Studies in Probabilistic Sequence Alignment and Evolution

Thesis by

Ian Holmes

In Partial Fulfilment of the Requirements
for the Degree of
Doctor of Philosophy

University of Cambridge

Queens' College

and

The Sanger Centre
Wellcome Trust Genome Campus
Hinxton Hall, Hinxton
Cambridge

(Submitted on December 18, 1998)

Acknowledgements

During the past three years I have been very fortunate in having had the opportunity to discuss computational biology and Bayesian methods with Alex Bateman, Ewan Birney, Bill Bruno, Richard Durbin, Sean Eddy, David Haussler, Des Higgins, Tim Hubbard, Anders Krogh, Chip Lawrence, David MacKay, Graeme Mitchison, Richard Mott, Kimmen Sjölander, Guy St.Clair Slater, Victor Solovyev, Erik Sonnhammer and many other workers at the Wellcome Trust Genome Campus in Hinxton.

I especially thank my supervisor, Richard Durbin, for his thoughtful encouragement, guidance and advice, for being a fair and consistent critic and for his tireless patience and support.

I am very grateful for the assistance I have received from the Sanger Centre. I would particularly like to thank Sharon Thornton and Sheila Skingley, the Informatics group secretaries during the course of my PhD, and the Systems Support team for their ever-rapid response.

I would also like to thank my proofreaders, whose comments were vital: Alex Bateman, Ewan Birney, Tony Cox, Richard Durbin and Guy Slater.

Thank you to the people who have supported me most during the last year: Andrew Holmes; Jennifer Holmes; and Claire Hibbitt.

This work was supported by a grant from the Medical Research Council.

Summary

The complete sequencing of whole genomes presents opportunities for detailed study of molecular evolution. This thesis combines theoretical developments of Bayesian approaches in bioinformatics with analysis of duplications in the recently completed *C.elegans* genome.

Developments in the Bayesian probabilistic framework for sequence analysis using hidden Markov models (HMMs) are described. The principal HMM algorithms are reviewed including alignment, training and model comparison. Theory is developed for prediction of alignment accuracy and tested using simulations. Software to provide accuracy measures for multiple alignments, based on the popular HMMER suite of profile-based alignment algorithms, is presented and evaluated with reference to the Pfam database of multiple alignments.

Several of these statistical techniques are applied to an analysis of genomic duplications in the *C.elegans* genome. The completion of this - the first animal genome - offers an opportunity to study the random duplication that are believed to be the first step in the evolution of a new gene. The construction of a database of non-coding duplications is described and measurements of molecular evolutionary parameters in *C.elegans* are calculated from the data and reported. A method of dating gene duplications using alignments between conserved introns is presented and compared to existing methods using Bayesian techniques developed earlier in the dissertation. Amongst the principal agents involved in creating genomic duplications are transposons; one of the simplest families of transposon is the Tc1-*mariner* family, of which two distinct active subfamilies are well-known in *C.elegans.* Using HMM profiles, six new subfamilies of *mariner*-like transposon have been identified in the *C.elegans* genome. Several of the new subfamilies display interesting homologies to one another, suggestive of common mechanisms of transpositional catalysis.

Finally, the software tools developed during this project are described and made available for public retrieval from the Sanger Centre web site.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Preamble

The role of bioinformatics is expanding from one of post-experimental data analysis to include data management and organisation. As a consequence of this, concepts of what is possible in bioinformatics are likely to influence the future design and planning of the grand experiments that will succeed the present genome projects. Researchers who want to access the results of these grand experiments will do so through layers of software that filter, organise and interpret the volumes of available biological data. If the trends apparent in the rest of the software world apply to computational biology then users will be merciless in their demand for tools that are intuitive, powerful, simple and interoperable. To meet this demand, computational biologists will need to be aware of the solid mathematical frameworks that can support apparently simple algorithms; they will need to complement the drive to develop sophisticated techniques with a sensitivity to the changing needs of the scientific community; and they will need to respond quickly and authoritatively to the inevitable developments in technology that are on the way. In short, bioinformatics - and related informatics such as chemoinformatics and clinical informatics - are ***at least*** as important as they have been hyped to be!

It may be suggested that this view of the future of bioinformatics is influenced by the role of sequence analysis with respect to the genome projects (see e.g. [CSC98, FAW$^+$95, WS98]). These projects are generating megabases of DNA sequence, coding for hundreds of thousands of genes whose structure and function - thanks to the infeasibility of total simulation of the quantum mechanics of peptide molecules [Sha97] - can often only be elucidated by detection of sequence-level homologies to previously characterised proteins. Nevertheless it seems unlikely that the trend of high-throughput data collection will be reversed in the near future, with new technologies like microarrays generating new types of data and new informatics challenges [DIB97, BJVU98]. If this trend does continue, then there are two aspects of bioinformatics research that can be predicted to survive: the development of self-consistent families of algorithms for finding patterns in data and the presentation of these algorithms to a wider audience of researchers in the form of succinct and intuitive database interfaces. These two activities are closely linked: while it is often true that the program with the better user interface wins, the use of a good interface on a bad algorithm, or a bad interface on a good algorithm, will always result in tension. It is much easier to put a positive spin on a technique that has a good, simple idea at its core than it is to make a complex collection of heuristics seem intuitive. Furthermore, good algorithms stimulate excitement and interest, which is exactly what is needed to motivate the fast-changing world of interface design. It should also be stressed that the meaning of "user interface" here refers to *all* the methods that are used to access large data sets, so that the distinctions between data and methods, interface and algorithm, data-structure and object-model become increasingly blurred. An example of the new class of database is the set of protein family databases (such as BLOCKS [HH91], PRINTS [AB94] and Pfam [SEB$^+$98]), which at one level are simply a clustering of the protein databases, but actually provide considerable added value in the form of annotation, links to other databases and to literature and - crucially - algorithms to make use of the contained information for protein family analyses.

This dissertation addresses some of these issues by example. It is divided into two sections. The first half describes some mathematical and technological developments in the new Bayesian approach to sequence analysis. There is some practical development of the ideas - in particular a software tool for analysing the accuracy of sequence alignments - although the section is essentially theoretical. The second half of the thesis describes construction of a database for the study of gene duplication events in the nematode *Caenorhabditis elegans*, whose genomic sequence was recently completed [CSC98]. This section is more applied than the first and the emphasis is more on the biology than the mathematics, though a number of algorithms and tools are developed that have wider applicability than the database construction problem described. It is hoped that the results of the first half inform the second half, and that this account demonstrates how sound mathematics can provide a solid foundation for the development of natural and intuitive tools and data sets.

The first half of this introductory chapter begins by outlining the context and history of biological sequence analysis, with particular reference to the use of Bayesian statistics. It proceeds to summarise the work described in the first half of the thesis. The second half gives a brief review of the study of the evolutionary models and mechanisms of gene duplication as a prelude to describing the work in the second half of the thesis. First, sequence analysis.

