COMPARATIVE GENOMICS VISUALIZATION

by

YANQI SU

(Under the Direction of Eileen T. Kraemer)

ABSTRACT

As an increasing number of genomes are sequenced, comparative genomics analysis at different evolutionary distances plays a crucial role in decoding genomic information and discovering the similarities and differences between the genomes. It consists of three process steps: data preparation, sequence alignment, and visualization. In this thesis the major alignment methods and visualization tools are discussed. Based on an analysis of current work, we have designed and implemented a comparison visualization approach that satisfies the specific requirements of the CryptoDB project (http://cryptodb.org) (45). Our approach is written in Perl and based on the GBrowse framework. In addition, this study also presents an analysis of the comparison of *C. hominis* and *C. parvum* to find candidate insertions, deletions, and synteny blocks. Future work is also discussed.

INDEX WORDS: comparative genomic analysis, evolutionary distance, CryptoDB, GBrowse, Perl, insertion and deletion, synteny block, rearrangement.

COMPARATIVE GENOMICS VISUALIZATION

by

YANQI SU

B. S., Tongji University, Shanghai, China, 2002

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2005

COMPARATIVE GENOMICS VISUALIZATION

by

YANQI SU

Major Professor: Eileen T. Kraemer

Committee: John A. Miller
Jessica C. Kissinger

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2005

ACKNOWLEDGEMENTS

I would like to thank my major professor Dr. Eileen T. Kraemer for her patience in guiding me, encouraging me, and supporting me throughout this study. This project can not be done without her invaluable advice on the design and implementation.

My special thanks go to John A. Miller and Jessica C. Kissinger for being my committee members and for opening up new fields of knowledge to me.

I am very grateful to Dr. Kissinger's lab, especially Mr. Haiming Wang for his invaluable help.

TABLE OF CONTENTS

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

1.1 Motivation

The availability of DNA sequences has grown exponentially. However, knowledge of the primary DNA sequence does not directly provide the desired information about function, and we are now faced with the challenge of decoding the information within these DNA sequences. Comparative genomics analysis of the sequence from multiple species at different evolutionary distances allows us to discover the similarities and differences between the genomes and is thus an important tool in addressing this challenge.

The assumption that underlies all comparative genomics is that the two genomes had a common ancestor and, therefore, that every base pair in each organism can be explained as the combination of this original ancestral genome and the action of evolution (60).

The purpose of comparative genomics is to gain a better understanding of how species have evolved. By comparing the genomic sequences of different species, the following features can be identified and determined: sequence similarity, gene location, the length and number of coding regions within genes, the amount of noncoding DNA in each genome, and highly conserved regions maintained in organisms (23).

Different questions can be addressed by comparing genomes at different phylogenetic distances (Figure1) (31). Here we define three kinds of phylogenetic distance:

- Long distance: larger than 1 billion years of evolution since separation;
- Moderate distance: about 70 – 100 million years of evolution since separation;

- Short distance: about 5 million years of evolution since separation;

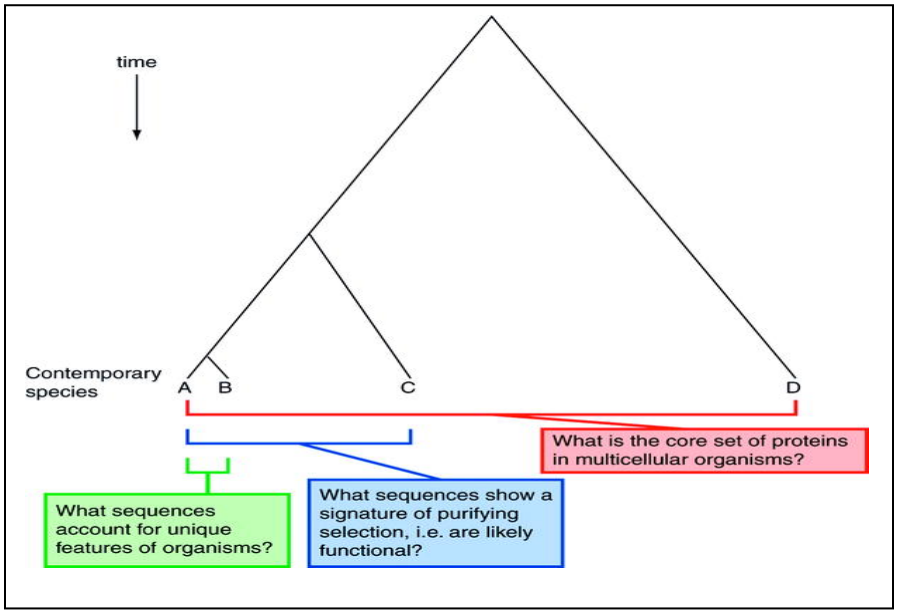


Figure 1. Genomes comparisons at different phylogenetic distances are able to address different questions. A generalized phylogenetic tree is shown with four different organisms: A, B, C, and D. A and D are the most distantly related pairs. Examples of questions addressed by genomes comparisons at the different distances are given in the boxes. (31)

Normally over long distances, the order of genes and the sequences regulating their expression are generally not conserved easily. By the comparison of genomes, the core set of proteins in multi-cellular organisms can be found (31). At moderate phylogenetic distances, both functional and nonfunctional DNA can be found within the conserved DNA. The comparative genomics approach can not only distinguish conserved from divergent, and functional from nonfunctional DNA, but also identify the general functional class of certain DNA segments, such as coding exons and some gene regulatory regions (31). At short distances, sequence data is highly conserved. Thus the comparison of similar genomes can help researchers to find the major sequence differences that may explain the differences between the organisms. Thus comparative genomics is a powerful and important approach, and as genomic sequence data accumulate it will become more and more informative.

1.2 Process steps

The comparative genomics approach can be divided into three steps: generation of annotated sequence data, alignment of sequences from two or more species, and visualization of the alignments (Figure 2).

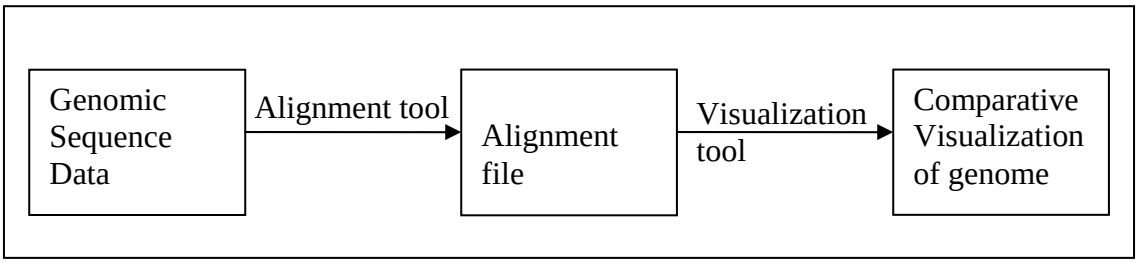


Figure 2. The process flow chart of comparative genomics analysis

Obtaining genomic sequences data for comparative analyses:

DNA sequence generation is a process that determines the exact order of the chemical building blocks (called bases and abbreviated A, T, C, and G) in the genetic material of an organism. It was the greatest technical challenge in the Human Genome Project. After DNA sequencing is performed, the sequences are assembled and submitted into a public sequence database with public web sites and are available to the world. From some of these web database servers, such as the NCBI (National Center for Biotechnology), most of the publicly available DNA sequences for all species can be obtained. Also, some species-specific sequence databases exist such as NCBI (http://www.ncbi.nlm.nih.gov/), TIGR (http://www.tigr.org/), and Sanger (http://www.sanger.ac.uk/), to name a few. Because some genome sequences in those web databases are not annotated, programs are required to predict the locations of putative genes by searching the DNA sequences for common features (23). Some gene annotation and prediction programs exist such as GENSCAN (http://genes.mit.edu/GENSCAN.html). These genes are then often displayed on the resulting visualization.

Alignment of sequences:

Sequence alignments are the core process in the field of comparative genomics. The alignment maps the nucleotides in one sequence onto those in the other sequence, with gaps either introduced or not. Introduction of gaps into sequence alignments allows the alignment to be extended into regions where one sequence may have lost or gained sequence characters not found in the other.

Several types of alignments exist:

- **local alignments** that identify local similarities between regions of each sequence;
- **global alignments** over the entire length of each DNA (nucleic acid) or protein (amino acid) sequence;
- **glocal alignments** that are a combination of the methods of global and local alignment;
- **pair-wise alignments** that find the best matching local or global alignments of two DNA or protein sequences;
- **multiple alignments** that are an extension of pair-wise alignment to incorporate more than two genomic sequences into an alignment;

When genomic sequence consists of several unordered regions, or when genomic sequence rearrangements are frequent, local alignments are more suitable to find the true conservation than global alignments. Global alignments are useful for the closely-related sequences, but these closely-related sequences are also easily identified by local alignment methods. If the genomic sequences are not known to be homologous over their entire length, a local alignment should be better. Sometimes these two methods will give similar answers, but if the homology is distant, a local alignment will be more likely to find the regions of homology (7).

So the local alignment is more flexible with evolutionary changes than global alignment. Here the comparison of local and global alignment algorithm strategy will be given (Figure 3).

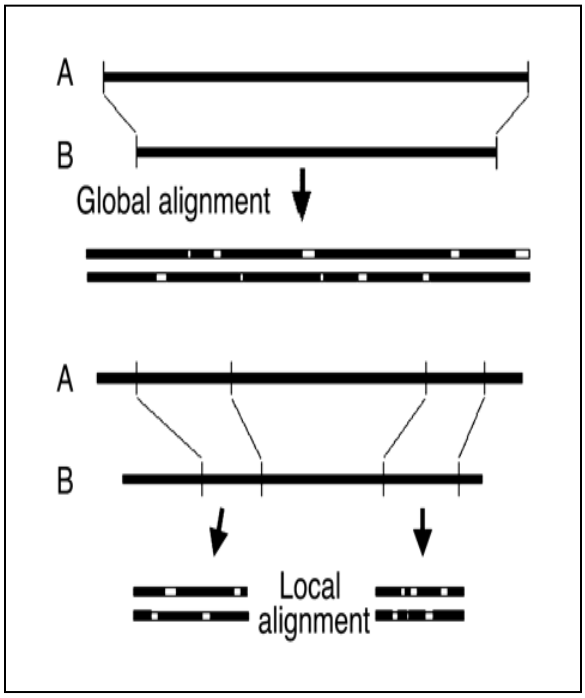


Figure 3. Comparison of local and global alignment algorithm, Top: Global alignments are generated with two DNA sequences (A and B) and an optimal similarity score is obtained over the entire length of the two sequences. Bottom: Local alignments are produced with two DNA sequences (A and B) and optimal similarity scores are obtained over many sub-regions along these two sequences. The local-alignment algorithm works by first finding very short common segments, and then expanding out the matching regions as far as possible. (44)

With an increasing of the number of genomic sequences, the pair-wise alignments may have difficulty providing an overall picture of conservation, but the multiple alignments can help in the identification of common regions between the sequences and provide a much clearer picture of genome evolution, and will become more important as the number of available genomes increases. Multiple sequence alignment is computationally difficult and is classified as an NP-hard problem (20). The results of experiments show that multiple alignments are better than pair-wise alignments at aligning conserved exons between distant species (23). For example,

the multiple alignment among the *human*, *mouse*, and *fugu* genomes may obtain a more accurate result than the pair-wise alignment between the *human* and *fugu* genomes (10).

Local alignment methods, such as Smith-Waterman (54), BLAST (1, 2), BLASTZ (53), SSAHA (39) and so on, can align the sequences only in certain regions and find locations of rearrangements between two sequences. For example, the positions 25-50 of sequence A may be aligned with the positions 85-110 of sequence B. The most famous local alignment method is Smith-Waterman (54). This kind of alignment may be thought of as minimizing the evolutionary distance or maximizing the similarity between the two compared sequences (1).

Global alignment tools include Needleman-Wunsch (38), DIALIGN (37), MUMmer (14-16), GLASS (4), WABA (34), and AVID (7). These tools will align the sequences across their entire length and can produce a more accurate alignment at the base-pair level than local alignment tools when the features are in the same order. The best-known global alignment algorithm is Needleman-Wunsch (38), but unfortunately this algorithm is too time-consuming for comparing long genomic sequences (10). Most of these methods have been proven effective in aligning genomic sequences from two closely related organisms, such as *human* and *mouse* or *Caenorhabditis elegans* and C.*briggsae*, but have not been tested in alignments between distant relatives such as *human* and *fugu* (10).

Glocal alignment is a new notion of sequence alignment that combines global and local alignment methods and can handle rearrangements between the sequences. The Shuffle-LAGAN (SLAGAN) algorithm (13), a kind of *glocal* alignment method capable of quickly aligning long genomic sequences, will be introduced in the following chapters. It is based on the CHAOS (12) local alignment algorithm and the LAGAN (10) global alignment algorithm.

The availability of multiple genomic sequences helps scientists to learn more about DNA mutations during evolution (13). For example, the multiple alignment of several homologous sequences can detect similarity across large evolutionary distances, which might reveal some important biological features (27, 6, 33, 59). In recent years, a number of computer programs have been developed for the alignment of multiple sequences including CLUSTALW (58), MULTALIGN (3), MULTAL (57), YAMA (29, 30), PRRP (26), DIALIGN (37), MAVID (8) and MLAGAN (10). Multiple alignments are more difficult to compute than pair-wise alignments because most of the multiple alignment programs use a progressive method of successive applications of a pair-wise alignment algorithm. These methods can effectively align proteins and relatively short genomic regions, but are not efficient enough to align entire genomes.

Visualization of the alignment:

Visualization of complex alignment data is very important for computational biologists. The interactive graphics and visual presentations are effective methods to interpret large quantities of scientific data. Currently several comparative genomic visualization tools exist: VISTA (Visualization Tool for Alignment) (36, 24), Phylo-VISTA (52), SynBrowse (43), Mauve (16), Mulan (41), PipMaker and MultiPipMaker (39, 40), BSR (46), ABC (Application for Browsing Constraints) (15), ACT (14), Alfresco (32) and so on. The capabilities of these tools are discussed in detail in chapter 3. Several categories of these browsers for the pre-computed genome alignment can be described:

- **Static images browser**: static images will be generated according to the pre-computed alignment.

- **Dynamic and interactive browser**: visual presentation will be changed at varying levels of resolution and with the different parameters.
- **Web-based alignment browser**: sequence data can be loaded into the web server that will provide the outcome the visualization of comparative genomics with static images or dynamic and interactive images;
- **Stand-alone browser**: download and run the program locally, thus eliminating any limitation of networking.

For example, the VISTA browser generates static images that are not interactive; the phylo-VISTA and PipMaker browsers require the use of a particular alignment program; the ABC browser allows many annotation types and colors with the stand-alone version but it is not suited for genome-wide visualization; the Mauve browser is quite useful for browsing large scale genome with genomic rearrangements.

1.3 Implementation

In this thesis work, a comparative genomics visualization for the CryptoDB project (http://cryptodb.org) (45) will be introduced and implemented according to the above described three process steps. See chapter 5 for detailed discussion of the design and implementation.

- The sequence data used by the CryptoDB project consists of the *C. hominis* and *C. parvum* genomes. The *C. hominis* genome has 18 long contigs [(AAEE01000001…AAEE01000018)] and the *C. parvum* genome has 1422 short contigs [(AAEL01000001…AAEL01001422)]. Currently, both DNA and protein sequence data can be obtained from the NCBI GenBank database.
- Two phases are required for these two particular genomes. The first phase involves

contig level comparison by the WU-BLASTN alignment method (25). At this phase, one of the long *C. hominis* genome contigs will be set as a reference contig, and by WU-BLASTN alignment, many short *C. parvum* genome contigs will be compared to this *C. hominis* contig. The second step uses the OrthoMCL program (http://orthomcl.cbil.upenn.edu/) (48) to identify the orthologous groups. These analysis programs provide the mappings between features of the two genomes and serve as the basis for the visualization. The WU-BLASTN results provide genomic synteny information. The OrthoMCL program provides information about orthologous genes and their positions in the genome.

- ✧ Three steps of the visualization component are implemented in this project. Firstly, the annotated gene features are added onto each genome contig; secondly, according to the ortholog gene group, a comparison shadow is drawn; finally, on these comparison shadows, an image map is generated with a popup window and detailed text alignments are shown in this window: ClustalW, BlastP and Promer diagrams.

In this project, the genome browser used is GBrowse (http://www.gmod.org/) (56), a web-based application for displaying genomic annotations and other features. Although several optional visualization tools exist with very useful and powerful functions, none of them can be quite integrated into our framework. Thus, we developed our own unique and novel comparative genomics visualization approach.

CHAPTER 2

REVIEW OF ALIGNMENT APPROACHES

A genomic alignment is a mapping of one DNA sequence onto another evolutionarily related DNA sequence in order to identify regions that have been conserved. Several major genomic local and global alignment approaches exist, including LAGAN and Multi-LAGAN (10), Shuffle-LAGAN (13), AVID (7) and MAVID (8), MUMmer (17, 18, 35), CHAOS/DIALIGN (12), and Mauve (16).

In the following sub-sections, we present each of these approaches, providing some detail on the approach, its assumptions, and the algorithm employed.

2.1 LAGAN and Multi-LAGAN (MLAGAN)

Introduction:

LAGAN (Limited Area Global Alignment of Nucleotides) is a kind of global alignment method that can rapidly align two large-scale homologous genomic sequences. The key idea behind the approach is the anchoring technique that can reduce the computation time. This method was tested on 12 vertebrate species with greater than 12 Mb sequence and it was found that it can produce more accurate alignments than other leading global alignment methods such as DIALIGN (37), MUMmer (17, 18, 35), GLASS (4), and AVID (7), especially between distant homologous organisms such as *human* and *chicken*, or *human* and *fugu*. (10).

Multi-LAGAN is a kind of global alignment method that aligns multiple genomic DNA sequences, based on progressive alignment with LAGAN. It is a practical method to generate

multiple alignments of long genomic sequences at any evolutionary distance (10). The source code for LAGAN and Multi-LAGAN is available from the authors (http://www.cs.toronto.edu/~brudno/).

Assumption:

LAGAN and MLAGAN assume the orthologous regions between two species have already been identified and that no genomic rearrangements exist. In addition, Multi-LAGAN assumes that the phylogenetic tree is given (10).

Outline of algorithm:

The LAGAN algorithm has three main steps: (Figure 4) (10):

- **Generation of local alignments between the two sequences.** At this step, LAGAN finds all the local alignments using the CHAOS algorithm (12), a highly sensitive method that detects local alignments and relies on multiple short inexact words instead of longer exact words. It then assigns a weight to each local alignment according the score of each alignment (Figure 4B).
- **Construction of a rough global map by chaining an ordered subset of the local alignments (Figure 4C).** At this step, two local alignments will be chained if the end of one precedes the start of the other. Many chains are generated, and weights are computed. Each local alignment in this chain is called an anchor. The highest-weight chains are reported.
- **Computation of the final global alignment (Figure 4D).** The rectangle boxes in Figure 4D are found between each consecutive pair of the anchors. Each box includes the area between the end position of first anchor and the begin position of second anchor. Then, the global alignment can be computed within these rectangle

boxes areas by the Needleman-Wunsch-like dynamic programming.

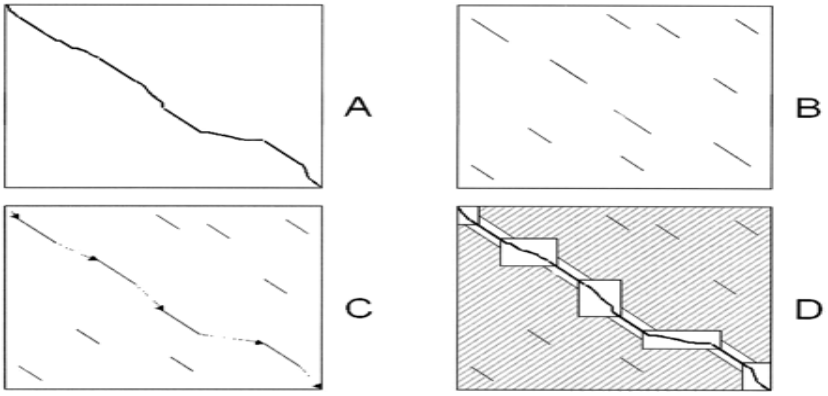


Figure 4. The LAGAN algorithm. (A) A global alignment between two sequences is a path from the top-left to the bottom-right corner. (B) The first step of LAGAN algorithm is to find all local alignments. (C) The second step of LAGAN algorithm is to generate a global map with the highest-weight chain. (D) The third step of LAGAN algorithm is to find the cell boxes between each anchor of the chain and then compute the optimal Needleman-Wunsch alignment limited to this area. (10)

The MLAGAN alignment algorithm is based on progressive alignment: A multiple *L*-sequence alignment is constructed in *L–1* pair-wise alignment steps and in each step two closest sub-sequences according to the phylogenetic tree are aligned. Two major phases for MLAGAN exist given *L* sequences $X^1, \ldots, X^L$ and a phylogenetic tree.

- The progressive alignment phase. At this phase the multiple alignment by successively aligning the two closest sequences according to the phylogenetic tree. The LAGAN algorithm is used to align the pair of sequences;
- An iterative improvement phase. During this phase, the algorithm successively removes each sequence from the multiple alignment and re-aligns it to the consensus alignment until no significant improvements occurs.

Input and output files:

The input to LAGAN consists of two sequences files, in FASTA format.

The output of LAGAN should be three files: an .alignment file, a .fasta.masked file, and a .fasta.out.

The input to Multi-LAGAN consists of several sequences files, in FASTA format, and the phylogenetic tree that relates all of these sequences..

As with LAGAN, the output of Multi-LAGAN is three files: an .alignment file, a .fasta.masked file and a .fasta.out file.

Visualization:

The alignment result produced by LAGAN or MLAGAN can be visualized by VISTA (33, 34) or Phylo-VISTA tools (52). See section 4.1 for detailed discussion of the visualization.

Evaluation:

Compared with several leading global and local alignment methods such as DIALIGN (37), MUMmer (17, 18, 35), GLASS (4), AVID (7) and BLASTZ (53) in orthologous sequences, LAGAN, MLAGAN and BLASTZ were the most accurate at aligning close homolog sequences, and LAGAN and MLAGAN also were the best at aligning distant homolog sequences. Although MLAGAN was slower than the other aligners, it was more accurate for the multiple sequence alignment (10).

Most of the global alignment algorithms such as DIALIGN, MUMmer, GLASS, WABA (34), and AVID were designed for highly similar organisms sequences, such as *human* and *mouse*, but have not been proven efficient between two distant organisms such as *human* and *fugu*. However, LAGAN can work well between both distant and close organisms because it uses the CHAOS method to detect the local alignments by multiple short inexact words instead of longer exact words (10).

The results of LAGAN and MLAGAN show that multiple genomic sequence alignments are better than pair-wise genomic sequences alignments in distant species (10). Thus, a multiple global alignment of the *human*, *mouse*, and *fugu* genomes may be better to obtain accurate alignments than a pair-wise global alignment of the *human* and *fugu* genomes (10).

2.2 Shuffle-LAGAN (SLAGAN)

Introduction:

Global alignment algorithms such as MUMmer, AVID, and LAGAN, can produce more accurate alignments than local alignment algorithms in closely related genomic sequences with the same order of features, but they do not handle rearrangement events because they require that the map between two sequences must be monotonically increased in both sequences (13). Local alignment algorithms such as BLAST, CHAOS, and BLASTZ are able find the locations of rearrangement between genomic sequences, but they can not show how these two sequences evolved from their common ancestor. Thus, a new notion of alignment method known as *glocal* alignment, a combination of global and local methods, has been introduced (13). Such *glocal* alignment algorithms create an alignment map that transforms one sequence into the other and allows for rearrangement events.

SLAGAN (13) is an example of a *glocal* alignment algorithm and is based on the CHAOS (12) local alignment algorithm and the LAGAN (10) global alignment algorithm, and is able to quickly align long genomic sequences of length greater than 1Mbp in the presence of rearrangements. It was the first genome alignment method that permitted genome rearrangements (13) and has been used to align the *mouse* and the *human* genomes. The source code is available at http://lagan.stanford.edu/glocal.

Outline of algorithm:

The SLAGAN algorithm consists of three distinct stages (Figure 4) (13).

- First, local alignments between two sequences are generated using the CHAOS local alignment algorithm. See Figure 5A. CHAOS works by chaining together short words matched between two sequences and then scores each chain by the sum of each anchor's score in the chain.
- Second, a "1-monotonic conservation map" is built. See Figure 5B. Most global alignment algorithms began with a local alignment algorithm and then build a rough global map, which must be non-decreasing in both sequences. In order to satisfy the rearrangement events, the global map should be non-decreasing in only one sequence. This kind of global map is called "1-monotonic conservation map".
- Third, the maximal consistent sub-segments are aligned. Computing the maximal consistent sub-segments is very important. After the "1-monotonic conservation map" is produced, generating the maximal consistent sub-segments in this map is straight-forward (shown as the dashed boxes of Figure 5C), by sorting all the local alignments in the map: set the first alignment as the start of a consistent sub-segment and then add other local alignments if they are consistent. Once an alignment is inconsistent with the current sub-segment, the new sub-segment will be generated. Two local alignments are consistent if they can both be a part of a global alignment (13). After producing these maximal consistent sub-segments, the global alignment algorithm such as LAGAN is used to align these maximal consistent sub-segments.

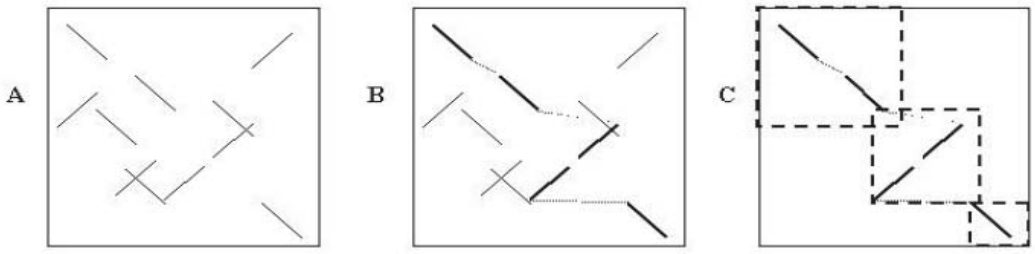


Figure 5. Outline of SLAGAN Algorithm: (A) the local alignment between the two sequences is generated by CHAOS local alignment method. (B) The "1-monotonic conservation map" with bold line is found. (C) The maximal consistent sub-segments of the 1-monotonic map (dashed boxes) are found and then aligned by LAGAN. (13)

Input and output files:

The inputs to SLAGAN consist of two genomic sequences file, in FASTA format, and two annotation files.

The outputs of SLAGAN are the alignments formats, the MFA (multi-FASTA alignment) formats, and a list of conserved regions from the alignment (CNS) formats. Actually the alignments files are the same as MFA file in content, but different in format.

Visualization:

The alignment result produced by SLAGAN can be visualized by VISTA (33, 34) or Dotplots (11).

Evaluation:

SLAGAN was tested by aligning the human and mouse genomes using the whole genome alignment technique used in the Berkeley Genome Pipeline (13). Compared with the regular LAGAN global alignment, SLAGAN has better sensitivity and similar specificity and compared to the BLASTZ (53) local alignment, SLAGAN shows better specificity at a modest cost in sensitivity (13).

