Honours Thesis

# Computational Gene Finding in the Human Malaria Parasite *Plasmodium vivax*

Alexander David Stivala
271025

Supervisor: Dr Anthony Wirth

October 25, 2006

**Abstract**

Different approaches to genome annotation are reviewed and compared with reference based annotation using GeneMapper in the human malaria parasite *Plasmodium vivax*. It is found that the latter approach does not achieve sensitivity and specificity as high as those for some *ab initio* techniques. Potential reasons for this are identified and discussed. As part of the process of using GeneMapper, codon substitution matrices are constructed and examined. This leads to the discovery of evidence from which we derive a conjecture regarding *Plasmodium* evolution.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

There are four species of malaria parasites that infect humans (*P. falciparum*, *P. vivax*, *P. ovale*, and *P. malariae* [14]), the two with greatest prevalence being *Plasmodium vivax* and *Plasmodium falciparum* [55]. Although, unlike *P. falciparum* malaria, *P. vivax* malaria is rarely fatal, it is a debilitating disease and it has been estimated that there are at least 70 million cases of *P. vivax* malaria infection per year worldwide [55]. Hence research into *P. vivax* genomics, in order to identify possible drug targets for example, is important. However, as will be described in Chapter 2, *P. vivax* research has tended to lag behind *P. falciparum* research.

Since *P. falciparum* has been sequenced and annotated, and the *P. vivax* annotation project is incomplete, a system which uses an annotation from one species in order to annotate another related species would appear to be useful. Comparative or reference based gene finding with systems such as Projector [57] and GeneMapper [17] have been very successful on closely related species such as human and mouse.

For this reason, we investigate the application of reference based gene finding with GeneMapper, to use the *P. falciparum* annotation to assist in annotating the *P. vivax* genome.

Section 2.5 gives an overview of comparative malaria genomics, and in particular describes the marked difference in GC-content between *P. falciparum* and *P. vivax*. GeneMapper uses a particular kind of substitution matrix to model evolution, as will be described in Section 4.2.1. The construction of such a matrix to model evolutionary changes between the *P. falciparum* and *P. vivax* genomes provides evidence from which we conjecture that background change in GC-content, rather than selective pressure, is responsible for non-synonymous substitutions between these genomes.

# Chapter 2

# Genetics and genomics

Genetics, molecular biology and genomics are large and complex fields and only the briefest overview can be given here. An introduction to genetics can be found in Gonick and Wheelis [32], and a standard reference for genetics and molecular biology is Alberts *et al.* [2]. Gibson and Muse [29] give an overview of genome sequencing projects and genomics.

## 2.1 DNA

The cells of of all known living things encode their hereditary information using the same code. This code consists of only four different chemicals called *nucleotides*, conventionally denoted A, T, C, and G for adenine, thymine, cytosine and guanine, respectively. These bases are divided into two classes based on their chemical structures. A and G are *purines*, while C and T are *pyrimidines*. Strictly speaking, A, T, C, and G are *bases*, and a nucleotide consists of a sugar called deoxyribose (from which deoxyribonucleic acid, DNA, obtains its name) along with a phosphate group and one of the four bases protruding from it ([2], p. 5). The sugars and phosphate groups form a chemical linkage which is asymmetric and therefore gives a chain, or *strand*, of such nucleotides a direction.

The sugars in the nucleotides contain five carbon atoms which are numbered with a prime mark (i.e. $1'$, "one-prime" through to $5'$ "five-prime"). The asymmetry arises because the nucleotides are joined by a linkage between the hydroxyl group on the $3'$ carbon of one nucleotide and the phosphate group on the $5'$ carbon atom of the adjacent nucleotide in the strand ([2], p. 121). Hence a strand of DNA has an unlinked hydroxyl group at one and and an unlinked phosphate group at the other, and thus we can refer to them unambiguously as the $3'$ *end* and $5'$ *end* respectively ([2], pp. 193–195).

The bases A, T, C and G have complementary structures so that A forms hydrogen bonds with T and C forms hydrogen bonds with G. This allows a strand complementary to an existing strand to be formed by each base binding to its complementary base on the existing strand. These bonds are weaker than the sugar-phosphate bonds forming the backbone of the strands, allowing DNA to be replicated by the strands separating without their backbones breaking, each strand then being able to form a template for synthesizing a new DNA strand complementary to itself in a process known as *templated polymerization* ([2], p. 6). In addition to being double-stranded, DNA twists around itself to form the well-known double-helix structure.

Since the two strands of DNA are complementary, to write down the sequence of A, T, C, G encoded in a double-stranded DNA molecule we need only write down the sequence on one strand. Hence, when talking about a sequence on a double-stranded DNA molecule, the bases are often referred to as base pairs. By convention, a nucleotide sequence is written from the 5′ end to the 3′ end ([2], p. 196), the direction in which transcription occurs.

Organisms are divided into the two classes *eukaryotes*, which have a nucleus and complex cell structure, and *prokaryotes*, such as bacteria, which do not. In eukaryotic cells, the DNA sequences are distributed in a set of *chromosomes*, long DNA molecules folded into a compact structure with proteins. The number of chromosomes differs between organisms. The human genome, for instance, consists of 46 chromosomes (22 chromosome pairs, as well as the X and Y chromosomes) ([2], p. 198). The *Plasmodium* species each have 14 chromosomes [34].

## 2.2 DNA encodes proteins via RNA

This section gives a basic overview of how a cell uses DNA to build proteins, sufficient for the purposes of understanding the rest of the thesis. Specifically, I describe the process only for eukaryotes (which includes the malaria parasites) only, omitting the (simpler) processes for prokaryotes. A detailed explanation can be found in chapter 6 of Alberts *et al.* [2].

It has been mentioned that the DNA sequence is a form of code. But what does it encode? It encodes the information necessary to synthesize *proteins*, macromolecules that are amino acid polymers, i.e. chains of amino acids. Hence a protein itself can be considered as a sequence (of amino acids, rather than bases). Proteins have structure in addition to their sequence, but unlike DNA structure, it varies widely amongst different proteins. In general, all of the structure and activity of a cell consists of proteins, and interactions amongst them. A DNA sequence that encodes a protein is called a *gene*, although, as will be described in Section 2.2.1, a single gene can actually encode multiple proteins.

DNA sequence is not directly converted to proteins. Rather, copies of small sections of DNA are made in a process called *transcription*, which is similar to replication. The result of transcription is a single-stranded RNA molecule that is a copy of the strand of DNA used as a template.

RNA is similar to DNA in that it is a nucleotide polymer, but amongst other differences, its sequence consists of the bases A, G, C, and U, where A, G, and C are the same bases as DNA, but U (uracil), another pyrimidine, is used in place of T in DNA. The RNA sequence is then used by the cell to synthesize proteins, in a process known as *translation*. That all cells synthesize proteins in this way, i.e. from DNA to RNA to protein is known as the *central dogma* of molecular biology ([2], p. 301).

### 2.2.1 Transcription

RNA is transcribed from DNA from the 5′ end to the 3′ end on one strand (hence the convention of writing sequences from the 5′ end), using complementary base pairing, similar to the process of templated polymerization for DNA replication described previously.

This produces an RNA sequence known as a *transcript*, and the transcribed section of DNA is called *transcription unit*. A transcription unit corresponds to a gene in that it encodes one protein. An RNA molecule which is a transcript that will encode a protein is known as an *mRNA* (messenger RNA) molecule.

Clearly, there must be a mechanism that allows the cell to know where to start and stop transcription. The particular nucleotide sequences on the DNA (such special sequences are in general referred to as *signals*) that serve this purpose are called *promoters* and *terminators* respectively. Note that promoter and terminator signals are somewhat upstream (i.e. towards the $5'$ end) and downstream of the transcription unit itself respectively. The exact sequences that constitute these signals, and their distance up- and down-stream vary between genes and organisms; it is this sort of thing that makes gene prediction a non-trivial problem. The promoter signal consists primarily of T and A nucleotides and for this reason is called a *TATA box*.

The pre-mRNA sequence that results directly from transcription must undergo a process of maturation before it becomes mRNA which is the input to the translation process.

The most important modification that the pre-mRNA undergoes before it can become mRNA is *splicing*. This process removes long segments of non-coding sequence called *introns* from the smaller segments of actual coding sequence, known as *exons*. Boundaries between introns and exons are known as *splice sites*. A $5'$ splice site (i.e. the boundary where an intron begins) is called a *donor* splice site and a $3'$ splice site (i.e. the boundary where an intron ends) is called an *acceptor* splice site [61]. Figure 2.1 (adapted from Figure 6-28 of Alberts *et al.* [2], which does not use the acceptor/donor terminology) shows a typical intron and its removal to create a contiguous sequence from the exons on either side of it.

There are many hypotheses as to why such an apparently wasteful phenomenon as introns exist. One of these is that this phenomenon has evolved because it provides for *alternative splicings* so that one gene can actually encode several different proteins ([2], p. 318).

The signals that determine splicing contain little information, and vary somewhat, making them difficult to accurately predict. Determining which parts of a nucleotide sequence are exons is the major challenge in computational gene finding discussed in this thesis. *Consensus* (i.e. "average" or typical) sequences for splice sites for different organisms have been found: these are necessary, but not sufficient, for determining that a site is a splice site. For example, the donor splice site usually contains the dinucleotide GT (i.e. GU in RNA) and the acceptor splice site AG [61], as shown in Figure 2.1. Computational models so far can only make (increasingly accurate) predictions, as the exact mechanism of splicing is not yet fully understood.

It should be noted that the vast majority of the DNA sequence is not transcribed, and largely has no known function. In addition, not all transcribed sequences are translated; sometimes the RNA created by transcription is the end product.

### 2.2.2 Translation

The set of rules for translation from a nucleotide sequence to an amino acid sequence is known as the *genetic code*. The code consists of codewords three nucleotides long called

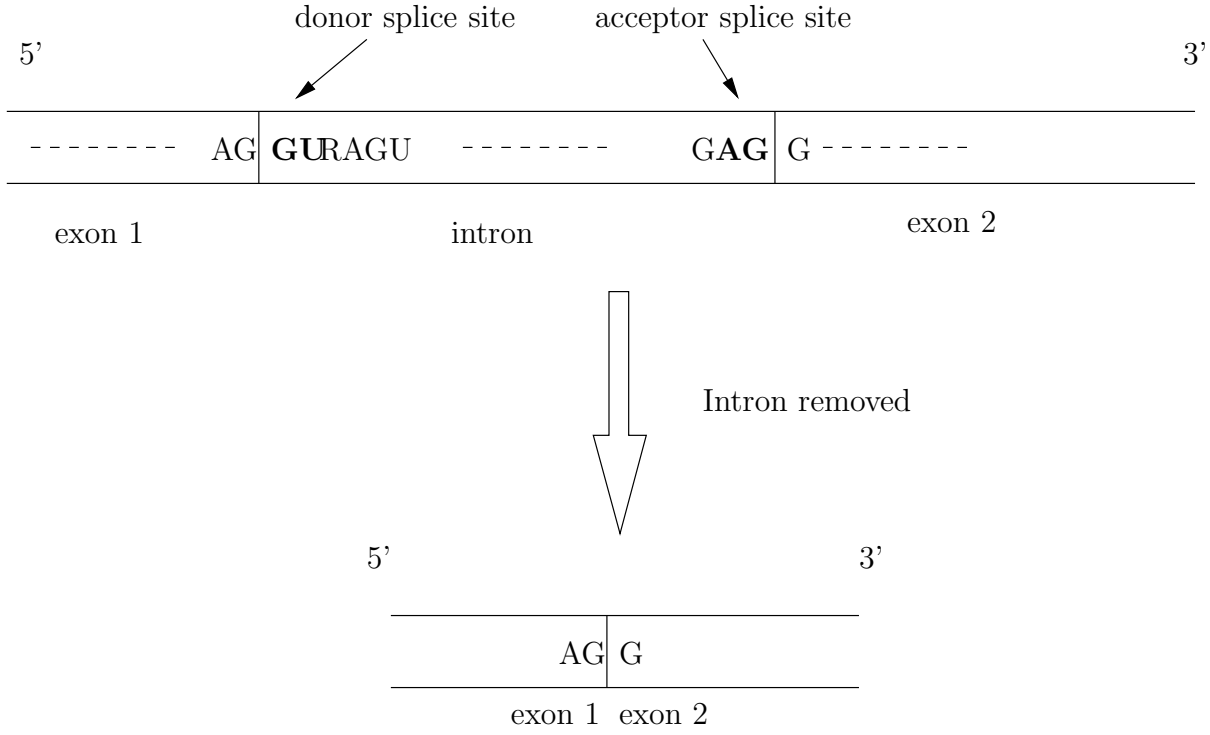



Figure 2.1: Splicing removes introns from pre-mRNA.

*codons*. There are $4^3 = 64$ possible codons and only 20 amino acids commonly used in proteins, and some amino acids (also sometimes referred to as *residues*) are coded for by multiple codons. There are also some codons that indicate a beginning or end of the translation process, called *start codons* and *stop codons* respectively. Stop codons signal the end of translation and do not encode an amino acid, while start codons indicate the start of translation and also encode an amino acid. The most common, but not the only, start codon is AUG (which codes for the amino acid methionine). A long stretch of codons not containing, but terminated by, a stop codon is called an *open reading frame* (*ORF*).

The region bounded on the $5'$ end by the beginning of the transcribed mRNA (i.e. the start of transcription) and on the $3'$ end by the start of translation is known as the $5'$ *UTR* ($5'$ untranslated region). Similarly the region bounded on the $5'$ end by the end of translation and on the $3'$ end by the end of transcription is called the $3'$ UTR. The general structure of a gene, including these regions, is shown in Figure 2.2.

Despite being described as coding segments earlier, exons are not necessarily translated. They are precisely defined as the segments of pre-mRNA that are not spliced out and so remain in the mRNA. Hence an exon may in fact contain a UTR, as well as coding sequence. To make this distinction, a coding exon is sometimes referred to as a *CDS* (coding sequence). In particular, the CDS terminology is specified for use in the GTF[1] gene annotation format [41], although it has also been used in the older GFF[2] format [64].

[1]Gene Transfer Format. `http://ardor.wustl.edu/GTF2.html`
[2]Generic Feature Format. `http://song.sourceforge.net/gff3.shtml`.