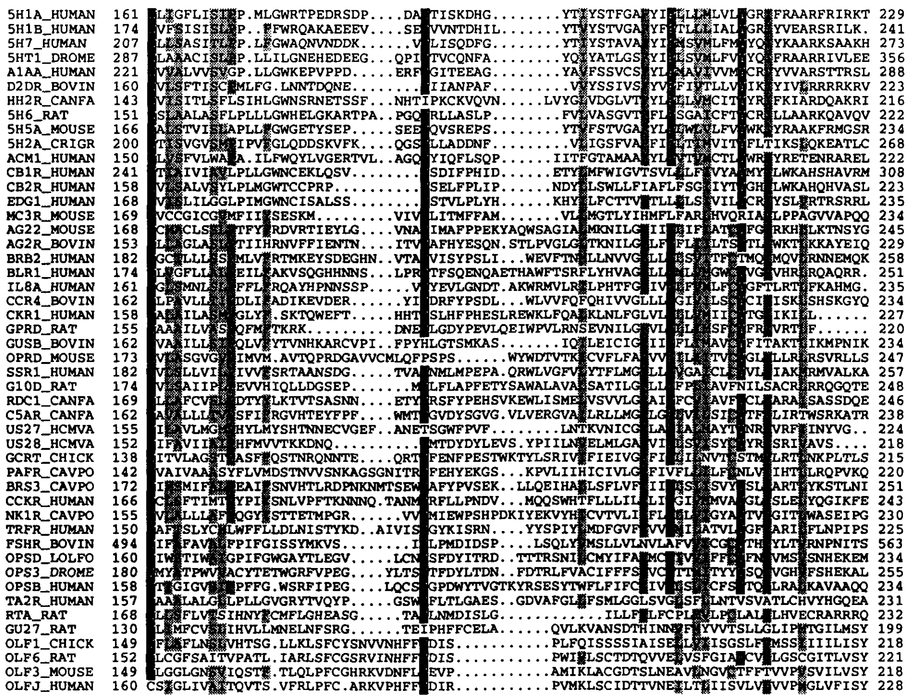
1.2 Sequence analysis

Sequence analysis of peptides can be seen as the biologist's practical response to the intractability of predicting a protein's higher-level structure and function from its sequence [Sha97]. Nature uses a limited number of structural motifs to construct its cellular machinery - over 40% of known protein sequence belongs to under 1500 families or "domains" [Cho92, SEB$^+$98] - and these structural homologies often correspond to sequence-level homologies, representing as they do an evolutionary connection by means of the gradual accumulation of mutations [Kim83]. The idea of protein sequence analysis is to search for these homologies and to exploit them in order to make inferences about the structure and function of novel proteins based on experimentally determined properties of well-characterised proteins [DEKM98]. The mutations that accumulate during evolution typically include residue substitutions and small insertions and deletions; there are thus a number of ways in which two sequences could be related and the exact nature of a homology between a set of sequences is usually represented in diagrammatic form as an "alignment". Part of a multiple alignment of protein sequences from the rhodopsin-like G-protein-coupled receptor family [AF94] is shown in Figure 1.1: sequences are laid out in rows with gap characters inserted so that homologous residues are aligned in columns. The time-intensive task of working out where to put the gap characters in order to get the best alignment is ripe for automation; algorithms that do this task are called *sequence alignment algorithms.* Parallel problems are encountered in DNA sequence analysis; the precise mechanisms by which nature recognises which parts of the DNA of a cell are genes, and which parts are regulatory sequences controlling the expression or splicing of the genes, are either unknown or too difficult to model completely. However, statistical comparisons with well-characterised sequences can answer some of these questions and are regularly used to locate genes in newly sequenced DNA [Hau98].

In recent years, the theory of sequence analysis has benefited considerably from the discovery [KBM$^+$94] that many of the alignment algorithms it had been using - which had been classified under the broad umbrella of "dynamic programming" - could be related to a mathematical framework that had been used very successfully in other fields, notably speech recognition [Rab89]. The framework is that of hidden Markov models (HMMs). The probabilistic nature of the HMM formulation provides strong links to the field of Bayesian statistics, a powerful revision of statistical ideas that has enthusiastic support in the machine learning community where HMMs were primarily in use. A brief discussion of the context of Bayesian methods and of pre-HMM sequence analysis may illustrate the happy significance of their combination.

1.2.1 Bayesian methods

At the most fundamental level it can be difficult to pin down the difference between the Bayesian approach to statistics and the classical or "frequentist" approach against which the Bayesians set themselves in opposition. All statistics essentially involves postulating probabilistic models and seeing how well these models fit observed data. Perhaps the definitive mark of Bayesian analysis is the emphasis on the application of Bayes' rule to likelihoods and priors



```
5H1A_HUMAN   161 █L█GFLIS██P.MLGWRTPEDRSDP....DA█TISKDHG........YT█YSTFGA█YI█LL█MLVL█GR█FRAARFRIRKT 229
5H1B_HUMAN   174 █VF█ISIS██P..█FWRQAKAEEEV....SE█VVNTDHIL.......YT█YSTVGA█YF█TLL█IAL█GR█YVEARSRILK. 241
5H7_HUMAN    207 █LL█ASIT██P.L█GWAQNVNDDK......V█LISQDFG........YT█YSTAVA█YI█MSV█LFM█YQ█YKAARKSAAKH 273
5HT1_DROME   287 █LA█ACIS██P.LLILGNEHEDEEG...QPI█TVCQNFA........YQ█YATLGS█YI█LSV█LFV█YQ█FRAARRIVLEE 356
A1AA_HUMAN   221 █VV█LVV█VGP.LLGWKEPVPPD.....ERF█GITEEAG........YA█FSSVCS█YL█MAV█VVM█CR█YVVARSTTRSL 288
D2DR_BOVIN   160 █VL█FTIS█ MLFG.LNNTDQNE........█IIANPAF.........V█YSSIVS█YV█FI█TLLV█IK█YIV█RRRRKRV 223
HH2R_CANFA   143 █VI█ITL█FLSIHLGWNSRNETSSF...NHTIPKCKVQVN.....LVYG█VDGLVT█YL█LLV█CIT█YR█FKIARDQAKRI 216
5H6_RAT      151 █SL█ALA█FLPLLLGWHELGKARTPA..PGQ█RLLASLP........FV█VASGVT█FL█SGA█CFT█CR█LLAARKQAVQV 222
5H5A_MOUSE   166 █AL█TVI██APLL█GWGETYSEP.....SEE█QVSREPS........YT█FSTVGA█YL█WL█LFV█WK█YRAAKFRMGSR 234
5H2A_CRIGR   200 █TI█VGV██IPV█GLQDDSKVFK....QGS█LLADDNF.........V█IGSFVA█FI█LT█MVIT█FLTIKS█QKEATLC 268
ACM1_HUMAN   150 █LV█FVLWA█A.ILFWQYLVGERTVL..AGQ█YIQFLSQP......IITFGTAMAA█YL█VT█CTL█WR█YRETENRAREL 222
CB1R_HUMAN   241 █T█IVI█VLPLLGWNCEKLQSV........█SDIFPHID......ETY█MFWIGVTSVL█LF█VYA█MY█LWKAHSHAVRM 308
CB2R_HUMAN   158 █VL█ALV█YLPLMGWTCCPRP.........█SELFPLIP......NDY█LSWLLFIAFLFSG█IYT█GH█LWKAHQHVASL 223
EDG1_HUMAN   168 █V█LILGGLPIMGWNCISALSS........█STVLPLYH......KHY█LFCTTV█TLL█LS█VIL█CR█YSL█RTRSRRL 235
MC3R_MOUSE   169 █VCCGICG█MFII█SESKM.........VIV█LITMFFAM......VL█MGTLYIHMFLFAR█HVQRIA█LPPAGVVAPQQ 234
AG22_MOUSE   168 █C█CLS██TFY█RDVRTIEYLG....VNA█IMAFPPEKYAQWSAGIA█MKNILG█II█IF█AT█FG█RKH█LKTNSYG 245
AG2R_BOVIN   153 █L█GLA██TIIHRNVFFIENTN....ITV█AFHYESQN.STLPVGLG█TKNILG█LF█FL█LTS█TL█WKT█KKAYEIQ 229
BRB2_HUMAN   182 █GC█LLL█S█MLV█RTMKEYSDEGHN..VTA█VISYPSLI...WEVFTN█LLNVVG█LL██S█TF█TMQ█MQV█RNNEMQK 258
BLR1_HUMAN   174 █L█GFLL██EIL█AKVSQGHHNNS....LPR█TFSQENQAETHAWFTSRFLYHVAG█LL█L█GW█VG█VHR█RQAQRR. 251
IL8A_HUMAN   161 █G█MNL██FFL█RQAYHPNNSSP.....V█YEVLGNDT.AKWRMVLR█LPHTFG█IV█F█LF█GFTLRT█FKAHMG. 235
CCR4_BOVIN   162 █LP█VLL██DLI█ADIKEVDER......YI█DRFYPSDL...WLVVFQFQHIVVGLLL█GI█LS█CI█ISK█SHSKGYQ 234
CKR1_HUMAN   158 █A█ILA██GLY█.SKTQWEFT.....HHT█SLHFPHESLREWKLFQA█KLNLFGLVL█L█II█TG█IKI█L...... 227
GPRD_RAT     155 █AA█ILV█S█QFM█TKRK..........DNE█LGDYPEVLQEIWPVLRNSEVNILG█VL█L█SF█FR█VRT█F...... 220
GUSB_BOVIN   162 █VA█ILL██QLV█YTVNHKARCVPI..FPYHLGTSMKAS.......IQ█LEICIG█II█FL█AV█FITAKT█IKMPNIK 234
OPRD_MOUSE   173 █V█SGVG██IMVM.AVTQPRDGAVVCMLQFPSPS......WYWDTVTK█CVFLFA█VV█L█TV█GL█LLR█RSVRLLS 247
SSR1_HUMAN   182 █V█LLVI██IVV█SRTAANSDG.....TVA█NMLMPEPA.QRWLVGFV█YTFLMG█LL█GA█CL█VL█IAK█RMVALKA 257
G10D_RAT     174 █V█AIIP██EVVHIQLLDGSEP.......M█LFLAPFETYSAWALAVA█SATILG█LL█FP█AVFNILSACR█RRQGQTE 248
RDC1_CANFA   169 █L█FCV███DTY█LKTVTSASNN....ETY█RSFYPEHSVKEWLISME█VSVVLG█AI█FC█AVF█CL█ARA█SASSDQE 246
C5AR_CANFA   162 █A█LLL███SFI█RGVHTEYFPF....WMT█GVDYSGVG.VLVERGVA█LRLLMG█LG█V█SI█TF█LIRTWSRKATR 238
US27_HCMVA   155 █I█VLMG██HYLMYSHTNNECVGEF..ANETSGWFPVF........LNTKVNICG█LA█A█AYT█NR█VRF█INYVG.. 224
US28_HCMVA   152 █IF█VII██HFMVVTKKDNQ...........█MTDYDYLEVS.YPIILN█ELMLGA█VI█S█SY█YR█SRI█AVS.... 218
GCRT_CHICK   138 █ITVLAG█T█ASF█QSTNRQNNTE....QRT█FENFPESTWKTYLSRIV█FIEIVG█FI█I█NVT█STM█LRT█NKPLTLS 215
PAFR_CAVPO   142 █VAIVAA█SYFLVMDSTNVVSNKAGSGNITR█FEHYEKGS...KPVLIIHICIVLG█FIVFL█LF█NLV█IHT█LRQPVKQ 220
BRS3_CAVPO   172 █I█MIF███EAI█SNVHTLRDPNKNMTSEW█AFYPVSEK..LLQEIHA█LSFLVF█II█S█SVY█SL█ART█YKSTLNI 251
CCKR_HUMAN   166 █C█FTIMT█YPI█SNLVPFTKNNNQ.TANM█RFLLPNDV...MQQSWHTFLLLIL█LI█GI█MVA█GL█SLE█YQGIKFE 243
NK1R_CAVPO   155 █V█LLL█F█QGY█STTETMPGR......VV█MIEWPSHPDKIYEKVYH█CVTVLI█FL█L█GYA█TV█GIT█WASEIPG 230
TRFR_HUMAN   150 █AF█SLYC█LWFFLLDLNISTYKD..AIVIS█GYKISRN....YYSPIY█MDFGVF█VV█I█ATVL█GF█ARI█FLNPIPS 225
FSHR_BOVIN   494 █IF█FAV██FPIFGISSYMKVS........I█LPMDIDSP....LSQLY█MSLLVLNVLAFV█CG█TH█YLT█RNPNITS 563
OPSD_LOLFO   160 █IW█TIW██GPIFGWGAYTLEGV.....LCN█SFDYITRD..TTTRSNI█CMYIFA█MC█V█FF█FN█VMS█SNHEKEM 234
OPS3_DROME   180 █MY█TPWV█ACYTETWGRFVPEG....YLTS█TFDYLTDN..FDTRLFVACIFFFS█VC█TT█TYY█SQ█VGH█FSHEKAL 255
OPSB_HUMAN   158 █T█GIGV███PFFG.WSRFIPEG....LQCS█GPDWYTVGTKYRSESYTWFLFIFC█IV█S█CFS█TQ█LRA█KAVAAQQ 234
TA2R_HUMAN   157 █AA█LALG█LPLLGVGRYTVQYP.....GSW█FLTLGAES..GDVAFGL█FSMLGGLSVG█SF█LNTVSVATLCHVYHGQEA 231
RTA_RAT      168 █L█FLV█SIHNY█CMFLGHEASG.....TA█LNMDISLG...........ILLF█LFCPL█LP█LAL█LHVECRARRRQ 232
GU27_RAT     130 █L█MFCV██IHVLLMNELNFSRG......TEIPHFFCELA......QVLKVANSDTHINN█F█YVVTSLLGLIP█TGILMSY 199
OLF1_CHICK   149 █F█FLN██VHTSG.LLKLSFCYSNVVNHFF█DIS...........PLFQISSSSIAISE█L█ISGSLF█MSS█IIILISY 218
OLF6_RAT     152 █LCGFSAITVPATL.IARLSFCGSRVINHFF█DIS...........PWI█LSCTDTQVVE█VSFGIA█CV█LGSCGITLVSY 221
OLF3_MOUSE   149 █LGGLGN█VIQST█.TLQLPFCGHRKVDNFL█EVP...........AMIKLACGDTSLNEAV█NGV█TFFTVVP█SVILVSY 218
OLFJ_HUMAN   160 CS█GLIVA█TQVTS.VFRLPFC.ARKVPHFF█DIR...........PVMKLSCIDTTVNE█LT█IISVLV█VVP█GLVFISY 228
```