In order to catch rearrangement events, the SLAGAN algorithm relaxes the assumption of global alignment and only allows the global map to be non-decreasing in one sequence, without any restriction in the other sequence. Thus, when one sequence aligns duplicated regions; it will be reported only in one of the sequences. One of the major drawbacks of the SLAGAN algorithm is that it is not symmetric, and will miss duplications (13).

2.3 AVID and Multi-AVID (MAVID)

Introduction:

AVID is a global alignment method designed to be fast, memory efficient and practical for sequence alignments of large genomic regions up to Mbp long. It is sensitive in finding homologous regions, and has been used to align thousands of submitted sequence pairs and also as a key component in an alignment of the entire *human* and *mouse* genomes (7). It uses an efficient data structure called a suffix tree.

MAVID is a multiple alignment program suitable for many large genomic DNA sequences up to Mbp long. It is also integrated with various phylogenetic tree construction programs and visualization tools, and can be used to identify conserved regions in phylogenetically related sequences. It has been used to align *vertebrate* genomic sequences, mitochondrial DNA, viruses, and other sequences (8).

Assumption:

The critical assumption is that the aligned sequences' functional elements should be preserved in order and orientation, in order to reduce false alignments. At the same time, this restricts the possible applications of the program. Certainly this kind of problem can be fixed

using the idea of *glocal* alignment by combing AVID global alignment with some local alignment (7).

Outline of algorithm:

The AVID alignment algorithm consists of three major steps (Figure 6) (7):

- **Repeat Masking.** Unlike standard alignment methods, AVID use both the unmasked and masked sequences that are generated by the RepeatMasker program (http://ftp.genome.washington.edu/RM/RepeatMasker.html, Smit and Green). In comparing the masked sequences and unmasked sequences, two kinds of matches will be produced: repeat matches with overlapping repeats and clean matches without overlapping repeats.
- **Finding Matches Using Suffix Trees**. The maximal repeated substring in one string can be obtained by construction of a suffix tree. Based on this idea, in order to find all maximal matches between two sequences, we can concatenate these two sequences and place a default character N between them and then find the maximal repeated substrings in this single string. If the maximal repeat string crosses the N boundary, it is the maximal match between two sequences
- **Anchor Selection.** After all the matches are found, the recursive process of anchoring and aligning proceeds. An anchor set is a collection of non-overlapping, non-crossing matches. First, short matches that the length is less than half the length of the longest match will be eliminated from the entire match set and can not be considered as anchors. Those short matches will be reconsidered for anchoring in the later round. The remaining matches are ordered by the length. Second, the anchors are selected by the Smith-Waterman-like algorithm. Third, after the selection of

anchors, the regions connecting these anchors remain to be aligned. For instance, if *n* anchors have been selected, *n+1* regions between these anchors need to be aligned.

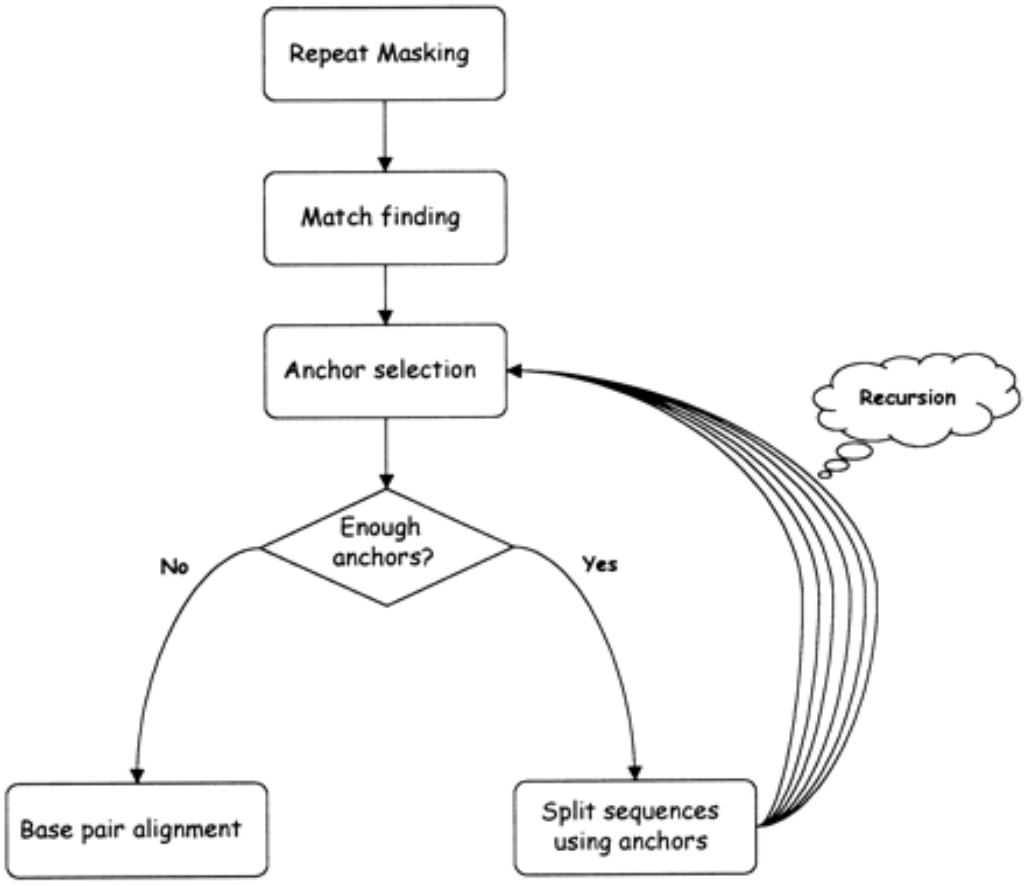


Figure 6. The AVID algorithm (7).

Input and output files:

The input to AVID consists of two sequences files, in FASTA format. The input to Multi-AVID consists of several sequences files, in Multi-FASTA format.

After running the web server, the following output file can be obtained: an .aln file in alignment format consists of detailed alignment information for two sequences; an .mfa alignment file in multi-FASTA format consists of detailed alignment information for multiple sequences; and a .phy file in PHYLIP format consists of the relationship of multiple sequences.

Visualization:

AVID and MAVID integrate well with the VISTA (33, 34) and Poly-VISTA (52) visualization tools.

Evaluation:

AVID has been successfully used for the alignment of closely related species such as *human* and *primates*, but has bad results for the alignment of more distant organisms such as *human* and *fugu*. MAVID is currently a DNA sequence alignment program and cannot align protein sequences (8). An experiment has been done to compare the human genome with *cat*, *chicken*, *cow*, *dog*, *pig*, and *rat* using several local and global alignments such as AVID, BLASTZ, CHAOS, GLASS, and MUMmer (8). AVID has worse results on the *human* and *chicken* alignment compared with other organisms because on the DNA level the similarity between *human* and *chicken* is lower than between *human* and other organisms (7). Compared with MUMmmer and GLASS global alignments, AVID is fastest.

2.4 MUMmer

Introduction:

MUMmer is a kind of global pair-wise alignment of very large scale DNA sequences up to hundreds of Mbp. The latest version of MUMmer is 3.0. MUMmer uses the efficient suffix tree data structure. As a result it was the first system that could rapidly perform genome comparisons of this large scale (17). Normally sequence alignment algorithms use dynamic programming or hashing. In contrast, the suffix tree structure used by MUMmer directly finds maximal unique matches. Then these matches can easily be ordered to align two genomes. This

algorithm assumes that genomic sequences are closely related, thus it can perform large scale alignments quickly and precisely (17).

The MUMmer release consists of the five most common components: mummer, NUCmer, PROmer, run-mummer1, and run-mummer3.

- ✧ mummer efficiently generates lists of maximal unique matches between the two sequences by the suffix tree data structure.
- ✧ NUCmer is a Perl script pipeline to compare multiple closely related sequences. There are three steps: (1) finding maximal exact matches; (2) clustering these matches to form larger inexact alignment regions; (3) extending alignments from each of the matches. In order to increase NUCmer's accuracy, the input sequences need to be processed by the RepeatMasker program to avoid the alignment of uninteresting sequences.
- ✧ PROmer is also a Perl script pipeline to align multiple divergent sequences. It works like NUCmer. It is also recommended to mask the input sequences to avoid the alignment of uninteresting sequences.
- ✧ run-mummer1 and run-mummer3 are cshell script pipelines to compare two sequences. They follow the same three steps as NUCmer and PROmer; however they handle any input sequence, not just nucleotide sequence.

Assumption:

This algorithm assumes that the sequences are closely related. Using this assumption it can quickly compare sequences that are millions of nucleotides in length.

Outline of algorithm:

MUMmer uses a combination of ideas from three approaches: suffix trees, longest increasing subsequences (LIS) (28), and the Smith-Waterman algorithm. The basis of the algorithm is the suffix tree data structure. The detailed alignment process consists of four major steps (17):

- **Perform a maximal unique match (MUM) in the two genomes.** A MUM is a maximal subsequence that occurs only once in one sequence and once in the other. If a perfect matching subsequence occurs only once in each sequence and is significantly long, it will quite likely be part of the global alignment because two genome sequences are assumed as highly similarity. Thus the global alignment can be built based on the MUMs. The assumption of highly similarity between two sequences ensures that numerous MUMs can be found.
- **Sort these matches, and find the longest possible set of matches with the same order in both sequences.** In this step, the LIS algorithm will be used to find the longest set of MUMs. See Figure 7 for the example of sorting principle.
- **Remove the gaps in the alignment from the MUMs sorted at the second step.** A gap is defined as an interruption between the sorted matches. There are four categories: (1) an SNP, (2) an insertion, (3) a highly polymorphic region, (4) a repeat (17). See Figure 8. To complete the global alignment, these four types of gaps should be removed after sorting the MUMs.
- **Output the alignment with all the matches by the previous three steps as well as the detailed regions that do not match exactly in the MUM alignment.**

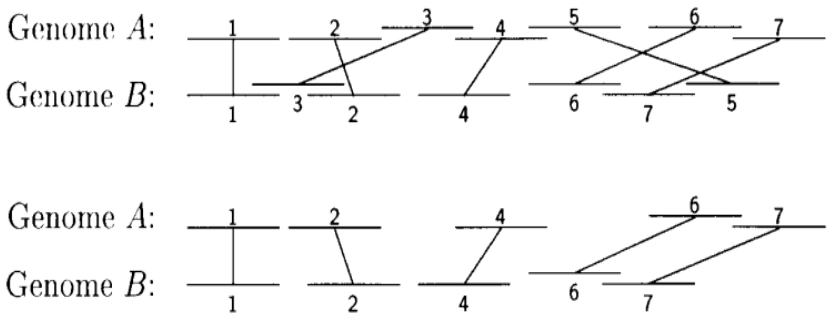


Figure 7. The example of sorting the MUMs by LIS algorithm. Each MUM is indicated only by a unique number. The top alignment shows all the MUMs. The bottom alignment shows all sorted MUMs by LIS algorithm. (17)

1. SNP: exactly one base (indicated by ^) differs between the two sequences. It is surrounded by exact-match sequence.

```
Genome A:  cgtcatgggcgttcgtcgttg
Genome B:  cgtcatgggcattcgtcgttg
                      ^
```

2. Insertion: a sequence that occurs in one genome but not the other.

```
Genome A:  cggggtaaccgc.................cctggtcggg
Genome B:  cggggtaaccgcgttgctcggggtaaccgccctggtcggg
                     ^^^^^^^^^^^^^^^^^^
```

3. Highly polymorphic region: many mutations in a short region.

```
Genome A:  ccgcctcgcctgg.gctggcgcccgctc
Genome B:  ccgcctcgccagttgaccgcgcccgctc
                    ^ ^^ ^^^
```

4. Repeat sequence: the repeat is shown in uppercase. Note that the first copy of the repeat in Genome *B* is imperfect, containing one mismatch to the other three identical copies.

```
Genome A:  cTGGGTGGGACAACGTaaaaaaaaaTGGGTGGGACAACGTc
Genome B:  aTGGGTGGGGCgACGTgggggggggTGGGTGGGACAACGTa
           ^          ^                          ^
```

Figure 8. Four types of gaps in MUM alignment (17).

Input and output files:

The input files should be in Multi-FASTA file format. The output file format varies depending on the command line parameters used. The MUMmer homepage provide the user manual and examples (http://mummer.sourceforge.net/manual/).

Visualization:

In release 3.0, two graphical viewers exist such as MapView and mummerplot. MapView is a visualization tool to display sequence alignments provided by NUCmer and PROmer (http://mummer.sourceforge.net/manual/). Mummerplot is a utility to view the outputs of mummer, NUCmer, and PROmer with a format suitable for plotting based on gnuplot (http://www.gnuplot.info).

Evaluation:

With the availability of whole genome DNA sequences up to hundreds of Mbp size, many alignment algorithms including Needleman-Wunsch and Smith-Waterman can not handle entire genome sequences efficiently because these algorithms either run out of memory or take unacceptably long to complete. Moreover, previous algorithms such as Needleman-Wunsch and Smith-Waterman were designed mainly to find insertions, deletions, and point mutations, not to discover the large-scale changes in entire genome comparisons. MUMmer global alignment, using the suffix tree technology, addresses these problems.

In the CryptoDB project, we use the OrthoMCL program to identify the orthologous groups with the input of the protein sequences. Although MUMmer can align protein sequences, it can not find the orthologous groups. Thus, the OrthoMCL program is chosen to identify the orthologous groups. Based on the OrthoMCL result, PROmer is run to display the graphical alignment. MUMmer 3.0 is able to run a multi-contig query with another multi-contig reference. The WU-BLASTN (25) comparison data on the contig level has been loaded into the GUS database. Thus, we currently did not use the MUMmer 3.0.

2.5 CHAOS/DIALIGN

Introduction:

CHAOS/DIALIGN employs the concept of anchors in an attempt to increase the speed of multiple alignments without sacrificing accuracy.

DIALIGN is a method for multiple large DNA and protein genomic sequences alignments, but has been limited by the long running time.

CHAOS is an approach for local alignments of two sequences. It can be used as a stand-alone program for local alignment, and also as a pre-processing step to help global alignment find the anchor points. It works by chaining together pairs of local similar regions from each input sequence (12). Then, the remaining regions between these anchors of the chain are aligned by global alignment approaches such as DIALIGN and LAGAN with slow speed but more accuracy. Thus, the global alignment can be sped up without reducing the alignment accuracy.

Outline of alignment:

The CHAOS algorithm is based on two steps (12):

- **Find all of the seeds.** A seed is a pair of similar regions. More accurately, a seed is a pair of words of length *l* with at least *n* identical base pairs.
- **Chain all of the seeds.** A seed can be chained to the other seed when the indices of first seed in both sequences are higher then the indices of the second seed, and also these two seeds should be near. Finally, the score of each chain is calculated using the total number of matching base pairs. By a fixed score cutoff, the maximal chains with score higher than the cutoff score are generated.

After chains of local sequence similarities are identified, they can be used as anchor points for global alignment. Based on the longest increasing subsequence algorithm, the highest

scoring chain of local alignments can be found. For pair-wise alignment, this chain can be directly used to anchor the global alignment performed with an algorithm such as DIALIGN and LAGAN.

For multiple alignments, the following processes are required (12):

- **Apply CHAOS local alignment to all pairs of input multiple sequences.** A list of anchors with similarities regions are obtained and considered as candidate anchor points.
- **Sort all candidate anchors by the scores associated with each anchor.**
- **Use the greedy algorithm DIALIGAN uses to find the consistent set of anchors.**
- **Accept anchors using the following rules:** The candidate anchors are accepted one by one as final anchor points. Starting with the highest scoring anchor, an anchor will be accepted if it is consistent with those accepted candidate anchors, otherwise discarded. Finally, the consistent anchor points are found.

Input and output files

The input files required by the CHAOS/DIALIGN server are in multi-FASTA format. The output files of the CHAOS/DIALIGN server contain four different types of files:

- The alignment file in DIALIGN is own format.
- The fragments file, which contains the aligned gap-free segment pairs.
- The anchor points created by CHAOS.
- The phylip tree file.

Visualization:

The visualization tool ABC (Application for Browsing Constraints) (15) is used for interactive graphical representation of the multiple alignment output.

2.6 Mauve

Introduction:

Early sequence alignment methods such as Smith-Waterman (54) and Needleman-Wunsch (38) are designed to align small nucleotide sequences. With the availability of genome sequences, the global alignment methods such as GLASS, AVID, and LAGAN have been developed, but they must satisfy the assumption that the genome sequences are highly similar and without significant rearrangements (16). Recently, *glocal* alignment methods, such as SLAGAN, have been developed. It can handle long genome sequences in the presence of rearrangement, but currently can not align multiple genome sequences. MultiPipMaker (50) is another alignment method, based on BLASTZ (53), which can align genome sequences with rearrangement. Also it can be used for multiple genome sequence alignment. Because it uses the BLASTZ local alignment method on each pair genome sequences, more divergent regions between local alignments might not be aligned. Moreover, both SLAGAN and MultiPipMaker can not identify the breakpoints of rearrangement events.

Mauve, a type of multiple genomic sequences alignments with rearrangement can address these limitations. It has been applied to align nine *enterobacterial* genomes and three *mammalian* genomes (16). Also a simple viewer has been developed to display the rearrangement structure between genome sequences. Mauve alignment and visualization software is available: http://gel.ahabs.wisc.edu/mauve. It combines three alignment ideas: MUM (maximal unique match), LCB (locally collinear blocks), and the use of a progressive algorithm. The detailed algorithm will be described in the following sub-section.

Outline of algorithm:

The alignment algorithm can be summarized as follows (16):

- **Find local alignments (multi-MUMs).** Multi-MUMs are exactly matching subsequences shared by two or more genome sequences that occur only once in each sequence. As described above, the generation of multi-MUMs is based on the suffix tree data structure.
- **Construct a phylogenetic guide tree.** Mauve uses the multi-MUMs to construct a phylogenetic guide tree using a Neighbor Joining distance matrix (49). The sequence similarity is based on the ratio of base pairs shared in the two genomes to the average genome sequence length. Based on the multi-MUMs, we can find the number of MUMs between two sequences. Then the similarity value defined above can be calculated. Thus, the phylogenetic guide tree will be constructed according to these similarity values.
- **Select the anchors from the multi-MUMs.** Two steps exist to choose the anchors. 1. Generate the LCBs (locally collinear blocks) from the set of multi-MUMs. Each LCB can be considered a subset of the multi-MUMs with one or more continuous matches. Thus, each LCB does not contain any rearrangements. At the same time the weight values are added to each LCB. The weight is defined by the sum of each match's length of the LCB. 2. Remove the low-weight LCBs using a greedy algorithm given a minimum weight criteria Minimum Weight $\geqslant 0$ and then merge adjacent LCBs into a new LCB. After that the new set of LCBs is considered as a new set of anchors and used to guide the remainder of the alignment process. See Figure 9 for an example of anchor selection.

- **Perform recursive anchoring to identify additional alignment anchors within and outside each LCB.** The third step is not enough to detect the full LCBs. From Figure 9C, we see that two types of regions are considered to recursively anchor. First type of region is between anchors and the second type of region is inside the anchors. Because the low-weight LCBs are removed and the adjacent LCBs are merged into a new one, the second type of regions occurs. Thus, the recursive anchor operation is needed.
- **Perform a progressive alignment of each LCB using the guide tree.** After finding the complete set of alignment anchors, Mauve perfumes a progressive alignment using the phylogenetic guide tree.

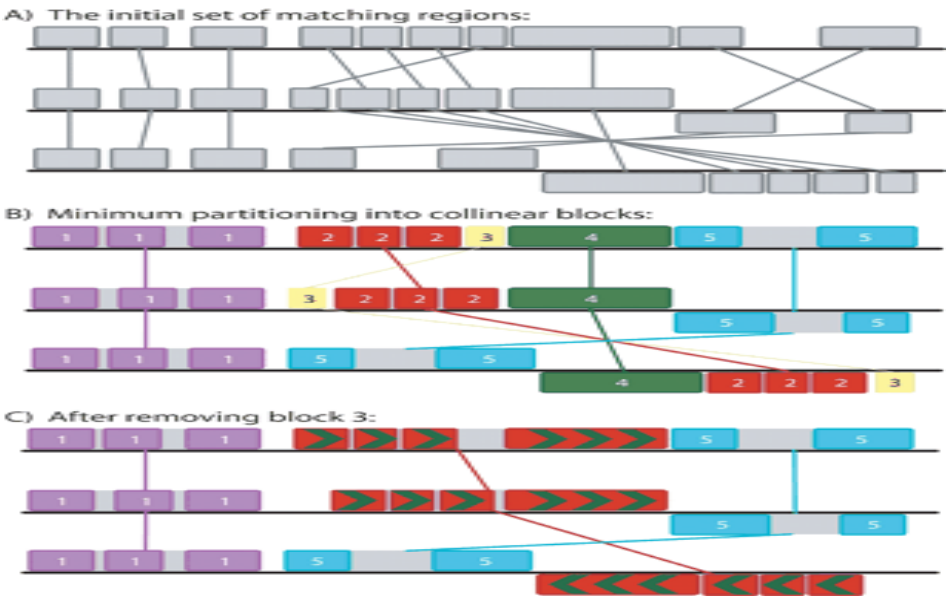


Figure 9. An example of anchors selection. (*A*) The set of multi-MUMs in three genome sequences (*B*) The set of multi-MUMs is split into a minimum set of LCBs. Each LCB is shown as one colored block with one or more matches. (*C*) Remove the low-weight LCBs and merge the adjacent LCBs into a new single LCB. (16)

Input and output files:

Input file formats should be FASTA, multi-FASTA, GenBank with annotation information, sequence, or raw format with the follow file name extensions: .fasta, .gbk, .seq, or .raw.

Output files mainly contain two different file formats: .mauve/.mln and .alignment, and the other auxiliary files such as the phylogenetic guide tree, the .backbone, and the .islands.

- ✧ The .alignment file contains the complete genome alignment in extended multi-FASTA alignment (XMFA) file format.
- ✧ The .mauve or .mln file stores the coordinates of large exactly matching regions.
- ✧ The phylogenetic guide tree is in the standard Newick tree file format. The description of Newick tree file format can be seen at: http://evolution.genetics.washington.edu/phylip/newicktree.html.
- ✧ The .islands file lists all genomic islands found in the alignments. Each island represents a region where one or more genomes have a sub-sequence, but one or more others lack.
- ✧ The .backbone file records each conserved region by one line.

Visualization:

Mauve provides a simple viewing system to display the rearrangement structure of several genome sequences. A detailed description of the features of this viewing system will be introduced in the following sub-section.

Evaluation:

Because Mauve uses the multi-MUM as anchors that are exactly matching sub-sequences without any substitutions or indels, it is not suited to align divergent genome sequences. Compared with Mauve, MLAGAN use the CHAOS anchoring technology with substitutions and

indels and the LAGAN global alignment method, so it can compare much more distant genomes sequences than Mauve, and also it is suitable for cross species comparison (16).

CHAPTER 3

REVIEW OF VISUALIZATION TOOLS

Several comparative sequence visualization tools are publicly available, including VISTA (24), Phylo-VISTA (52), SynBrowse (43), Sybil (http://sybil.sourceforge.net/), Mauve (16), ACT (14), ABC (15), BSR (46), MUMmer (17, 18, 35), and Mulan (41). In the following sub-sections we present each of these visualization tools and provide a detailed description of the features of the tools.

3.1 VISTA & Phylo-VISTA

Introduction:

VISTA **(**VISualization Tool for Alignments) (24) is a program, based on global alignment strategies such as AVID or LAGAN and curve-based visualization techniques, to describe two or more organisms' DNA sequence alignments with various types of annotation. It allows for identification of conserved sequences, straightforward configuration, and enables the visualization of alignments of various lengths at different levels of resolution. The VISTA server is available on the web at http://www-gsd.lbl.gov/vista. The source code is available at vista@lbl.gov.

Phylo-VISTA (52) is an interactive visualization and analysis tool for aligned multiple genomic sequences. Phylo-VISTA is available at http://www-gsd.lbl.gov/phylovista.

Input files:

The input files to VISTA consist of three parts:

- ✧ A global alignment file such as that produced by AVID (7) and LAGAN (10). If unaligned sequences are provided, the AVID or LAGAN global alignment algorithm is used to generate the standard alignment output format.
- ✧ The annotation files for the base sequences in GFF (Generic Feature Format) (http://www.sanger.ac.uk/Software/formats/GFF/) format.

The input file to Phylo-VISTA consists of two parts:

- ✧ A global multiple alignment file such as that provided by MAVID or MLAGAN. If unaligned multiple genomics sequences in multi-FASTA format are provided, MAVID or MLAGAN will generate the standard multiple alignment output format.
- ✧ The phylogenetic tree structure should be provided in the Newick format such as "[(human mouse) chicken]".

Visualized features:

The VISTA tool visualizes alignments of up to several Mbp at different levels of resolution with clear configurable graphical output. It uses a continuous curve to represent the level of identity. The *X*-axis represents the base sequence and the *Y*-axis represents the percent identity. It displays sequence annotations including repeats, coding exons, and UTRs above the plot. The web-based application has many additional interactive functions such as zoom, extraction of a region to be displayed, and user-defined parameters for conservation level.

The Phylo-VISTA tool displays the multiple DNA sequence alignments at varying levels of resolution through the phylogenetic tree. It simultaneously visualizes alignments of the subsets of given sequences at the same scale and displays the gaps and gene annotations for all sequences. The subsets are defined by the phylogenetic tree. It also displays the phylogenetic tree structure and shows the base-pair sequence in a text format. In figure 11A, the pair-wise

phylogenetic tree is shown: all sequences in the alignment are represented by red leaf nodes and each black node represents a similarity plot for all the descendent leaf nodes; In figure11B, the sequence traversal panel and sequence similarity plots are displayed; In figure 11C, the detailed color-coded text alignment window is shown (52).

Evaluation:

The web-based applications of VISTA and Phylo-VISTA can interactively and dynamically visualize the result of comparative sequence analysis on the scale of whole genome with annotations. However, they can not be integrated into the GBrowse framework currently used by the CryptoDB project. Also, the *C. parvum* genome has 18 long contigs and the *C. hominis* has 1422 short contigs. It is difficult to display the comparison on the contig and gene level between two genomes because one *C. parvum* contig is much longer than a *C. hominis* contig. VISTA and Phylo-VISTA also can generate the graphical comparison output in postscript format without the interactive and dynamical features. Thus, although VISTA and Phylo-VISTA have many visualization features, they are not suitable for use in the CryptoDB project.