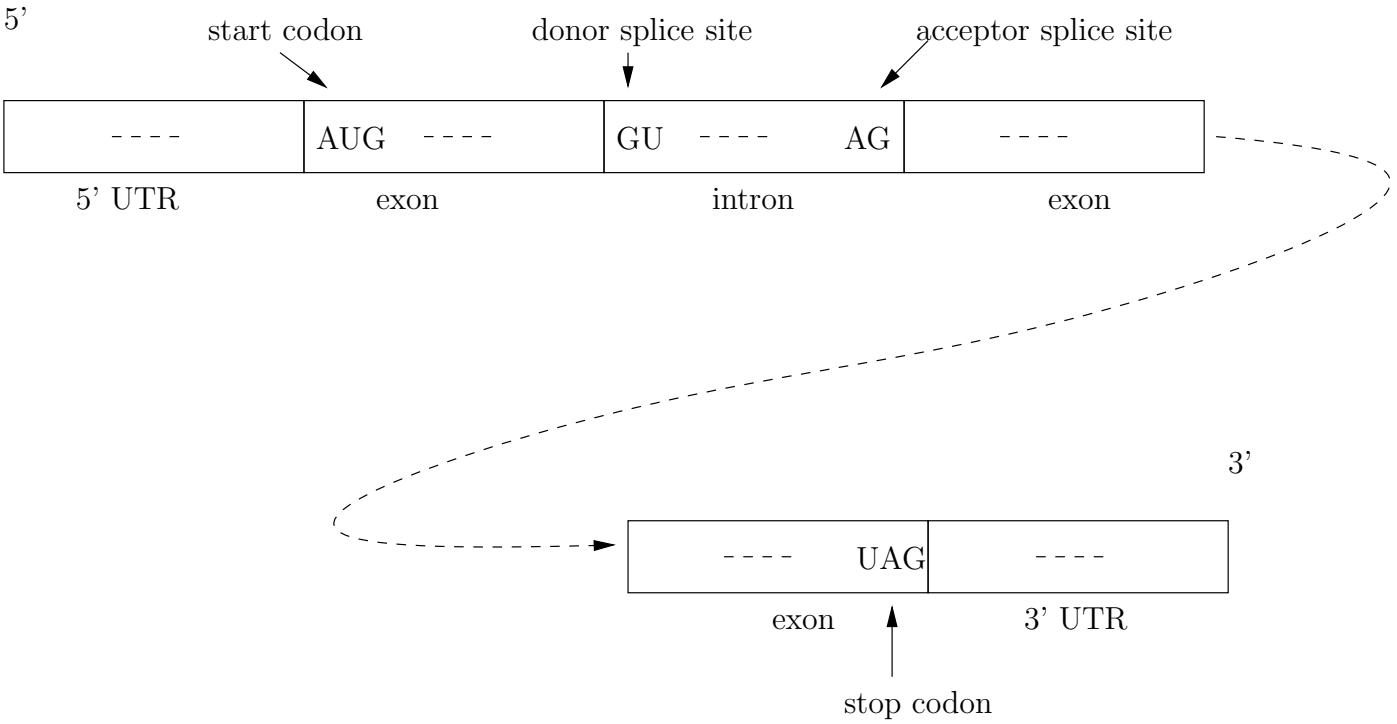



Figure 2.2: Structure of a gene on transcribed pre-mRNA.

It should not be thought that the UTR, as it does not code for for a protein, has no function. On the contrary, signals related to gene expression (not discussed in this thesis) are often contained in the UTR. For example, a motif in the 3′ UTR of *Plasmodium* species has been found to be involved in post-transcriptional gene silencing [35].

As codons are three nucleotides long, there are three possible translations of a given sequence, depending on where the translation starts. Only one of these *frames* gives the correct protein, and the recognition of the start codon determines the correct frame. It is easy to see that the accidental omission of a single nucleotide will result in subsequent codons being incorrectly interpreted (a so-called *frame-shift* error), which can be a significant problem in genomics due to the presence of errors in genome sequencing. Splicing does not necessarily occur on codon boundaries, so a codon (including a start or stop codon) can be partly in one exon and partly in another.

The mechanism by which codons in mRNA are translated to amino acids is by means of *tRNA* (transfer RNA), which are small RNA molecules that bind to a codon on one surface and the amino acid that it encodes on another surface. The phenomenon that many alternative codons for an amino acid differ only in the third base (known as third base pair *wobble*) is due to some tRNA molecules being such that they only require the first two bases to match exactly ([2], p. 338).

## 2.3 Comparative genomics

Much can be learned by comparing the genomes of related species. For example, the process of evolution tends to lead to coding sequences being more highly conserved than

non-coding sequences, and this can be used to help find coding sequences in two related species [25]. Known genes in one species can also be used to locate the corresponding genes in a a related species. This is, in fact, the strategy we examine in thesis, and it is examined in general in Section 3.3 and in particular application to *P. vivax* in Chapter 4. The aim of this section is simply to explain the necessary terms.

There has been some confusion over the term *synteny* in the literature [60], so we will be careful to use it correctly here. Synteny (or syntenic) refers to "gene loci on the same chromosome regardless of whether or not they are genetically linked by classic linkage analysis" [60]. It is often used to refer to conservation of gene order, however.

We are therefore careful to distinguish synteny from the different concepts of *homology*, *orthology* and *paralogy*. Two loci are homologous if they have similar sequences as a result of having evolved from a common ancestral gene. Homology can be either orthology or paralogy. Two genes are orthologous if they are homologs as a result of a species diverging into two different species — the gene in one of the resultant species is an ortholog of that in the other. Paralogous genes are those that are homologs as a result of a gene duplication event within a species.

A phylogeny is an evolutionary tree showing the ancestral relationships between species or sequences. Phylogenetics and the inference of phylogenies based on sequence similarity, or other quantitative data, is an entire research area on its own, and is described in some detail by Felsenstein [24].

## 2.4 Genome sequencing and annotation

The aim of genome sequencing projects is to find the entire genetic sequence of an organism, make it available to researchers, and derive useful information from it, such as the full set of genes in the genome, and the proteins expressed by the organism. The science and technology involved in studying the genomes of organisms has come to be known as genome science or "genomics", and the application of computer science and information technology in these fields as computational biology and bioinformatics ([29], Ch. 1).

We may distinguish (at least) two major stages in genome projects: first, *sequencing*, and second, *annotation*. Genome sequencing is an interesting laboratory and computational problem in itself, and high throughput systems now exist for (at least partly) automated genome sequencing. An overview of such technology is given in Chapter 2 of Gibson and Muse [29]. Only a brief summary of the technology sufficient to understand the nature of the data used in this thesis is given here.

The so-called *shotgun sequencing* approach to sequencing that is used in most current genome projects ([29], p. 82) (including the malaria genome projects) proceeds by breaking the genome into many small pieces that can be sequenced, then using computer programs to assemble the resulting large number (hundreds of thousands ([29], p. 86)) of overlapping sequences into *contigs*.

The *coverage* of a sequence is the number of times a nucleotide is represented in a random raw sequence, and it is generally regarded that fivefold to tenfold (or 5× to 10× to use a common notation for coverage) coverage is necessary for a high quality finished sequence ([29], p. 87). A *finished* sequence is regarded as a complete sequence with no gaps and an accuracy greater than 99.9% [12]. The *finishing* process (closing gaps,

resolving ambiguities, etc.) can take as much time as the initial sequencing effort ([29], p. 87). Hence *partial shotgun coverage* sequences, which may be 4× to 6× coverage [12], are often released to allow researchers to work with preliminary data, although they may contain significant errors.

Full genome sequencing generates extremely large amounts of data, for example, the human genome is over 3000 Mb (i.e. 3000 million bases; often the notation Mbp, for million base pairs, is used instead) long ([29], p. 17). The amount of sequence information is growing rapidly; the Wellcome Trust Sanger Institute[3] (the largest sequencing centre in Europe) alone generates 30 Mbp of sequence per day.[4]

Genome annotation is the process of identifying genes (although this particular part of the process is also referred to be the more specific term "gene finding") in a genomic sequence, and determining the protein coded for by the gene. A genome sequence on its own is of limited practical use – annotation is vital in order to obtain useful information from the sequence, for the purposes of identifying possible drug targets for example.

Perhaps the most obvious method of finding genes is to use the techniques of molecular biology to isolate the actual transcript. This means that we find the mRNA sequence after the cell has performed splicing, solving this problem for us. This is, in fact, done, by using *cDNA* (complementary DNA), which is DNA reverse-transcribed from the mRNA, so we end up with the DNA sequence of a gene, with the introns already removed by the cellular processes. cDNAs, however, are usually too long to be sequenced as a single unit, and as a result it is usually necessary to use multiple "expressed sequence tags" instead. An *expressed sequence tag* (EST) is a short (400 to 600 bases long) contiguous subsequence of a transcribed DNA sequence (i.e. a short cDNA subsequence) [62].

cDNA synthesis is an error-prone process ([29], p. 95) and lacks the speed, reliability and automation of genome sequencing. In addition, using these alignments to predict gene structures is not simple in itself, due to the presence of paralogs and low quality mRNA sequences among other issues [64].

A further shortcoming of cDNA and EST approaches is that they often represent only partial mRNA, and genes which are expressed at a low level or only under specific conditions are often not represented at all [54].

As a result of these difficulties, and due to the large volumes of sequence data being generated, the preferred method for determining the amino acid sequence encoded by a genome is through interpretation of the genomic sequence itself ([2], p. 506). As has been explained in the previous sections, gene finding and annotation is certainly not as simple as finding the open reading frames, and translating codons therein by the genetic code. Such things as splicing mean that this problem is in fact extremely challenging: a particular application of a technique for solving some of these problems is the subject of the present thesis.

## 2.5 Malaria genomics

The shotgun sequencing strategy, described in Section 2.4, was used in sequencing the *Plasmodium* species *vivax* [12] and *falciparum* [27, 38, 33, 28], which infect humans and

[3] `http://www.sanger.ac.uk`

[4] Dr Chris Peacock (Sanger Institute), Bio21 seminar, 19 July 2006.

| Species, strain | Institute(s) | Status | Publication(s) |
|---|---|---|---|
| *P. falciparum* | Sanger, TIGR, Stanford | Finishing | Gardner *et al.*[27], Hyman *et al.*[38], Hall *et al.*[33], Gardner *et al.*[28] |
| *P. berghei* | Sanger | 8× coverage | Hall *et al.*[35] |
| *P. chabaudi* | Sanger | 8× coverage | Hall *et al.*[35] |
| *P. yoelii* | TIGR | 8× coverage | Carlton *et al.*[13] |
| *P. vivax* | TIGR | 10× coverage | unpublished |

Table 2.1: Status of *Plasmodium* genome sequencing projects.

*yoelii yoelii*, *berghei* and *chabaudi* [13], which infect rodents. *Plasmodium* species have 14 chromosomes containing a total of approximately 23 Mbp [13]. A useful review of comparative malaria genomics is given by Hall and Carlton [34].

As malaria parasites are host-specific, and it has not been possible to maintain a complete life-cycle *in vitro* culture for *P. vivax*, *P. vivax* research has not advanced as quickly as it has for *P. falciparum* [56]. The lower priority accorded to *P. vivax* research due to the much lower mortality [55] associated with it, compared to *P. falciparum*, has also been a contributing factor. As described in Section 2.4, gene annotation using ESTs is a difficult process; this is particularly the case for *P. vivax*, as parasite mRNA has had to be obtained from infected human patients, and separated from host and *P. falciparum* mRNA [56]. Nevertheless, more than 21 000 *P. vivax* ESTs have been sequenced (albeit with high redundancy) [20]. However the stringent filtering and clustering applied to ESTs to build the TIGR Gene Indices resulted in fewer than 200 tentative consensus sequences and fewer than 600 singleton ESTs in the *P. vivax* Gene Index [47].[5]

The current[6] status of the *Plasmodium* genome sequencing projects for which sequence data is available from PlasmoDB [6] is shown in Table 2.1. Status information was obtained from project websites.[7]

GC-content (sometimes the complementary measure AT-content is used) varies both between species and within a genome. This difference between species is particularly marked between different *Plasmodium* species. As can be seen from Table 2.2[8], *P. falciparum* and the rodent malaria parasite *P. yoelii* have similar (very low) GC-content, while *P. vivax* has a much higher GC-content. The closely related rodent malaria parasites *P. berghei* and *P. chabaudi*, as well as *P. yoelii*, also have a very low GC-content, as they have high average nucleotide identity [35].

It has been speculated, based on EST analysis, that *P. vivax* genes differ markedly in GC-content depending on the region of the chromosome on which they reside [56]. Further evidence for the truth of this conjecture has been found by the *P. vivax* genome sequencing

[5]The figures in the cited paper were for release 0.5 of the *P. vivax* Gene Index. The current release, 2.0, contains over 6000 unique sequences (`http://compbio.dfci.harvard.edu/tgi/cgi-bin/tgi/gimain.pl?gudb=p_vivax`).

[6]September 2006

[7]Sanger: `http://www.sanger.ac.uk/Projects/Protozoa/`, TIGR: `http://www.tigr.org/tdb/e2k1/pva1/new.shtml`, PlasmoDB: `http://www.plasmodb.org`.

[8]*P. vivax* value from the *P. vivax* Genome Project website `http://www.tigr.org/tdb/e2k1/pva1/intro.shtml`. *P. falciparum* and *P. yoelii* values are from Carlton *et al.* [13]

| Species | GC-content |
|---|---|
| *P. falciparum* | 19.4% |
| *P. yoelii* | 22.6% |
| *P. vivax* | 45% |

Table 2.2: GC-content of the genomic DNA of some *Plasmodium* species.

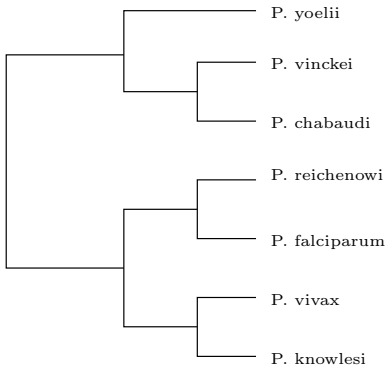



Figure 2.3: A possible *Plasmodium* species tree.

project, which established that the ends of *P. vivax* chromosomes are considerably more AT-rich than other regions [12]. However, *P. falciparum* has a uniform GC-content with the exception of a small region of extremely high AT-content on each chromosome [33]. Regions of differing GC-content such as this are known as *isochores* and their biological significance is unknown [12].

Such differences in GC-content would cause problems with *ab initio* gene finders which use statistical models where GC-content is used as an attribute to help determine if a region is coding. In addition, it means that genome alignments on the nucleotide level between, for example, *P. falciparum* and *P. vivax* are unlikely to be successful.

The evolutionary relationships between malaria species are not yet fully understood. A possible malaria species tree from Bourgon *et al.* [8] is shown in Figure 2.3.