Figure 1.1: Part of a multiple alignment of protein sequences from the rhodopsin-like 7-transmembrane receptor family [AF94], displayed by the BELVU alignment browser [SD94]. Each row represents a sequence in the family. Gap characters (in this case, dots ".") are inserted into the sequences so that homologous residues are aligned in columns. In this figure, residues are shaded by column conservation.

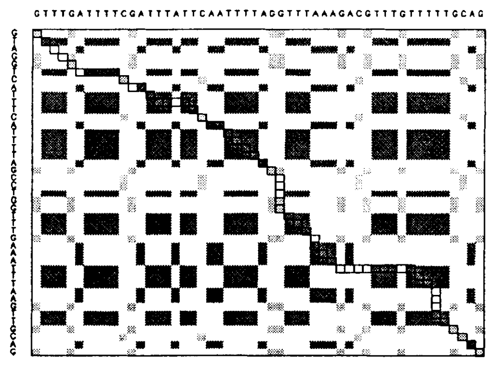
in order to obtain a *posterior* probability denoting a level of belief in a hypothesis [Mac92a]. In the author's personal experience, Bayesians tend to be convincingly more open to discussion of the nature of the models they use and of the fundamental ideas in probability that underpin their methods. Counterbalancing this refreshing openness there is often a lack of interest in heuristic approaches that smack of frequentism, even though such approaches are often later found to have good theoretical justification.

Many biologists find this skepticism unnecessarily pedantic. Nonetheless, it is from the Bayesian camp that the theory of hidden Markov models - one of the most exciting developments in bioinformatics in recent years - has emerged. To understand the significance of HMMs it is useful to put them in context, by delving a little deeper into the history of sequence analysis techniques. The following brief sketch draws on the accounts in [Wat95] and [DEKM98].

In 1970, Needleman and Wunsch published the first dynamic programming alignment algorithm for aligning pairs of sequences [NW70]. Their algorithm finds the highest-scoring path through a dynamic programming matrix by exploring out from the top-left corner (see Figure 1.2). A path through the matrix corresponds to an alignment of the sequences, which lie on the horizontal and vertical axes of the matrix. The Needleman-Wunsch paper was succeeded by a number of further algorithms for pairwise alignment, e.g. [SW81, Got82, WE87, AGM$^+$90]. A range of publically available programs implementing these algorithms are available, two of the most widely used being SSEARCH [PL88] and BLAST [AG96]. A considerable amount of research has been directed towards the problem of assigning statistical significance to the scores obtained by these methods, most notably by Karlin and Altschul [KA90, KA93, AG96].

In parallel with pairwise alignment, many algorithms for simultaneous or progressive alignment of multiple sequences were also developed [SK83, WP84, CL88, FD87, Tay87, BS87, HS89]. Without going into excessive detail, these algorithms attempt to find shortcuts to calculating the full multi-dimensional

Optimal path:



Sequence alignment:

```
GTTTGATTTTCGATTTATTCAATTTTAG----GTTT-AAAGACGTTTGTTT---TTGCAG
GTAGG----TC-AT---TTC-ATTTTAGCCTGGTTTGAAA--------TTTAAGTTGCAG
```

Figure 1.2: The dynamic programming matrix for Needleman-Wunsch alignment of two intron sequences from *C.elegans*.

analogue of the two-dimensional dynamic programming matrix in Figure 1.2 (a task too lengthy for practical analysis of large numbers of sequences). Typically, the algorithm estimates a "guide tree" and progressively aligns subgroups of sequences. Some programs implementing these algorithms are PILEUP [But98] and CLUSTALW [THG94a]. The latter makes use of sequence profiles during the progressive alignment. Sequence profiles are structures that model the position dependence of the gap and substitution propensities; see e.g. [GV96, THG94b].

