page) shows the curves for the *p*-value distribution of sequence pairs annotated for biological process. As seen clearly the percentage increase is not monotonic from level 4 to level 6. There is a downward slope at level 4 and level 5, may be due to less number of proteins annotated for GO terms at these level. Especially if we see at log $p \leq -100$ ($p \leq 10^{-100}$) curves which represent the maximum similarity % there is a steep increase from level 3 to 4 suggesting that there is a steep increase in the similarity between the protein sequence pairs when we go from level 3 to 4. The level of a GO term is defined as the lowest level on which it appears in the GO Directed Acyclic Graph.

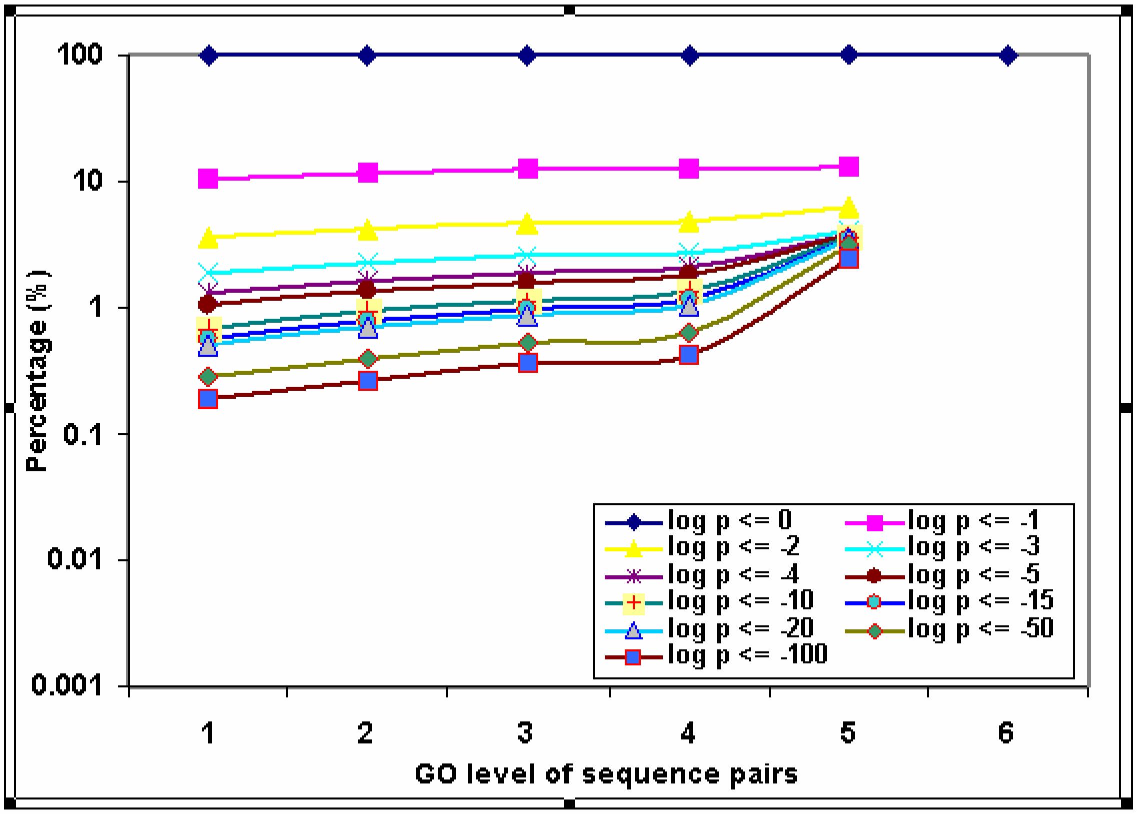


Figure 3.10 *p*-value distribution of sequence pairs annotated for cellular component

Figure 3.10 (previous page) depicts curve for the *p*-value distribution of sequence pairs annotated for cellular component ontology. For the cellular component Ontology, Over 90% (Table 3.6 next page) of the sequence pairs at all levels have *p*-values greater than 0.1. However, the number of similar sequence pairs does increase steadily with their GO levels. In particular, the percentage of pairs with high similarity scores ($p \leq 10^{-10}$) has a steep increase from level 4 to level 5. At level 4 around 2.7 % of the sequence pairs have similarity *p*-value less than or equal to 10^{-3} while for level 5 over 4 % sequence pairs. Level 5 has the highest percentage of similar sequence pairs apparently much higher than molecular function and biological process.

Table 3.6 *p*-value distribution of sequence pairs annotated for cellular component

	GO levels					
p value range	1	2	3	4	5	6
log p <= 0	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000
log p <= -1	10.5781	11.6281	12.4448	12.6145	13.1602	0.0000
log p <= -2	3.6151	4.1525	4.6553	4.8455	6.2338	0.0000
log p <= -3	1.8885	2.2546	2.5893	2.7613	4.0693	0.0000
log p <= -4	1.2898	1.6215	1.9089	2.1288	3.7229	0.0000
log p <= -5	1.0430	1.3437	1.5980	1.8247	3.7229	0.0000
log p <= -10	0.6886	0.9434	1.1266	1.3421	3.6364	0.0000
log p <= -15	0.5759	0.7945	0.9862	1.1759	3.5498	0.0000
log p <= -20	0.5126	0.7106	0.8876	1.0624	3.4632	0.0000
log p <= -50	0.2797	0.3941	0.5232	0.6407	3.1169	0.0000
log p <= -100	0.1899	0.2665	0.3611	0.4298	2.4242	0.0000

Similar to the other two ontologies there is a rise in the percentage of similar sequence pairs from level 1 to level 3 but in particular, there is a significant percentage increase in similar pairs from level 4 to level 5 for cellular component. Overall there is a percentage in increase in similarity of sequence pairs with their GO levels.