# Chapter 3

# Computational gene finding

Apart from gene finding by aligning sequences against ESTs, we distinguish two major methods of computational gene finding. One, *ab initio* gene finding, locates genes, exons and introns, using only the genomic DNA sequence itself. This is possible due to various statistical features of genome sequences such as the signals described in Section 2.2.1. The other finds genes by using homology information between the target species and one or more related species. We may further distinguish the latter method into two classes, *comparative* and *reference based* gene finding. Comparative gene finding uses only alignments between the different (related) species to find homologous regions and predict genes and exons. Reference based [17] methods use additional information such as known gene annotations in one of the species to aid prediction of genes in another.

Note that *ab initio* and comparative or reference based methods are not mutually exclusive and some gene finding algorithms make use of both, and that some techniques, such as hidden Markov models, are used in many of the methods.

## 3.1 Sequence similarity and alignment algorithms

Sequence alignment tools are some of the most important in bioinformatics; the BLAST program [5] in particular is the most commonly used bioinformatics tool ([29], p. 88).

Sequence alignment is the problem of aligning two sequences (DNA or amino acid sequences in this case) in such a way that we have the *most probable* alignment, that is, insertions/deletions (resulting in a *gap* in one of the sequences) and mismatches are minimized, subject to some probabilistic model of sequence evolution. Sometimes we may want to constrain alignments to be *ungapped*, and alignment tools often have an option to constrain the alignments to not contain gaps (if they allow gaps normally).

One of the fundamental algorithms for solving this problem is the Smith-Waterman algorithm [68] which uses dynamic programming to find an optimal local alignment, using a matrix to avoid repeating calculations arising from the recurrence relation describing the alignment problem. This requires an $m \times n$ matrix, where $m$ and $n$ are the lengths of the sequences to be aligned [68]. Many optimizations and heuristics have been implemented over the years to make programs such as NCBI BLAST [5] and WU-BLAST [30] useful and efficient.

Alignment and search tools such as BLAST require a *substitution matrix* to assign penalties to mismatches based on their probabilities. Substitution matrices in general,

and a particular kind constructed for use with GeneMapper, are discussed in Section 4.2.1.

Not only do alignments allow us to discover homologous genes by finding the best alignments between sequences in two species (although we should note that this alone does not allow us to distinguish paralogs from orthologs [17]), they allow large databases of DNA or proteins to be searched for sequences similar to a query sequence. When used in this manner, we refer to a "hit" on the database when our query sequence can be aligned with part of the database sequence. BLAST specifically refers to such a locally optimal alignment as a "*high scoring [segment] pair*"[1] (*HSP*) [5]. BLAST gives each HSP an *E-value*, which is the number of hits with the same (or better) similarity score that would be expected to occur purely by chance, given the sizes of the input sequence and the searched database [5, 29]. Hence smaller E-values mean a hit is more significant.

Such local alignments give us a simple method of gene finding, and perhaps more importantly, verification of genes predicted by some other method. We can search a database of known genes for hits to the coding sequence of our predicted gene, or search a database of known proteins with the translation of the coding sequences of our predicted gene. In addition, programs, such as sim4 [26], have been developed to align cDNA with genomic DNA sequences (allowing for introns) specifically for the purpose of verifying predicted genes against cDNA evidence. Of course, since this depends on the existence of gene and protein databases, or ESTs, it isn't much help for a gene that codes for a protein not yet known, or not deposited in such a database.

As well as such local alignment tools, there are global alignment tools such as BLASTZ [66], MAVID [9], and MUMmer [46] for finding whole-genome alignments of (multiple) related species in order to identify sets of homologs. Such alignments can be used to predict genes in related species, as will be discussed in Section 3.3. As will be described in Section 4.2.1, these programs could also be useful in generating a species-specific substitution matrix.

## 3.2 *Ab initio* gene finding

*Ab initio* gene finders predict structural elements such as genes and exons in genomic sequences using a statistical model of the genome. Hidden Markov models (HMMs) [63] have proved to be well suited to this task, and books have been written that are wholly [45] or in large part [23] devoted to the use of HMMs in bioinformatics.

A hidden Markov model consists of a set of states, a set of output symbols, and probability distributions over state transition probabilities and output symbol emission probabilities, as well as an initial state probability distribution. It may be thought of as a probabilistic finite-state machine, where the states also emit symbols according to certain probability distributions. The states are "hidden", in that the model is based on the idea that we observe a sequence of symbols, which the HMM generates, but we do not observe the states.

Typically, the set of states is determined by the creator of the model according to their knowledge of the system being modelled, and the parameters of the model (the

[1] I enclose "segment" in brackets, as, although HSP is defined thus in Altschul *et al.* [5], the "segment" seems to be omitted in common use.

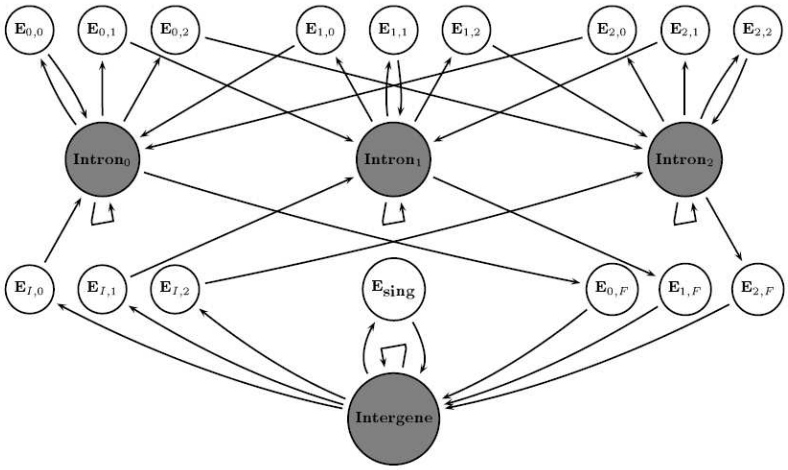



Figure 3.1: A GPHMM for alignment and exon prediction, from Pachter *et al.* [59].

state transition and symbol emission probabilities) are obtained by "training" the model on an observed sequence that has been annotated with the states of the model that generated the sequence. Given this training sequence, maximum likelihood techniques can be employed to determine the parameters which were most likely to account for the observed sequence.

Once the parameters have been learned, an HMM can be used to predict the state transition sequence that has the maximum likelihood of generating an observed sequence, using the Viterbi decoding algorithm [63].

In the context of *ab initio* gene finding, the states of the HMM will be designed to represent features that are to be found, such as exons and introns, and the alphabet is just {A, T, C, G}. Hence a path through the HMM for a given sequence classifies regions of that sequence into exons, introns, intergenic regions, and so on, thereby predicting genes and exons in the sequence.

Some generalizations of HMMs, specifically generalized HMMs (GHMMs, sometimes also known as duration or semi-Markov models) and pair HMMs, are often used in gene finding applications. GHMMs assign an explicit probability distribution to the duration of each state, rather than using self-transitions. This allows state duration probability distributions to be non-geometric [63]. Pair HMMs operate on pairs of sequences and can be used to align the sequences [23]. GHMMs and pair HMMs can be combined into GPHMMs and used to simultaneously align sequences and predict genes, as described in Pachter *et al.* [59], from which Figure 3.1 is taken. In this figure, the $E_{xy}$ states represent different exon types (initial, internal in each of the phases, final) followed or preceded by introns in different phases.

*Ab initio* gene finders based on GHMMs include GENSCAN [10], GenomeScan [74], Phat [15, 72], TigrScan and GlimmerHMM [52, 53] and SNAP [43]. GeneMark.hmm ES-3.0 [50] is also based on a GHMM, but differs from the other gene finders mentioned previously in that it is an unsupervised model, i.e. it does not require annotated training sequences. It achieves this by an iterative procedure whereby the Viterbi training algo-

rithm is run to estimate model parameters (with some initial parameter settings for the first iteration), predictions are made with these parameters, and the process is iterated with the predictions as the input training sequence, until a convergence criterion is met. Very good results with this algorithm are reported [50], which we also find to be the case, as will be described in Section 4.3.

## 3.3 Comparative and reference based gene finding

Comparative gene finding systems use sequence similarity between related organisms to improve accuracy. This technique is based on the principle that coding regions tend to be more conserved than non-coding regions [3]. One of the early comparative gene finding systems is Twinscan [44, 25]. Twinscan extends the GENSCAN GHMM by computing a joint probability of the GENSCAN prediction and the "conservation sequence". The latter consists of the designation of "aligned", "unaligned" or "mismatched" to each nucleotide in the target sequence based on high-scoring local alignments (from BLAST) with the informant sequence. This approach led to a significant improvement in the accuracy of gene predictions in the human genome using the draft mouse sequence [25].

Another comparative gene finding system is SLAM, the first implementation of a GPHMM to simultaneously align and predict genes in orthologous sequences [3]. The state space of the SLAM GPHMM is a refinement of that shown Figure 3.1. SLAM was tested on a single homologous gene pair between *P. falciparum* and *P. vivax*, where it correctly found 9 exon pairs, of 13 exons in *P. falciparum* and 14 in *P. vivax* [3]. It is important to note that the fact that the number of exons differs in these homologous genes violates an assumption underlying such GPHMM systems [3]; namely that features (such as exons and introns) are emitted in pairs [51]. There were very few *P. falciparum*-*P. vivax* orthologous gene sequences available at that time, and we are unaware of any more recent results using SLAM with *P. falciparum* and *P. vivax*.

The TWAIN system [51], or, more specifically, the OASIS component of TWAIN, is also a GPHMM system. Majoros *et al.* [51] discuss the possibility of new work to enable GPHMMs such as OASIS to be able to relax the assumption described above without increasing the time and space requirements of the algorithm to impractical levels. To the best of our knowledge no progress on this has been published to date, however.

### 3.3.1 GeneMapper

The GeneMapper reference based annotation system [17, 16] uses gene annotations in one species (the reference species) to predict gene structure in the sequence of another, related, species (the target species). It makes use of a "bottom up" algorithm, predicting the ortholog of each reference exon in the target sequence, then combining these predictions to determine gene structure. The ExonAligner module of GeneMapper finds the best alignment of the reference exon with a subsequence of the target sequence using a version of the Smith-Waterman algorithm [68]. ExonAligner enhances the dynamic programming matrix in order to model codon evolution and allow for sequencing errors and frame-shifts. Codon evolution is modelled with a "COD matrix", which is discussed in detail in Section 4.2.1.

The GeneMapper system consists of three stages, as follows:

1. Highly conserved exons are mapped, by using BLAST to find approximate locations of orthologous exons. The most significant BLAST hit is then extended to give an approximate location of the orthologous exon in the target sequence. ExonAligner is then used to predict the exact ortholog.

2. Linearity of transcription is used to map exons missed by stage 1. That is, mapped exons are used to find the approximate location of as yet unmapped exons based on the assumption that conserved exons do not change their relative positions between orthologous genes in related species. The exact position is then found by a process similar to that of stage 1.

3. In the first two stages, it has been assumed that the orthologous genes have the same number of exons. In this stage, exon fusion and splitting is detected, using the fact that introns have a minimum length; if two adjacent exon predictions are closer than this, an exon fusion event is predicted to have occurred. Similarly, exon splitting is detected by searching for gaps greater than the minimum intron length with splice sites at their ends.

GeneMapper uses an external splice site model to constrain predicted exons to have splice sites at their ends. This is discussed further in Section 4.2.2.

The time complexity of the GeneMapper algorithm on a single gene is

$$\mathrm{O}(\sum_{i=1}^{N} l_i^2)$$

where $N$ is the number of exons in the gene and $l_i$ is the length of the $i$th exon.

GeneMapper is also capable of using sequences from multiple species to improve the accuracy of annotations. This facility has not been exploited in our investigations, however, as there is only one *Plasmodium* species with a sufficiently high quality annotation at this time.

The exon fusion/splitting model described in item 3 above is that published in [17], and which is implemented in the version of GeneMapper used in this thesis. A more sophisticated model of exon fusion and splitting based on a pair HMM was used in a modified version of GeneMapper for studying intron gain and loss in the evolution of gene structure [16].

## 3.4 Evaluating gene finders

Gene finders are generally evaluated with the sensitivity (Sn) and specificity (Sp) measures defined by Bursèt and Guigó [11] as:

$$\mathrm{Sn} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3.1}$$

$$\mathrm{Sp} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{3.2}$$

where TP, TN, FP, and FN are counts of true positives, true negatives, false positives, and false negatives, respectively. A true positive is counted when a nucleotide is correctly

predicted as coding, and a false positive when it is incorrectly predicted as coding. Similarly, a true negative is counted when a nucleotide is correct predicted as non-coding, and a false negative when it is incorrectly predicted as non-coding.

It is worth noting that Equation 3.2 is not the usual definition of specificity. Since the frequency of noncoding nucleotides tends to be much higher than that of coding nucleotides, TN tends to much much larger than FP. This results in the traditional specificity measure, defined as

$$\mathrm{Sp}_0 = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{3.3}$$

producing very large values which are not indicative of the actual gene finder performance. In machine learning and data mining literature [70], the alternative specificity measure of Equation 3.2, which is used to overcome this problem, is usually referred to as "precision" to avoid confusion with specificity as defined by Equation 3.3. It is, however, traditionally the case in gene structure prediction literature that the metric defined by Equation 3.2 is referred to as "specificity" [11].

As well as calculating these metrics on the nucleotide level, they are calculated on the exon and gene level. An exon is considered correctly predicted only when there is an exact match between the actual and predicted exons, i.e. both boundaries of the predicted exon match those of the actual exon exactly. Similarly, genes are required to match exactly to be considered true positives.

The Eval [42, 41] software package computes these metrics from the output of gene finding systems, and was used to generate the sensitivity and specificity figures in Section 4.3.

## 3.5 Data exchange formats

As is frequently the case in computing disciplines, bioinformatics has a multiplicity of incompatible standards. As long ago as 2000, it was noted that a common format for gene annotations was required for a genome annotation project, and the GFF format was used for that project [64]. By 2002, this format was considered as at least a potential standard [54].

Since then, the situation has become more complex. GFF is up to version 3, and hasn't maintained continuity in what its name stands for. It has previously been known as both "General Feature Format" and "Gene Feature Finding format" [64] but now stands for "Generic Feature Format"[2]. In addition, a GFF-like format called GTF[3] (Gene Transfer Format), now up to revision 2.2, is being used by some projects. Specifically, the gene finding evaluation tool used in this thesis, Eval [42], requires GTF, not GFF annotations. This has necessitated writing scripts to convert GFF, often in program or project specific forms, or other, even more specific, formats, to GTF in order to evaluate them. As part of this process, a publicly available Python GFF library[4] was enhanced to work with the attributes specified in GFF 3.

