INTEGRATIVE APPROACHES TO PROTEIN HOMOLOGY SEARCH

by

Intikhab Alam

A thesis submitted in partial fulfilment of the requirements for the degree of

Doctor of the Natural Sciences (Dr. rer. nat.)

to the

International Graduate School in Bioinformatics and Genome Research, Center for Biotechnology (CeBiTec), University of Bielefeld, Bielefeld, Germany

Faculty of Technology, University of Bielefeld, Bielefeld, Germany

March 2005

Dedicated

to

Allah, the most merciful, the magnificent

and to

My Parents

Abstract

Many methods have been developed to search for homologous members of a protein family in data bases, and the reliability of results and conclusions may be compromised if only one method is used, neglecting the others. This thesis introduces an integrative approach to homology search and shows that an effective combination of homology search methods reveals superior results (Alam et al., 2004). Two protein sequence database search methods (called CHASE (Comparative Homology Agreement SEarch) and GenCHASE (Genomic CHASE)) were developed, which serve as a major step to improve the detection of remote homology. CHASE combines methods that search proteins in protein databases. We implemented some improvements in CHASE that we now call CHASE2. An evaluation based on the SCOP data base reveals that, on average, a coverage of 55% and 49% can be obtained by CHASE2 and CHASE respectively, in searches for distantly related homologues (i.e. members of the same superfamily, but not the same family – the most difficult task), accepting only 10 false positives, while the individual methods obtain a coverage of 31 to 44%. GenCHASE combines methods that search proteins in genomic sequences and predict gene structure. Using GenCHASE we have found several candidates for ABC, S100, and Cadherin proteins. Experimental verification of some of these candidates is underway. CHASE can be downloaded at http://www.mathematik.uni-bielefeld.de/~intikhab/chase-release1.0.tar.gz .

Table of Contents

1 Introduction

1.1. Overview and Problem Statement

Sequence database search, finding homologous sequences that are related to a given single sequence (or a set of sequences) by common evolutionary descent, is one of the most important tasks in computational biology. Any improvement in this field is of high relevance to phylogeny and function prediction because by discovering how sequences are related to known proteins we can make predictions of their structural, functional and evolutionary features (Lindahl and Elofsson, 2000). A number of homology search methods are available, as described in section 2.4. Different methods report different results and the growing number of database search methods poses a major problem to wet-lab users, as it is difficult to decide which method should be used and which not. Thus, the question arises whether it may be possible to combine methods, and how a combination may be accomplished.

In this thesis, I present two consensus protein database search methods, or, more precisely, combination schemes for homology search (called CHASE (Comparative Homology Agreement SEarch) and GenCHASE (Genomic CHASE)), which improve the coverage of remote homologues and give better performance than any of their component methods. CHASE combines methods that search proteins in protein databases while GenCHASE combines methods that search proteins in genomic sequences and predict gene structure. We combine only those protein database search methods in CHASE and GenCHASE that report confidence estimates.

Chapter 2 describes the background of homology search. The PHASE4 system (Rehmsmeier, 2002) that evaluates homology search methods is explained as well. In chapter 3, I describe the CHASE scheme that combines several protein

database search methods. In this chapter the evaluation of CHASE and its component methods, using PHASE4, is also explained. Chapter 4 explains several improvements on the basic version of CHASE (now called CHASE2) such as the implementation of a modular structure and re-calculation of E-values that significantly improved the performance of CHASE in detecting close and distant homologues. An evaluation of CHASE2, CHASE1, and all of the component methods, using the PHASE4 system, is also explained in detail.

Chapter 5 of the thesis explains the development of GenCHASE and its application on the human genome in finding hitherto unknown members of several protein families.

1.2. Publications, Posters, Application Notes, Case Studies

CHASE (chapter 3 of the thesis) was published in Proceedings of the National Academy of Sciences in 2004 (PNAS). Previously this work appeared as a technical report at the FSPM. This work was also presented as posters at ECCB 2002 and RECOMB 2003. Improvements on CHASE (chapter 4) will soon be submitted to an appropriate journal.

The development of GenCHASE and its application to the human genome to find members of several protein families (as case studies) was presented as posters at ISMB/ECCB 2004 and GCB 2004. Drafts for two separate publications on GenCHASE (one explaining its development and elucidation of ABC transporters in the human genome, the other explaining its application on the human genome in finding members of protein families such as S100s and Cadherins) will soon be submitted to appropriate journals.

1.3. Acknowledgments

Praise be to **Allah** (the unique God), the Cherisher and Sustainer of the Worlds *(Al Quran*, Chapter 1:2), who blessed me with the ability to undertake and finally complete this work.

I would like to thank my family for their immense support; they have been patient, encouraging and understanding.

My sincere gratitude goes to my supervisors, Dr. Georg Fuellen, Prof. Dr. Andreas Dress, and Prof. Dr. Ralf Hofestädt. This work would not have been possible without their constant encouragement, interest, and guidance. I am particularly thankful to Dr. Georg Fuellen for his intense supervision.

I pay special thanks to Professor Robert Giegerich, the dean of the faculty of technology, for his continuous support and for allowing me to complete my degree at the Faculty of Technology.

I would like to pay thanks to PD Dr. Klaus Prank, the executive director of the NRW International Graduate School in Bioinformatics and Genome Research, and his staff and the computing support at the Faculty of Technology and the Genetics department for their immense co-operation.

I am grateful to Marc Rehmsmeier for his work and advice on using Phase4 and reviewing the manuscript and thesis. I am particularly grateful to Dr. Stefan Lorkowski, Institute of Arteriosclerosis Research, the University of Münster, for scientific discussions and his experimental work on the verification of ABC transporters.

I am thankful to Dr. Claus Kerkhoff, Institute of Experimental Dermatology, Münster, and Dr. Eivind Hovig, Department of Tumor Biology, Institute for Cancer Research, The Norwegian Radium Hospital, Montebello, Oslo, Norway, for their work on S100 proteins.

I am thankful to Mohammad Shahid for developing the interfaces for CHASE.I would like to say my special thanks to all of my friends particularly Bjoern Oleson, Leila Taher, Mark Möller, ConnKhurram Ghayas, Arshad Mahmood, Nadir Pervez, and finally Jana for their scientific discussions, reading of my thesis, for their love and great encouragement throughout my work.

I am sure I have forgotten some names. I assure you that this is a shortcoming on my part and not on yours. I beg you to forgive me for my oversight.

2. Background --Concepts in Protein Homology Search

2.1. DNA and Proteins

All living things are based on information written in the universal language of DNA (DeoxyriboNucleic Acid). A nucleotide is a subunit of DNA consisting of a nitrogenous base (adenine, guanine, thymine, or cytosine), a phosphate, and a sugar molecule. DNA attains its double helical structure due to the hydrogen bonding between purine (adenine or guanine) and pyrimidine (thymine, or cytosine) bases, where adenine bonds with thymine, and guanine with cytosine (Watson and Crick, 1953). A gene is a sequence of bases, usually located at a specific position on a chromosome in a cell's nucleus. During transcription, nuclear genes render a messenger called “messenger RNA” that travels to the cytoplasm of the cell and gets translated into a specific protein by assembling a polypeptide chain of amino acids according to its code of nucleotides, where three nucleotides (also known as a triplet) encode one amino acid. There are 20 standard amino acids. Any protein can be represented as a sequence of amino acids, varying in length from around 50 to over 5000. Proteins are of scientific interest because they perform many of the tasks that a cell needs to carry out at the molecular level. And many diseases can be traced back to malfunctioning proteins.

Mutation, selection and recombination are the basis of evolution. Organisms pass on their genes from one generation to another; this requires the replication (identical duplication) of DNA, which is made possible by the unambiguous association of adenine with thymine, and guanine with cytosine. This replication of DNA can be erroneous, for example it may cause substitutions. Such mutations (that are fixed in the population) can be the change of one nucleotide to another. Deletion or insertion of nucleotide(s) is also possible. There could also be changes of more global nature such as meiotic recombination, transpositions or reversals of segments of DNA. Some of the changes remain

less significant as they do not change the function of a gene product while others result in the loss of its function (Rehmsmeier, 2002).

To understand the molecular machinery of the cell it is important to understand the meaning, or function, of each protein encoded in the genome. Let us mention three different means for inferring or predicting the function of a protein. It can be inferred directly through wet-lab experiments, or indirectly by elucidating and investigating the three-dimensional structure of the protein. Another way to predict the function of a protein is to find one or more homologous (see below) proteins through sequence similarity. If the function of such homologues is already known it provides hints to determine the function of the protein in question. One goal of sequence analysis is to make inferences about the structure and function of a protein based on its primary amino acid sequence.

2.2. Sequence Homology, Similarity, and Alignment

Sequence homology and similarity are basic concepts of sequence analysis. Proteins that share a common evolutionary ancestor are said to be homologous. A set of homologous proteins, all of which descended from a common ancestor, is called a protein (sub)family. Because the overall three-dimensional structure, or fold, of a protein remains fairly constant over evolutionary time, various members of a protein family typically share a common fold. This similarity of fold implies a similarity of function, with the degree of functional similarity depending upon the degree of evolutionary divergence that has occurred within the family. Unfortunately, the only way to conclusively prove that two sequences are homologous would be to trace their descent from a common ancestral sequence (Grundy 1998b). In practice, this common ancestor is not available. Nevertheless homology can be deduced from similarity with a restricted amount of certainty that, however, increases with the degree of similarity. Sequence similarity can be investigated, detected, and quantified. Protein homology inference is a core

problem and one goal in sequence analysis is the deduction of homology from similarity.

A)

```
X=0
C 12
S  0  2
T -2  1  3
P -3  1  0  6
A -2  1  1  1  2
G -3  1  0 -1  1  5
N -4  1  0 -1  0  0  2
D -5  0  0 -1  0  1  2  4
E -5  0  0 -1  0  0  1  3  4
Q -5 -1 -1  0  0 -1  1  2  2  4
H -3 -1 -1  0 -1 -2  2  1  1  3  6
R -4  0 -1  0 -2 -3  0 -1 -1  1  2  6
K -5  0  0 -1 -1 -2  1  0  0  1  0  3  5
M -5 -2 -1 -2 -1 -3 -2 -3 -2 -1 -2  0  0  6
I -2 -1  0 -2 -1 -3 -2 -2 -2 -2 -2 -2 -2  2 5
L -6 -3 -2 -3 -2 -4 -3 -4 -3 -2 -2 -3 -3  4 2  6
V -2 -1  0 -1  0 -1 -2 -2 -2 -2 -2 -2 -2  2 4  2  4
F -4 -3 -3 -5 -4 -5 -4 -6 -5 -5 -2 -4 -5  0 1  2 -1  9
W  0 -3 -3 -5 -3 -5 -2 -4 -4 -4  0 -4 -4 -2 -1 -1 -2  7 10
Y -8 -2 -5 -6 -6 -7 -4 -7 -7 -5 -3  2 -3 -4 -5 -2 -6  0  0 17
   C  S  T  P  A  G  N  D  E  Q  H  R  K  M  I  L  V  F  W  Y
```

B)

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

Figure 1. Standard scoring (or substitution) matrices:

A) PAM250, B) BLOSUM62. PAM (percent accepted mutation) and BLOSUM (blocks substitution matrix) are matrices that define scores for each of the possible amino acid substitutions. The PAM250 matrix is appropriate for searching for sequences that have strongly diverged (250 PAM means 250 mutations per 100 amino acids of sequence). The BLOSUM62 matrix is calculated from gap-free protein sequence alignments of sequences that are

more than 62% identical. BLOSUM62 is best for detecting the majority of weak protein similarities.

The concept of sequence similarity usually implies a similarity score - a statement of quantitatively how similar we judge two sequences to be. For comparative sequence analysis, we first need to compile the similarity scores that we give to different amino-acid pairs into a matrix; this is frequently called a scoring matrix. In other words, a scoring matrix is a two-dimensional matrix that contains all possible pair-wise amino acid scores. Scoring matrices are evolution in a nutshell. In the context of sequence comparison, scoring matrices are also called substitution matrices because the scores represent relative rates of evolutionary substitutions (Korf et al., 2003). Well-known substitution matrices are for example Dayhoff's Percent Accepted Mutations 250 or PAM250 (Dayhoff et al., 1978) or BLOcks SUbstitution Matrix 62 or BLOSUM62 (Henikoff and Henikoff, 1991), see Figure 1.

If we are investigating a new sequence, then our first task is to find out whether it shares similarities with other protein sequences that are known. If we render each amino acid of a protein sequence (the query sequence), position-by-position, comparable to at most one amino acid of another protein sequence (the target sequence), considering a scoring matrix, the procedure is called alignment (or more precisely an alignment trace). Comparing just two sequences is known as pairwise alignment, as shown in Figure 2, while comparing more than two sequences is called multiple sequence alignment, as shown in Figure 3. Nowadays various heuristic algorithms are available for multiple sequence alignment. Some of the well-known algorithms among many are ClustalW (Higgins, 1994), Dialign (Morgenstern, 1999) and Mafft (Katoh et al., 2002). An example of multiple sequence alignment, generated using ClustalW, is shown in Figure 3. When scanning a database of target sequences the query sequence is

heuristically compared, or aligned pairwise, with each sequence present in the database.

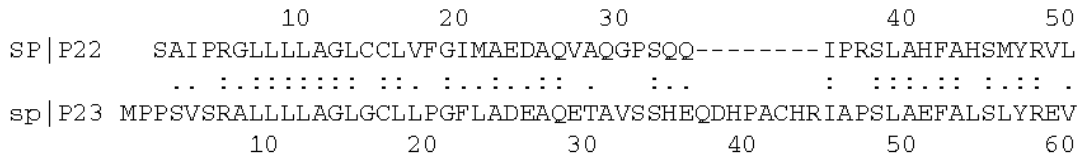

Figure 2. An example Pairwise alignment

A part of a pairwise alignment is shown. The middle row shows the conservation of amino acids for a particular column. The ":" symbol means an amino acid is identical in both sequences, "." means that a similar amino acid is aligned and the "-" means the gap character.

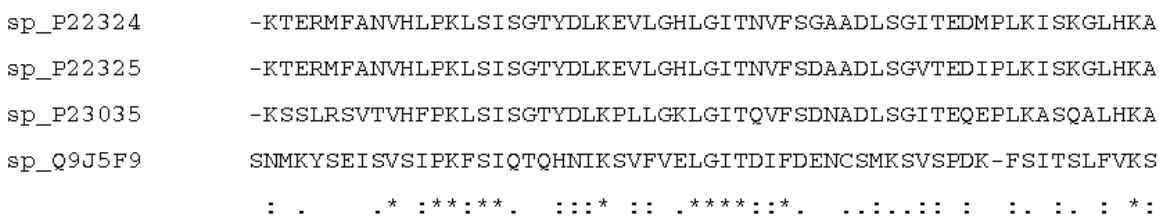

Figure 3. An example Multiple Alignment

A part of an alignment, generated using ClustalW, is shown. The last row shows the conservation of amino acids for a particular column. The "*" symbol means an amino acid is identical in all sequences, ":"means that the amino acids are strongly conserved, and "." means that the amino acids are weakly conserved. Note the gap towards the end of the forth sequence.

2.3. Sequence Databases

Given the growing number of sequenced genomes, along with the completion of the human genome project, the amount of accumulated DNA sequence data keeps growing at an accelerated rate. There are about 37,893,844,733 bases in

32,549,400 sequence records as of February 2004 at GenBank, an annotated collection of all publicly available DNA sequences (Benson et al., 2004). The SWISS-PROT protein knowledgebase (Boeckmann et al., 2003), on the other hand, connects amino acid sequences with the current knowledge in the life sciences. It is the leading universal curated protein sequence database. SWISS-PROT Release version 43.1 as of 13-Apr-2004 contains 148516 entries. TrEMBL (Translation from EMBL) is another database that consists of computer-annotated entries derived from the translation of all coding sequences in the EMBL/ GenBank nucleotide sequence database that are not yet included in Swiss-Prot. TrEMBL Release version 26.1 as of 13-Apr-2004 contains 1067463 entries. (For a review on protein sequence databases see Apweiler et. al., 2004).

More specialized are the protein signature databases that describe structures of specific amino acids typical for a certain protein family. These databases are based on several different methods that evolved with the need for efficient automatic methods of protein sequence classification and characterisation. Recently, the major signature databases such as PROSITE (Falquet et al., 2002), PRINTS (Attwood et al., 2002), Pfam (Bateman et al., 2002), and ProDom (Corpet et al., 2000) formed a Consortium and agreed to integrate their data into a new database that became known as InterPro (Apweiler et al., 2001). Subsequently SMART (Letunic et al., 2002) and TIGRFAMs (Haft et al., 2001) have joined the Consortium. The Structural Classification of Proteins (SCOP) database (Andreeva et al., 2004) is another specialized database built on comprehensive ordering of all proteins of known structure, according to their evolutionary and structural relationships. Protein domains in SCOP are hierarchically classified into families, superfamilies, folds, and classes. The accumulation of sequence and structural data allows more rigorous analysis and provides important information for understanding the protein world and its evolutionary repertoire.

2.4. Sequence Database Searching

In bioinformatics we apply information technology systems and strategies to store, organize and analyze the vast amount of biological data. This data is available on the one hand in the form of sequence databases of nucleic acids (the information carrier) and on the other hand it is available as sequence and structure databases of proteins (the building blocks of cells and organisms). Aside from maintaining the large databases, mining useful information from these is also very important. It is the search tools that integrate the user and the databases.

Every pairwise comparison yields a raw score of a heuristic alignment, where large scores usually indicate higher degree of similarity. The discovery of a statistically significant similarity (see section 2.5) between two proteins is frequently used, therefore, to justify inferring homology and a common functional role for the two proteins (Liao et al., 2002).

Sequence-homology search algorithms are important computational tools in molecular biology. Lots of efficient algorithms have been developed for sequence database searching. These algorithms are sometimes computationally intensive and need swift and parallel computing facilities for handling multiple queries simultaneously. There exist at least three general classes of techniques employed in searches for protein homologues, namely pairwise sequence comparisons such as Blast, profile-based searches such as HMMsearch, and motif- or pattern- based analyses such as PHI-Blast (see Altschul et al., 1990, Bork & Gibson 1996, Eddy 1998, Grundy 1998, Zhang et al., 1998). A detailed description of five search techniques, including PHI-Blast and HMMsearch, is given in section 2.4.4.

2.4.1. Pairwise Sequence Comparison

In a pairwise search, a query sequence is compared to a database sequence, yielding a confidence estimate (see section 2.5) that is supposed to indicate the chance of finding a comparably similar sequence in a database of random sequences, of the same size. The comparison is done for every sequence in the database, and the sequences with highest confidence ("the hits") are reported. The most popular pairwise-search tool so far is Blast (Basic Local Alignment Search Tool) (Altschul et al., 1990). FASTA (Pearson, 1990) is another example of a pairwise database search tool.

2.4.2. Profile-based Sequence Comparison

For most pairwise alignment programs, the twilight zone of very uncertain homology inference falls between 20-25% sequence identity (Chung and Subbaih, 1996). Additional information is needed in order to discover even more remote homologues to push back the twilight zone. Individual members might be missed by pairwise search analysis, in a diverse family of proteins, if they have very low pairwise similarity (Grundy 1998b). However, using a representative set of sequences from the family can uncover such missed relationships (Altschul, 1997).

Simple profile searches like PSI-Blast make use of position-specific scoring matrices based on a set of sequences and are usually more sensitive than pairwise comparisons. The introduction of Hidden Markov Models (HMMs) appears to provide a firmer statistical basis for profile search. The majority of currently available profile tools use HMMs, for example the *HMMER* package (Eddy, 1996).

2.4.3. Motif/Pattern-based Sequence Comparison

Kinship between protein sequences can also lead to (and, thus, be recognized by) the occurrence of particular amino-acid patterns (also known as motifs, signatures, or fingerprints) that were conserved throughout the evolution of the protein family in question and are believed to correlate with specific structural features and function. Motif analysis, therefore, can also be used for identifying new members of a protein family (Bairoch et al., 1997, Hudak et al., 1999, Jonassen et al., 1995). Motifs are the backbone of homology-search methods such as PHI-Blast (Zhang et al., 1998). In contrast to profiles, motifs are usually short, they include a short stretch of very specific amino acids deemed relevant for function, and they are denoted by specific regular expressions (Falquet et al., 2002) designed to represent a family-specific pattern.

2.4.4. Sequence-based Homology Search Methods: 5 Examples

The ultimate task of all homology search methods is the same, namely, to identify related sequences from a database, given a single or a set of sequences. However, they differ in the technique they use to accomplish this task. As described, three main techniques on the basis of which homology search methods can be classified are a) pairwise sequence comparisons, b) profile based sequence comparison, and c) Motif/pattern based sequence comparisons. Among many available homology search methods, we will combine only those five methods known to use a collection of sequences as search input and to report a confidence estimate, such as an E-value, for each hit. The first two methods perform profile-based searches by using a Hidden Markov Model (HMM). PSI-Blast and PHI-Blast are profile-based and pattern-based, respectively. MAST is also pattern-based.

- **HMMsearch** (Eddy, 1998) tries to align a Hidden Markov Model (HMM) with every database sequence and reports the matches. Methods such as hmmbuild (Eddy, 1998) can turn a multiple sequence alignment into the necessary HMM. Here, a carefully defined nonredundant set of sequences that belong to the protein family in question results in a better HMM and ultimately better detection of remote homologues. Because sequence families preferentially conserve certain critical residues and motifs, and this information can be incorporated accurately into an HMM, HMMs can often allow very sensitive database searches to be done (Eddy 1998b). HMMs generated using hmmbuild are calibrated using hmmcalibrate that automatically estimates some parameters needed for calculating accurate E-values by HMMsearch in database searches (Eddy 1998b). The run-time of HMMsearch is approximately proportional to the product of the lengths of the query sequence and the database searched (see Eddy, 1998).

- **Treesearch** (Rehmsmeier & Vingron, 2001) requires a multiple alignment of the query sequences and a phylogenetic tree based on this alignment, in addition to an HMM. Treesearch then compares the HMM with the database sequences one by one, as HMMsearch does, and temporarily inserts the database sequence into the given phylogenetic tree, adding a new edge to the existing tree. Homology between the given family of proteins and this sequence is then judged from the length of this edge (the tree augmentation). Treesearch is based on phylogenetic trees and it shows that phylogenetic information improves the detection of distant homologies. Treesearch runs, under certain assumptions, proportional to the product of the lengths of the query sequence, of the HMM, and of the database searched, see (Rehmsmeier & Vingron, 2001).

- **PSI-Blast** (Altschul et al., 1997) can be started using a profile, read off from the multiple sequence alignment of the input sequences. Usually, PSI-Blast is run iteratively, each new run being based on the output of the previous one, though we will use only one iteration step of this algorithm. The PSI-BLAST program runs at approximately the same speed per iteration as gapped BLAST, that is proportional to the product of the lengths of the query sequence and the database searched, but in many cases it is much more sensitive to weak but biologically relevant sequence similarities (Altschul et al., 1997). However, much care should be taken when selecting the set of sequences to be given to PSI-Blast before the iteration step, because one unrelated sequence may pollute the search profile and result in irrelevant hits with significant E-values (Eddy 1998b).

- For **PHI-Blast** (Zhang et al., 1998), that combines matching of regular expressions with local alignments surrounding the match, we need a motif in the form of a "regular expression" (Falquet et al., 2002) designed to represent a family-specific pattern. An example pattern for Serpins, extracted from the PROSITE (Falquet et al., 2002) database is shown below:

  (LIVMFY)-x-(LIVMFYAC)-(DNQ)-(RKHQS)-(PST)-F-(LIVMFY)-(LIVMFYC)-x-(LIVMFAH)

  It says that any one of ‘LIVMFY’ amino acids is allowed at position 1. The dash “-“ character coming next means what follows is the specification of the amino acid at the next position. The “x” character that follows means any amino acid is allowed at this position, etc.

  Such an expression can be derived from the input sequences using PRATT (Jonassen, 1997). PHI-BLAST then searches a protein database for other instances of the input pattern, and uses those found as seeds for the construction of local alignments to the query sequence. The

distribution of PHI-BLAST alignment scores has been studied analytically and empirically. In many instances, the program is able to detect statistically significant similarity between homologous proteins that are not recognizably related using traditional single-pass database search methods (Zhang et al., 1998). Finally, following a suggestion from the Blast software README for improving the competitiveness of PHI-Blast, we will apply PSI-Blast just once, using a profile derived from the PHI-Blast result. No information could be found on the run-time of PHI-Blast, except for empirical data listing run-time in seconds (Zhang et al., 1998).

- Finally, **Mast** (Bailey & Gribskov, 1998) searches biological sequence databases for sequences that contain one or more of a group of known motifs. Motifs are provided in a specific format, derived using a tool called MEME (Multiple EM for Motif Elicitation), available with the MEME-Mast package. Mast compares the MEME-derived group of motifs with each of the sequences in the database and reports the matches. Mast considers each piece of evidence (for a motif match) in the form of a p-value, and then uses the product of these p-values as the measure of membership in the family. For calibration, it uses an algorithm (QFAST) for calculating the statistical distribution of the product of n independent p-values. Mast demonstrates that sorting sequences by this p-value effectively combines the information present in multiple motifs, leading to highly accurate and sensitive sequence homology searches (Bailey & Gribskov, 1998). No information could be found on the run-time of MAST.

2.5. Significance Of Database Search Results

Sequence database search, given a single sequence or a set of sequences as query to find out similar sequences, is a widely used tool in bioinformatics. Sequence homology search tools report a list of matched sequences or hits that are aligned with the query either locally (in case of pairwise methods) or semi-globally (in case of profile based methods). Each hit is assigned a numerical

score and only the significant hits are reported that give a score greater than some threshold (Pagni et al., 2001). The value of such a numerical score is usually based on an alignment score.

2.5.1. Alignment Score or Raw Score

Sequence homology search algorithms usually are heuristics for finding the highest-scoring alignment of segments from the two sequences being compared. A typical alignment score (also called raw score) is determined either by using a substitution matrix such as BLOSUM50 or PAM250 --, and if gaps are allowed, the gap opening or gap extension penalties, -- or by a position specific scoring matrix. More precisely, an alignment score is calculated by summing up the substitution score, defined for each aligned pair of letters, and gap scores for each run of letters in one segment aligned with gap characters inserted into the other (Altschul et al., 2001). A specific type of normalized alignment score is called the bit score.

2.5.2. Expect (E)-value

The result of a database search can be classified into true and false positive hits. The hits related to the query sequence(s) are called true positives and others (where the observed similarity is attributable to chance) are the false positives (ones that have not been found are the negatives). It is important to know that only biological arguments can let one distinguish if a sequence can be regarded as a true or false positive, though in terms of computational sequence analysis, a statistical analysis based on sound principles can also help in distinguishing the true from the false positives (Pagni et al., 2001). The E-value is the most frequently used such statistical estimate to represent the significance of database search results. In the simplest pairwise case of standard Blast searches, given a normalized pairwise-comparison score *S*, the E-value estimates the expected number of distinct local matches with normalized score at least *S* in an equally

large database of random sequences (see (Altschul et al., 1997)). This concept can be generalized to other search methods, with different degrees of mathematical rigor.

2.6. Evaluating Database Search Methods

There exist many methods that search databases to find out homologous members of a protein family. An evaluation of such methods is necessary in order to figure out their performance in comparison to each other. Usually the evaluation of homology search methods is done in four steps (Rehmsmeier 2002b). As a first step, a sequence database of known relationships is divided into the training and test set of sequences where each test set is associated with a training set of homologous sequences. Database search methods are executed, given the training set of sequences, as a second step. *The more test sequences (i.e. homologs of the training sequences) are found, the better a search method performs.* In the third step in evaluating the methods, their result reports are read to split the scores into ones for homologues and ones for non-homologues. *A particular search method is designated better if it assigns better scores/E-values to homologs than to non-homologs.* In the final step, further statistical analysis based on the scores for homologues and non-homologues can be shown in tables or figures in a variety of ways. One such system that evaluates the performance of homology search methods is Phase4 (Rehmsmeier 2002b), where all above-mentioned steps are automated.

2.6.1. Phase4

Phase4 is a system for the automatic evaluation of database search methods. It offers the logical structure of the framework in which evaluations are usually accomplished. Automatically, a benchmark (e.g. SCOP (Murzin et al., 1995)) database is split into test and training sets, methods are executed, their performances evaluated and presented in selected tables and diagrams

(Rehmsmeier 2002b). The performance of a method is evaluated by its ability to find a test set of sequences in a target database, using a training set of sequences for “learning” e.g. for calculating an HMM. To construct test and training sets, Phase4 relies on target databases like SCOP that classify proteins according to membership in *families* (of closely related sequences) and in *superfamilies* (of not so closely related sequences). It should be noted that Phase4 does not claim to offer its own or any new way to evaluate the homology search methods but it is an interface that offers the usual evaluations in an automatic way. Phase4 has already been used to evaluate the "Jumping Alignment" method (Spang et al., 2002).

2.6.2. Phase4 Evaluation Scenarios

In Phase4, an *evaluation scenario* is defined by specifying a training and a test set of sequences in the target database. For example, the scenario “*Distant Family One Model”* is used to evaluate a homology search method for its ability to report distant relationships in protein superfamilies by splitting off one family from a given superfamily to provide the test sequences, and keeping the rest of the superfamily as training sequences. Such a test is executed, for each family in turn, for every superfamily (see Table 1 and Figure 4 for commonly used scenarios, and (Rehmsmeier 2002b) for more details). As noted, *the more test sequences (i.e. homologs of the training sequences) are found, the better a search method performs.*

Table 1. Evaluation scenarios defined by Phase4, given a sequence database that is organized into families and superfamilies*.

Scenario	Description
"Distant relationship" (Distant Family One Model (DFOM))	From a superfamily, each family in turn is chosen to provide the test sequences. The remaining families within that superfamily provide the training sequences.
"Close relationship" (Family Halves One Model (FHvOM))	For each superfamily, half of the sequences of each of its families are chosen as training, the remaining ones as test sequences.
"Very close relationship" (Family Half One Model (FHfOM))	For each superfamily: For each family, half of its sequences are chosen as test, the remaining ones as training sequences. The sequences of the surrounding superfamily are ignored in the evaluation.

*Note that training sequences are always ignored in the evaluation, and that the division into test and training sequences as described above is performed for each superfamily in turn. For the last model, average performance is calculated over an additional inner loop that considers each family in turn.

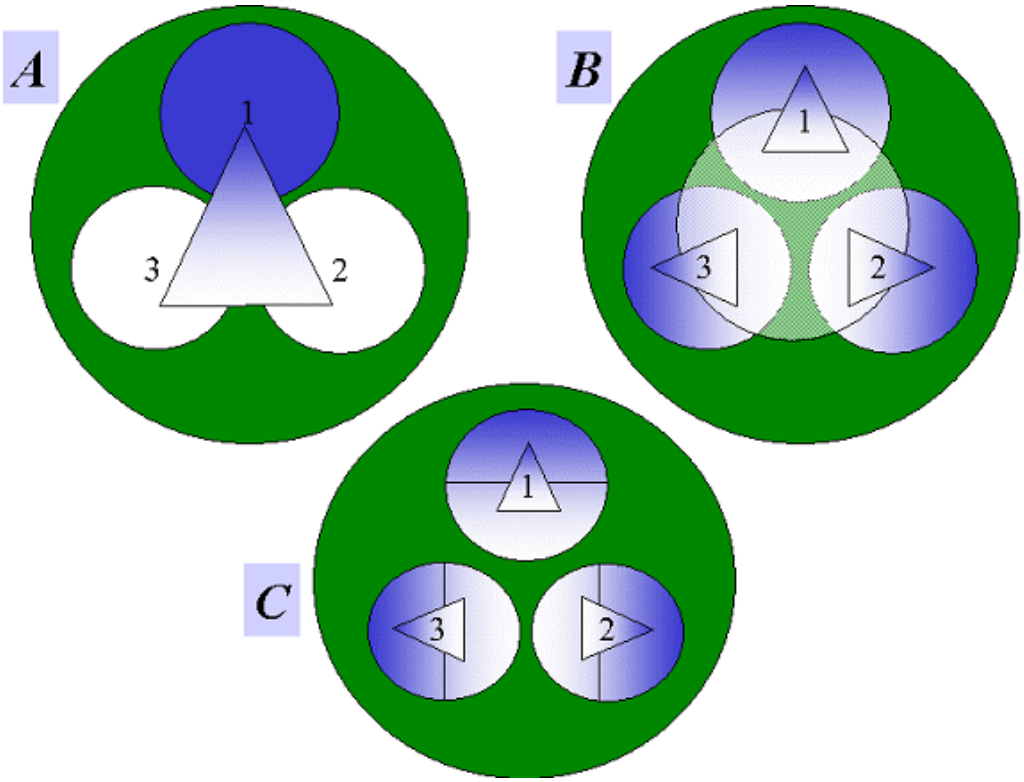

Figure 4. Three Evaluation Scenarios are visualized: A) Distant Relationships, B) Close Relationships, and C) Very Close Relationships. Big green circles represent a SCOP superfamily, blue (parts of) circles represent the test set of sequences to be found by a method using the training set of sequences which are represented as white (parts of) circles (See table1 for definitions of evaluation scenarios). Numbering of circles shows that, for example in case of A) "1" is the first test protein family while "2" and "3" are used for training. In the second round "2" will be the test protein family while the others are used for training, and so on.

2.6.3. Phase4 Performance Plots

To evaluate the performance of any method numerically, Phase4 offers *evaluators*. These make use of the lists of sequences found that are ranked according to a confidence estimate, e.g. an *E-value* (as shown in Table 2), or according to a score. (E-values are reported by each of the search methods we want to combine, and our combination scheme will report a combined E-value.)

We now need to formalize our statement "*The more test sequences (i.e. homologs of the training sequences) are found, the better a search method performs".* We already noted the following variant: "*A particular search method is designated better if it assigns better scores/E-values to homologs than to non-homologs".* Accordingly, the "coverage versus false positive counts" evaluator, provided by Phase4, does the desired formalization. For a given test, this evaluator compares the "good" and the "bad" guys as follows: It calculates the percentage $P(k)$ of true positives (relative to the set of *all* true positives in the database) with an E-value smaller than or equal to that threshold value for which exactly k false positives are found, thus rendering the *percentage* coverage P as a function $P=P(k)$ of the ***absolute*** number k of misclassifications considered acceptable. Finally, results are averaged over all tests executed, They are then presented, for example, as a coverage versus false positive counts plot shown in Figure 5.