The percentage of similarity of sequence pairs across their GO levels was also examined branch wise. Percentage of sequence pairs for each GO group lying in an interval (interval was defined for $p \in [0, 10^{-20}]$) was calculated and the ratio was taken between the parent GO group (second column) and the child GO group (third column):

$$\text{Ratio} = \frac{\%\text{ of sequence pairs in parent GO group lying in interval } (p \in [0,10^{-20}])}{\%\text{ of sequence pairs in child GO group lying in interval } (p \in [0,10^{-20}])}$$

The value for ratio < 1 implied that percentage of similarity in sequence pairs is more in child GO group as compared with its parent GO group for that interval.

Table 3.7 *p*-value analysis for molecular function branch wise

Parent GO term	Child GO term	Ratio (% of similar pairs in parent / % of similar pairs in child)	Count 1 if ratio is < 1
GO:0003674	*GO:0005488	1.7920	
*GO:0005488	**GO:0005509	0.0815	1
*GO:0005488	**GO:0003682	0.0360	1
*GO:0005488	**GO:0003676	0.3309	1
**GO:0003676	***GO:0003677	0.8355	1
***GO:0003677	****GO:0003700	0.9362	1
**GO:0003676	***GO:0003723	0.6591	1
*GO:0005488	**GO:0000166	0.1246	1
*GO:0005488	**GO:0005515	0.7629	1
**GO:0005515	***GO:0008092	0.0518	1
***GO:0008092	****GO:0003779	1.6970	
**GO:0005515	***GO:0005102	0.0440	1
GO:0003674	*GO:0003824	1.1838	
*GO:0003824	**GO:0009055	0.0508	1
*GO:0003824	**GO:0016787	0.6609	1
**GO:0016787	***GO:0008233	0.1647	1
*GO:0003824	**GO:0016740	0.6368	1
GO:0003674	*GO:0003774	0.0430	1
GO:0003674	*GO:0004871	0.0263	1
*GO:0004871	**GO:0004872	0.8364	1
GO:0003674	*GO:0030528	0.5201	1
*GO:0030528	**GO:0003700	0.8919	1
GO:0003674	*GO:0005215	0.0401	1

Note: GO term with no '*' represents root node (level 1).

GO term with one '*' represents node at level 2.

GO term with two '*' represents node at level 3.

GO term with three '*' represents node at level 4.

In particular, the percentage of pairs with high similarity scores with *p*-value lying in interval $p \in [0, 10^{-20}]$ was studied. The ratio of the parent-child GO term relationship was taken only once for a branch.

As observed from the above table there is general trend of increase in the similarity of sequence pairs while traversing down the GO levels branch wise. Among 23 parent-child GO group pairs, there is a short trend of decreasing similarity from parent to child GO group, and only 3 pairs have ratio > 1. This may be due to few number of protein sequences annotated for that child GO group. Overall with the increase in the GO levels there is a general increase in the similarity of the sequence pairs as we move down the GO tree branch wise. On the same parallel lines we examined the *p*-value distribution of the sequence pairs for cellular component.

Table 3.8 *p*-value analysis for cellular component branch wise

Parent GO term	Child GO term	Ratio (% of similar pairs in parent / % of similar pairs in child)	Count 1 if ratio is < 1
GO:0005575	*GO:0005623	0.8358	1
*GO:0005623	*GO:0005622	0.9966	1
**GO:0005622	***GO:0005929	####	No count
**GO:0005622	***GO:0005737	0.9574	1
***GO:0005737	****GO:0016023	0.0386	1
***GO:0005737	****GO:0005829	0.1479	1
***GO:0005737	****GO:0005783	0.1463	1
***GO:0005737	****GO:0005768	0.0386	1
***GO:0005737	****GO:0005794	0.1286	1
***GO:0005737	****GO:0005739	0.5465	1

Table 3.8 *p*-value analysis for cellular component branch wise

***GO:0005737	****GO:0005840	0.135	1
***GO:0005737	****GO:0005773	####	No count
****GO:0005773	*****GO:0005764	####	No count
**GO:0005622	***GO:0005856	0.0431	1
**GO:0005622	***GO:0005634	0.3742	1
***GO:0005634	****GO:0005635	0.0658	1
****GO:0005635	****GO:0005730	####	No count
****GO:0005730	****GO:0005654	####	No count
GO:0005575	**GO:0005886	0.0987	1
GO:0005575	*GO:0005576	0.3156	1
*GO:0005576	**GO:0005578	0.0266	1
*GO:0005576	**GO:0005615	1.1588	
GO:0005575	*GO:0043226	0.6601	1
*GO:0043226	**GO:0005929	####	No count
*GO:0043226	**GO:0016023	0.0466	1
*GO:0043226	**GO:0005856	0.0544	1
*GO:0043226	**GO:0005783	0.1767	1
*GO:0043226	**GO:0005768	0.0466	1
*GO:0043226	**GO:0005794	0.1553	1
*GO:0043226	**GO:0005739	0.6602	1
*GO:0043226	**GO:0005634	0.4722	1
**GO:0005634	***GO:0005635	0.0658	1
**GO:0005634	***GO:0005730	####	No count
**GO:0005634	***GO:0005654	####	No count
*GO:0043226	**GO:0005840	0.1631	1
*GO:0043226	**GO:0005773	####	No count
**GO:0005773	***GO:0005764	####	No count
GO:0005575	*GO:0043234	0.1077	1
*GO:0043234	**GO:0005840	1	1