[2] `http://song.sourceforge.net/gff3.shtml`

[3] `http://ardor.wustl.edu/GTF22.html`

[4] Knudsen 2002, Python GFF, `http://sourceforge.net/projects/pythongff`

Another difficulty in bioinformatics applications is that the standard practice for transferring BLAST query results and alignments into other programs is to parse the formatted (for human readability) BLAST output. Of course, this results in things breaking whenever the BLAST version is updated, and even when this is stable causes fragility due to the formatting changing to fit sequence names on a line and so on. This is mitigated somewhat by the availability of libraries to parse BLAST output, such as those available in BioPerl [69], yet it can still be problematic. For example, in writing a Python script to generate COD matrices (see Section 4.2.1) from TBLASTX alignments, it was necessary to use Perl for part of the process as the BioPython[5] BLAST parser was not compatible with the version of BLAST being used. Even then it was found that some (but not all) of the sequences had names too long for the BLAST output and so an extra parameter had to be added to signal that truncation of names was occurring. It also meant that more CPU time was used in parsing human-readable BLAST output than was spent by BLAST generating the alignment: in a typical run for generating a COD matrix, TBLASTX took 105 CPU minutes to generate the alignments, but parsing the result with BioPerl took 296 CPU minutes.

Fortunately sequence data is almost always in FASTA format, which consists simply of a description line and the sequence consisting of A, T, C, G characters. The only complication is that sometimes N (for an unknown base) or other ambiguous base characters are used. However, these characters are defined by the IUBMB[6] so, unlike annotation file formats, consistency is maintained.

[5] `http://www.biopython.org`

[6] "Nomenclature for Incompletely Specified Bases in Nucleic Acid Sequences", `http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html`

# Chapter 4

# Applying the GeneMapper system to *Plasmodium vivax*

## 4.1 Data sources

Sequences and annotations of all of the *Plasmodium* species used in this thesis were downloaded from PlasmoDB[1] release 5.0. PlasmoDB is a central database for *Plasmodium* data [6]; the original sources of the data and the publications (if any) are shown in Table 2.1. As the *P. vivax* data is as yet unpublished, we are unsure of the sources and reliability of the annotations. It is likely to be a manually curated fusion of different sources of evidence including the results from various *ab initio* gene finders, homology information, and EST data. Nevertheless, in the absence of anything known to be better, we are using these annotations as the "gold standard" against which to compare computational gene finding results. The *P. falciparum* data is considered relatively reliable, it is in the "finishing" stages of sequencing, has been studied for longer and has a number of published papers describing it (see Table 2.1). It has been suggested, however, based on long EST sequencing, that even in the *P. falciparum* annotation, sometimes 5′ (initial) exons are missed, especially short initial exons[2].

We may be able to improve the quality of the annotations in the case of missing initial exons by using a tool such as sim4 [26] to align ESTs against genomic DNA and find the 5′ exon based on its cDNA transcript.

A set of five known orthologous regions between *P. falciparum* and *P. vivax*, containing 156 genes in each, was supplied by Tobias Sargeant of the Walter and Eliza Hall Institute[3]. This data is used as our "development test" set — we run GeneMapper using the *P. falciparum* annotations as the reference to annotate the *P. vivax* sequences, then use the *P. vivax* annotations from the data to evaluate GeneMapper accuracy.

The orthologous genes in the first of our five regions (the one with the fewest genes) are shown in Figure 4.1, created using the PlasmoDB genome browser. The synteny[4] maps in PlasmoDB were created using Mercator[5] [22]. Note that there are some differ-

[1] `http://www.plasmodb.org`

[2] Tobias Sargeant, personal communication, May 2006.

[3] Personal communication, 12 July 2006.

[4] An incorrect use of the term synteny — see Section 2.3.

[5] According to the "Data Sources" section on the PlasmoDB website.

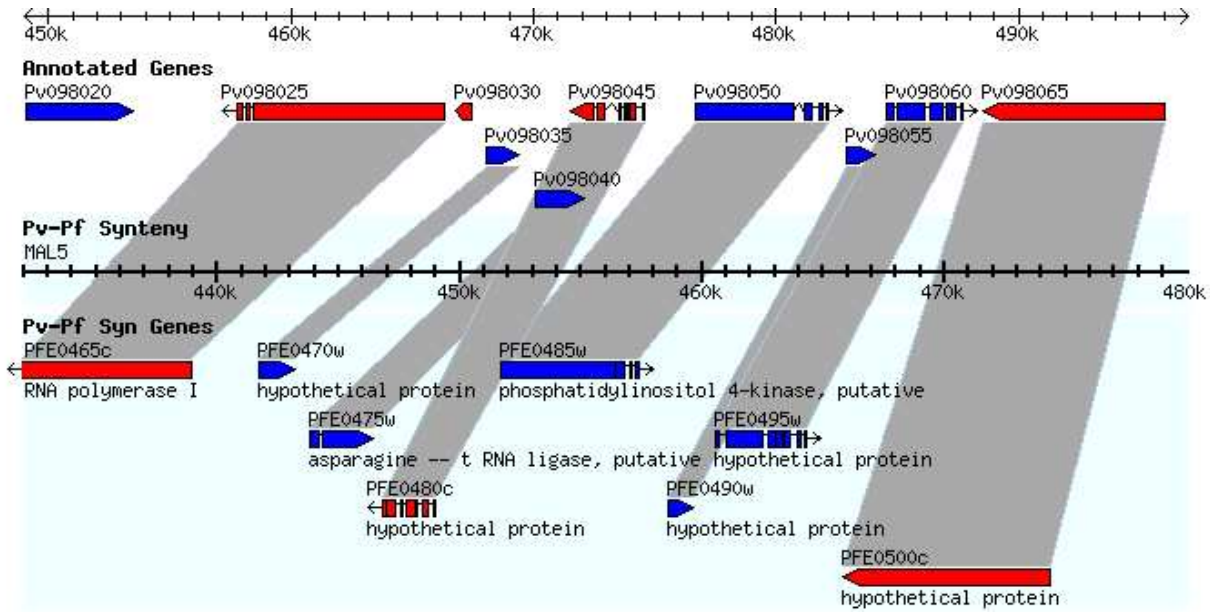



Figure 4.1: Orthologous genes in *P. falciparum* chromosome 5 and *P. vivax* contig 7027.

ences between this and the manually selected region. In the development test set region, *P. vivax* gene Pv098020 is known to be orthologous to *P. falciparum* gene PFE0450w[6], and *P. vivax* gene Pv098030, with no *P. falciparum* ortholog, is not included.

In the case of *ab initio* gene finders that require training data, we train on the *P. vivax* sequence and annotation, with the development test set regions removed. Note that we call this a "development test" set as we are, in some sense, training GeneMapper on it by trying different COD matrices and parameters and evaluating against this set. By restricting to this known set we avoid overfitting, as we hope to be able to run GeneMapper on many more regions in the *P. vivax* sequence by using an automated method such as ROSE [51] (see Section 4.2.3), OrthoMCL [49] (see Section 5.2.3), or Mercator to find orthologous regions.

## 4.2 Methods

GeneMapper [17] was downloaded[7] and installed on our research system, along with necessary support software WU-BLAST [30], StrataSplice [48], and BioPerl [69]. GeneMapper was modified to work on our system, and some constants were changed so that it worked with the *Plasmodium* data. As will be described in Section 4.2.2, it was also modified to be able to use GeneSplicer [61] rather than StrataSplice as the splice site model, however, for reasons described in that section, this modification was not enabled when generating the results shown in Section 4.3.

[6]Apparently Mercator does not find these as orthologs. Querying PlasmoDB for orthologs, however, shows that OrthoMCL [49] clusters these genes as orthologs, as does our own OrthoMCL run (see Section 5.2.3).

[7]`http://bio.math.berkeley.edu/genemapper/geneMapper.tgz`

Bourne shell and Python scripts were written to read orthologous regions described in GFF 3 format (see Section 3.5) and create the appropriate input files, one gene at a time, for GeneMapper. For genes on the reverse strand, the reversed complement of the sequence is provided, as GeneMapper assumes everything is on the forward strand. Shell scripts were then used to run GeneMapper on each gene, remap the co-ordinates in the resulting GFF output to their original origin in the whole sequence, concatenate the output and convert it to GTF format. Eval [42] was then used to evaluate the results against the *P. vivax* annotation from PlasmoDB. In addition, `gff2ps` [1] is run in order to generate a graphical view of the results. On our research system[8] GeneMapper typically takes between 30 seconds and one minute to process each gene, resulting in a total running time for the five region development test set of approximately two hours.

### 4.2.1 Codon matrices

Alignment algorithms require a substitution matrix to assign measures of evolutionary similarity between (usually) amino acids in order to score mismatches in the alignment. The most frequently used matrices are empirical, i.e. derived directly from observed amino acid frequencies. One of the earliest of these, still in common use, is the *PAM* (point accepted mutation) matrix of Dayhoff *et al.* [21], although the most frequently used substitution matrices are probably *BLOSUM* (block substitution) [36] matrices. PAM and BLOSUM matrices are $20 \times 20$ (there being 20 amino acids) *log-odds* (logarithm of odds) matrices; each entry is the log of the ratio of observed to expected frequencies of an amino acid. Such matrices are derived from proteins at a particular evolutionary distance.

Purely empirical approaches to modelling protein evolution (such as PAM) do not force the model to have any relation to the genetic code. Muse & Gaut [58] and Goldman & Yang [31] developed a codon-based model of protein change. As described by Felsenstein [24], a standard base substitution model is assumed, but the probability of the acceptance of the changes is given by a formula with a probability of rejection increasing as the chemical properties of the two amino acids differ to a greater degree. The coefficients of these models can be estimated by empirical study of amino acid or base changes, and we obtain a $64 \times 64$ matrix that should more accurately reflect longer-term change than amino acid based models like PAM.

The theoretical shortcomings of empirically-derived matrices have led to the development of other derivations based on well understood statistical techniques such as expectation maximization [37]. A thorough treatment of the statistics of estimating substitution matrices is given by Yap and Speed [73]. A more recent development in the estimation of codon substitution matrices is to allow for codon substitution probabilities to be neighbour-dependent [19] (rather than the assumption of independence that has hitherto been made). These developments, however, have not so far gained widespread use. This is most probably due to the higher level of complexity involved in deriving more sophisticated models, and their lack of significant improvement in practice over the simpler PAM and BLOSUM matrices.

GeneMapper takes as a parameter a particular kind of substitution matrix, known as a COD matrix (codon matrix). This particular matrix was introduced by Chatterji

[8] Dual 3.2 GHz Xeon, 8GB memory, GNU/Linux 2.6.9

and Pachter [17], and, as the name suggests, models codon rather than amino acid or nucleotide evolution, and so is a $64 \times 64$ matrix. It is similar to PAM and BLOSUM matrices, and the human-chimp matrix supplied with GeneMapper was learned from whole genome alignments in the UCSC genome browser database [40]. As the UCSC genome browser does not contain any *Plasmodium* data we need our own procedure for constructing COD matrices. We learned from Chatterji[9] that the script he used to generate this matrix was specific to the UCSC genome browser database format. However the elements of the matrix are simply a normalized frequency count of codon changes in the alignment, considering coding segments only (only exon evolution is being modelled). Entries must be nonzero, and stop codons are constrained only to change to stop codons (this differs from the approach taken by Muse & Gaut, where stop codons are excluded from the model [58]). Frame-shifts cannot be recovered from, so affected codons must be ignored and the process continued at a later point. Based on this information, we wrote a script to construct our own COD matrices to model *Plasmodium* exon evolution.

We considered using BLASTZ for generating alignments, however BLASTZ is tuned for sensitivity in aligning neutrally evolving sequences [66], whereas we want to align only coding sequences, so it is probably not the most suitable tool for the task. In addition, the output format of BLASTZ seems difficult to understand and deal with, in contrast to the simple, and well documented[10] formats generated by MUMmer [46], or the human-readable format generated by BLAST.

We decided, in order to simplify the process, to use ungapped alignments. Therefore we could use NCBI BLAST (specifically TBLASTX, which finds matching proteins by translating both the query and database nucleotide sequences in all six reading frames, and cannot produce gapped alignments). A parser for NCBI BLAST output is available in the BioPerl [69] toolkit (see Section 3.5). Only ungapped alignments with a minimum length of 100 amino acids and with an E-value of $10^{-30}$ or lower were used. The E-value cutoff was chosen to try to ensure alignments were significant, and was based on the value used by Merino *et al.* [56] for annotating *P. vivax* ESTs by means of a BLASTX search against *P. falciparum* genome databases. We consider this value to be conservative enough to generate alignments at the amino acid level that are significant. It is considerably more conservative, for example, than the value of $10^{-15}$ used for finding BLASTP matches between *P. falciparum* and *P. yoelii* by Carlton *et al.* [13] and for finding BLASTX matches between *P. vivax* ESTs and a protein database by Cui *et al.* [20].

A Perl script using BioPerl, based on `blast-hits.pl`, which is used by GeneMapper as part of its processing, was written to parse the resulting BLAST output. A Python script using BioPython[11] was written to generate a COD matrix by using this parsed output and the sequence data supplied to TBLASTX to calculate the codon change frequencies. A pseudocount was used to ensure that no zero probabilities occur in the matrix, and stop codons were only counted if they changed to stop codons. We allow a stop codon to change to a different stop codon, although the matrices supplied with GeneMapper allow a stop codon to change only to itself.[12] Each row of the matrix is then

[9] Sourav Chatterji, personal communication, 17 August 2006.

[10] `http://mummer.sourceforge.net/manual/`

[11] `http://www.biopython.org`

[12] Sourav Chatterji, personal communication, 17 August 2006.

normalized by dividing each element by the row sum, resulting in a stochastic matrix. GeneMapper was modified so that it does not force the COD matrix to be symmetric.

The $64 \times 64$ codon matrix is indexed from 0 to 63. The index for a codon consisting of the three nucleotides $x_0x_1x_2$ is:

$$16b_0 + 4b_1 + b_2 \tag{4.1}$$

where, for each $i$, $0 \leq i \leq 3$:

$$b_i = \begin{cases} 0 & \text{if} \quad x_i = \text{A} \\ 1 & \text{if} \quad x_i = \text{C} \\ 2 & \text{if} \quad x_i = \text{G} \\ 3 & \text{if} \quad x_i = \text{T} \end{cases}$$

Then the $i, j$ entry of the COD matrix $A$, $0 \leq i \leq 63$, $0 \leq j \leq 63$, is:

$$A_{ij} = \frac{c_{ij} + 1}{\sum_{k=0}^{63}(c_{ik} + 1)} \tag{4.2}$$

where $c_{ij}$ is the count of the number of times codon $i$ in the genomic coding sequence of the first species is aligned with codon $j$ in the genomic coding sequence of the second species. However we impose the constraint that:

$$\begin{array}{llll} \text{for} & i \in S, & c_{ij} \neq 0 & \text{only if} \quad j \in S \\ \text{for} & i \notin S, & c_{ij} \neq 0 & \text{only if} \quad j \notin S \end{array} \tag{4.3}$$

where

$$S = \{48, 50, 56\}$$

is the set of indices of stop codons.