In 1993 and 1994, Anders Krogh and co-workers from the Haussler group at UC Santa Cruz published several papers on the use of hidden Markov models (HMMs) for modelling protein and DNA sequences [HKMS93, Kro94, KBM$^+$94, KMH94]. Hidden Markov models were previously widely used in speech recognition [Rab89]. Krogh and co-workers realised they provided a formal probabilistic model with all the behaviour of sequence profiles [DEKM98]. The Santa Cruz group published numerous subsequent papers on the use of HMMs in sequence analysis (e.g. [BHK$^+$93, SKB$^+$96, HKMS93, KHRE96, REKH97]) and HMM ideas were also quickly latched onto and extended by workers at the MRC Laboratory of Molecular Biology in Cambridge, England [EMD95, MD95, Edd95, KM95] and other places [BC94, BH96, PBBC96]. The probabilistic view of sequence alignment had also been described by analogy with statistical mechanics [LAB$^+$93, Miy94].

Hidden Markov models did not directly solve the problem that was perceived by many to be of chief mathematical importance in sequence alignment: that of assigning statistical significance levels to the alignment scores. What HMMs did do was to form a solid connection between biological sequence alignment algorithms and Bayesian machine learning. The Bayesian framework yielded insight into how sequence profiles and other alignment "machines" should be "trained" on data; some of the immediate dividends (there were many) included the principled justification of pseudocounts for sparse data sets [SKB$^+$96], the discriminative training framework [EMD95] and the use of the Baum-Welch algorithm for finding the optimal scoring scheme [Kro94, HK96]. The probabilistic formulation has been extended to pairwise alignment [BH96, ZLL98] and it has also been widely used for gene prediction (see [Hau98] for a review) as well as more eclectic HMM architectures [BD97]. The connection to Bayesian machine learning research continues to generate interesting new ideas such as probability density networks [Mac96a, Pov98] and Fisher kernels [JH98].

With this context, Part I of this thesis can be summarised.

1.2.2 Summary of Part I of the thesis

Part I of this dissertation comprises three chapters exploring issues that have arisen during the development of the HMM theory of sequence analysis. Chapter 2 is in the nature of a mathematical introduction to the remainder of the dissertation; it reviews some elementary terms, definitions and algorithms that are central to HMM theory and introduces certain ideas that will be used later on. The algorithms described can be divided into two categories: those pertaining to alignments of sequences to Markov models, and those pertaining to parameterisation of the model. Many of these algorithms are already well-explored, but there is some new material. Some of the new material introduces the elements of alignment accuracy and Bayesian decision theory that will be used in the second and third chapters. The rest outlines some possible improvements to training and model comparison algorithms for HMMs. A statistical mechanical view of HMM score distributions is also outlined; this will also be useful for Chapter 3. The idea of the generalised HMM is touched upon, and finally some molecular evolution models are introduced.

Chapter 3 of the dissertation describes investigations into the issue of how accurate a sequence alignment algorithm is. In the HMM view, a dynamic programming alignment is essentially trying to reconstruct an evolutionary history from what can be seen as noisy data. The question addressed in this chapter is: to what extent does this reconstruction procedure give an accurate result when the assumed evolutionary model is the correct one? (Of course, in reality the model is not actually expected to be precisely correct - it is, after all, just a way of finding homologies between sequences - but this analysis gives us insight into what kind of errors the algorithm would make even in the best of circumstances.) The question is explored using computer simulations and a Bayesian technique for extracting very weak alignments from sequences is presented and evaluated.

Chapter 4 of the dissertation - the final chapter of Part I - describes a practical tool `postal` that was designed with the aim of applying some of the more useful results from Chapter 3 to the analysis of protein sequences using HMM profiles. Given a multiple protein sequence alignment, `postal` uses posterior probability techniques (described in Chapters 1 and 2) to identify which sequences (and which parts of those sequences) may be poorly aligned. While `postal` does not identify every misaligned sequence, it can pick up some obvious errors and flag low-information-content sections of alignments. It is thus suitable for use as a semi-automatic quality control tool for curators of large databases of gapped multiple alignments such as the Pfam database [SEB$^+$98]. `postal` is constructed using parts of the HMMER package [Edd96].

1.3 Gene duplications

Part II of the dissertation deals with the application of some of these techniques, along with other bioinformatics methods, to a specific problem of interest in molecular evolution: the study of gene duplications in the nematode *Caenhorabditis elegans.*

Why is it interesting to study gene duplications? The study of evolution is of central academic interest because of the attractive - if often elusive - idea of a driving principle underlying biology. Evolutionary frameworks can lend meaning and context to descriptions of biological mechanisms and hence organise knowledge. One such framework is the hypothesis of gene evolution via duplication and genetic redundancy, which suggests that new genes evolve by duplication of existing genes. When a gene is copied, the selective constraints on each copy are proposed to be relaxed, so that each copy is free to evolve (slightly) modified function [Ohn70, Oht89]. With international sequencing collaborations generating data for organisms' entire genomes rather than isolated genes or fragments of chromosomes, the prospect of analysing the long-timescale dynamics of genes in genomes is a realistic option. The completion of the genome sequence of the yeast *Saccharomyces cerevisiae*, long known to contain a number of gene duplications [Smi87, Ols91, Kab95], enabled the beautiful demonstration by Wolfe and Shields [WS97] that the chromosomal positioning and orientation of the duplicate genes were consistent with duplication on a large scale by the hypothesised mechanism of whole-genome (polyploid) duplication proposed by Ohno [Ohn70].

As the largest eukaryotic genome to be completely sequenced (and the first animal genome), the nematode *C.elegans* is a natural candidate for further systematic study of gene duplications [CSC98]. The progress of the nematode genome project has seen a rapid increase in the number of gene families to be characterised in this organism, with functions ranging through neuronal development [NKML98, BH97], chemosensation [Rob98, TCD+95], miscellaneous cell signalling and development [PS98, TW96, SDF+96, SLP+96, BR93] and other categories [AKW+98, DGPB98, LC95]. The resolution of standards and file formats in the annotation phases of the genome project and improvements in the technology of protein family databases have enabled automatic classification of many gene families [SEB+98, CSC98].

A large number of *C.elegans* gene families are found to comprise genes that are located close together on the chromosomes. These gene clusters are of particular interest since *C.elegans* was the first eukaryotic genome found to contain *operons* [BS97]. Operons in *C.elegans* consist of clusters of genes typically separated by around 100bp, that are transcribed together as a single strand of pre-mRNA and subsequently separated by *trans*-splicing. Co-transcription implies shared modes of transcriptional regulation. Many operons contain groups of genes that are not homologous and therefore cannot be said to have arisen from duplication; some of these non-homologous clusters are known to code for genes whose functions demand co-expression and, in these cases, a clear argument can be made for the evolutionary importance of the transcriptional co-regulation experienced by these genes [Blu98]. Operons were well-known in bacteria and archaea before their discovery in *C.elegans*; they have been found in other orders of the nematode phylum [EZM+97] and in flatworms [DH97].

1.3.1 Agents of change

There are a number of ways that duplications of sections of genomic DNA can occur in nature. Duplications in *C.elegans* can be induced in the laboratory by irradiation or by the introduction either of mutagenic chemical agents (formaldehyde, for example) or of mutator loci that up-regulate transposable element activity such as *mut*-2 [PvL97, JB97]. All these types of duplication can occur in the wild, with the additional possibility of random errors in replication and recombination. In some strains the principle cause of spontaneous duplications and other mutation appears to be transposable element activity [PvL97]. These elements are worth describing in more detail.