Screen shots:

(1) VISTA:

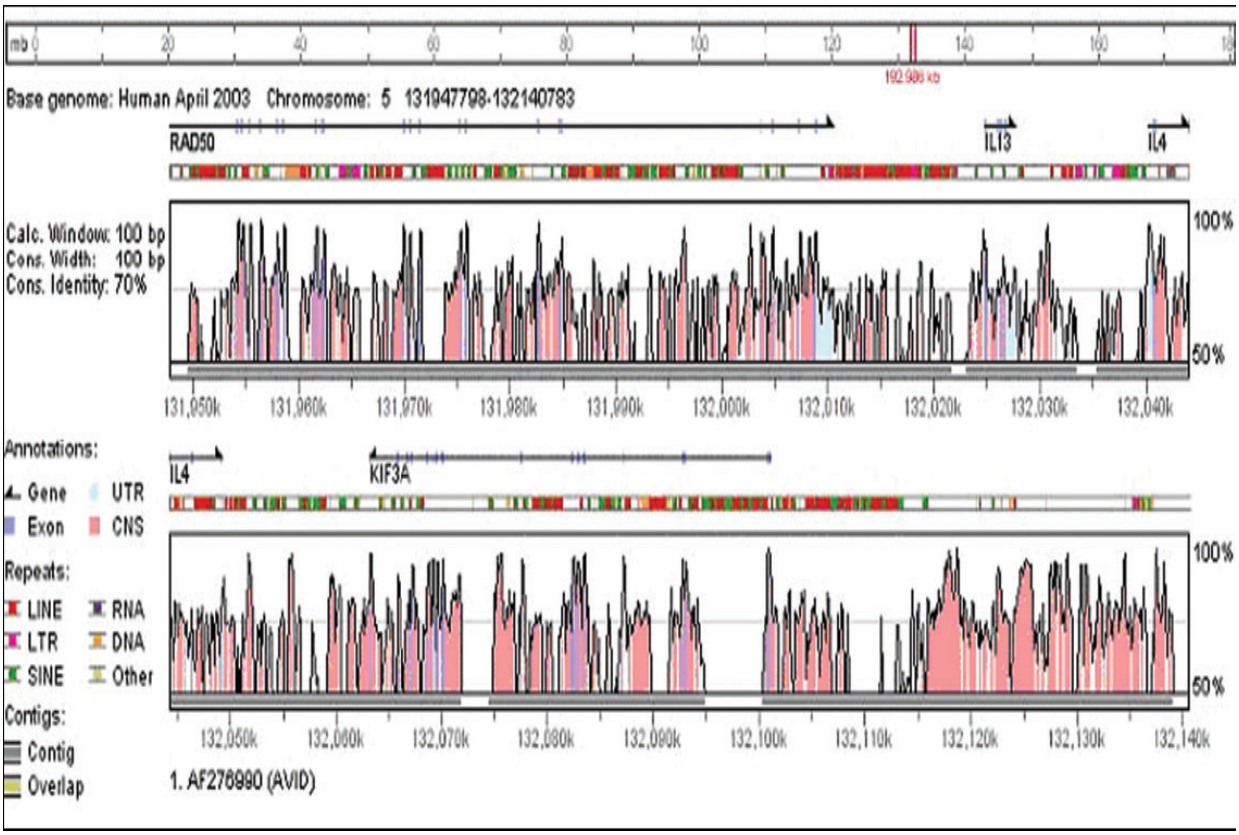


Figure 10. VISTA Browser plot (24)

(2) Phylo-VISTA:

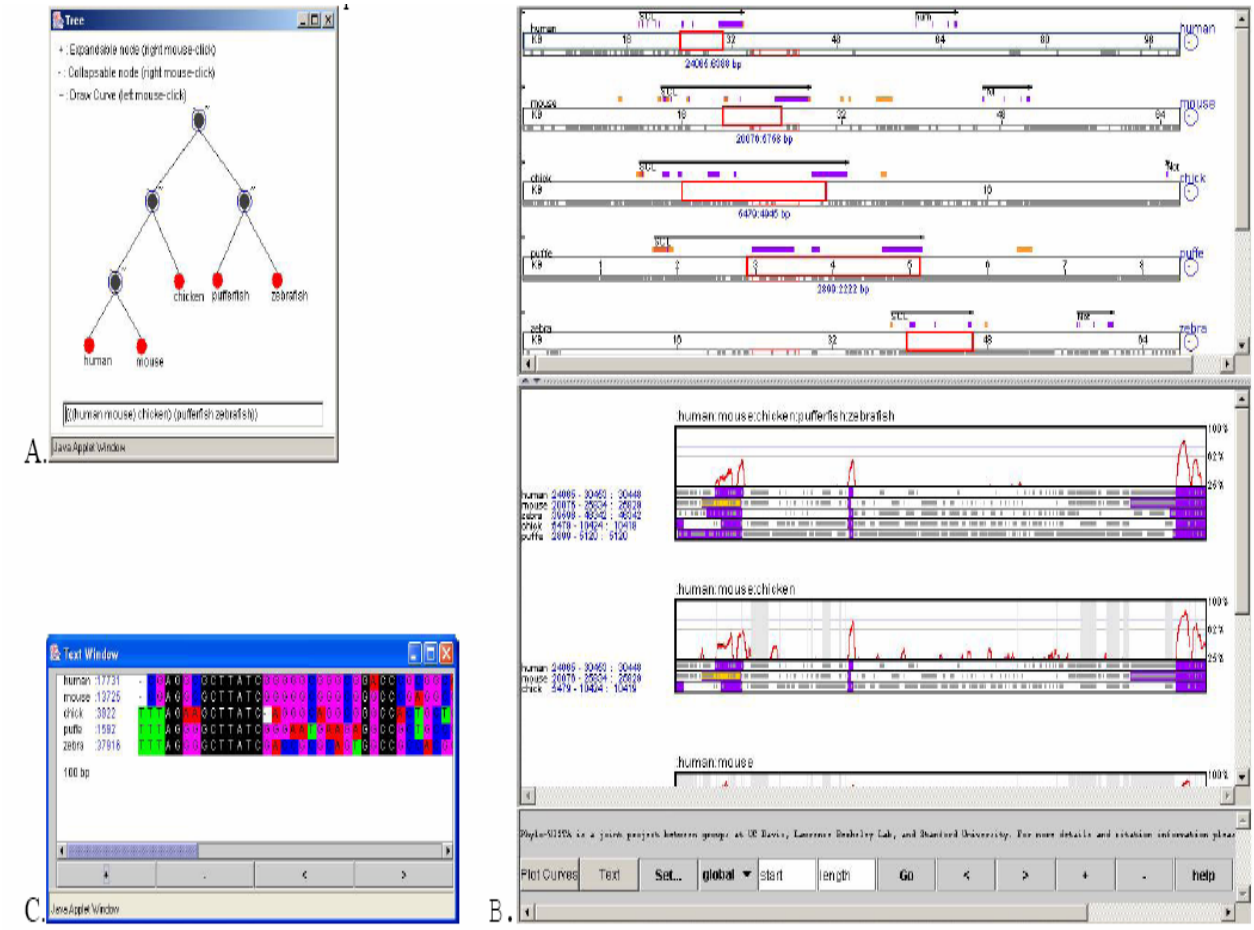


Figure 11. Phylo-VISTA. **A**. Phylogenetic tree - In this pairwise phylogenetic tree, all sequences in the alignment are represented by red leaf nodes. Each black node represents a similarity plot for all the descendent leaf nodes. **B**. Sequence Traversal Panel and Similarity Plots. **C**. Text Window - Part of the alignment in color-coded text. (52)

3.2 SynBrowse

Introduction:

One major concept in comparative genomic analysis is “synteny”, which means that a set of genes and other features share the same relative ordering on the chromosomes of two species or between duplicated chromosomes of a species (43). SynBrowse is a synteny browser for visualizing and analyzing genome alignments of two or more species. It studies macro-synteny,

micro-synteny and homologous genes between genomic sequences and identifies uncharacterized genes, and putative regulatory elements (43).

GBrowse (the Generic Genome Browser) (56) is a web-based application to display genomic annotations and other features, but it can not display the relationships among annotated features for more than one genome. SynBrowse is a GBrowse family software tool that runs on top of the BioPerl (55) modules for comparative genomic sequence analysis. It consists of two components: a web-based front-end and a relational database back-end. Each database stores pre-computed protein and nucleotide alignment data from a focus sequence to reference sequences with genome annotations. By default, the local alignment program BLASTZ (53) is used to generate nucleotide-level alignments, and the GeneSeqer (9) program is used to produce protein to genomic DNA spliced alignments (43).

Input files:

Because SynBrowse uses a relational database to store alignment data, the alignment output files must be formatted. Several steps are described as follows:

- Generating alignment files: The nucleotide-level alignments files are generated using the BLASTZ local alignment program.
- Formatting alignment files: The alignment files are converted by perl scripts into two GFF (http://www.synbrowse.org/pub_docs/GFF.txt) files, to serve as input to the focus species and to the reference species databases.
- Identifying synteny blocks: A perl script finds synteny blocks between the two genomic sequences. It takes the converted GFF alignment files as the input and searches for gene pairs occurring in the same order in both the compared sequences at a distance less than some maximal distance defined by the user. A set of such gene

pairs of some minimum size is considered a synteny block. This minimum number can be specified by the user. Thus, the definition of synteny blocks is flexible. The synteny blocks' files are also in the GFF format.

- Other genome annotation data. All genome annotation data are in the same GFF format as required by GBrowse.

Each species data is stored in the corresponding relational database by the BioPerl scripts *bulk_load_gff.pl* or *bp_bulk_load_gff.pl* (56).

Visualized features:

SynBrowse provides a macro-synteny visualization of the global relationships among the species and a micro-synteny visualization of gene-to-gene or exon-to-exon comparison. It also can identify the uncharacterized genes and putative regulatory elements and display the genome duplications among species.

Each species is shown in one panel with a different colored background and each panel can display six tracks: gene, nucleotide alignment, protein alignment, EST, Repeat, and BAC. Additional tracks can be defined by the user. The focus species is on the top panel and the following panels show the reference species. The conserved regions between the focus species and the reference species are shown in different color boxes based on the value of conservation. The comparison regions between species are connected by lines between the start points and the end points.

Evaluation:

The interactive, dynamic display components of SynBrowse, showing microsynteny and macrosynteny of compared species, could be integrated into the CryptoDB site in a straightforward manner. However, each comparison panel (one per species) has its own zoom

scale, and does not easily permit all species to be viewed at the same scale. More critically, SynBrowse has its own database schema and MySQL database, while the CryptoDB project uses the GUS database schema and Oracle database. It would be difficult to modify the GUS database schema in order to integrate SynBrowse application. Thus, it was not used in the CryptoDB project.

Screen shots:

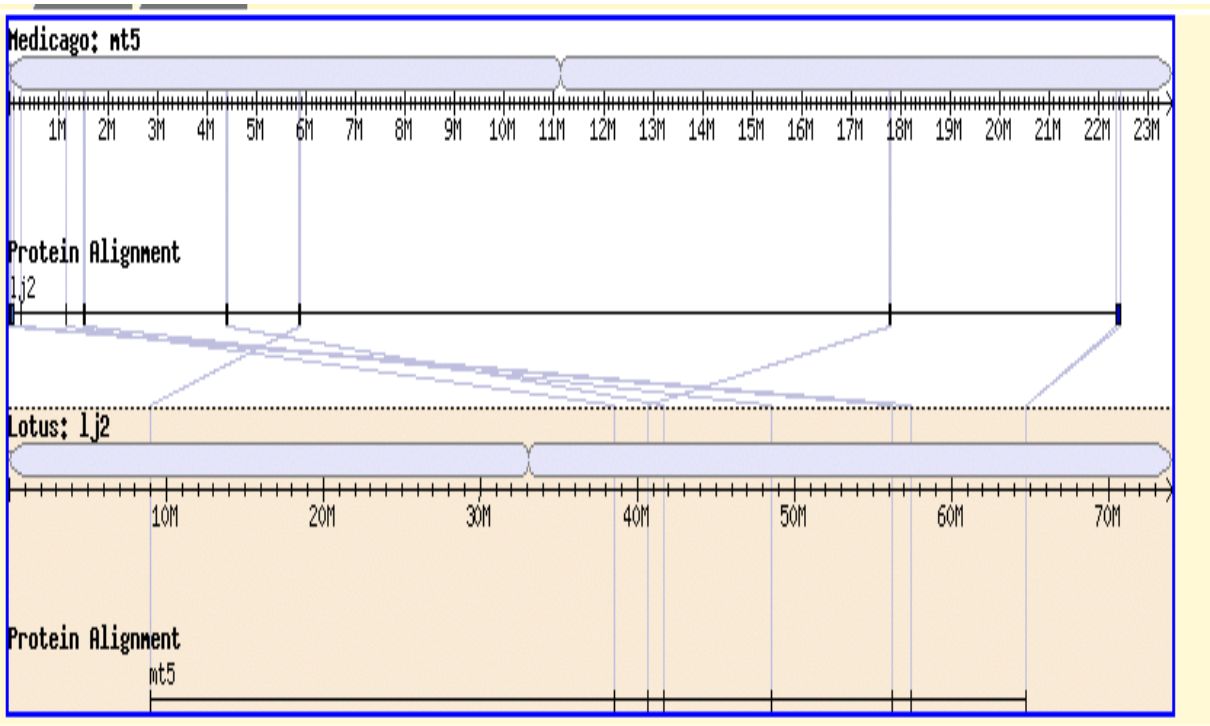


Figure 12. Whole Genome Synteny viewer - two chromosome comparisons (43).

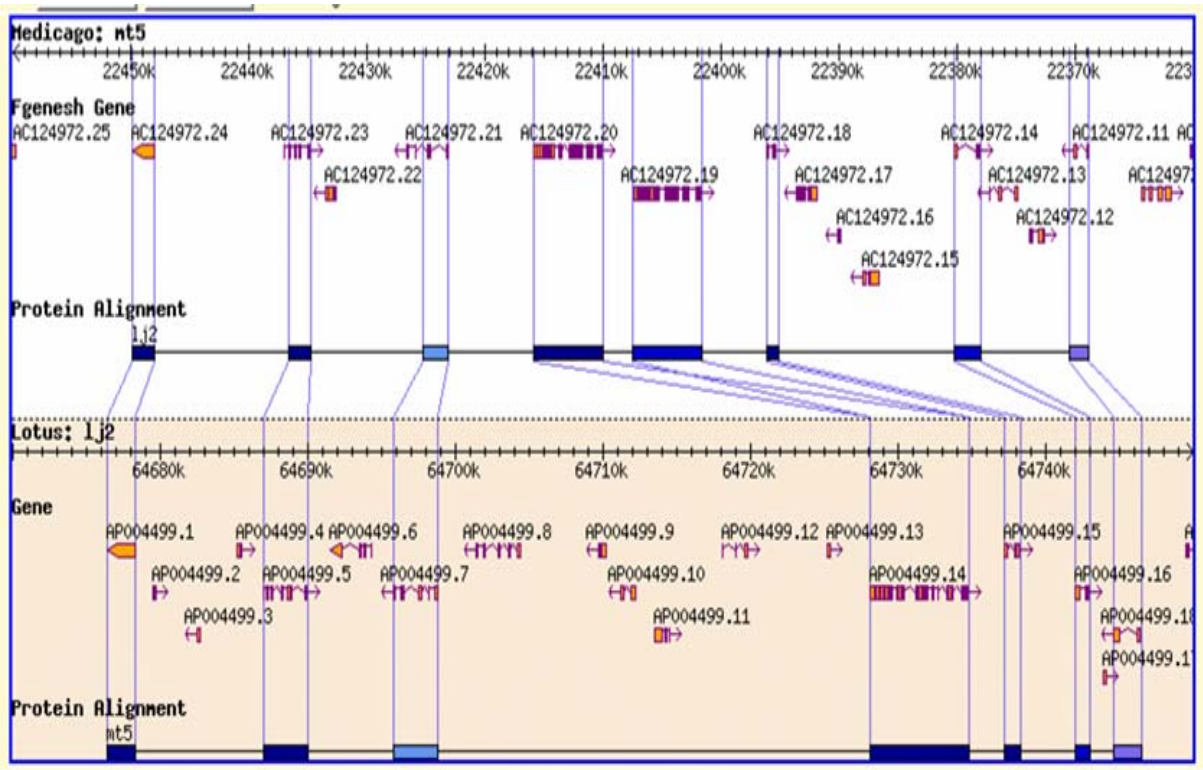


Figure 13. Micro-synteny viewer - gene-to-gene comparisons (43).

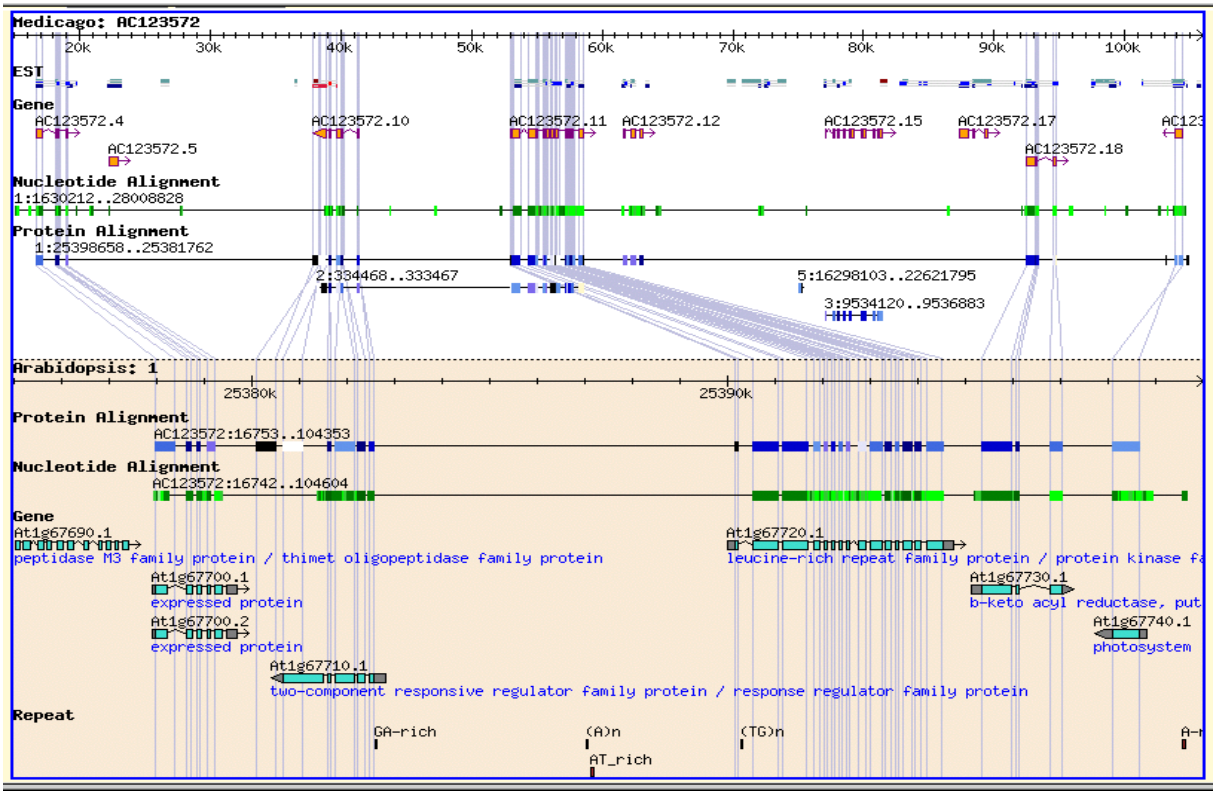


Figure 14. Micro-synteny viewer - exon-to-exon comparisons (43).

3.3 Sybil

Introduction:

Sybil (http://sybil.sourceforge.net/) is a web-based software package for comparative genomics analysis and visualization with a Chado relational database (http://www.gmod.org/schema). It was developed by the Bioinformatics department at The Institute for Genomic Research (TIGR) in Perl using a 3-tiered software framework and had been used internally at TIGR to support many research projects involving comparative genome analysis.

The Sybil demonstration database and source code can be downloaded at the website: http://sybil.sourceforge.net/downloads.html.

Input files:

It uses a Chado comparative database as a data source. The schema of Chado is available at the GMOD website (http://www.gmod.org/schema).

Visualized features:

Sybil provides two kinds of displays: clickable interactive graphical displays, and publication-ready figures in a variety of formats including SVG and PDF.

The interactive graphical display includes a genome sequence overview, a chromosome overview, protein cluster comparison, protein cluster summary, protein/gene summary, SNP (Single Nucleotide Polymorphism) summary report, SNP detail report, and indel detail report. The publication-ready display includes a comparative sequence display that allows the user to select a reference sequence or segment and set a number of parameters and then display a set of similar sequences from the database.

Evaluation:

Some of the Sybil displays, such as the protein cluster summary, satisfy the requirements for CryptoDB comparative genomics visualization. The Chado database schema is used by Sybil, while the CryptoDB project uses the GUS database and schema. Only a subset of the displays is currently available in the Sybil software source code which is running on the demo website at http://sybil.sourceforge.net/demos.html. The web demo of Sybil provides gene or protein searching and links for each protein cluster. However, on the comparison viewer of each protein cluster, additional tracks can not be added and the genome regions can not be zoomed. Although the Sybil viewer could not be used directly in the CryptoDB project, some ideas from Sybil have been integrated into the CryptoDB project.

Screen shots:

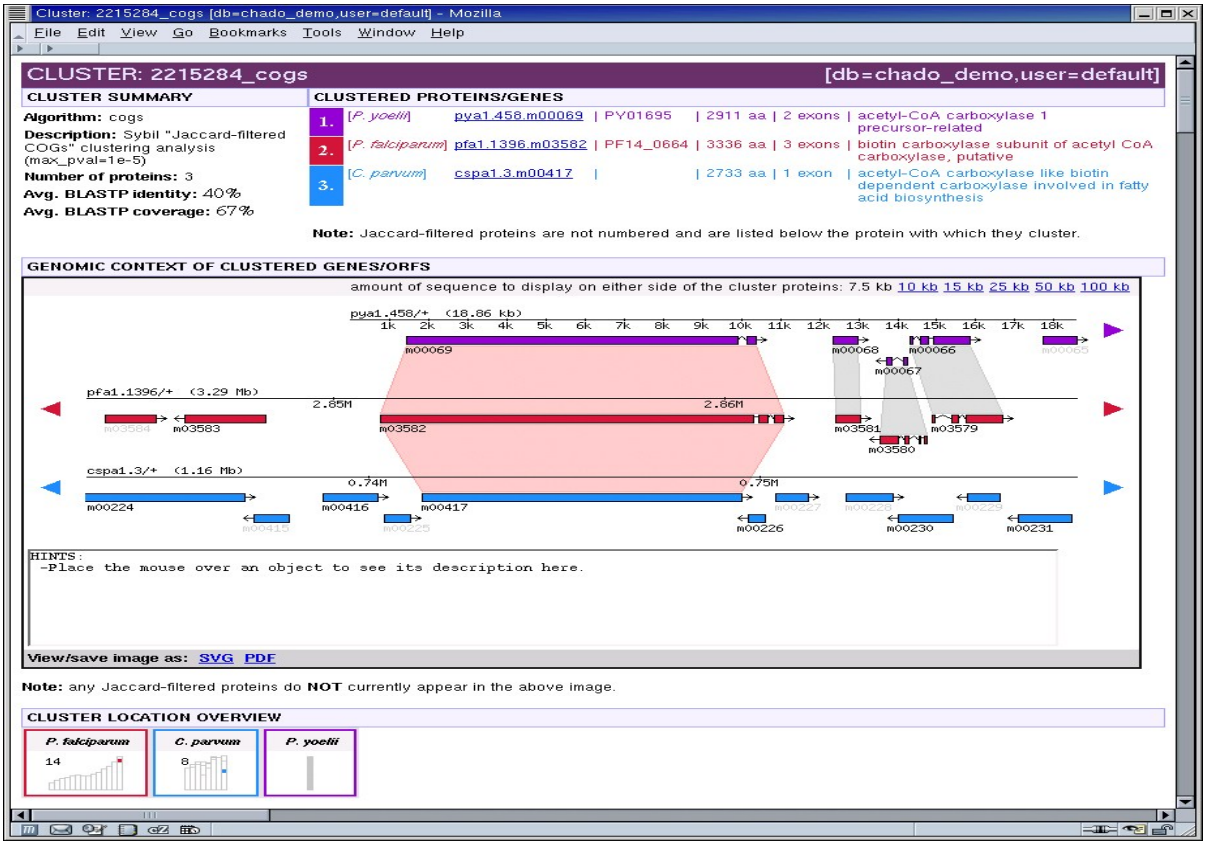


Figure 15. A sample of the protein cluster summary display (http://sybil.sourceforge.net/).

3.4 Mauve

Introduction:

Mauve is a multiple genomic sequence analysis and visualization tool that supports rearrangement events (16). The analysis component is discussed in section 2.6. In this section we focus on the display features of Mauve.

Input files:

In section 3.6, the input files of Mauve are listed.

Visualized features:

Each input genome is displayed in one horizontal panel containing the genome sequence name, a scale with the sequence coordinates, and a single black horizontal center line. Colored blocks appear above or below the center line. Blocks above the center line mean the aligned regions are in the forward orientation relative to the first genome sequence. Blocks below the center line indicate the aligned regions are in the reverse orientation. Regions outside blocks are too divergent to be aligned successfully. Inside each block, Mauve draws a similarity curve of the genome sequence. The height of the grey similarity bars corresponds to the average level of conservation in that region of the genome sequence. Areas that are completely white are not aligned. Mauve highlights the aligned regions of each genome with a black vertical bar as the user moves the mouse over the alignment display. (16)

Mauve displays the annotated features including CDS features, tRNAs, rRNAs, and other features next to the sequence similarity profiles and provides an interactive display to view the detailed information by a popup window (16). By zooming the main panel in close, each genome’s nucleotide sequence may be viewed.

Evaluation:

Mauve is a stand-alone comparison analysis and visualization tool. It displays the comparison with the whole genome on the DNA level. It can not dynamically display the regions that the user selects. It also can not provide comparison on the gene level. In addition, Mauve is stand-alone software with available source code (http://gel.ahabs.wisc.edu/mauve/download.php) and thus could not be integrated into the GBrowse framework. It visualizes the alignments over the entire genome sequence. So, it can not be used for comparison visualization in the CryptoDB project.

Screen shots:

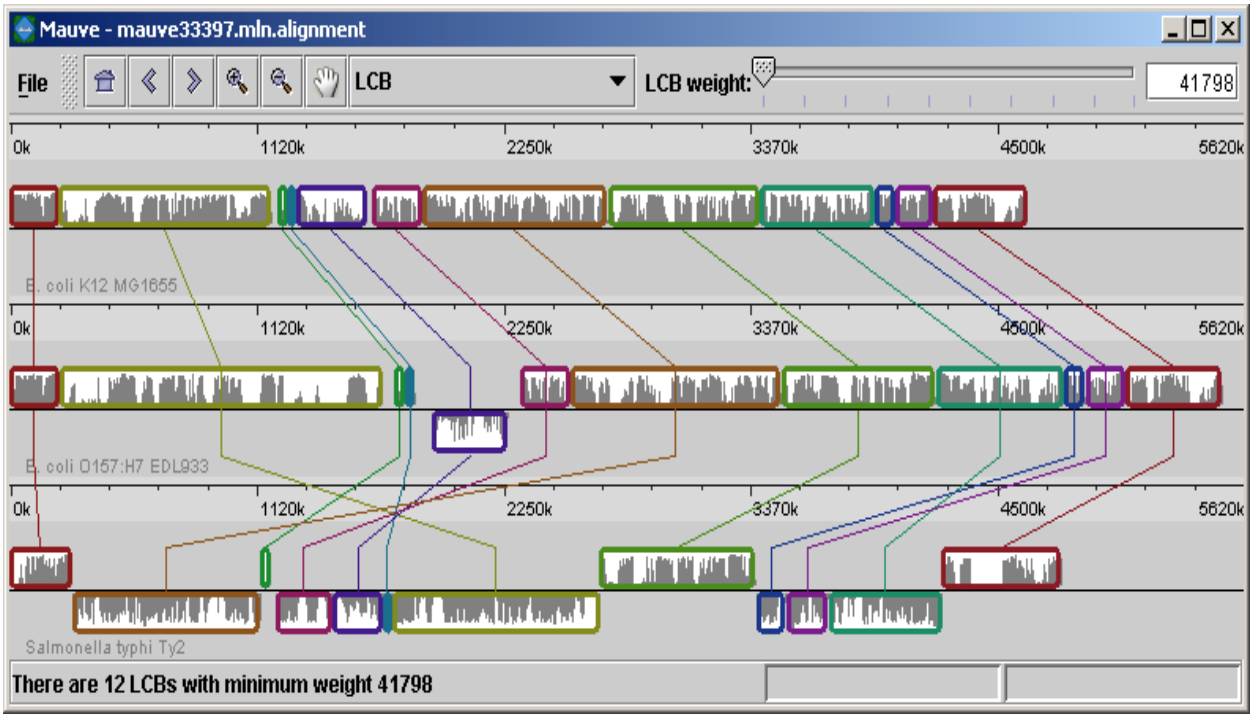


Figure 16. The sample of Mauve main panel display.