The COD matrix $A$ has the property that, for $0 \leq i \leq 63$,

$$\sum_{j=0}^{63} A_{ij} = 1 \tag{4.4}$$

but it is not necessarily symmetric.

Figure 4.2 shows a graphical interpretation of four of the COD matrices. The intensity of each of the $64 \times 64$ pixels in the image is proportional to the probability value in the COD matrix (raised to the power of 0.4 to allow more image intensity space to low probability values), with a black pixel representing 1 and a white pixel representing 0. Each image is captioned with the two species from which it was generated. A row in the image represents the probability of a codon from the first species being substituted with a codon from the second species in each column (i.e. the first species codon varies down the columns, and the second across the rows). The origin is the top left where the codon is AAA and the codons are ordered by the third base pair varying fastest, in the order A, C, G, T, as described by equation 4.1.

From these images we can see some properties of the codon evolution model inferred from the genome alignments that the COD matrix summarizes. For all of the matrices there is, to different degrees, a higher intensity evident on codons along the main diagonal.

This reflects the fact that the single highest probability for any codon is not to change. There is also a four by four block structure evident in all of the matrices, due to the way they are indexed by codon; any four by four block, in phase with the origin, represents codons differing only in the third base pair. Four by four blocks of these blocks represent codons differing in the second and possibly third base pair. Three vertical and horizontal white lines are also visible. These are the entries corresponding to the stop codons (TAA, TAG, TGA), which are constrained only to change to stop codons.

The human-chimp matrix, Figure 4.2(a), has a very strong diagonal line and very little probability on any off-diagonal element, indicating a low probability of codon changes. This is likely to be a reflection of the very small evolutionary distance between human and chimpanzee. The *Plasmodium* matrices, Figures 4.2(b), 4.2(c) and 4.2(d), have much less probability along the main diagonal and hence higher probabilities elsewhere, reflecting the higher probabilities of codons in homologous regions being different in these species. This is particularly evident in the *P. falciparum*-*P. vivax* COD matrix, Figure 4.2(d). Recall from Table 2.2 in Section 2.3, that *P. falciparum* has a very low GC-content, while *P. vivax* does not. This fact should lead us to expect such a deviation from the identity codon substitution matrix, since a substitution matrix from *P. falciparum* to *P. vivax* must assign relatively high probabilities to substitutions from A or T to G or C.

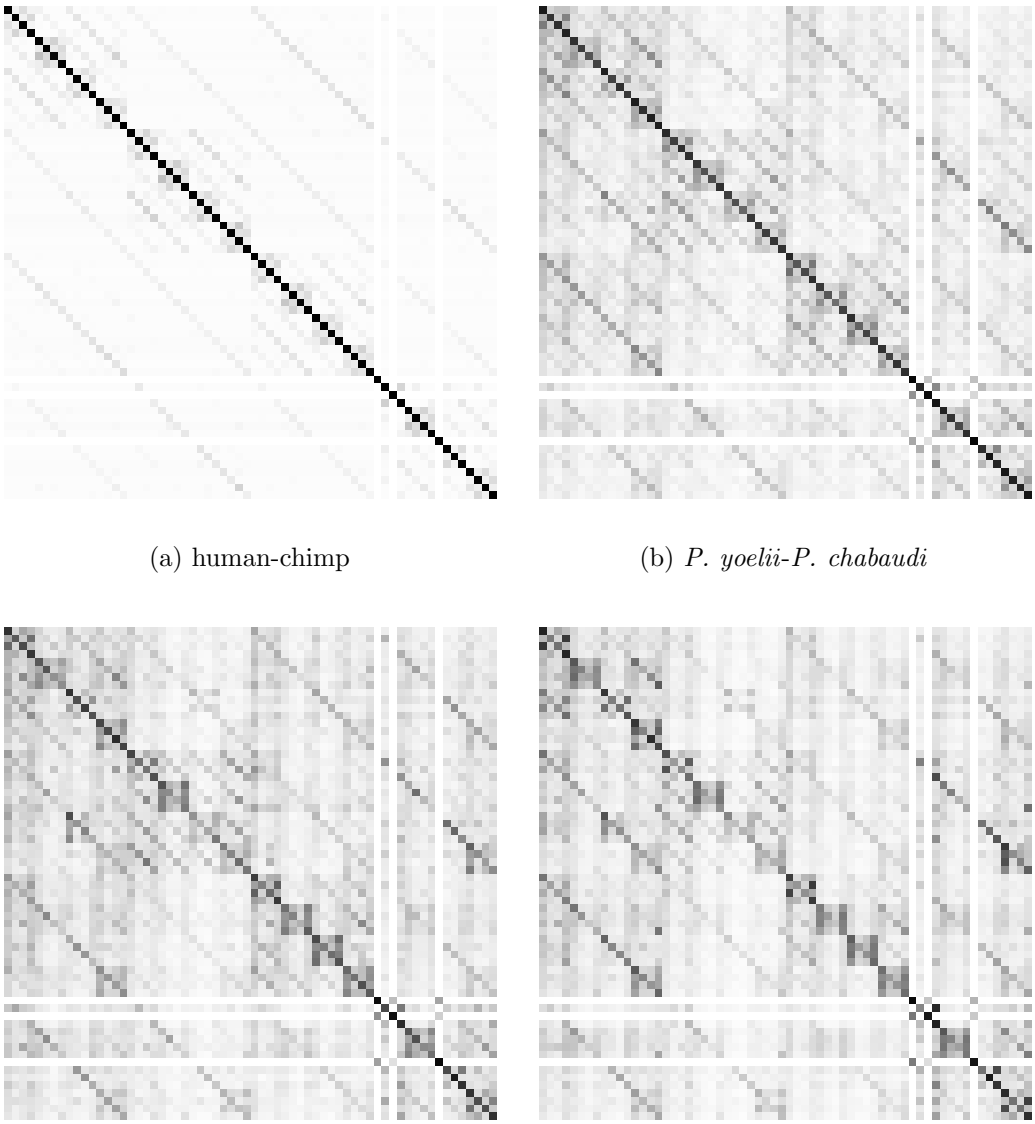
Another feature evident in Figure 4.2(d) is a pattern of vertical lines, visible as columns of high intensity pixels, with two columns of lower intensity pixels in between them, in the blocks along the main diagonal. This is again due to the very low GC content of the *P. falciparum* genome versus the moderate GC-content of the *P. vivax* genome; as the ordering of bases in the COD matrix is A, C, G, T, this pattern results from the higher probabilities of an A or a T being substituted with a different base, relative to a C or a G.

Figure 4.3 shows an enlargement of an interesting region of the *P. falciparum*-*P. vivax* COD matrix. This region shows a group of relatively high probability substitutions detailed in Table 4.1. The last column of the table shows substitution score from the BLOSUM62 matrix [36] for that amino acid substitution. Like the PAM matrices [21], these are log odds scores with a neutral score of 0, so negative scores indicate substitutions that are observed to occur less frequently than they would by chance. The lowest score in the BLOSUM62 matrix is -4, meaning that these substitutions are the least likely to occur.

Therefore, the last five rows in the table are not very interesting; they represent either synonymous substitutions or substitutions of one hydrophobic amino acid to another. Hence it is not particularly surprising that substitutions in these codons occur in such a way that A and T are less frequent and G and C correspondingly more frequent.

The first row, however, shows that an unusual substitution from phenylalanine to proline occurs with relatively high probability in our *P. falciparum*-*P. vivax* alignment. The second group of three rows shows the relatively high probability in the COD matrix of two further unusual substitutions; from cysteine to arginine and from tryptophan to arginine. The mutability of cysteine is generally low due to its chemical structure and unique functions [21], so it is particularly interesting that this substitution appears with a relatively high probability in this COD matrix.

Hence it would appear from these results that, between *P. falciparum* and *P. vivax*, the high probability of nucleotide substitutions from T to C (from the high AT-content



(a) human-chimp

(b) *P. yoelii-P. chabaudi*

(c) *P. falciparum-P. yoelii*

(d) *P. falciparum-P. vivax*

Figure 4.2: COD matrices generated from different alignments.

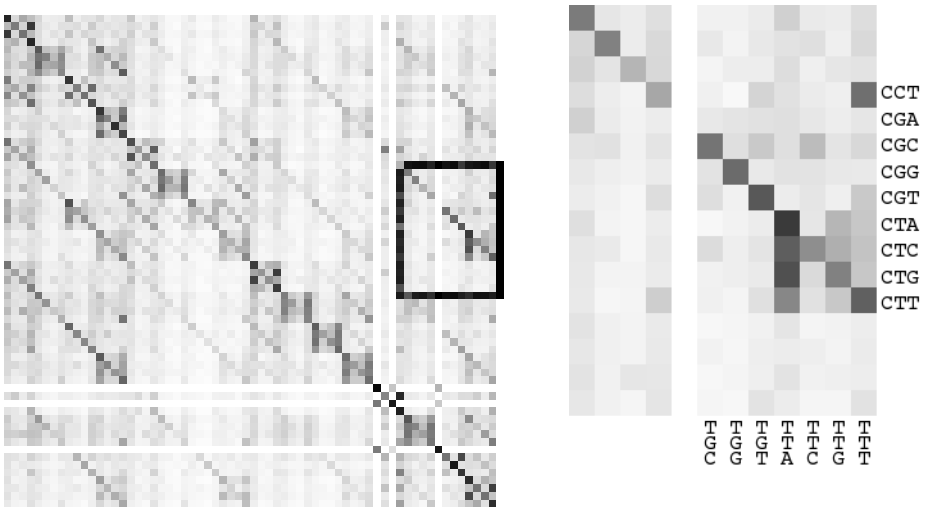



Figure 4.3: Enlargement of part of the *P. falciparum*-*P. vivax* COD matrix.

genome of *P. falciparum* to the moderate AT-content genome of *P. vivax*) is enough to outweigh the usual low probability of these particular amino acid substitutions.

### 4.2.2 Splice site models: StrataSplice and GeneSplicer

Simply by using the `top` command on Linux, it was observed that a large proportion of the GeneMapper running time was consumed by running StrataSplice, which is a human splice site model written in Java, with no source code available. As StrataSplice is only being used by GeneMapper to constrain the ends of exons to be splice sites after it has aligned them, it seemed wasteful to be spending so much time executing a Java program for this purpose. It is possibly problematic that it is a human splice site model whereas we are dealing with *Plasmodium* species. Hence we modified GeneMapper to use GeneSplicer [61], which is able to be re-trained, is distributed with source code, and is written in C and hence expected to be considerably faster than StrataSplice.

GeneSplicer uses GlimmerHMM [52] models, and is supplied with human, "malaria" (*P. falciparum*) and *P. yoelii* models, amongst others. As we were not sure of the provenance of the supplied "malaria" and *P. yoelii* models (only the models, no training data, were supplied), a *P. falciparum* model was created by using a Bourne shell script to convert the entire PlasmoDB *P. falciparum* annotation into the correct format for the GlimmerHMM training program, which is then run to train the model. A *P. vivax* model was created by following the same procedure with the annotation for *P. vivax*. The different models were then tested against *P. vivax* contig 7027, and the results are shown in Table 4.2.

As can be seen from the table, GeneSplicer is indeed faster (at least 16 times faster) than StrataSplice, and the supplied "malaria" model is evidently very different from the one we trained. Unfortunately, using GeneSplicer with any of these models in GeneMapper results in GeneMapper finding no exons at all. We expect that this is related to the

| *P. falciparum* | | *P. vivax* | | BLOSUM62 |
|---|---|---|---|---|
| codon | amino acid | codon | amino acid | score |
| TTT | Phe | CCT | Pro | -4 |
| TGC | Cys | CGC | Arg | -3 |
| TGG | Trp | CGG | Arg | -3 |
| TGT | Cys | CGT | Arg | -3 |
| TTA | Leu | CTA | Leu | 0 |
| TTA | Leu | CTG | Leu | 0 |
| TTC | Phe | CTC | Leu | 0 |
| TTG | Leu | CTG | Leu | 0 |
| TTT | Phe | CTT | Leu | 0 |

Table 4.1: Some relatively high probability *P. falciparum*-*P. vivax* codon substitutions from Figure 4.3.

| Program | Training organism | Sites found | CPU seconds |
|---|---|---|---|
| StrataSplice | human | 173210 | 84.31 |
| GeneSplicer | human | 2761 | 2.71 |
| GeneSplicer | "malaria" | 515 | 2.70 |
| GeneSplicer | *P. yoelii* | 5842 | 2.80 |
| GeneSplicer | *P. falciparum* | 1554 | 4.80 |
| GeneSplicer | *P. vivax* | 15338 | 3.52 |

Table 4.2: Comparison of GeneSplicer and StrataSplice on *P. vivax* contig 7027.

fact that StrataSplice is finding at least 10 times more putative splice sites than GeneSplicer, and suppose that GeneMapper is relying on this extreme overprediction. There are several threshold parameters that are adjustable so further work on changing these is probably called for.

### 4.2.3 Comparative gene finders for comparison

Although it is open source, we were unable to get TWAIN [51] working on our research system. Hence the attempt to obtain results with TWAIN for comparisons was abandoned, however one part of TWAIN, ROSE (the Region-of-Synteny[13] Extractor) is a Perl script that processes an alignment generated by MUMmer [46] and hence should work on our system. ROSE seems ideal for generating orthologous pairs for input to GeneMapper, and we intend to do so to assess GeneMapper on a larger scale than our manually selected orthologous regions.

We have also not been able to reconfigure SLAM to work on data other than human-mouse.

Twinscan [44, 25] 3.5 was downloaded[14] and installed (along with its required support software RepeatMasker [67] and RepBase [39]), and run on the five *P. falciparum* and *P. vivax* regions containing the development test set, with default (*C. elegans*-*C. briggsae*) parameters. Parameter re-estimation for Twinscan requires a process not dissimilar to the creation of COD matrices for GeneMapper (see Section 4.2.1); we have not yet done this, so the Twinscan results are not a particularly fair comparison, given the large effort we have put into re-estimating GeneMapper parameters. Twinscan is the only comparative gene finder we were able to test.[15]

### 4.2.4 *Ab initio* gene finders for comparison

For gene finders that require training, we used the entire PlasmoDB *P. vivax* sequence and annotation, less the five contigs used in our development test set, which were used as the test set for evaluating the gene finders. For all such gene finders, Bourne shell scripts, or in some cases, Perl scripts based on those supplied with Jigsaw [4], were written to convert the PlasmoDB GFF annotation to the formats required to train the gene finders, and the output of the gene finders to GTF format suitable for use with the Eval program.