Transposable elements (or *transposons*) are well-defined (albeit fast-evolving) sequences that are found to spontaneously copy themselves (or have themselves copied) into genomic DNA [Jur98, PvL97, Smi96]. They are distinguishable from viruses - their closest relatives [GL95a, Jur98] - by not normally crossing cell boundaries invasively; they are not infectious. For this reason they are generally less destructive to their hosts and more restrained in their rate of proliferation. Transposons may be primarily subdivided according to whether the intermediate genetic component in the transposition cycle is DNA or RNA. DNA-mediated transposition proceeds by a "cut-and-paste" mechanism, whereby transposase proteins excise the mobile sequence and re-integrate it at the target site by DNA cleavage [PvL97, HLL97]. While the cut-and-paste mechanism is not itself replicative (it merely moves the sequence around), repair of the double-strand breakage from where the sequence was excised can generate a copy of the transposable element from the homologous chromosome, which the repair machinery uses as a template [vLCP94]. Characteristics of DNA transposons include flanking inverted repeat sequences and site-specific integration, both consequences of the transposition mechanism [HLL97]. RNA-mediated transposition, on the other hand, involves first transcription by RNA polymerase, then reverse transcription back to DNA by reverse transcriptase [Jur98]. Transposons may be further subdivided by whether the genes coding for the transposase proteins that catalyse transpositon are contained within the transposon sequence itself (in which case the transposon is said to be *autonomous*) or elsewhere (in which case the transposon is a *non-autonomous*

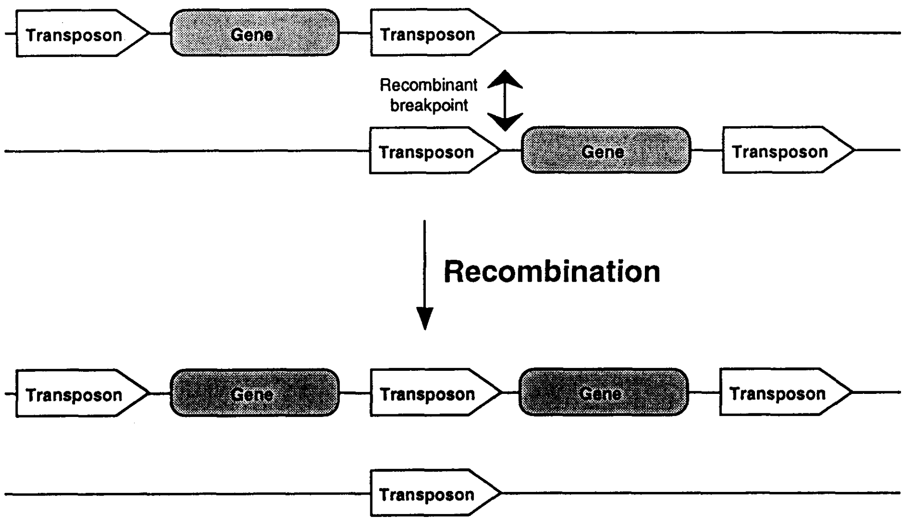


Figure 1.3: Transposons can precipitate gene duplication during recombination if pairing between adjacent copies leads to unequal crossing-over.

"hitch-hiker") [Smi96]. All of the above kinds of transposon are found in the sequenced Bristol N2 strain of *C.elegans*. The completion of the genomic sequence has led to a rise in the number and variety of transposons and other repetitive sequences reported in *C.elegans* [OGB95, OGB96, RvLDP97].

Transposons can promote gene duplication during meiotic recombination if homologous pairing between adjacent copies of a repetitive element leads to unequal crossing-over as in Figure 1.3 [NCC+92, YWB97, FBT+91]. (A similar mechanism underlies gene conversion, the phenomenon of expansion or contraction of existing clusters of homologous genes [LG91, Rob98].) Double-strand breakage repair following transposon excision can lead to local duplications [MKW91] and it is also possible that transposons may carry flanking sequence with them when they transpose [GL95b]. Transposon insertions into all kinds of host-important sequences - including introns, exons and promoter regions -

have been observed [Wes89, OGB95, BHP89]. Indeed the potential for transposons to have quite drastic effects on the evolution of their host organism is considerable [HLNL97] and the mechanisms by which their transpositional activity is regulated so as not to over-burden their hosts' replicative machinery have been topics of some interesting research [LC97]. One of the best-studied of transposon families is the relatively simple Tc1-*mariner* group, of which several variants are known in *C.elegans* [PvL97]. The canonical Tc1-*mariner* transposon in *C.elegans*, Tc1, is a DNA-based transposon consisting of an inverted repeat flanking a two-exon gene that codes for a single transposase protein. The protein catalyses the entire "cut-and-paste" transposition process [VBP96]. The ecology of *mariner*-like transposons is better understood than most; there are a number of ways in which these elements can interact to reduce transpositional activity [HLL97], including burdening of active transposase by non-autonomous and autonomous-but-defective transposons ("transposase titration" and "subunit poisoning" respectively) and poorly-understood interactions between active transposase proteins ("overproduction inhibition").

The aspects of genome duplication described above have been observed in the laboratory or in isolation, but an overview of their relative importances to the dynamics of gene duplication is lacking. With the completion of the *C.elegans* DNA sequence, the timing is ideal for a genome-wide systematic study of gene duplication in a model animal.

1.3.2 Summary of Part II of the thesis

Part II of this thesis dissertation describes the construction of a database, named Wormdup, to facilitate the study of genome duplications in *C.elegans*.

Chapter 5 (the first chapter of Part II) describes the construction and preliminary analysis of the main portion of the Wormdup database. The purpose of Wormdup is to provide a resource for answering questions about the sizes, frequencies, locations, causes and other aspects of genomic duplications. The chapter describes how the data in Wormdup are used to calculate various molecular evolutionary parameters for *C.elegans* such as the transition/transversion ratio, the rate of small indels and the rate and size distribution of fixation of non-coding and coding duplications. The fixation rates for non-coding and coding duplications are compared and the results are discussed with reference to the reliability of molecular clocks in general and the selective pressures that may act on duplicated DNA.

Chapter 6 (the second chapter of Part II) looks at a new method of dating gene duplications. Many of the interesting questions in molecular evolution rely on dates and times of speciation and divergence events. To answer these questions, it is useful to have an accurate "molecular clock" that can be used, for example, to compare rates of duplication for coding and non-coding DNA as in the previous chapter. The most commonly used kind of clock counts the number of nucleotide substitutions that have occurred at synonymous codon positions in the sequences and uses this to estimate a maximum-likelihood divergence time. This kind of clock is called a "codon clock". In this chapter a new kind of "intron clock" that counts substitutions and indels within conserved introns is proposed and evaluated using Bayesian techniques described in the first part of the thesis. The results suggest that intron evolution can be fitted to time-dependent models but that the clocks, as proposed, do not synchronise well with codon clocks. Possible reasons for this and potential improvements to the model are discussed.

Chapter 7, the final chapter of Part II and of the dissertation, looks at a specific class of DNA-based transposon - the Tc1-*mariner* group - that has been well-studied in nematodes. Using hidden Markov model and other dynamic programming techniques, statistics are obtained for the representation of previously characterised *mariner* families in the sequenced strain of *C.elegans*, as well as for six previously uncharacterised families. The new families are analysed using protein sequence analysis and phylogenetic techniques and found to be more closely related to one another than the previously characterised families. These results are discussed in the context of transposon ecology and evolution.

The dissertation ends with an appendix which describes the software tools that were developed specifically for this project (but with re-useability in mind). It is hoped that these may prove useful to other projects.

1.4 Statement of originality

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration.

Part I

Studies in Probabilistic Sequence Alignment

Chapter 2

Bayesian Methods for Hidden Markov Models in Biological Sequence Analysis

2.1 Introduction

A wide variety of score-based dynamic programming algorithms are commonly used for sequence alignment [AG96, PL88, BS87, LAK89, THG94a]. As early as 1992, Anders Krogh pointed out that the dynamic programming methods being used could be viewed as special cases of the Viterbi algorithm, widely used in speech recognition. The premise of this algorithm is that the sequences were generated by a probabilistic Markov model and that the exact state path is hidden from view, but can be reconstructed by inference. The recursive algorithm for performing this inference is an example of dynamic programming [Kro94].

Casting sequence alignment as an HMM problem does not avoid the question of what scores are significant. However, it does connect sequence alignment to a large published literature on HMM methods [DEKM98, Rab89, Mac96b]. This research puts the scores into context, explores how to choose the best scores for a particular problem, demonstrates how scores can be combined and opens up a wide range of new algorithms. Suddenly bioinformatics has a solid link to machine learning.

This chapter is a review of hidden Markov models in bioinformatics, of the main algorithms and techniques that can be used for HMMs and of certain properties they have. Sections 2.2 to 2.4 introduce notation and concepts that are used throughout the dissertation. Sections 2.5 to 2.7 are more speculative and less relevant to the rest of the dissertation.

2.2 Notation

In this section, hidden Markov models will be treated as machines that generate a single sequence, though it is only slightly more complicated to write down a definition of a "pair HMM" that generates a pair of sequences, and by extension, a "multiple HMM" that generates a whole set of sequences (this latter would be suitable for multiple alignment) [DEKM98].