Note: #### denotes 'un-defined' value

In the above Table 3.8 *p*-value distributions of sequence pairs for cellular component had been listed out as a relationship between parent-child GO term groups for *p* values lying in interval $p \in [0, 10^{-20}]$. Interestingly, cellular component ontology has 10 parent-child GO group pairs (ratio as "####") for which nothing can be said. This may be due to there were no protein sequences annotated for this interval for parent-child GO groups or child GO group alone so nothing can be stated for these 10 pairs. Out of the remaining 29 parent-child GO group pairs, noticeably 27 of them have ratio < 1 which implies that there is more similarity in sequence pairs as we go down the GO level branch wise. Only 1 pair has a ratio > 1 and the other 1 has ratio = 1. This implies that there is a general trend in the increase in the similarity as we traverse cellular component GO tree branch wise.

We also examined percentage of similarity between the sequences pairs for biological process (Table 3.9)

Table 3.9 *p*-value analysis for biological process branch wise

Parent GO term	Child GO term	Ratio (% of similar pairs in parent / % of similar pairs in child)	Count 1 if ratio is ≤ 1
GO:0008150	*GO:0007610	0.0342	1
GO:0008150	*GO:0007154	0.0959	1
*GO:0007154	**GO:0007267	0.7138	1
*GO:0007154	**GO:0007165	0.9636	1
GO:0008150	*GO:0007275	3.5934	
*GO:0007275	**GO:0030154	0.1429	1
*GO:0007275	**GO:0009790	####	No count
*GO:0007275	**GO:0009653	####	No count
**GO:0009653	***GO:0016049	####	No count
GO:0008150	**GO:0040007	####	No count
**GO:0040007	**GO:0016049	####	No count
GO:0008150	*GO:0007582	3.0544	
*GO:0007582	**GO:0007049	####	No count
*GO:0007582	**GO:0016043	####	No count
**GO:0016043	***GO:0016049	####	No count
**GO:0016043	***GO:0006996	####	No count
***GO:0006996	****GO:0007010	####	No count
*GO:0007582	**GO:0008283	0.1046	1
*GO:0007582	**GO:0008152	0.727	1
**GO:0008152	***GO:0009058	0.524	1
***GO:0009058	****GO:0006412	1.147	
**GO:0008152	***GO:0006091	0.1753	1

Table 3.9 *p*-value analysis for biological process branch wise

***GO:0006091	****GO:0006118	1	
**GO:0008152	***GO:0044238	0.4576	1
***GO:0044238	****GO:0005975	0.1613	1
**GO:0008152	***GO:0044238	0.4576	1
***GO:0044238	****GO:0006629	0.0337	1
***GO:0044238	****GO:0006139	0.3088	1
****GO:0006139	*****GO:0006350	0.7855	1
***GO:0044238	****GO:0019538	1.1788	
****GO:0019538	*****GO:0006412	1.1143	
****GO:0019538	*****GO:0006464	0.0286	1
*GO:0007582	**GO:0006810	0.107	1
GO:0008150	**GO:0006950	####	No count

Note: #### denotes 'un-defined' value

There are 17 parent-child GO group pairs which have ratio > 1, implies that there is an increase in similarity of sequence pairs as we go down the GO tree but as such there is a vague trend of increasing degree of similarity with the GO levels. As we can see that there are 11 pairs for which no inference can be drawn, 1 pair has ratio = 1 and 5 pairs have ratio > 1.

These above results suggest that proteins of similar biological functions tend to have higher sequence similarity. In general, we see, from the three *p*-value distribution tables that the deeper a GO group is the more similarity it will have. More convincingly, 20 out of 23 molecular function groups, 27 out of 39 cellular component groups, and 17 out of 34 biological process groups have higher percentage of sequence pairs having *p*-values $\leq 10^{-20}$ than those of their parents. This result indicates the strong correlation between sequence similarity and function similarity.

To actually determine the relationship between similarity between two protein sequences and its contribution in predicting the biological function we computed the posterior probabilities using Bayes' theorem [45] (Duan et al) for acid binding branch of the molecular function ontology tree.

$$P(S_2 \in G \mid S_1 \in G, p(S_1, S_2) \le \varepsilon) = \frac{\mathrm{P}\ (\mathrm{S}_2 \in \mathrm{G}, \mathrm{S}_1 \in \mathrm{G}, \mathrm{p}(\mathrm{S}_1, \mathrm{S}_2) \le \varepsilon\ )}{\mathrm{P}\ (\mathrm{S}_1 \in \mathrm{G}, \mathrm{p}(\mathrm{S}_1, \mathrm{S}_2) \le \varepsilon\ )} \qquad \text{(Eq. 3.1)}$$

Where,

L.H.S. of the equation describes the posterior probability of finding a random sequence S_2 in GO group *G* when S_1 is already present in the same GO group *G* with a *p*-value between S_1 and S_2 greater than or equal to a certain threshold value .