GeneZilla 1.0, the new incarnation of TigrScan [53], was downloaded[16] and installed. We were unable to train GeneZilla due to the training programs not compiling on our system. Hence GeneZilla could only be run with the downloadable *P. falciparum* model, as binaries for the training programs were not downloadable.

[13]Note that this is an incorrect use of "synteny" as discussed in Section 2.3. ROSE is actually a Region-of-Homology Extractor, however that doesn't work as nicely as an acronym. We presume the authors are well aware of this as they correctly use the term orthology rather than synteny in their paper [51] and the ROSE documentation (`http://www.cbcb.umd.edu/software/rose/Rose.html`).

[14]`http://ardor.wustl.edu/software/download/Twinscan_3.5_src.tar.gz`

[15]Twinscan 3.0 would not work on our data either, but Twinscan 3.5 when it became available seems to have fixed whatever bug caused this.

[16]`http://bioinformatics.org/download.php/genezilla/`

Phat [72, 15] (release 20001212) was downloaded[17] and installed. Phat comes pre-trained for both *P. falciparum* and *P. vivax* so no further work was required, other than writing a script to convert the output to GTF.

SNAP [43] (release date 2006-05-18) was downloaded[18] and installed, and run with supplied default and *Drosophila melanogaster* parameters.

GeneMark.hmm ES-3.0 [50, 7] is available as a webserver[19], and is unusual in that it is an *ab initio* gene finder, yet requires no training annotation as it is "self-training". It requires sequences less than 5 Mbp each, and between 10 Mbp to 50 Mbp in total. We supplied our development test set, plus enough extra contigs to make slightly more than 10 Mbp in total.

GlimmerM 2.5.1 and GlimmerHMM 2.2.0 [52] were downloaded[20], installed, and trained on *P. vivax* as described above.

## 4.3 Results

Tables in this section show sensitivity and specificity figures, as percentages, for the nucleotide, exon, and gene level. These figures were calculated with the Eval [42] software, as described in Section 3.4. In addition, a balanced F-measure between sensitivity and specificity, calculated by

$$F = \frac{2 \times \mathrm{Sn} \times \mathrm{Sp}}{\mathrm{Sn} + \mathrm{Sp}}$$

is shown. Figures have been rounded to zero decimal places as percentages, and the highest value in each column is shown in boldface.

Results for the comparative and *ab initio* gene finders as described in Sections 4.2.3 and 4.2.4 are shown in Table 4.3.

Performance measurements of GeneMapper, with different COD matrices and evolutionary distance multipliers (the power to which the COD matrix is raised), run on the regions described in Section 4.1, are shown in Table 4.4. All of the COD matrices were generated from coding sequence alignments only, as described in Section 4.2.1. An exception is the matrix labelled "*P. falciparum*-*P. vivax* all", which was generated from the entire genomic sequence of both species. The performance of GeneMapper using the *P. falciparum*-*P. vivax* COD matrix, with distance multiplier 1, broken down by performance individually on each of the five regions in the development test set, is shown in Table 4.5.

In order to understand gene predictions better, and where errors may be occurring, it is useful to have a visual representation of the predictions. Figure 4.4 shows GeneMapper and Phat predictions on the first of our development test regions. This figure was generated with `gff2ps` [1], and is interpreted as follows. There are three data sources represented: `ApiDB` is the "gold standard" data, `FullPhat` is the `Phat` *ab initio* prediction, and `ExonAligner` is the GeneMapper prediction (ExonAligner is the name of an

[17] `http://bioinf.wehi.edu.au/Phat/phat.tar.gz`

[18] `http://homepage.mac.com/iankorf/`

[19] `http://exon.gatech.edu/GeneMark/gmseuk.cgi`

[20] `ftp://ftp.tigr.org/pub/software/GlimmerM/GlimmerM-2.5.1.tar.gz`, `ftp://ftp.cbcb.umd.edu/pub/software/glimmerhmm/GlimmerHMM-2.2.0.tar.gz`

| Program | Trained | Nucleotide | | | Exon | | | Gene | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Sn. | Sp. | F | Sn. | Sp. | F | Sn. | Sp. | F |
| GeneZilla | *P. falciparum* | 65 | 59 | 62 | 3 | 0 | 1 | 2 | 1 | 1 |
| Phat | *P. falciparum* | 72 | 82 | 77 | 7 | 4 | 5 | 11 | 5 | 7 |
| Phat | *P. vivax* | 97 | 92 | 95 | 16 | 10 | 12 | 5 | 6 | 5 |
| SNAP | "minimal" | 63 | 86 | 72 | 8 | 6 | 7 | 14 | 9 | 11 |
| SNAP | *D. melan.* | 66 | 95 | 78 | 17 | 13 | 15 | 5 | 7 | 6 |
| GeneMark.hmm-ES3 | N/A | **99** | **97** | **98** | **72** | 62 | 67 | 53 | 55 | 54 |
| GlimmerM | *P. vivax* | 94 | 86 | 90 | 25 | 6 | 10 | 4 | 2 | 2 |
| GlimmerHMM | *P. vivax* | **99** | **97** | **98** | **72** | **68** | **70** | **64** | **57** | **60** |
| Twinscan | defaults | 57 | 94 | 71 | 6 | 8 | 7 | 0 | 0 | 0 |

Table 4.3: *Ab initio* and comparative gene finder performance (%).

| COD matrix | dist | Nucleotide | | | Exon | | | Gene | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Sn. | Sp. | F | Sn. | Sp. | F | Sn. | Sp. | F |
| *D. melanogaster-D. yakuba* | 1 | 54 | **99** | 70 | 54 | 74 | 62 | 31 | 43 | 36 |
| *D. melanogaster-D. yakuba* | 2 | 61 | **99** | 75 | 56 | 73 | **64** | 35 | **45** | 40 |
| *D. melanogaster-D. yakuba* | 10 | **68** | 98 | **80** | **58** | 70 | **64** | **38** | 44 | 41 |
| *D. melanogaster-D. yakuba* | 15 | 67 | 98 | **80** | **58** | 71 | **64** | **38** | 44 | 41 |
| *D. melanogaster-D. yakuba* | 20 | 67 | 98 | **80** | **58** | 70 | 63 | **38** | 44 | 41 |
| human-chimp | 1 | 21 | 98 | 34 | 44 | **78** | 57 | 21 | 37 | 26 |
| human-chimp | 20 | 60 | **99** | 74 | 56 | 73 | **64** | 35 | 44 | 39 |
| human-chimp | 40 | 63 | **99** | 77 | 57 | 72 | **64** | 37 | **45** | 40 |
| human-chimp | 50 | 64 | **99** | 78 | **58** | 72 | **64** | **38** | **45** | 41 |
| human-chimp | 60 | 65 | 98 | 78 | **58** | 72 | **64** | **38** | **45** | **42** |
| *P. falciparum-P. vivax* | 1 | 64 | **99** | 78 | **58** | 72 | **64** | 37 | **45** | 41 |
| *P. falciparum-P. vivax* | 2 | 66 | 98 | 79 | 57 | 71 | 63 | 37 | 43 | 40 |
| *P. falciparum-P. vivax* all | 1 | 64 | 98 | 78 | 54 | 68 | 60 | 31 | 38 | 34 |
| *P. falciparum-P. vivax* all | 2 | 50 | 97 | 66 | 53 | 71 | 61 | 29 | 40 | 34 |
| *P. falciparum-P. yoelii* | 1 | 64 | **99** | 78 | 57 | 72 | **64** | 37 | **45** | 41 |
| *P. falciparum-P. yoelii* | 2 | 64 | 98 | 78 | 55 | 68 | 61 | 33 | 40 | 36 |
| *P. falciparum-P. yoelii* | 20 | 20 | 95 | 33 | 48 | **78** | 59 | 22 | 39 | 29 |
| *P. yoelii-P. chabaudi* | 1 | 60 | **99** | 75 | 57 | 72 | **64** | 35 | 44 | 39 |
| *P. yoelii-P. chabaudi* | 2 | 63 | **99** | 77 | 57 | 71 | 63 | 36 | 44 | 40 |
| *P. yoelii-P. chabaudi* | 3 | 64 | 98 | 77 | 55 | 69 | 61 | 33 | 40 | 36 |
| *P. yoelii-P. chabaudi* | 20 | 30 | 96 | 46 | 51 | 77 | 61 | 26 | 41 | 32 |

Table 4.4: GeneMapper performance (%) for different COD matrices.

| | Nucleotide | | | Exon | | | Gene | | |
|---|---|---|---|---|---|---|---|---|---|
| Region | Sn. | Sp. | F | Sn. | Sp. | F | Sn. | Sp. | F |
| 1 | 43 | 100 | 60 | 44 | 65 | 52 | 11 | 14 | 13 |
| 2 | 81 | 99 | 89 | 47 | 54 | 50 | 42 | 45 | 43 |
| 3 | 49 | 98 | 66 | 66 | 78 | 72 | 43 | 52 | 47 |
| 4 | 61 | 98 | 75 | 55 | 74 | 63 | 32 | 43 | 37 |
| 5 | 91 | 98 | 94 | 40 | 50 | 44 | 32 | 35 | 33 |

Table 4.5: Breakdown by region of GeneMapper performance (%) with *P. falciparum*-*P. vivax* COD matrix.

algorithm which is part of GeneMapper). Each of the data sources has two tracks; the ones above the centre line are features on the forward strand, and those below the centre line are features on the reverse strand. The features, which are exons, are rendered as blocks, with the colour representing the phase of the exon. Note that the `ApiDB` exons are always black, as this annotation does not provide phase information. Genes are represented by over- or under-lines on the exons, with introns then being represented by the gaps between the blocks within a single over- or under-line. The height of a block represents the confidence score, which is not provided in the `ApiDB` annotation or by GeneMapper, so these blocks are all the same size. The small height of the `FullPhat` features in this figure demonstrates a relatively low confidence assigned to these features by Phat.

## 4.4 Discussion

First we should consider the results obtained from *ab initio* and comparative gene finders, as a baseline to see if reference-based gene finding is able to do better. These results are shown in Table 4.3, from which it is apparent that the best performing gene finder is GlimmerHMM, with GeneMark.hmm ES-3.0 not far behind. All the others have considerably lower performance figures, although we should exclude those that were not trained on *P. vivax* from this comparison in order to be consistent, leaving only Phat. It is perhaps not surprising that GlimmerHMM is the best performing, as we believe GlimmerHMM predictions may have been used to aid in the annotation of the *P. vivax* data[21] we are using as the "gold standard". This data is not yet published, however, so this is not definitely known to be the case. Perhaps more surprising is the performance of GeneMark.hmm ES-3.0, which performs nearly as well as GlimmerHMM and does not even require annotated training data. This result was certainly much better than expected, especially in light of the fact that the authors had found that the *P. falciparum* genome, with a comparatively low number of introns per gene, was not a good target for the GeneMark.hmm ES-3.0 algorithm.[22]

Turning now to the GeneMapper results shown in Table 4.4, we find that no com-

[21]Tobias Sargeant, personal communication.

[22]Mark Borodovsky, 2006 (unpublished seminar). Winter School in Mathematical and Computational Biology, 26 June, The University of Queensland.

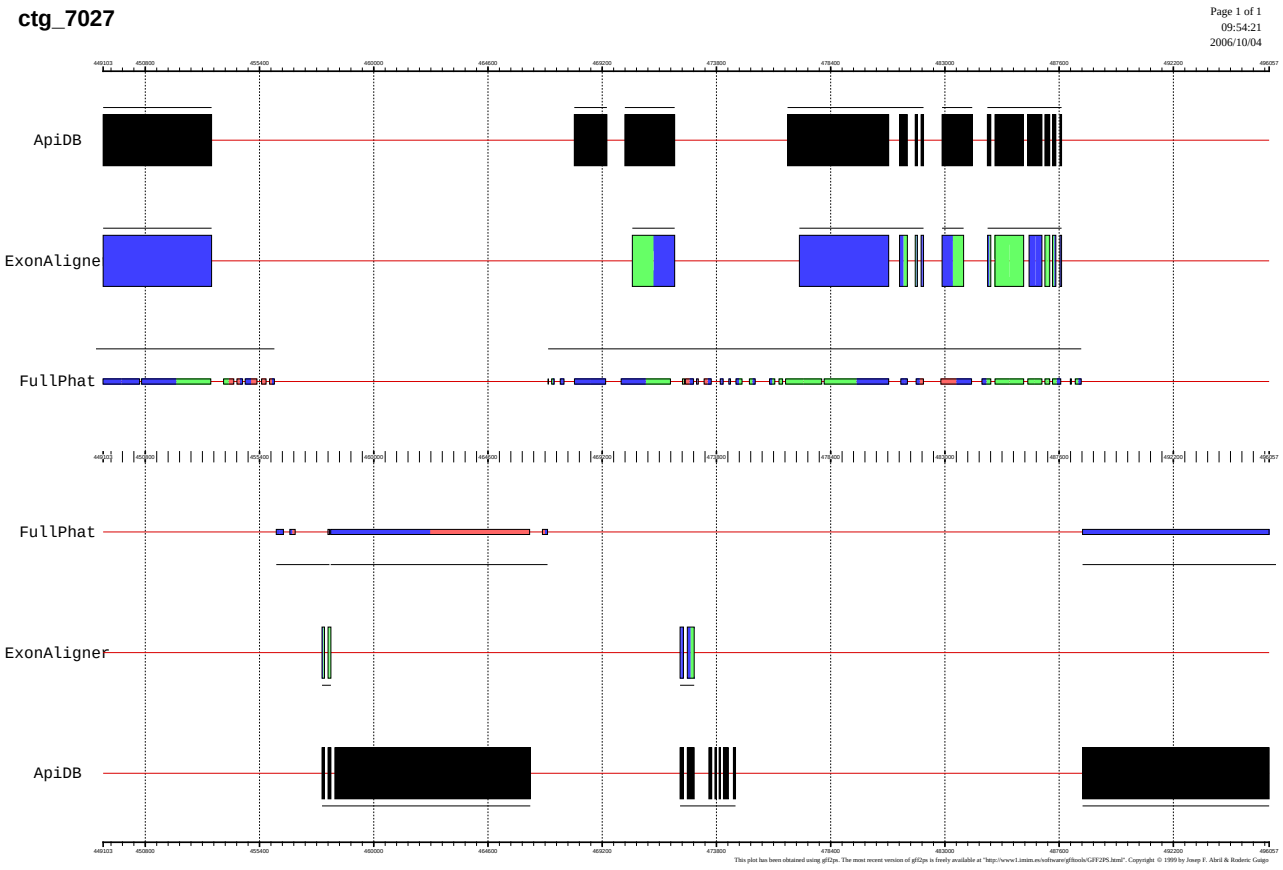



Figure 4.4: GeneMapper exon predictions in a region of *P. vivax* contig 7027.

bination of COD matrices and distances was able to achieve results as good as those of GlimmerHMM or GeneMark.hmm ES-3.0. We find these results somewhat surprising given GeneMapper's excellent performance on human-mouse annotations [17], and the fact that intron loss/gain rates (to which GeneMapper may be sensitive due to the relatively simple exon fusion/splitting model currently implemented [17, 16]) are thought to be relatively low (comparable to vertebrate rates) in *Plasmodium* [65].