An HMM has S states. The transition from state a to state b, labelled with residue X (with $X \in \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$ for DNA, for example), has probability t_{abX}. (It is conventional to talk of the transition "emitting" residue X and this convention will be used from now on.) The probabilities of all the transitions leaving a particular state must sum to 1. Two states have special names: the begin state $\mathcal{B}$ and the end state $\mathcal{E}$.

Denote the set of all the t_{abX} values by $\mathbf{t}$. This set $\mathbf{t}$ is often called the parameterisation of the HMM, or equivalently a point in the parameter space of the HMM.

Suppose that $\mathbf{X}$ is a sequence with L residues, whose i'th residue is X_i (the bold typeface $\mathbf{X}$ indicates the entire sequence and the light typeface X_i indicates an individual residue in the sequence). It is possible to trace a path of L steps through the HMM so that the i'th step uses a transition with residue label X_i. Such a path can also be called an *alignment* of the sequence to the HMM, because it aligns each residue in the sequence to a transition in the HMM. If a path begins in the begin state $\mathcal{B}$ and winds up in the end state $\mathcal{E}$, it will be called a *complete* path.

Call the alignment path $\mathbf{a}$, and suppose that at the i'th step the path is in state a_i (the path starts in state a_0). The i'th step in the path thus uses a transition from state a_{i-1} to state a_i and, to be consistent with the sequence $\mathbf{X}$, this transition must emit residue X_i. The corresponding transition probability is $t_{a_{i-1} a_i X_i}$. The joint likelihood of the sequence and the alignment is defined to be the product of all the transition probabilities along the path:

$$\Pr\left[\mathbf{a}, \mathbf{X} | \mathbf{t}\right] = \prod_{i=1}^{L} t_{a_{i-1} a_i X_i} \tag{2.1}$$

The likelihood of the sequence is the sum of the joint likelihoods of all complete paths of the same length as the sequence:

$$\Pr[\mathbf{X}|\mathbf{t}] = \sum_{\mathbf{a}:|\mathbf{a}|=L} \Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}] \tag{2.2}$$

The model is "hidden" because one typically knows the sequence **X** but not the alignment **a**. The main HMM algorithms address the problem of dealing with the missing information and these are reviewed below.

2.2.1 Other formulations of HMMs

Pair HMMs require a little more flexibility in that some transitions only emit residues for one of the two sequences. The most common type of pair HMM is the model for Needleman-Wunsch global alignment with affine gap penalties [NW70], which has three states (in addition to the start and end states). This model is shown in Figure 2.1. The three states include a match state and two indel states. Transitions into the match state emit paired residues in both sequences, whereas transitions into the indel states only emit residues in one or other of the two sequences. The indel states are often called "insert" and "delete" to distinguish each other. Transitions from the match to either of the indel states corresponds to opening a gap, so their probabilities are associated with the gap-opening penalty; looping transitions within the indel states correspond to gap-extension penalties. The probability distribution for paired match emissions corresponds to the substitution matrix. There is a more detailed discussion of the Needleman-Wunsch model in Chapter 3.

Alignments of pairs of sequences to pair HMMs specify residue→transition mappings for *both* the sequences. They therefore also specify which pairs of residues in the two sequences are aligned together. This is the commonly understood definition of sequence alignment.

A variant of the Needleman-Wunsch HMM used for local alignment - corresponding to the Smith-Waterman algorithm [SW81] - is actually a generalised HMM. Generalised HMMs are discussed in more detail in Section 2.8.

The pair HMM software described in Appendix A implements the kinds of

HMM described above, together with a limited class of generalised HMMs (including the Smith-Waterman model and the Bayesian block aligner mentioned in Section 2.8 of this chapter).

2.3 Aligning sequences to HMMs

Usually the sequence $\mathbf{X}$ is known and the alignment $\mathbf{a}$ is "missing information". Two useful tricks are: (i) to find the most likely alignment $\mathbf{a}$; (ii) to find the sum of the likelihoods of all alignments $\mathbf{a}$ (the sequence likelihood $\Pr[\mathbf{X}|\mathbf{t}]$ defined in (2.2)). (i) is a classic "maximum likelihood" approach, whereas (ii) is necessary if Bayes' rule is to be applied.

These tasks are accomplished using the Viterbi and Forward algorithms, respectively. Both are dynamic programming algorithms.

2.3.1 Maximising the alignment likelihood: the Viterbi algorithm

The Viterbi algorithm finds the most likely alignment $\mathbf{a}$ consistent with an observed sequence $\mathbf{X}$ [Vit67]. It works by building up the sequence $\mathbf{X}$ one residue at a time, so that there is a series of subsequences starting with nothing at $i = 0$ and ending up with the full sequence when $i = L$. The i'th subsequence corresponds to the first i residues of $\mathbf{X}$.

The optimal path $\mathbf{a}$ is built up step-by-step at the same time as the sequence, but the "missing information" problem is addressed by keeping track of S different optimal paths (one for each state of the model) at each value of i. It is not necessary to keep track of any more paths than this because the Markov nature of the model means that best path of length i for some state b contains the best path of length $i - 1$ for some state a; and so on down to $i = 0$.

This can be expressed more formally. Let $\{\mathbf{a}_{i,b}\}$ be the set of all the paths of length i that start in the begin state $\mathcal{B}$ and end in state b. Let $V_{i,b}$ be the maximum likelihood of all these paths, i.e.

$$V_{i,b} = \max_{\mathbf{a} \in \{\mathbf{a}_{i,b}\}} \Pr[\mathbf{a}, \mathbf{X} | \mathbf{t}] \tag{2.3}$$

(The likelihood of the complete Viterbi path is then $V_{L,\mathcal{E}}$, where $\mathcal{E}$ is the end state and L is the sequence length.)

Let the penultimate state of $V_{i,b}$ be a. The first $i-1$ steps of the $V_{i,b}$ path must also be an optimal path for some state a, so:

$$V_{i,b} = \max_{a} [t_{abX_i} V_{i-1,a}] \tag{2.4}$$

Equation (2.4) defines a recursion for the maximal path likelihood. Together with the boundary condition:

$$V_{a,0} = \begin{cases} 1 & \text{if } a = \mathcal{B} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

which just means "start in the start state", this recursion is the Viterbi algorithm. The algorithm only calculates the likelihoods of the paths; the paths themselves can be reconstructed by traceback. The maximal likelihoods $V_{i,b}$ form an $L \times S$ array of "cells" called the "dynamic programming matrix".

For sequence alignment, the algorithm is usually expressed in terms of the logs of the likelihoods rather than the likelihoods themselves. This is both intuitively natural (since log-likelihoods are additive, which corresponds better to the idea of scores) and more computationally well-behaved (since it avoids underflow problems).

2.3.2 Summing alignment likelihoods: the Forward algorithm

The Viterbi algorithm finds the likelihood of the most likely path consistent with the observed sequence (and by traceback, the path itself). The Forward algorithm finds the sum of the likelihoods of all paths consistent with the observed sequence (as in (2.2)) and is obtained essentially by replacing the max in equations (2.3)-(2.5) with a sum.

$$V_{i,b} = \max_{\mathbf{a} \in \{\mathbf{a}_{i,b}\}} \Pr[\mathbf{a}, \mathbf{X} | \mathbf{t}] \tag{2.3}$$

(The likelihood of the complete Viterbi path is then $V_{L,\mathcal{E}}$, where $\mathcal{E}$ is the end state and L is the sequence length.)

Let the penultimate state of $V_{i,b}$ be a. The first $i-1$ steps of the $V_{i,b}$ path must also be an optimal path for some state a, so:

$$V_{i,b} = \max_a [t_{abX_i} V_{i-1,a}] \tag{2.4}$$

Equation (2.4) defines a recursion for the maximal path likelihood. Together with the boundary condition:

$$V_{a,0} = \begin{cases} 1 & \text{if } a = \mathcal{B} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

which just means "start in the start state", this recursion is the Viterbi algorithm. The algorithm only calculates the likelihoods of the paths; the paths themselves can be reconstructed by traceback. The maximal likelihoods $V_{i,b}$ form an $L \times S$ array of "cells" called the "dynamic programming matrix".

For sequence alignment, the algorithm is usually expressed in terms of the logs of the likelihoods rather than the likelihoods themselves. This is both intuitively natural (since log-likelihoods are additive, which corresponds better to the idea of scores) and more computationally well-behaved (since it avoids underflow problems).