G denotes a particular GO group.

p (S_1, S_2) is the *p*-value calculated as Equation 2.1 (materials & methods section)

ε represents a particular value for which *p*-value threshold for S_1 and S_2 .

Posterior probabilities for molecular function, biological process and cellular component were calculated branch wise for intervals ($p \le$ [0, 1]) (using Eq. 2.2).

Table 3.10 Posterior probability for a molecular function's branch.

GO ID	Log p value threshold intervals										
	≤ 0	≤-1	≤-2	≤-3	≤-4	≤-5	≤-10	≤-15	≤-20	≤-50	≤-100
GO:0003674 (molecular function)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
*GO:0005488 (binding)	58.87	71.72	78.01	82.94	85.68	86.93	89.60	91.03	90.75	93.84	95.57
**GO:0003676 (nucleic acid binding)	15.02	19.28	23.09	31.48	40.68	49.28	66.67	76.26	80.00	88.51	91.43
***GO:0003677 (DNA binding)	10.07	13.78	18.03	24.85	35.64	44.10	59.63	64.29	66.29	69.23	69.81
****GO:0003700 (transcription factor activity)	3.11	4.21	5.85	8.28	16.42	20.83	36.00	36.36	38.89	33.33	40.00

Table 3.10 above represents the acid binding branch (GO:0005488) of molecular function ontology. We can see clearly that posterior probability of a correct assignment varies greatly from group to group. For example, if a database search hits the acid binding group then one can have 95% confidence that the query sequence belongs to *G*, for *p*-value lying in the interval $\log p \in (-100, 0]$. Noticeably, on the other hand a hit corresponding to transcription factor activity group would have 40% confidence that the protein belongs to the group for the same p-value interval. With the distribution of posterior probabilities of the GO groups over a wider range, indicates that posterior probability could serve as one of the features in determining unknown function prediction.

CHAPTER IV

CONCLUSION

There is an association between the similar protein sequences of M. Musculus (chromosome 1 [20]) and the functions pertinent to the three fundamental principles of gene ontology -- molecular function, biological process and cellular component [12]. The similarity distribution curves for the three ontologies indicate that the protein pairs have higher sequence similarity when picked up from the same functional GO group rather than a sample randomly drawn from a pool of protein sequence pairs. The range of *p*-value distribution of the sequence pairs is wide and varies across the groups. Interestingly, as we traverse down the GO levels branch-wise, we find that the *p*-values consistently decreases indicating the protein sequences in a child node are more similar than those in the parent node. Furthermore, our study on posterior probability of a correct prediction indicates that the protein function prediction confidence increases steady with the decrease of *p*-values. These studies done in conjunction with the yeast study by Duan et al [19] suggest that sequence similarity approach can play a vital role as a preliminary tool in the prediction of protein functions. These results need to be validated through other complimentary approaches, encompassing different features including gene ontology structure attributes, gene expression patterns, protein structure similarity, etc. The complimentary approaches are more likely to provide a much accurate function prediction

REFERENCES

[1] WIKI cited; http://en.wikipedia.org/wiki/Genome_project

[2] http://www.ornl.gov/sci/techresources/Human_genome/project/about.shtml

[3] Yu Zheng, R.J.R., and Simon Kasif, Genomic functional annotation using co- evolution profiles of gene clusters. Genome Biol., 2002. 3(11): p. research 0060.1– research 0060.9.

[4] John A. Gerlt, P.C.B., Can sequence determine function? Genome Biology, 2000. 1: p. reviews0005.1-0005.10.

[5] Hudak J, M.M., A comparative analysis of computational motif-detection methods. Pac Symp. Biocomput., 1999: p. 138-49.

[6] http://www.pubmedcentral.nih.gov/articlerender.fcgi?tool=pubmed&pubmedid=12429059

[7] Pellegrini M., M.E., Thompson M.J., Eisenberg D., Yeates T.O., Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. Proc Natl Acad Sci U S A., 1999. 96(8): p. 4285-8.

[8] T. F. SMITH, M.S.W., Identification of Common Molecular Subsequences. J Mol Biol. 147(1): p. 195-7 7265238

[9] NCBI - BLAST. cited; Available from: http://www.ncbi.nlm.nih.gov/blast/

[10] Altschul, S.F., W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, Basic local alignment search tool. J. Mol. Biol., 1990. 215: p. 403-410.

[11] http://www.accessexcellence.org/AE/AEPC/NIH/gene03.html

[12] http://www.geneontology.org/GO.doc.shtml

[13] http://www.geneontology.org/GO.slims.shtml

[14] Arunachalam Vinayagam RK, Jutta Moormann, Falk Schubert, Roland Eils, Karl-Heinz, Glatting and Sándor Suhai: Applying Support Vector Machines for the Gene ontology based gene function prediction. BMC Bioinformatics 2004, 5(116), doi:10.1186/1471-2105-5-116.

[15] Zehetner, G., OntoBlast function: from sequence similarities directly to potential functional annotations by ontology terms, Nucleic Acids Research, 2003.