(http://gel.ahabs.wisc.edu/docserver/mauve/display.stx)

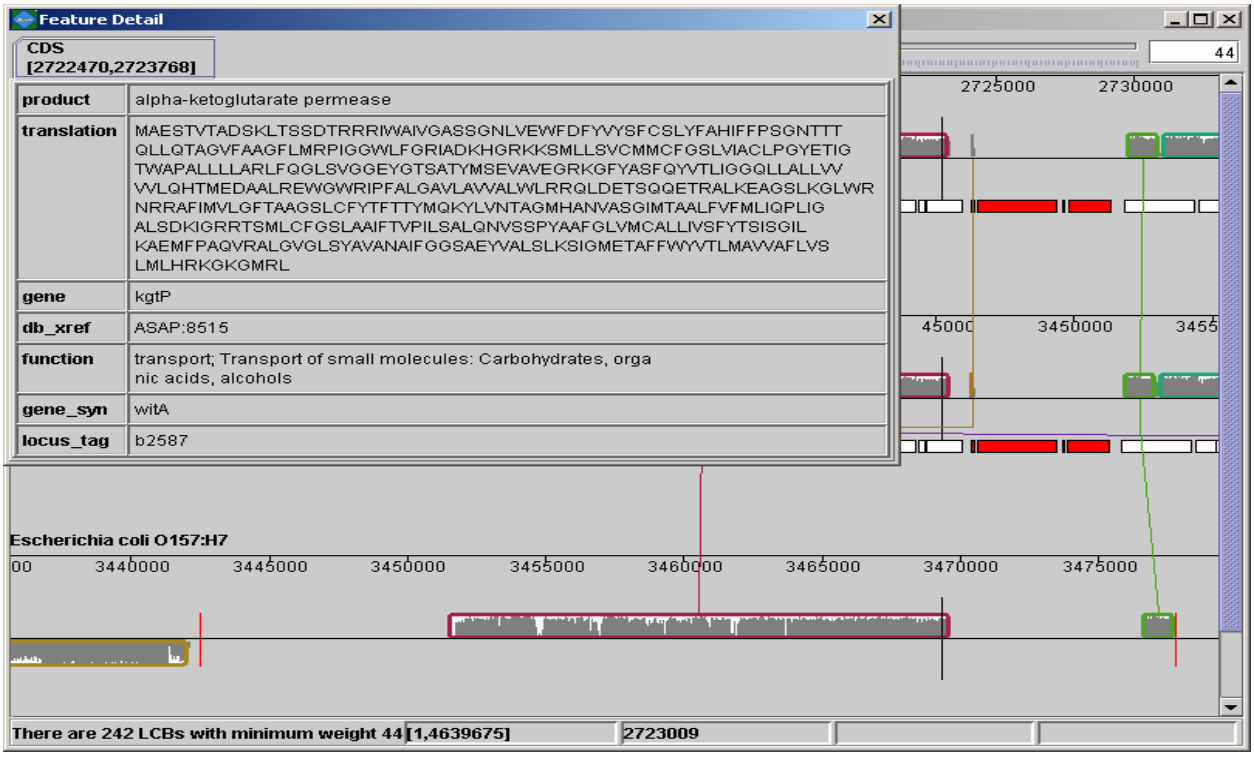


Figure 17. The sample of the Mauve annotated feature popup window. (http://gel.ahabs.wisc.edu/docserver/mauve/display.stx)

3.5 ACT

Introduction:

The ACT (Artemis Comparison Tool) (14) interactively visualizes complete pair-wise genome sequence alignments. It can identify regions of similarity, insertions, and rearrangements at any level from the whole genome to base-pair. ACT is written in Java with open source and is available from the Sanger Institute web site (http://www.sanger.ac.uk/Software/ACT/).

Input files:

ACT can directly read the output of BLAST (version 2.2 or higher) programs including BLASTN, TBLASTX or MEGABLAST with genomic DNA sequences in GenBank, GFF, or

FASTA formats. Output of other alignment programs such as WUBLAST, BLAST 1.4, and MUMmer must be parsed into the simple format used by ACT.

Visualized features:

ACT has two bands: red bands represent forward matches and blue bands represent reverse matches. A highlighted match region turns yellow. The associated matching regions are centered when the users double clicks on one band. The intensity of the color bands is proportional to the percent identity of the match. Reverse matches are viewed by flipping either of the sequences around. To view, or scroll through all the matches that overlap a selected feature or region, the *view selected matches* function can be used. The *feature selector* and *navigator* are tools for searching for features by location, feature name, length and position. (14)

ACT can create features from non-matching regions. All annotation features can be displayed simultaneously by ACT.

Evaluation:

ACT is also a stand-alone comparison analysis and visualization tool, implemented in Java. It displays the comparison over the whole genome on the DNA level. It can not dynamically display the regions that the user selects. Although it can display the annotated protein features, it can not view the comparison on the gene level. The *C. parvum* and *C. hominis* genomes are very similar, so the display on ACT with these two sequences is very busy (Figure 17). Thus, it is not suitable for use as the comparison visualization in the CryptoDB project.

Screen shots:

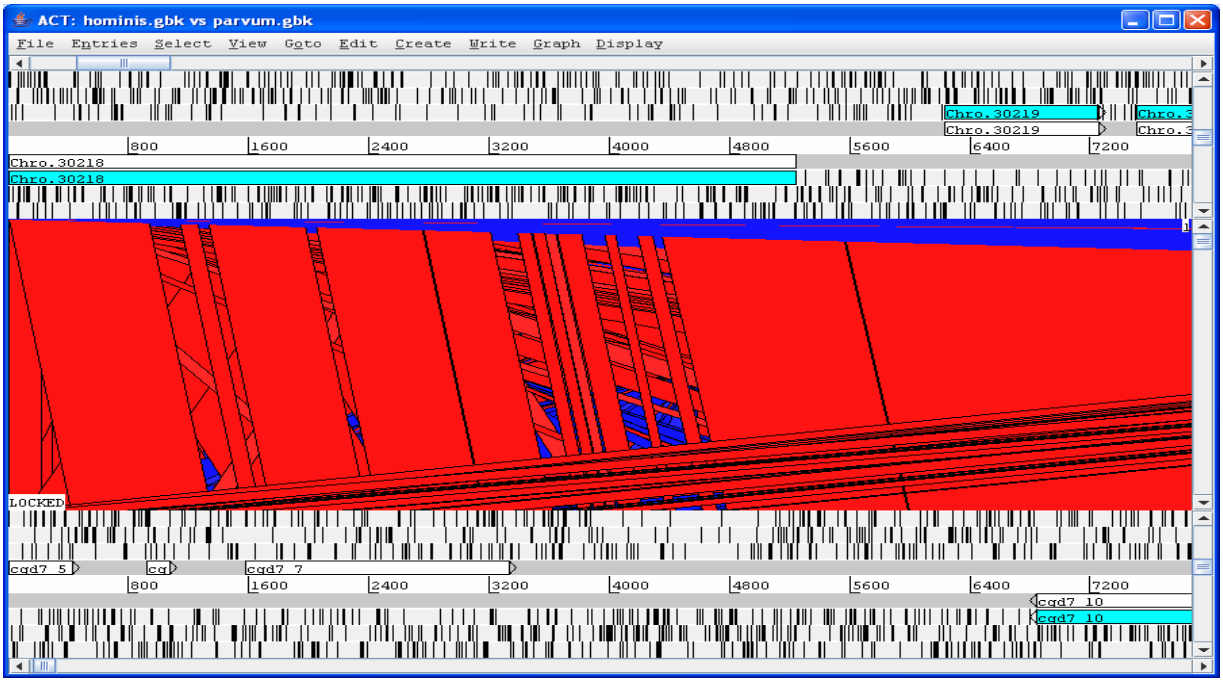


Figure 18. An example of comparison view of *C. parvum* and *C. hominis* genomes by ACT. This graph shows the relationship between two sequences. The displays are collapsed to show all forward and reverse features on single lines (colored boxes represent coding sequences).

3.6 ABC

Introduction:

The ABC (Application for Browsing Constraints) (15) is interactive Java software for exploration of multiple sequence alignments, such as those produced by CHAOS and DIALIGN, with annotation sequence data. It is a lightweight, stand-alone, graphical user interface for browsing genomic multiple genome sequence alignments, up to a few Mbp (15). Documentation and some sample data are available at the website:

http://mendel.stanford.edu/sidowlab/downloads.html.

ABC is not suitable for the following types of comparative genome analysis:

- ✧ Genome-wide browsing of alignment data. It is built for the several-Mbp scale.

- ✧ Generation of alignments. The ABC is a visualization tool only, and requires the alignment to be pre-generated.
- ✧ Tree-building or phylogenetic analysis.

Input files:

The major inputs consist of three files: a multiple sequence file in multi-FASTA format, standard phylogenetic tree description file, and an annotation file.

Visualized features:

ABC displays the alignments in three distinct modes based on the density of the information (Figure 18) through the mobile and scalable zoom window. The user can drag and resize the rectangle and then a more detailed viewer is expanded below the parent viewer when a desired region is selected by the *GoTo* feature. At very low resolution, a histogram is displayed that plots the number of data points (Figure 18, top-panel). At intermediate resolutions, a more detailed region of the selected black box in the top-panel is displayed (Figure 18, middle-panels). Finally, at very high resolution, the sequence data is displayed directly along with the sequence names and a phylogenetic tree structure (Figure 18, lowest-panel). The black rectangle can be moved and resized to highlight particular regions. All annotated features tracks are displayed above the panel. The colors for annotated features can be specified. (15)

Evaluation:

ABC is a stand-alone Java program. The display is curve-based. It supports comparison on the DNA level. It can not be used to view the comparison on the gene level. Also, it displays the comparison over the entire genome. The user can not dynamically select the genome regions to display. The lack of a gene level comparison and dynamic selection make ABC unsuitable for use in the CryptoDB project. It can be used to facilitate basic comparative sequence, such as

export the data in plain-text formats, visualization of phylogenetic trees, and generation of summary graphical alignment.

Screen shots:

Figure 19. A screenshot of the ABC. (A), the upper-most panel consists of a histogram describing the regional density of columns. (B), the middle panels consist of more detailed alignment of the black rectangle box of upper panel. The black rectangle can be moved and resized to highlight particular regions. (C), the bottom panel shows a view of individual alignment columns corresponding to a small region of the alignment. Also a phylogenetic tree is also displayed in the bottom panel. Above all panels, all annotation tracks are displayed. (15)

3.7 BSR

Introduction:

The BSR (BLAST Score Ratio) (46) approach is a comparison analysis and visualization tool for any three genomes based on the ratio of BLAST scores. It was implemented in Perl scripts. The output of the BSR approach enables the global visualization of the degree of

proteome similarity among all three genomes and the genomic synteny (conserved gene order) between each genome pair.

Input files:

- ✧ The predicted proteomes files of each of the three genomes are in multi-FASTA format. The user selects one proteome as the Reference, and the other two are Query1 and Query2.
- ✧ An additional file for each proteome contains a unique identifier, matching the multi-FASTA files header, and the relative genomic location of the start and stop of the coding regions.

Visualized features:

The graphical output files are viewed by Gnuplot (http://www.gnuplot.info) to reveal the global similarity of the compared genomes. PostScript and xfig graphic files are subsequently generated by Gnuplot. (46)

- ✧ The similarity plot provides an overall view of the level and number of similar and different proteins in the Reference proteome when compared to the Query proteomes.
- ✧ The regions of the graph are color-coded depending on the level of similarity among the three genomes.
- ✧ Two additional plots, known as synteny plots, are generated for comparison of the Reference proteome to each Query proteome, by plotting the genomic location of the Reference peptide on the *X*-axis and the genomic location of the most similar Query peptide on the *Y*-axis.
- ✧ Additional XML files for the similarity and synteny plots are generated in order to view the annotations interactively. These files are the input for the freely available

GGobi (http://www.ggobi.org/) software, a data visualization system for viewing high-dimensional data.

Evaluation:

BSR is a stand-alone comparison analysis and visualization tool for any three genomes. It produces output files that can be viewed in gnuplot (http://www.gnuplot.info) and written out as postscript and xfig. It can not dynamically and interactively display the comparison on the contig and gene level, and this is not suitable for use as the comparative visualization for the CryptoDB project.

Screen shots:

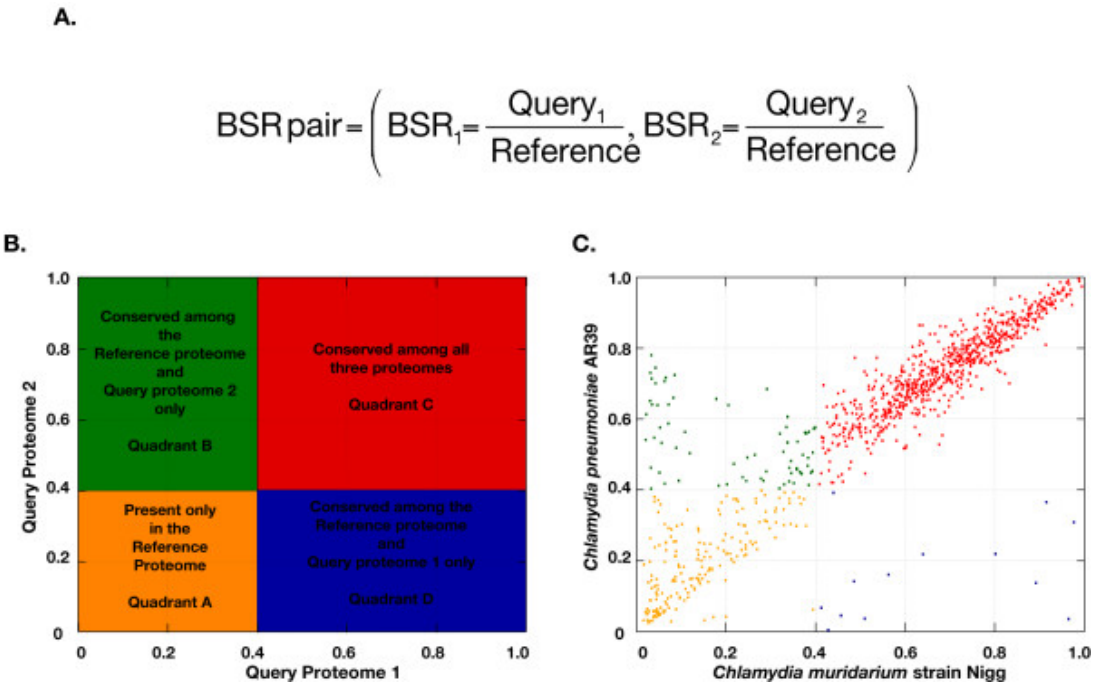


Figure 20. BSR rationale and plot display example**. A.** BSR calculation demonstrating how the two coordinates for plotting in figures B and C are calculated. **B.** Locations of the peptide spot discover the similarity. A 0.4 is used as separator. **C.** Sample data obtained from comparison of *Chlaymidia caviae* GPIC (GenBank Accession Number AE015925) to the proteomes of *Chlamydia muridarum* strain Nigg (GenBank Accession Number AE002160) and *Chlamydia pneumoniae* AR39 (GenBank Accession Number AE002161) (47). Each point in the figure represents a single peptide in *Chlaymidia caviae* GPIC. This analysis discovers that while these organisms are very similar, *C. caviae* GPIC is more similar to *C. pneumoniae* AR39 than *C. muridarum* strain Nigg because the skew of peptides with a slope of greater than 1. (46)

3.8 Mulan

Introduction:

Mulan is a new integrative comparative tool that dynamically generates multiple sequences local alignments based on BLASTZ (53) local alignment method. Mulan consists of several data analysis and visualization for identification of functional coding and non-coding regions that are conserved across the large evolutionary distances (41). It can determine the phylogenetic relationships and generate the phylogenetic trees, construct graphic and textual alignments, dynamically detect evolutionary conserved regions (41). Mulan is available at http://mulan.dcode.org.

Mulan allows for multiple draft and finished sequences alignments. It uses the *TBA* (threaded blockset aligner) program for finished sequences and *refine* program for draft sequences (5). Most global multiple alignments require colinearity between input sequences, but the Mulan does not require it.

Compared with other multiple global alignment methods, the TBA program detects and process DNA rearrangements characteristic of synteny among the distantly related genomes. Thus, Mulan permits the dynamic interchange of reference sequences and interactively generates textual and graphical multiple sequences local alignments (41).

Input files:

It consists of three kinds of input data: the sequence data in FASTA format, the annotation data in annotation format, and the NCBI accession number.

Visualized features:

Mulan comparative visualization is based on the zPicture display (42). The reference sequence is linear along the horizontal axis and the percent identity is plotted along the vertical

axis. Moreover, it consists of a graphical annotation display that contig names and alignment blocks can be visualized as tracks on the top. Synteny blocks are color-coded (41).

In the Mulan, the reference sequence can be dynamically changed, and then the new order of conservation for the rest of the species is automatically determined by the phylogenetic tree. More closely related species are at the bottom. Color density is used to indicate the number of species that share a particular region. More species share a sequence, darker color the conservation is display. Thus, the color density can highlight different DNA segments in the base sequence with unique evolutionary character (41). Also the evolutionary criteria such as length and percent identity can be selected by the users.

Two additional data representations: phylogenetic shadowing and “summary of conservation” are implemented by the Mulan. The “summary of conservation” collects all the shared nucleotide similarities from all the pair-wise comparisons into a single conservation profile; the phylogenetic shadowing option collects all the nucleotide mismatches (40).

The detailed alignments are displayed when clicking on the conservation area. The gene annotation can be modified dynamically.

Evaluation:

Mulan is a web-based multiple sequence analysis and visualization tool. It is curve-based. Currently, the Mulan just has the web server version because the source code of Mulan web scripts and the visualization programs has not been distributed. Our graphical display is based on GBrowse framework with GUS database. Thus, it is not suitable for the CryptoDB project.

Screen shots:

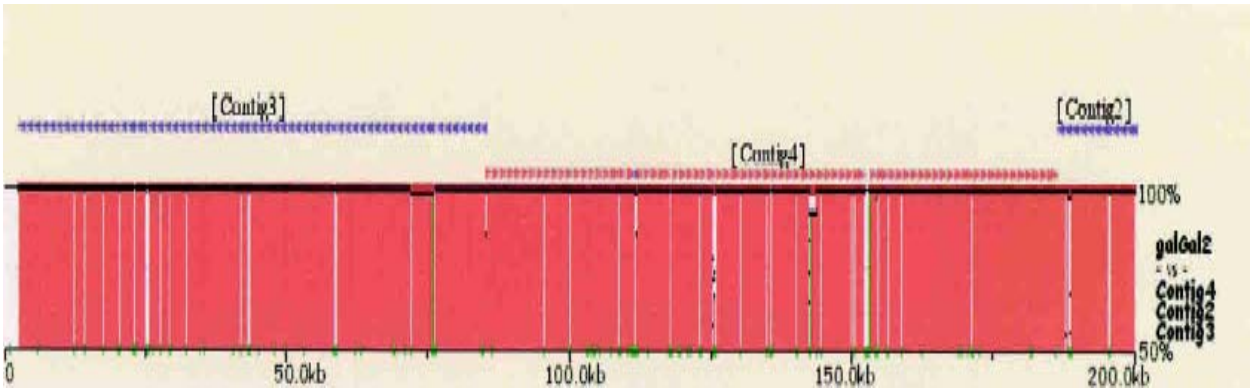


Figure 21. Mulan contig ordering based on homology to the reference sequence. The top layer of shaded lines indicates the location of contigs from a second sequence aligned to the base sequence. Red triangles point to the right specify forward-strand alignments, and purple triangles point to the left correspond to reverse-strand alignments. Contig names are indicated in square brackets. (41)

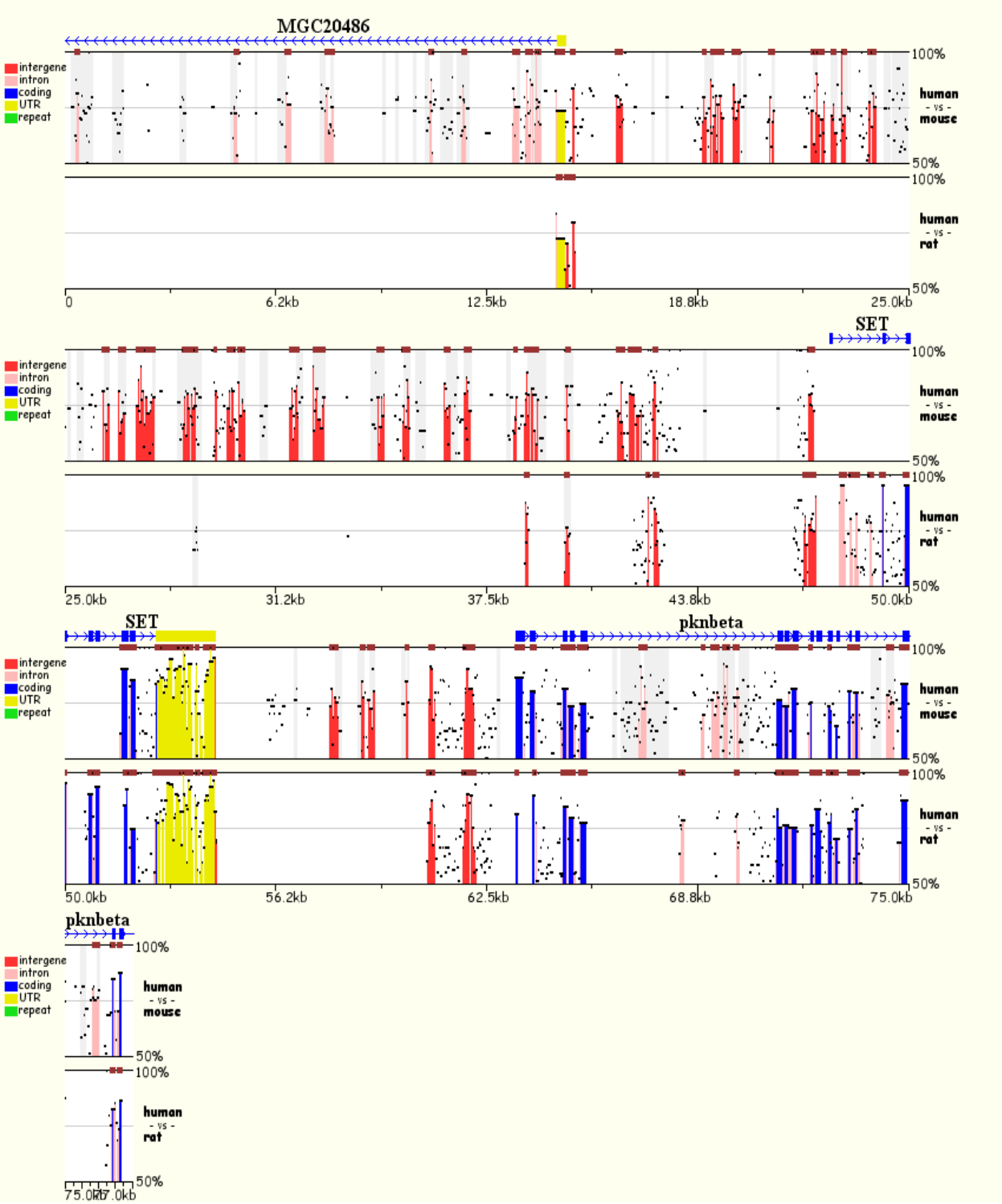


Figure 22. The example of the Mulan visualization. The sample data consist of three segments of genome sequences: human, mouse, and rat with the annotation data. This picture is generated by running these three sequence and annotation data on the Mulan web server.

3.9 MUMmer

Introduction:

In the chapter 2.4, we have introduced the MUMmer alignment method. Moreover, the MUMmer provides the visualization tools to display the alignment generated. Two graphical viewers are developed: the MapView and the mummerplot. The latest version of MUMmer is available at http://www.tigr.org/software/mummer. (17, 18, 35)

- The MapView is for displaying sequence alignment as provided by NUCmer or PROmer. By default, it produces FIG files that can be viewed with xfig or converted to PDF or postscript.
- The mummerplot takes output from mummer, nucmer, promer or show-tiling, and converts it to a format suitable for plotting with gnuplot.

Input files:

It takes the input files that are generated by MUMmer. The detailed information is available at http://mummer.sourceforge.net/manual/.

Visualized features:

The MapView is useful for mapping multiple query contigs against an annotated reference sequence. The blue rectangle represents the reference sequence with annotated genes shown above it and the PROmer alignments shown below it. Alternative splice variants of the same gene are stacked vertically. Exons are shown as boxes, with intervening introns connecting them. The 5' and 3' UTRs are colored pink and blue to indicate the gene's direction of translation. PROmer matches are shown twice, once just below the reference genome, where all matches are collapsed into red boxes, and in a larger display showing the separate matches within each contig, where the contigs are colored differently to indicate contig boundaries. The vertical position of

the matches indicates their percent identity, ranging from 50% at the bottom of the display to 100% just below the red rectangles. Matches from the same query sequence are connected by lines of the same color. (Figure 23) (17, 18, 35)

In the mummerplot output, the reference sequence is laid across the *X*-axis, while the query sequence is on the *Y*-axis, and a point is plotted at every position where the two sequences show similarity. The forward matches are displayed in red, while the reverse matches are displayed in green. If the two sequences were perfectly identical, a single red line would go from the bottom left to the top right. (Figure 24)

Evaluation:

MUMmer is a stand-alone sequences alignment and visualization tool. It does not support the web-based version. Currently, the WU-BLASTN (25) alignment data has been loaded into the GUS database, and then the OrthoMCL program is used to identify the orthologous groups. In addition, the display system of the CryptoDB project is based on GBrowse framework. We use the PROmer to produce the pre-computed images that are displayed in the detailed alignment page.