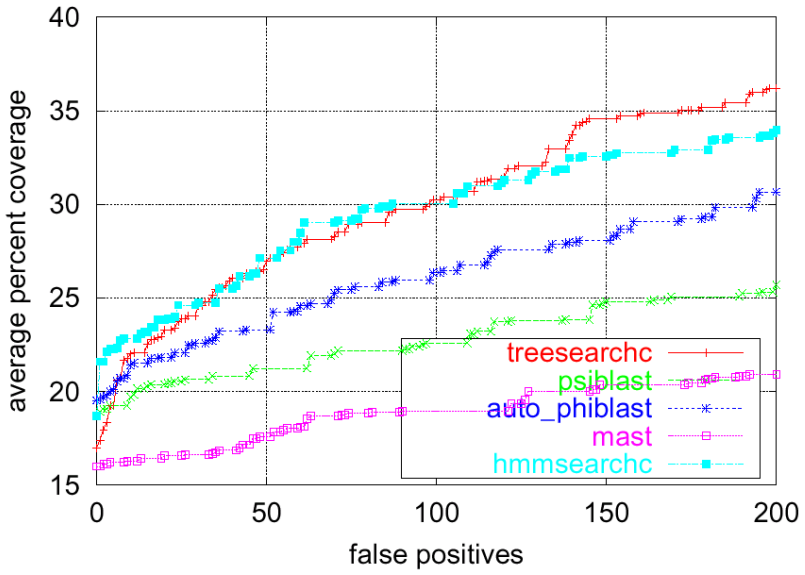

Figure 5. Coverage vs. False Positives Count Plot, an Evaluation Plot For Sequence Database Search Methods Produced Using PHASE4 System.

A plot showing average percent coverages of true positives, accepting 0 to 200 false positives, obtained by Treesearch, PSIBlast, PHI-Blast, Mast, and HMMsearch. Generally speaking, the faster a curve goes up, the better. Thus, HMMsearch and Treesearch perform best. For example, considering 50 false

positives acceptable, HMMsearch and Treesearch achieve true positives average coverage of 27%, PHI-Blast achieves 24%, PSI-Blast achieves 21%, and Mast 17%. This evaluation was performed under the distant relationship scenario (for more detail on evaluation scenarios see Table 1) for all protein superfamilies in one half of the SCOP database.

Table 2 A list of hits from different methods, sorted on the basis of HMMsearch E-values. The search was started with a set of sequences from superfamily 3.3.1 (SCOP version 1.53), featuring the FAD/NAD(P)-binding domain.

No	2	Description	HMMsearch	Treesearch	PSIBlast	PHIBlast	Mast
1	3.3.1.2.2	Cholesterol oxidase	1E-113	2E-22	7E-65	1E-154	2E-86
2	3.3.1.2.7	Glucose oxidase	2E-137	2E-22	2E-82	1E-19	2E-95
3	3.3.1.2.8	Glucose oxidase	2E-136	2E-22	3E-84	6E-12	5E-94
4	3.3.1.2.1	Cholesterol oxidase	5E-106	2E-22	3E-65	1E-146	2E-91
5	3.3.1.2.9	Polyamine oxidase	2E-100	1E-19	3E-81	3	0.00009
6	3.3.1.4.2	Fumarate reductase	5E-98	1E-22	2E-56	5E-67	1E-61
7	3.3.1.4.3	Fumarate reductase	1E-97	1E-22	2E-61	4E-16	4E-57
8	3.3.1.4.4	Flavocytochrome	5E-95	2E-22	7E-56	0.000000003	1E-64
9	3.3.1.4.5	Flavocytochrome	2E-94	2E-22	1E-53	0.0001	5E-62
10	3.3.1.4.1	L-aspartate oxidase	2E-93	2E-22	8E-57	2E-61	5E-53
11	3.3.1.3.1	Guanine nucleoase	7E-88	5E-21	2E-71	4	0.12
12	3.3.1.2.3	p-Hydroxybenzonate	1E-69	2E-20	5E-46	0.2	0.0007
13	3.3.1.2.5	Sarcosine oxidase	1E-66	3E-21	3E-46	0.008	0.0002
14	3.3.1.2.6	Phenol hydroxylase	5E-45	4E-19	1E-33	0.0005	0.00000003
15	3.3.1.1.3	Adrenodoxin reductas	8E-23	7E-20	1E-18	1000	3.5
16	3.3.1.1.2	Trimethylamine dehyd	0.003	2E-16	0.0000009	10	1000
17	3.32.1.13.8	HslU {Bacteria	4	18	1000	1000	280
18	3.3.1.5.8	Dihydrolipoamide	5.9	22	0.3	0.8	0.01
19	3.3.1.5.8	Dihydrolipoamide	19	15	0.4	0.9	0.0003
20	3.3.1.5.10	Dihydrolipoamide	51	34	1000	1000	0.01
21	5.8.1.3.1	T7 RNA polymerase	77	23	50	1000	90
22	1.111.5.1.1	70 KDa soluble lytic	90	1000	6	300	1000
23	3.3.1.2.6	Phenol hydroxylase	110	2E-16	0.002	700	96
24	3.3.1.5.1	Glutathione reductase	110	27	8	1000	0.0002
25	3.3.1.5.9	Dihydrolipoamide	130	22	0	3	0.12
26	1.119.1.1.2	Fumarase Escherichia	200	700	50	20	72
27	3.3.1.5.3	Trypanothione reductase	250	91	30	700	0.56
28	3.68.1.1.1	Ribokinase {Escherichia	330	88	1000	9	10
29	3.84.1.1.1	Asparaginase type II	610	1000	0.002	100	1000
30	3.62.1.3.3	Cystathionine gamma-	650	270	3	400	1000

31	3.3.1.5.4 Trypanothione reductase	710	120	20	4	0.005
32	3.3.1.5.8 Dihydrolipoamide	740	45	1000	100	0.005
33	3.3.1.5.5 Thioredoxin reductase	920	97	7	300	7.2
34	3.2.1.5.15 Lactate dehydrogenase	1000	1000	4	30	0.09
35	3.4.1.2.1 D-amino acid	1000	290	20	6	0.41
36	3.3.1.5.7 NADH peroxidase	1000	1000	3	60	0.52
37	3.4.1.1.2 Trimethylamide	1000	440	1	1000	1.2
38	3.3.1.5.2 Glutathione reductase	1000	480	40	9	8.2
39	3.3.1.5.6 Thioredoxin reductase	1000	1000	200	1000	0.03
40	3.2.1.2.1 Uridine diphosphogala	1000	5	200	30	1000

3. CHASE (Comparative Homology-Agreement SEarch)

3.1.1. Related Work on Methods Combination

The combination of methods is an advanced form of a meta-study. Important medical questions are typically studied more than once, and a meta-study compiles and analyses the results of all relevant studies. InterPro (Apweiler et al., 2001) and Metafam (Silverstein et al., 2001) present such compilations in protein-family research. Combining methods directly to generate a consensus result is also practiced in some areas of bioinformatics. Unfortunately, combining methods on a large scale is complicated by the fact that different programs often have different input requirements and output formats. Nevertheless, algorithms that efficiently combine different methods and standardise their input and output requirements can improve the accuracy of results considerably. Examples of such algorithms in the area of structure prediction, fold recognition, phylogenetic tree reconstruction and gene prediction are Jpred (Cuff et al., 1998), Pcons (Lundström et al., 2001), Hybrid (Huson et al., 2000) and Combiner (Allen et al., 2004), respectively. Jpred is a simple majority-wins based consensus predictor for secondary structure. Pcons is a neural-network-based consensus predictor for fold recognition. Hybrid is a method for combining outputs of different tree reconstruction methods (thus producing a ``hybrid'' method), and the authors have shown experimentally how one such hybrid method has better performance than its constituent parts (Huson et al., 2000). Combiner uses the output from gene finders, splice site prediction programs and sequence alignments to predict gene models. In the following section, Jpred and Combiner are explained in more detail.

3.1.1.1. Jpred

Jpred is a system that provides a consensus secondary structure prediction. It accepts two input types, a family of aligned protein sequences or a single protein

sequence. If a single sequence is submitted, an automatic process creates a multiple sequence alignment, prior to prediction (Cuff & Barton, 1998). To automatically generate the multiple sequence alignment, if a single query sequence is given, the OWL (Bleasby et al., 1994) database is searched with BLAST (Atschul et al., 1990). This returns a number of sequences that are then filtered using a Smith Waterman dynamic programming implementation, SCANPS (Barton, 1993), and then aligned.

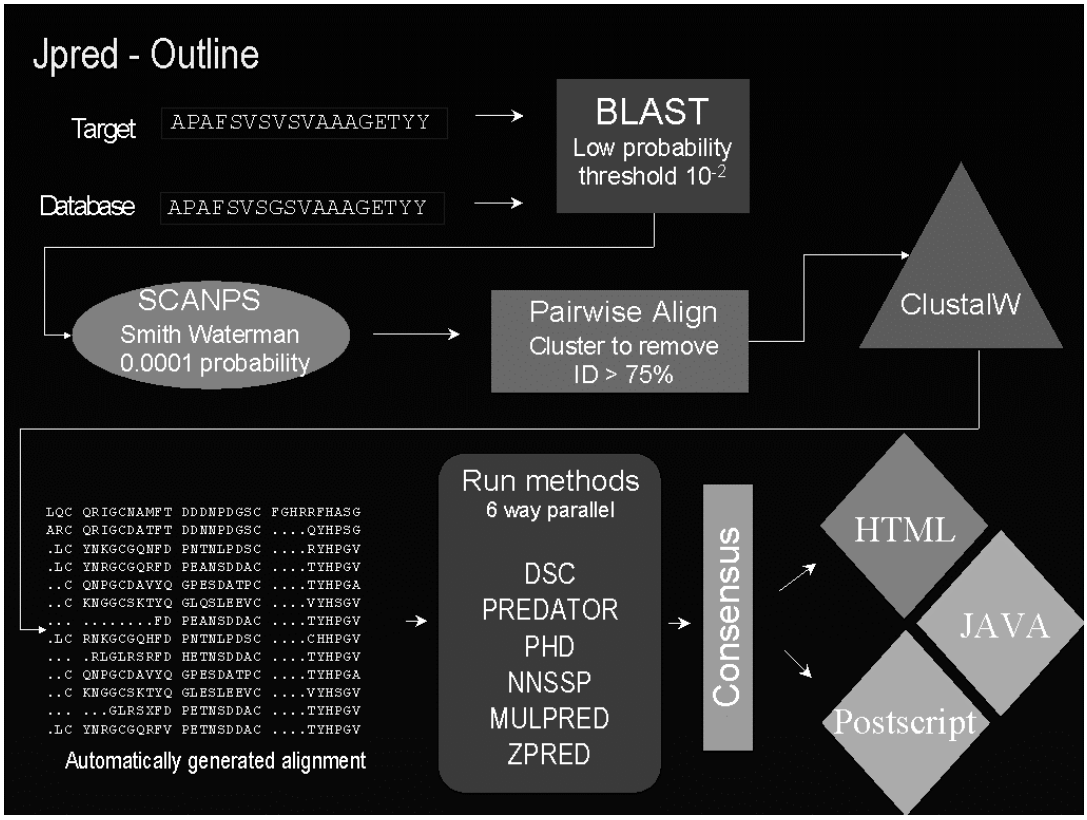

Figure 6. An outline of Jpred

Given a single sequence, similar sequences are extracted from a database using Blast, filtered using SCANPS, aligned using ClustalW, and provided as an input to six different structure prediction methods. At the end the results from each method are combined into a simple file format (java, html or postscript) along with a simple majority wins based consensus prediction (Clamp and Cuff, 1999).

Six different prediction methods (DSC (King & Sternberg, 1996), PHD (Rost & Sander, 1993), NNSSP (Salamov & Solovyev, 1995), PREDATOR (Frishman & Argos, 1997), ZPRED (Zvelebil et al., 1987), and MULPRED (Barton, 1988, unpublished) are then run, and the results from each method are combined into a simple file format along with a simple majority wins based consensus prediction. An outline of Jpred is shown in Figure 6.

3.1.1.2. Combiner

Combiner is a computational method to construct gene models by using evidence generated from a diverse set of sources. It takes as input a genomic sequence and the locations of gene predictions from ab initio gene finders, protein sequence alignments, expressed sequence tag and cDNA alignments, splice site predictions, and other evidence. Three different algorithms for combining evidence in the Combiner were implemented namely the simple Linear Combiner (LC1), the second combiner (LC2), and the Statistical Combiner (SC). LC1 uses a voting function to combine multiple gene prediction programs. Gene models predicted by any of the gene prediction programs are considered. For each position, each gene finder must vote for either coding or non-coding, and the highest-scoring combination of intervals of consecutive coding (or non-coding) positions of the gene is pieced together to form a gene model. Each gene finder is given equal weight, that is, one vote, in LC1.

The second Combiner (LC2) uses a similar algorithm to LC1, but with two significant enhancements. First, it adds sequence alignments (both DNA and protein) and splice site prediction program output to the inputs. Second, it uses different weights for the different forms of evidence. Finally, the Statistical Combiner (SC) uses decision trees to correlate evidence patterns with candidate gene models. SC also uses the confidence scores returned by the gene finders themselves (when available) to combine outputs from gene finders. In other words, instead of a simple linear function combining all the inputs, SC builds a

non-linear model based on a decision tree including confidence scores. (See Allen et al., 2004 for more details on Combiner and Mathe et al., 2002 for a detailed review on gene prediction methods).

3.1.2. Combining Sequence-based Homology-Search Methods

In the version of CHASE presented, we are dealing with the following five homology search methods, described in section 2.4.4: HMMsearch (Eddy, 1996), Treesearch (Rehmsmeier & Vingron, 2001), PSI-Blast (Position-Specific Iterated Blast) (Altschul et al., 1997), PHI-Blast (Pattern-Hit Initiated Blast) (Zhang et al., 1998), and Mast (Motif Alignment and Search Tool) (Bailey & Gribskov, 1998).

These methods display a significant difference in their performance (see Figures 8, 9, and 10 below). In this study, we will show that the overall performance of homology searches can be improved if these methods are combined appropriately. To date, to the best of our knowledge, there is no method available that produces a consensus over sequence-based homology-search methods.

We developed a system called CHASE (**Comparative Homology-Agreement SEarch**) that combines the five different methods (from now on called CHASE component methods; described in section 2.4.4 above) as follows: First, given a collection of query sequences, method-specific input queries structured according to the specific requirements of the individual search algorithms are automatically derived for each of the five component algorithms. Then, after these have been applied using their respective input queries, we compute and report a “consensus hit list”. An outline of CHASE is shown in Figure 7.

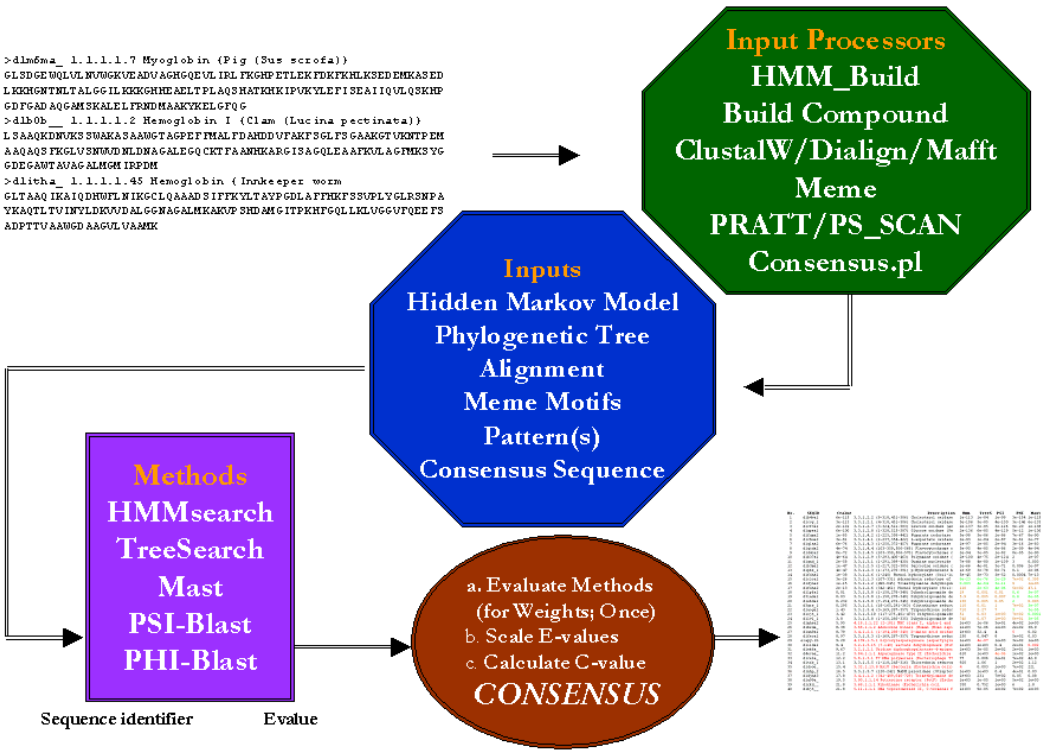

Figure 7. An outline of CHASE

CHASE uses input processors that transform a set of sequences into inputs for various homology-search methods, namely HMMsearch, Treesearch, Mast, PSI-Blast, and PHI-Blast. CHASE executes the underlying homology-search methods, the results of which are combined by the CHASE scheme to get a consensus.

We present a comparative evaluation of the performance of CHASE in section 3.4. It is needless to say that the evaluation of CHASE is of course performed on a database that is disjoint from the database used to calibrate this tool.

3.1.2.1 Input Processing

All of the 5 homology-search methods that we combine provide confidence estimates for their results. To perform their task, they require a query and a target database such as SWISSPROT or SCOP. The exact query format requirements, however, vary from method to method. We developed scripts called *input processors (IPs*) that take a collection of sequences and process these as follows to obtain the specific type of input for each of these homology-search methods.

- **HMMsearch IP:** We use *ClustalW* (Higgins et al., 1994) to generate a multiple alignment that in turn is used by *HMMbuild*, available with the *HMMER* package, to build a Hidden Markov Model. We calibrate the required HMM using *hmmcalibrate,* also available as part of HMMER.
- **Treesearch IP:** We use *build_compound*, available with Treesearch, to generate, as required, a sequence alignment (using ClustalW), a phylogenetic tree (using *fitch* (Felsenstein, 1998)), and an HMM (using *HMMbuild*).
- **PSI-Blast IP:** We use ClustalW to align the input sequences, and format the alignment such that it can be used to “jumpstart” a "single run" PSI-Blast search. A multiple alignment that is used to jumpstart PSI-Blast must include the query sequence as one of the sequences, but it need not be the first sequence. PSI-Blast further requires that the jumpstart alignment does not contain some of the headers and trailers that are usually present in ClustalW-based alignments.
- **PHI-Blast IP:** We use *PRATT* to generate a Prosite-like pattern from given un-aligned sequences, and a ClustalW alignment to generate a consensus sequence (by relative majority rule) for starting a PHI-Blast search, followed by a “single run” of PSI-Blast.
- **Mast IP:** We use *MEME* (Multiple EM for Motif Elicitation) (Bailey, 1994), given un-aligned sequences, to generate motifs that are used to run Mast.

3.2. A Scheme for Combining Homology Search Methods

We describe a scheme to combine several homology search methods based on the numbers such as E-values that they report for every hit. As shown in Figure 7, in our scheme for combining different homology-search methods we run them one after the other. Since they use various kinds of input information we provide this information automatically, employing input processors as described above. Once the searches are complete, the results of each method are parsed to extract specific information such as the unique sequence identifiers of the hits and the corresponding E-values. Tallying data for all methods, we obtain a preliminary list of all hits, each row containing one sequence identifier and the corresponding E-values reported by the different methods. Such a list was already presented in Table 2 (section 2.6.3).

Our basic idea of combining methods works in three steps. (a) Evaluating the performance of CHASE component methods, (b) placing methods on a common scale based on E-values, (c) calculating the combined E-value (called C-value).

3.2.1 Evaluating the Performance of CHASE Component Methods

We used the Phase4 system to evaluate, only once, the individual homology-search methods (to be combined in CHASE) to derive a *weighting scheme* that is based on their performance. Before starting the evaluation the SCOP database is split into two databases: the odd database, containing every second SCOP superfamily starting with the first one, and the even database, containing the rest. Among several available scenarios offered by Phase4 that define training and test sequences using the SCOP database as described before, we choose one for detecting distant, one for detecting close, and one for detecting very close relationship (see Table 1 on page 25 for details). To get a long list of hits an E-value cut-off as large as E_C = 1,000 was set for all individual homology-search

methods; sequences with a larger E-value than the cut-off are not listed. Before the implementation of modular structure of CHASE (described in chapter 4), we used only ClustalW-based alignments for alignment-based methods such as HMMsearch, Treesearch, PSI-Blast, and PHI-Blast.

As a result of the evaluation of CHASE component methods, we got three performance plots i.e. coverage versus false positive count plots, one for each of the evaluation scenarios mentioned above.

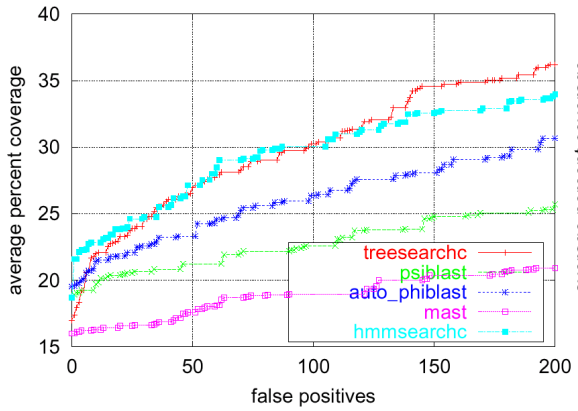

Figure 8

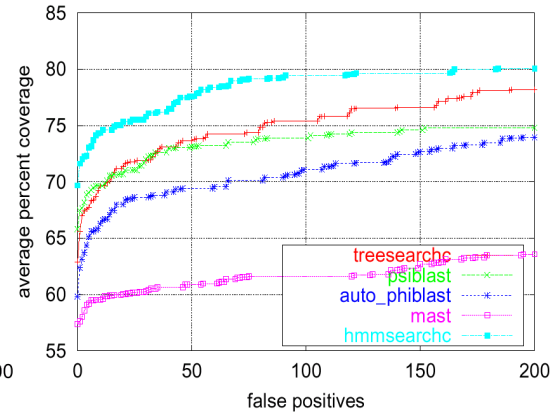

Figure 9

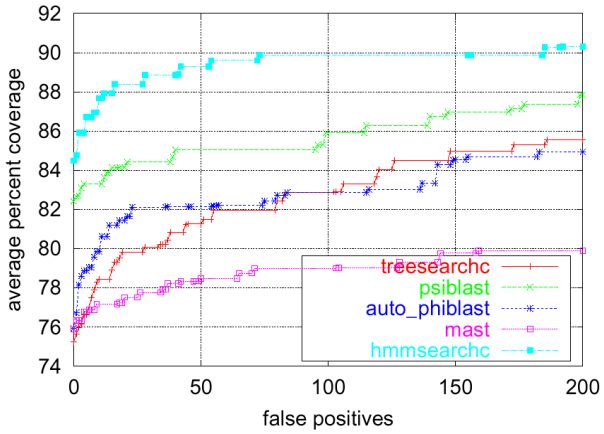

Figure 10

Figures 8, 9, 10: Coverage versus false positive counts for CHASE component methods as in Distant (DFOM), Close (FHvOM), and Very Close Relationships (FHfOM) scenarios, respectively, in the odd half of SCOP.

For each CHASE component method $i=1, \ldots, 5$ (i.e. HMMsearch, Treesearch, PSI-Blast, PHI-Blast, and MAST), the *average* percentages $P_i(k)$ of true positive coverages (from the protein super-families present in a particular evaluation scenario, i.e., distant, close, and very close relationship scenarios, respectively) are plotted on the Y-axis while accepted misclassifications (false positives) *k from 0 to 200* are plotted on the X-axis in Figure 8, 9, and 10. See section 2.6.3 for detailed explanations of these plots. Further, to measure the performance of component method i, we plot its average percent coverage for $k=50$ false positives, for each evaluation scenario mentioned above, in a histogram as shown in Figure 11.

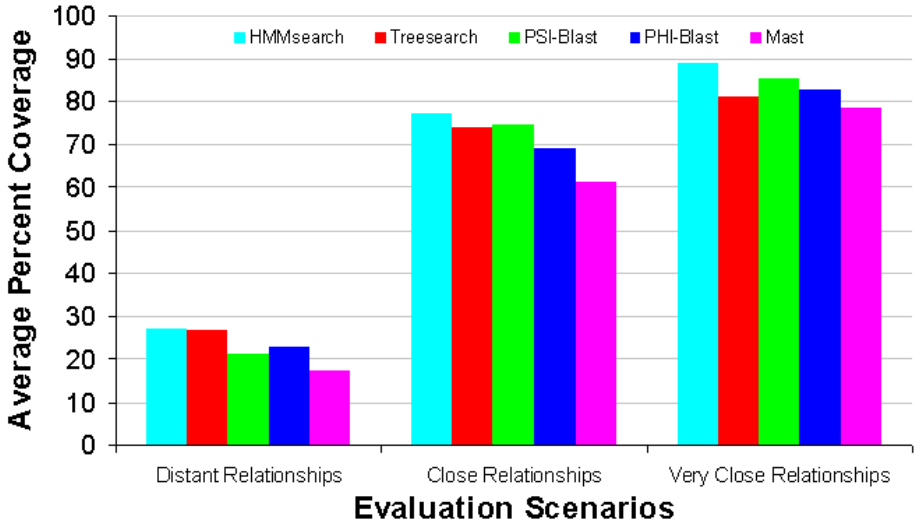

Figure 11. Average percent coverages permitting k=50 false positives. Data are based on the odd half of SCOP.

3.2.1.1. Weighting Scheme Based on the Performance of Methods

Based on the odd half of SCOP, we considered average percent coverages for $k=50$ false positives (see Figure 11) to calculate method weights $W=W_1, \ldots, W_n$, where n is the number of methods, and the weight W_i of method i is set to $P_i/(P_1 +$

... P_n), the average coverage P_i divided by the total sum of the average coverages of all n methods so that $\sum_{i=1}^{n} W_i = 1$ holds, as shown in Table 3.

Table 3. Weights for CHASE component methods (see Table 1, section 2.6.2, for evaluation scenarios)

Evaluation Scenarios	**HMMsearch**	**Treesearch**	**PSI-Blast**	**PHI-Blast**	**Mast**	Sum of Total Coverages
Distant Relationships	27.1429	26.9762	21.2381	23.3095	17.5714	
Close Relationships	77.5778	73.6667	73.1333	69.4222	60.9111	
Very Close Relationships	89.2791	81.2558	85.0465	82.1628	78.4884	
Total Coverages	193.9997	181.8987	179.4179	174.8945	156.9709	887.1818
W_i=Total Coverages / Sum of Total Coverages	0.2187	0.2050	0.2022	0.1971	0.1769	$\sum_{i=1}^{n} W_i = 1$

3.2.1.2. Limits on The Performance Of Homology Search Methods

We will briefly describe some possible explanations for the performance differences that give rise to the different method weights. We choose methods, to be combined in CHASE, that employ different techniques to search databases. HMMsearch reports matches to the Hidden Markov Model used. Treesearch searches the database using an HMM and in addition it utilizes phylogenetic information. PSI-Blast searches a database using a profile, generated from a multiple alignment. PHI-Blast uses a pattern or regular expression and a query

sequence that has this motif. Mast uses multiple motifs, generated using MEME, to search the database.

Naturally, these different approaches have their own specific advantages/ disadvantages or limits influencing the coverage of possible homologs reported by these methods. HMMsearch and Treesearch are HMM based methods and generally perform better, shown in figures 8, 9, and 10. A large number of sequences, or more precisely a non-redundant set of as many sequences as possible which belong to the protein family in question, is required to build a good HMM (Eddy 1998b). These methods reveal better results if a good HMM is applied otherwise using irrelevant sequences to build an HMM adversely effects the performance of these methods. Similarly, an obvious problem with PSI-BLAST is that if a unrelated chance similarity is mistakenly included in the profile training set, the search algorithm picks up relatives of the unrelated sequence rather than members of the query family (Holms, 2000). A Prosite like pattern or a regular expression constructed either manually or automatically, is the backbone of methods like PHI-Blast. A large number of sequences are required to build a good pattern. Weak motifs are not very sensitive and reveal a limited number of possible homologs. Mast is also a motif-based method and its performance is often worse, as shown in Figures 8, 9, and 10, since it is the only method that does not use the whole sequence but multiple motifs considering the intervening region between motifs as random (Eddy 1998b).

3.2.2. Placing Methods on a Common Numerical Scale

A major problem in combining confidence estimates is the variability in the size of the E-values estimated by different homology-search methods. We rescale E-values to homogenize the confidence estimates in order to combine them. More precisely, to construct a consensus hit list from these data, we first rescale the E-values $E_i(s)$ obtained by the individual methods $i=1, \ldots, n,$ for each sequence s, to produce E-values $E_i^*(s)$ of comparable size. We then use the weights (as

described above) to obtain a weighted average E-value. These two steps are now described in detail.

It is relatively easy to compare the scores that a particular method assigns to distinct data sets rather than scores assigned by different methods. To compare scores that are assigned by different methods, for each method $i=1, \ldots, n,$ and each sequence s in the database, we report the sequence provided its E-value $E_i(s)$ is below or equal to a cut-off value E_C of 1,000. Then, we choose one method as the reference method, on the basis of which the E-values of the other methods are rescaled (Yona et al., 2000). In CHASE, we use HMMsearch as our reference method. Next, before doing any E-value manipulation, we take the logarithm to the base 10 to transform the E-values for all methods. This transformation is necessary since E-values may be very close to zero for good database hits, and we must avoid rounding problems. This way, we obtain, for each sequence s taken into consideration and each method $i=1,\ldots, n,$ a number $e_i(s):=log_{10}E_i(s)$ that we call the "e-value" of the sequence (with a small e), for conciseness. Next, we use a regression procedure such as the Ordinary Least Square (OLS) regression (Gujarati, 1988), yielding the slopes and the intercepts for HMMsearch versus Treesearch, PSI-Blast, PHI-Blast, and Mast, to rescale their e-values. OLS is described in section 3.2.2.1. The slope $a,$ and the intercept b depend on the specific data under consideration - there is no universal data-independent regression line for the various methods. For each sequence s we then put

$$e_{PSI}^*(s) := min \{ a \bullet e_{PSI}(s) + b, e_0 \} \text{ in case } e_{PSI}(s) < e_0, \quad \textbf{(1)}$$

$$\text{and } e_{PSI}^*(s) := e_{PSI}(s) \text{ else.}$$

For a small scaling threshold e_0, the formula rescales small e-values according to the regression line, and keeps large e-values as they are. Keeping large e-values as they are may be useful, because they may be "downscaled" otherwise, suggesting a significance that is not there. In the rare case that rescaled e-values exceed the threshold, they are set to precisely this threshold in order to keep the

ranking as is. For larger e_0, fewer e-values are kept as they are. If we set $e_0 = \log_{10}(E_C) = 3$, no hits are considered for which the E-value exceeds the E-value cut-off $E_C = 1000$, and all values are rescaled in this case. Nevertheless, results improve slightly for smaller e_0, as discussed later on.

The same scaling procedure is applied to the e-values reported by the other three methods. For notational consistency, we set $e^*_{HMM(s)} := e_{HMM(s)}$ for our reference method HMMsearch.

3.2.2.1. Using Ordinary Least Square (OLS) Regression to Rescale E-values

In many problems, two or more variables are inherently related, and it is necessary to explore the nature of this relationship. *Regression analysis* is a statistical technique for modelling and investigation of the relationship between two or more variables. The principal objective in a simple regression analysis is to establish a quantitative relationship (in the form of an equation) between two variables. Simple linear regression is the modelling of n pairs of data (X_i, Y_i) in a linear relationship, where $i=1, \ldots, n$. The relationship between X_i and Y_i is represented in the form $Y_i = aX_i + b$. The estimates of regression coefficients i.e. the slope 'a' and the intercept 'b' should result in a line that is a "best fit" to the data. Ordinary Least Squares or OLS, proposed by German scientist Karl Gauss, is a method that estimates the regression coefficients such that the sum of squared errors is minimized (For the derivation of linear regression equations see Kirchner, 2001).

Table 4. Example Data showing e-values for 22 sequences reported by PSI-Blast and HMMsearch. The column SeqID represent the sequence identifiers while X (PSIBlast) and Y (HMMsearch) present e-values reported by PSI-Blast and HMMsearch, respectively. E-values are transformed using log10 and we call these 'e-values' (i.e. with a small e). X**2 is the square of PSI-Blast e-values and X*Y is the product of PSIBlast and HMMsearch e-values.