[16] Hanqing Xie, A.W., Zurit Levine, Amit Novik, Vladimir Grebinskiy, Avi Shoshan and Liat Mintz, Large-Scale Protein Annotation through Gene Ontology Genome Research, 2002. 12(5): p. 785-794.

[17] Schug J DS, Mazzarelli J, Brunk BP, Stoeckert CJ Jr.: Predicting gene ontology functions from ProDom and CDD protein domains. PubMed, 12(4):648-655.

[18] HouseMouse, E. Explore the Mus musculus genome. [cited; Available from: www.ensembl.org/Mus_musculus/]

[19] Zhong-Hui Duan BH, Lothar Reichel, Dianne M Perez and Ting Shi: The relationship between protein sequences and their gene ontology functions. BMC Bioinformatics 2006, 7(Suppl4):S11. doi: 10.1186/1471-2105-7-S4-S11

[20] Cited from http://www.ebi.ac.uk/integr8/FtpSearch.do?orgProteomeId=59

[21] http://lectures.molgen.mpg.de/Variants/LocalAli/index.html

[22] http://gel.ym.edu.tw/~chc/AB_papers/03.pdf

[23] http://cse.stanford.edu/class/sophomore-college/projects-00/computers-and-the-hgp/smith_waterman.html

[24] http://www.cs.uakron.edu/~zduan/BioinformaticsSummerWorkshop/

[25] http://bioinformatics.unc.edu/docs/tutorial_fasta/fasta.doc

[26] www.ncbi.nlm.nih.gov/Education/BLASTinfo/tut1.html

[27] http://en.wikipedia.org/wiki/BLAST

[28] http://www.lacim.uqam.ca/~anne/BIF7000/Pertsemlidis.pdf

[29] http://www.broad.mit.edu/annotation/blast/blast_help.html

[30] http://www.cas.org/express/help/601/blast/topics/settings.htm

[31] Henikoff, S.H., J. G., Amino acid substitution matrices from protein blocks. Proc Natl Acad Sci U S A, 1992. 89(22): (10915–10919)

[32] Dayhoff, M.O., Schwartz,R.M. and Orcutt,B.C. , Atlas of Protein Sequence and Structure. Dayhoff, M.O.(ed.) 1978

[33] http://selab.janelia.org/publications/Eddy-ATG2/Eddy-ATG2-reprint.pdf

[34] http://www.egeen.ee/u/vilo/edu/2003-04/ATABI_2004k/Lists/blosum.html

[35] http://helix.biology.mcmaster.ca/721/distance/node10.html

[36] http://www.ncbi.nlm.nih.gov/Class/NAWBIS/Modules/Similarity/simsrch16.html

[37] http://bioweb.pasteur.fr/seqanal/interfaces/bl2seq.html

[38] http://biowulf.nih.gov/apps/blast/doc/bl2seq.html

[39] http://web.csb.ias.edu/blast/blast.txt.

[40] http://bioinformatics.ubc.ca/resources/workshops/web_based_blast/tutorial.