Screen shots:

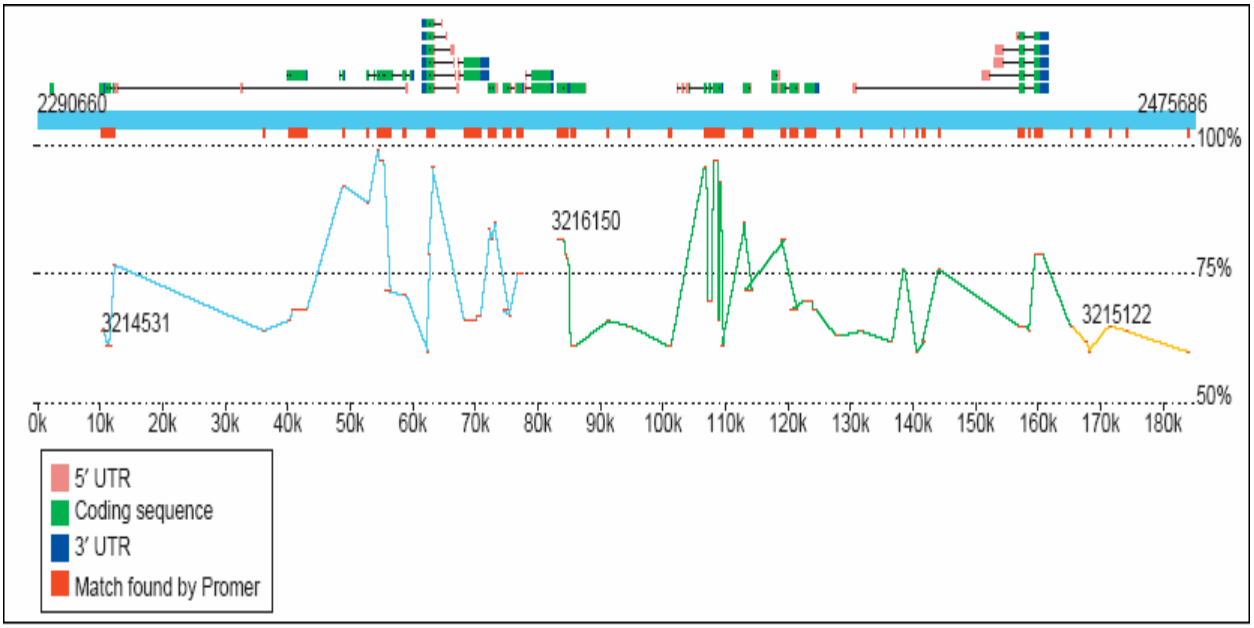


Figure 23. The sample display created by the MapView program. The alignment, generated by Promer, shows all regions of conserved amino acid sequence. The blue rectangle represents the reference sequence, with annotated genes shown above it. Alternative splice variants of the same gene are stacked vertically. Exons are shown as boxes, with intervening introns connecting them. The 5' and 3' UTRs are colored pink and blue to indicate the gene's direction of translation. Promer matches are shown twice. Below the reference genome, all matches are collapsed into red boxes. In a larger display, the separate matches are shown within each contig, where the contigs are colored differently to indicate contig boundaries. The vertical position of the matches indicates their percent identity, ranging from 50% to 100%. (35)

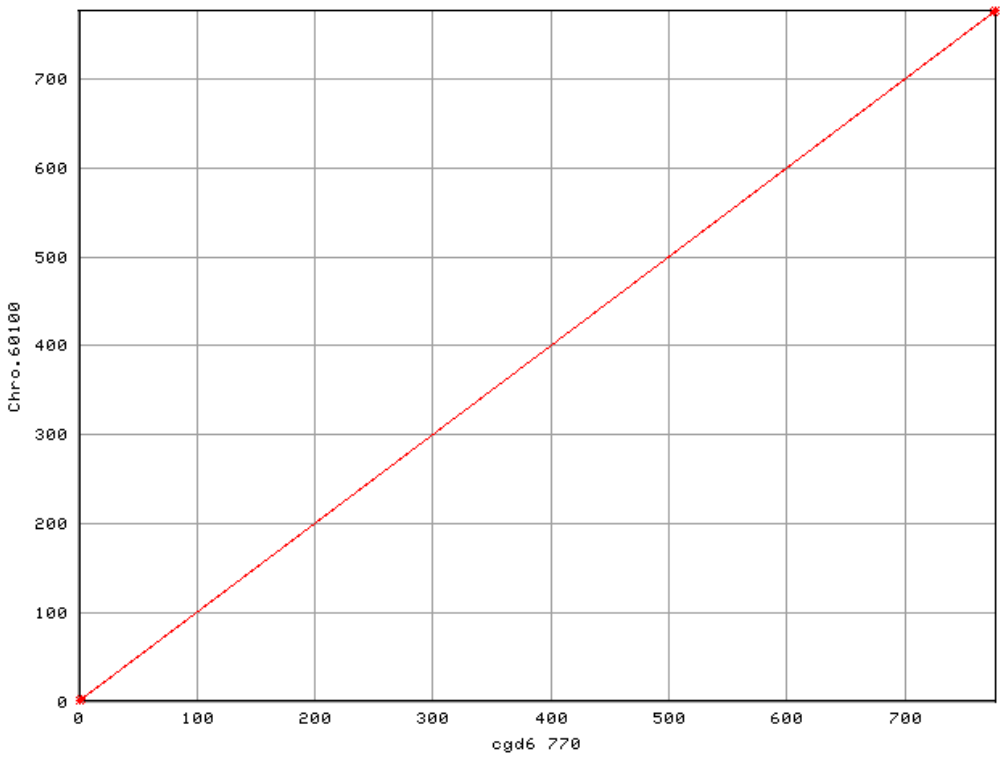


Figure 24. A sample picture of the Promer alignment output by the mummerplot program. The *X*-axis shows the *C. parvum* gene and the *Y*-axis shows the *C. hominis* gene. The red line is composed of the match points between two genes. This example means a perfect match between the *C. parvum* gene (*cgd6_770)* and the *C. hominis* gene (*Chro.60100).*

CHAPTER 4

METHOD AND IMPLEMENTATION

4.1 Introduction

The goal of this study is to provide comparative genomic visualization for the CryptoDB project (www.cryptodb.org), which is now part of an NIH/NIAID funded Bioinformatics Resource Center to provide Apicomplexan Database Resources. By reviewing some existing comparative genomic visualization tools such as VISTA, SynBrowse, Sybil, Mauve, ACT, ABC, and BSR, we find that none of these tools quite meets the needs of the CryptoDB project, which requires that the comparative genome display be dynamically generated and interactive. Further, other displays in CryptoDB such as gene displays and protein feature prediction displays currently employ GBrowse. Providing both single genome and comparative displays in the same format simplifies the user's task of understanding these displays.

The VISTA tool can be used on a web-based server or as a stand-alone program. The web-based server version would not easily support the interaction desired nor would it provide the real-time generation of displays within the context of the CryptoDB site that we wish to achieve. The stand-alone program generates static images. Our requirements are to display the comparative genome information dynamically and interactively.

The Mauve and ACT tools are stand-alone programs to display the comparison of the whole genome, with rearrangements interactively. They do not support the display of regions defined by the user dynamically. They can visualize the comparison on the DNA level, but not on the gene level. The *C. parvum* and the *C. hominis* genome are very similar, so the displays by

ACT or Mauve are very “busy”. Also they are stand-alone programs and difficult to integrate into the current graphical display framework.

The ABC tool is also a stand-alone program to display the alignment data. It can interactively display whole genome features and alignments, but can not dynamically display the comparison only within the genome regions that users define. Also it has its own complex alignment file format as input.

The SynBrowse display could be used on our system because it is also based on GBrowse framework. The problem is that the CryptoDB project uses the GUS database schema and Oracle database, but SynBrowse uses its own database schema and the MySQL database. Modifications to incorporate these tables would be significant. Further, the display presented by SynBrowse is not entirely satisfactory. Each species, in its own panel, has its own zoom scale. The user can not choose to view all species at the same zoon level.

For Sybil, the protein cluster summary display can satisfy our comparison requirements. Also it supports searching by gene or protein. Images are generated interactively. The problems are that it does not support the zoom function and that additional tracks can not be added into the panel. Also the user can not select the genome regions to display. Another problem is that it uses the Chado database schema and own software package, while our project uses the GUS database schema and GBrowse framework. It is hard to merge these two systems.

BSR is primarily an analysis tool. It produces output files that can be viewed in gnuplot (http://www.gnuplot.info) and written out as postscript and xfig.

Inspired by the Sybil tool, Mr. Haiming Wang and I have developed our own approach that builds upon the GBrowse software. Features of our solution include:

✧ It can visualize the comparison on the contig-to-contig level based on a WU-

BLASTN alignment result.

- ✧ It can visualize the comparison on the gene-to-gene level with the orthologous groups assigned by the OrthoMCL (48) program.
- ✧ It can highlight some genes that are on the orthologous group, but that are not on the panel region.
- ✧ Different colors are used to display according to the e-value of each comparison match pair. Because most match pairs' e-values are equal to 0, two colors are used currently: pink (=0) and orange ($\neq$0).
- ✧ Some interactive operations are added, for example, a pop-up window is displayed when the user clicks on the comparative region. Detailed alignment information is included in this page including the ClustalW, BlastP, and Promer alignments. Other descriptions related to this match are also shown, such as the name, functionality, and double stranded e-values.
- ✧ Two links are used to turn on or off the comparison visualization function.
- ✧ It is easily integrated into the GBrowse framework without any modification on the BioPerl level.

4.2 Preparing sequence data

C. parvum and *C. hominis* genome sequence data are used by the CryptoDB project. The *C. hominis* genome consists of 18 long contigs and the *C. parvum* genome consists of 1422 short contigs. For each genome sequence, we have three files prepared: the GenBank files (Figure 25), the DNA sequence file in multi-FASTA format (Figure 26), and the protein sequence file in multi-FASTA format (Figure 27).

```
LOCUS       AAEL01000001           90444 bp    DNA     linear   INV 27-OCT-2004
DEFINITION  Cryptosporidium hominis strain TU502 chromosome 3 CHR0013096, whole
            genome shotgun sequence.
ACCESSION   AAEL01000001 AAEL01000000
VERSION     AAEL01000001.1  GI:54659871
KEYWORDS    WGS.
SOURCE      Cryptosporidium hominis
  ORGANISM  Cryptosporidium hominis
            Eukaryota; Alveolata; Apicomplexa; Coccidia; Eimeriida;
            Cryptosporidiidae; Cryptosporidium.
REFERENCE   1  (bases 1 to 90444)
  AUTHORS   Xu,P., Widmer,G., Wang,Y., Ozaki,L.S., Alves,J.M., Serrano,M.G.,
            Puiu,D., Manque,P., Akiyoshi,D., Mackey,A.J., Pearson,W.R.,
            Dear,P.H., Bankier,A.T., Peterson,D.L., Abrahamsen,M.S., Kapur,V.,
            Tzipori,S. and Buck,G.A.
  TITLE     The genome of Cryptosporidium hominis
  JOURNAL   Nature 431, 1107-1112 (2004)
REFERENCE   2  (bases 1 to 90444)
  AUTHORS   Xu,P., Widmer,G., Wang,Y., Ozaki,L.S., Alves,J.M., Serrano,M.G.,
            Puiu,D., Manque,P., Akiyoshi,D., Mackey,A.J., Pearson,W.R.,
            Dear,P.H., Bankier,A.T., Peterson,D.L., Abrahamsen,M.S., Kapur,V.,
            Tzipori,S. and Buck,G.A.
  TITLE     Direct Submission
  JOURNAL   Submitted (08-JUN-2004) Center for the Study of Biological
            Complexity, Virginia Commonwealth University, Trani Center for Life
            Sciences, 1000 W Cary St, Richmond, VA 23298, USA
FEATURES             Location/Qualifiers
     source          1..90444
                     /organism="Cryptosporidium hominis"
                     /mol_type="genomic DNA"
                     /strain="TU502"
                     /db_xref="taxon:237895"
                     /chromosome="3"
     gene            complement(<1..5233)
                     /locus_tag="Chro.30218"
     CDS             complement(<1..5233)
                     /locus_tag="Chro.30218"
                     /codon_start=1
                     /product="hypothetical protein"
                     /protein_id="EAL38445.1"
                     /db_xref="GI:54659884"
                     /translation="MKSRPNGSSTLSNILQQYSDSVSQGEGVGYIGIEWKGPFPLHNS
                     IIQHVKRVQLLIGKTPSLVYGVIPNIKYGYVGLENMLYIFPIEETIQKVLRRESSIGS
                       ......
ORIGIN
        1 tagtactagt aagcttctga gactggtcag cagagtatgc gttttgaata taagctaaag
       61 tactaattaa aaagttattt gccatagaaa tatgattttt attaagtggg aagagtttcc
        ....
```

Figure 25. The sample GenBank format with segment of the *C. hominis* genome.

```
>ChTU502_VII_AAEL01001419
CAAAAAACACAATCTGCCCACTGGACTATGGGCAGATTGTGTTTTTTGATGAATTGCTGTTTCTTGTAGA
ACGTGGAAGCCTACTTCTTTTTTTCATGGAGTCAATCAAGACTAGAATGGGTTCAGACCCCTATTCTCTAT
CTATTTGACGCTTTTATAAGAAATGAAGAGCAATTTAAACTTAATAACTATATTTCTTATTCCAGAATGA
AAAGATCTGGATACAAATTTAAAACTAGTTCAAATGAAAATAAAGTTAGAGTCCAAGATAACATTCACTC
GGCCACAAAATTTTTAAAATGCATTGATGAGGCAGAA

>ChTU502_VIII_AAEL01001420
TTATACTCTAATTCTGGACTGAGATTTATGCAGAATCTCGTAATTTGGACCTGAAGATTTACATTATAAT
ATTGAACAGGCATACAACTTTTTGTAAAGAGTAGCCATATTTATATATATAAATATAAATATATATATAA
ATATATATATTTATATATATATATATATATATATATATATATATATATATATATATATGGATATATATAT
ACAATATGTAGAATGCCAGTTCAATATTATAGTGGAGCGCGTGAGGTCCAAATTACGAGATTCTGCATAA
CTCTCAGACCAGAA

>ChTU502_VIII_AAEL01001421
TCACTTCTATCCTTCTAGACCATCTCAAAGCTAATAATCCGGAAGATTCTGTCCTCCAAACAAAACTACT
TGAAGTTAATCTCCTTCATGCTCCACAAGTTGCTGAAGCCCTTTTCCAAATGGACCTTTTCACTCATTAT
GATAAACATGCAATTGCCGCCCTCTGCGAAAAGGCGGGCTCTTGAGAATTTCTCTGACATGCGTGACATA
AGGAGAATATTAGGAGTAGCTTGTGGGAGTTTAAATACCGATTGGTT

>ChTU502_VI_AAEL01001422
TAAGAAAATTGATCGCAAAAAGGTTGATATTTAAGTTATTAGAATAAGATATGATGAATTAATAATATAA
AGTTCTCGTCATCGAGTGCTGATGCATAGAAAAAAAAAGGAAATGAGTACAATCTAAATTCTGTTGAAGT
ATCTAATAATAGTAGTAGATCCATAATTCGTTTTTGGTTTAGAGATCAGAAACCAATTTGGGATGAAGAA
GACTATTTTGTGAAGAGTGAATTGTTTAGGTCAACCCCAGA
```

Figure 26. The sample multi-FASTA format with segment of the *C. hominis* DNA sequence.

```
>ChTU502_EAL37859
MEIKIFWLLLFCVKFAREVILTTGNENIYSGLVKELGDEQDLDQTKIIDSIRKIGSDTQHTEPFQPLLDA
SEEKQDEKDSIIDAAKVASTRKSESLPTSAEKIDLIGAGVSDSRKDTGARASGPDFKFGEPGPSIREEEL
EMYVHPSEELIFSIPIPTMKPEHQFNTIRRLSGSKYRGLPKAHQNERGNVNWCKVIATREPEIMTKVNNI
YRMIQSDLVSVLGGIGLDGSSSSQGYSGSRAGSGWTQQHICEGIYLLNTYRGEGDCERIFVDLMARFHGQ
WSLFYKMFRQKFRQICKEAGYSTGRWDVYNDRAYSQYAVLRSAKKPIKTEPTKERREFVRLLRTRSGICS
LLPQNLREKAIQLKTSVDSIPQIKHITQRISLSDYCDIILGKQSIIECAEALITFFKLKAETYYDSTERY
RPIRSLLIKACDDALFHEKDSNTRPEKVSQYDRMENKE

>ChTU502_EAL35611
MAKLYFYYSAMNAGKSTVLLQSSFNYQERGMKTILFTISKDCRFKEGSICSRIGLSEKAHTFTPDLKLLD
IINQENNKEKIDCVLVDESQFLTKFQVRELCIVVDKLDIPVLCYGLRTDFRGNLFEGSKYLLAWADKLTE
IKTICRCGKKATMTIRLNSNGEPVFSGEQILIGDNSIYTSVCRKHHIISCEEYNF

>ChTU502_EAL38445
MKSRPNGSSTLSNILQQYSDSVSQGEGVGYIGIEWKGPFPLHNSIIQHVKRVQLLIGKTPSLVYGVIPNI
KYGYVGLENMLYIFPIEETIQKVLRRESSIGSSTERRSTDRMRSAPSIFEQGFDGKPASDESILTIVFPF
SIKNMTGTFPRVGIFTSDVEYLLCVITEKSVHLLALKFDNFDLAGDCSQAYDNDRTVPRGAGLIKVPIMN
IRGKDIGMLQCSLPGSHNSKFHSLQGTLDGRFFFLEENSSSIHELVYQSSEGWITPYCYIHSHQIYSSFT
...
```

Figure 27. The sample multi-FASTA format with segment of the *C. hominis* protein sequence.

These files are used for the next step. The DNA sequence files are used by the WU-BLASTN (25) analysis at the contig level and the protein sequence files are used by the OrthoMCL (48) program to find the orthologous group on the gene level. Because the gene features' names in the CryptoDB project use the value of *locus_tag* in the GenBank file but in the protein sequence file the gene features' names use the value of *protein_id*, we need to combine the *protein_id* and *locus_tag* value together (Figure 29) in order to facilitate the OrthoMCL analysis.

```
>ChTU502_EAL37859_Chro.60143
MEIKIFWLLLFCVKFAREVILTTGNENIYSGLVKELGDEQDLDQTKIIDSIRKIGSDTQHTEPFQPLLDA
SEEKQDEKDSIIDAAKVASTRKSESLPTSAEKIDLIGAGVSDSRKDTGARASGPDFKFGEPGPSIREEEL
EMYVHPSEELIFSIPIPTMKPEHQFNTIRRLSGSKYRGLPKAHQNERGNVNWCKVIATREPEIMTKVNNI
YRMIQSDLVSVLGGIGLDGSSSSQGYSGSRAGSGWTQQHICEGIYLLNTYRGEGDCERIFVDLMARFHGQ
WSLFYKMFRQKFRQICKEAGYSTGRWDVYNDRAYSQYAVLRSAKKPIKTEPTKERREFVRLLRTRSGICS
LLPQNLREKAIQLKTSVDSIPQIKHITQRISLSDYCDIILGKQSIIECAEALITFFKLKAETYYDSTERY
RPIRSLLIKACDDALFHEKDSNTRPEKVSQYDRMENKE

>ChTU502_EAL35611_Chro.50398
MAKLYFYYSAMNAGKSTVLLQSSFNYQERGMKTILFTISKDCRFKEGSICSRIGLSEKAHTFTPDLKLLD
IINQENNKEKIDCVLVDESQFLTKFQVRELCIVVDKLDIPVLCYGLRTDFRGNLFEGSKYLLAWADKLTE
IKTICRCGKKATMTIRLNSNGEPVFSGEQILIGDNSIYTSVCRKHHIISCEEYNF

>ChTU502_EAL38445_Chro.30218
MKSRPNGSSTLSNILQQYSDSVSQGEGVGYIGIEWKGPFPLHNSIIQHVKRVQLLIGKTPSLVYGVIPNI
KYGYVGLENMLYIFPIEETIQKVLRRESSIGSSTERRSTDRMRSAPSIFEQGFDGKPASDESILTIVFPF
SIKNMTGTFPRVGIFTSDVEYLLCVITEKSVHLLALKFDNFDLAGDCSQAYDNDRTVPRGAGLIKVPIMN
IRGKDIGMLQCSLPGSHNSKFHSLQGTLDGRFFFLEENSSSIHELVYQSSEGWITPYCYIHSHQIYSSFT
...
```

Figure 28. The sample renamed multi-FASTA file with segment of the *C. hominis* protein sequence.

4.3 Generating and formatting alignment

Because the *C. parvum* sequence has 18 long contigs, and the *C. hominis* sequence has 1422 short contigs, we set the *C. parvum* genome as the reference sequence and the *C. hominis* genome as the subject sequence. In order to implement the comparative visualization part of the CryptoDB project on the gene level with this specific sequence data, we perform two steps:

- ✧ Run WU-BLASTN (25) on the contig level with the DNA sequence data;
- ✧ Run OrthoMCL with the protein sequence data;

The WU-BLAST (Washington University Basic Local Alignment Search Tool) program finds regions of sequence similarity or homology quickly, with minimum loss of sensitivity. It includes blastp, blastn, blastx, tblastn, and tblastx programs, all based on the BLAST algorithm.

- ✧ **Blastp**: compares an amino acid query sequence with a protein sequence database.
- ✧ **Blastn**: compares a nucleotide query sequence with nucleotide sequence database.
- ✧ **Blastx**: converts a nucleotide query sequence into protein sequences in all 6 reading frames and then the translated protein products are compared with the protein sequence databases.
- ✧ **Tblastn**: compares a protein query sequence with a nucleotide sequence database and dynamically translates in all six reading frames (both strands).
- ✧ **Tblastx**: compares the six-frame translations of a nucleotide query sequence with the six-frame translations of a nucleotide sequence database.

Because the available *C. parvum* and *C. hominis* sequence on the contig level is nucleotide sequence, we use the WU-BLASTN program. First, format the *C. parvum* nucleotide sequence in order to create a BLAST database. The command is as follows: “formatdb –i *parvum*

–p F –o T". Second, use the WU-BLASTN program to obtain alignments by the command: "blastn *parvum* hominis <options>". The option parameters can be specified according to user requirements or set as default.

The OrthoMCL tool constructs ortholog and paralog groups using a Markov cluster algorithm (MCL) (19) (http://micans.org/mcl/). Orthologs evolve from a common ancestor by speciation. Paralogs are related by duplication events within a genome (53, 54). Orthologs typically involve comparison between two species and paralogs typically involve comparisons within a species. OrthoMCL generates clusters of proteins in which each cluster consists of orthologs or paralogs from at least two species (48). The detailed procedures include (Figure 29):

- ✧ It starts with all-against-all BLASTP local alignment of the protein sequences.
- ✧ The putative orthologs are identified between pairs of genomes by best similarity pairs, and for each putative ortholog, some paralogs are identified within the same genome.
- ✧ These orthologs and paralogs are converted into a graph in which the nodes represent protein sequences, and the weighted edges represent each orthologs or paralogs. Then a symmetric matrix is used to represent this graph.
- ✧ The MCL algorithm is applied. It considers all relationships in the graph and symmetric matrix and separates diverged paralogs and distant orthologs by clustering.

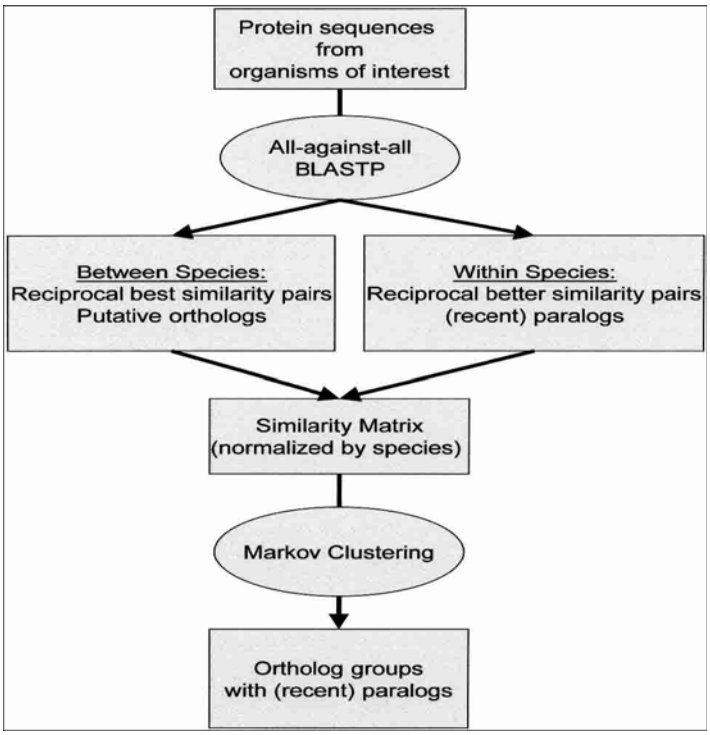


Figure 29. Flow chart of the OrthoMCL procedure (47).

Three major output files are generated by the OrthoMCL tool: *all_orthomcl.out*, *all_blast.bbh*, and *all.blast*.

The *all_orthomcl.out* file (Figure 30) lists all ortholog groups with at least two species. For instance, for the first line: the "*ORTHOMCL3"* means the id of this group; the *"4 genes"* means fours genes are included in this group; the "*2 taxa"* means two kinds of species; the remaining part of each line lists the member of this group.

```
ORTHOMCL3(4 genes,2 taxa):    ChTU502_EAL35925_Chro.10120(ChTU502_protein_rename) ChTU502_EAL36266_Chro.80274(ChTU502_protein_rename) cgd1_990 ...
ORTHOMCL29(3 genes,2 taxa):   ChTU502_EAL36831_Chro.70267(ChTU502_protein_rename) ChTU502_EAL37169_Chro.70266(ChTU502_protein_rename) cgd7_2340...
ORTHOMCL32(2 genes,2 taxa):   ChTU502_EAL34582_Chro.50358(ChTU502_protein_rename) cgd5_350(c.parvum.with.409.protein)
ORTHOMCL33(2 genes,2 taxa):   ChTU502_EAL34584_Chro.80262(ChTU502_protein_rename) cgd8_2230(c.parvum.with.409.protein)
```

Figure 30. The sample segment of *all_orthomcl.out* file.

The *all_blast.bbh* file (Figure 31) lists each pair in all groups along with two stranded e-values. For example: for the each line, the first two segments are the names of each gene in the pair; the remaining two segments are the two e-values of these genes. The lower the e-value, the more "significant" the match is.

```
ChTU502_EAL35454_Chro.50011   ChTU502_EAL35955_Chro.50010   2e-28 5e-29
cgd1_2060   cgd8_1040   8e-53 1e-52
ChTU502_EAL34582_Chro.50358   cgd5_350    6e-48 3e-47
```

Figure 31. The sample segment of *all_blast.bbh* file.