No	SeqID	X (PSIBlast)	Y (HMMsearch)	X**2	X*Y
1	d1b4va1	-64.39794	-113.221849	4147.095	7291.254
2	d1coy_1	-64.69897	-105.522879	4185.957	6827.222
3	d1cf3a1	-82	-137	6724	11234
4	d1gpea1	-83.69897	-136	7005.518	11383.06
5	d1fuma2	-56	-97.522879	3136	5461.281
6	d1chua2	-56.30103	-93	3169.806	5235.996
7	d1qlaa2	-61	-97.221849	3721	5930.533
8	d1qjda2	-55.39794	-94.522879	3068.932	5236.373
9	d1d4ca2	-53.09691	-94	2819.282	4991.11
10	d1b37a1	-80.69897	-100.045757	6512.324	8073.59
11	d1gnd_1	-71	-87.39794	5041	6205.254
12	d1b3ma1	-45.69897	-66.221849	2088.396	3026.27
13	d1pbe_1	-45.52288	-69.221849	2072.332	3151.178
14	d1foha1	-33.09691	-44.522879	1095.405	1473.57
15	d1cjca1	-18.09691	-22.30103	327.4982	403.5797
16	d1djna2	-6.30103	-2.69897	39.70298	17.00629
17	d1foha2	-0.69897	1.78533	0.488559	-1.24789
18	d3lada1	-1	0.518514	1	-0.51851
19	d1lpfa1	-3	1.041393	9	-3.12418
20	d3grs_1	0.69897	1.799341	0.488559	1.257685
21	d1ebda1	-0.221849	1.857332	0.049217	-0.41205
22	d1aoga1	1	2.60206	1	2.60206
Sum		-880.2293	-1350.81863	55166.27	85939.83
Sum/n		-40.01042	-61.40084723		

Given the sample e-values for sequences *s* where $s=1, \ldots, n\ (n=22)$ for PSI-Blast ($X_s = e_{PSI(s)}$) *and* for HMMsearch ($Y_s = e_{HMM(s)}$) i.e $\{(X_1, Y_1), \cdots, (X_n, Y_n)\}$, as shown in **Table 4**, Ordinary Least Square (OLS) estimates the slope a and the intercept

b by the following steps, such that the sum of squared errors $\sum_{s=1}^{n}(e_{HMM(s)} - a \bullet e_{PSI(s)} - b)^2$ is minimized and we get a straight regression line:

1. Using data from the Table 4, calculate mean: $\overline{X} = \sum_{i=1}^{n} \frac{X_i}{n} = -40.01042$ and

   mean $\overline{Y} = \sum_{i=1}^{n} \frac{Y_i}{n} = -61.40084723$

2. Calculate the sum of squares (SS):

$$SS_{XX} = \sum_{i=1}^{n}(X_i)^2 - \frac{1}{n}(\sum_{i=1}^{n} X_i)^2$$
$$= 774803.58 - 1/22(55166.27)$$
$$= 19947.929$$

And

$$SS_{XY} = \sum_{i=1}^{n} X_i Y_i - \frac{1}{n}(\sum_{i=1}^{n} X_i)(\sum_{i=1}^{n} Y_i)$$
$$= 85939.83 - (1/22)(-880.2293 * -1350.818639)$$
$$= 31893.008$$

3. Calculate slope a and intercept b as:

$$a = \frac{SS_{XY}}{SS_{XX}} = 31893.008 / 19947.929$$
$$= 1.598813$$

$$b = \overline{Y} - b\,\overline{X} = -61.40084723 - (1.598813) * (-40.01042)$$
$$= 2.5683354$$

If we plot the regression line for the e-values on the log-log scale from data used in the example above, we get a straight line as shown in Figure 12. Once such a relationship between variables is established, by finding out the slope and intercept, it is possible to predict the value of one of the variables, if the value of the other is known.

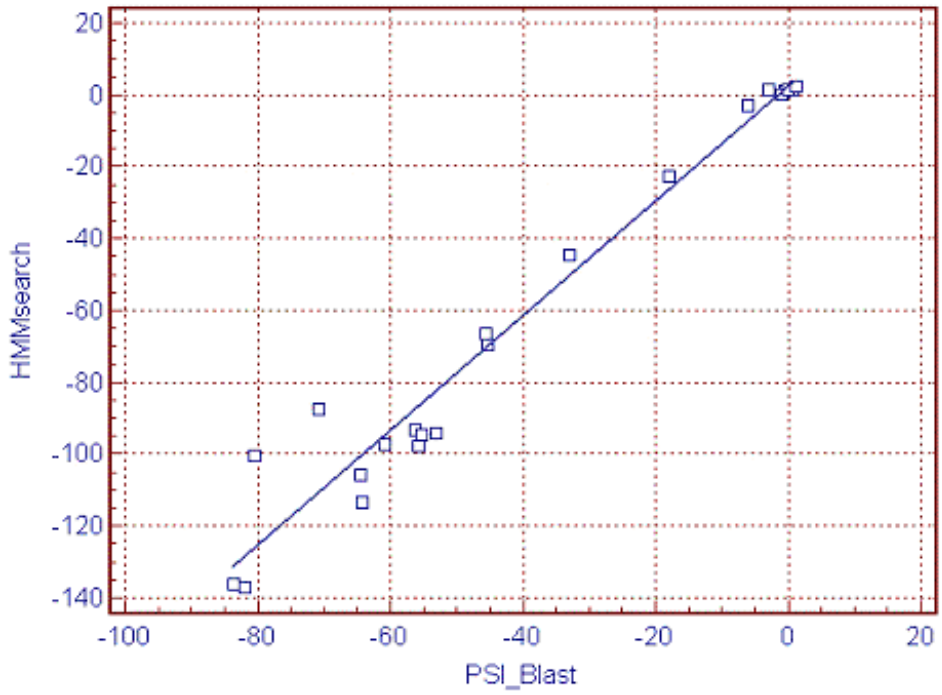

Figure 12. Scatter Diagram showing a straight regression line for HMMsearch and PSIBlast E-values on log-log scale. This diagram was produced using the Demo Version 7.3.0.0 of MedCalc Software, http://www.medcalc.be.

In the example shown above, ordinary least-squares regression applied to HMMsearch e-values $e_{HMM(s)}$ and corresponding PSI-Blast e-values $e_{PSI(s)}$ provides the slope a and the intercept b for which the sum $\sum_{s=1}^{n}(e_{HMM(s)} - a \bullet e_{PSI(s)} - b)^2$ is minimized. Here, the sum is taken over all sequences s with both e-values $e_{HMM(s)}$ and $e_{PSI(s)}$ below or equal to a certain threshold e_0.

3.2.3. Calculating Combined E-value or the C-value

Once we have got the rescaled e-values $e^*_1,\ldots,e^*_n$ for all n methods, we calculate the *c-value* for each sequence *s* as the W-weighted sum:

$$c\text{-}value(s) := \sum_{i=1}^{n} e_i^*(s) \bullet W_i . \qquad \textbf{(2)}$$

The final C-value (on the original E-value scale) is then obtained as *C-value(s)*: = $10^{\,c\text{-}value(s)}$. This yields a consensus over individual homology-search methods. "Missing E-values" arise if a homology search method finds a sequence not found by another, given the E-value cut-off E_C. In the c-value formula, these are set to the cut-off e-value (log10 of E_C).

3.3. CHASE Evaluation Methodology

As noted above, our tool CHASE implements the above scheme using five homology-search methods. Using the weights $W_1, \ldots, W_n$ of the component search algorithms calculated once and for all, we compute the regression lines and the resulting C-values of the sequences in each database search. Treating the C-values as E-values, we can use Phase4 again to evaluate the performance of CHASE and to compare its performance with that of the component algorithms. Clearly, the weights that are incorporated in (and thus the performance of) CHASE depends on the database that was used for determining these weights. In particular, if a component algorithm does very well on that database, it will get a high weight implying that it will strongly influence the outcome of the consensus method, making it look good on that particular database, too.

To avoid this kind of circularity, we have split the SCOP database (version 1.53) into two separate databases: the odd database, containing every second SCOP superfamily starting with the first one, and the even database, containing the rest. We used the odd database to compute the weights, $W_1, \ldots, W_n$, as listed in Table 2, and the even database to evaluate the performance of the resulting consensus

method and to compare this performance with that of its component algorithms, using again the three scenarios offered by Phase4 as described in Table 1. As before, we used "*coverage versus false positive count"* in Phase4 as a performance *evaluator*, and sorting of CHASE hits was based on the C-value. Sequences with a C-Value exceeding E_C = 1000 are not listed. By default, CHASE sets the E-value cut-off E_C to 1,000, and the e-value threshold used for rescaling e_0 to *3 (*=$\log_{10}$*1000)* so that all values are rescaled. However, other cut-off values can also be specified.

3.4. CHASE Evaluation, Results and Discussion

We conducted a comparative evaluation of five homology-search methods and our consensus method *Comparative Homology Agreement Search* (CHASE). We used three different scenarios offered by Phase4, as listed in Table 1, to define distant, close, and very close relationship between SCOP database entries. If one considers the averaged coverage of true positives at the cost of *zero* false positives, as shown in Figure 13, and ranks the methods according to their ability to find *distant* homologous proteins, CHASE obtains a coverage of 34%, and HMMsearch comes next with a coverage of 28%. Then come Mast, PSI-Blast, Treesearch, and PHI-Blast, with coverages between 27 and 21%. It is important to note that we do not claim to conduct a valid comparison of these individual methods. Such a comparison would need to do more justice to their different input requirements. The comparative analysis of the individual methods, starting with the *same* training data of sequences for each, suffers from the application of the *Input Processors* (described above) by which some of the input information may be lost. It is also worth noting that methods that do not perform well on average can still give excellent results in specific instances - a remarkable fact that clearly needs to be investigated further.

If we plot coverages of true positives at the cost of 10 false positives, performance of CHASE goes up, covering 47% on average in case of distant relationship, compared to 38% coverage by HMMsearch. Permitting 50 false positives, as presented in Figure 13, these numbers go up to 59% and 49%, respectively.

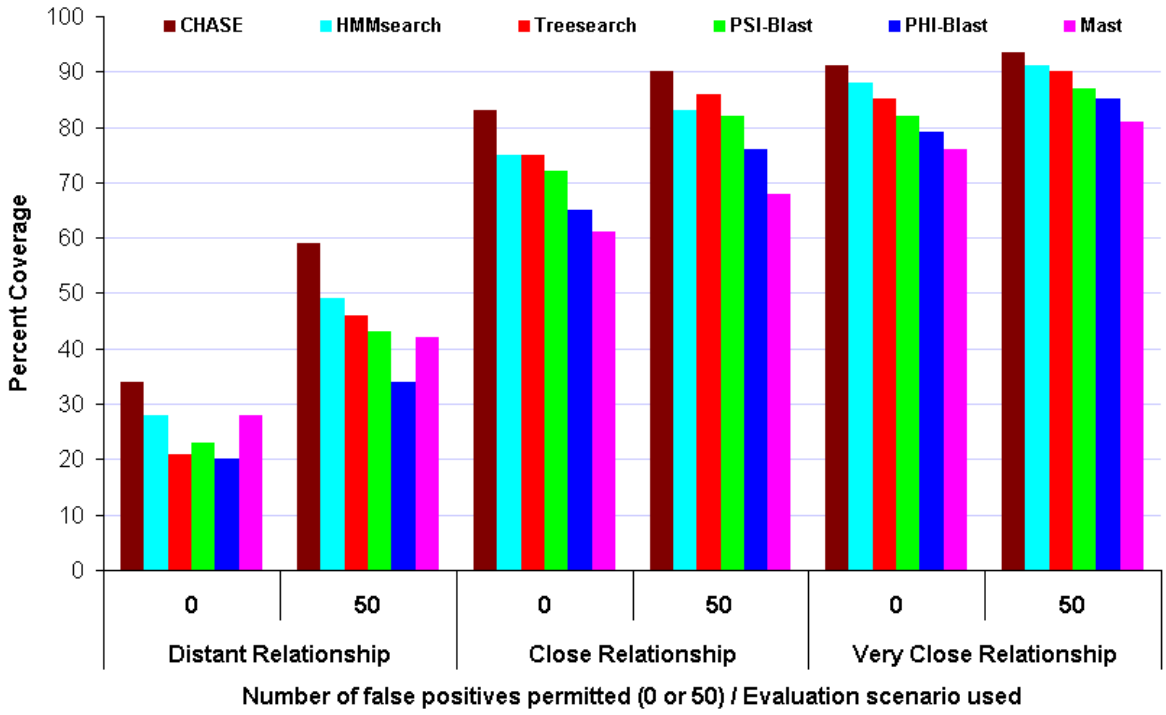

Fig. 13. Average coverage of CHASE and its component algorithms

The averaged coverage of true positives permitting zero and fifty false positives using SCOP (*even* half) as the target database and evaluation scenarios provided by Phase4 (as described in Table 1)

The advantage of CHASE is smaller in case of close, and very close relationship scenarios, but it still outperforms all component methods by a good margin. The "Coverage versus false positive count" plots in Figure 14, 15, and 16 for the various Phase4 scenarios give a more detailed picture of the coverage of true positives, for up to 200 false positives.

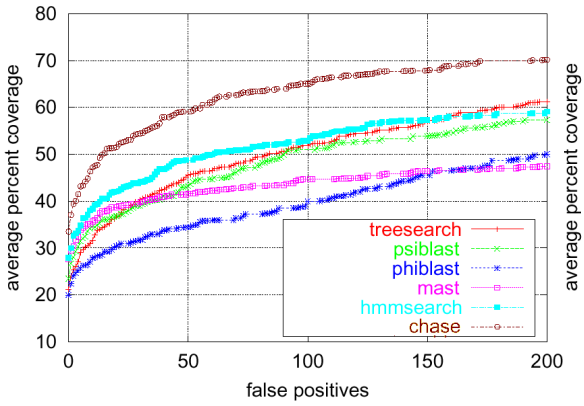

Figure 14.

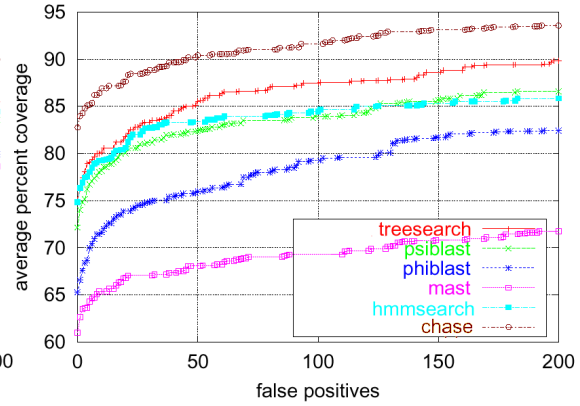

Figure 15.

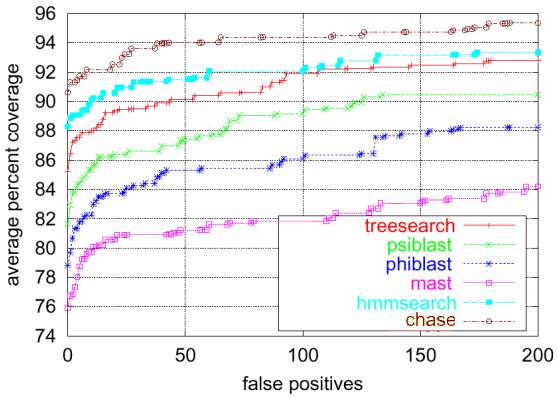

Figure 16.

Figures 14, 15 & 16: Coverage versus false positive counts

This figure shows the Phase4 evaluation in the form of "*coverage versus false positive counts"* for CHASE as well as for the five component algorithms, using three different *scenarios* (as described in Table 1) offered in Phase4. Averaging is done over all SCOP families included in the *even* half of the database. (The *odd* half was used to determine the weights used by the CHASE combination scheme.) It is notable that CHASE achieves a remarkable coverage, better than any one of its component methods, of 47% for just 10 false positives as shown in Figure 14. CHASE exploits the fact that most of the true positives above the twilight zone are, usually, reported by all methods but within the twilight zone these are not reported by all methods. CHASE gives a higher weight to hits which are reported by a highly weighted method. A lower E-value assigned by

any CHASE component method further places such a hit at a higher ranking in the list.

No	C-value	Description	HMMsearch	Treesearch	PSI-Blast	PHI-Blast	Mast
1	2e-115	3.3.1.2.2 Cholesterol oxidase	6e-114	2e-85	5e-100	2e-154	3e-126
2	4e-114	3.3.1.2.1 Cholesterol oxidase	3e-106	8e-86	2e-100	2e-149	1e-133
3	7e-103	3.3.1.2.7 Glucose oxidase { Asp	1e-137	8e-86	7e-127	2e-23	2e-139
4	9e-101	3.3.1.2.8 Glucose oxidase { Pen	1e-136	1e-85	2e-129	6e-13	3e-137
5	1e-85	3.3.1.4.2 Fumarate reductase f	3e-98	9e-87	3e-87	1e-64	2e-90
6	5e-83	3.3.1.4.1 L-aspartate oxidase	1e-93	2e-85	1e-87	4e-68	5e-78
7	1e-76	3.3.1.4.3 Fumarate reductase f	6e-98	2e-86	7e-95	6e-17	5e-84
8	2e-74	3.3.1.4.4 Flavocytochrome c3 (	3e-95	6e-86	2e-86	4e-08	1e-94
9	5e-72	3.3.1.4.5 Flavocytochrome c3 (	1e-94	5e-86	8e-83	0.002	6e-91
10	1e-64	3.3.1.2.9 Polyamine oxidase { M	9e-101	6e-76	7e-125	3	4e-08
11	2e-59	3.3.1.3.1 Guanine nucleotide d	4e-88	6e-81	4e-110	0.03	0.001
12	2e-48	3.3.1.2.5 Sarcosine oxidase { B	6e-67	7e-82	1e-71	1e-05	8e-08
13	1e-47	3.3.1.2.3 p-Hydroxybenzoate hy	6e-70	5e-79	3e-71	0.1	7e-07
14	4e-40	3.3.1.2.6 Phenol hydroxylase {	3e-45	5e-74	2e-52	3e-10	3e-13
15	7e-27	3.3.1.1.3 Adrenodoxin reductas	5e-23	1e-76	2e-29	30	0.167
16	7e-17	3.3.1.1.2 Trimethylamine dehyd	0.002	8e-65	2e-11	6e-05	1.00E+03
17	1e-13	3.3.1.2.6 Phenol hydroxylase {	61	3e-64	2e-06	400	19.6
18	0.008	3.3.1.5.8 Dihydrolipoamide deh	11	0.0002	0.007	1	2e-07
19	0.02	3.3.1.5.8 Dihydrolipoamide deh	3.3	0.001	0.003	0.7	3e-05
20	0.04	3.3.1.5.1 Glutathione reductas	63	0.002	0.6	3	1e-07
21	0.135	3.3.1.5.9 Dihydrolipoamide deh	72	0.001	0.03	7	0.001
22	0.44	3.3.1.5.4 Trypanothione reduct	400	0.333	3	0.4	1e-05
23	1.71	3.3.1.5.10 Dihydrolipoamide de	28	0.004	1.00E+03	600	5e-05
24	2.66	3.3.1.5.8 Dihydrolipoamide deh	410	0.01	1.00E+03	300	1e-05
25	2.74	5.8.1.3.1 T7 RNA polymerase { Ba	**43**	0.001	10	20	**18.1**
26	3.22	3.3.1.5.3 Trypanothione reduct	140	0.14	4	100	0.01
27	4.42	3.3.1.5.5 Thioredoxin reductas	520	0.184	0.6	40	0.471
28	11.1	3.1.8.2.1 beta-Amylase { Soybean	1.00E+03	**3e-05**	400	600	23.4
29	11.8	3.84.1.1.1 Asparaginase type II	340	1.00E+03	**2e-06**	600	1.00E+03
30	15.6	3.72.1.1.1 Alkaline phosphatase	550	0.0001	400	60	1.00E+03
31	16.6	3.3.1.5.7 NADH peroxidase { Str	1.00E+03	1.00E+03	0.2	200	0.01
32	16.8	3.3.1.5.2 Glutathione reductas	1.00E+03	52.9	9	2	0.556
33	19.4	3.36.1.1.2 Subtilisin Carlsberg	150	0.003	100	80	885
34	22.1	4.81.1.9.2 Neutrophil collagena	1.00E+03	9e-05	200	500	1.00E+03
35	25.5	3.75.1.1.1 Tryptophan synthase,	940	0.01	2	600	1.00E+03
36	26.1	1.83.1.2.1 Catechol oxidase { Sw	1.00E+03	0.06	50	**5**	1.00E+03
37	26.2	2.75.1.6.1 Chondroitinase B { Fl	790	0.004	60	400	202
38	28	4.147.1.1.10 Lactate dehydroge	1.00E+03	0.04	1	600	1.00E+03
39	29	2.75.1.1.2 Pectate lyase { Erwin	960	0.0003	1.00E+03	90	1.00E+03
40	30.2	4.140.1.1.1 L-aminopeptidase D-	430	0.02	10	600	672
41	32.5	3.65.1.6.2 Proline iminopeptida	190	0.794	0.6	600	1.00E+03

Figure 17. Sample CHASE result

CHASE result for superfamily 3.3.1 (SCOP version 1.53), featuring the FAD/NAD(P)-binding domain are shown. The hits are sorted by C-value. Rescaled E-values (as calculated by the scaling formula (2) in the text, but

displayed in terms of the original E-value scale not taking the logarithm) are presented in the 5 columns on the right. The first 24 CHASE hits are all true positives. The false positives (numbers 25, 28-30, 33-41) and the respective minima of their E-values in each column are marked in red. E-values in the first 24 rows and the last 5 columns that are larger (and, hence, "worse") than these respective minima are marked in orange. They indicate where forming consensus C-values was more successful than the corresponding single method. (Consider, for example, the HMMsearch E-values presented in the first of the last 5 columns. The minimum of these values taken over all false positives is 43, and the values in rows 17, 20, 21, 22, and 24 are larger than 43 and, hence, marked in orange.) Apparently, each single method addresses different aspects of (super) family membership, and a strong showing for some method(s), not counterbalanced by very poor showings for others, seems to be a good membership indication that is (independently of which single method is involved) picked up by our consensus approach.

If the e-value threshold used for rescaling is set to -1 instead of 3, not all values are rescaled anymore in the c-value formula **(2).** Remarkably, CHASE appears to perform even slightly better in this case, for example, it obtains 36% coverage of distant relatives at a cost of zero false positives (+ 2%), 50% permitting 10 false positives (+ 1%) and 60% coverage permitting 50 false positives (+ 1%).

The results of running CHASE for the SCOP superfamily featuring the FAD/NAD (P)-binding domain are shown in Figure 17. C-values along with rescaled E-values from different methods are printed. The names of the sequences from the given family are printed in black (in the “description” column), the others (the names of the false positives) in red. We consider a family member to be classified correctly by method i if its rescaled E-value is smaller than the rescaled E-value of the first false positive. For the false positives listed by method i, the minimum rescaled E-value is printed in red. Rescaled E-values of family

members that would not be classified correctly using method i alone are marked in orange. They are larger than the smallest rescaled E-value of the false positives for method *i* (printed in red) so that the false positives with the smallest rescaled E-Value would precede the family members in the ranking based on method i. In the twilight zone of rows 15 to 24, CHASE performs well, triggered by the rescaled E-values marked in green that indicate success for at least one method. Inspecting the consensus hit lists for all protein families under consideration in the "Distant relationship" scenario, we noted that each method detects specific true positives that would not be detected if we had restricted ourselves to combining the other four.

4. Improvements on CHASE

CHASE performed very well in finding the remote homologs of protein families, as shown in figure 14, 15, and 16. However, only one alignment scheme i.e. ClustalW was used wherever required. To further improve CHASE in terms of its performance on the one hand and its generality (for example to accommodate new homology search method(s) or alignment scheme(s)) on the other hand, we made some enhancements resulting in CHASE2. These enhancements, discussed in detail in the following sections, include the implementation of a modular structure, use of optimum input processors (based on testing the effect of different alignment schemes) and re-calculation of E-values, an approach that is theoretically more sound than the regression scheme but not applicable to all component methods. We re-evaluated CHASE after the implementation of these enhancements and got a better performance than before.

4.1. Modular Structure of CHASE

Separate module files, namely IPs.pm, Run_Parse.pm and DBreader.pm were written for CHASE2. In addition, a driver script was written that reads an XML (eXtensible Mark-up Language) (Achard et al., 2001) configuration file and calls these modules to carry out one complete CHASE run. In the following section all these are discussed in detail.

4.1.1. CHASE Modules

- **IPs.pm**

The input set of sequences is first validated by the CHASE driver script and then passed on to the Input Processors (IPs.pm) module. The Input Processors module, as shown in Figure 18, transforms the given set of sequences into a specific data format (for example an alignment, an HMM, a phylogenetic tree, a

Prosite like pattern or a MEME motif profile) that is required by a particular homology search method, as discussed in section 3.1.3. All of the homology search methods, except Mast, use an alignment in one way or the other. For example, PSI-Blast requires an alignment in a specific format. An HMM is required by HMMsearch and Treesearch, a phylogenetic tree is required by Treesearch, and a consensus sequence is required by PHI-Blast. These all are based on an alignment scheme. Before the implementation of modular structure of CHASE it required a significant amount of changes in the CHASE script in order to use an alignment scheme other than the default ClustalW. Further, build_compound, an input processor for Treesearch input, was not compatible with any other alignment scheme.

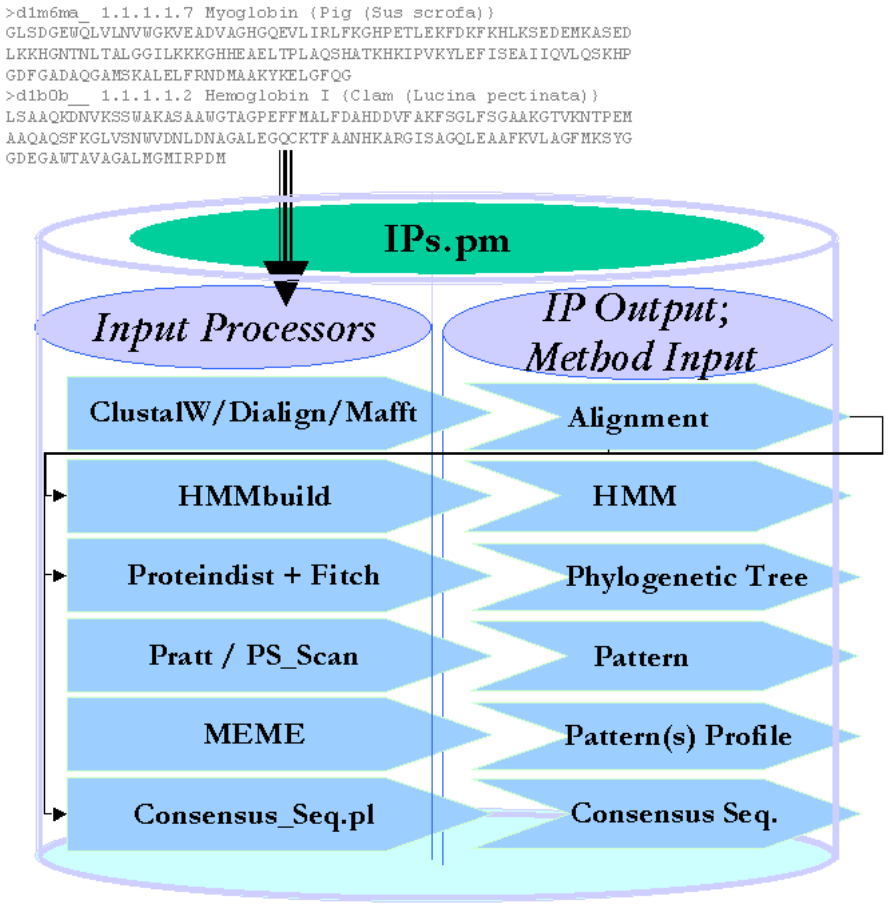

Figure 18. An outline of Input Processors (IPs.pm) Module

Input processors are used to generate specific inputs required by different homology search methods. Pattern and pattern profile are generated directly

from the input set of sequences, while the HMM, the phylogenetic tree, and the consensus sequence are based on an alignment.

The newly implemented input processors module, as shown in Figure 18, made it possible to easily use any of the ClustalW, Dialign or Mafft alignment schemes, whenever it is required to generate an HMM, a phylogenetic tree or a consensus sequence. (Pratt/PS_Scan and MEME produce a pattern or a pattern profile, directly using the input set of sequences.) To save time and computing resources, once a specific input such as an alignment is produced that is used by a particular method, it is not generated again if it is required by another homology search method.

In this module we also overcome the limitations of build_compound to use alignments other than ClustalW. The function of build_compound was to generate a ClustalW alignment, an HMM using HMMbuild and a phylogenetic tree, using proteindist (available with the Phylip or the Treesearch package) and the "Fitch" program, based on the ClustalW alignment. To get rid of build_compound, we make direct use of proteindist that calculates the phylogenetic distances among protein sequences, given as an alignment, and "Fitch" to generate a phylogenetic tree. Now, we can handle any of the above-mentioned alignment algorithms using IPs.pm. We implemented a version of the "Fitch" program that has the functionality to use input on the command-line and generate user-defined filenames for the output phylogenetic tree.

As a specific example of the PERL module IPs.pm, a subroutine from the PHI-Blast IP, called "construct_pattern", is shown in Figure 19. This subroutine is implemented to construct a pattern in a format that is compatible with PHI-Blast, given a set of sequences and a pattern finding program (i.e. PS_Scan or PRATT).

```
1  sub construct_pattern {
2    my ($SeqFastaFile, $PattProg, $AlnProg)=@_;
3    my $patternfile;
     # Check (-e) if the Fasta File exists
4    if (-e $SeqFastaFile){
5      if ($PattProg = ~/ps_scan/i){
6          $patternfile = $SeqFastaFile;
7          $patternfile = ~s/\.fasta/\.ps_scan/;
         # Check if the pattern file size is non-zero (that it already exists)
8        if (-s $patternfile){print "IPs.pm:WARNING: $patternfile already exists\n";}
9        else{
10        my $ps_scan_temppatternfilename = $SeqFastaFile;
11        $ps_scan_temppatternfile = ~s/\.fasta/\.ps_scantmp/;
          # Run PS_Scan program to get pattern in raw (temp) form
12        my $ps_scan_temppatternfile = run_ps_scan($SeqFastaFile, $AlnProg,
13                                                  $ps_scan_temppatternfilename);
          # Reformat PS_Scan raw output to PROSITE format
14        $patternfile = refo_pspat_2_prospat($ps_scan_temppatternfile, $pattenfile);
          # Check if the pattern file size is zero (empty file)
15        if ( -z $patternfile){
16          print "\n\nERROR: No Prosite-like Pattern Found In The Input Sequence(s),".
17              " Try Pratt\n";
18        }
19      }
20   }
21   if ($PattProg = ~/pratt/i){
22       $patternfile = $SeqFastaFile;
23       $patternfile = ~s/\.fasta/\.pratt/;
24     if (-s $patternfile){print "IPs.pm:WARNING: $patternfile already exists\n";}
25     else{
26         my $pratt_temppatternfile = $SeqFastaFile;
27         $pratt_temppatternfilename = ~s/\.fasta/\pratt_tmp/;
           # Run PRATT program to get its output
28         my $pratt_temppatternfile = run_pratt($FastaFile, $pratt_temppatternfilename);
           # Extract PROSITE format pattern from PRATT output
29         $patternfile = extract_prattpat($pratt_temppatternfile, $patternfile, $SeqFastaFile);
30       if ( -z $patternfile){
31         print "\n\n\tERROR: Pratt produced no Pattern from The Input Sequence(s)\n";
32         print "\t PHI-Blast Run 'll Fail\n";
33       }
34     }
35   }
36   return $patternfile;
37 }else{
38   print "IPs.pm FATAL_ERROR: construct_pattern subroutine requires".
39     $SeqFastaFile ."that does not exist\n";
40   return 0;
41 }
42 }
```

Figure 19. The PHI-Blast IP subroutine "construct_pattern" is shown. It explains that, given a set of sequences and a pattern finding tool (e.g. PRATT or PS_Scan), how the PROSITE-like pattern is generated. Such a pattern is required to execute a PHI-Blast search.

```
1  sub refo_pspat_2_prospat {
2    my ($InFile, $OutFile) = @_;

3   open (PSSCANRE,"$InFile") || die "Can't Open PS_SCAN Temp Out File\n";

4   while (<PSSCANRE>){
5     my $line = $_;

6     if ($line = ~s />| ://g){
7       my ($ProtName, $PatAC, $PatID) =  split ' ', $line;

8       $Patterns->{$PatAC}{pid} = $PatID;
9     }

10    if ($line = ~s /PA  //g){

11      $Expression = $line;
12      $Expression = ~s /X|B|Z|\(|\)//g;              #Remove X, B, Z, ( or ) characters
13      $Expression = ~s /\./x-/g;                     #Reformat . to x-
14      $Expression = ~s /\((\d+)\)/\{$1\}\-/g;        #Reformat e.g (8) to {8}-
15      $Expression = ~s /\[\^([A-Za-z]+)\]/{\1}\-/g;  #Reformat e.g [^FYWHP] to {FYWH}
16      $Expression = ~s /\[([A-Za-z]{1})\]/\1\-/g;    #Reformat e.g [G] to G-
17      $Expression = ~s /\]/\]-/g;                    #Insert - between ][
18      $Expression = ~s /\-$//g;                      #Remove the - at the end
19      $Expression = ~s /\-\(/\{/g;                   #Reformat -( with {

20      $Patterns->{$PatAC}{pat} = $Expression;
21    }
22  }
23  close PSSCANRE;

24  open OUT, ">$OutFile" || die "..Can't Open $OutFiles ($!)\n";

25  foreach my $key (sort keys %{$Patterns}){

26    print OUT "AC  $key\nID  $Patterns->{$key}{pid}\nPA  $Patterns->{$key}{pat}\n\n";

27  }
28  close OUT;

29  return $OutFile;
30 }
```

Figure 20. The subroutine "refo_pspat_2_prospat" showing the perl script that reformats the pattern extracted using Ps_Scan, to be used to run PHI-Blast.