[41] http://en.wikipedia.org/wiki/Gene_Ontology

[42] www.geneontology.org/GO.slims.shtml

[43] http://www.perl.com/pub/q/documentation

[44] www.ibm.com/developerworks/linux/library/l-perl-parsing

[45] http://en.wikipedia.org/wiki/Bayes'_theorem

APPENDICES

APPENDIX A

CRITICAL SOURCE CODE

```
#############################################################################
#                                   protein_count.pl                                    #
# Script counts the number of unique proteins in each of the genome component for any #
# organism. Place the name of the genome component i.e. “*.dat” file as input and         #
# number of unique proteins.                                                            #
#############################################################################

#!/usr/bin/perl
#use strict;

open INPUT, "< Musculus_1.dat" or die "cannot open file Musculus_1.dat $!";
$count=0;

# checks for each entry line whether it has an identifier ‘ID’ & ‘AC’

foreach $line(<INPUT>)
{
   @item = split (/\s+/, $line);
    $key1 = shift @item;
    $key2 = shift @item;

    if($key1 eq 'ID')
    {
      $flag=1111;
     }

    if ($key1 eq 'AC' && $flag==1111)
     {
       $flag=9999;
       $count++;
     }
 }

print  "Number of Protein Sequences: ".$count;
```

```
############################################################################
#                              extract_proteins.pl                          #
# Script extracts sequences for each protein and stores it into individual files and also     #
# stores the name of the proteins in ‘values.txt’ file so that number of proteins extracted #
# can be verified against the first script too.
############################################################################

#!/usr/bin/perl
#use strict;

open INPUT, "< Musculus_1.dat" or die "cannot open file Musuculus_1.dat $!";

$sequence = '';
$SQ_start = FALSE;

open  EV, ">>$values.txt" or die  "Cannot open the file!";

foreach $line(<INPUT>)
 {
   @item = split (/\s+/, $line);
   $key1 = shift @item;
   $key2 = shift @item;

   if ($key1 eq 'AC')
   {
      $accession_num = $key2;
      $accession_num =~ s/;//g;
      $fname = 'Protein_Sequences/'.$accession_num.'.txt';
      print " $accession_num\n";
      print EV  "$accession_num\n";
      open SQ_OUTPUT, "> $fname" or die "cannot open file $fname; $!";
   }

  if ($key1 eq 'SQ' && $key2 eq 'SEQUENCE')
  {
      $SQ_start = TRUE;
   }

   if ($key1 eq ''&& $SQ_start eq TRUE) \
   {
      $line =~ s/\s+//g;
      $sequence .= $line."\n";
    }

   if ($key1 eq '//' && $key2 eq '')
```

```
    {
      print SQ_OUTPUT '>', $accession_num,"\n";
      print SQ_OUTPUT $sequence;
      #print   $accession_num,"\n";
      #print  $sequence;
      $SQ_start = FALSE;
      $sequence='';
      close SQ_OUTPUT;
    }
}
close INPUT;

###############################################################################
#                              blast_proteins.pl                              #
# Blasts the proteins against each other. Picks up the name of each protein from #
# ‘values.txt’ file and then picks up the sequences associated with it from its particular #
# sequences text file, uses a blast.exe program and compares and stores the result in #
# text files.
###############################################################################

#!/usr/bin/perl
#use strict;

open(INFO,'values.txt');
@lines1=<INFO>;
@checks=@lines1;

$count=0;

foreach $line1(@lines1)
{
   @item1 = split (/\s+/, $line1);
    $key1=shift @item1;
    $count++;
    print "$line1:\n";
    print "$count\n";

   foreach $check(@checks)
    {
     @item2 = split (/\s+/, $check);
     $key2=shift @item2;

      if($key1 ne $key2)
      {
      @lines = `bl2seq -i Protein_Sequences/$key1.txt -j Protein_Sequences/$key2.txt -p blastp`;
      $result=$key1.'_query';
      open  EV, ">>Result/$result.txt" or die  "Cannot open Result/$result.txt!";
      print EV "$key1 - ";
      print EV "$key2\n";


foreach $line(@lines)
{
 $pattern1="Score";
 $pattern2="Identities";

 if ($line =~ m/$pattern1/i)
 {
   print EV  " $line\n";
  }

 if ($line =~ m/$pattern2/i)
 {
   print EV  " $line\n";
 }
}
print EV "-----------------------------------------------\n";
 }
 }
}
}


##############################################################################
#                                   GO_Slim.pl                                #
# Checks out GO term from GO Slim file and classifies it into Biological process,  #
# cellular component and molecular function ontologies                       #
##############################################################################

#!/usr/bin/perl
#use strict;

open(INFO2,'goslim_generic.obo');
@myarray3=<INFO2>;

$fname1 = 'Total_GO_Molecular.txt';
open EV1, ">>$fname1" or die "cannot open file $fname1; $!";

$fname2 = 'Total_GO_Cellular.txt';
open EV2, ">>$fname2" or die "cannot open file $fname2; $!";
```

```
$fname3 = 'Total_GO_Biological.txt';
open EV3, ">>$fname3" or die "cannot open file $fname3; $!";

$temp;
$count4=0;

print "\nList of GO terms \n";
foreach $g(@myarray3)
 {
       @item_3 = split (/\s+/, $g);