The *all.blast* file (Figure 32) lists the detailed alignment information for each group that consists of one or more match pairs.

```
BLASTP 2.2.11 [Jun-05-2005]

Reference: Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer,
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997),
"Gapped BLAST and PSI-BLAST: a new generation of protein database search
programs",  Nucleic Acids Res. 25:3389-3402.

Query= ChTU502_EAL38201_Chro.10153
         (52 letters)

Database: ./Oct_13/tmp/all.fa
           7679 sequences; 4,025,839 total letters

Searching..................................................done

                                                                  Score    E
Sequences producing significant alignments:                       (bits) Value

cgd1_1320 |  | developmental protein, putative                        68   4e-13
ChTU502_EAL38201_Chro.10153                                          68   4e-13

>cgd1_1320 |  | developmental protein, putative
          Length = 188

 Score = 67.8 bits (164), Expect = 4e-13
 Identities = 36/52 (69%), Positives = 36/52 (69%)

Query: 1  MGNRISIDDSIFELKLQKKELERQYNXXXXXXXXXXXXXXXXIISGKADLSK 52
          MGNRISIDDSIFELKLQKKELERQYN                IISGKADLSK
Sbjct: 1  MGNRISIDDSIFELKLQKKELERQYNKRDRDSKSERKKAKDAIISGKADLSK 52


>ChTU502_EAL38201_Chro.10153
          Length = 52

 Score = 67.8 bits (164), Expect = 4e-13
 Identities = 36/52 (69%), Positives = 36/52 (69%)

Query: 1  MGNRISIDDSIFELKLQKKELERQYNXXXXXXXXXXXXXXXXIISGKADLSK 52
          MGNRISIDDSIFELKLQKKELERQYN                IISGKADLSK
Sbjct: 1  MGNRISIDDSIFELKLQKKELERQYNKRDRDSKSERKKAKDAIISGKADLSK 52


  Database: ./Oct_13/tmp/all.fa
    Posted date:  Oct 13, 2005  9:41 PM
```

Figure 32. The sample segment of *all.blast* file.

We analyze and format the *all_orthomcl.out* and *all_blast.bbh* files in order to generate a new file named *comparison_data* file (Figure 33). Each line consists of one *C. hominis* and *C.*

parvum gene that are in the same ortholog group. It extracts the pairs in both species and removes those pairs in only one species. Then, we parse the *all.blast* file into many sub-files with detailed text alignments for each pair (Figure 34).

```
Chro.50358--cgd5_350--6e-48--3e-47
Chro.80262--cgd8_2230--9e-59--1e-58
Chro.50355--cgd5_380--1e-50--4e-50
Chro.20414--cgd2_3880--1e-43--2e-42
Chro.60042--cgd6_300--3e-70--4e-70
```

Figure 33. The sample of the *comparison_data* file. The first column means the *C. hominis* genes and the second column means the *C. parvum* genes. The third and fourth columns mean the two stranded e-values. Each line shows one match pair between *C. hominis* genes and *C. parvum* genes with the corresponding e-values.

```
Query= ChTU502_EAL38201_Chro.10153

Subject= cgd1_1320

>cgd1_1320
    Length = 188

 Score = 67.8 bits (164), Expect = 4e-13
 Identities = 36/52 (69%), Positives = 36/52 (69%)

Query: 1  MGNRISIDDSIFELKLQKKELERQYNXXXXXXXXXXXXXXXXIISGKADLSK 52
          MGNRISIDDSIFELKLQKKELERQYN                IISGKADLSK
Sbjct: 1  MGNRISIDDSIFELKLQKKELERQYNKRDRDSKSERKKAKDAIISGKADLSK 52
```

Figure 34. The sample of detailed text alignments for each pair. The first line shows the query *C. hominis* gene and the second line shows the subject *C. parvum* genes. The following lines show the detailed text alignment.

In addition, we run the other two alignment tools: ClustalW and Promer. The results are shown in Figure 35 and Figure 36.

```
 CLUSTAL W (1.83) multiple sequence alignment


cgd6_770        MSYSNNANQITCQCGKQDRLLPVLPQSSNSRMMRMIQQQGCGCMTPSSLNNNQNYYFDNQ 60
Chro.60100      MSYSNNANQITCQCGKQDRLLPALPQSSNSRMMRMIQQQGCGCMTPSSLNNNQNYYFDNQ 60
                **********************.************************************

cgd6_770        SNISGCLFCQRLQHCCICDFSRRTGTGMMNQGNMFCSCCTCPMQQQFPSCQCGTNYFNTQ 120
Chro.60100      SNISGCLFCQRLQHCCICDFSRRTGTGMMNQGNMFCSCCTCPMQQQFPSCQCGTNYFNTQ 120
                ************************************************************

cgd6_770        HSNPVTPQQAQITEITNSNNGTMNGNVVGNGLDNTIPNNSTPGNVLVSTNLNSIQNQGQQ 180
Chro.60100      HSNPVTPQQAQITEITNSNNGTMNGNVVGNGLDNTIPNNSTPGNVLVSTNLNSIQNQGQQ 180
                ************************************************************

cgd6_770        ITTQSELLNLKNTSINPEMINIALSGQTPESLSISIDSKGVITLGVNNHPAPVSNIPTTT 240
Chro.60100      ITTQSELLNLKNTSINPEMINIALSGQTPESLSISIDSKGVITLGVNNHPAPVSNIPTTT 240
                ************************************************************

cgd6_770        NSNEESRNRIRNALGDRH 258
Chro.60100      NSNEESRNRIRNALGDRH 258
                ******************
```

Figure 35. The sample of ClustalW alignment output. The first line displays the version of the ClustalW alignment and the following lines shows the detailed text alignment.

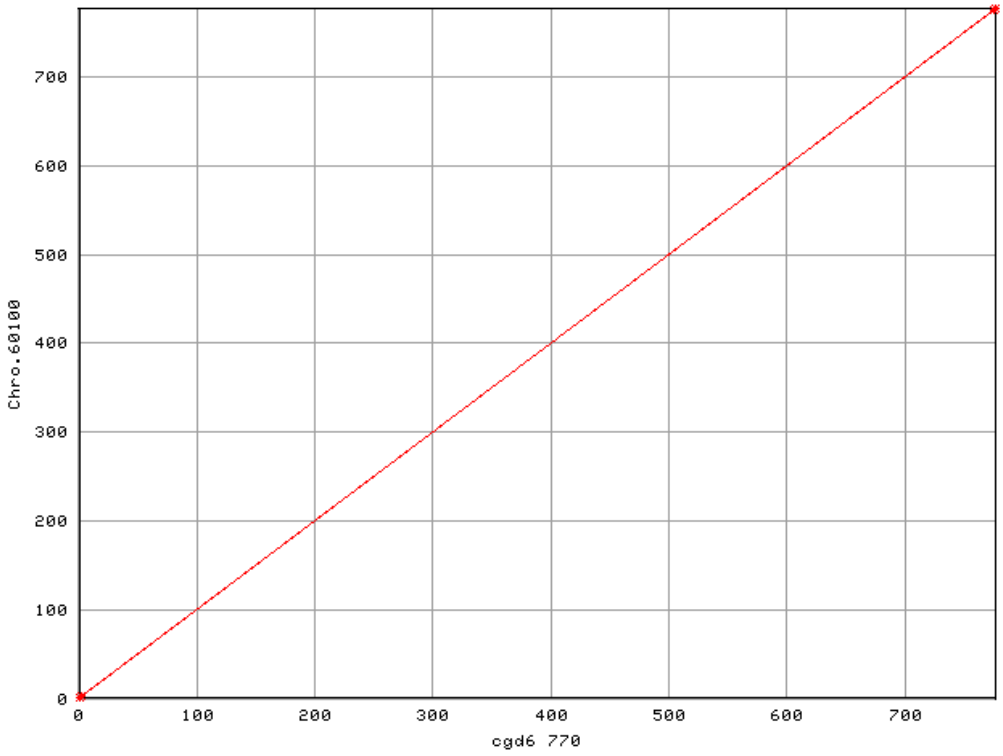


Figure 36. A sample picture of the Promer alignment output by the mummerplot program. The *X*-axis shows the *C. parvum* gene and the *Y*-axis shows the *C. hominis* gene. The red line is composed of the match points between two genes. This example means a perfect match between the *C. parvum* gene (*cgd6_770)* and the *C. hominis* gene (*Chro.60100).*

4.4 Visualization approach

In order to satisfy our requirements with minimal modifications to the CryptoDB project, which uses the GBrowse software as the genome features' graphical viewer, we have designed and implemented a comparison visualization approach that can be easily integrated into the GBrowse framework. Below, we introduce the GBrowse framework.

GBrowse is a web-based application for displaying genomic annotations and other features. It permits scrolling and zooming through specific regions of a genome, searching for a landmark or specific genes, performing a full text search of all features, highlighting specific genes or regions, enabling and disabling tracks and changing the relative track order and track background color.

The GBrowse framework consists of three layers (Figure 37):

- ✧ The first level is a CGI (Common Gateway Interface) script named gbrowse, which is responsible for managing the user interface, including generating the HTML forms, accepting and processing requests, managing the cookies, and displaying the rendered images of annotated regions.
- ✧ The second level is a BioPerl library (55), which connects the CGI script and the underlying databases or flat files. The *Graphics* BioPerl module is responsible for generating the genome images. The other BioPerl modules are used for connecting the database or flat files by adapters. Figure 38 lists several major BioPerl Modules.
- ✧ The third level is a relational database including MySQL and Oracle or flat files in GFF format that is responsible for storing genome information. The BioPerl modules can access these databases or flat files by adapters.

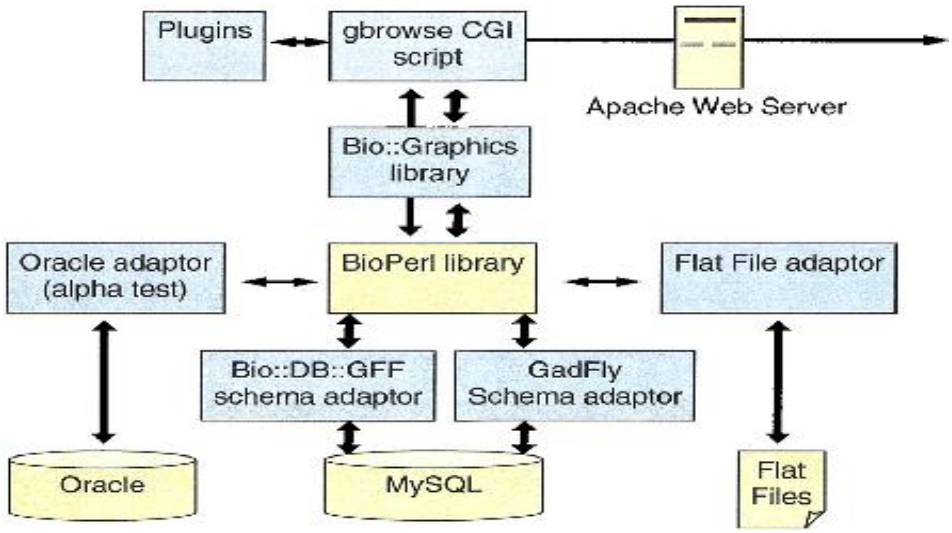


Figure 37. The GBrowse Framework (56)

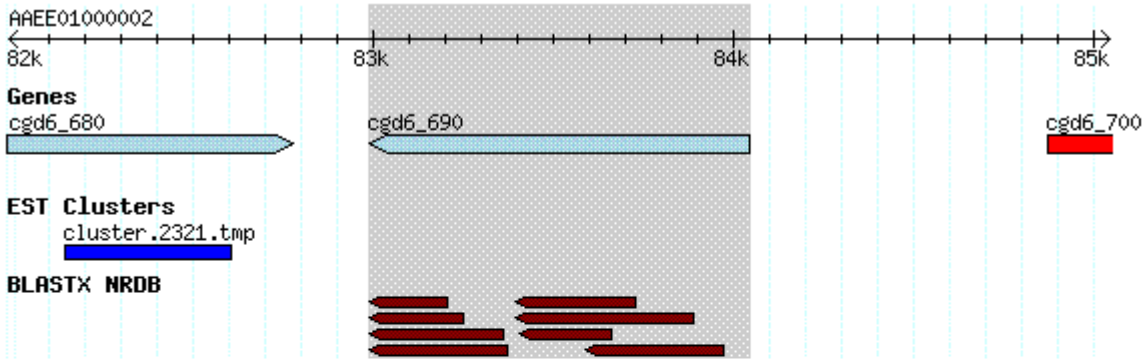
Table 1. Major Bioperl Module Groups

Modules	Description
Bio::Seq	Sequences and their properties
Bio::SeqIO	Sequence data input/output
Bio::Index	Flat-file sequence database indexing and retrieval
Bio::DB	Remote database access for sequences and references via HTTP
Bio::DB::GFF	SQL GFF database for DAS and GBrowse backends
Bio::SeqFeature	Annotations or features that have a sequence location
Bio::Annotation	Generic annotations such as Comments and References
Bio::AlignIO, Bio::SimpleAlign	Multiple sequence alignments and their Input/Output
Bio::LiveSeq, Bio::Variation	Sequence variations and mutations
Bio::Search, Bio::SearchIO	Sequence database searches and their Input/Output
Bio::Tools	Miscellaneous analysis tools
Bio::Tools::Run	Wrapper for executing local and remote analyses
Bio::Tree, Bio::TreeIO	Phylogenetic trees and their Input/Output
Bio::Structure	Protein structure data
Bio::Map, Bio::MapIO	Biological maps and their Input/Output
Bio::Biblio, Bio::DB::Biblio	Bibliographic References and Database retrieval
Bio::Graphics	Graphical displays of sequences

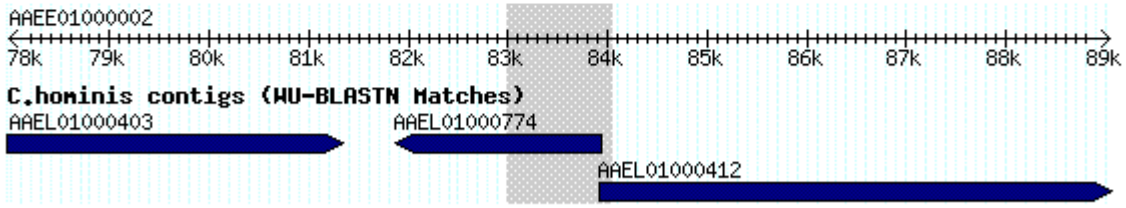
Figure 38. Several major BioPerl modules and description (55)

Some images are generated by GBrowse for the CryptoDB project (Figure 39). In figure 39(A), the gene features and corresponding EST and BLASTX features are displayed based on the contig overview. For example, figure 39(A) shows the *C. parvum* contig (*AAEE01000002*) ranging from 82kbp to 85kbp on the top overview arrow. Under the contig arrow, three tracks are displayed: Genes, EST Clusters, and BLASTX NRDB. The grey shadow area shows the gene

searched by the user. In figure 39(B), the comparison on the contig level between the *C. parvum* and *C. hominis* is displayed. For example, figure 39(B) shows the *C. parvum* contig ranging from 78kbp to 89kbp on the top overview arrow. Under this arrow, the compared *C. hominis* contigs, including *AAEL01000403*, *AAEL01000774*, and *AAEL01000412*, are displayed corresponding to the *C. parvum* contig coordinates. The grey shadow area indicates that the located gene is found in this area.



(A) DNA Context



(B) Genome Comparison - BLASTN of *C. hominis* contigs vs *C. parvum* contigs

Figure 39. The sample images are generated by GBrowse for the CryptoDB project. (A) It shows the single genome features display including the contig, genes, EST Cluster, and BLASTX. (B) It shows the comparison on the contig level between the *C. parvum* and *C. hominis* genome. (http://cryptodb.org)

Our comparison visualization approach aims to display the comparison on the gene level. It includes five steps (Figure 40):

- ✧ First, we run WU-BLASTN (25) on the contig level and load the alignment data into the GUS database. The GUS database adaptors have been implemented by Mr.

Haiming Wang. Figure 41 depicts the contig-level comparative visualization. The top overview arrow shows one *C. parvum* contig with the region ranging from 102kbp to 121kbp. By running the WU-BLASTN tool, the two bottom overview arrows of the *C. hominis* contigs (*AAEL01000137* and *AAEL01000498*) on the grey track are displayed. For instance, the *AAEL01000137 C. hominis* contig ranging from 2kbp to 16kbp matches the top *C. parvum* contig ranging from 102kbp to 116kbp according to the WU-BLASTN alignment.

- ✧ Second, we add the gene features to each contig of both genomes. Each gene has a start point and stop point base on the contig. This information is stored in the GUS database and then retrieved by the GUS adapter. The gene features of each contig are added under the overview arrow of the *C. parvum* and above the overview arrows of the *C. hominis* of figure 41 to form figure 42.
- ✧ Third, we run OrthoMCL to obtain the comparison pairs between *C. parvum* and *C. hominis* genes. Figure 33 shows the sample comparison pairs file.
- ✧ Fourth, we run several functions that we have implemented into the GBrowse CGI script, without any modification on the BioPerl level. From figure 42, we see that many genes of both genomes are shown, but we do not know which genes are orthologs. Because the orthologs match pairs on the gene level have been obtained by the third step, our functions are used to connect the genes of the orthologs pairs. If the region of the panel viewed by the user has changed, our functions still can connect the orthologs genes dynamically. If one of the orthologs gene pair is not in the current region of the panel, it is highlighted. See appendix A, B, C, and D. Appendix A shows the *get_matches* function that obtains all orthologs pair from the

comparison match pairs file. Currently we store it into a flat file instead of the Oracle database. Appendix B shows the *get_glyph_coord* function that obtains the glyph coordinates of the specific gene feature. Appendix C shows the *highlight_feature* function that obtains the coordinates of the gene feature that its matched orthologs genes are not shown in the current panel. Appendix D shows the *draw_comparison_line* function that draws the shadow areas according to the coordinates by the *get_glypy_coord* and *highlight_feature* function. With other minor functions, figure 43 of the sample of comparison visualization is generated.

- ✧ Fifth, add the image-maps into each comparison area and highlight area. Each image-map popup shows a detailed alignments window (Figure 44). We implemented this function in the *Browser.pm* module of GBrowse. In addition, the results of other alignment approaches are added into this popup window.

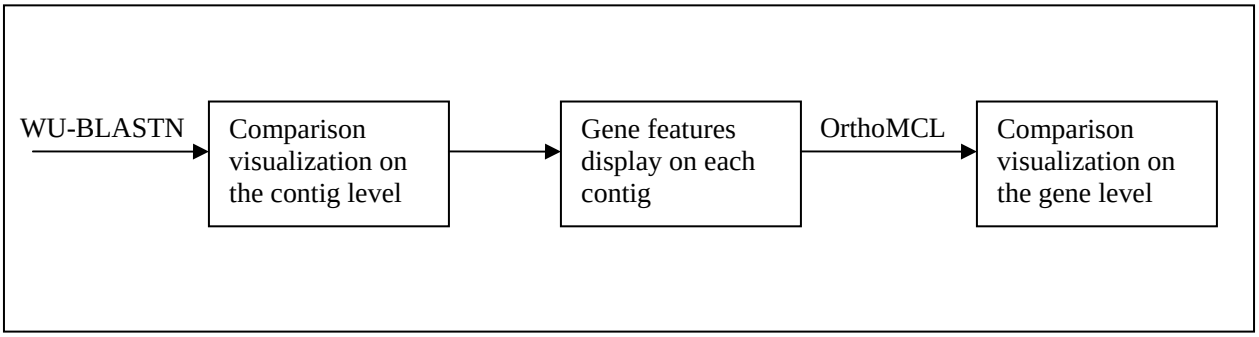


Figure 40. The flow chart of comparison visualization on the gene level based on our approach.

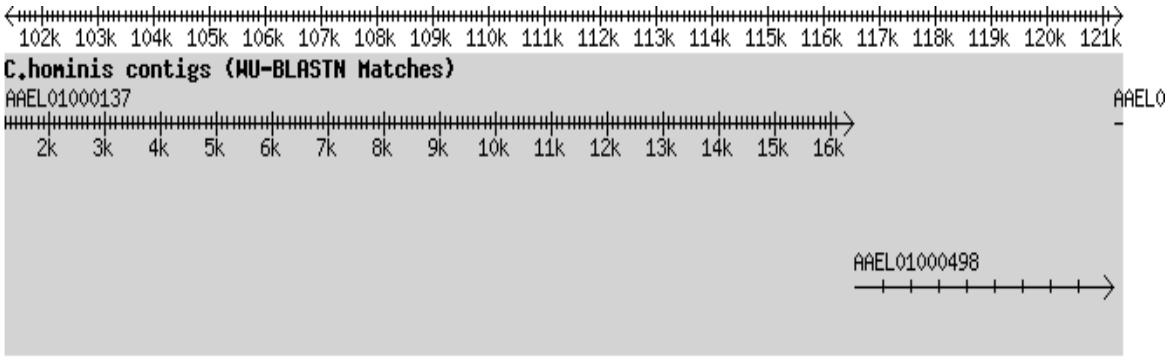


Figure 41. The sample diagram of contig-level comparative visualization by the GBrowse. The top overview arrow shows one *C. parvum* contig with the region ranging from 102kbp to 121kbp. By running the WU-BLASTN tool, the two bottom overview arrows of the *C. hominis* contigs on the grey track are displayed. For instance, the *AAEL01000137 C. hominis* contig ranging from 2kbp to 16kbp matches the top *C. parvum* contig ranging from 102kbp to 116kbp.

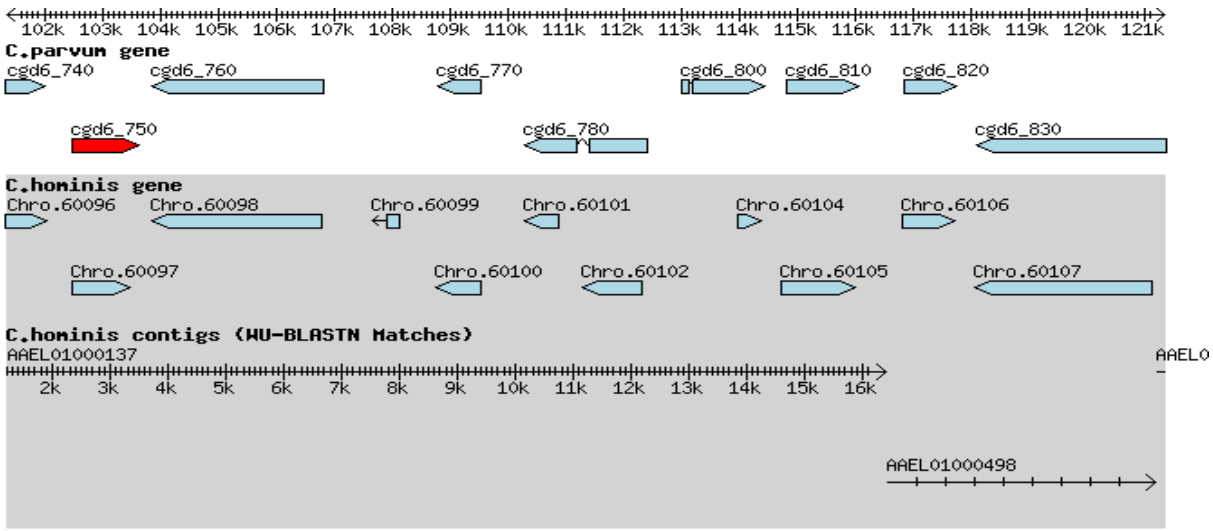


Figure 42. The sample diagram of contig-level comparison viewer with gene features. Based on figure 37, the gene features of each contig are added below the overview arrow of the *C. parvum* and above the overview arrow of the *C. hominis*.

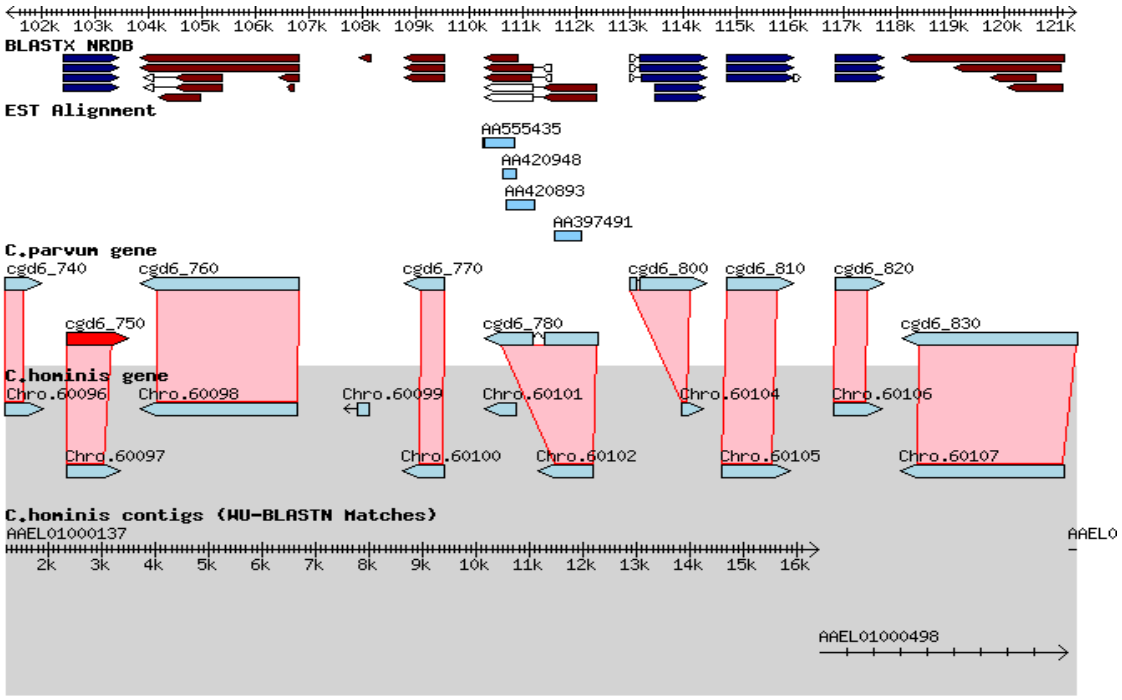


Figure 43. The sample diagram of gene-level comparison viewer. The orthologous genes are connected based on figure 37.