Further, the choice of COD matrix appears not to have a very significant effect on the results, although the distance multiplier must be set at an appropriate value for the COD matrix. Hence, the best results for the *P. falciparum-P. vivax* are for distance 1, and decrease with higher distances as we would expect. The same is true for all of the Plasmodium COD matrices that were used, although it is not as clear in these cases that this is true by necessity. Both the human-chimp and *D. melanogaster-D. yakuba* matrices perform similarly with distances up to a maximum (of 20 and 60 respectively) before performance falls off. We expect relatively high values such as these to be necessary for these matrices as they are generated from species with very similar genomes. Further, the fact that increasing the distance multiplier quickly degrades performance with the *Plasmodium* COD matrices, and has a much slower effect with the human-chimp and *D. melanogaster-D. yakuba* matrices is to be expected as the latter are much closer to identity COD substitution matrices than the former.

From Table 4.5, we can see that GeneMapper has significantly lower gene level sensitivity and specificity in region 1 of the development test set, relative to the other four regions. Figure 4.4 shows the results in this region, and indicates some areas in which GeneMapper may not be performing well. There are two instances where single-exon genes are missed entirely (one on the forward strand, and one on the reverse strand). We can also notice something unusual about the third gene (from the left) on the forward strand, which is a single exon gene. GeneMapper (ExonAligner) correctly predicts a single exon, but the exon is shown with two colours, meaning it contains a frame-shift. Although, on the scale of the figure, it may look as if this exon is predicted exactly, this is not really the case. Examination of the GeneMapper GFF output shows that it is in fact predicted as being 1706 base pairs in length, while the ApiDB annotation shows this exon as being 2007 base pairs long (and neither end matches exactly). Hence this exon (and therefore the gene) is not counted at all towards the true positives for calculating sensitivity and specificity at the exon and gene level, as predictions must be exact. As to the frame-shift, this is because the predicted exon length is not a multiple of three. If this were an internal exon in a multi-exon gene, this would not be very remarkable since splicing does not have to occur in phase with the reading frame. However a single-exon gene does not have splice sites at its ends, and should therefore always contain an exact number of codons. At first we suspected that there may have been an assumption in GeneMapper that all exons end in splice sites, due to its having been initially used for human-mouse annotations where single-exon genes may be comparatively rare. However, an examination of the Projector data set of human-mouse orthologs [57] which was used for evaluating GeneMapper [17] shows that this data set contains 31 single-exon human genes, so it seems unlikely that the problem is so simple. We have not yet been able to determine what is causing this problem, or the problem of the entirely missing single-exon genes. We believe it may be useful to more thoroughly determine exactly what GeneMapper is taking into account in predicting the ends of single-exon genes, and suspect it may

still be related to its use of splice site models as discussed in section 4.2.2.

As well as these potential errors, we should note that we should not expect to attain 100% accuracy on this data set, since the *P. vivax* and even *P. falciparum* annotations are imperfect, and the development data set is known to contain errors.[23] This is in contrast to the Projector data set, which was carefully selected to ensure that the orthologs and annotations were as correct as possible, being fully supported by mRNA evidence [57]. Hence an upper bound on the possible accuracy is imposed both by potential errors in the "gold standard" and the *P. falciparum* annotation, as well as missing orthologs. It is, however, not possible for us to know precisely what this upper bound is.

[23]Tobias Sargeant, personal communication.

# Chapter 5

# Conclusions and future work

## 5.1 Conclusions

We have found that GeneMapper, a reference-based genome annotation system, does not achieve better performance than some *ab initio* gene finders on a manually selected set of *P. vivax* genes with known *P. falciparum* orthologs, despite its success in transferring mouse annotations to human. A number of potential reasons for this, which require further research to verify and potentially correct, have been identified. If this is done, we hope to achieve better accuracy than *ab initio* gene finders with this approach, although factors pertaining to the quality of the data limit the maximum possible accuracy to a certain, not precisely quantified, degree.

In the process of generating codon substitution matrices for *Plasmodium* species, some unusual substitutions were identified as occurring with relatively high probability. These substitutions constitute evidence for the conjecture that non-synonymous substitutions are being driven by background change in GC-content rather than selective pressure.

## 5.2 Future work

### 5.2.1 *P. falciparum-P. vivax* codon evolution

We believe it would be valuable to verify that the unusual substitutions revealed by the *P. falciparum-P. vivax* COD matrix described in section 4.2.1 really are significant (and not, for example, some artifact of our choice of BLAST parameters), possibly by regenerating the matrix based on different alignments. This could include whole-genome alignments generated by MAVID [9], and MUMmer [46]. We could then use background frequencies of codons in *P. vivax* to transform the COD matrix to a log-odds matrix in order to enable comparison with standard substitution matrices. If these results are verified as significant, there are further areas of interest in the COD matrix that could also be investigated.

### 5.2.2 GeneMapper improvements

As described in Section 4.4, we would like to verify our conjecture that GeneMapper is erroneously allowing the exon in a single-exon gene to be other than a multiple of three

bases long, and, if true, correct it. We would then hope that this would substantially improve its performance on the development test data set. In addition, as discussed in Section 4.2.2, we would like to be able to have GeneMapper work correctly with an alternative splice site model in order to improve efficiency and possibly obtain better results. We may be also able to improve the quality of the annotations in the case of missing initial exons by using a tool such as sim4 [26] to incorporate EST evidence.

### 5.2.3 Other orthologous gene sets

If we are able to improve GeneMapper performance enough to obtain significantly better exact exon sensitivity and specificity than the best *ab initio* gene finders on our development test data set, we would then like to investigate its predictions on other data sets, such as the serine repeat antigen (SERA) gene family [8]. For larger data sets, as described in Section 4.2.3, we could use ROSE to generate orthologous regions. In addition, the following subsection describes another way in which sets of orthologs have been generated, which could be used as input to GeneMapper.

#### OrthoMCL and OrthoMCL-DB

The Markov Clustering (MCL) algorithm [71] has been used to cluster eukaryotic genes into ortholog groups [49] and build a publicly available database of these groups [18]. However, a clustering including *P. vivax* is not yet publicly available in a usable form in the OrthoMCL database[1], and the PlasmoDB web interface provides only a web browsing capability whereby orthologs of one gene at a time can be selected. As the OrthoMCL software is freely available we used it to generate the ortholog pairs ourselves. We downloaded OrthoMCL version 1.2 and used it to find orthologs between *P. falciparum* and *P. vivax* using the *P. falciparum* and *P. vivax* annotated proteins downloaded from PlasmoDB. Based on the second-lowest WU-BLAST E-value, we set the `MAX_WEIGHT_DEFAULT` value to 298, and obtained 4474 clusters, of which 4382 contain genes from both species.

### 5.2.4 BioPython

As described in Section 3.5, an already existing GFF module for Python was enhanced to support GFF 3. It would possibly be valuable to incorporate this module into the BioPython project, which currently lacks support for parsing GFF files.

## 5.3 Acknowledgements

I thank Tony Wirth, for his supervision; Tobias Sargeant[2] for contributing many ideas for this project and for supplying *Plasmodium* data and much helpful advice on *Plasmodium* and bioinformatics in general; Terry Speed[3], via whom we obtained the idea for this research project from Lior Pachter, and for advice on substitution matrices; Lior Pachter[4],

[1] `http://orthomcl.cbil.upenn.edu/`
[2] Walter and Eliza Hall Institute of Medical Research (WEHI)
[3] WEHI and University of California (UC) at Berkeley
[4] UC Berkeley

for the idea of using GeneMapper; Sourav Chatterji[5], for writing GeneMapper and making it open source, and advice on using it; Tony Papenfuss[6] for getting me a copy of AVID. I also thank the University of Melbourne for supporting my Honours year with a Melbourne Honours Scholarship.

[5]UC Davis, formerly at UC Berkeley
[6]WEHI

# Bibliography


[1] Josep F. Abril and Roderic Guigó. gff2ps: visualizing genomic annotations. *Bioinformatics*, 16(8):743–744, 2000.

[2] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular Biology of the Cell*. Garland Science, 4th edition, 2002.

[3] Marina Alexandersson, Simon Cawley, and Lior Pachter. SLAM: Cross-species gene finding and alignment with a generalized pair hidden Markov model. *Genome Research*, 13(3):496–502, 2003.

[4] Jonathan E. Allen and Steven L. Salzberg. Jigsaw: integration of multiple sources of evidence for gene prediction. *Bioinformatics*, 21(18):3596–3603, 2005. doi:10.1093/bioinformatics/bti609.

[5] Stephen F. Altschul, Thomas L. Madden, Alejandro A. Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[6] Amit Bahl, Brian Brunk, Jonathan Crabtree, Martin J. Fraunholz, Bindu Gajria, Gregory R. Grant, Hagai Ginsburg, Dinesh Gupta, Jessica C. Kissinger, Philip Labo, Li Li, Mathhew D. Mailman, Arthur J. Milgram, David S. Pearson, David S. Roos, Jonathan Schug, Christian J. Stoeckert, and Patricia Whetzel. PlasmoDB, the *Plasmodium* genome resource. A database integrating experimental and computational data. *Nucleic Acids Research*, 31(1):212–215, 2003. doi:10.1983/nar/gkg081.

[7] John Besemer and Mark Borodovsky. GeneMark: web software for gene finding in prokaryotes, eukaryotes and viruses. *Nucleic Acids Research*, 33(Web Server Issue):W451–W454, 2005. doi:10.1093/nar/gki487.

[8] Richard Bourgon, Mauro Delorenzi, Tobias Sargeant, Anthony N. Hodder, Brendan S. Crabb, and Terence P. Speed. The serine repeat antigen (SERA) gene family phylogeny in *Plasmodium*: The impact of GC content and reconciliation of gene and species trees. *Molecular Biology and Evolution*, 21(11):2161–2171, 2004.

[9] Nicolas Bray and Lior Pachter. MAVID: Constrained ancestral alignment of multiple sequences. *Genome Research*, 14(4):693–699, 2004. `http://www.genome.org/cgi/doi/10.1101/gr.1960404`.

[10] Chris Burge and Samuel Karlin. Prediction of complete gene structures in human genomic DNA . *J. Mol. Biol.*, 268:78–94, 1997.

[11] Moises Bursèt and Roderic Guigó. Evaluation of gene structure prediction programs. *Genomics*, 34(3):353–367, 1996. doi: 10.1006/geno.1996.0298.

[12] Jane Carlton. The *Plasmodium vivax* genome sequencing project. *Trends in Parasitology*, 19(5):227–231, 2003. doi:10.1016/S1471-4922(03)00066-7.

[13] Jane M. Carlton, Samuel V. Angiuoli, Bernard B. Suh, Taco W. Kooij, Mihaela Pertea, Joana C. Silva, Maria D. Ermolaeva, Jonathan E. Allen, Jeremy D. Selengut, Hean L. Koo, Jeremy D. Peterson, Mihal Pop, Daniel S. Kosack, Martin F. Shumway, Shelby L. Bidwell, Shamira J. Shallom, Susan E. van Aken, Steven B. Riedmuller, Tamara V. Feldblyum, Jennifer K. Cho, John Quackenbush, Martha Sedegah, Azadeh Shoaibi, Leda M. Cummings, Laurence Florens, John R. Yates, J. Dale Raine, Robert E. Sinden, Michael A. Harris, Deirdre A. Cunningham, Peter R. Preiser, Lawrence W. Bergman, Akhil B. Vaidya, Leo H. van Lin, Chris J. Janse, Andrew P. Waters, Hamilton O. Smith, Owen R. White, Steven L. Salzberg, J. Craig Venter, Claire M. Fraser, Stephen L. Hoffman, Malcolm J. Gardner, and Daniel J. Carucci. Genome sequence and comparative analysis of the model rodent malaria parasite *Plasmodium yoelii yoelii*. *Nature*, 419:512–519, 2002.

[14] Jane M. Carlton, Mary R. Galinski, John W. Barnwell, and John B. Dame. Karyotype and synteny among the chromosomes of all four species of human malaria parasite. *Mol. Biochem. Parasitol.*, 101(1–2):23–32, 1999.

[15] Simon E. Cawley, Anthony Wirth, and Terence P. Speed. Phat - a gene finding program for *Plasmodium falciparum*. *Mol. Biochem. Parasitol.*, 118(2):167–174, December 2001.

[16] Sourav Chatterji. *Computational Analyses of Eukaryotic Gene Evolution*. PhD thesis, University of California, Berkeley, 2006. `http://www.cs.berkeley.edu/~souravc/thesis.pdf`.

[17] Sourav Chatterji and Lior Pachter. Reference based annotation with GeneMapper. *Genome Biology*, 7(4):R29, 2006.

[18] Feng Chen, Aaron J. Mackey, Christian J. Stoeckert Jr., and David S. Roos. OrthoMCL-DB: querying a comprehensive multi-species collection of ortholog groups. *Nucleic Acids Research*, 34(Database issue):D363–D368, 2005. doi:10.1093/nar/gkj123.

[19] Ole F. Christensen, Asger Hobolth, and Jens L. Jensen. Pseudo-likelihood analysis of codon substitution models with neighbor-dependent rates. *J. Comput. Biol.*, 12(9):1162–1182, 2005.

[20] Liwang Cui, Qi Fan, Yi Hu, Svetlana A. Karamycheva, John Quackenbush, Benjawan Khuntirat, Jetsumon Sattabongkot, and Jane M. Carlton. Gene discovery in *Plasmodium vivax* through sequencing of ESTs from mixed blood stages. *Mol. Biochem. Parasitol.*, 144(1):1–9, 2005.