2.3.2 Summing alignment likelihoods: the Forward algorithm

The Viterbi algorithm finds the likelihood of the most likely path consistent with the observed sequence (and by traceback, the path itself). The Forward algorithm finds the sum of the likelihoods of all paths consistent with the observed sequence (as in (2.2)) and is obtained essentially by replacing the max in equations (2.3)-(2.5) with a sum.

Define $F_{i,b}$ to be the sum of the likelihoods of all the paths of length i that end in state b, i.e.:

$$F_{i,b} = \sum_{\mathbf{a}\in\{\mathbf{a}_{i,b}\}} \Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}] \tag{2.6}$$

The $F_{i,b}$ form another $L \times S$ dynamic programming matrix. The Forward algorithm for calculating them is:

$$F_{i,b} = \sum_a t_{abX_i} F_{i-1,a} \tag{2.7}$$

$$F_{0,a} = \begin{cases} 1 & \text{if } a = \mathcal{B} \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

From the point of view of scores, the transition from equations (2.3)-(2.5) to equations (2.6)-(2.8) correspond to replacing the $z = \max(x, y)$ rule in the dynamic programming for choosing between two scores x and y with a modified rule $z = \max(x, y) + B(|x - y|)$, where $B = \log(1 + \exp -|x - y|)$ is a "bonus" function that rewards similar scores. When $x \simeq y$, then $B \sim \log 2 - \frac{|x-y|}{2}$, i.e. the similarity bonus directly penalises the difference in scores when the difference is small; but when $\max(x, y) \gg \min(x, y)$ then $B \sim \exp -|x - y|$, i.e. the similarity bonus decays rapidly when the difference in scores is large.

2.3.3 Posterior probabilities of alignments: the Forward-Backward algorithm

Given the joint likelihood $\Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}]$ and the sequence likelihood $\Pr[\mathbf{X}|\mathbf{t}]$ (the latter of which is calculated using the Forward algorithm), a posterior probability for the path can be calculated using Bayes' rule:

$$\Pr[\mathbf{a}|\mathbf{X}, \mathbf{t}] = \frac{\Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}]}{\Pr[\mathbf{X}|\mathbf{t}]}$$

The number of paths aligning a sequence of length L to a model of size S is $\sim S^L$. There are only $L \times S$ entries in the dynamic programming matrix, each representing the alignment of an individual residue to an individual state. It is usually sufficient to work with these rather than the entire path distribution.

Let the notation $(i \diamond b)$ mean "residue X_i is aligned to a transition that ends in state b". The posterior probability of $(i \diamond b)$ is defined as the sum of the posterior probabilities of all the paths $\mathbf{a}$ that include $(i \diamond b)$ (i.e. all the paths that align residue i to a transition that ends in b):

$$\begin{aligned}
\Pr\left[(i \diamond b)|\mathbf{X},\mathbf{t}\right] &= \sum_{\mathbf{a}:(i\diamond b)\in\mathbf{a}} \Pr\left[\mathbf{a}|\mathbf{X},\mathbf{t}\right] \\
&= \sum_{\mathbf{a}:(i\diamond b)\in\mathbf{a}} \frac{\Pr\left[\mathbf{a},\mathbf{X}|\mathbf{t}\right]}{\Pr\left[\mathbf{X}|\mathbf{t}\right]} \\
&= \sum_{\mathbf{a}_{i,b},\bar{\mathbf{a}}_{i,b}} \frac{\Pr\left[\mathbf{a}_{i,b},\mathbf{X}|\mathbf{t}\right]\Pr\left[\bar{\mathbf{a}}_{i,b},\mathbf{X}|\mathbf{t}\right]}{\Pr\left[\mathbf{X}|\mathbf{t}\right]} \\
&= \frac{\left(\sum_{\mathbf{a}_{i,b}} \Pr\left[\mathbf{a}_{i,b},\mathbf{X}|\mathbf{t}\right]\right)\left(\sum_{\bar{\mathbf{a}}_{i,b}} \Pr\left[\bar{\mathbf{a}}_{i,b},\mathbf{X}|\mathbf{t}\right]\right)}{\Pr\left[\mathbf{X}|\mathbf{t}\right]} \\
&= \frac{F_{i,b}B_{i,b}}{F_{L,\mathcal{E}}} \qquad (2.9)
\end{aligned}$$

where $\mathbf{a}_{i,b}$ is a path of length i ending in state b as before and $\bar{\mathbf{a}}_{i,b}$ is a path of length $L-i$ that *starts* in state b, continuing on to the end state $\mathcal{E}$. The $B_{i,b}$ are called the Backward sums; they are defined as the sums of the likelihoods of all the paths $\bar{\mathbf{a}}_{i,b}$ and may be computed by flipping equations (2.6)-(2.8) in the i-direction. The algorithm for calculating the posterior probabilities $\Pr\left[(i \diamond b)|\mathbf{X},\mathbf{t}\right]$ is called the Forward-Backward algorithm [DEKM98].

2.3.4 Comparing alignments

There are various ways to compare two alignments quantitatively. Perhaps the simplest metric is the *overlap*, which counts the number of residues that both alignments agree on as being aligned to the same state of the HMM [DEKM98]. A related method just counts residues aligned to a particular subset $\{\varsigma\}$ of states. This will be referred to as the *partial overlap*.

Many pair HMMs allow pairs of residues in the two sequences to be aligned to the same state. When counting the number of residue-to-state mappings that pair HMM alignments agree on, it is common to require that *both* residues in the pair are aligned to the same state, in both alignments. This is consistent with the view that such residues are aligned to each other, rather than to a common state.

The *fractional overlap* is just the overlap divided by the total number of residues in the sequence. A *partial fractional overlap* can also be defined, by only counting residues that are aligned to a subset $\{\varsigma\}$ of states, as before. For the partial fractional overlap it is no longer unambiguous what the total number of $\{\varsigma\}$-labellings should be; a choice must be made as to which alignment is the reference alignment. The partial fractional overlap for a pair HMM, counting only match states, is called the *fidelity* [HL96].

More sophisticated measures of alignment similarity include *edit distance* [SK83] and *shift score*, which is rather like a length-normalised edit distance [CK98].

The edit distance, the overlap and the partial overlap are all additive functions. A function $F(\mathbf{a}_\alpha, \mathbf{a}_\beta)$ between two alignments $\mathbf{a}_\alpha$ and $\mathbf{a}_\beta$ of the same sequence S is additive if, when the sequence is split into subsequences S_1 and S_2 (and the alignments split into $\mathbf{a}_{\alpha 1}$, $\mathbf{a}_{\alpha 2}$, $\mathbf{a}_{\beta 1}$ and $\mathbf{a}_{\beta 2}$), then the sum of the parts $F(\mathbf{a}_{\alpha 1}, \mathbf{a}_{\beta 1}) + F(\mathbf{a}_{\alpha 2}, \mathbf{a}_{\beta 2})$ is equal to the whole $F(\mathbf{a}_\alpha, \mathbf{a}_\beta)$. Additivity is a useful property for an alignment accuracy measure since it means the alignment that optimises the measure with respect to the posterior distribution can be found using a variant of the Viterbi algorithm. This kind of "optimal accuracy" algorithm is an application of Bayesian decision theory. An example of such an algorithm that uses the fidelity as an accuracy measure has been proposed [DEKM98] and is explored further in Chapter 2.

Alignment accuracy issues are dealt with in more depth in Chapter 3, in which simulation results for the accuracy of the Viterbi algorithm for a Needleman-

Wunsch pair HMM are given. It is shown that the accuracy of the algorithm can be predicted, both for the average case (using entropy methods) and for specific cases (using posterior probabilities). The performance of the "optimal accuracy" algorithm is also assessed in this chapter. A program to calculate posterior probability tables and implement the optimal accuracy algorithm for profile HMMs is presented in Chapter 4.

2.4 Hidden Markov models in molecular evolution

The most commonly asked questions in molecular evolution involve the relative or absolute dates of divergence of biological sequences. These questions are often answered by fitting time-dependent models to alignments between the sequences (see e.g. [DEKM98]). A natural extension is to take advantage of the power of HMM algorithms to sum over all alignments by allowing the transition probabilities t_{abX} of a pair HMM to be functions of a time parameter τ, and using optimisation algorithms to find the maximum-likelihood value of τ. This approach was proposed by Thorne *et al* [TKF91, TKF92].

There are two main things that a pair HMM of this kind has to get right: the substitution probabilities and the gap probabilities. These will be covered in turn.