The first part of the main subroutine (lines 5-20), shown in Figure 19, deals with obtaining the Prosite like pattern using PS_Scan. The standard version of the Ps_Scan program scans a protein sequence against the Prosite database to report the occurrence of a pre-defined pattern. As an output PS_Scan reports only the pattern id from the Prosite database and the matching region but not the actual regular expression pattern. Therefore, the PS_Scan program was modified so that it reports the regular expression pattern in the following form:

```
>serpin : PS00284 SERPIN Serpins signature.
    396 - 406  LfFNKPFLFlI
PA    ([LIVMFYX])(.)([LIVMFYACX])([DBNBQZX])([RKHQZSX])([PSTX])([FX])
([LIVMFYX])([LIVMFYCX])(.)([LIVMFAHX])
```

To reformat such a pattern into Prosite format, a subroutine was implemented, called “refor_ps_2_Prosite” which is marked in blue colour in Figure 19 and shown in detail in Figure 20. Lines 11-19 of this subroutine, shown in Figure 20, reformat the above-mentioned pattern into Prosite pattern format that is compatible with PHI-Blast, as shown here:

```
AC      PS00284
ID      SERPIN
PA      [LIVMFY]-x-[LIVMFYAC]-[DNQ]-[RKHQS]-[PST]-F-[LIVMFY]-[LIVMFYC]-x-
[LIVMFAH]
```

The second part of the main subroutine (lines 21-35) in Figure 19 prepares the Prosite like pattern based on a given set of sequences using the PRATT program. A lot of information is reported in the standard output of PRATT, part of which is shown in Figure 21.

```
Best Patterns before refinement:
         fitness     hits(seqs)    Pattern
    1:   36.5305      4(   4)   K-x-S-I-x(1,2)-T-x(3,4)-K-x(5)-L-G-I-T
Best Patterns (after refinement phase):
         fitness     hits(seqs)    Pattern
A   1:   54.7170      4(   4)   K-[FL]-S-I-x(1,2)-T-x(3,4)-K-[EPS]-[LV]-[FL]-[GV]-[EHK]-L-G-I-T
Best patterns with alignments:
         fitness     hits(seqs)    Pattern
A   1:   54.7170      4(   4)   K-[FL]-S-I-x(1,2)-T-x(3,4)-K-[EPS]-[LV]-[FL]-[GV]-[EHK]-L-G-I-T
Occurrences: 4(4)
     sp|P22324 :   299-   318: nvhlp KLSIsgTydl-KEVLGHLGIT nvfsg
     sp|P23035 :   309-   328: tvhfp KLSIsgTydl-KPLLGKLGIT qvfsd
     sp|P22325 :   301-   320: nvhlp KLSIsgTydl-KEVLGHLGIT nvfsd
     sp|Q9J5F9 :   260-   279: svsip KFSIq-TqhniKSVFVELGIT difde
```

Figure 21. Part of PRATT output showing the patterns and related information extracted from a set of sequences.

```
1  sub extract_prattpat {
2      my ($InFile, $SeqFastaFile, $OutFile) = @_;

3      my $FamilyName = $SeqFastaFile;
4      $FamilyName =~s /\.fasta//;
       #Remove the path from the fasta file to get the protein family name
5      $FamilyName =~ s/.*\///;

6      my $pats = {};
7
8      open $fh, "$InFile" || die "Can't Open $InFile ($!)\n";
9      while (<$fh>) {
10       my $line = $_;

         #go to lines containing pattern specific symbols
11       if ($line =~ /A   1:   |B   2:   |C   3:   |D   4:   /g) {

           #Construct an array of elements based on spaces in the line
12         my @Elements = split "  ", $line;
           #Consider first element of the array as the pattern ID
13         my $id = $Elements[0];
           #Last element of the array is the required pattern
14         my $pattern = $Elements[-1];
15         chomp $pattern;
           #Store the pattern and its id in the hash table
16         $pats->{$id} = $pattern;
17       }
18     }
19     close $fh;

       #Create a new file to write the pattern stored in the hash
20     open OUT, ">$OutFile" or die "Can't Create $OutFile ($!)\n";

21     my $count = 1;
22     foreach my $key (keys %{$pats}) {
23       print OUT "ID  PRATT_$FamilyName$count\nPA  $pats->{$key}\n\n";
24       $count++;
25     }
26     close OUT;

27     return $OutFile;
28 }
```

Figure 22. The subroutine "extract_prattpat" showing the perl script that extracts the Prosite like pattern from the PRATT program output, to be used to run PHI-Blast.

A subroutine was implemented called "extract_prattpat", marked in blue colour in Figure 19 and shown in detail in Figure 22, which extracts the required pattern from the PRATT output. Pattern extracted from the PRATT output looks like the following, and it is compatible with PHI-Blast.

```
ID   PRATT_serpin1
PA   K-[FL]-S-I-x(1,2)-T-x(3,4)-K-[EPS]-[LV]-[FL]-[GV]-[EHK]-L-G-I-T
```

- **RunParse.pm**

This module as shown in Figure 23, deals with the execution of homology search methods, given the required inputs. One method is executed at a time. When a particular database search is completed its report is parsed, using the parsing scripts implemented in RunParse.pm, to extract some specific information such as the sequence identifiers and the E-values of hits. This information is returned in the form of a method-specific table to the driver script for further analysis.

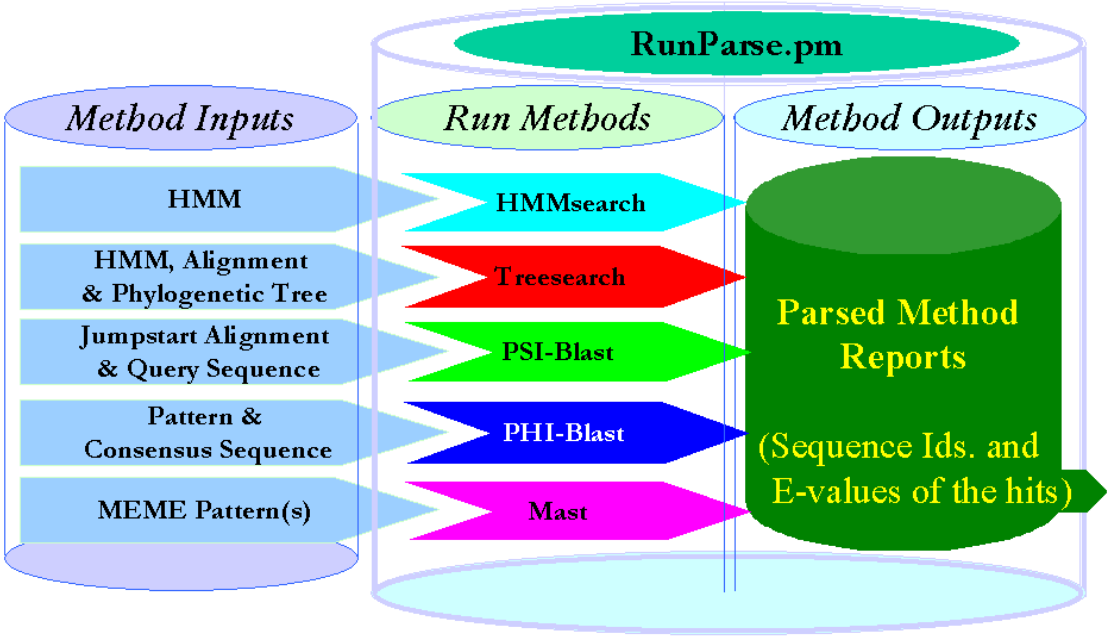

Figure 23. An outline of the Run and Parse (Run_Parse.pm) Module

Given the specific method input, prepared using input processors, homology search methods are executed one by one and their output is parsed to get the specific information that is used later by the CHASE driver script for further analysis.

The Run_Parse module contains a subroutine for each of the database search methods that extracts the specific information mentioned above. Such a subroutine, as shown in Figure 24, requires the output of a database search

method, the name of the database search method and the regular expression to extract the lines containing the required information.

```
sub parse_database_search_report {

    my ($report,  $reg_expression, $method)=@_;

    my $result={};
    my $count_hits=0;

    open REP, $report or die "Can't open $report ($!)\n";

    while (<REP>){

        if ($_=~/$reg_expression/){

        my @items=split / \t/, $_;

        my $seqid=$item[0];
        my $evalue=$item[-1];

        $result->{$seqid}{method}{evalue}=$evalue;

        $count_hits++;

        }
    }

   return ($count_hits, %{$result});

}
```

Figure 24: A subroutine showing the parsing of database search reports to extract the sequence identifiers and the E-values of hits.

As shown in Figure 24, given the name of the output filename, the regular expression and the name of the database search method, this subroutine initialises a perl hash to store the sequence identifier and a hits counter to count the number of hits, as shown in lines 2-3. The subroutine opens the database search report file, as shown in line 4, and exits if the file cannot be opened. Once the file is opened, it starts a while loop to read through the lines of the report file (line 5) and searches for the given regular expression (line 6). If a line containing the given regular expression is reached, it splits the line on the basis of the tab

delimiter and stores the resulting text into an array variable (line 7). This example subroutine stores the first element of such an array as a sequence identifier and the last element as an E-value in the result hash, as shown in line 10 of the subroutine in Figure 24. In line 11, the hits counter is incremented. Once the loop through all the required lines of the database search report is completed, this subroutine returns the hash (containing the sequence identifiers and E-values of all the hits) and the number of hits, as shown in line 14, to the CHASE driver script.

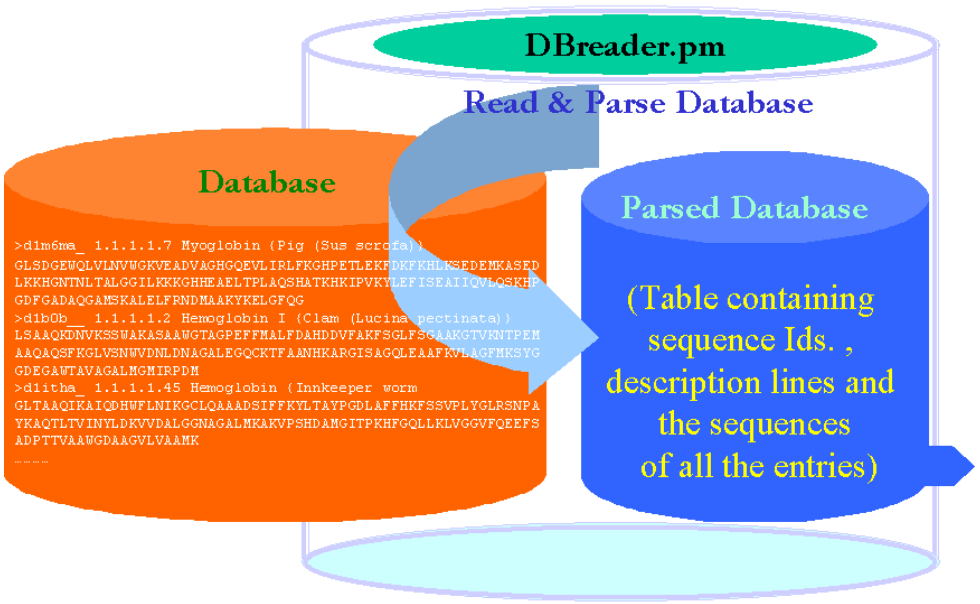

Figure 25: An outline of Database Reader (DBreader.pm) Module

The database Reader module reads the sequence database, given in Fasta format, to extract some specific information that is then passed back to the CHASE driver script.

- **DBreader.pm**

The database reader (DBreader.pm) module, as shown in Figure 25, is written to read the Fasta formatted database such as SWISSPROT or SCOP to report its size and extract the information such as the sequence identifiers, description lines and the sequences of all the entries. This information is then stored in a table that is later used by the driver script.

4.1.2. CHASE Configuration File

In modular CHASE we make use of a configuration file so that one may be able to apply the user-defined configurations without changing the CHASE main script. The configuration file contains information such as the path to several tools, databases and directories, and method specific information such as the name of the method, its class (e.g. whether it is classified as an alignment based method), the alignment scheme that it may use and its weight, etc.

The CHASE configuration file, as shown in Figure 26, follows the conventions of the eXtensible Mark-up Language (XML). In XML format one has to place the contents enclosed in a specific opening and a closing tag, in a hierarchical fashion. For example we start the CHASE configuration file with a main opening tag ‘<CHASECONF>’. We have a section for paths tagged ‘<Paths>’ and a section for the method-specific information, one per component method, tagged ‘<Method>’. Within the paths or the method section each element has its own opening and closing tags. At the end of each section it has its closing tag and the whole document ends with the main closing tag ‘</CHASECONF>’. A particular browser, such as Internet explorer or Mozilla, that recognizes the XML syntax, highlights the tags and differentiates the actual data that is enclosed within these tags. In an example XML document (opened in the Mozilla browser) as shown in Figure 26, sections or subsections start with a negative sign (-) where the data is

shown and the ones where the data is hidden start with a positive sign. Clicking on the positive sign of a section or subsection shows the actual data inside the tags.

```
- <CHASECONF>
  - <Paths>
      <PERLPATH>/vol/perl-5.6.1/bin/perl</PERLPATH>
      <SWISSPROT>/vol/project/chase-1.0/share/biodata/fasta/sprot_test.fas</SWISSPROT>
      <TREMBL>/vol/project/chase-1.0/share/biodata/fasta/trembl_test.fas</TREMBL>
      <TREMBLNEW>/vol/project/chase-1.0/share/biodata/fasta/tremblnew_test.fas</TREMBLNEW>
      <PROSITE>/vol/project/chase-1.0/share/biodata/prosite/prosite.dat</PROSITE>
      <SCOP>/vol/phase4/addons/pdb90d_1.53.sf_split.even</SCOP>
      <HMMER>/vol/project/chase-1.0/share/hmmer-2.1.1/binaries/</HMMER>
      <BLASTPGP>/vol/project/chase-1.0/share/blast/blastpgp</BLASTPGP>
      <MEME_MAST>/vol/project/chase-1.0/share/meme.3.0.4/bin/</MEME_MAST>
      <TREESEARCH>/vol/project/chase-1.0/share/treesearch-0.4/bin/</TREESEARCH>
      <PRATT>/vol/project/chase-1.0/share/pratt/pratt</PRATT>
      <PS_SCAN>/vol/project/chase-1.0/share/ps_scan/ps_scan.pl</PS_SCAN>
      <FITCH>/vol/project/chase-1.0/share/phylip-3.5/fitch2</FITCH>
      <FITEVD>/vol/project/chase-1.0/share/fitevd/fitevd</FITEVD>
      <CLUSTALW>/vol/project/chase-1.0/share/clustalw1.83/clustalw</CLUSTALW>
      <MAFFT>/vol/project/chase-1.0/share/mafft/scripts/fftns</MAFFT>
      <READSEQ>/vol/project/chase-1.0/share/readseq/readseq</READSEQ>
      <SREFORMAT>/vol/project/chase-1.0/share/hmmer-2.1.1/binaries/sreformat</SREFORMAT>
      <DIALIGN>/vol/project/chase-1.0/share/dialign2_dir/dialign2-2</DIALIGN>
      <CONSENSUS>/vol/project/chase-1.0/share/consensus_seq/consensus.pl</CONSENSUS>
      <CHASE_OUTPUT>tmp</CHASE_OUTPUT>
      <TEMP>tmp</TEMP>
    </Paths>
  - <Method>
      <Name>HMMsearch</Name>
      <Weight>0.22048</Weight>
      <Class>ALN</Class>
      <IParg1>mafft</IParg1>
      <IParg2>HMM</IParg2>
      <IParg3>None</IParg3>
    </Method>
  + <Method>
  + <Method>
  + <Method>
  + <Method>
  </CHASECONF>
```

Figure 26: CHASE configuration file, an example

The CHASE configuration file is implemented in XML format. It starts with a main opening tag and within that the user-defined paths to several tools, databases and directories can be implemented. User-defined method specific information is placed in the methods section.

4.1.3. CHASE Driver Script

Given a set of sequences as input and the user-defined/default options, it is the CHASE driver script that integrates all of the above mentioned modules to get the information that is processed further to carry out a complete CHASE run, as

shown in Figure 27. The CHASE driver script works in stages S0-S6. In the initial stage (S0), it validates the input set of sequences and user-defined/default options. It then reads the configuration file in stage S1 and verifies that all the tools, databases and the directories are accessible. In the second stage (S2), it instructs the input processing or the IPs.pm module, as mentioned above, to reformat the input data into a format that is required by the homology search methods.

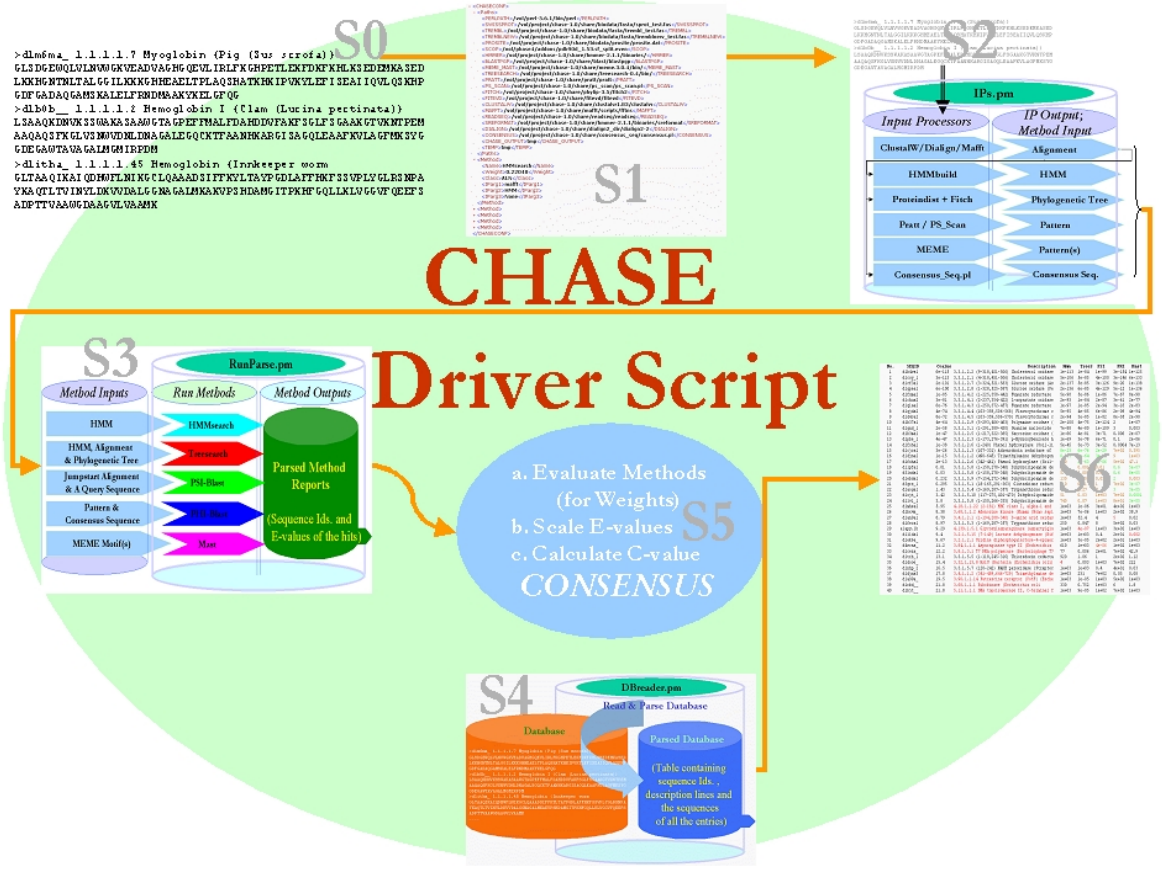

Figure 27. An outline of the CHASE driver script

CHASE driver script validates the input data and utilizes all modules to perform one complete CHASE run in stages S0 to S6.

The third stage (S3) in the CHASE driver script deals with calling the run and parse module to execute the homology search methods and parse their reports,

one by one, to report the sequence ids and the E-values of the hits. In stage 4 (S4) it reads the target database to report sequences and their descriptions required for CHASE output, as explained in the database reader module section. The Stage 5 (S5) is the point where CHASE combines the homology search methods using the C-value formula that requires the method performance weights and the scaled E-values, as discussed above. The last stage (S6) of the CHASE driver script deals with reporting the CHASE results either in HTML, XML or in a simple text format.

Using modular CHASE has several advantages. Modular CHASE is general enough to easily accommodate a new homology search method, given its required configuration in XML, the code to run the method and to parse its report. It is now possible to use different IPs (e.g. different alignment schemes) for the same homology search method, one just need to update the configuration in the XML file.

4.2. Effect of Different Alignment Schemes

All homology search methods combined in CHASE, called the CHASE component methods, make use of alignments in one way or the other, except MAST. Using the modular version of CHASE it is quite easy to use different alignment schemes for the same homology search method. Previously our results were based only on the ClustalW alignment scheme and now we test the effect of two additional alignment schemes i.e. Dialign (REF) and Mafft (REF) on the performance of all alignment-based CHASE component methods.

We calculate composite weights, considering the performance of CHASE component methods on the odd half of the SCOP database, at k=50 false positives, as explained in section 3.2.1, using three alignment schemes namely ClustalW, Dialign, and Mafft, shown in Table 5.

Table 5. Composite weights for CHASE component method performances in the odd half of the SCOP database at k=50 false positives (FPs). See Table 1 in section 2.6.2 for evaluation scenarios.

Mafft_Component Weights SCOP ODD: 50 FPs

	DFOM	FHvOM	FHfOM	Total	Weights
HMMsearch	27.9762	80.5111	88.9302	197.4175	0.2203
Treesearch	26.9762	73.6667	81.2558	181.8987	0.2030
PSI-Blast	23.6190	73.6000	87.5349	184.7539	0.2062
PHI-Blast	23.3095	69.4222	82.1628	174.8945	0.1952
Mast	17.5714	60.9111	78.4884	156.9709	0.1752
				895.9356	1.0000

Dialign_Component Weights SCOP ODD: 50 FPs

	DFOM	FHvOM	FHfOM	Total	Weights
HMMsearch	30.7619	77.5556	90.4186	198.7361	0.2234
Treesearch	26.9762	73.6667	81.2558	181.8987	0.2045
PSI-Blast	21.8333	69.9111	85.2093	176.9537	0.1989
PHI-Blast	23.3095	69.4222	82.1628	174.8945	0.1966
Mast	17.5714	60.9111	78.4884	156.9709	0.1765
				889.4539	1.0000

ClustalW_Component Weights SCOP ODD: 50 FPs

	DFOM	FHvOM	FHfOM	Total	Weights
HMMsearch	27.1429	77.5778	89.2791	193.9997	0.2187
Treesearch	26.9762	73.6667	81.2558	181.8987	0.2050
PSI-Blast	21.2381	73.1333	85.0465	179.4179	0.2022
PHI-Blast	23.3095	69.4222	82.1628	174.8945	0.1971
Mast	17.5714	60.9111	78.4884	156.9709	0.1769
				887.1818	1.0000

Composite Weights

	Clustalw	Dialign	Mafft	Total	Cweights
HMMsearch	0.2187	0.2234	0.2203	0.6625	0.2208
Treesearch	0.2050	0.2045	0.2030	0.6126	0.2042
PSI-Blast	0.2022	0.1989	0.2062	0.6074	0.2025
PHI-Blast	0.1971	0.1966	0.1952	0.5890	0.1963
Mast	0.1769	0.1765	0.1752	0.5286	0.1762
					1.0000

We run CHASE and its component methods, for distant relationship scenario, using ClustalW, Dialign, and Mafft alignment schemes, as shown in Figures 28, 29, and 30, respectively, to see if any improvement in their performance can be achieved. Looking at the results, it can be seen that some of the alignment-based CHASE component methods achieved their best performance in terms of coverage of true positives using Dialign and others using Mafft, in comparison to using the ClustalW alignment, except PSI-Blast that did not improve much. The overall coverage of true positives can be increased even further, through CHASE, using the best alignment scheme suitable for a particular CHASE component method.

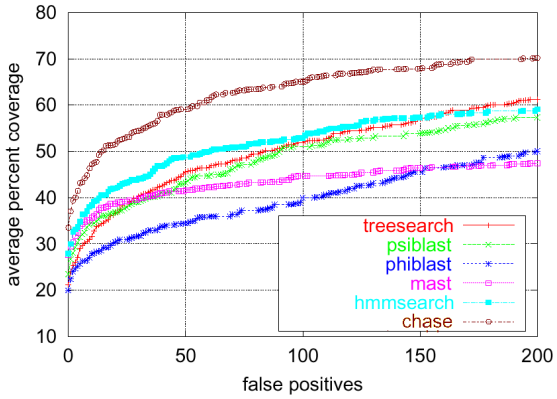

Figure 28

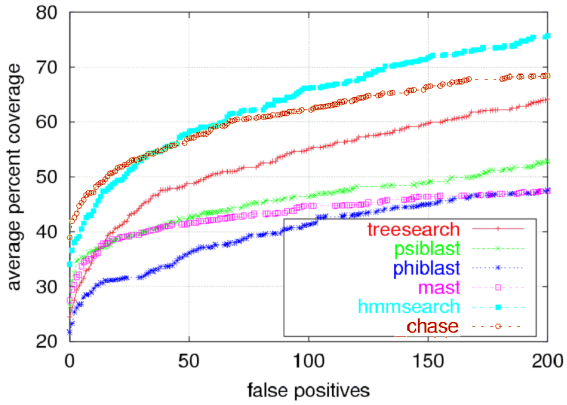

Figure 29

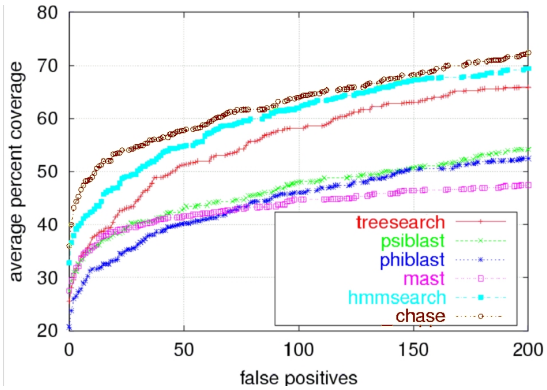

Figure 30

Figures 28, 29, and 30: Coverage versus False positive counts are shown for CHASE and its component methods, for Distant Relationships scenario, using **ClustalW, Dialign,** and **Mafft** Alignment, respectively.

Since the performance of PSI-Blast did not improve much by using different alignment schemes, we changed its input parameters by adding an additional iteration (i.e setting option –j 2) and CHASE component methods were run again using ClustalW, Dialign, and Mafft as shown in Figures 31, 32, and 33, respectively.

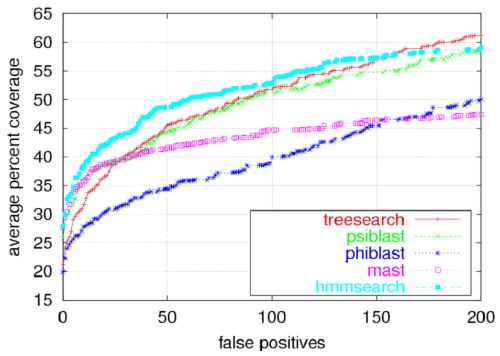

Figure 31

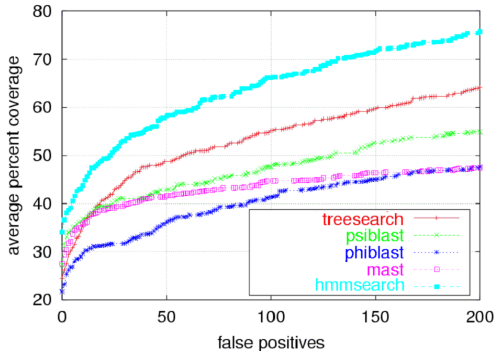

Figure 32

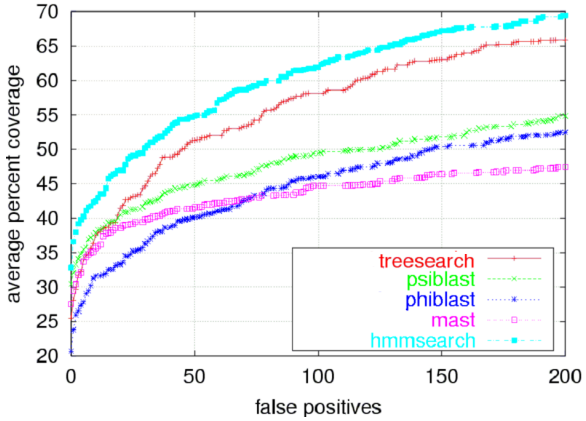

Figure 33

Figures 31, 32, and 33: Coverage versus False positive counts plot is shown in Distant Relationships scenario, using **ClustalW**, Dialign, and Mafft Alignment, respectively. An **additional iteration** (-j 2) was used to improve the performance of PSI-Blast.

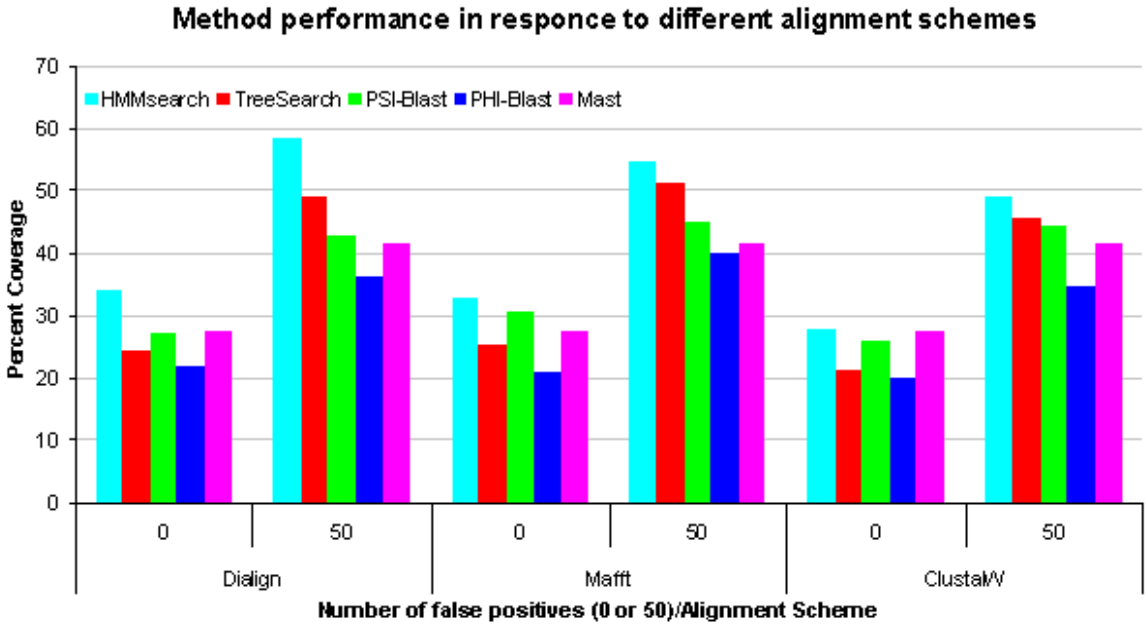

Figure 34: Coverage versus False positive counts are shown in the Distant Relationships scenario, permitting zero and 50 false positives, using different alignment schemes. An **additional iteration** (-j 2) was used to improve the performance of PSI-Blast. Searches were done in the odd half of the SCOP database.

To see which method performs at its best using which alignment scheme, coverages allowing zero or fifty false positives, for each alignment scheme used, were plotted as shown in Figure 34. Looking at the coverages obtained by CHASE component methods for the cost of 50 false positives, it is seen that HMMsearch achieves its best performance using Dialign alignment, while all other alignment based CHASE component methods show their best coverage of true positives using Mafft alignment. The average percent coverage of true positives at the cost of either zero or fifty false positives, however, represents the facts imprecisely and therefore we calculate the area under the curve for zero to fifty false positives, as described in the next section, for a more precise analysis.

4.2.1 Choosing the best Input Processors for Alignment-based CHASE Component Methods

CHASE and its component methods were run on the odd half of the SCOP database using either ClustalW, Dialign or the Mafft alignment scheme as shown in Figures 35, 36, and 37, respectively. For a more comprehensive analysis of the performance of each alignment-based CHASE component method in response to a particular alignment scheme used, the area under the curve, as shown in Figure 38, was measured. This was done by first summing up the average percent coverages at the cost of zero until fifty false positives, obtained by a particular CHASE component method in response to the particular alignment scheme used, and then dividing that sum by 51. This served as the basis to choose the best input processor (i.e. an alignment scheme in this case) for a particular alignment-based CHASE component method.

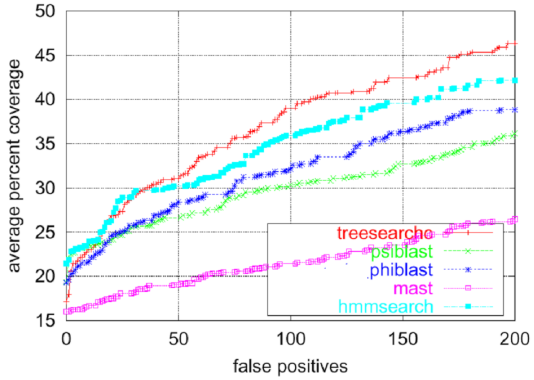

Figure 35

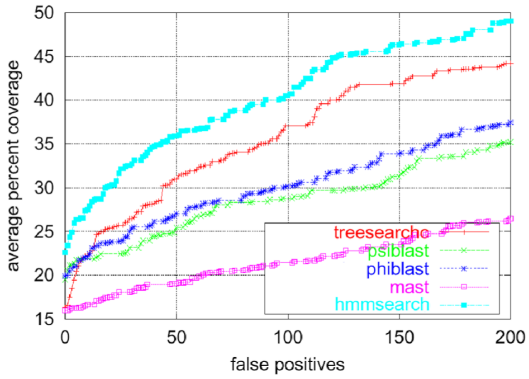

Figure 36

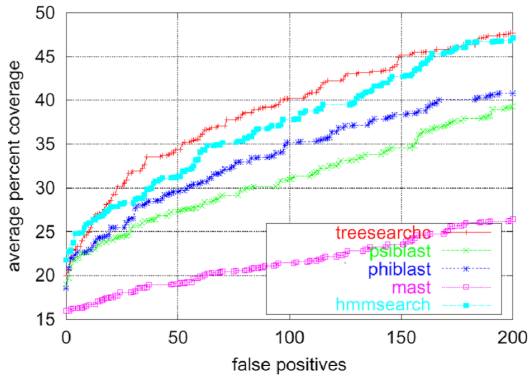

Figure 37

Figures 35, 36, and 37: Coverage versus False positive counts are shown in the Distant Relationships scenario on the **odd** half of the SCOP database using **ClustalW, Dialign, and Mafft** Alignments respectively. An **additional iteration** (-j 2) was used to improve the performance of PSI-Blast.

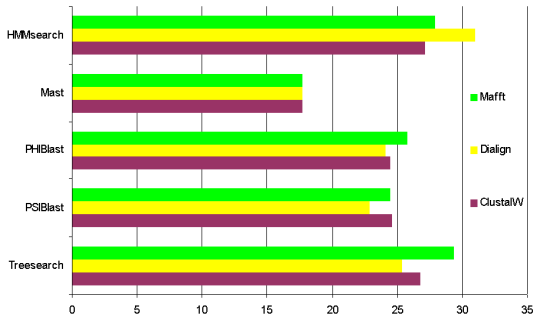

Figure 38

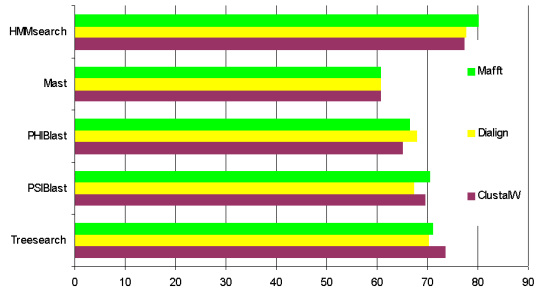

Figure 39

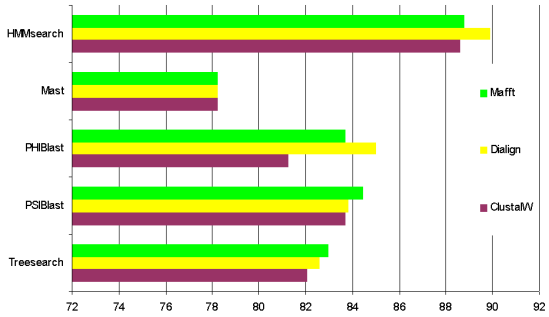

Figure 40

Figures 38, 39, and 40: The area under the curve is shown for the average percent coverages of true positives at the cost of zero to fifty false positives in distant, close, and very close relationship scenarios, respectively.