Print this page | Close

CryptoDB.org

C.parvum---cgd6_770	hypothetical protein
C.hominis---Chro.60100	hypothetical protein
E-value:	1e-134 <---> 1e-134
Detailed alignment:	Clustalw | Blastp | Promer

ClustalW Top

```
 CLUSTAL W (1.83) multiple sequence alignment

cgd6_770        MSVSNNANQITCQCGKQDRLLPVLPQSSNSRMMRMIQQQGCGCMTPSSLNNNQNVYFDNQ 60
Chro.60100      MSYSNNANQITCQCGKQDRLLPALPQSSNSRMMRMIQQQGCGCMTPSSLNNNQNYYFDNQ 60
                ********************.*************************************

cgd6_770        SNISGCLFCQRLQHCCICDFSRRTGTGMMNQGNMFCSCCTCPMQQQFPSCQCGTNYFNTQ 120
Chro.60100      SNISGCLFCQRLQHCCICDFSRRTGTGMMNQGNMFCSCCTCPMQQQFPSCQCGTNYFNTQ 120
                ************************************************************

cgd6_770        HSNPVTPQQAQITEITNSNNGTMNGNVVGNGLDNTIPNNSTPGNVLVSTNLNSIQNQGQQ 180
Chro.60100      HSNPVTPQQAQITEITNSNNGTMNGNVVGNGLDNTIPNNSTPGNVLVSTNLNSIQNQGQQ 180
                ************************************************************

cgd6_770        ITTQSELLNLKNTSINPEMINIALSGQTPESLSISIDSKGVITLGVNNHPAPVSNIPTTT 240
Chro.60100      ITTQSELLNLKNTSINPEMINIALSGQTPESLSISIDSKGVITLGVNNHPAPVSNIPTTT 240
                ************************************************************

cgd6_770        NSNEESRNRIRNALGDRH 258
Chro.60100      NSNEESRNRIRNALGDRH 258
                ******************


For the detailed, editable alignment, please click the button
```

Blastp Top

```
 Query= CpIOWA_EAK89767_cgd6_770

Subject= ChTU502_EAL36676_Chro.60100

>ChTU502_EAL36676_Chro.60100
          Length = 258

 Score =  473 bits (1218), Expect = e-134
 Identities = 229/258 (88%), Positives = 229/258 (88%)

Query:  1    MSYSNNANQITCQCGKQDRLLPVLPQSSNSRMMRMIQQQGCGCMTPSSLNNNQNYYFDNQ 60
            MSYSNNANQITCQCGKQDRLLP LPQSSNSRMMRMIQQQGCGCMTPSSLNNNQNYYFDNQ
Sbjct:  1    MSYSNNANQITCQCGKQDRLLPALPQSSNSRMMRMIQQQGCGCMTPSSLNNNQNYYFDNQ 60

Query:  61   SNISGCLFCQRLQHCCICDFSRRTGTGMMNQGNMFCSCCTCPMQQQFPSCQCGTNYFNTQ 120
            SNISGCLFCQRLQHCCICDFSRRTGTGMMNQGNMFCSCCTCPMQQQFPSCQCGTNYFNTQ
Sbjct:  61   SNISGCLFCQRLQHCCICDFSRRTGTGMMNQGNMFCSCCTCPMQQQFPSCQCGTNYFNTQ 120

Query:  121  HSNPVTPQQAQITEIXXXXXXXXXXXXXXXXLDNTIPNNSTPGNVLVSTNLNSIQNQGQQ 180
            HSNPVTPQQAQITEI                 LDNTIPNNSTPGNVLVSTNLNSIQNQGQQ
Sbjct:  121  HSNPVTPQQAQITEITNSNNGTMNGNVVGNGLDNTIPNNSTPGNVLVSTNLNSIQNQGQQ 180

Query:  181  ITTQSELLNLKNTSINPEMINIALSGQTPESLSISIDSKGVITLGVNNHPAPVSNIPTTT 240
            ITTQSELLNLKNTSINPEMINIALSGQTPESLSISIDSKGVITLGVNNHPAPVSNIPTTT
Sbjct:  181  ITTQSELLNLKNTSINPEMINIALSGQTPESLSISIDSKGVITLGVNNHPAPVSNIPTTT 240

Query:  241  XXXXXXXXXXXXALGDRH 258
                        ALGDRH
Sbjct:  241  NSNEESRNRIRNALGDRH 258
```

Promer (PROtein MUMmer) version 3.06 Top

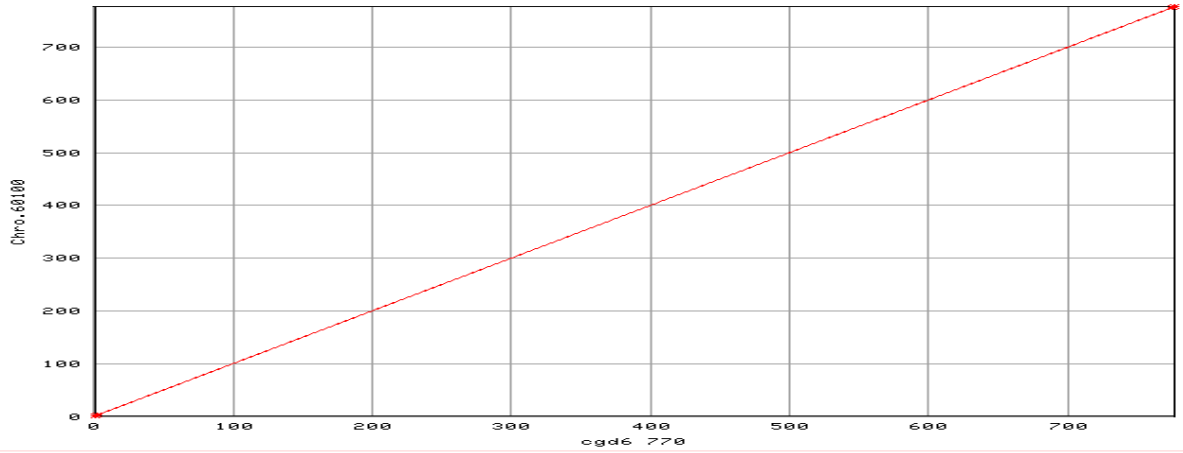


Figure 44. An example of the popup window with detailed protein alignments. It displays the ClustalW, BlastP and PROmer protein alignments. For PROmer alignment, the *X*-axis indicates the *C. parvum* gene and *Y*-axis indicates the *C. hominis* gene. (http://cryptodb.org)

CHAPTER 5

ANALYSIS BASED ON COMPARISON RESULT

In this chapter, we demonstrate the utility of comparative genomic visualization generated by our approach. In figure 45, the rectangle boxes display the synteny blocks and the ellipses show the candidate insertion and deletion genes. However, because the *C. parvum* and *C. hominis* genomes are long and consist of numerous genes, it is time-consuming work to manually find all these candidate genes and synteny blocks by selecting the regions continuously throughout the whole *C. parvum* genome and locating the candidate genes and synteny blocks on the comparison visualization (Figure 45). Thus, our goal is to design a batch approach to accurately and quickly search for all candidate insertion and deletion genes and synteny blocks. For the *C. parvum* genome, 93 candidate insertion or deletion genes are found. For the *C. hominis* genome, 68 candidate genes are found. Also we have added the links for each kind of genes to display the corresponding comparison visualization. The visualizations of the candidate insertions and deletions are useful in helping the scientist to verify their existence.

Future work will focus on analyzing the candidate insertion and deletion genes, removing those “noisy” candidate genes, and then merging those blocks separated by the “noisy” candidate genes, and also finding the rearrangements events.

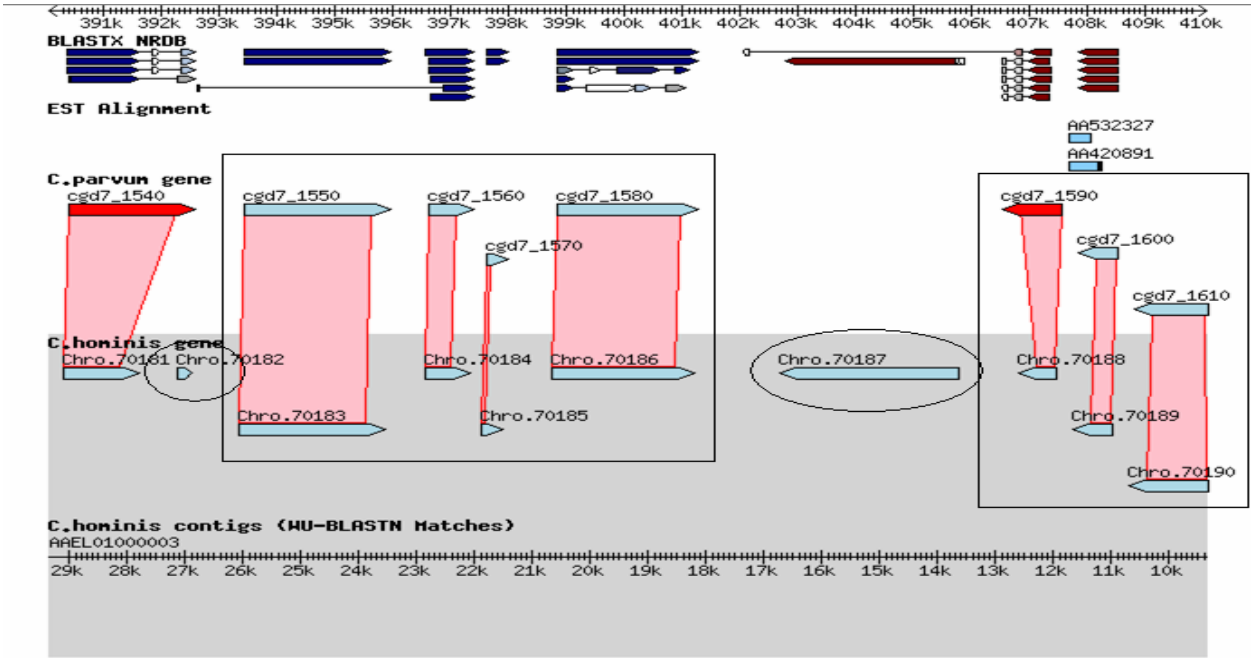


Figure 45. The sample picture generated by our comparison approach. The rectangle boxes mean the synteny blocks of genes between the *C. parvum* and *C. hominis* genomes. The ellipse boxes show the candidate insertion and deletion genes of the *C. parvum* and *C. hominis* genomes.

Three steps of our analysis approach include:

- ✧ First, prepare several files: the ordered genes list file under each contig of the *C. parvum* genome (Figure 46(A)), the ordered genes list file under each contig of the *C. hominis* genome (Figure 46(B)), and the comparison matches pairs file between *C. hominis* and *C. parvum* genomes (Figure 46(C)).
- ✧ Second, generate the numbered matches file for each genome. (1) Paste each match pair to the ordered *C. parvum* genes list file (Figure 47(A)). (2) Put the same number in the front of each of these matches, if two or more consecutive match pairs exist. The number is increased by 1 (Figure 47(B)). (3) Apply step (1) and step (2) into the ordered *C. hominis* genes list files to obtain another half-numbered file (Figure 47 (C)). (4) Number the right column of *C. hominis genes* according to the number of

the *C. hominis* half-numbered file on the *C. parvum* half-numbered file (Figure 47 (D)). Vice versa.

- ✧ Third, by two analysis files on the second step, we can summary the total candidate insertion and deletion genes in the *C. parvum* and *C. hominis* genomes. And we can summary how may synteny blocks and how many genes in each synteny block.

Several examples (Figure 48-50) are provided to learn how to find the candidate insertion and deletion genes and synteny blocks from the analysis files. Thus, the users can quickly locate those candidate genes they are interested in. In addition, we can use our approach to verify whether the gene features are displayed correctly.

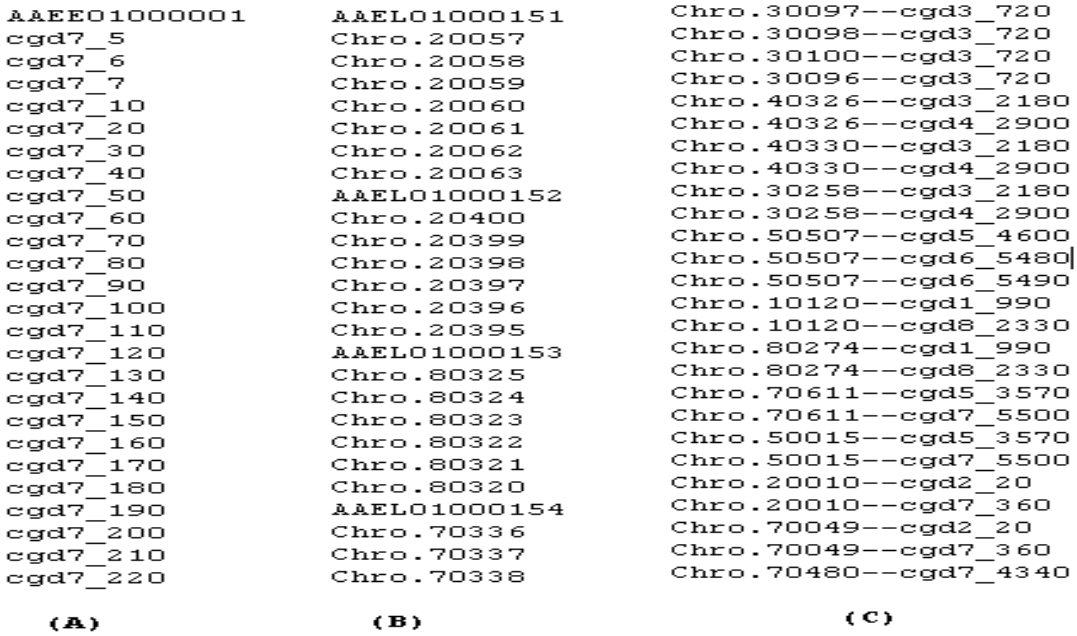


Figure 46. Three files for analysis: (A) *C. parvum* ordered genes list file; (B) *C. hominis* ordered genes list file; (C) the gene comparison pair file.

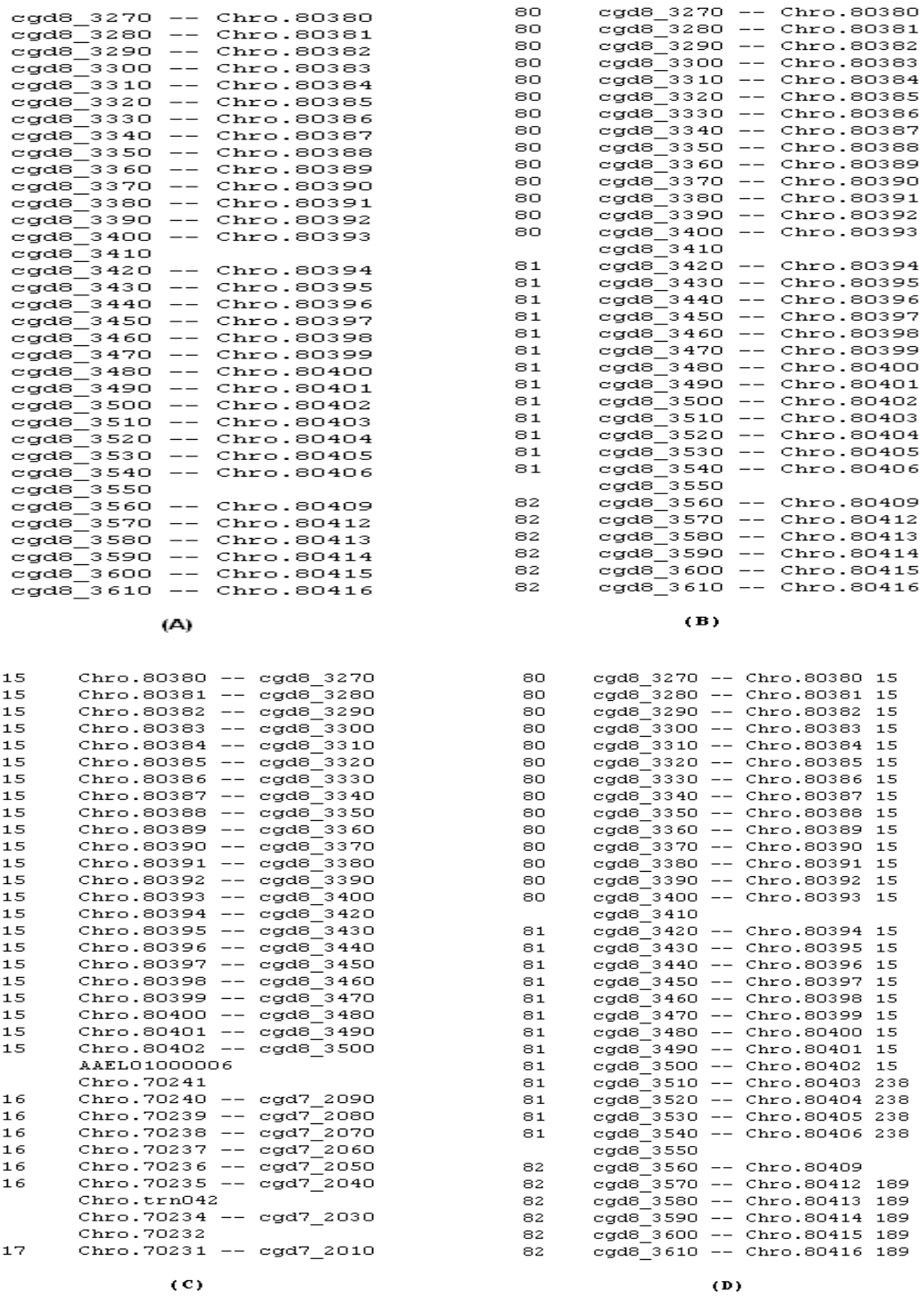


Figure 47. The example of generating the analysis files.

```
80      cgd8_3270  --  Chro.80380  15
80      cgd8_3280  --  Chro.80381  15
80      cgd8_3290  --  Chro.80382  15
80      cgd8_3300  --  Chro.80383  15
80      cgd8_3310  --  Chro.80384  15
80      cgd8_3320  --  Chro.80385  15
80      cgd8_3330  --  Chro.80386  15
80      cgd8_3340  --  Chro.80387  15
80      cgd8_3350  --  Chro.80388  15
80      cgd8_3360  --  Chro.80389  15
80      cgd8_3370  --  Chro.80390  15
80      cgd8_3380  --  Chro.80391  15
80      cgd8_3390  --  Chro.80392  15
80      cgd8_3400  --  Chro.80393  15
        cgd8_3410
81      cgd8_3420  --  Chro.80394  15
81      cgd8_3430  --  Chro.80395  15
81      cgd8_3440  --  Chro.80396  15
81      cgd8_3450  --  Chro.80397  15
81      cgd8_3460  --  Chro.80398  15
81      cgd8_3470  --  Chro.80399  15
81      cgd8_3480  --  Chro.80400  15
81      cgd8_3490  --  Chro.80401  15
81      cgd8_3500  --  Chro.80402  15
81      cgd8_3510  --  Chro.80403  238
81      cgd8_3520  --  Chro.80404  238
81      cgd8_3530  --  Chro.80405  238
81      cgd8_3540  --  Chro.80406  238
        cgd8_3550
82      cgd8_3560  --  Chro.80409
82      cgd8_3570  --  Chro.80412  189
82      cgd8_3580  --  Chro.80413  189
82      cgd8_3590  --  Chro.80414  189
82      cgd8_3600  --  Chro.80415  189
82      cgd8_3610  --  Chro.80416  189
```

(A)

```
        AAEL01000001
1       Chro.30218  --  cgd3_1840  111
1       Chro.30219  --  cgd3_1850  111
1       Chro.30220  --  cgd3_1860  111
1       Chro.30221  --  cgd3_1870  111
1       Chro.30222  --  cgd3_1880  111
1       Chro.30223  --  cgd3_1890  111
1       Chro.30224  --  cgd3_1900  111
1       Chro.30225  --  cgd3_1910  111
1       Chro.30226  --  cgd3_1920  111
1       Chro.30227  --  cgd3_1930  111
1       Chro.30228  --  cgd3_1940  111
                        cgd3_1950
1       Chro.30229  --  cgd3_1960  112
1       Chro.30230  --  cgd3_1970  112
1       Chro.30233  --  cgd3_1980  112
1       Chro.30234  --  cgd3_1990  112
1       Chro.30235  --  cgd3_2000  112
        Chro.30236
2       Chro.30237  --  cgd3_2010  112
2       Chro.30238  --  cgd3_2020  112
2       Chro.30240  --  cgd3_2030  112
2       Chro.30241  --  cgd3_2040  112
2       Chro.30242  --  cgd3_2050  112
2       Chro.30243  --  cgd3_2060  112
2       Chro.30244  --  cgd3_2070  112
2       Chro.30245  --  cgd3_2080  112
2       Chro.30246  --  cgd3_2090  112
2       Chro.30247  --  cgd3_2100  112
2       Chro.30248  --  cgd3_2110  112
2       Chro.30249  --  cgd3_2120  112
2       Chro.30250  --  cgd3_2130  112
2       Chro.30251  --  cgd3_2140  112
2       Chro.30252  --  cgd3_2150  112
2       Chro.30253  --  cgd3_2160  112
```

(B)

Figure 48. The example of two numbered genes comparison pairs files. The circles indicate the candidate insertion and deletion genes.

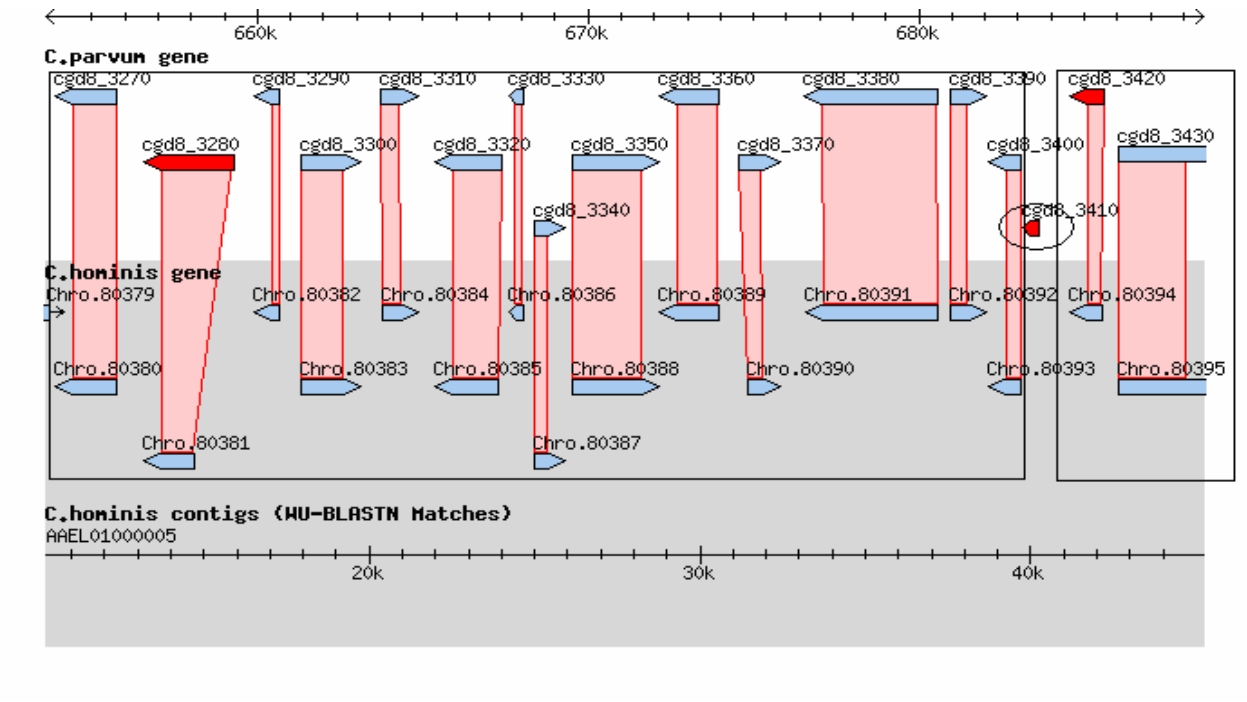


Figure 49. An example of displaying the candidate in-del genes and synteny blocks. The red glyphs indicate the un-annotated genes; the circle shows the candidate insertion gene (cgd8_3410) that is found in the analysis file of the figure 48(A). The rectangular boxes indicate the synteny blocks of genes of the *C. parvum* and *C. hominis* genomes.

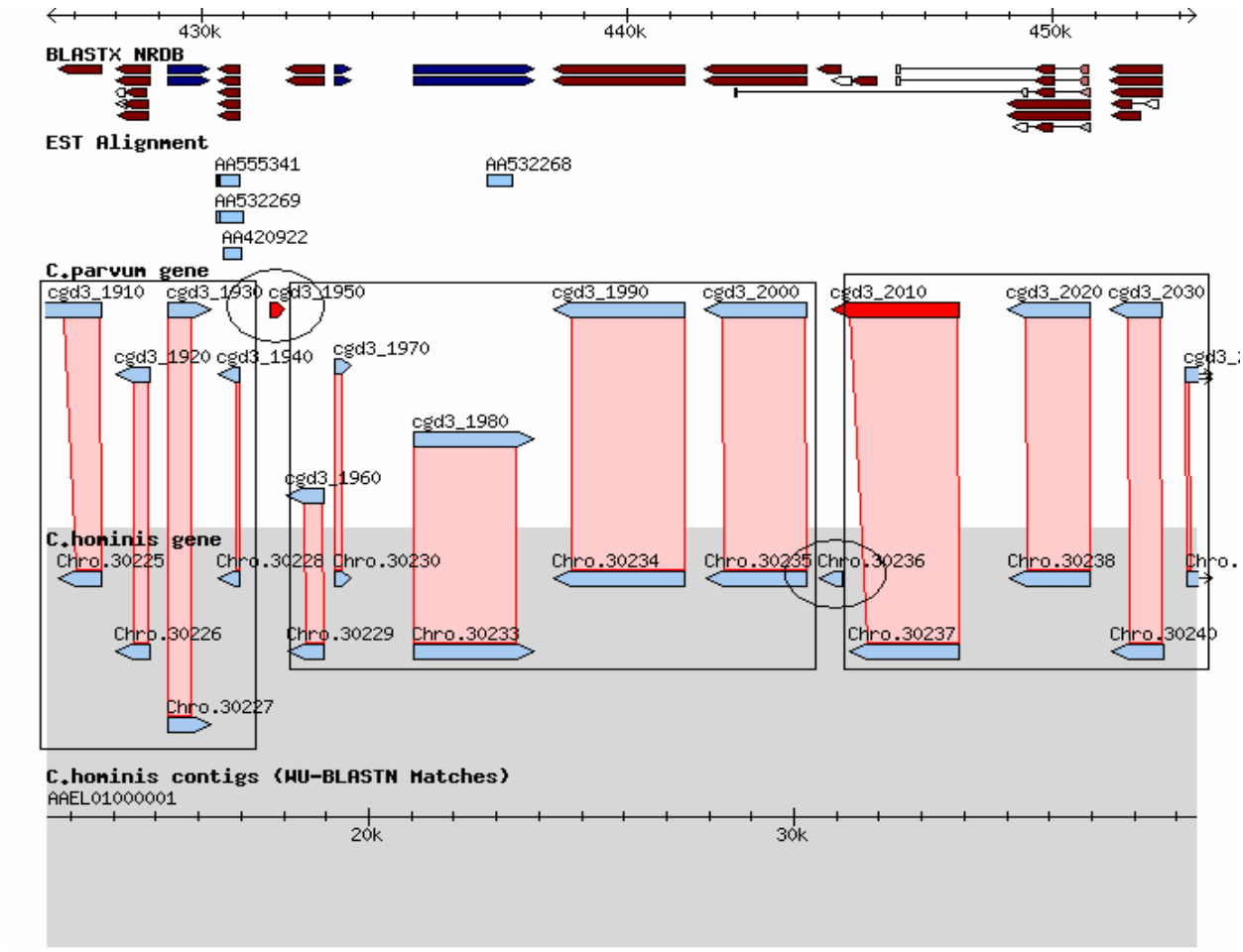


Figure 50. An example of displaying the candidate insertion and deletion genes and synteny blocks. The red glyphs indicate the un-annotated genes; the rectangular boxes indicate the synteny blocks of genes between the *C. parvum* and *C. hominis* genomes; the circles indicate the candidate insertion and deletion genes of the *C. parvum* and *C. hominis* genomes that are found by analyzing the file of the figure 48(B).

CHAPTER 6

CONCLUSION AND FUTURE WORK

The use of comparative genome sequence analysis and visualization is becoming more important with the availability of more complete genome sequences. In order to implement the comparative genome visualization of the CryptoDB project, we investigated several major genome alignment approaches and comparative visualization tools. Under these cases study, we have designed and implemented our approach suited for this project and tried to make our implementation more flexible and powerful so that it can be used for the other related projects. In addition, we have analyzed the comparison result data to find all candidate insertion and deletion genes, synteny blocks, and rearrangement events.