       $key1_1 = shift @item_3;
       $key2_1 = shift @item_3;

        if($key1_1 eq 'id:')
        {
         $temp=$key2_1;
         $count4++;
         print "$key2_1 - $count4 \n";
        }

        if($key1_1 eq 'namespace:')
        {
          if($key2_1 eq 'molecular_function')
           {
             print EV1 "'$temp;', ";
             $count1++;
             $count++;
           }

         if($key2_1 eq 'cellular_component')
           {
             print EV2 "'$temp;', ";
             $count2++;
             $count++;
           }

         if($key2_1 eq 'biological_process')
           {
             print EV3 "'$temp;', ";
             $count3++;
             $count++;
           }
        }
    }
```

```
$count5=0;

foreach $g(@myarray3)
 {
  @item_3 = split (/\s+/, $g);
  $key1_1 = shift @item_3;
  $key2_1 = shift @item_3;

 if($key1_1 eq '[Term]')
 {
  $count5++;
 }
}

print "\n\n\nGO terms for Molecular function - "."$count1 \n";
print "GO terms for Cellular component - "."$count2 \n";
print "GO terms for Biological process - "."$count3 \n";
print "Total GO terms - "."$count5\n";


##############################################################################
#                                   Bio.pl                                   #
# Picks up GO term from Biological process and maps the proteins from dataset for   #
# each GO term and also counts the proteins for each of them                 #
##############################################################################

#!/usr/bin/perl
#use strict;

@myarray3= ('GO:0000003;', 'GO:0005975;', 'GO:0006091;', 'GO:0006118;',
'GO:0006139;', 'GO:0006259;', 'GO:0006350;', 'GO:0006412;', 'GO:0006464;',
'GO:0006519;', 'GO:0006629;', 'GO:0006810;', 'GO:0006811;', 'GO:0006950;',
'GO:0006996;', 'GO:0007005;', 'GO:0007010;', 'GO:0007028;', 'GO:0007049;',
'GO:0007154;', 'GO:0007165;', 'GO:0007267;', 'GO:0007275;', 'GO:0007582;',
'GO:0007610;', 'GO:0008037;', 'GO:0008150;', 'GO:0008152;', 'GO:0008219;',
'GO:0008283;', 'GO:0009056;', 'GO:0009058;', 'GO:0009605;', 'GO:0009607;',
'GO:0009628;', 'GO:0009653;', 'GO:0009719;', 'GO:0009790;', 'GO:0015031;',
'GO:0016032;', 'GO:0016043;', 'GO:0016049;', 'GO:0016265;', 'GO:0019538;',
'GO:0019725;', 'GO:0019748;', 'GO:0030154;', 'GO:0040007;', 'GO:0040029;',
'GO:0044238;', 'GO:0044403;', 'GO:0050789;');


open(INFO4,'Musculus_1.dat');
@myarray4=<INFO4>;
```

```
$fname = 'GOCHECK_BIO_DUPLICATE.txt';
open EV, ">>$fname" or die "cannot open file $fname; $!";

$count=0;
$tot=0;
foreach $g(@myarray3)
 {
     $tot=990;
     print EV "\n";
     print EV "--";
     print EV "\n";

     $d=substr($g,0,10);
     print EV "$d\n";
     print "$d\n";

   foreach $g1(@myarray4)
   {
     @item_3 = split (/\s+/, $g1);
     $key1_1 = shift @item_3;
     $key2_1 = shift @item_3;
     $key3_1 = shift @item_3;

    if($key1_1 eq '//')
     {
      $sq=FALSE;
     }

    if($key1_1 eq 'AC')
    {
     $sq=TRUE;
     $tem_protein=$key2_1;
     }

    if($sq eq 'TRUE' && $key1_1 eq 'DR')
    {
     $sq1=TRUE;
    }

    if($sq1 eq 'TRUE' && $key3_1 eq $g)
     {
       $d1=substr($tem_protein,0,6);
       print EV "$d1\n";
       $count++;
     }
    }
   print EV 'count '."$count";
   $tot=$tot+$count;
   $count=0;
 }

##########################################################################
#                                   Mol.pl                               #
# Picks up GO term from Molecular function and maps proteins from dataset for #
# each GO term and also counts the proteins for each of them            #
##########################################################################

#!/usr/bin/perl
#use strict;

@myarray3= ('GO:0000166;', 'GO:0003674;', 'GO:0003676;', 'GO:0003677;',
'GO:0003682;', 'GO:0003700;', 'GO:0003723;', 'GO:0003774;', 'GO:0003779;',
'GO:0003824;', 'GO:0004518;', 'GO:0004672;', 'GO:0004721;', 'GO:0004871;',
'GO:0004872;', 'GO:0005102;', 'GO:0005198;', 'GO:0005215;', 'GO:0005216;',
'GO:0005326;', 'GO:0005488;', 'GO:0005509;', 'GO:0005515;', 'GO:0008092;',
'GO:0008135;', 'GO:0008233;', 'GO:0008289;', 'GO:0009055;', 'GO:0016209;',
'GO:0016301;', 'GO:0016740;', 'GO:0016787;', 'GO:0019825;', 'GO:0030188;',
'GO:0030234;', 'GO:0030246;', 'GO:0030528;', 'GO:0030533;', 'GO:0031386;',
'GO:0045182;', 'GO:0045735;');

open(INFO4,'Musculus_1.dat');
@myarray4=<INFO4>;

$fname = 'GOCHECK_MOL_DUPLICATE.txt';
open EV, ">>$fname" or die "cannot open file $fname; $!";

$count=0;
$tot=0;

foreach $g(@myarray3)
 {
     $tot=990;
     print EV "\n";
     print EV "--";
     print EV "\n";

     $d=substr($g,0,10);
     print EV "$d\n";
     print "$d\n";
```

```
  foreach $g1(@myarray4)
  {
     @item_3 = split (/\s+/, $g1);
     $key1_1 = shift @item_3;
     $key2_1 = shift @item_3;
    $key3_1 = shift @item_3;

    if($key1_1 eq '//')
    {
     $sq=FALSE;
     }

    if($key1_1 eq 'AC')
     {
       $sq=TRUE;
       $tem_protein=$key2_1;
     }

    if($sq eq 'TRUE' && $key1_1 eq 'DR')
     {
      $sq1=TRUE;
     }

    if($sq1 eq 'TRUE' && $key3_1 eq $g)
      {
       $d1=substr($tem_protein,0,6);
       print EV "$d1\n";
       $count++;
      }
    }