The area under the curve for the distant relationship scenario, as shown in Figure 38, designates Dialign alignment scheme as the best input processor for HMMsearch, Mafft for Treesearch, and PHI-Blast, and ClustalW for PSI-Blast. A similar analysis was done for close and very close relationship scenarios as shown in Figures 39 and 40, respectively. In the close relationship scenario, the area under the curve assigns Mafft alignment scheme as the best input processor for HMMsearch and PSI-Blast, Dialign for PHI-Blast and ClustalW for Treesearch. In the very close relationship scenario the Dialign alignment scheme is shown to be the best input processor for HMMsearch and PHI-Blast while Mafft for Treesearch and PSI-Blast.

To test the performance of CHASE for the distant relationship scenario, in the light of the results from the odd half of SCOP database, we applied the best input processors to all the alignment based CHASE component methods (i.e. Dialign alignment scheme for HMMsearch, Mafft for Treesearch and PHI-Blast, and the ClustalW alignment scheme for PSI-Blast). The results are shown in Figure 41.

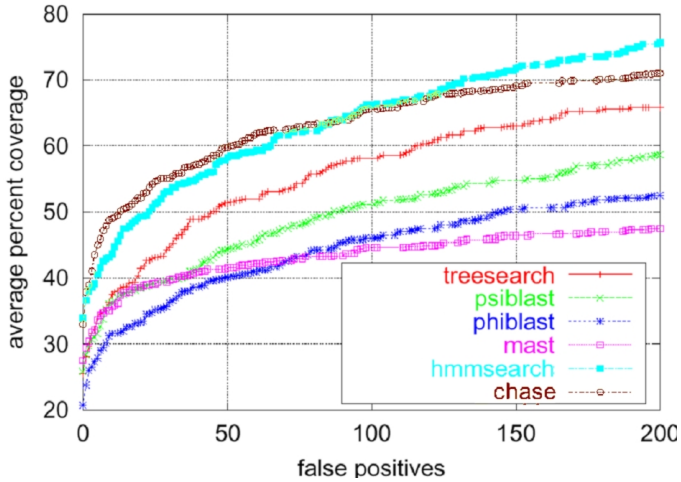

Figure 41: Coverage versus False positive counts is shown for CHASE and its component methods, on even half of SCOP database, for Distant Relationships scenario, using **best Input Processors**.

The performance of CHASE, based on using best input processors for its alignment based component methods as shown in Figure 41, depicts that it achieves a better average percent coverage of true positives until the cost of 60 false positives however the performance of HMMsearch is competitive from 60 to 100 false positives. Furthermore the performance of HMMsearch becomes better than CHASE after 100 false positives. The reason for inferior CHASE performances seems to be the significant difference among the overall performance of CHASE component methods. One may break down the range of overall performances achieved by CHASE component methods into three categories: similar performances, different performances and very different performances. In the first category methods perform similarly. In such a case the performance of CHASE may be similar to that of its component methods,

because there are not so many cases where evidence can be combined to yield a better estimate, by outvoting the outliers. In the second category methods may perform differently to some extent, as shown in Figure 28, and there are many cases where a strong showing for some method(s), not counterbalanced by very poor showings for others, seems to be a good membership indication that is (independently of which single method is involved) picked up by our consensus approach. In the last category, component methods perform significantly different, as shown in figures 29 and 41 and there may be too many cases dominated by very poor showings.

4.3. Improving C-value by Recalculating E-values

To further improve the performance of CHASE, we decided not to use E-values provided by individual methods. Instead we wish to recalculate E-values in a standard way. More precisely, we calculate P-values for CHASE component methods in a standardized fashion, combine them and derive one combined E-value or C-value from the combined P-values, as discussed below. This concludes the development of the CHASE2. The P-value is the probability of an alignment occurring by chance with a score equal or better than the observed similarity score while searching a database of randomly generated sequences of the same size as the actual database. P-values and E values are different ways of representing the significance of an alignment or a database hit. The maximum P-value is 1.0 while the maximum E-value can be equal to the total number of sequences in a database.

As discussed in chapter 4, an E-value, based on the distribution of sequence similarity scores (for example the alignment scores of unrelated sequences) is a frequently used statistical estimate to represent the significance of database search results. Accurate measures of the statistical significance of alignment scores greatly enhance the usefulness of similarity searches. Knowing the distribution of the alignment scores of unrelated sequences allows the estimation

of the expected number of false positives at a given threshold. The E-value also depends on the size of the searched database that is the number of its sequences or the total number of residues that it has.

It has been shown that the gapped alignment scores (that allow indels) approximately follow an extreme value distribution that has certain computable parameters i.e *K* and *lambda*, as long as the gap penalties are severe enough (Altschul et al., 1997). The parameters K and lambda can be thought of simply as natural scales for the search space size and the scoring system (scoring matrix and the gap penalties) respectively.

There is more than one way to calculate an E-value, and different methods can produce values that differ by several orders of magnitude (Pagni et al., 2001). Currently, most sequence similarity algorithms estimate alignment score significance in one of two basic ways (Bailey et al., 2002). One type is a lookup table approach, such as the one implemented in Blast, which precalculates the parameters of the distribution for a variety of scoring table/gap penalty combinations (Altschul and Gish, 1996, Altschul et al., 1997). The other type is sometimes referred to as the 'empirical' approach because it estimates the parameters of the distribution function directly from the scores observed in the database search, such as the one implemented in FASTA (Pearson, 1990) or HMMER (Eddy, 1998).

4.4.2. "fitevd"

CHASE component methods calculate E-values each in a different way and instead of combining their original E-values in CHASE, it would be more meaningful if we recalculate E-values, where possible, using a uniform approach. Recently (Bailey et al., 2002) a novel algorithm, called "fitevd", based on maximum likelihood has been presented for estimating the distribution of alignment scores using the scores of unrelated sequences from a database

search. It has been shown (Bailey et al., 2002) that this algorithm is more accurate and better than the existing lookup table approach. This algorithm also has an advantage, especially with databases containing relatively few sequences, in reducing the limitation concerning the number of unrelated sequences required by empirical methods. Furthermore, a specialty of this algorithm is a technique for stratifying the target sequences into length groups and estimating score distributions for each of the length groups.

Given a list of hits, each with an alignment score, the length of the query and of the hit, this algorithm, known as fitevd, reports the corresponding P-values. These P-values are based on the parameters that fitevd estimates from the score distribution of each of the length groups of target sequences. We empirically set the query length divided by 10 as our standard value for the length group size, however if the query length is very small the value for length group size becomes so small that the fitevd program reports arbitrary P-values. To circumvent this problem we estimate a weight, *W*, using a sigmoid activation function (Fuellen et al., 2001), given the query length, *Q*, by using the following formula:

$$W = 1/(1+ \exp(-(Q-100)/100)).$$

Such a weight, W, is then used to smoothen the value for the length group size, *S,* as follows:

$$S = ((W*(Q/10))+((1-W)*1000)).$$

HMMsearch, PSI-Blast, and PHI-blast report alignment scores in their search reports so it is fairly easy to use fitevd to calculate P-values for their hits. However, it is not possible to extract the alignment score from the results of Treesearch or Mast, so the fitevd approach is not applicable here. To get the P-values for Treesearch, and Mast hits, we divide their E-values by the size of the database. In turn, Treesearch calculates the E-values by multiplying the augmentation score with the size of the database and Mast calculates the E-values by multiplying the product of P-values, from the non-overlapping motifs

found in a hit sequence, by the size of the database (MEME-Mast “README”). To place the P-values of Treesearch and Mast on a common numerical scale, these are rescaled in reference to the P-values of HMMsearch using ordinary least square regression.

In the context of the fitevd algorithm, the P-value of a score x for a target sequence of length t, is defined as the probability of an unrelated sequence of length t having the observed score x or greater. The E-value is then defined as the P-value times the number of target sequences in the database search. So, in CHASE, if we multiply the P-value with the size (total number of sequences) of the searched database, it represents an E-value. Before we multiply, we use the P-values as they are in our C-value formula (described in section 6.2.3.). For a particular hit sequence the P-values of all CHASE-based component methods are combined using the C-value formula. This combined P-value is later multiplied with the size of the database to get a better combined E-value or C-value.

4.5. CHASE2 Results and Discussion

We run CHASE2, with fitevd-based E-values where possible, on the even half of SCOP database along with CHASE1 and all CHASE component methods using the best input processors in distant, close, and very close relationship scenarios. For a fair comparison the component methods of CHASE2 and CHASE1 are using the same alignment schemes. As shown in Figure 42 for distant relationship scenario, CHASE2 easily outperforms CHASE1 and all of its component methods by a reasonable margin and it is the best performance for a consensus homology search method, achieved to date. Particularly, accepting just 10 false positives, 55% coverage of true positives was obtained by CHASE2 and 49% by CHASE. Individual methods obtained coverage of true positives between 31 to 44% for same number of false positives.

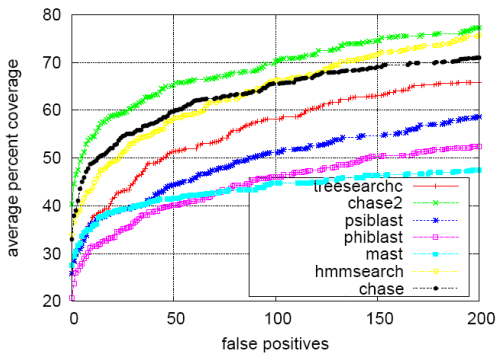

Figure 42

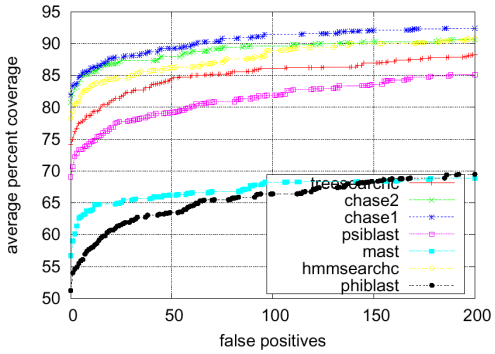

Figure 43

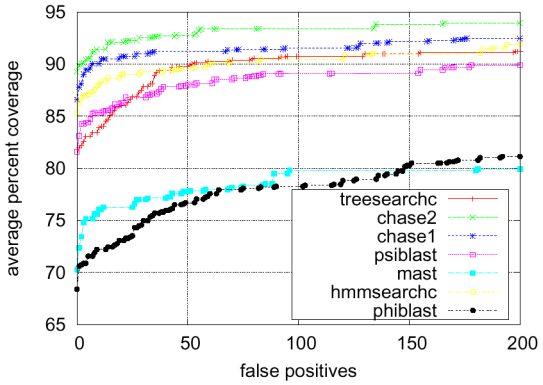

Figure 44

Figures 42, 43, and 44: Coverage versus False positive counts plot is shown for CHASE2, CHASE1, and all CHASE component methods using the best input processors in distant, close, and very close relationship scenarios, respectively

---

4.5.1. Evaluation of CHASE & CHASE2 on Three Different Databases

Previously, CHASE and its component methods were evaluated only on one "odd" and one "even" database, derived from SCOP. To show that the performance of CHASE is not just a chance product of using these particular two databases, we now evaluate CHASE and its component methods on three different "odd" and "even" databases. To derive these databases we split up the SCOP database randomly, into three odd and three even databases by using a function available with the PHASE4 package. We call these databases A-odd and A-even, B-odd and B-even, C-odd and C-even.

To evaluate the performance of CHASE1, CHASE2, and their component methods on these odd and even databases in distant relationship scenario, we first run CHASE component methods on the “odd” halves of the SCOP database to derive their performance weights and best IPs as described previously. Performance of CHASE component methods is shown in the form of average coverage of true positives versus false positive count plots in A-odd, B-odd and C-odd of SCOP database in Figures 45, 46 and 47, respectively.

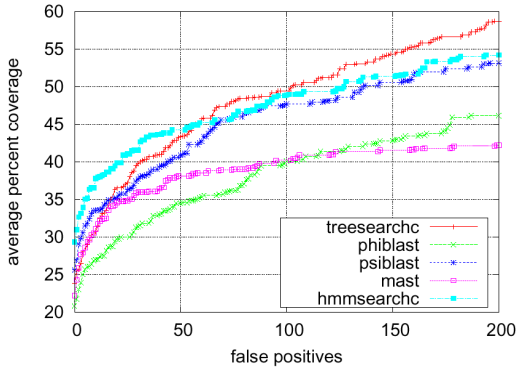

Figure 45

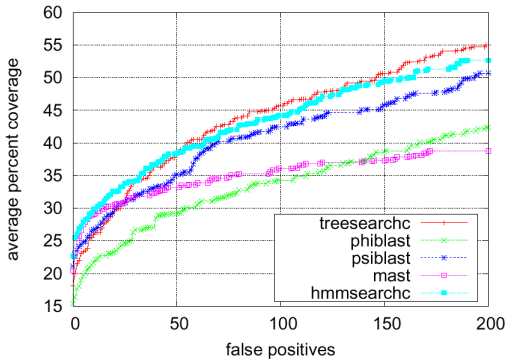

Figure 46

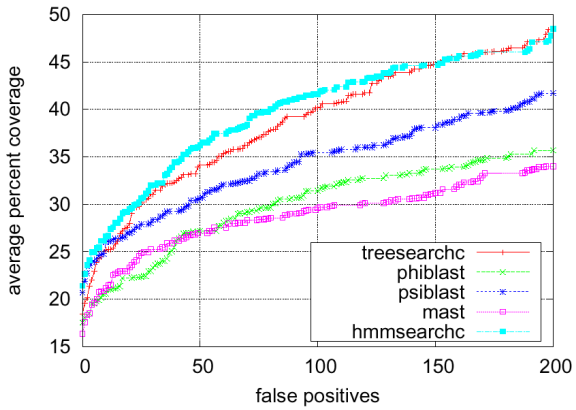

Figure 47

Figure 45, 46, and 47: Performance of CHASE component methods in distant relationship scenario is shown on **A-odd**, **B-odd** and **C-odd** half of SCOP database, respectively

---

Once we estimate the weights for CHASE component methods on each of A-odd, B-odd and C-odd half of SCOP database, we run CHASE1, CHASE2, and their

component methods on each A-even, B-even, C-even half of SCOP database. Performance of all methods on these three even halves of SCOP database is shown in Figures 48, 49, and 50.

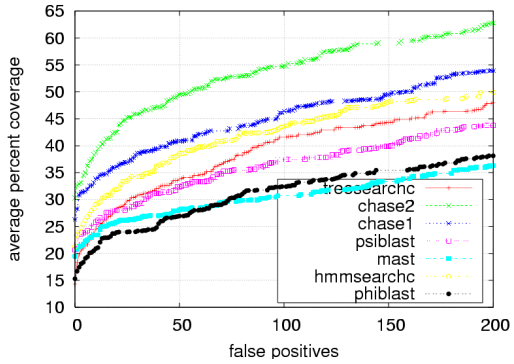

Figure 48

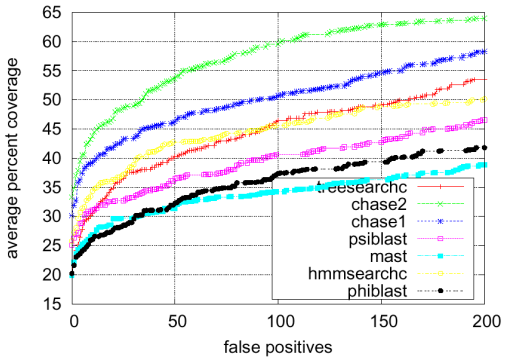

Figure 49

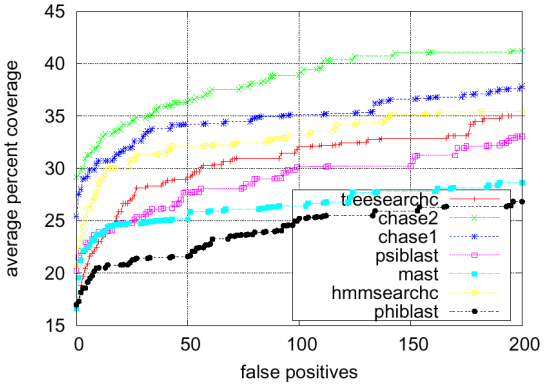

Figure 50

Figures 48, 49, and 50: Performance of CHASE1, CHASE2, and CHASE component methods in distant relationship scenario is shown on **A-even**, **B-even** and **C-even** half of SCOP database.

These results show that the performance of CHASE1 and CHASE2, in terms of average percent coverage of true positives in distant relationship scenario was not dependent on the specific odd/even split of SCOP that we used up to now. It is still better than any of their component methods and further the performance of CHASE2 is far better than CHASE1. This evaluation supports the claim that a

good combination of methods performs better than any of their component methods.

4.5.2. Run-Time Experiment

Since the run-time (proportional to input size) is not known for all component methods, and since it is not known for many of the steps employed by the input processors, some empirical data on run-time is given in this section. To compare the speed of CHASE and its five component methods, namely HMMsearch, Treesearch, PSI-Blast, PHI-Blast, and Mast, we timed these methods to search members of protein families, as shown in Table 6, in the even half of the SCOP database (containing 2734 sequences). Zhang et al., 1998, did a similar timing experiment for PHI-Blast. Here, we choose a representative set of sequences with different sequence lengths as input to our methods, from the ten protein families as shown in the Table 6. Run-time for each method was recorded in seconds, using the *time* function in Perl.

Table 6. Dataset for run-time experiment: Details of protein family sequences, sorted on the basis of average query length.

Query No.	Query Name	No. of Sequences in query	Average Query Length	Maximum Query Length
1	S100	27	44	44
2	1.36.1.2	6	71	87
3	1.41.1.2	6	92	100
4	1.23.1.1	7	103	125
5	1.73.1.1	6	126	151
6	1.128.1.1	8	127	292
7	1.27.1.1	10	170	189
8	ABC Transporters	4	300	345
9	3.3.1.no5	17	300	392
10	Serpins	42	415	423

This timing experiment was run on an UltraSparc computing machine with 96 Gbytes of RAM, running the operating system Solaris, version Generic_117171-07, release 5.9, which is an implementation of UNIX. All of the methods used the same set of sequences as an input and the method-specific inputs, e.g. the HMMs required by HMMsearch, were prepared using the input processors. Thus, the run-time of individual methods includes the time spent on preparing their specific inputs. Since a load unbalance at the computing machine could increase the execution time of a user application (Liu et.al., 2003), resulting in different run-times, we replicated our experiment three times (see Appendix A) to get an average run-time for each method.

Table 7. Running time experiment; Results of the timed searches conducted using HMMsearch, Treesearch, PSI-Blast, PHI-Blast, Mast and CHASE in the *even half* of SCOP database, using members of 10 protein families (shown in table5). Results are sorted on the basis of average query length.

Query No.	Query Name	HMMseach	Treeseach	PSIBlast	PHIBlast	Mast	Sum	Sum W/O HMMsearch	CHASE
1	S100	8	83	3	8	20	122	114	106
2	1.36.1.2	14	39	2	4	21	80	66	61
3	1.41.1.2	15	49	6	3	17	90	75	69
4	1.23.1.1	18	52	3	3	19	95	77	79
5	1.73.1.1	24	63	4	4	41	136	112	109
6	1.128.1.1	34	104	4	5	33	180	146	142
7	1.27.1.1	26	79	4	6	58	173	147	153
8	ABCs	43	106	6	8	21	184	141	145
9	3.3.1.no5	49	199	9	27	213	497	448	454
10	Serpins	50	373	11	68	1026	1528	1478	1428

As shown in Table 7, Treesearch consumes more time, on average, in conducting its database searches than the time consumed by any of the other CHASE component methods. Usually, Mast is faster than Treesearch but it takes much longer if the number of input sequences and their length is larger than average. This is due to MEME (part of the Mast package) that extracts motifs, from the input set of sequences, required by Mast to conduct the database search. Time taken by HMMsearch, on average, is less than that of Treesearch and Mast. The Blast family of methods namely PSI-Blast and PHI-Blast turns out to be very fast, as expected. The time taken in conducting database searches by PHI-Blast is similar to that of PSI-Blast but PHI-Blast can take longer if the number of input sequences and their length is larger than average. This happens because PRATT may then take long to construct the PROSITE-like pattern that is required to run PHI-Blast. Analysis of the results from Table 7 shows that, on average, CHASE run-time is less than the total run-time of all of its component methods. Time taken by CHASE to complete a particular search includes steps like input processing, running of its individual homology search methods, their output processing, C-value calculation and the CHASE output report preparation. One advantage for CHASE is that during its input processing stage, some of the inputs required by its component methods, e.g. the HMM required by HMMsearch and Treesearch, are generated only once and this saves some run-time. In fact, since HMM generation dominates HMMsearch, and since CHASE takes about as long as all the other methods except HMMsearch taken together (see “sum without HMMsearch” in table 7), it seems that the time that the CHASE script itself needs is about equal to the other savings, in particular the savings due to generating the multiple alignment only once.

4.5. Conclusions, and CHASE Future Work

Regarding the future development of CHASE, the following issues are noteworthy.

- Inclusion of more homology search methods in CHASE, such as jumping alignments (Spang et al., 2002) and THMM (Qian et al., 2003).
- Improving memory consumption of modular CHASE, by improving the handling of the database by the database reader module.
- Implementation of a parallel version of CHASE using Biopipe (Hoon et al., 2003).
- It is future work to use CHASE for searching protein databases to find missing members to improve databases such as Pfam.
- One should use CHASE to search in translated genomic databases to find novel proteins.

GenCHASE was developed to handle the last issue. It is the topic of the next and last chapter. Further conclusions can be found in the last section of that chapter.

5. GenCHASE (Genomic Comparative-Homology Agreement SEarch)

5.1. Overview

Genomes of many organisms have been sequenced over the last few years resulting in an enormous amount of sequence data. About 206 genomes have been completely sequenced (and published), while the sequencing of 506 prokaryotic and 418 eukaryotic genomes is underway (see http://www.genomesonline.org/) as of July 09, 2004. Unfortunately, the annotation of such huge datasets is not keeping pace, considering that annotating a genomic sequence is not an easy task (Claverie et al., 1997). Further, if no good annotation is available, it is difficult to find out whether any members of a protein family of interest exist in a newly sequenced genome. Many methods have been developed (for a review see: Mathe et al., 2002) to find genes along a genomic sequence using either intrinsic (ab initio) or extrinsic (homology-based) approaches (Borodovsky et al., 1994, Rouzé et al., 1999). On one hand, ab initio gene prediction methods do not give precise predictions of all the genes in a given sequence without false negative or false positive errors. On the other hand only about half of the genes can be found using the extrinsic approach alone (Mathe et al., 2002).

Following the success of CHASE (Alam et al., 2004), using the same basic idea, we combine extrinsic (homology-based gene finding) and intrinsic (ab initio gene structure prediction) approaches in a tool called GenCHASE (Genomic Comparative Homology Agreement Search). An outline of GenCHASE is shown in Figure 51. GenCHASE finds a maximum number of possible homologues in a single genomic sequence, for example a chromosome, given a set of protein sequences.

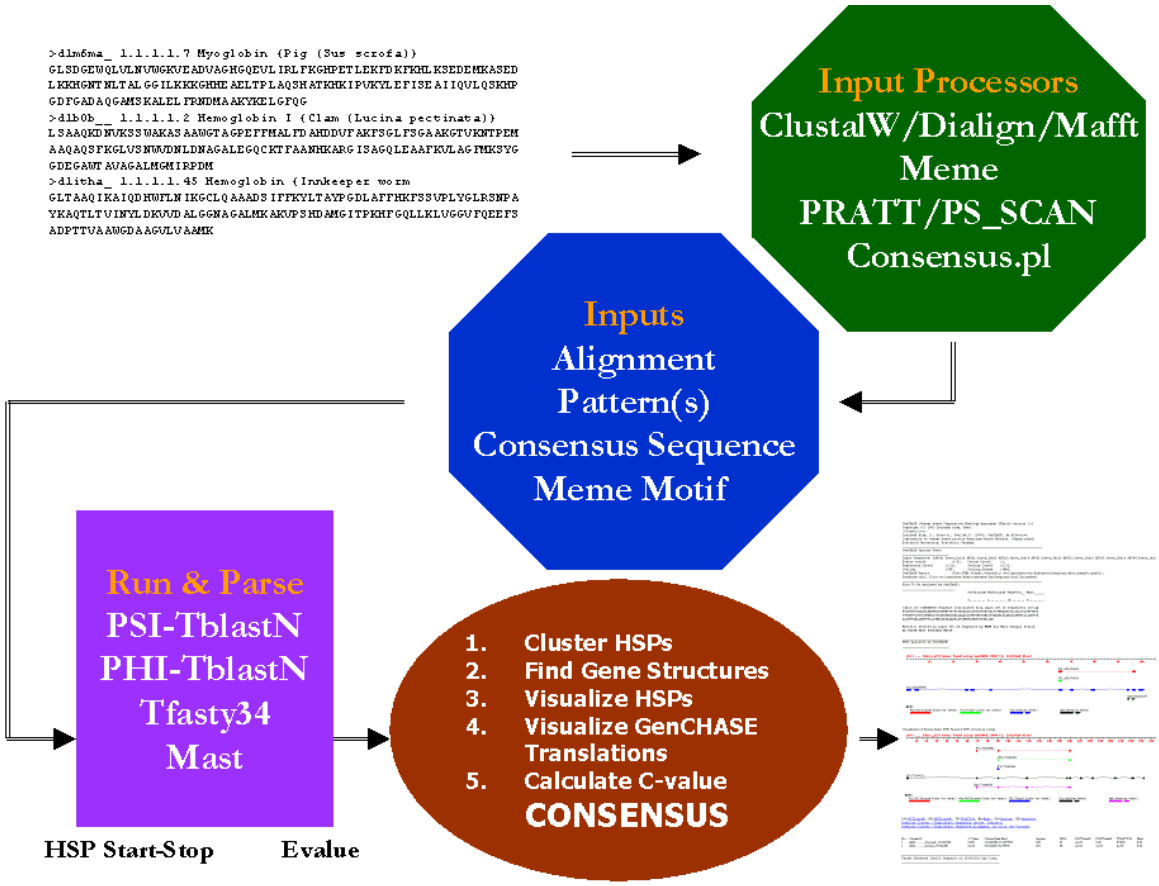

Figure 51. An outline of GenCHASE

Given a set of protein sequences, specific inputs are prepared for homology search methods. GenCHASE then combines the results from different methods in 5 steps, given the co-ordinates and E-values for the High Scoring segment pairs (HSPs; as shown in Figure 52).

To run GenCHASE on all the genomic sequences of a particular genome such as the human genome, one by one, and to analyse its results we have developed another program on top of GenCHASE called GenCHASE-Analyser. Here we present our approach in detail as well as results for the search of ATP Binding Cassette (ABC), S100, and Cadherin proteins (described below) in the human genome, as an evaluation of GenCHASE.

5.2. GenCHASE Component Methods

In GenCHASE we combine homology-based gene finding methods such as PHI/PSI-TblastN (Altschul et al., 1997), TfastY (Pearson, 1990), and Mast (Bailey & Gribskov, 1998), homology-based gene structure prediction methods like Genewise (Birney et al., 2004) and ab initio gene structure prediction methods like Genscan (Burge et al., 1997). We call them GenCHASE component methods. The ultimate goal of these methods is to find out or predict genes in a genome, given a single or a set of protein or DNA sequences. However, they differ in the technique they use to accomplish this task. All similarity search methods that we combine in GenCHASE use a collection of protein sequences as search input, translate the target genomic sequence into six reading frames on the fly, take care of frameshifts (except Mast) and report a confidence estimate such as an E-value, for each of the hits they find. Genewise and Genscan are gene structure prediction methods that we apply to the genomic regions where we find potential genes using homology-based gene finding methods. Following are the details of these methods:

- **PSI-TblastN** is a similarity-based gene finding method that searches a protein query sequence against a nucleotide sequence database (target database) using a position specific scoring matrix created by PSI-BLAST. A single genomic or chromosomal sequence can also be used as the target database. To save the results as a position specific scoring matrix, in the first step, we run a normal PSI-Blast search against a protein database such as SCOP (Andreeva et al., 2004) or SWISSPROT (Boeckmann et al., 2003), given the input set of protein sequences as a jumpstart alignment and a query sequence. The query sequence can be any sequence from the jumpstart alignment. In the second step we use the position specific scoring matrix we saved and the query sequence to run a PSI-TblastN search in a nucleotide/genomic sequence database.

PSI-TblastN translates the target genomic sequence into six reading frames on the fly, so that the given query sequence can be aligned to the translated genomic region.

- **PHI-TblastN** is similar to PSI-TblastN, except in the first step, where a consensus sequence and a motif (in the form of a regular expression) are used instead of the jumpstart alignment.

- **TfastY** compares a protein sequence to a DNA sequence database, translating the DNA sequence in six reading frames and aligning the protein sequence to each sequence of the database, allowing gaps and frameshifts. We use the consensus sequence of the input set of sequences to start the TfastY search.

- **Mast** searches biological sequence databases for sequences that contain one or more of a group of known motifs that we provide in a specific format derived using a tool called MEME (Multiple EM for Motif Elicitation; available in MEME-MAST package). Mast also translates the genomic database into six reading frames on the fly.

- **Genewise** is a homology-based gene structure prediction method. In Genewise, we use a consensus sequence that is calculated from the input set of protein sequences, to predict the possible gene structure of the regions in a genomic sequence that show sufficient similarity. Genewise is a very slow method and it takes very long if we apply it on the complete genomic sequences. So we restrict ourselves to applying Genewise on genomic region(s) that we find, using homology-based gene finding methods, as described in section 5.4.3. We use the program “genewise” if we need to run Genewise on one genomic region, however if there are more then one regions, we apply the program “genewisedb”.

- **Genscan** is an ab initio gene structure prediction method. It uses an organism specific parameter file (available with the Genscan package) to predict the gene structure of all possible genes that a given genomic sequence may have. As we do not want Genscan to predict the gene structure of regions of a genomic sequence that are not homologous to the query, we apply it only on regions of a genome that we find similar using homology-based gene finding methods. Genscan gene structures in the flanking region are deleted because they are not supported by any of the homology-based gene-finding methods.

5.3. Input Processing

All of the homology-based gene-finding methods (i.e PHI/PSI-TblastN, TfastY and Mast) that we combine return confidence estimates, usually E-values, for their results. To perform their task, they require a single or a set of protein sequences as query and a genomic/DNA sequence(s) as a target database. These methods translate the target genomic sequence into six reading frames and all except Mast are capable of detecting frame-shift errors. The exact query format requirements, however, vary from method to method. As for CHASE, we developed scripts called *input processors (IPs*) that take a collection of sequences and process these as follows to obtain the specific type of input for each of these similarity-based gene-finding methods.

- **PSI-TBlastN IP:** In the first step, we use Mafft (Katoh et al., 2002) to align the input sequences, and we format the alignment such that it can be used, together with a single query sequence, to “jumpstart” a "single run" PSI-Blast search in a protein database such as SCOP or SWISSPROT. (The multiple alignment that is used to jumpstart PSI-Blast must contain the query sequence. PSI-Blast further requires that the jumpstart alignment does not contain some of the headers and trailers that are usually present in alignments). In the second step, that is, the search in

the target genomic sequences, PSI-TblastN requires that the single query sequence must be the same as the one used in the first step for creating a position specific scoring matrix. As described in the README, by default, the filtering (or masking of low complexity regions) is off (-F F) in the protein-protein blast while it is on (-F T) in the protein-DNA blast. To ensure consistent usage of the protein-protein and protein-DNA blast combination, the -F option should be explicitly set in one or the other run. Since we use already masked sequences, we turn filtering off in our second step of the PSI-TblastN search.

- **PHI-TBlastN IP:** We use *PRATT* to generate a Prosite-like pattern from the given un-aligned sequences. A ClustalW (Higgins, 1994) alignment is used to generate a consensus sequence by relative majority rule for starting a PHI-Blast search with the Prosite-like pattern, followed by a "single run" of PSI-Blast, to save the position specific scoring matrix that is required in the second step of the PSI-TblastN search that follows.

- **TfastY IP:** We use the Mafft alignment to generate a consensus sequence, by relative majority rule, for starting the TfastY search.

- **Mast IP:** We use *MEME* (Multiple EM for Motif Elicitation) (Bailey and Elkan, 1994), given un-aligned sequences, to generate motifs that are used to run Mast.

5.4. GenCHASE; A Scheme for Combining Homology and Gene-Finding Methods

We describe a scheme to combine several extrinsic gene-finding methods, namely PSI-TblastN, PHI-TblastN, TfastY, and Mast, based on the confidence estimates such as E-values that they report for every hit or high-scoring segment pair (HSP; local alignments with no gaps that achieve one of the top alignment

scores in a given search (Blast Glossary)). As shown in Figure 51, in our scheme for combining different homology-based gene finding methods, we run them one after the other. Since they use various kinds of input information we provide this information automatically, employing input processors as described above. Once these homology-based gene searches are completed, to combine these methods GenCHASE carries out the steps explained in sections (5.4.1-5.4.6) below.

5.4.1. Extract HSPs

Results of each method are parsed to extract specific information, such as the co-ordinates or the start and stop positions of the HSPs (high-scoring segment pairs) and the corresponding E-values. We assign an identifier to each HSP, which consists of the name of the chromosome, the frame information (i.e. forward or reverse) and its start position. For example, given a set of ATP Binding Cassette (ABC) transporter protein sequences PHI-TblastN reports a particular HSP in the reverse frame of human chromosome X starting at position 73149064 until 73148969 with an E-value of 2e-07. We assign the identifier chrX_reverse_73149064 to such an HSP and extract the information as shown in Table 8.