[21] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, 5:345–358, 1978. Supp. 3.

[22] Colin Dewey and Lior Pachter. Mercator: Multiple whole-genome orthology map construction, 2006. `http://www.biostat.wisc.edu/~cdewey/mercator/`.

[23] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis*. Cambridge University Press, 1998.

[24] Joseph Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Sunderland, Massachusetts, 2004.

[25] Paul Flicek, Evan Keibler, Ping Hu, Ian Korf, and Michael R. Brent. Leveraging the mouse genome for gene prediction in human: From whole-genome shotgun reads to a global synteny map. *Genome Research*, 13(1):46–54, 2003. `http://www.genome.org/cgi/doi/10.1101/gr.830003`.

[26] Liliana Florea, George Hartzell, Zheng Zhang, Gerald M. Rubin, and Webb Miller. A computer program for aligning a cDNA sequence with a genomic DNA sequence. *Genome Research*, 8(9):967–974, 1998. doi:10.1101/gr.8.9.967.

[27] Malcolm J. Gardner, Neil Hall, Eula Fung, Owen White, Matthew Berriman, Richard W. Hyman, Jane M. Carlton, Arnab Pain, Karen E. Nelson, Sharen Bowman, Ian T. Paulsen, Keith James, Jonathan A. Eisen, Kim Rutherford, Steven L. Salzberg, Alister Craig, Sue Kyes, Man-Suen Chan, Vishvanath Nene, Shamira J. Shallom, Bernard Suh, Jeremy Peterson, Sam Angiuoli, Milhaela Pertea, Jonathan Allen, Jeremy Selengut, Daniel Haft, Michael W. Mather, Akhil B. Vaidya, David M. A. Martin, Alan H. Fairlamb, Martin J. Fraunholz, David S. Roos, Stuart A. Ralph, Geoffrey I. McFadden, Leda M. Cummings, G. Mani Subramanian, Chris Mungall, J. Craig Venter, Daniel J. Carucci, Stephen L. Hoffman, Chris Newbold, Ronald W. Davis, Claire M. Fraser, and Bart Barrell. Genome sequence of the human malaria parasite *Plasmodium falciparum*. *Nature*, 419:498–511, 2002.

[28] Malcolm J. Gardner, Shamira J. Shallom, Jane M. Carlton, Steven L. Salzberg, Vishvanath Nene, Azadeh Shoaibi, Anne Ciecko, Jeffery Lynn, Michael Rizzo, Bruce Weaver, Behnam Jarrahi, Michael Brenner, Babak Parvizi, Luke Tallon, Azita Moazzez, David Granger, Claire Fujii, Cheryl Hansen, James Pederson, Tamara Feldblyum, Jeremy Peterson, Bernard Suh, Sam Angiuoli, Mihaela Pertea, Jonathan Allen, Jeremy Selengut, Owen White, Leda M. Cummings, Hamilton O. Smith, J. Craig Venter, Daniel J. Carucci, Stephen L. Hoffman, and Claire M. Fraser. Sequence of *Plasmodium falciparum* chromosomes 2, 10, 11 and 14. *Nature*, 419:531–534, 2002.

[29] Greg Gibson and Spencer V. Muse. *A Primer of Genome Science*. Sinauer Associates, Sunderland, Massachusetts, 2004.

[30] W. Gish. WU-BLAST 2.0. `http://blast.wustl.edu`, 1996–2006.

[31] Nick Goldman and Ziheng Yang. A codon-based model of nucleotide substitution for protein-coding DNA sequences. *Molecular Biology and Evolution*, 11(5):725–736, 1994.

[32] Larry Gonick and Mark Wheelis. *The Cartoon Guide to Genetics*. HarperCollins, New York, updated edition, 1991.

[33] N. Hall, A. Pain, M. Berriman, C. Churcher, B. Harris, D. Harris, K. Mungall, S. Bowman, R. Atkin, S. Baker, A. Barron, K. Brooks, C. O. Buckee, C. Burrows, I. Cherevach, C. Chillingworth, T. Chillingworth, Z. Christodoulou, L. Clark, R. Clark, C. Corton, A. Cronin, R. Davies, P. Davis, P. Dear, F. Dearden, J. Doggett, T. Feltwell, A Goble, I. Goodhead, R. Gwilliam, N. Hamlin, Z. Hance, D. Harper, H. Hauser, T. Hornsby, S. Holroyd, P. Horrocks, S. Humphray, K. Jagels, K. D. James, D. Johnson, A. Kerhornou, A. Knights, B. Konfortov, S. Kyes, N. Larke, D. Lawson, N. Lennard, A. Line, M. Maddison, J. McLean, P. Mooney, S. Moule, L. Murphy, K. Oliver, D. Ormond, C. Price, M. A. Quail, E. Rabbinowitsch, M.-A. Rajandream, S. Rutter, K. M. Rutherford, M. Sanders, M. Simmonds, K. Seeger, S. Sharp, R. Smith, R. Squares, S. Squares, K. Stevens, K. Taylor, A. Tivey, L. Unwin, S. Whitehead, J. Woodward, J. E. Sulston, A. Craig, C. Newbold, and B. G. Barrell. Sequence of *Plasmodium falciparum* chromosomes 1, 3–9 and 13. *Nature*, 419:527–531, 2002.

[34] Neil Hall and Jane Carlton. Comparative genomics of malaria parasites. *Current Opinion in Genetics & Development*, 15(6):609–613, 2005.

[35] Neil Hall, Marianna Karras, J. Dale Raine, Jane M. Carlton, Taco W. A. Kooij, Matthew Berriman, Laurence Florens, Christoph S. Janssen, Arnab Pain, Georges K. Christophides, Keith James, Kim Rutherford, Barbara Harris, David Harris, Carol Churcher, Michael A. Quail, Doug Ormond, Jon Doggett, Holly E. Trueman, Jacqui Mendoza, Shelby L. Bidwell, Marie-Adele Rajandream, Daniel J. Carucci, John R. Yates III, Fotis C. Kafatos, Chris J. Janse, Bart Barrell, C. Michael R. Turner, Andrew P. Waters, and Robert E. Sinden. A comprehensive survey of the *Plasmodium* life cycle by genomic, transcriptomic, and proteomic analyses. *Science*, 307:82–86, 2005.

[36] Steven Henikoff and Jorja G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, 89:10915–10919, November 1992.

[37] I. Holmes and G. M. Rubin. An expectation maximization algorithm for training hidden substitution models. *J. Mol. Biol.*, 317:753–764, 2002.

[38] Richard W. Hyman, Eula Fung, Aaron Conway, Omar Kurdi, Jennifer Mao, Molly Miranda, Brian Nakao, Don Rowley, Tomoaki Tamaki, Fawn Wang, and Ronald W. Davis. Sequence of *Plasmodium falciparum* chromosome 12. *Nature*, 419:534–537, 2002.

[39] J. Jurka, V. V. Kapitonov, A. Pavlicek, P. Klonowski, O. Kohany, and J. Walichiewicz. Repbase update, a database of eukaryotic repetitive elements. *Cytogenetic and Genome Research*, 110(1–4):462–467, 2005.

[40] D. Karolchik, R. Baertsch, M. Diekhans, T. S. Furey, A. Hinrichs, Y. T. Lu, K. M. Roskin, M. Schwartz, C. W. Sugnet, D. J. Thomas, R. J. Weber, D. Haussler, and W. J. Kent. The UCSC genome browser database. *Nucleic Acids Research*, 31(1):51–54, 2003.

[41] Evan Keibler. Eval: A gene set comparison system. Master's thesis, Washington University in St Louis, 2003. `http://ardor.wustl.edu/eval/eval-documentation.pdf`.

[42] Evan Keibler and Michael R. Brent. Eval: A software package for analysis of genome annotations. *BMC Bioinformatics*, 4(50), October 2003.

[43] Ian Korf. Gene finding in novel genomes. *BMC Bioinformatics*, 5(59), May 2004. doi:10.1186/1471-2105-5-59.

[44] Ian Korf, Paul Flicek, Daniel Duan, and Michael R. Brent. Integrating genomic homology into gene structure prediction. *Bioinformatics*, 17(Suppl. 1):S140–S148, 2001.

[45] Timo Koski. *Hidden Markov Models for Bioinformatics*, volume 2 of *Computational Biology*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001.

[46] Stefan Kurtz, Adam Phillippy, Arthur L. Delcher, Michael Smoot, Martin Shumway, Corina Antonescu, and Steven L. Salzberg. Versatile and open software for comparing large genomes. *Genome Biology*, 5(2):R12, 2004.

[47] Y. Lee, J. Tsai, S. Sunkara, S. Karamycheva, G. Pertea, R. Sultana, V. Antonescu, A. Chan, F. Cheung, and J. Quackenbush. The TIGR gene indices: clustering and assembling EST and known genes and integration with eukaryotic genomes. *Nucleic Acids Research*, 33(Database issue):D71–D74, 2005. doi:10.1093/nar/gki064.

[48] Aaron Levine. StrataSplice — a human splice site predictor. `http://www.sanger.ac.uk/Software/analysis/stratasplice/`, 2001.

[49] Li Li, Christian J. Stoeckert Jr., and David S. Roos. OrthoMCL: Identification of ortholog groups for eukaryotic genomes. *Genome Research*, 13(9):2178–2189, 2003. doi:10.1101/gr.1224503.

[50] Alexandre Lomsadze, Vardges Ter-Hovhannisyan, Yury O. Chernoff, and Mark Borodovsky. Gene identification in novel eukaryotic genomes by self-training algorithm. *Nucleic Acids Research*, 33(20):6494–6506, October 2005. doi:10.1093/nar/gki937.

[51] W. H. Majoros, M. Pertea, and S. L. Salzberg. Efficient implementation of a generalized pair hidden Markov model for comparative gene finding. *Bioinformatics*, 21(9):1782–1788, 2005.

[52] W.H. Majoros, M. Pertea, and S. L. Salzberg. TigrScan and GlimmerHMM: two open-source *ab initio* eukaryotic gene-finders. *Bioinformatics*, 20(16):2878–2879, 2004.

[53] William H. Majoros, Mihaela Pertea, Arthur L. Delcher, and Steven L. Salzberg. Efficient decoding algorithms for generalized hidden Markov model gene finders. *BMC Bioinformatics*, 6(16), January 2005.

[54] Catherine Mathé, Marie-France Sagot, Thomas Schiex, and Pierre Rouzé. Current methods of gene prediction, their strengths and weaknesses. *Nucleic Acids Research*, 30(19):4103–4117, 2002.

[55] Kamini Mendis, Barabara J. Sina, Paola Marchesini, and Richard Carter. The neglected burden of *Plasmodium vivax* malaria. *Am. J. Trop. Med. Hyg.*, 64(1,2):97–106, 2001.

[56] Emilio F. Merino, Carmen Fernandez-Becerra, Alda M.B.N. Madeira, Ariane L. Machado, Alan Durham, Arthur Gruber, Neil Hall, and Hernando A. del Portillo. Pilot survey of expressed sequence tags (ESTs) from the asexual blood stages of *Plasmodium vivax* in human patients. *Malaria Journal*, 2(21), 2003. doi:10.1186/1475-2875-2-21.

[57] Irmtraud M. Meyer and Richard Durbin. Gene structure conservation aids similarity based gene prediction. *Nucleic Acids Research*, 32(2):776–783, 2004. doi:10.1093/nar/gkh211.

[58] Spencer V. Muse and Brandon S. Gaut. A likelihood approach for comparing synonymous and nonsynonymous nucleotide substitution rates, with application to the chloroplast genome. *Molecular Biology and Evolution*, 11(5):715–724, 1994.

[59] Lior Pachter, Marina Alexandersson, and Simon Cawley. Applications of generalized pair hidden Markov models to alignment and gene finding problems. *J. Comput. Biol.*, 9(2):389–399, 2002.

[60] Eberhard Passarge, Bernhard Horsthemke, and Rosann A. Farber. Incorrect use of the term synteny. *Nature Genetics*, 23:387, 1999.

[61] Mihaela Pertea, Xiaoying Lin, and Steven L. Salzberg. GeneSplicer: a new computational method for splice site prediction. *Nucleic Acids Research*, 29(5):1185–1190, 2001.

[62] Joan U. Pontius, Lukas Wagner, and Gregory D. Schuler. Unigene: A unified view of the transcriptome. In J. McEntyre and J. Ostell, editors, *The NCBI Handbook*, chapter 21. National Library of Medicine (USA), National Center for Biotechnology Information, 2002. `http://www.ncbi.nlm.nih.gov/books/bv.fcgi?call=bv.View..ShowTOC&rid=han%dbook.TOC&depth=2`.

[63] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

[64] Martin G. Reese, George Hartzell, Nomi L. Harris, Uwe Ohler, Josep F. Abril, and Suzanna E. Lewis. Genome annotation assessment in drosophila melanogaster. *Genome Research*, 10(4):483–501, 2000. doi:10.1101/gr.10.4.483.

[65] Scott William Roy and Daniel L. Hartl. Very little intron loss/gain in plasmodium: Intron loss/gain mutation rates and intron number. *Genome Research*, May 2006. doi:10.1101/gr.4845406.

[66] Scott Schwartz, W. James Kent, Arian Smit, Zheng Zhang, Robert Baertsch, Ross C. Hardison, David Haussler, and Webb Miller. Human-mouse alignments with BLASTZ. *Genome Research*, 13(1):103–107, 2003. `http://www.genome.org/cgi/doi/10.1101/gr.809403`.

[67] A. F. A. Smit, R. Hubley, and P. Green. RepeatMasker Open-3.1.5, 1996–2004. `http://www.repeatmasker.org`.

[68] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.

[69] Jason E. Stajich, 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*, 12(10):1611–1618, 2002. doi:10.1101/gr.361602.

[70] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2006.

[71] Stijn Van Dongen. A cluster algorithm for graphs. Technical Report INS-R0010, National Research Institute for Mathematics and Computer Science in the Netherlands, 2000.

[72] Anthony Wirth. A *Plasmodium falciparum* genefinder. Honours thesis, The University of Melbourne, 1998.

[73] Von Bing Yap and Terry Speed. Estimating substitution matrices. In Rasmus Nielsen, editor, *Statistical Methods in Molecular Evolution*, Statistics for Biology and Health, chapter 15, pages 407–438. Springer, 2005.

[74] Ru-Fang Yeh, Lee P. Lim, and Christopher P. Burge. Computational inference of homologous gene structures in the human genome. *Genome Research*, 11(5):803–816, 2001. doi:10.1101/gr.175701.

$Id: astivala_thesis.tex,v 1.126 2006/10/25 07:35:48 astivala Exp $