To make our comparative genome visualization approach more powerful, the following functions may be implemented in the future:

- ✧ Load the comparison data into the GUS database. Currently, we store the comparison data in the flat files. Thus, each time when we display one genome's region, the whole comparison matches data are retrieved. This adversely affects the performance. If we store these data into the GUS database, this shortcoming will disappear.
- ✧ Locate the detailed alignment coordinates of each matched gene. The OrthoMCL tool is used to predict the orthologs groups between genes. It does not provide the detailed gene alignment coordinates. For example, the long and the short genes are in the same orthologs group. Actually, the short gene just matches part of the long gene. Our current approach just connects these two genes together without any

consideration of detailed coordinates. We will try to obtain each coordinate of the matched genes by parsing the detailed alignment in the future work.

- ✧ Provide the display of comparison data base in tabular form. In the future work, we can add two tabs: Graphical view and Text-table view. It can help users to find the matched genes easily and quickly because the graph can not be searched easily.
- ✧ Make our code more general. It can be easily used by the other group if we make our code more clear and general.

REFERENCES

1. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J., Basic local alignment search tool, 1990.
2. Altschul, S.F., Madden, T.L., Schoffer, AA., Zhang, J., Zhang, Z., Miller, and Lipman, D.J., Gapped BLAST and PSI-BLAST: A new generation of protein database search programs, Nucleic Acids Research. 25: 3389-3402, 1997.
3. Barton, G.J. and Sternberg, M.J.E., A strategy for the rapid multiple alignment of protein sequences. J. Mol. Biol. 198: 327-337, 1987.
4. Batzoglou, S., Pachter, L., Mesirov, J., Berger, B., and Lander, E.S., Human and mouse gene structure: Comparative analysis and application to exon prediction. Genome Research. 10: 950-958, 2000.
5. Blanchette, M., Kent, W.J., Riemer, C., Elnitski, L., Smit, A.F., Roskin, K.M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E.D., et al., Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res.* 14**:** 708–715, 2004.
6. Boffelli, D., McAuliffe, J., Ovcharenko, D., Lewis, K.D., Ovcharenko, I., Pachter, L., and Rubin, E.M., Phylogenetic shadowing of primate sequences to find functional regions of the human genome. Science 299: 1391-1394, 2003.
7. Bray, Nicolas, Dubchak, I., and Pachter, L., AVID: A global alignment program. Genome, Res. 13: 97-102, 2003.
8. Bray, Nicolas and Lior Pachter, MAVID: multiple alignment server, Nucleic Acids Res., 31(13): 3525–3526, 2003.

9. Brendel, Volker, Liqun Xing and Wei Zhu, Gene structure prediction from consensus spliced alignment of multiple ESTs matching the same genomic locus. Bioinformatics, 2004.
10. Brudno, Michael, Chuong B. Do, Gregory M. Cooper, Michael F. Kim, Eugene Davydov, NISC Comparative Sequencing Program, Eric D. Green, Arend Sidow and Serafim Batzoglou, LAGAN and Multi-LAGAN: Efficient Tools for Large-Scale Multiple Alignment of Genomic DNAGenome Research Vol 13, Issue 4, 721-731, 2003.
11. Brodie, Ryan, Rachel L. Roper and Chris Upton, JDotter: a Java interface to multiple dotplots generated by dotter, Bioinformatics, Vol. 20 no. 2 pages 279-281, 2004.
12. Brudno, Michael, Michael Chapman, Berthold Göttgens, Serafim Batzoglou and Burkhard Morgenstern, Fast and sensitive multiple alignment of large genomic sequences, BMC Bioinformatics, 2003
13. Brudno, Michael, Sanket Malde, Alexander Poliakov, Chuong B. Do, Olivier Couronne, Inna Dubchak and Serafim Batzoglou, Glocal alignment: finding rearrangements during alignment, Bioinformatics Vol. 19, 2003.
14. Carver, Tim J., Kim M. Rutherford, Matthew Berriman, Marie-Adele Rajandream, Barclay G. Barrell and Julian Parkhill, ACT: the Artemis comparison tool, Bioinformatics 2005.
15. Cooper, Gregory M., Senthil AG Singaravelu and Arend Sidow. ABC: software for interactive browsing of genomic multiple sequence alignment data. BMC Bioinformatics 2004.
16. Darling, Aaron C.E., Bob Mau, Frederick R. Blattner and Nicole T. Perna, Mauve: Multiple Alignment of Conserved Genomic Sequence With Rearrangements Genome Research 14:1394-1403, 2004.

17. Delcher, A.L., Kasif S, Fleischmann RD, Peterson J, White O, Salzberg SL, Alignment of whole genomes. Nucleic Acids Res, 27:2369-76, 1999.
18. Delcher, A.L., Phillippy, A., Carlton, J., and Salzberg, S.L., Fast algorithms for large-scale genome alignment and comparison. Nucleic Acids Res. 30: 2478-2483, 2002.
19. Dongen, S., Van, "Graph clustering by flow simulation." Ph.D thesis, University of Utrecht, The Netherlands, 2000.
20. Ebedes, Justin and Amitava Datta, Multiple sequence alignment in parallel on a workstation cluster, Bioinformatics 20(7), 2004.
21. Fitch, W.M., Distinguishing homologous from analogous proteins. Syst. Zool. 19: 99–113, 1970.
22. Fitch, W.M., Homology, a personal view on some of the problems. Trends Genet. 16: 227–231, 2000.
23. Frazer, Kelly A., Laura Elnitski, Deanna M. Church, Inna Dubchak and Ross C. Hardison, Cross-Species Sequence Comparisons: A Review of Methods and Available Resources, Vol 13, Issue 1, 1-12, January 2003.
24. Frazer, Kelly A., Lior Pachter, Alexander Poliakov, Edward M. Rubin, and Inna Dubchak, VISTA: computational tools for comparative genomics, Nucleic Acids Research, Vol. 32, Web Server issue © Oxford University Press, 2004.
25. Gish, W., (1996-2004) http://blast.wustl.edu.
26. Gotoh, O., Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. J. Mol. Biol. 264: 823-838, 1996.

27. Gottgens, B., Barton, L.M., Chapman, M.A., Sinclair, A.M., Knudsen, B., Grafham, D., Gilbert, J.G., Rogers, J., Bentley, D.R., and Green, A.R., Transcriptional regulation of the stem cell leukemia gene (SCL)-comparative analysis of five vertebrate SCL loci. Genome Res. 12: 749-759, 2002.
28. Gusfield, D, Algorithm on Strings, Trees and Sequences: Computer Science and Computational Biology, Cambridge University Press, New York, 1997.
29. Hardison, R., Chao, K.-M, Adamkiewicz, M., Price, D., Jackson, J., Zeigler, T., Stojanovic, N., and Miller, W., Positive and negative regulatory elements of the rabbit ε-globin gene revealed by an improved multiple alignment program and functional analysis. DNA Seq. 4: 163-176, 1993.
30. Hardison, R., Chao, K.-M., Schwartz, S., Stojanovic, N., Ganetsky, M., and Miller, W., Globin Gene Server: A prototype E-mail database server featuring extensive multiple alignments and data compilation for electronic genetic analysis. Genomics 21: 344-353, 1994.
31. Hardison, RC, Comparative Genomics, PLoS Biol 2003.
32. Jareborg, Niclas and Richard Durbin1, Alfresco-A Workbench for Comparative Genomic Sequence Analysis, Genome Research Vol. 10, Issue 8, 1148-1157, August 2000.
33. Kellis, M., Patterson, N., Endrizzi, M., Birren, B. and Lander, E.S., Sequencing and comparison of yeast species to identify genes and regulatory elements. Nature 423: 241-254, 2003.
34. Kent, W.J. and Zahler, A.M., Conservation, regulation, synteny, and introns in a large-scale C. briggsae–C. elegans genomic alignment. Genome Res. 10: 1115-1125, 2000.
35. Kurtz, S., Phillippy A, Delcher AL, Smoot M, Shumway M, Antonescu C, and Salzberg SL, Versatile and open software for comparing large genomes. BioMed Central Ltd, 2004.

36. Mayor, C, Brudno M, Schwartz JR, Poliakov A, Rubin EM, Frazer KA, Pachter LS, Dubchak I: VISTA: visualizing global DNA sequence alignments of arbitrary length. Bioinformatics, 16:1046-1047, 2000.
37. Morgenstern, B., DIALIGN: Improvement of the segment-to-segment approach to multiple sequence alignment. Bioinformatics 15: 211-218, 1999.
38. Needleman, S.B. and Wunsch, C.D., A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. 48: 443-453, 1970.
39. Ning, Z., Cox, A.J., and Mullikin, J.C., SSAHA: A fast search method for large DNA databases. Genome Res. 11: 1725-1729, 2001.
40. Ovcharenko, I., Boffelli, D., and Loots, G.G., eShadow: A tool for comparing closely related sequences. *Genome Res.* 14: 1191–1198, 2004.
41. Ovcharenko, Ivan, Gabriela G. Loots, Belinda M. Giardine, Minmei Hou, Jian Ma, Ross C. Hardison, Lisa Stubbs and Webb Miller, Mulan: Multiple-sequence local alignment and visualization for studying function and evolution Genome Research Vol 0, 2004.
42. Ovcharenko, I., Loots, G.G., Hardison, R.C., Miller, W., and Stubbs, L., zPicture: Dynamic alignment and visualization tool for analyzing conservation profiles. *Genome Res.* 14**:** 472–477, 2004.
43. Pan, Xiaokang, Lincoln Stein and Volker Brendel, SynBrowse: a synteny browser for comparative sequence analysis, BIOINFORMATICS Vol. 21 no.17, pages 3461–3468, 2005.
44. Pennacchio, Len A. and Edward M. Rubin, Comparative genomic tools and databases: providing insights into the human genome, *J. Clin. Invest.* 111:1099-1106, 2003.

45. Puiu, Daniela, Shinichiro Enomoto, Gregory A. Buck, Mitchell S. Abrahamsen and Jessica C. Kissinger CryptoDB: the Cryptosporidium genome resource Nucleic Acids Research, Vol. 32, Database issue D329-D331, 2004.
46. Rasko, David A., Garry SA Myers and Jacques Ravel, Visualization of comparative genomic analyses by BLAST score ratio, BMC Bioinformatics, 2005.
47. Read, TD, Myers GS, Brunham RC, Nelson WC, Paulsen IT, Heidelberg J, Holtzapple E, Khouri H, Federova NB, Carty HA, Umayam LA, Haft DH, Peterson J, Beanan MJ, White O, Salzberg SL, Hsia RC, McClarty G, Rank RG, Bavoil PM, Fraser CM: Genome sequence of Chlamydophila caviae (Chlamydia psittaci GPIC): examining the role of niche-specific genes in the evolution of the Chlamydiaceae. Nucleic Acids Res, 31(8):2134-2147, 2003.
48. Roos, David S., Li Li, Christian J. and Stoeckert Jr., OrthoMCL: Identification of Ortholog Groups for Eukaryotic Genomes, Gernome Research 2003.
49. Saitou, N. and Nei, M., The neighbor-joining method: A new method for reconstructing phylogenetic trees, Mol. Biol. Evol. 4: 406–425, 1987.
50. Schwartz, S., Elnitski L, Li M, Weirauch M, Riemer C, Smit A, Green ED, Hardison RC, Miller W: MultiPipMaker and supporting tools: Alignments and analysis of multiple genomic DNA sequences. Nucleic Acids Res, 31:3518-3524, 2003.
51. Schwartz, S., Zhang, Z., Frazer, K.A., Smit, A., Riemer, C., Bouck, J., Gibbs, R., Hardison, R., and Miller, W., PipMaker – A web server for aligning two genomic DNA sequences. Genome Res. 10: 577-586, 2000.
52. Shah, Nameeta, Olivier Couronne, Len A. Pennacchio, Michael Brudno, Serafim Batzoglou, E. Wes Bethel, Edward M. Rubin, Bernd Hamann, and Inna Dubchak, Phylo-VISTA: An

Interactive Visualization Tool for Multiple DNA Sequence Alignments,.Lawrence Berkeley National Laboratory. Paper LBNL-54136, April 1, 2004.

53. Schwartz, S., Kent,W.J., Smit,A., Zhang,Z., Baertsch,R., Hardison,R.C., Haussler,D. and Miller,W., Human-mouse alignments with BLASTZ. *Genome Res.*, 13, 103–105, 2003.
54. Smith, T.F. and Waterman, M.S., Identification of common molecular subsequences. J. Mol. Biol. 147: 195-197, 1981.
55. Stajich, Jason E., David Block, Kris Boulez, Steven E. Brenner, Stephen A. Chervitz, Chris Dagdigian, Georg Fuellen, James G.R. Gilbert, Ian Korf, Hilmar Lapp, Heikki Lehväslaiho, Chad Matsalla, Chris J. Mungall, Brian I. Osborne, Matthew R. Pocock, Peter Schattner, Martin Senger, Lincoln D. Stein, Elia Stupka, Mark D. Wilkinson, and Ewan Birney, The Bioperl Toolkit: Perl Modules for the Life Sciences, Genome Research Issue 10, 1611-1618, 2002.
56. Stein, Lincoln D., Christopher Mungall, ShengQiang Shu, Michael Caudy, Marco Mangone, Allen Day, Elizabeth Nickerson, Jason E. Stajich, Todd W. Harris, Adrian Arva, and Suzanna Lewis, The Generic Genome Browser: A Building Block for a Model Organism System Database, Genome Research Vol. 12, Issue 10, 1599-1610, 2002.
57. Taylor, W., A flexible method to align large numbers of biological sequences. J. Mol. Evol. 28: 161-169, 1988.
58. Thompson, J.D., Higgins, D.G., and Gibson, T.J., CLUSTALW: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. Nucleic Acids Res. 22: 4673-4680, 1994.
59. Thomas, J.W., Touchman, J.W., Blakesley, R.W., Bouffard, G.G., Beckstrom-Sternberg, S.M., Margulies, E.H., Blanchette, M., Siepel, A.C., Thomas, P.J., McDowell, J.C., et al.,

Comparative analyses of multi-species sequences from targeted genomic regions. Nature 424: 788-793, 2003.

60. Ureta-Vidal, Abel, Laurence Ettwiller and Ewan Birney, Comparative Genomics: Genome-Wide Analysis in Metazoan Eukaryotes, Nature Publishing Group 2003.

.

Web Resources:

1, some species-specific sequence databases:

NCBI: http://www.ncbi.nlm.nih.gov/

TIGR: http://www.tigr.org/

Sanger: http://www.sanger.ac.uk/

Ensembl: http://www.ensembl.org/

TAIR: http://www.arabidopsis.org/home.html

SGD: http://genome-www.stanford.edu/Saccharomyces/

MGD: http://www.informatics.jax.org/

Human Genome Browser: http://www.genome.ucsc.edu/

NISC: http://www.nisc.nih.gov/

Rat Genome Database: http://www.rgd.mcw.edu/

FlyBase: http://flybase.bio.indiana.edu/

Wormbase: http://brie2.cshl.org:8081/

ExoFish: http://www.genoscope.cns.fr/externe/tetraodon/

2, some gene annotation and prediction programs:

GENSCAN: http://genes.mit.edu/GENSCAN.html

GenomeScan: http://genes.mit.edu/genomescan/

EST Genome: http://www.sanger.ac.uk/Software/Alfresco/download.shtml

FGENESH: http://genomic.sanger.ac.uk/gf.html

GrailEXP: http://compbio.ornl.gov/grailexp/

TwinScan: http://genes.cs.wustl.edu/query.html

Genie http://www.fruitfly.org/seq_tools/genie.html

SLAM: http://baboon.math.berkeley.edu/~syntenic/slam.html

3, some sequence alignment approaches:

LAGAN and Multi-LAGAN: http://lagan.stanford.edu

Shuffle-LAGAN: http://lagan.stanford.edu/lagan_web/shuffle.shtml

AVID: http://bio.math.berkeley.edu/avid/

MAVID: http://baboon.math.berkeley.edu/mAVID

Mauve: http://gel.ahabs.wisc.edu/mauve/

MUMmer: http://mummer.sourceforge.net/

PipMaker and Multi-PipMaker: http://bio.cse.psu.edu

CHAOS: http://www.cs.stanford.edu/~brudno/chaos/

DIALIGN: http://bibiserv.techfak.uni-bielefeld.de/dialign/

CHAOS & DIALIGN: http://dialign.gobics.de/chaos-dialign-submission

ClustalW: http://www.ebi.ac.uk/clustalw/

OrthoMCL: http://orthomcl.cbil.upenn.edu/cgi-bin/OrthoMclWeb.cgi

BLAST: http://www.ncbi.nlm.nih.gov/BLAST

LALIGN: http://www.ch.embnet.org/software/LALIGN_form.html

4, some comparison visualization tools:

VISTA: http://www-gsd.lbl.gov/vista

Phylo-VISTA: http://www-gsd.lbl.gov/phylovista

SynBrowse: http://www.synbrowser.org/

ACT: http://www.sanger.ac.uk/Software/ACT/ http://www.webact.org

ABC: http://mendel.stanford.edu/sidowlab/downloads/ABC_GERP/abcgerp.html

Sybil: http://sybil.sourceforge.net/

Mulan: http://mulan.dcode.org/

Mauve: http://gel.ahabs.wisc.edu/mauve/

MUMmer: http://mummer.sourceforge.net/

APPENDIX A

DEFINITIONS

Conserved: Derived from a common ancestor and retained in contemporary related species.

DNA: The chemical inside the nucleus of a cell that carries the genetic instructions for making living organisms.

Exon: The region of a gene that contains the code for producing the gene's protein.

Genome: The entire DNA contained in an organism or a cell, which includes both the chromosomes within the nucleus and the DNA in mitochondria.

Genomics: The study of genes and their function.

Gene: The functional and physical unit of heredity passed from parent to offspring.

Highly conserved sequence: A DNA sequence that is very similar in several different kinds of organisms.

Homology: Sequence similarity that can be attributed to descent from a common ancestor.

Homolog: A gene related to a second gene by descent from a common ancestral DNA sequence.

Intron: A non-coding sequence of DNA that is initially copied into RNA but is cut out of the final RNA transcript.

Non-coding DNA: The strand of DNA that does not carry the information necessary to make a protein.

Non-coding RNA: any RNA molecule that functions without being translated into a protein.

Ortholog: Genes in different species that evolved from a common ancestral gene by speciation.

Paralog: Genes related by duplication within a genome.

Phylogenetic tree: A tree that shows the evolutionary interrelationships among various species that are believed to have a common ancestor.

Protein: A large complex molecule made up of one or more chains of amino acids.

Regulatory region: a DNA base sequence that controls gene expression.

Similarity: The extent to which sequences are related.

Synteny: Genes occurring in the same order on chromosomes of different species.

APPENDIX B

MAJOR FUNCTIONS

B.1 The sample code of *get_matches* function aims to retrieve the match pairs from the file generated by OrthoMCL.

```
sub get_matches {

      my $boxes = shift;
      my %seen = ();
      my @array;

      foreach (@$boxes){

            my ($feature,$left,$top,$right,$bottom,$track) = @$_;
            my $name = $feature->name;
            my $start = $feature->start;
            my $end = $feature->end;

            $seen{$name} = 1;
      }

      open( INPUT, "< /tmp/all_orthomcl_transfer.out" ) || die "cannot open the file";

      while(<INPUT>) {

            my ($tar, $ref, $e1, $e2) = split("--", $_);
            chomp $tar;
            chomp $ref;
            chomp $e1;
            chomp $e2;
            if(exists $seen{$tar} || exists $seen{$ref}) {
                  push (@array, [$ref, $tar, $e1, $e2]);
            }
      }
      return @array;
}
```

B.2 The sample code of *get_glyph_coord* function aims to obtain the coordinate of specific gene glyph on the Panel used by the *draw_comparison_line* function.

```
sub get_glyph_coord {

      my ($self, $id,  $flag) = @_;
      my $boxes = $self->boxes;
      my $scale = $self->scale;
      my ($offset, $indent, $y);

      foreach (@$boxes){
            my ($feature,$left,$top,$right,$bottom,$track) = @$_;
            my $pad_bottom = $track->option('pad_bottom');

            if( $feature->name eq $id){   # this is a reference feature

                  if($feature->start < $track->start){ # feature cross left edge

                        $offset = ($feature->stop - $track->start) * $scale;

                  } elsif($feature->stop > $track->stop){  # feature cross right edge

                        $offset = ($track->stop - $feature->start)*$scale;

                  } else {

                        $offset = ($feature->stop - $feature->start)*$scale;
                  }

                  $indent = $track->option('height') < $offset ?
                              $track->option('height') : $offset /2;

                  $y = $flag >= 0 ? $bottom-$pad_bottom * 3 : $bottom - $track->option('height')-$pad_bottom*2 ;

                  return [$left, $y, $left+$offset-$indent, $y] if ($feature->strand >= 0);
                  return [$left+$indent, $y, $left+$offset, $y];
            }
      } # end foreach (@$boxes) loop
}
```

B.3 The sample code of *highlight_feature* function aims to highlight the specific gene features in order to show the match pairs that are not in the same panel region.

```
sub highlight_feature {
      my ($self, $id, $flag) = @_;
      my $boxes = $self->boxes;
      my $scale = $self->scale;
      my ($offset, $indent);
      my $shadow_off = 40 * $scale * 2;

      foreach (@$boxes){
            my ($feature,$left,$top,$right,$bottom,$track) = @$_;
            my $pad_bottom = $track->option('pad_bottom');
            $pad_bottom = $flag >=0 ? $pad_bottom*3 : $pad_bottom*2;

            if( $feature->name eq $id){   # this is a reference feature
                  if($feature->start < $track->start){ # feature cross left edge
                        $offset = ($feature->stop - $track->start) * $scale;
                  } elsif($feature->stop > $track->stop){  # feature cross right edge
                        $offset = ($track->stop - $feature->start)*$scale;
                  } else {
                        $offset = ($feature->stop - $feature->start)*$scale;
                  }

                  $indent = $track->option('height') < $offset ?
                              $track->option('height') : $offset /2;

                  return [$left-$shadow_off+1,               $bottom-$track->option('height')-$shadow_off-$pad_bottom+2,
                        $left+$offset-$indent+$shadow_off+1, $bottom-$track->option('height')-$shadow_off-$pad_bottom+2,
                        $left+$offset+$shadow_off+1,         $bottom-$track->option('height')/2-$pad_bottom,
                        $left+$offset-$indent+$shadow_off+1, $bottom+$shadow_off-$pad_bottom,
                        $left-$shadow_off+1,                 $bottom+$shadow_off-$pad_bottom] if ($feature->strand >= 0 );

                        return [$left+$indent-$shadow_off-1, $bottom-$track->option('height')-$shadow_off-$pad_bottom+2,
                                $left+$offset+$shadow_off-1, $bottom-$track->option('height')-$shadow_off-$pad_bottom+2,
                                $left+$offset+$shadow_off-1, $bottom+$shadow_off-$pad_bottom,
                                $left+$indent-$shadow_off-1, $bottom+$shadow_off-$pad_bottom,
                        $left-$shadow_off-1,         $bottom-$track->option('height')/2-$pad_bottom];
            }
      } # end foreach (@$boxes) loop
}
```

B.4 The sample code of *draw_comparison_line* function aims to draw the comparison areas that are in the same orthologous group.

```
sub draw_comparison_line {
      my ($gd, $boxes, $panel, $match_array, $flag) = @_;
      my $line = $gd->colorAllocate($panel->color_name_to_rgb('red'));
      my $region = $gd->colorAllocate($panel->color_name_to_rgb('pink'));
      my @region_cord_array;

      foreach my $match (@$match_array) {
            my $poly = new GD::Polyline;
            my $highlight_cord;
                #10/11/0 add different color for different e-value
                my $evalue = $2 if $match->[2] =~ m/(.*?)e(.*?)$/;
            if($match->[2] ne "0e0"){
                  $region = $gd->colorAllocate($panel->color_name_to_rgb('pink'));
                }else{
                  $region = $gd->colorAllocate($panel->color_name_to_rgb('pink'));
            }#end

            my $ref = get_glyph_coord($panel, $match->[0], 1);
            my $tar = get_glyph_coord($panel, $match->[1], -1);
            # 10/03/2005

            if ($ref != "" && $tar !="" ) {
                  $poly->addPt($ref->[0],$ref->[1]);
                  $poly->addPt($ref->[2],$ref->[3]);
                  $poly->addPt($tar->[2],$tar->[1]);
                  $poly->addPt($tar->[0],$tar->[3]);
                  if($flag == 0) {
                        $gd->filledPolygon($poly,$region);
                  } else {
                        $gd->polygon($poly,$line);
                  }

                  # 9/22/2005
                  my $match_name = $match->[0]. "-". $match->[1]. ".aln";
                  push(@region_cord_array, [$ref->[0],$ref->[1],$ref->[2],$ref->[3],
                                                $tar->[2],$tar->[1],$tar->[0],$tar->[3],$match_name]);

            }
            #10/07/2005


           elsif ($ref != "" && $tar == "") {
                  $highlight_cord = highlight_feature($panel, $match->[0], 1);
                  $poly->addPt($highlight_cord->[0],$highlight_cord->[1]);
                  $poly->addPt($highlight_cord->[2],$highlight_cord->[3]);
                  $poly->addPt($highlight_cord->[4],$highlight_cord->[5]);
                  $poly->addPt($highlight_cord->[6],$highlight_cord->[7]);
                  $poly->addPt($highlight_cord->[8],$highlight_cord->[9]);

                  $gd->filledPolygon($poly,$region);
                  $gd->polygon($poly, $line);

                  my $match_name = $match->[0]. "-". $match->[1]. ".aln";
                  push(@region_cord_array, [$highlight_cord->[0],$highlight_cord->[1],$highlight_cord->[2],
                       $highlight_cord->[3],$highlight_cord->[4],$highlight_cord->[5],
                       $highlight_cord->[6],$highlight_cord->[7],$highlight_cord->[8],
                       $highlight_cord->[9],$match_name]);

            } elsif ($ref == "" && $tar != "") {
                  $highlight_cord = highlight_feature($panel, $match->[1], -1);
                        $poly->addPt($highlight_cord->[0],$highlight_cord->[1]);
                        $poly->addPt($highlight_cord->[2],$highlight_cord->[3]);
                        $poly->addPt($highlight_cord->[4],$highlight_cord->[5]);
                        $poly->addPt($highlight_cord->[6],$highlight_cord->[7]);
                        $poly->addPt($highlight_cord->[8],$highlight_cord->[9]);
                        $gd->filledPolygon($poly,$region);
                        $gd->polygon($poly, $line);

                        my $match_name = $match->[0]. "-". $match->[1]. ".aln";
                        push(@region_cord_array, [$highlight_cord->[0],$highlight_cord->[1],$highlight_cord->[2],
                                                  $highlight_cord->[3],$highlight_cord->[4],$highlight_cord->[5],
                                                   $highlight_cord->[6],$highlight_cord->[7],$highlight_cord->[8],
                                                   $highlight_cord->[9],$match_name]);
            }
      } # end foreach
      #store the comparative region coordination
      $$panel{"comp_array"} = \@region_cord_array;
      return $panel;
}
```