Table 8: HSP Co-ordinates, an example: This table shows the information such as an identifier (that we assign), name of the method, start and end positions and the corresponding E-value of an HSP which we extract from the output of homology search methods.

HSP Identifier	Method	Start – End	E-value
chrX_reverse_73149064	PHI-TBlastN	73149064 - 73148969	2e-07

5.4.2. Cluster HSPs

Based on overlap criteria, described in 5.4.3, we then assemble the HSPs (that we get from different methods) into clusters using the Bit::Vector module (Beyer,

2004). Identifiers are assigned to such clusters. E-values of many HSPs, reported by a particular method, that belong to the same cluster are multiplied, since the E-values below 0.01 are similar to P-values (Koonin and Galperin, 2003). For example the HSPs from PSI-TblastN (PS), PHI-TblastN (PH), and TfastY (TF) are considered overlapping and placed into a cluster as shown in Figure 52 and Table 9.

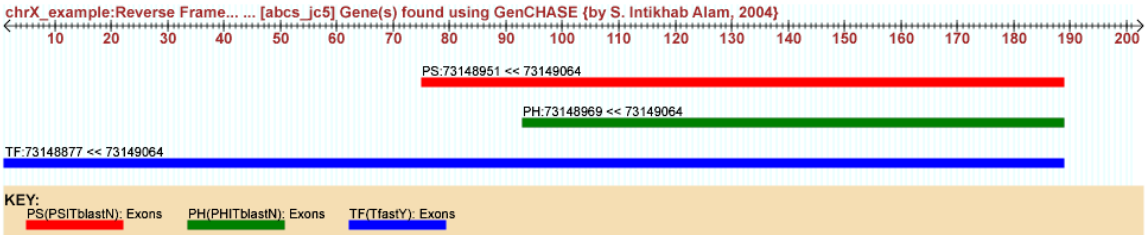

Figure 52. An example of HSP cluster

HSPs are placed into a single cluster if they overlap e.g. HSPs from PSI-TblastN, PHI-TblastN, and TfastY are placed into one cluster (e.g. a cluster for reverse frame HSPs is shown, right to left).

Table 9: An example of HSP Co-ordinates placed into a cluster

Cluster Identifier	HSP Identifier	Method	Start-End	E-value
ChrX_reverse_73149064	ChrX_reverse_73149064	PS	73149064 - 73148951	3e-08
	ChrX_reverse_73149064	PH	73149064 - 73148969	2e-07
	ChrX_reverse_73149064	TF	73149064 - 73148877	2.2e-06

5.4.3. Formulate Temporary HSP Super-Cluster(s) and Predict Gene Structure

Next we cluster the HSPs. We do not consider single-HSP “clusters”; empirically, we noted too many false positives if we support single-HSP “clusters” that do not overlap with HSPs from another method. In the human genome only about 5.24% of introns are more than 200 kilo base pairs (kbps) long (Sakharkar et al., 2004). Thus we assemble all HSP clusters into a temporary super-cluster if these are located at a distance of less than 200 kbps from each other. The user can

provide this threshold called the maximum intron length. These temporary super-clusters are combined into a gene if a gene structure prediction method identifies the gap(s) between such clusters as introns.

For example, extending the above-mentioned HSP cluster, chrX_reverse_73149064, from human chromosome X, we assemble a temporary "super-cluster", as shown in Figure 53 and Table 10 (HSPs are sorted by method).

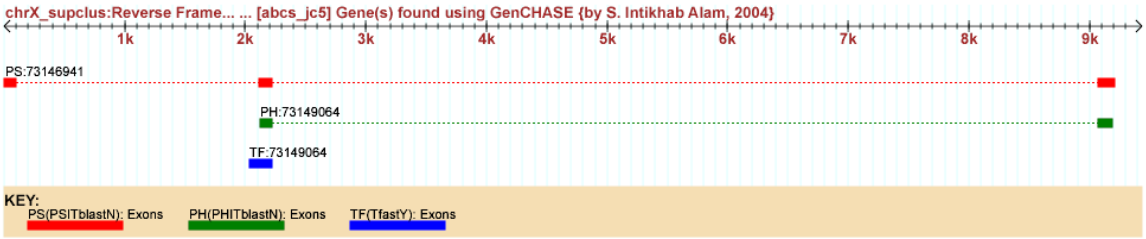

Figure 53. An example of temporary HSP super cluster

A temporary HSP super cluster is formed if two or more HSPs (or HSP clusters) are close enough, considering a distance of some, user-defined, length in base pairs. Gene structure prediction methods are then applied to the genomic region (including some flanking region) represented by such a temporary super-cluster.

Table 10: Co-ordinates for an example HSP super cluster

Cluster Identifier	HSP Identifier	Method	Start	End	E-value
ChrX_reverse_73156047	ChrX_reverse_73146941	PS	73146941	73146843	0.007
	ChrX_reverse_73149064	PS	73149064	73148951	3e-08
	ChrX_reverse_73156047	PS	73156047	73155916	5e-05
	ChrX_reverse_73149064	PH	73149064	73148969	2e-07
	ChrX_reverse_73156032	PH	73156032	73155916	0.001
	ChrX_reverse_73149064	TF	73149064	73148877	2.2e-06

GenCHASE further extends the super-cluster to both sides of the genomic region, if possible, employing a user-defined flanking region length (default 10 kbps). GenCHASE then applies gene structure prediction methods such as Genewise and Genscan to predict the structure of possible genes in such a super cluster. Genewise provides the gene structure in the genomic region where

certain similarity, in relation to the query, is found. However, Genscan is not a similarity-based, but an ab initio method, and it predicts the gene structure in the given genomic region, as complete as possible. Once the possible gene structure of such regions has been established, all the HSPs and predicted introns and exons are assembled to represent a particular gene. For example Genewise (GW) and Genscan (GS) predicted introns and exons and HSPs reported by homology-based gene finding methods, such as PSI-TblastN (PS), PHI-TblastN (PH), and TfastY (TF), assemble a gene with the co-ordinates shown in Figure 54 and Table 11. Please note the PSI-TblastN, PHI-TblastN or TfastY do not return complete exons but may return partial exons; Intron-exon boundaries are usually predicted by gene structure prediction methods.

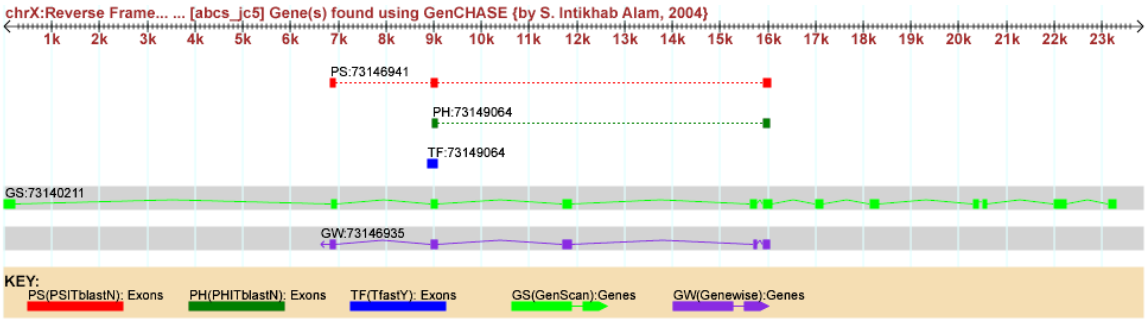

Figure 54. Visualization of a GenCHASE assembled gene

HSPs and introns/exons found for a particular ABC gene found in human chromosome X are shown. HSPs found using homology-based gene finding methods have a white background while the introns and exons found using gene prediction methods have a grey background.

Table 11: Co-ordinates of a gene assembled by GenCHASE

Cluster ID	HSP ID	Method	Start	Stop	E-value
chrX_reverse_73163280	chrX_reverse_73146941	PS	73146941	73146843	0.007
	chrX_reverse_73149064	PS	73149064	73148951	3e-08
	chrX_reverse_73156047	PS	73156047	73155916	5e-05
	chrX_reverse_73149064	PH	73149064	73148969	2e-07
	chrX_reverse_73156032	PH	73156032	73155916	0.001
	chrX_reverse_73149064	TF	73149064	73148877	2.2e-06

	chrX_reverse_73140211	GSex	73140211	73139996	-
	chrX_reverse_73146848	GSint	73146848	73140212	-
	chrX_reverse_73146956	GSex	73146956	73146849	-
	chrX_reverse_73148953	GSint	73148953	73146957	-
	chrX_reverse_73149057	GSex	73149057	73148954	-
	chrX_reverse_73151695	GSint	73151695	73149058	-
	chrX_reverse_73151867	GSex	73151867	73151696	-
	chrX_reverse_73155632	GSint	73155632	73151868	-
	chrX_reverse_73155762	GSex	73155762	73155633	-
	chrX_reverse_73155916	GSint	73155916	73155763	-
	chrX_reverse_73156080	GSex	73156080	73155917	-
	chrX_reverse_73156990	GSint	73156990	73156081	-
	chrX_reverse_73157148	GSex	73157148	73156991	-
	chrX_reverse_73158134	GSint	73158134	73157149	-
	chrX_reverse_73158309	GSex	73158309	73158135	-
	chrX_reverse_73160314	GSint	73160314	73158310	-
	chrX_reverse_73160402	GSex	73160402	73160315	-
	chrX_reverse_73160493	GSint	73160493	73160403	-
	chrX_reverse_73160582	GSex	73160582	73160494	-
	chrX_reverse_73161987	GSint	73161987	73160583	-
	chrX_reverse_73162256	GSex	73162256	73161988	-
	chrX_reverse_73163147	GSint	73163147	73162257	-
	chrX_reverse_73163280	GSex	73163280	73163148	-
	chrX_reverse_73146935	GWex	73146935	73146843	-
	chrX_reverse_73148954	GWint	73148954	73146935	-
	chrX_reverse_73149057	GWex	73149057	73148954	-
	chrX_reverse_73151696	GWint	73151696	73149057	-
	chrX_reverse_73151881	GWex	73151881	73151696	-
	chrX_reverse_73155713	GWint	73155713	73151881	-
	chrX_reverse_73155762	GWex	73155762	73155713	-
	chrX_reverse_73155917	GWint	73155917	73155762	-
	chrX_reverse_73156032	GWex	73156032	73155917	-

5.4.4. Visualize HSPs

GenCHASE visualizes the HSPs, reported by different homology-based gene finding methods, and the introns/exons, predicted by gene prediction methods, that belong to a particular gene, using the Bio::Graphics module from the Bioperl

package (Stajich et al., 2002). GenCHASE component methods search for genes in one chromosome at a time and GenCHASE provides two visualizations; one for the genes found in the forward frame of a particular chromosome and the other for the genes found in reverse direction. For example, a visualization using the co-ordinates for a predicted gene, in Table 11, is shown in Figure 54. In GenCHASE visualizations, the background for the intron/exons predicted by gene prediction methods is shown in grey while the background for the HSPs from homology-based gene finding methods is shown in white. Each particular track in GenCHASE visualizations represents all the co-ordinates for a particular gene, reported by a single method. For the purpose of visualization, all the genes found in a frame of a particular chromosome using GenCHASE are shown next to each other not showing the original gap between them, as shown (see reverse frame) in Figure 56.

5.4.5. Calculate C-value

Just as in the case of CHASE, we transform E-values using log 10, before doing any further manipulation, and denote them as e-values with a "small e", for conciseness. Once we have got e-values $e_1,\ldots,e_n$ and method performance weights $W_1,\ldots,W_n$ for all n methods, we calculate the *c-value* for each cluster of HSPs s (see page 93) as the W-weighted sum:

$$c\text{-}value(s) := \sum_{i=1}^{n} e_i\ (s) \bullet W_i \quad .$$

The final *C-value* (on the original E-value scale) is then obtained as:

$$C\text{-}value(s) := 10^{\,c\text{-}value(s)}$$

Since it is difficult to estimate the method performance weights for GenCHASE homology-based gene finding methods, due to a lack of a generally accepted standard of truth, we use equal weights for all these methods. "Missing E-values" arise if a homology-search method finds a sequence not found by another, given

the E-value cut-off (E_C =2). In the *c-value* formula, these are set to the log to the base 10 of cut-off E-value E_C.

5.4.6. Visualization of the Gene Translation

The translations for the gene(s) that GenCHASE predicts, are visualized using a tool called VisCoSe (Spitzer et al., 2004). VisCoSe first aligns the sequences using an alignment tool, Mafft, and then displays the alignment where the amino acids are coloured according to conservation. VisCoSe also displays a consensus sequence beneath the alignment. We visualize the translations of the genes that GenCHASE finds, together with the query set of proteins, so that one can quickly find out whether the characteristic motifs of the protein family in question are present in the translations.

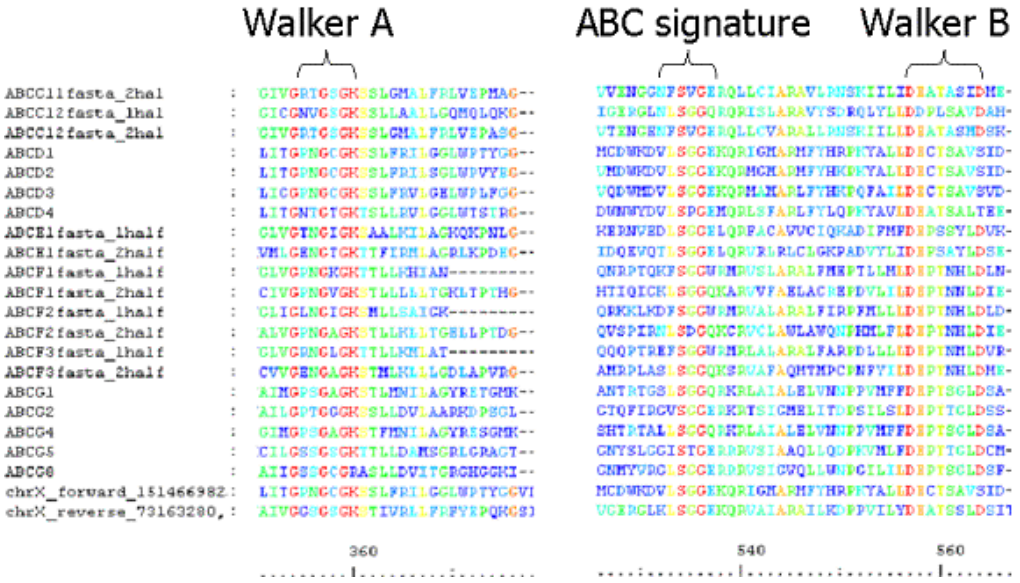

Figure 55. Visualization of GenCHASE gene translations using VisCoSe

Partial translations of two GenCHASE genes (chrX_forward_151466982 and chrX_reverse_73163280), along with some of the query sequences, are visualized using the tool VisCoSe. This tool first uses Mafft to align the sequences and then displays the alignment in colour according to the conservation of amino acids. Here in this Figure the conserved ABC motifs are

visible in red, both in query and the gene translations, which tells that these GenCHASE predictions are candidate ABC genes.

One such visualization is shown in Figure 55 displaying, along with some of the query sequences, the translations of two ABC genes (chrX_forward_151466982 and chrX_reverse_73163280) that we found in human Chromosome X using GenCHASE. In this Figure, the characteristic motifs of ABC transporters (namely ATP/GTP binding or Walker A motif (PROSITE: PDOC00017), ABC signature motif (PROSITE: PDOC00185) and the Walker B motif) are visible.

When these steps (5.4.1-5.4.6) are completed, GenCHASE writes its output in a report as shown in Figure 56. Please note that Mast is not compatible with larger genomic sequences, such as whole human chromosomal sequences, so its results are not available for human genome analysis.

Figure 56. Example GenCHASE report

A GenCHASE report can be divided into three sections where the first section shows the input information, the second section displays the visualization of HSPs (and the gene structure) and the last section provides links to individual method reports, and to the visualization of translations. GenCHASE hits are sorted on the basis of C-value.

5.5. GenCHASE-Analyser (GCA)

The GenCHASE-Analyser (or GCA) is used to run GenCHASE on all chromosomes, of for example the human genome, one by one and to analyse the results given an annotation file (that can be obtained from Ensembl using Ensmart, see: http://www.ensembl.org/Multi/martview), representing the co-ordinates of all known genes of the protein family in question and given the parameters to run GenCHASE. This is particularly helpful in looking at all the GenCHASE genes, from all the chromosomes of one complete genome, and separating the known genes from new or unknown genes for further analysis. For example we use GCA to compare the co-ordinates of the above-mentioned chrX_reverse_73163280 gene with the Ensembl-based annotation of all known human genes of the ABC protein family, and we found that chrX_reverse_73163280 is a known ABC Transporter gene namely, ABCB7, as shown in the example GCA report in Figure 57.

GenCHASE-Analyser (Genome based Comparative Homology Agreement SEarch)-Analyser
version 1.0 Copyright (C) 2004 Intikhab Alam
-Please_Cite:-
a) Alam, I., Dress, A., Rehmsmeier, M., Fuellen, G. (2004a). Comparative Homology Agreement SEarch: An effective combination of homology-search methods. Proceedings of the National Academy of Sciences (**PNAS**), USA, volume 101 (38) 13814-19.
b) Alam, I., Dress A., Fuellen, G. (2004b). GenCHASE, An Effective Combination of Genome based protein Homology-Search Methods. {Unpublished} Bielefeld University, Bielefeld, Germany

Analysing [abcs_jc5] Searches in [gclPvalues] set of Chromosomes/Genomes

*Important: Candidate {abcs_jc5} hits are shown in **bold**, wherever the Ensembl annotation is 'Not Availab search hit Evalue from GenCHASE gene translations is <2. To see their alignments click on HMM-Evalue

No.	ClusterID	CValue	HMM-Evalue	Cluster Length BPs, AAs	Pointers	Ensembl Annotation
1	chr17_reverse_67894547	1e-59	2.7e-154	222310 bp, 2746 aa	UCSC, Ensembl	67735261-67671970 ABCA6 67
2	chr7_reverse_86795277	1e-59	4.2e-130	78625 bp, 953 aa	UCSC, Ensembl	86716818-86643395 ABCB4 869
3	chr7_reverse_86704202	1e-58	5.3e-128	60935 bp, 1171 aa	UCSC, Ensembl	86716818-86643395 ABCB4
4	chr7_forward_20427123	7e-57	2e-121	112440 bp, 1131 aa	UCSC, Ensembl	20412600-20539563 ABCB5
5	chr2_reverse_170045338	1e-56	3.1e-124	62662 bp, 964 aa	UCSC, Ensembl	170090377-169981993 ABCB11
6	chr10_forward_101228363	3e-39	3.6e-127	47612 bp, 1160 aa	UCSC, Ensembl	101207158-101276168 ABCC2
7	chr16_reverse_16253058	6e-39	3e-08	42714 bp, 127 aa	UCSC, Ensembl	16283668-16210344 ABCC6
8	chr16_forward_16131853	5e-35	100	69639 bp, 36 aa	UCSC, Ensembl	16009884-16202628 ABCC1
9	chr3_reverse_185016325	1e-33	7.9e-121	56324 bp, 1001 aa	UCSC, Ensembl	185056590-184958632 ABCC5
10	**chr15_reverse_20246491**	**2e-33**	**9.5e-18**	7843 bp, 351 aa	UCSC, Ensembl	Not Available
11	chr16_reverse_47958295	4e-33	1.8e-129	62709 bp, 1293 aa	UCSC, Ensembl	47958521-47895218 ABCC12
12	chr6_forward_43442572	5e-32	9.2e-107	22112 bp, 1522 aa	UCSC, Ensembl	43442147-43465018 ABCC10
13	chr16_reverse_2298543	2e-31	1.2e-84	40563 bp, 1652 aa	UCSC, Ensembl	2319649-2266599 ABCA3
14	chr12_reverse_121871991	1e-29	3.3e-59	20007 bp, 528 aa	UCSC, Ensembl	121888566-121851067 ABCB9
15	chr16_reverse_48028197	3e-29	5.3e-127	49052 bp, 1030 aa	UCSC, Ensembl	48056093-47978782 ABCC11
16	chr21_forward_14593630	5e-28	1.6e-20	67910 bp, 607 aa	UCSC, Ensembl	14567991-14656967 ABCC13
17	chr17_forward_49215680	3e-27	1.9e-123	27519 bp, 1195 aa	UCSC, Ensembl	49186866-49243700 ABCC3
18	chr2_reverse_216058297	3e-25	5.6e-82	56953 bp, 1357 aa	UCSC, Ensembl	216205693-215998809 ABCA12
19	**chr3_forward_58287207**	**3e-25**	**2e-06**	1866 bp, 287 aa	UCSC, Ensembl	Not Available
20	chr7_forward_150119974	5e-25	1.2e-63	14203 bp, 745 aa	UCSC, Ensembl	150117278-150134393 ABCB8
21	chr6_reverse_32868374	1e-23	4.5e-55	12677 bp, 966 aa	UCSC, Ensembl	32868712-32859948 TAP1
22	chr6_reverse_32852970	1e-23	3.4e-51	9327 bp, 686 aa	UCSC, Ensembl	32852970-32837030 TAP2
23	chr7_forward_116734728	2e-22	1e-86	126937 bp, 998 aa	UCSC, Ensembl	116674520-116863218 CFTR
24	chr19_forward_992361	5e-22	1.1e-84	24063 bp, 2235 aa	UCSC, Ensembl	992361-1016424 ABCA7
25	chr12_reverse_21919944	3e-21	3e-58	74699 bp, 763 aa	UCSC, Ensembl	21980875-21845245 ABCC9
26	chr1_reverse_226636549	1e-20	1.2e-64	21412 bp, 337 aa	UCSC, Ensembl	226655653-226613541 ABCB10
27	chr7_forward_48173425	2e-20	2.8e-67	228897 bp, 1363 aa	UCSC, Ensembl	47982644-48431892 NM_152701
28	chr17_reverse_67534139	3e-19	3.7e-91	72790 bp, 1164 aa	UCSC, Ensembl	67535298-67461349 ABCA8
29	chr9_reverse_102973946	5e-19	1.5e-82	47512 bp, 1740 aa	UCSC, Ensembl	103070274-102923121 ABCA1
30	chr17_reverse_67654889	6e-19	2.8e-47	85710 bp, 1270 aa	UCSC, Ensembl	67644390-67568108 ABCA9
31	chr1_reverse_93994179	9e-19	1.3e-98	58991 bp, 1448 aa	UCSC, Ensembl	94058489-93933466 ABCA4
32	chr11_reverse_17418100	2e-17	3.2e-90	39253 bp, 1119 aa	UCSC, Ensembl	17462632-17378847 ABCC8
33	chrX_reverse_73163454	7e-17	5.6e-58	23458 bp, 601 aa	UCSC, Ensembl	73242910-73139907 ABCB7
34	**chr7_reverse_23345133**	**2e-15**	**4.6e-12**	664 bp, 204 aa	UCSC, Ensembl	Not Available
35	chr13_reverse_93545192	2e-14	6.5e-64	36785 bp, 432 aa	UCSC, Ensembl	93651684-93370091 ABCC4

Figure 57. A (truncated) **GCA report**

A part of a GCA report is shown where all the GenCHASE hits are sorted on the basis of their combined E-value (or C-value). Cluster IDs (or GenCHASE gene IDs) are hyperlinked to the individual GenCHASE reports, while the co-ordinates of these genes are hyperlinked to UCSC and Ensembl genome browsers. Length of GenCHASE genes (with introns) and their translations are listed in nucleotide base pairs and amino acids, respectively. Available annotation for each of the GenCHASE genes is shown in the last column. Annotation for our example gene,

chrX_reverse_73163280 i.e ABCB7 is marked with an arrow. Where no annotation is available, we write 'Not Available'.

GCA enlists all the genes that we find using GenCHASE, ranked on the basis of their C-values, along with their annotations and hyperlinks to the University of California Santa Cruz (UCSC) (and from there on to NCBI) and the Ensembl genome browsers. This helps to further explore the existing evidence at the NCBI or Ensembl genome browsers, and to focus on the GenCHASE genes that are, for example, unknown according to Ensembl annotations. To track down every detail, gene Ids of genes enlisted in the GCA report are further hyperlinked to individual GenCHASE reports, which contain HSP visualizations and the reports of individual GenCHASE component methods.

5.6. Modular Structure of GenCHASE

Separate module files namely GenIPs.pm, GenRunParse.pm and GenDBreader.pm were written for GenCHASE. In addition, a driver script was written that reads an eXtensible Mark-up Language (XML) (Achard et al., 2001) configuration document file and calls the modules to carry out one complete GenCHASE run. In the following section all these are discussed in detail.

5.6.1. GenCHASE Modules

- **GenIPs.pm**

The input set of sequences is first validated by the GenCHASE driver script and then passed on to the GenCHASE Input Processors (GenIPs.pm) module. The Input Processors module, as shown in Figure 58, transforms the given set of sequences into a specific format (for example an alignment, a Prosite-like pattern or a MEME motif profile) that is required by a particular GenCHASE component method, as discussed before. All homology search methods, except Mast, use an

alignment scheme in one way or the other. For example PSI-TblastN requires an alignment in a specific format and the consensus sequence required by PHI-TblastN and TfastY is also based on an alignment scheme.

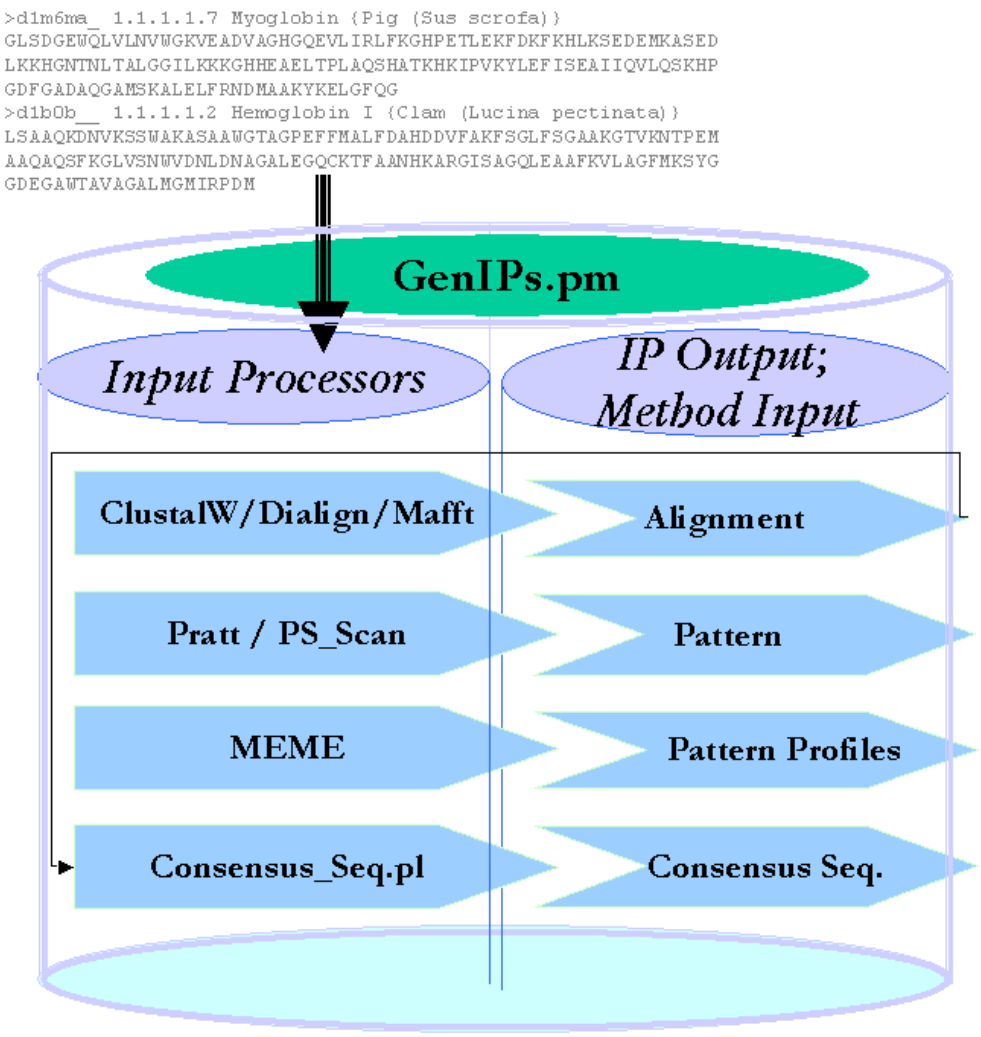

Figure 58. An outline of GenCHASE Input Processors (GenIPs.pm) Module

Input processors are used to generate specific inputs required by different homology search methods. The Pattern or pattern profile is generated directly from the input set of sequences while the consensus sequence is based on an alignment.

A newly implemented input processors module, as shown in Figure 58, made it possible to easily use any of the ClustalW, Dialign or Mafft alignment schemes, whenever it is required to generate the PSI-TblastN specific alignment or the consensus sequence. Pratt/PS_Scan and MEME produce a pattern or a pattern profile, directly using the input set of sequences. To save time and computing resources, once a specific input such as an alignment is produced that is required by a particular method, it is not generated again if the same is required by another homology search method.

- **GenRunParse.pm**

This module, as shown in Figure 59, deals with the execution of GenCHASE component methods given the required inputs. One method is executed at a time. When a particular database search is completed, its report is parsed using the parsing scripts implemented in GenRunParse.pm to extract some specific information as mentioned in section 5.4.1. This information is returned (in the form of a method specific table) to the driver script for further analysis.

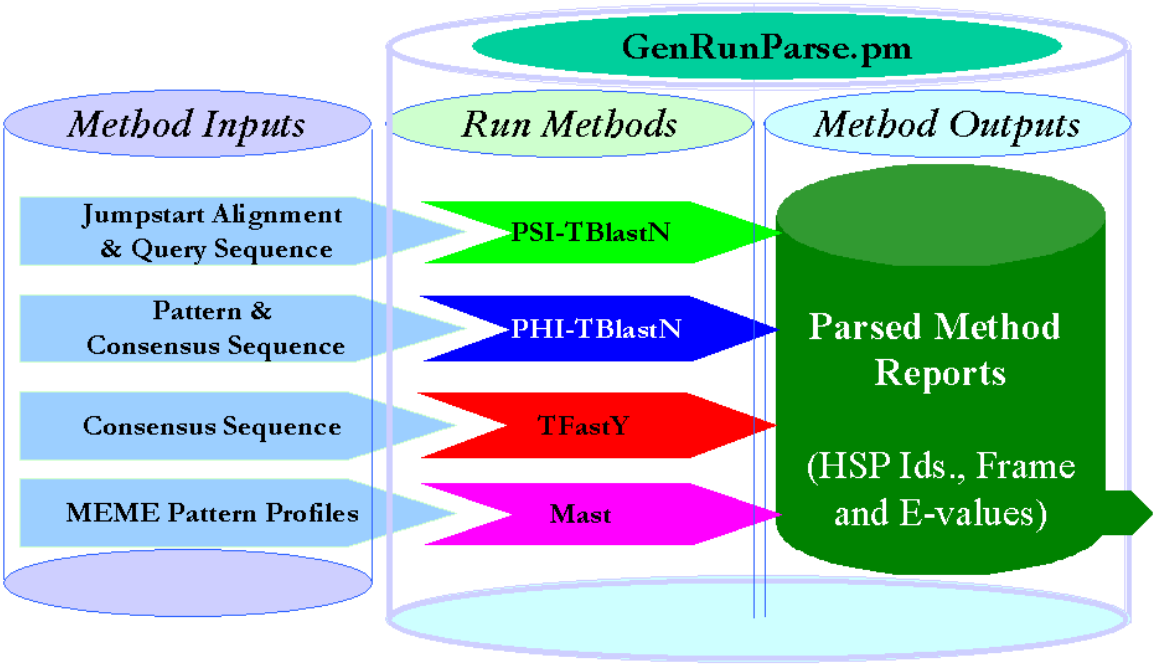

Figure 59. An outline of the GenCHASE Run and Parse (GenRunParse.pm) Module

Given the specific method input, prepared using input processors, GenCHASE component methods are executed one by one and their output is parsed to get the specific information that is used later by the GenCHASE driver script for further analysis.

- **GenDBreader.pm**

The database reader or the DBreader.pm module, as shown in Figure 60, is written exclusively to read the Fasta formatted database of DNA/genomic sequence(s). This module reads the target database to report its size (total number of nucleotides) and extracts information such as the sequence identifier(s), description lines and the sequence(s). This information is then stored in a table that is later used by the driver script.

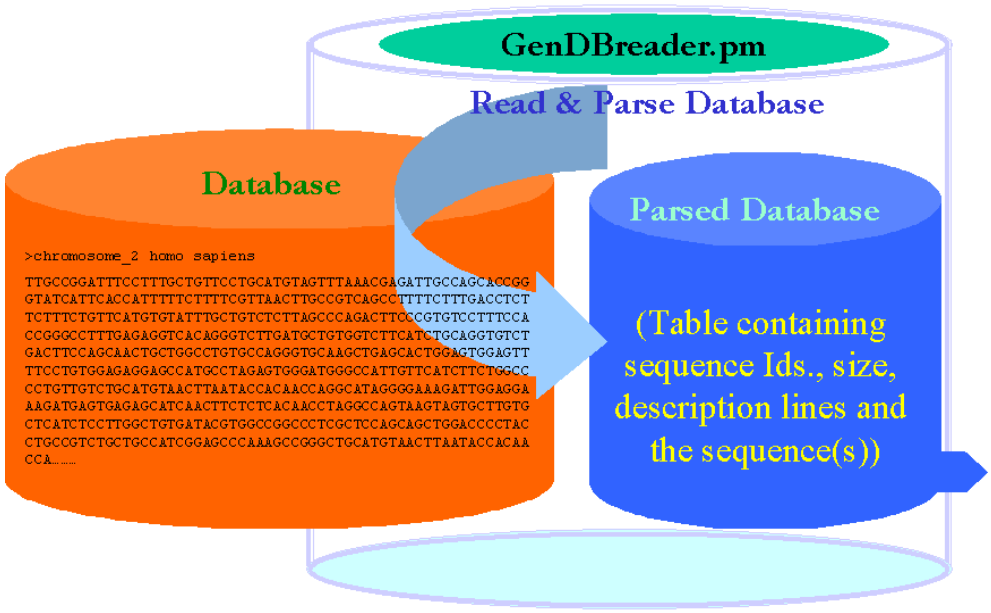

Figure 60. An outline of GenCHASE Database Reader (GenDBreader.pm) Module

The database reader module reads the sequence database (usually a single chromosome of a genomic sequence), given in Fasta format, to extract some specific information that is then passed back to the GenCHASE driver script.