  print EV 'count '."$count";
  $tot=$tot+$count;
  $count=0;
 }


#############################################################################
#                                   Cell.pl                                 #
# Picks up GO term from Cellular component  and maps proteins from dataset for       #
# each GO term and also counts the proteins for each of them                  #
#############################################################################

#!/usr/bin/perl
#use strict;
```

```
@myarray3= ('GO:0000228;', 'GO:0000229;', 'GO:0005575;', 'GO:0005576;',
'GO:0005578;', 'GO:0005615;', 'GO:0005618;', 'GO:0005622;', 'GO:0005623;',
'GO:0005634;', 'GO:0005635;', 'GO:0005654;', 'GO:0005694;', 'GO:0005730;',
'GO:0005737;', 'GO:0005739;', 'GO:0005764;', 'GO:0005768;', 'GO:0005773;',
'GO:0005777;', 'GO:0005783;', 'GO:0005794;', 'GO:0005811;', 'GO:0005815;',
'GO:0005829;', 'GO:0005840;', 'GO:0005856;', 'GO:0005886;', 'GO:0005929;',
'GO:0005941;', 'GO:0009536;', 'GO:0009579;', 'GO:0016023;', 'GO:0030312;',
'GO:0030313;', 'GO:0043226;', 'GO:0043234;');

open(INFO4,'Musculus_1.dat');
@myarray4=<INFO4>;

$fname = 'GOCHECK_CELL_DUPLICATE.txt';
open EV, ">>$fname" or die "cannot open file $fname; $!";

$count=0;
$tot=0;

foreach $g(@myarray3)
 {
      $tot=990;
      print EV "\n";
      print EV "--";
      print EV "\n";

      $d=substr($g,0,10);
      print EV "$d\n";
      print "$d\n";

   foreach $g1(@myarray4)
   {
       @item_3 = split (/\s+/, $g1);
       $key1_1 = shift @item_3;
       $key2_1 = shift @item_3;
       $key3_1 = shift @item_3;

       if($key1_1 eq '//')
        {
         $sq=FALSE;
        }

      if($key1_1 eq 'AC')
       {
         $sq=TRUE;
         $tem_protein=$key2_1;
       }

    if($sq eq 'TRUE' && $key1_1 eq 'DR')
     {
      $sq1=TRUE;
     }

    if($sq1 eq 'TRUE' && $key3_1 eq $g)
     {
      $d1=substr($tem_protein,0,6);
      print EV "$d1\n";
      $count++;
      }
    }
   print EV 'count '."$count";
   $tot=$tot+$count;
   $count=0;
 }


#############################################################################
#                                   cal.pl                                  #
# For each GO term, corresponding to sequence pairs in it, calculates the posterior   #
# probability for each of them and puts them into sub-intervals                       #
#############################################################################

#!/usr/bin/perl
#use strict;

$pattern='GO:0043234';
$pattern1=substr($pattern,3);
open(INFO1,$pattern1.'_n.txt');


@myarray1=<INFO1>;

open(INFO2,'all_cell.txt');
@myarray3=<INFO2>;

$fname = $pattern1.'_post.txt';
open EV, ">>$fname" or die "cannot open file $fname; $!";

print 'Opening file - '.$pattern1.'_n.txt'."\n";
print "--------------------------\n\n";
```

```
@num=('1', '0.1', '0.01', '0.001', '0.0001', '0.00001', '0.0000000001',
'0.000000000000001', '0.00000000000000000001',
'0.00000000000000000000000000000000000000000000000001',
'0.0000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000001');


foreach $comp(@num)
{

$num = 0 ;
$deno = 0;
$res=999999;

foreach $g(@myarray3)
 {
   @item_3 = split ('\t', $g);
   $key1_1 = shift @item_3;
   $key2_1 = shift @item_3;
   $key3_1 = shift @item_3;

   if($key3_1 <= $comp)
   {
     $sb1=check($key1_1, $pattern1);
     $sb2=check($key2_1, $pattern1);

     if($sb1 == 1111 || $sb2 == 1111)
      {
        $deno++;
       }

      if($sb1 == 1111 && $sb2 == 1111)
       {
         $num++;
        }
     } #if condition
   } #myarray3

if($deno != 0)
 {
   $res=$num/$deno;
 }

print "num-$num, deno=$deno\n";
```

```
if($comp eq '1')
{
  $res=$res*100;
  print EV "$res ";
  print  "LOG p <= 0 ----- $res\n";
 }

if($comp eq '0.1')
{
  $res=$res*100;
  print EV "$res ";
  print "LOG p <= -1 ----- $res\n";
}

if($comp eq '0.01')
{
  $res=$res*100;
  print EV "$res ";
  print "LOG p <= -2 ----- $res\n";
}

if($comp eq '0.001')
{
  $res=$res*100;
  print EV "$res ";
  print  "LOG p <= -3 ----- $res\n";
 }

if($comp eq '0.0001')
{
  $res=$res*100;
  print EV "$res ";
  print "LOG p <= -4 ----- $res\n";
}

if($comp eq '0.00001')
{
  $res=$res*100;
  print EV "$res ";
  print "LOG p <= -5 ----- $res\n";
}

if($comp eq '0.0000000001')
{
  $res=$res*100;
  print EV "$res ";
  print "LOG p <= -10 ----- $res\n";
 }

 if($comp eq '0.000000000000001')
 {
  $res=$res*100;
  print EV "$res ";
  print "LOG p <= -15 ----- $res\n";
 }

 if($comp eq '0.00000000000000000001')
 {
 $res=$res*100;
 print EV "$res ";
 print "LOG p <= -20 ------ $res\n";
 }

 if($comp eq '0.00000000000000000000000000000000000000000000000001')
 {
 $res=$res*100;
 print EV "$res ";
 print "LOG p <= -50 ------ $res\n";
 }

 if($comp eq
'0.0000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000001')
 {
 $res=$res*100;
 print EV "$res ";
 print "LOG p <= -100 ------ $res\n";
 }
}

sub check
{
 $val=$_[0];
 $pattern1=$_[1];

 open(INFO4,$pattern1.'_n.txt');
 @mya = <INFO4>;

 foreach $a(@mya)
  {
   @az = split (/\s+/, $a);
   $s1 = shift @az;
   $s2 = shift @az;

   if($s1 eq 'protein:' && $s2 eq $val)
    {
     return 1111;
    } # end of if
 } #end of mya
}
close EV;
```