5.6.2. The Configuration File

In modular GenCHASE we make use of a configuration file so that one may be able to apply the user-defined configurations without changing the GenCHASE main script, similar to the one we used for modular CHASE. This configuration file contains information such as the path to several tools, databases and directories, and the method specific information such as the name of the method, its class (e.g. if it is classified as an alignment based method), the alignment scheme that it may use, its weight, etc.

The GenCHASE configuration file, as shown in Figure 61, follows the conventions of the eXtensible Mark-up Language (XML). In XML format one has to place the actual contents enclosed in a specific opening and a closing tag, in a hierarchical fashion. For example we start the GenCHASE configuration file with

a main opening tag ‘<GENCHASECONF>’, then we have a section for paths tagged ‘<Paths>’ and one for the method-specific information tagged ‘<Method>’. Within the Paths or the Method section each element has its own opening and closing tags. At the end, each section has its closing tag and the whole document ends with the main closing tag ‘</GENCHASECONF>’.

```
- <GENCHASECONF>
  - <Paths>
      <PERLPATH>/vol/perl-5.6.1/bin/perl</PERLPATH>
      <SWISSPROT>/vol/biodb/fasta/sprot.fas</SWISSPROT>
      <SCOP>/vol/genchase/project/genomic/share/scop1.65/pdb90_1.65</SCOP>
      <MTUBERCULOSIS>/vol/genchase/project/genomic/share/genomes/fasta/mtuberculosis.fas</MTUBERCULOSIS>
      <TFASTY34>/vol/genchase/project/genomic/share/fasta3/tfasty34</TFASTY34>
      <HMMER>/vol/genchase/project/genchase-1.0/share/hmmer-2.1.1/binaries/</HMMER>
      <BLASTPGP>/vol/genchase/project/genchase-1.0/share/blast/blastpgp</BLASTPGP>
      <BLASTALL>/vol/genchase/project/genchase-1.0/share/blast/blastall</BLASTALL>
      <MEME_MAST>/vol/genchase/share/progs/test/meme.3.0.4/bin/</MEME_MAST>
      <PRATT>/vol/genchase/project/genchase-1.0/share/pratt/pratt</PRATT>
      <PS_SCAN>/vol/genchase/project/genchase-1.0/share/ps_scan/ps_scan.pl</PS_SCAN>
      <FITCH>/vol/genchase/project/genchase-1.0/share/phylip-3.5/fitch2</FITCH>
      <CLUSTALW>/vol/genchase/project/genomic/share/clustalw</CLUSTALW>
      <MAFFT>/vol/genchase/project/genchase-1.0/share/mafft/scripts/fftns</MAFFT>
      <READSEQ>/vol/genchase/project/genchase-1.0/share/readseq/readseq</READSEQ>
      <DIALIGN>/vol/genchase/project/genchase-1.0/share/dialign2_dir/dialign2-2</DIALIGN>
      <CONSENSUS>/vol/genchase/project/genchase-1.0/share/consensus_seq/consensus.pl</CONSENSUS>
      <VISCOSE>/vol/genchase/project/genomic/share/viscose-1.0/viscose.pl</VISCOSE>
      <CHASE_OUTPUT>/vol/genchase/out/genetik_fas/</CHASE_OUTPUT>
      <INPUT>/vol/genchase/project/genchase-1.0/internal/</INPUT>
      <TEMP>/vol/genchase/project/genomic/tmp/</TEMP>
    </Paths>
  + <Method>
  + <Method>
  - <Method>
      <Name>TFASTY34</Name>
      <Weight>0.25</Weight>
      <Class>SPEC</Class>
      <IParg1>fasta_seq</IParg1>
      <IParg2>None</IParg2>
      <IParg3>None</IParg3>
    </Method>
  + <Method>
  </GENCHASECONF>
```

Figure 61. GenCHASE configuration file, an example

GenCHASE configuration file is similar to the one used for CHASE and it is implemented in XML format. It starts with a main opening tag and within that the user-defined paths to several tools, databases and directories can be specified. User-defined method-specific information is placed in the methods section.

A particular browser, such as Internet explorer or Mozilla, that recognizes the XML syntax, highlights the tags and differentiates the actual data that is enclosed within these tags, as shown in Figure 61 for an example XML document (opened in the Mozilla browser). Sections or subsections start with a negative sign (-) where the data is shown. The subsections where the data is hidden start with a positive sign. Clicking on the positive sign of a section or subsection shows the actual data inside the tags.

5.6.3. GenCHASE Driver Script

Given the set of sequences as an input and the user-defined or default options, the GenCHASE driver script integrates, as shown in Figure 62, all of the above mentioned modules to get the information that is processed further to carry out a complete GenCHASE run. The GenCHASE driver script works in stages S0-S6. In the initial stage (S0), it validates the input set of sequences, user-defined (or default) options and reads the configuration file in stage S1 to validate if all the tools, databases and directories are accessible. In the second stage (S2), it instructs the input processors (GenIPs.pm) module, as mentioned above, to reformat the input data into a format that is required by the GenCHASE component methods.

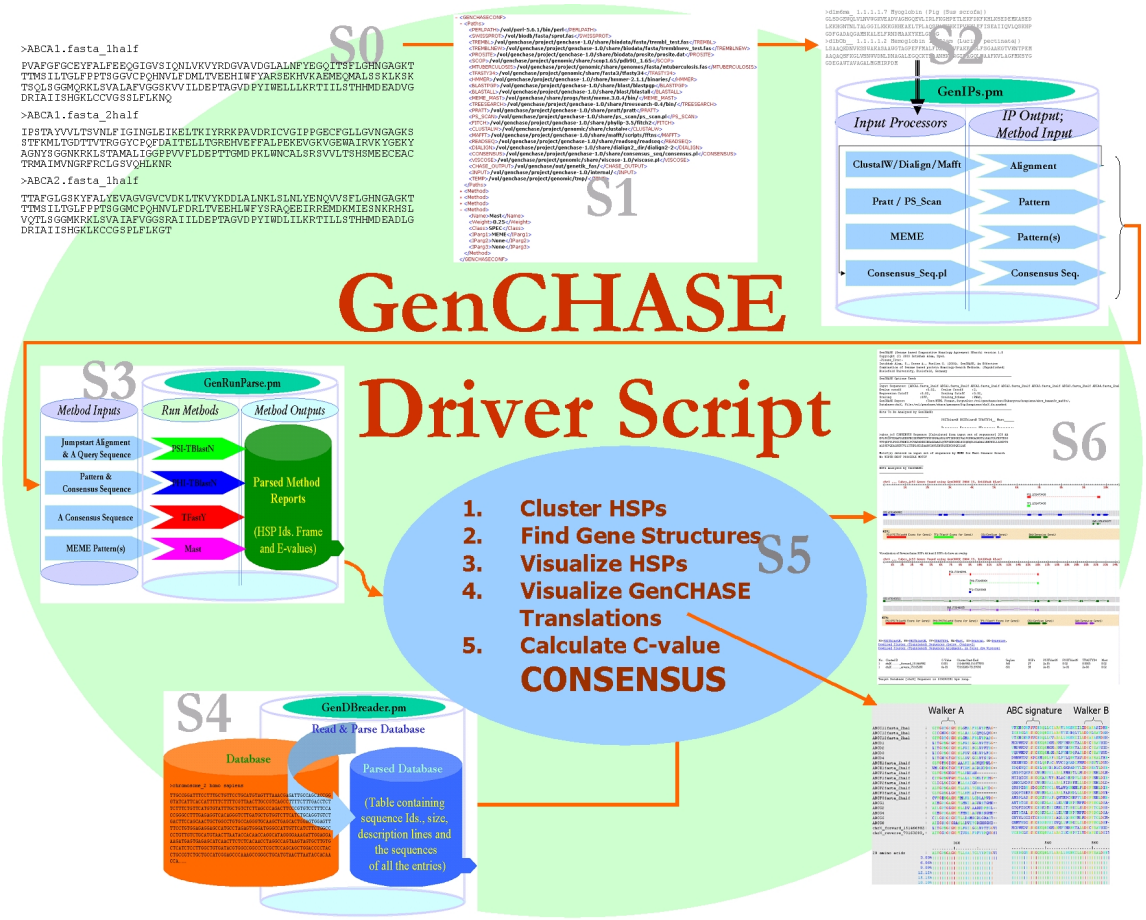

Figure 62. An outline of the GenCHASE driver script

Given a set of protein sequences (validation in stage S0) and user-defined/default options (validation in stage S1), the Input-processing module (stage S2) prepares the information used by the homology search methods. The GenRunParse module (stage S3) runs these methods and parses their output to report the co-ordinates and E-values of the High-Scoring Segment Pairs (HSPs). After reading the genomic database (stage S4) to retrieve sequences corresponding to the HSPs, in stage S5 the GenCHASE driver script clusters the overlapping HSPs, extends those genomic regions where these HSPs are found and predicts the gene structure using homology-based methods and the ab-initio methods. Finally in Stage S6, GenCHASE prepares a report that shows all the predicted genes, sorted on the basis of their C-values.

The third stage (S3) of the GenCHASE driver script calls the GenCHASE RunParse module to execute the homology-based gene finding methods and parses their reports, one by one, to report the HSP ids and the E-values, as described in section 5.4.1. After reading the genomic database (stage S4) to retrieve the sequence data corresponding to the HSPs, In Stage 5 (S5) GenCHASE combines the homology-based gene finding and ab initio gene structure prediction methods in five steps, as shown in Figure 62. E-values of the homology-based search methods are combined into a C-value as described in section 5.4.5. The last stage (S6) of the GenCHASE driver script deals with reporting the GenCHASE results as shown in Figure 66.

5.7. Case Studies

5.7.1. Overview

GenCHASE finds as many members as possible of a given protein family, in genomes, by combining homology search methods and ab initio gene structure prediction methods. To test the performance of GenCHASE we carried out case studies on test protein families. More precisely, using GenCHASE we find members of ABC, Cadherins and S100 protein families in the human genome.

5.7.2. Test Protein Families

5.7.2.1. ABC Proteins

The ATP-binding cassette (ABC) proteins constitute a large family, mostly membrane proteins responsible for the translocation of a wide variety of substances across extra/intracellular membranes (Stefkova et al., 2004; Schmitt and Tampe, 2002). ABCs are found in both prokaryotes and eukaryotes (Quentin and Fichant, 2000; Dassa and Bouige, 2001). Most ABC proteins contain two highly conserved domains, namely the ATP-Binding Cassette (ABC) domain (also known as the Nucleotide Binding Domain (NBD)) and the ABC transmembrane domain (TMD), (See PF00005 for ABC domain and PF00664 for the ABC transmembrane domain of the protein family from the PFAM database: http://www.sanger.ac.uk/Software/Pfam/). The eukaryotic ABC genes are mostly organized either as full transporters containing two TMDs and two NBDs, or as half transporters (Hyde et al., 1990). The ABC domains contain characteristic motifs such as the Walker A motif (PROSITE: PDOC00017), the ABC signature motif (PROSITE: PDOC00185) and the Walker B motif. The ABC signature motif is located just upstream of the Walker B site (Hyde et al., 1990).

5.7.2.2. Cadherin Proteins

Cadherins are a family of Ca2+-dependent cell-cell adhesion receptors that mediate cell adhesion and play a fundamental role in normal development (Yagi et al., 2000). They participate in the maintenance of proper cell-cell contacts. Cadherins depend on calcium for their function: removal of calcium abolishes adhesive activity and renders cadherins vulnerable to proteases. Their cell-adhesive property also makes them likely candidates for tumor suppressor genes (Kremmidiotis et al., 1998).

Cadherins typically consist of five tandem repeated extracellular domains or ectodomains, a single membrane-spanning segment and a cytoplasmic region (Wu et al., 1999). In addition to being homologous among each other, these repetitive cadherin ectodomains contain characteristic sequence motifs LDRE and DXDNDN in the C-terminal part and a DXD motif in the N-Terminal part, which participates in the formation of calcium binding pockets linking neighbouring cadherin ectodomains (Wu et al., 1999). The multiple repetitive domain organization of Cadherins and their crucial biological role make them an important candidate for identification and analysis with bioinformatics tools (Julia et al., 2004).

5.7.2.3. S100 Proteins

S100 proteins form one of the largest subfamilies of the EF-hand protein superfamily. These proteins feature Calcium-binding motifs composed of two helixes (E and F) joined by a loop. These are small, acidic calcium binding proteins that contain two distinct EF-hand calcium-binding motifs. The S100-specific EF-hand (PF01023) is located at the N-terminus, followed by a classical Ca2+-binding EF-hand (PF00036). In addition to Ca2+ many S100 proteins, of currently 20 known members in human, display high affinities towards Zn2+ and Cu2+ ions (Marenholz et al., 2004). These proteins regulate a variety of cellular

processes via interaction with different target proteins. Several diseases, including cancer and melanoma, are related to the abnormal expression of S100 proteins, which are expressed in a cell- and tissue-specific manner (Kanamori et al., 2004).

5.8. Application of GenCHASE on the Human Genome to Find Test Protein Families

We use the GenCHASE analyser (GCA) to apply GenCHASE on a particular genome to find members of our test protein families. GCA applies GenCHASE to all genomic sequences of a particular genome, one by one, and analyses the results as mentioned in section 5.5. To apply GenCHASE on a particular genome, we start with a training set of sequences taken from the protein family in question. Further, where possible, we compile the knowledge of already known genes into an annotation file using the Ensmart tool from Ensembl, as mentioned in section 5.5. Such an annotation of known genes serves as a basis to evaluate the performance of GenCHASE (i.e., how many of the known genes are picked up by GenCHASE, given a training set of sequences from a particular protein family) and to separate the known genes from the unknown. Hyperlinks to UCSC (and thereby to NCBI) and Ensemble genome browsers, applied to the co-ordinates of the genes found by GenCHASE, in the GCA report, further help to explore the existing evidence about the genes, which are unknown according to Ensembl annotations.

To find ABC proteins in the human genome, we used a refined dataset of all known human ABC proteins (Spitzer 2004b). There are two kinds of ABC transporters, full transporters containing two ABC domains and half transporters containing one ABC domain. This dataset contained only the conserved part of the original complete sequences. Full ABC transporters were split into two sequences, each containing one ABC domain so that the conserved parts of all the sequences can be aligned to each other. The co-ordinates of the known ABC

genes in the human genome were collected from the Ensembl database using the tool Ensmart.

To find Cadherins and S100s in the human genome, we took their seed alignment sequences, from the PFAM database and prepared annotation files containing co-ordinates for all of their known genes, using Ensmart.

5.9. GenCHASE Results and Discussion

We run GenCHASE to find the members of our test protein families in the human genome. As an evaluation of GenCHASE, the results show, as described below, that we have successfully found all the known genes for our test protein families, according to the Ensembl annotations with which we compare our results. In addition, GenCHASE found several hits where no Ensembl annotations were available. We use HMMsearch, based on query sequences, to search the GenCHASE-found gene translations. In the GCA report all those hits are displayed in **bold**, where the HMMsearch E-value is below a certain threshold (the same as the E-value cut-off, E_C=2) and no Ensembl annotations are available. In this way, interesting and un-annotated genes are separated from the known genes automatically. All such interesting genes are subjected to further analysis, such as the comparison with any further available annotation(s) (at, for example, NCBI) and an InterProScan analysis (Zdobnov et al., 2001). In an InterProScan analysis, the given protein sequence is searched against the InterPro database, an integrated resource of several motif databases. Such analysis helps to figure out whether these interesting hits are in fact new members of our test protein families, pseudo-genes or simply false positives.

5.9.1. ABC Proteins in The Human Genome

In response to the use of human ABC protein sequences as a training dataset for GenCHASE, we found all the 48 known human ABC proteins, as shown in Figure 63, according to the Ensembl annotations with which we compared our results. Apart from these 48 known ABC proteins, 29 additional hits (shown in bold in Table 12) were considered interesting since their HMMsearch E-values were below a certain threshold and no Ensembl annotation was available. InterProScan analysis revealed that 12 out of these 29 interesting hits show characteristic ABC motifs; the hits lacking motifs are called “partial”.

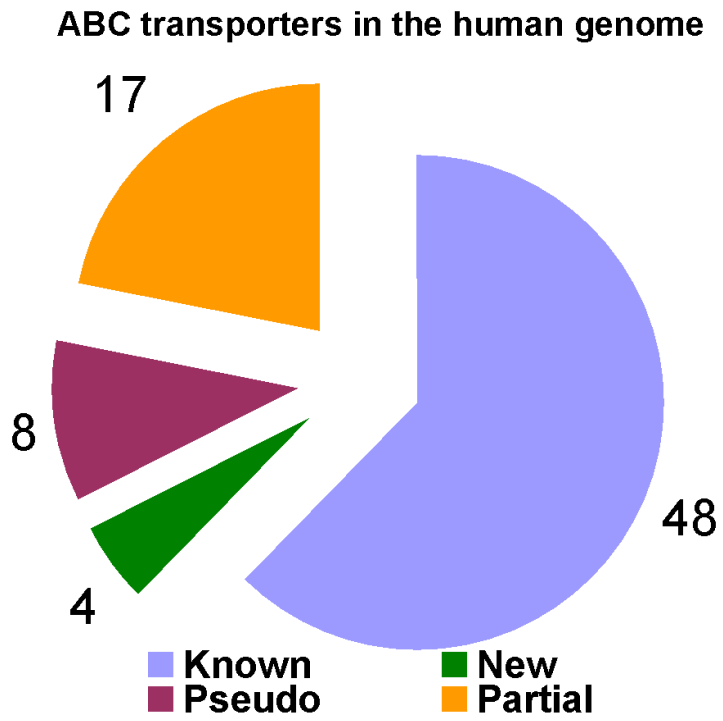

Figure 63: Summary of the search for ABC proteins in the human genome, using GenCHASE

Table 12: A table that shows the GenCHASE result for ABC protein searches in the human genome.

Ensembl, NCBI analysis and InterProScan analysis is also shown. All those hits are shown in bold where the HMMsearch E-value was below a certain cut-off and no annotation was available from Ensembl.

No.	ClusterID	C-value	HMM-E-value	PSITblastN	PHITblastN	TfastY34	Cluster Length	Ensembl Annotation	NCBI	InterProScan
1	chr17_reverse_67894547	1e-59	2.7e-154	2e-109	7e-63	2e-06	222310 bp, 2746 aa	67735261-67671970 ABCA6 67818622-67741272 ABCA10 67839914-67920413 ABCA5		
2	chr7_reverse_86795277	1e-59	4.2e-130	2e-66	2e-94	1e-17	78625 bp, 953 aa	86716818-86643395 ABCB4 86954599-86745209 ABCB1		
3	chr7_reverse_86704202	1e-58	5.3e-128	9e-68	2e-98	6e-10	60935 bp, 1171 aa	86716818-86643395 ABCB4		
4	chr7_forward_20427123	7e-57	2e-121	7e-60	3e-93	2e-17	112440 bp, 1131 aa	20412600-20539563 ABCB5		
5	chr2_reverse_170045338	1e-56	3.1e-124	2e-59	1e-88	1e-21	62662 bp, 964 aa	170090377-169981993 ABCB11		
6	chr10_forward_101228363	3e-39	3.6e-127	2e-51	1e-52	2e-13	47612 bp, 1160 aa	101207158-101276168 ABCC2		
7	chr16_reverse_16253058	6e-39	3e-08	3e-52	1e-59	7e-05	42714 bp, 127 aa	16283668-16210344 ABCC6		
8	chr16_forward_16131853	5e-35	100	1e-51	4e-49	0.0002	69639 bp, 36 aa	16009884-16202628 ABCC1		
9	chr3_reverse_185016325	1e-33	7.9e-121	1e-42	2e-46	7e-12	56324 bp, 1001 aa	185056590-184958632 ABCC5		
10	**chr15_reverse_20246491**	**2e-33**	**9.5e-18**	**1e-42**	**3e-52**	**2e-05**	**7843 bp, 351 aa**	**Not Available**	**ABCB10P**	
11	chr16_reverse_47958295	4e-33	1.8e-129	3e-45	2e-49	7e-05	62709 bp, 1293 aa	47958521-47895218 ABCC12		
12	chr6_forward_43442572	5e-32	9.2e-107	2e-43	3e-43	3e-09	22112 bp, 1522 aa	43442147-43465018 ABCC10		
13	chr16_reverse_2298543	2e-31	1.2e-84	2e-53	8e-39	0.04	40563 bp, 1652 aa	2319649-2266599 ABCA3		
14	chr12_reverse_121871991	1e-29	3.3e-59	1e-31	1e-44	2e-12	20007 bp, 528 aa	121888566-121851067 ABCB9		

15	chr16_reverse_48028197	3e-29	5.3e-127	5e-42	3e-42	0.001	49052 bp, 1030 aa	48056093-47978782 ABCC11	
16	chr21_forward_14593630	5e-28	1.6e-20	8e-39	5e-41	0.0004	67910 bp, 607 aa	14567991-14656967 ABCC13	
17	chr17_forward_49215680	3e-27	1.9e-123	2e-40	1e-38	0.006	27519 bp, 1195 aa	49186866-49243700 ABCC3	
18	chr2_reverse_216058297	3e-25	5.6e-82	4e-40	2e-28	3e-07	56953 bp, 1357 aa	216205693-215998809 ABCA12	
19	**chr3_forward_88287207**	**3e-25**	**2e-06**	**2e-49**	**2e-20**	**1e-05**	**1866 bp, 287 aa**	**Not Available**	**ABCF2P**
20	chr7_forward_150119974	5e-25	1.2e-63	3e-26	2e-36	2e-12	14203 bp, 745 aa	150117278-150134393 ABCB8	
21	chr6_reverse_32868374	1e-23	4.5e-55	3e-26	2e-34	3e-10	12677 bp, 966 aa	32868712-32859948 TAP1	
22	chr6_reverse_32852970	1e-23	3.4e-51	3e-25	4e-34	9e-12	9327 bp, 686 aa	32852970-32837030 TAP2	
23	chr7_forward_116734728	2e-22	1e-86	1e-30	1e-26	3e-10	126937 bp, 998 aa	116674520-116863218 CFTR	
24	chr19_forward_992361	5e-22	1.1e-84	3e-39	9e-25	0.06	24063 bp, 2235 aa	992361-1016424 ABCA7	
25	chr12_reverse_21919944	3e-21	3e-58	8e-27	2e-26	2e-10	74699 bp, 763 aa	21980875-21845245 ABCC9	
26	chr1_reverse_226636549	1e-20	1.2e-64	3e-22	7e-37	0.01	21412 bp, 337 aa	226655653-226613541 ABCB10	
27	chr7_forward_48173425	2e-20	2.8e-67	8e-34	1e-21	7e-06	228897 bp, 1363 aa	47982644-48431892 NM_152701	
28	chr17_reverse_67534139	3e-19	3.7e-91	2e-36	4e-19	0.03	72790 bp, 1164 aa	67535298-67461349 ABCA8	
29	chr9_reverse_102973946	5e-19	1.5e-82	2e-34	1e-20	0.05	47512 bp, 1740 aa	103070274-102923121 ABCA1	
30	chr17_reverse_67654889	6e-19	2.8e-47	4e-34	6e-19	0.001	85710 bp, 1270 aa	67644390-67568108 ABCA9	
31	chr1_reverse_93994179	9e-19	1.3e-98	5e-34	4e-20	0.04	58991 bp, 1448 aa	94058489-93933466 ABCA4	
32	chr11_reverse_17418100	2e-17	3.2e-90	4e-23	2e-26	0.01	39253 bp, 1119 aa	17462632-17378847 ABCC8	
33	chrX_reverse_73163454	7e-17	5.6e-58	1e-14	1e-29	2e-06	23458 bp, 601 aa	73242910-73139907 ABCB7	
34	**chr7_reverse_23345133**	**2e-15**	**4.6e-12**	**2e-25**	**9e-17**	**0.0005**	**664 bp, 204 aa**	**Not Available**	**ABCE1P**
35	chr13_reverse_93545192	2e-14	6.5e-64	6e-21	2e-19	0.01	36785 bp, 432 aa	93651684-93370091 ABCC4	
36	chr2_reverse_220289469	5e-14	3.6e-61	2e-12	6e-21	1e-08	12270 bp, 1136 aa	220285934-220277033 ABCB6	
37	chr21_forward_42579314	1e-13	6.8e-20	1e-18	2e-17	7e-05	26610 bp, 559 aa	42533403-42611488 ABCG1	

38	**chr1_reverse_189504876**	**5e-13**	**7.3e-09**	**3e-23**	**1e-13**	**0.04**	**781 bp, 204 aa**	**Not Available**	**ABCE1P**	
39	chr7_reverse_150315285	4e-12	1.8e-55	3e-21	2e-10	6e-05	18587 bp, 771 aa	150316058-150296664 ABCF2		
40	chr13_reverse_93466287	9e-12	1.2e-28	1e-15	6e-17	0.008	81427 bp, 565 aa	93651684-93370091 ABCC4		
41	chr4_forward_146603165	1e-10	3.1e-37	4e-17	1e-08	3e-06	23187 bp, 608 aa	146597108-146627958 ABCE1		
42	chr11_forward_118553483	2e-10	1e-28	1e-14	3e-12	0.0003	15985 bp, 674 aa	118557402-118571024 ABCG4		
43	chr9_reverse_135271051	2e-09	1.9e-102	4e-18	6e-08	0.03	11377 bp, 2021 aa	135280033-135258979 ABCA2		
44	**chr4_reverse_410752**	**1e-08**	**1.3e-22**	**2**	**3e-19**	**6e-06**	**1200 bp, 288 aa**	**Not Available**	**(FLJ14297) Hypothetical ABCA10L**	**ABC domain**
45	chr3_forward_185224909	3e-07	4.3e-45	2e-13	7e-05	0.002	7490 bp, 709 aa	185224799-185232706 ABCF3		
46	chr12_reverse_38287763	1e-06	5.9e-20	2e-12	0.003	0.0003	34987 bp, 288 aa	38300237-38232814 ABCD2		
47	chr4_reverse_89519561	1e-06	2.4e-24	3e-08	1e-05	3e-06	25057 bp, 322 aa	89538117-89471235 ABCG2		
48	chrX_forward_151466982	2e-06	2.1e-12	3e-11	0.0005	0.0003	10111 bp, 560 aa	151458227-151478120 ABCD1		
49	**chr16_forward_21910566**	**2e-06**	**2.3e-12**	**2e-10**	**8e-07**	**0.06**	**12311 bp, 237 aa**	**Not Available**	**(LOC342293) Hypothetical ABCA3L**	**ABC signature, Walker B**
50	**chr1_forward_189504906**	**3e-06**	**0.01**	**9e-12**	**2e-06**	**2**	**326 bp, 108 aa**	**Not Available**	**ABCE1P**	
51	chr14_reverse_72757012	5e-06	2.3e-17	5e-11	1e-06	2	22518 bp, 778 aa	72759736-72742190 ABCD4		
52	chr6_forward_30651340	8e-06	2.4e-42	2e-08	0.0002	0.0001	13284 bp, 752 aa	30645327-30665235 ABCF1		
53	**chr16_forward_2366164**	**1e-05**	**0.078**	**3e-09**	**3e-07**	**2**	**56815 bp, 781 aa**	**Not Available**	**ABCA3L**	**ABC signature, Walker B**
54	chr2_forward_228869943	0.0002	100	2e-05	0.007	5e-05	245 bp, 81 aa	Not Available		
55	**chr7_reverse_23345637**	**0.0008**	**0.15**	**1e-08**	**0.02**	**2**	**374 bp, 124 aa**	**Not Available**	**ABCE1P**	

56	chr20_reverse_25899975	0.001	100	0.0002	0.0001	0.09	4840 bp, 93 aa	Not Available	
57	chr1_forward_94427082	0.001	3.5e-18	8e-08	0.009	2	16941 bp, 230 aa	94355799-94455889 ABCD3	
58	chr15_forward_67030820	0.007	68	5e-05	0.07	0.09	224 bp, 74 aa	Not Available	
59	chr2_reverse_44040353	0.009	1.3e-26	7e-06	0.06	2	23603 bp, 543 aa	44040493-44014146 ABCG5	
60	chr2_forward_44040728	0.01	5e-22	0.004	0.03	0.01	13814 bp, 322 aa	44040638-44080139 ABCG8	
61	**chr2_forward_99340004**	**0.01**	**0.0093**	**0.003**	**0.0002**	**2**	**611 bp, 203 aa**	**Not Available**	
62	chr6_reverse_90486909	0.02	23	0.002	2	0.003	9051 bp, 115 aa	Not Available	
63	**chr5_forward_132049030**	**0.03**	**0.1**	**0.9**	**0.03**	**0.001**	**5242 bp, 286 aa**	**Not Available**	
64	chr5_forward_98092285	0.06	100	0.3	2	0.0005	182 bp, 60 aa	Not Available	
65	**chr3_forward_52349099**	**0.06**	**0.03**	**0.3**	**0.06**	**0.01**	**17140 bp, 1232 aa**	**Not Available**	
66	**chr11_forward_102577201**	**0.08**	**0.11**	**0.2**	**0.04**	**0.07**	**18831 bp, 741 aa**	**Not Available**	
67	chr7_reverse_23345851	0.08	100	0.002	0.1	2	83 bp, 25 aa	Not Available	
68	**chr10_forward_112022582**	**0.1**	**0.0046**	**0.3**	**0.003**	**2**	**13675 bp, 505 aa**	**Not Available**	
69	**chr19_forward_48914551**	**0.2**	**0.24**	**0.06**	**0.1**	**2**	**1391 bp, 463 aa**	**Not Available**	
70	**chr11_reverse_6707420**	**0.2**	**1.5**	**0.08**	**2**	**0.09**	**7269 bp, 2296 aa**	**Not Available**	
71	chr22_forward_35565149	0.3	100	2	0.1	0.08	12425 bp, 898 aa	Not Available	
72	chr2_reverse_17709362	0.3	100	2	0.8	0.01	5651 bp, 273 aa	Not Available	
73	**chr8_reverse_27954740**	**0.3**	**0.083**	**0.8**	**0.6**	**0.07**	**14885 bp, 374 aa**	**Not Available**	
74	**chr22_forward_15242951**	**0.3**	**4.7e-06**	**0.005**	**2**	**2**	**4007 bp, 273 aa**	**Not Available**	**ABCD1P**
75	**chr5_forward_68685588**	**0.4**	**0.59**	**0.03**	**1**	**2**	**12662 bp, 295 aa**	**Not Available**	
76	**chr7_reverse_43662076**	**0.4**	**0.17**	**0.6**	**2**	**0.06**	**12734 bp, 972 aa**	**Not Available**	
77	chr7_reverse_149836626	0.5	68	2	2	0.04	744 bp, 207 aa	Not Available	
78	chr7_random_forward_186310	0.5	100	2	1	0.07	8807 bp, 181 aa	Not Available	

79	**chrX_reverse_52417216**	**0.6**	**0.12**	**2**	**2**	**0.07**	**10381 bp, 299 aa**	**Not Available**		
80	chr16_forward_21924694	0.9	100	0.2	2	2	116 bp, 38 aa	Not Available		
81	**chr2_forward_91511626**	**0.9**	**1.5e-06**	**0.2**	**2**	**2**	**5755 bp, 291 aa**	**Not Available**	**ALDPL / ABCD1L**	**ABC signature, Walker B**
82	**chr3_forward_161462126**	**1**	**0.1**	**0.5**	**2**	**2**	**10384 bp, 564 aa**	**Not Available**		
83	chr18_random_forward_3936	1	100	2	0.8	2	26 bp, 8 aa	Not Available		
84	**chr16_forward_48073295**	**1**	**0.51**	**0.5**	**2**	**2**	**16055 bp, 220 aa**	**Not Available**		
85	chrUn_random_forward_1375793	1	100	0.4	2	2	8501 bp, 215 aa	Not Available		
86	chr1_forward_225302635	1	45	2	0.4	2	8394 bp, 320 aa	Not Available		
87	chr12_reverse_110611937	1	100	1	0.8	2	140 bp, 44 aa	Not Available		
88	chr5_forward_175773490	1	4.6	0.4	2	2	6331 bp, 260 aa	Not Available		
89	**chr22_forward_37346525**	**1**	**0.91**	**0.5**	**2**	**2**	**18959 bp, 501 aa**	**Not Available**		
90	**chr17_forward_41087806**	**1**	**0.48**	**2**	**0.5**	**2**	**8980 bp, 815 aa**	**Not Available**		
91	chr19_forward_34499568	1	100	2	0.4	2	95 bp, 31 aa	Not Available		
92	chr4_reverse_109073989	1	100	2	0.4	2	7443 bp, 113 aa	Not Available		
93	chr15_random_forward_1107893	1	100	2	0.8	2	7380 bp, 291 aa	Not Available		
94	chrUn_random_reverse_1831714	1	100	0.6	2	2	89 bp, 28 aa	Not Available		
95	**chr16_reverse_33521908**	**1**	**0.2**	**0.8**	**2**	**2**	**3199 bp, 198 aa**	**Not Available**		
96	**chr16_reverse_32525868**	**1**	**0.16**	**0.8**	**2**	**2**	**3199 bp, 198 aa**	**Not Available**		
97	chrM_forward_7327	1	100	2	0.3	2	53 bp, 17 aa	Not Available		
98	**chr12_forward_38982858**	**1**	**0.044**	**1**	**1**	**2**	**11384 bp, 402 aa**	**Not Available**		
99	chr9_random_forward_1203048	1	100	0.6	2	2	329 bp, 109 aa	Not Available		

100	chr5_forward_143209343	1	100	2	0.5	2	185 bp, 61 aa	Not Available	
101	chr18_random_forward_3124	1	100	0.6	2	2	32 bp, 10 aa	Not Available	
102	chr3_reverse_42383665	2	100	2	2	2	116 bp, 37 aa	Not Available	
103	chr17_reverse_80346796	2	100	1	2	2	167 bp, 55 aa	Not Available	
104	chr20_forward_38283153	2	33	2	2	2	4105 bp, 120 aa	Not Available	
105	chr13_random_forward_109059	2	100	1	2	2	56 bp, 18 aa	Not Available	
106	chr17_forward_5102208	2	4.6	2	2	2	9194 bp, 536 aa	Not Available	
107	chr14_forward_67540156	2	6.3	1	2	2	1506 bp, 411 aa	Not Available	
108	chr7_random_reverse_285443	2	100	2	2	2	14699 bp, 161 aa	Not Available	
109	chr22_reverse_36256210	2	100	2	2	2	122 bp, 40 aa	Not Available	
110	chr13_forward_33989332	2	100	2	2	2	80 bp, 26 aa	Not Available	
111	**chr10_forward_38898558**	**2**	**4.5e-05**	**0.9**	**2**	**2**	**3194 bp, 201 aa**	**Not Available**	**ABCD1P**
112	chr11_reverse_64421296	2	12	2	2	2	9274 bp, 208 aa	Not Available	
113	chrX_random_forward_280685	2	61	0.9	2	2	8607 bp, 757 aa	Not Available	
114	chr10_random_forward_172083	2	100	2	1	2	3407 bp, 122 aa	Not Available	
115	chr8_forward_41360328	2	100	2	1	2	5716 bp, 173 aa	Not Available	
116	chr13_random_forward_28064	2	100	2	1	2	29 bp, 9 aa	Not Available	
117	chr5_reverse_53490264	2	100	0.9	2	2	6776 bp, 79 aa	Not Available	
118	chr2_random_forward_404995	2	100	1	2	2	10474 bp, 141 aa	Not Available	
119	chr6_random_reverse_1002576	2	2.4	2	1	2	6917 bp, 1315 aa	Not Available	

Following the hyperlinks to NCBI, from GenCHASE reports, 8 out of these 12 candidate ABC proteins were found to be pseudogenes, as shown in Table 12. The remaining 4 hits are our candidate ABC proteins. These are hits number 44, 49, 53 and 81in Table 12.

Experimental verification of these ABC genes is underway (Lorkowski, 2004). As a first result, the mRNA of chr4_reverse_410752 (which we call ABCA10 like (or ABCA10L)) has been found, as shown in Figure 64. A specific intron-spanning region of mRNA was detected using primers designed specifically for ABCA10L. ABCA10 is thought to be involved in lipid homeostasis (Wenzel et al., 2003), ABCA10L might have a similar role. Cloning and sequencing are the next steps. Similar analysis will be done in due time for all other candidate ABC genes.

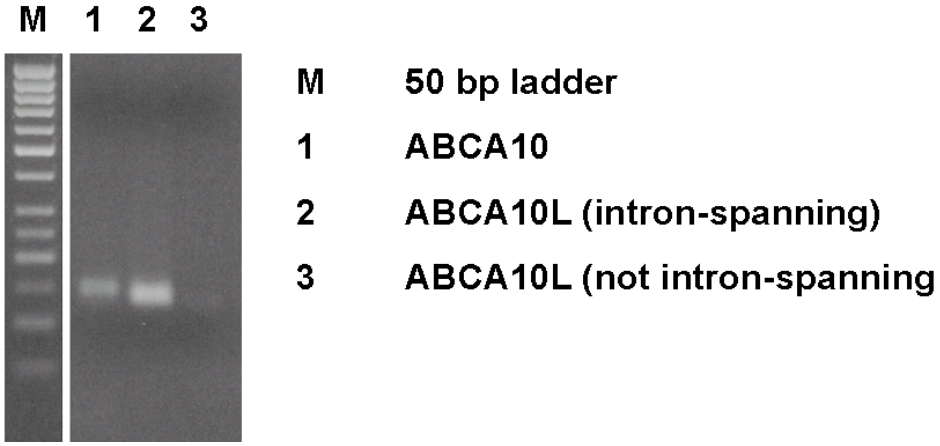

Figure 64: ABCA10L, experimental results (Lorkowski 2004)

Specific primers were designed to detect part of the mRNA for our ABCA10 like (or ABCA10L), chrr4_reverse_410752, GenCHASE hit. To avoid any bias, 3 experiments were done where in the first experiment part of ABCA10 mRNA was detected, in the second ABCA10L (intron-spanning) part of mRNA was detected and in the third experiment ABCA10L (not intron-spanning) part of mRNA was detected as shown in the gel lanes 1, 2, and 3, respectively. In the first lane 50 base pairs (bp) marker ladder was loaded.

5.9.2. Cadherins in The Human Genome

There are 70 cadherin genes that are known so far, according to an analysis carried out using Ensmart at Ensembl. However, Höng et al., 2004 claim that there are 182 known genes. Using GenCHASE we have identified all known 70 cadherin genes and in addition we find 16 candidates for cadherin genes, as shown in Table 13.

Table 13. InterProScan and NCBI/Ensembl analysis is shown for the candidate Cadherin genes in the human genome, found by GenCHASE. Please note that PF means a family from Pfam database while SSF means a family from Superfamily database (Pandit et al., 2002).

No	Hits	Interpro Analysis	NCBI/Ensembl Annotation
23	**chr11_forward_91773415**	**Cadherin Domain PF00028**	**LOC440062 Similar to fat3 (NCBI)**
32	**chr5_forward_140738753**	**Cadherin Domain PF00028**	protocadherin gamma subfamily A (1-6), B (1-3)
41	**chr5_forward_140607849**	**Cadherin Domain PF00028**	protocadherin bet 15, protocadherin beta 19 pseudo
58	**chr4_reverse_139030848**	**Cadherin Domain PF00028**	**No Info (NCBI)**
64	**chr5_forward_140648408**	**Cadherin Domain PF00028**	**No Info (NCBI)**
74	**chr13_forward_51563347**	**Cadherin Domain PF00028**	**MGC75495 similar to Serine/threonine-protein kinase Nek1**
80	**chr7_reverse_80832433**	**Cadherin Domain PF00028**	**No Info (NCBI)**
83	**chr11_forward_91930979**	**Cadherin SSF49313**	**LOC440062 Similar to fat3 (NCBI)**
87	**chr5_reverse_70358626**	**Cadherin SSF49313**	**No Info (NCBI)**
88	**chr5_forward_69682962**	**Cadherin SSF49313**	**No Info (NCBI)**
90	**chr5_forward_69169961**	**Cadherin SSF49313**	**No Info (NCBI)**
95	**chr3_reverse_63701481**	**Cadherin SSF49313**	**No Info (NCBI)**
96	**chr3_reverse_49796257**	**Cadherin Domain PF00028**	**No Info (NCBI)**
108	chr19_reverse_8046295	Calcium bindin EFG PF07645.2	FBN3 fibrillin (NCBI)
113	chr5_reverse_127678412	Calcium bindin EFG PF07645.2	FBN2 fibrillin2
115	**chr20_reverse_10868965**	**Cadherin SSF49313**	**No Info (NCBI)**

5.9.3. S100 Proteins in The Human Genome

In response to the use of S100 sequences from the Pfam seed alignment, as a training dataset for GenCHASE, we found all the 20 known human S100 proteins, according to the Ensembl annotations with which we compared our results. A summary for the search for S100 proteins in the human genome is presented in Figure 66.

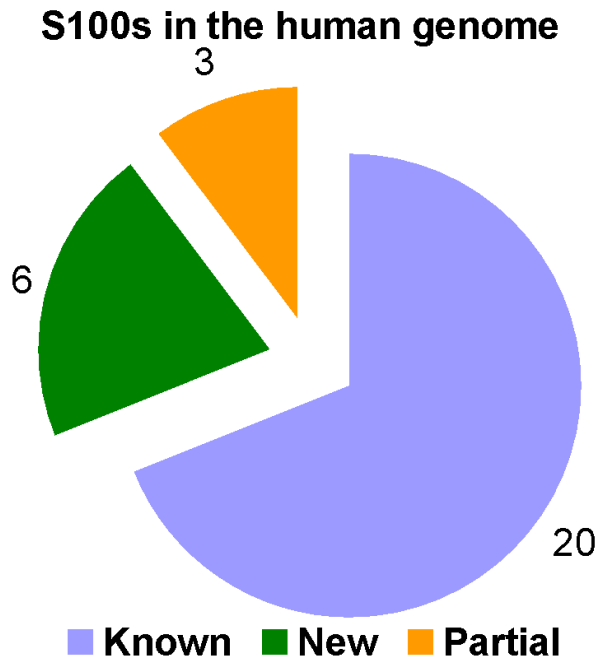

Figure 66: Summary of the search for S100 proteins in the human genome, using GenCHASE

Apart form the 20 known S100s, 9 additional hits were considered interesting since their HMMsearch E-values were below our threshold (which is the same as GenCHASE C-value threshold) and no Ensembl annotations were available. InterProScan analysis revealed that 6 out of these 9 interesting hits show characteristic S100 motifs while the rest (3 hits) show partial or no motifs, as described in Table14.

Table 14. InterProScan and NCBI/Ensembl analysis is shown for the candidate S100 genes found in the human genome. Please note that in the InterProScan analysis, PF means a family from Pfam database, SSF means a family from Superfamily database and PS means a motif/pattern entry from the Prosite database.

ID	Interproscan Analysis (Summry)	NCBI/Ensembl Annotation
chr12_forward_62331674	**S100 domain (PF01023.7), EF Hand (SSF47473)**	**hypothetical protein FLJ32949** (NCBI)
chr1_reverse_149346947	**S100 domain (PF01023.7), EF Hand (SSF47473)**	**Hypothetical protein FLJ39117**
chrX_reverse_51693359	**S100 (PS00303), EF hand (SS47473, PF00036.15)**	**SSX2 synovial sarcoma, X breakpoint 2 (NCBI)**
chrX_random_forward_733309	None	None
chr1_forward_21564237	**EF-hand (SSF47473)**	**(NCBI) ubiquitin specific protease 48 , Ensembl: UCH-2**
chr11_forward_33430932	**S100 domain (PD003407), EF Hand (SSF47473)**	**G2 Protein (NCBI), No info. (Ensembl)**
chr3_forward_63595702	**EF-hand_2 (PS50222), EF-hand (SSF47473)**	**similar to hypothetical protein A430083B19 (NCBI), EF HAND CONTAINING PROTEIN (ENSF00000011167) (Ensembl)**
chr3_forward_178034989	None	None
chr9_forward_26135713	None	None

6. Conclusions

In this thesis, I try to improve database searches, as explained in the beginning of this thesis. Different methods report different results (see Figures 8, 9, and 10) and the increasing number of database search methods poses a major problem to wet-lab users, as it is difficult to decide which methods should be used and which not. Approaches that provide a consensus over several methods may ease the work of wet-lab users and enable them to see the overall picture of results by all of the methods in one go. CHASE and GenCHASE are such approaches in the area of protein database and genomic searches.

CHASE results show that combining homology-search methods provides improved performance over an entire set of scenarios, ranging from the detection of distant to very close relationships between protein sequences. This corroborates, in the context of protein family research, the frequent claim that appropriately designed consensus methods can be more reliable than any of their component algorithms.

CHASE software version 1.0 was released with Alam et al., 2004 and is being used by a number of people particularly at The Howard Hughes Medical Institute, USA, Pasteur Institute, France, Wageningen University, the Netherlands, Massachusetts Institute of Technology (MIT), USA, Indian Institute of Chemical Biology, Kolkuta-India, Washington University, USA, and Institute of Biotechnology, Lithuania. This version of CHASE software is available at http://www.mathematik.uni-bielefeld.de/~intikhab/chase-release1.0.tar.gz The modular structure of CHASE makes it very easy to integrate a new homology search method. The programmer just needs to update the configuration file and add a run_parse module for the corresponding method.

Overall, the performance of GenCHASE is also very good as it identifies all the known genes from our test protein families and furthermore finds several new candidate genes. The unique feature of GenCHASE is its ability to predict the gene structure of genomic regions supported by a consensus of homology search methods, reducing the rate of false positives. The performance of GenCHASE can be further enhanced, given gene prediction methods such as Augustus (Stanke et al., 2003). The benefit of methods such as Augustus is its ability to predict genes in the genomic regions where some similarity information (such as the co-ordinates of Genewise predicted exons) is available. If no similarity information is provided Augustus still predicts genes with better accuracy than Genscan. It would be a task of the near future to integrate methods such as Augustus in GenCHASE.

Since searches for proteins in genomic databases are done more frequently than searches in protein databases, GenCHASE could be very useful in finding novel proteins that may even turn out to be useful drug targets. CHASE can also be used to search such targets in genomic sequences if a particular genome could be 6-frame translated into a database of small chunks.

An improvement in results often comes at a price. In general, the profile HMM software packages are very sensitive and specific, but they are far slower than other methods (Eddy 1998b). The same is true for CHASE (for a run-time analysis see section 4.4.2) and GenCHASE. If you search a large database of protein sequences using HMMsearch, it will take some time but the results are useful. Similarly, some of the methods combined in CHASE and GenCHASE such as HMMsearch, Treesearch, TfastY, and Genewise are slower than Blast-like methods. Due to this fact the overall time taken to run CHASE and GenCHASE is larger than Blast, in particular when searching larger databases. The amount of time taken by CHASE and GenCHASE to complete searches in

large databases could be reduced if a parallel version could be implemented, a task of the near future.

Approaches combining homology search methods such the CHASE are highly successful and the same seems true for approaches that combine homology search and gene-finding methods such as GenCHASE. The impact of GenCHASE remains to be seen where it can be applied to a large number of un-annotated genomes, a task of the near future.

7. Appendix A: Running Time Analysis For CHASE And Its Component Methods

A running time experiment was conducted to measure the time consumed by CHASE component methods and CHASE itself in the *even half* of SCOP database to find the members of 10 protein families (shown in Table 6). This experiment was replicated 3 times, as shown in the tables below, to estimate an average running time, as shown in Table 7, for all the methods. Sum of the running time for all methods is shown in the column headed ***Sum*** while sum of running time for CHASE component methods except HMMsearch is shown in the column headed ***Sum W/O HMMsearch***.

Replication1

Query No.	Query Name	No. of Sequences	Average Query Length	HMMseach	Treeseach	PSIBlast	PHIBlast	Mast	*Sum*	*Sum W/O HMMsearch*	CHASE
1	S100	27	44	9	81	3	5	18	*116*	107	113
2	1.36.1.2	6	71	14	42	2	5	24	*87*	73	63
3	1.41.1.2	6	92	16	48	3	4	19	*90*	74	79
4	1.23.1.1	7	103	19	55	4	4	19	*101*	82	85
5	1.73.1.1	6	126	27	64	5	5	42	*143*	116	118
6	1.128.1.1	8	127	39	116	4	5	35	*199*	160	153
7	1.27.1.1	10	170	29	88	4	6	63	*190*	161	166
8	ABCs	4	300	48	111	7	9	22	*197*	149	146
9	3.3.1.no5	17	300	56	223	9	27	228	*543*	487	463
10	Serpins	42	415	57	391	9	62	1103	*1622*	1565	1513

Replication2

Query No.	Query Name	No. of Sequences	Average Query Length	HMMseach	Treeseach	PSIBlast	PHIBlast	Mast	*Sum*	*Sum W/O HMMsearch*	CHASE
1	S100	27	44	9	97	4	17	26	*153*	144	117
2	1.36.1.2	6	71	18	42	3	4	20	*87*	69	67
3	1.41.1.2	6	92	17	53	12	3	16	*101*	84	72
4	1.23.1.1	7	103	21	54	3	4	20	*102*	81	82
5	1.73.1.1	6	126	26	69	3	5	45	*148*	122	113
6	1.128.1.1	8	127	38	110	4	5	37	*194*	156	152
7	1.27.1.1	10	170	28	80	5	6	57	*176*	148	163
8	ABCs	4	300	49	119	7	9	23	*207*	158	168
9	3.3.1.no5	17	300	53	197	11	29	221	*511*	458	499
10	Serpins	42	415	56	415	14	82	1102	*1669*	1613	1516

Replication3

Query No.	Query Name	No. of Sequences	Average Query Length	HMMseach	Treeseach	PSIBlast	PHIBlast	Mast	*Sum*	*Sum W/O HMMsearch*	CHASE
1	S100	27	44	6	70	3	3	15	*97*	91	87
2	1.36.1.2	6	71	10	34	2	3	18	*67*	57	54
3	1.41.1.2	6	92	12	45	4	2	15	*78*	66	56
4	1.23.1.1	7	103	13	47	3	2	19	*84*	71	70
5	1.73.1.1	6	126	18	57	3	3	35	*116*	98	95
6	1.128.1.1	8	127	26	85	3	5	28	*147*	121	122
7	1.27.1.1	10	170	20	70	3	5	53	*151*	131	130
8	ABCs	4	300	32	88	5	7	18	*150*	118	120
9	3.3.1.no5	17	300	37	176	6	25	191	*435*	398	400
10	Serpins	42	415	38	314	9	60	872	*1293*	1255	1256

8. References

1. Achard, F., Vaysseix, G. and Barillot, E. 2001. XML, bioinformatics and data integration. Bioinformatics, 17: 115-25.
2. Alam, I., Dress, A., Rehmsmeier, M., Fuellen, G. (2004a). Comparative Homology Agreement SEarch: an effective combination of homology-search methods. Proceedings of the National Academy of Sciences (PNAS), USA. Volume 101 (38) 13814-19.
3. Allen, J. E., Pertea, M., Salzberg, S. L. (2004) Computational gene prediction using multiple sources of evidence. Genome Res; 14(1): 142-8
4. Altschul, S. F., Bundschuh, R., Olsen, R., Hwa, T. (2001) The estimation of statistical parameters for local alignment score distributions. Nucleic Acids Res. 29(2):351-61.
5. Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., & Lipman, D. J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. 25, 3389-402.
6. Altschul, S., Gish, W. (1996) Local alignment statistics. Meth. Enzymol., 266, 460-480.
7. Andreeva, A., Howorth, D., Brenner S. E., Hubbard, T. J. P., Chothia, C., and Murzin, A. G. (2004) SCOP database in 2004: refinements integrate structure and sequence family data Nucleic Acids Res. 32(90001): D226 - 229.
8. Apweiler, R., Attwood, T.K., Bairoch, A., Bateman, A., Birney, E., Biswas, M., Bucher, P., Cerutti, L., Corpet,F., Croning, M.D. *et al.* (2001) The InterPro database, an integrated documentation resource for protein families, domains and functional sites. *Nucleic Acids Res.*, 29, 37–40.
9. Apweiler, R., Bairoch, A., Wu C. H. (2004) Protein sequence databases. Current Opinion in Chemical Biology 8: 76-80 (2004).

10. Attwood,T.K., Blythe,M.J., Flower,D.R., Gaulton,A., Mabey,J.E., Maudling,N., McGregor,L., Mitchell,A.L., Moulton,G., Paine,K. *et al.* (2002) PRINTS and PRINTS-S shed light on protein ancestry. *Nucleic Acids Res.*, 30, 239–241.

11. Bailey, T.L. & Elkan, C. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers. Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology, pages 28-36, AAAI Press.

12. Bailey, T.L., & Gribskov, M. (1998) Combining evidence using p-values: application to sequence homology searches. *Bioinformatics* 14, 48-54.

13. Bairoch, A., P. Bucher, & K. Hofmann. (1997) The PROSITE database, its status in 1997. *Nucleic Acids Res.* 25, 217-221.

14. Barton, G. J. (1993) Comput. Appl. Biosci. 9, 729-734.

15. Bateman, A., Birney,E., Cerruti,L., Durbin,R., Etwiller,L., Eddy,S.R., Griffiths-Jones,S., Howe,K.L., Marshall,M. and Sonnhammer,E.L.L. (2002) The Pfam Protein Families Database. *Nucleic Acids Res.*, 30, 276–280.

16. Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Wheeler DL (2004). GenBank: update. *Nucleic Acids Res* 2004 Jan 1, 32 Database issue: D23-6

17. Beyer, S. (2004) Bit-Vector Module:

    http://search.cpan.org/~stbey/Bit-Vector-6.4/

18. Birney, E., Clamp, M. and Durbin R. (2004) GeneWise and Genomewise. Genome Research 14: 988-995, 2004

19. Blast Glossary:
    http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/glossary2.html

20. Bleasby A. J. et. al., (1994) Nuc. Ac. Res., 22, 3574-3577.

21. Boeckmann B., Bairoch A., Apweiler R., Blatter M.-C., Estreicher A., Gasteiger E., Martin M.J., Michoud K., O'Donovan C., Phan I., Pilbout S., Schneider M. (2003)*The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003.* Nucleic Acids Res. 31:365-370.

22. Bork, P. & Gibson, T. J. (1996) Applying motif and profile searches. *Methods in Enzymol.* 266, 162-183.

23. Borodovsky, M., Rudd, K.E. and Koonin, E.V. (1994) Intrinsic and extrinsic approaches for detecting genes in a bacterial genome. Nucleic Acids Res., 22, 4756-4767.

24. Burge, C. and Karlin, S. (1997) Prediction of complete gene structures in human genomic DNA. J. Mol. Biol. 268, 78-94.

25. Chung, S. Y., and Subbiah, S. (1996). A structural explanation for the twilight zone of protein sequence homology. Structure, 4(10): 1123-1127

26. Clamp, M and Cuff. J (1999) JalView and Jpred - Multiple Sequence Visualisation and Analysis (Talk), Wednesday, 14 April 1999. http://industry.ebi.ac.uk/~alan/VisWorkshop99/agenda.html

27. Claverie, J.M., Poirot, O. and Lopez, F. (1997) The difficulty of identifying genes in anonymous vertebrate sequences Comput. Chem., 21, 203-214

28. Corpet, F., Servant,F., Gouzy,J. and Kahn,D. (2000) ProDom and ProDom-CG: tools for protein domain analysis and whole genome comparisons. *Nucleic Acids Res.*, 28, 267–269.

29. Cuff, J. A., Clamp, M. E. & Barton,G. J. (1998) Jpred: a consensus secondary structure prediction server. *Bioinformatics* 14, 892-893.

30. Dassa, E. and Bouige, P. (2001). The ABC of ABCS: a phylogenetic and functional classification of ABC systems in living organisms. Res. Microbiol. 152, 211-29.

31. Dayhoff, M. O., Schwartz, R. M., and Orcutt, B. C.(1978). A model of evolutionary changes in proteins. In Dayhoff, M. O., editor, Atlas of Protein Sequence and Structure, Vol. 5, Suppl. 3, pages 345-352. National Biomedical Research Foundation, Washington D.C.

32. Eddy, S. R. (1998) Profile hidden Markov models. *Bioinformatics*, 14, 755–763.

33. Eddy, S. R. (1998b). Multiple-alignment and sequence searches. Trends Guide to Bioinformatics, pp. 15-18

34. Eddy, S.R. (1996) Hidden Markov models. Current Opinion in Structural Biology 6, 361-365.

35. Falquet, L., Pagni M., Bucher P., Hulo N., Sigrist C.J.A., Hofmann K. and Bairoch A. (2002) The PROSITE database, its status in 2002. *Nucleic Acids Res.*, 30, 235–238.

36. Felsenstein, J. 1989. *Cladistics* 5, 164-166.

37. Frishman, D. & Argos, P. (1997). Seventy-five percent accuracy in protein secondary structure prediction. Proteins, 27, 329-335.

38. Fuellen, G., Wagele. J.W., Giegerich, R. (2001) Minimum conflict: a divide-and-conquer approach to phylogeny estimation. Bioinformatics. 17(12): 1168-78

39. Grundy, W. N. (1998) Homology Detection via Family Pairwise Search. *Journal of Computational Biology* 5, 479-492.

40. Grundy, W. N. (1998b) A Bayesian Approach to Motif-based Protein Modeling. A PhD Thesis submitted to the University of California, San Diego.

41. Gujarati, D. (1988) Basic Econometrics. McGraw-Hill, New York.

42. Haft, D.H., Loftus, B.J., Richardson, D.L., Yang, F., Eisen, J.A., Paulsen, I.T. and White, O. (2001) TIGRFAMs: a protein family resource for the functional identification of proteins. *Nucleic Acids Res.*, 29, 41–43.

43. Henikoff, S. and Henikoff, J. G. (1991). Automated assembly of protein blocks for database searching. Nucleic Acids Research, 19(23):6565{6572.

44. Higgins D., Thompson J., Gibson T.Thompson J.D., Higgins D.G., Gibson T.J.(1994). CLUSTAL W: improving the sensitivity of progressivemultiple sequence alignment through sequence weighting,position-specific gap penalties and weight matrix choice. Nucleic Acids Res. 22:4673-4680.

45. Höng J.C., Ivanov N.V., Hodor P., Xia M., Wei N., Blevins R., Gerhold D., Borodovsky M., & Y. Liu (2004) Identification of new human cadherin genes using a combination of protein motif search and gene finding methods J Mol. Biol, 337, 307-317

46. Holmes, I. (2000). Review of sequence homology search techniques on the World-Wide Web. HIV Sequence Compendium.

47. Hoon S, Ratnapu KK, Chia JM, Kumarasamy B, Juguang X, Clamp M, Stabenau A, Potter S, Clarke L, Stupka E. (2003) Biopipe: a flexible framework for protocol-based bioinformatics analysis. Genome Res. 2003 Aug;13(8):1904-15

48. Hudak, J. & McClure, M. A. (1999). A Comparative Analysis of Computational Motif-detection Methods. *Pacific Symposium on Biocomputing* 4, 138-149.

49. Huson, D.H., Rice, K., Nettles, S., Warnow, T.J. and Yooseph, S. Hybrid tree construction methods. J. Experimental Algorithms, 4, art. 5, 2000.

50. Hyde, S.C., Emsley, P., Hartshorn, M.J., Mimmack, M.M., Gileadi, U., Pearce, S.R., Gallagher, M.P., Gill, D.R., Hubbard, R.E., and Higgins, C.F. 1990. Structural model of ATP-binding proteins associated with cystic fibrosis, multidrug resistance and bacterial transport. Nature 346: 362-365

51. Jonassen, I. (1997). Efficient discovery of conserved patterns using a pattern graph. *Comput Appl Biosci.* 13, 509-522.

52. Jonassen, I., Collins, J. F., & Higgins, D. G. (1995) Finding flexible patterns in unaligned protein sequences. *Protein Science* 4, 1587-95.

53. Kanamori T, Takakura K, Mandai M, Kariya M, Fukuhara K, Sakaguchi M, Huh NH, Saito K, Sakurai T, Fujita J, Fujii S (2004) Increased expression of calcium-binding protein S100 in human uterine smooth muscle tumours. Mol Hum Reprod. 2004 Oct;10(10):735-42.

54. Katoh, K., Misawa, K., Kuma, K, and Miyata, T. (2002) MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Res. 30: 3059-3066.

55. King, R. D. & Sternberg, M. J. E. (1996). Identification and application of the concepts important for accurate and reliable protein secondary structure prediction. Prot. Sci. 5, 2298-2310.

56. Koonin, Eugene V. and Galperin, Michael Y (2003) Sequence - Evolution – Function: Computational Approaches in Comparative Genomics.Norwell (MA): Kluwer Academic Publishers; 2003.

57. Korf, I., Yandell, Mark., Bedell, J. (2003) Chapter 4: Sequence Similarity. Appeared in the book "BLAST". ISBN: 0-596-00299-8.

58. Kremmidiotis, G., Baker, E., Crawford, J., Eyre, H. J., Nahmias, J. & Callen, D. F. (1998) Localization of human cadherin genes to chromosome regions exhibiting cancer-related loss of heterozygosity. Genomics, 49, 467–471.

59. Letunic, I., Goodstadt, L., Dickens, N.J., Doerks, T., Schultz, J., Mott, R., Ciccarelli, F., Copley, R.R., Ponting, C.P. and Bork, P. (2002) Recent improvements to the SMART domain-based sequence annotation resource. *Nucleic Acids Res.*, 30, 242–244.

60. Liao, C., and Noble, W. S. (2002) Combining pairwise sequence similarity and support vector machines for remote protein homology detection. *Proceedings of RECOM*B.

61. Lindahl E. and Elofsson A. (2000) Identification of related proteins on family, superfamily and fold level. J Mol Biol. 295(3):613-25.

62. Liu, Y.T., Liang, T.Y., Huang, C.T., Shieh, C.K. (2003). Memory Resource Considerations in the Load Balancing of Software DSM Systems. Proceedings of the 2003 ICPP Workshops on CRTPC, p.71-78

63. Lorkowski, S. (2004). Personal communication.

64. Lundström, J., Rychlewski, L., Bujnicki, J., Elofsson, A. (2001) Pcons: A neural-network–based consensus predictor that improves fold recognition. *Protein Sci.* 10, 2354-62.

65. Marenholz I, Heizmann CW, Fritz G. (2004) S100 proteins in mouse and man: from evolution to function and pathology (including an update of the nomenclature). Biochem Biophys Res Commun. Oct 1;322(4):1111-22.

66. Mathe C, Sagot MF, Schiex T, Rouze P: Current methods of gene prediction, their strengths and weaknesses. Nucleic Acids Res 2002, 30:4103-4117.

67. Morgenstern, B (1999). DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. Bioinformatics 15, 211 - 218.

68. Murzin, A., Brenner, S.E., Hubbard, T., Chothia, C,. (1995) SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.* 247, 536-540.

69. Pagni M, Jongeneel CV. (2001) Making sense of score statistics for sequence alignments. Brief Bioinform. 2(1):51-67.

70. Pandit, S.B., Gosar, D., Abhiman, S., Sujatha, S., Dixit, S. S., Mhatre, N. S., Sowdhamini, R., Srinivasan. N. (2002) SUPFAM--a database of potential protein superfamily relationships derived by comparing sequence-based and structure-based families: implications for structural genomics and function annotation in genomes. Nucleic Acids Res. 2002 Jan 1;30(1):289-93.

71. Pearson W. R. (1990) Rapid and Sensitive Sequence Comparison with FASTP and FASTA. Methods in Enzymology 183:63 – 98.

72. Qian B and Goldstein, RA (2003) Detecting distant homologs using phylogenetic tree based HMMs. Proteins 52, 446-454

73. Quentin, Y. and Fichant, G. (2000). ABCdb: an ABC transporter database. J. Mol. Microbiol. Biotechnol. 2, 501-4.

74. Rehmsmeier, M. (2002) Database searching with phylogenetic trees. A PhD Thesis submitted to Faculty of Technology, University of Bielefeld, Germany.

75. Rehmsmeier, M. (2002b) Phase4: Automatic evaluation of database search methods. *Briefings in Bioinformatics* 3, 342-352.

76. Rehmsmeier, M., & Vingron, M. (2001) Phylogenetic Information Improves Homology Detection. *Proteins: Structure, Function, and Genetics* 45, 360-71.

77. Rost, B. & Sander, C. (1993). Prediction of protein secondary structure at better than 70% accuracy. J. Mol. Biol. 232, 584-599.

78. Rouzé, P., Pavy, N. and Rombauts, S. (1999) Genome annotation: which tools do we have for it? Curr. Opin. Plant Biol., 2, 90-95.

79. Sakharkar MK, Chow VT, Kangueane P. (2004) Distributions of exons and introns in the human genome. In Silico Biol. 2004 Jun 16;4(2):0032

80. Salamov, A. A. & Solovyev, V. V. (1995). Prediction of protein secondary structure by combining nearest- neighbor algorithms and multiple sequence alignments. J. Mol. Biol. 247, 11-15.

81. Schmitt L, Tampe R (2002) Structure and mechanism of ABC transporters. Curr Opin Struct Biol. 2002 Dec;12(6):754-60.

82. Silverstein, KA., Shoop, E., Johnson. J.E., Kilian, A., Freeman, J.L., Kunau, T.M., Awad, I.A., Mayer, M., & Retzel, E.F. (2001) The MetaFam Server: a comprehensive protein family resource. *Nucleic Acids Res.* 29, 49-51.

83. Spang R, Rehmsmeier M, Stoye J (2002) A novel approach to remote homology detection: jumping alignments. J Comput Biol. 2002;9(5):747-60

84. Spitzer, M (2004b). Personal communication.

85. Spitzer, M., Fuellen, G., Cullen, P., and Lorkowski, S. (2004) VisCoSe: visualization and comparison of consensus sequences. Bioinformatics 2004 20: 433-435

86. Stajich JE, Block D, Boulez K, Brenner SE, Chervitz SA, Dagdigian C, Fuellen G, Gilbert JG, Korf I, Lapp H, Lehvaslaiho H, Matsalla C, Mungall CJ, Osborne BI, Pocock MR, Schattner P, Senger M, Stein LD, Stupka E, Wilkinson MD, Birney E. (2002) The Bioperl toolkit: Perl modules for the life sciences. Genome Res. 2002 Oct;12(10):1611-8

87. Stanke, M. and Swaack, S. (2003). Gene Prediction with a Hidden-Markov Model and a new Intron Submodel. *Bioinformatics*, Vol. 19, Suppl. 2, pages ii215-ii225

88. Stefkova J, Poledne R, Hubacek JA. (2004) ATP-binding cassette (ABC) transporters in human metabolism and diseases. Physiol Res. 2004;53(3):235-43.

89. Watson, J. D., and Crick, F. H. (1953) Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid. Nature; 171(4356): 737-8.

90. Wenzel JJ, Kaminski WE, Piehler A, Heimerl S, Langmann T, Schmitz G (2003) ABCA10, a novel cholesterol-regulated ABCA6-like ABC transporter. Biochem Biophys Res Commun. 2003 Jul 11; 306(4): 1089-98

91. Wu, Q. & Maniatis, T. (1999). A striking organization of a large family of human neural cadherin-like cell adhesion genes. Cell, 97, 779–790.

92. Yagi, T. & Takeichi, M. (2000). Cadherin superfamily genes: functions, genomic organization, and neuro-logic diversity. Genes Dev. 14, 1169–1180.

93. Yona, G., Linial, N. & Linial, M. (2000) Nucleic Acids Research 28, 49-55.

94. Zdobnov E.M. and Apweiler R. "InterProScan - an integration platform for the signature-recognition methods in InterPro" Bioinformatics, 2001, 17(9): p. 847-8.

95. Zhang, Z., Schaffer, A. A., Miller, W., Madden, T. L., Lipman, D. J., Koonin, E. V. & Altschul, S. F. (1998) Protein sequence similarity searches using patterns as seeds. *Nucleic Acids Res.* 26, 3986-90.

96. Zvelebil, M., Barton, G., Taylor, W. & Sternberg, M. (1987). Prediction of protein secondary structure and active sites using the alignment of homologous sequences. J. Mol. Biol. 195, 957-961.