2.4.1 Time-dependent substitution matrices

The use of time-dependent substitution matrices predates the use of HMMs to sum over all alignments. The basic idea is that the four nucleotides (or twenty amino acids) are states in a completely interconnected Markov chain; transitions between the states correspond to residue substitutions and the more time that goes by, the more chance there is of making a substitution. In fact the PAM matrices are an example of this kind of matrix: the PAM_τ matrix is just the PAM_1 matrix raised to the τ'th power [DSO78].

A generalisation allowing τ to take continuous values (rather than just discrete ones) proceeds as follows [KT75]: let $P_{XY}(\tau)$ be the (time-dependent) probability that residue X is found to be aligned to residue Y, so that the P_{XY} matrix at time zero is the $A \times A$ identity matrix $\mathbf{P}(0) = \mathbf{I}$ (where $A = 4$ for nucleotides and $A = 20$ for amino acids). Define the rate matrix $\mathbf{R}$ by $\frac{d}{d\tau}\mathbf{P} = \mathbf{R}\mathbf{P}$. Suppose that the eigenvalues of $\mathbf{R}$ are $\{\lambda_X\}$ and that the associated right eigenvectors are $\{\mathbf{u_X}\}$; then the solution to the ordinary differential equation can be written $\mathbf{P} = \mathbf{U}\mathbf{\Lambda}(\tau)\mathbf{U}^{-1}$, where $\mathbf{\Lambda}(\tau)$ is the diagonal matrix $\Lambda_{XY} = \delta_{XY} \exp[\lambda_X \tau]$.

A general $A \times A$ rate matrix can have $A^2 - A$ free parameters, but usually a smaller parameter set is used. The simplest model would use one parameter (apart from the time τ), corresponding to the substitution rate - essentially a choice of scale for the time parameter. For nucleotide substitutions, this is called the Jukes-Cantor model [JC69]. The next simplest is Kimura's two-parameter model [Kim80], which allows different rates for "transitions" (substitutions within the purine (A,G) and pyrimidine (C,T) groups) and "transversions" (substitutions between those groups). These are observed to occur at different rates in nature, with transitions being more common.

Both the Kimura and Jukes-Cantor models assume a uniform background distribution over nucleotides. This is not the case in real organisms. For the work described in this dissertation, a modified model due to Hasegawa *et al* was used [HKY85]. This model allows for a non-uniform background nucleotide distribution. The equations for $P_{AX}(t)$ under the Hasegawa model are:

$$P_{AA} = f_A(1 + \frac{f_Y}{f_R}\exp[-s_2 t]) + \frac{f_G}{f_R}\exp[-(f_Y s_2 + f_R s_1)t]$$

$$P_{AG} = f_G(1 + \frac{f_Y}{f_R}\exp[-s_2 t]) - \frac{f_G}{f_R}\exp[-(f_Y s_2 + f_R s_1)t]$$

$$P_{AC} = f_C(1 - \exp[-s_2 t]$$

$$P_{AT} = f_T(1 - \exp[-s_2 t]$$

where f_X is the background frequency of nucleotide X, $f_Y = f_A + f_G$ and $f_R = f_C + f_T$ are (respectively) the purine and pyrimidine frequencies, s_1 is the transition rate and s_2 is the transversion rate. The other probabilities may be obtained by rotating the $\{A, C, G, T\}$.

None of the above models account for the correlations between neighbouring bases that are observed in nature. In this dissertation these effects are ignored, although they are certainly non-negligible in reality [Bul86].

2.4.2 Time-dependent gap probabilities

Thorne *et al* [TKF91, TKF92] proposed a birth-death process of fragment insertion and deletion that supplies transition probabilities for a six-state pair HMM (in the collapsed state space, where each state is allowed a probability distribution over the residues it emits) in terms of a birth and a death rate.

Although the birth-death model is simple, an even simpler model was used for the work described in this dissertation. The HMM used is depicted in Figure 2.1. It is essentially the model for Needleman-Wunsch global alignment with affine gaps; there are three (collapsed) states, one for matches and two for indels. Gaps occur with frequency p_G per residue per strand and their length l is geometrically distributed with parameter p_E: $\Pr[l = l'] = p_E^{l'}(1 - p_E)$ (so that the mean length $\langle l \rangle = (1 - p_E)^{-1}$). The dependence of the gap frequency p_G on the time parameter τ is $p_G = 1 - \exp[-g\tau]$, where g plays the rôle of a gap-open rate. The gap extension parameter p_E does not depend on τ. The substitution matrix in the match state is also time-dependent, as described in Section 2.4.1 of this chapter.

Although the HMM in Figure 2.1 appears asymmetric (there is a transition from *Insert* to *Delete*, but not from *Delete* to *Insert*) the gap length distributions for the two strands are identical and independent (in fact, it is the asymmetry

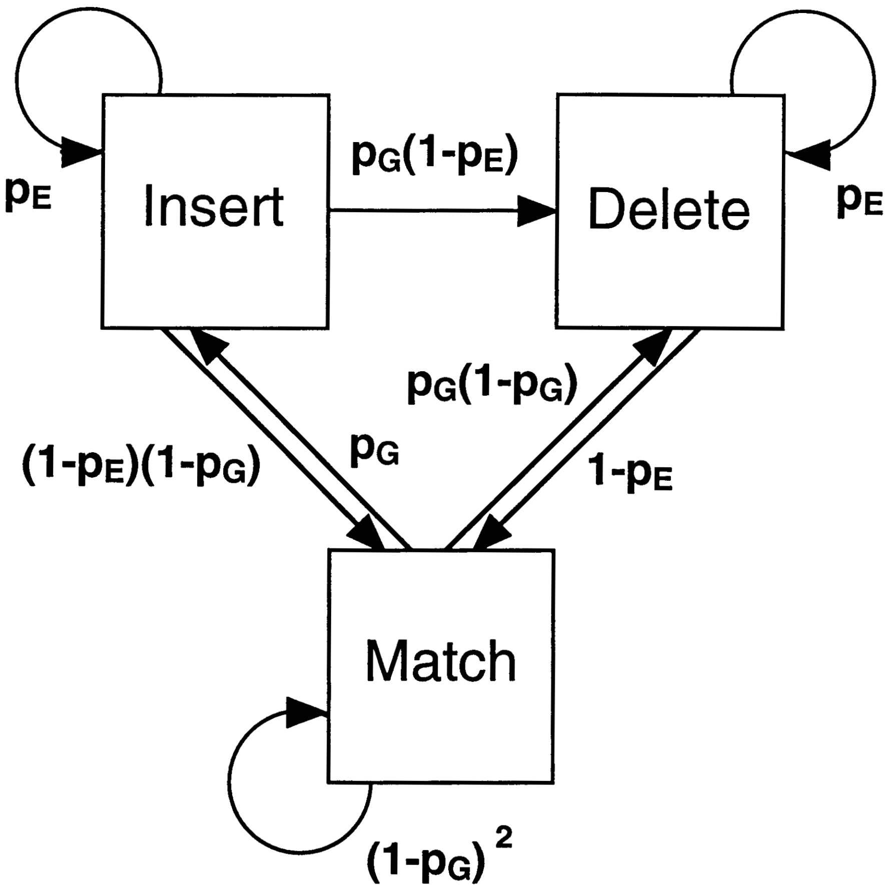


Figure 2.1: Hidden Markov model for Needleman-Wunsch global pairwise alignment with affine gaps. The start and end states are not shown. The gap penalty is determined by the gap frequency p_G (per residue per strand) and the gap extension probability p_E. The mean length of a gap is $(1 - p_E)^{-1}$.

of the model that ensures independence). For a fuller explanation of how the evolutionary model leads to the transition probabilities shown in Figure 2.1, see Chapter 3.

2.5 Likelihood derivatives and Fisher scores

There is a considerable amount of information in the posterior distribution that gets thrown away when an alignment is chosen, even if an optimal accuracy algorithm is used. This section looks at some of the ways that this information can be usefully digested.

Potentially the most useful result is that the derivatives of the sequence likelihood $\Pr[\mathbf{X}|\mathbf{t}]$ with respect to the parameters $\mathbf{t} = \{t_{abX}\}$ can be computed from the information in the Forward-Backward matrix. To see this, first write the path likelihood (2.1) as:

$$\Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}] = \prod_{a,b,X} t_{abX}^{n_{abX}} \tag{2.10}$$

where n_{abX} is the number of times that the alignment uses the transition t_{abX}. A more formal definition of n_{abX} is:

$$n_{abX}(\mathbf{a}) = \sum_{\substack{i\,:\,a_{i-1} = a \\ \text{and} \quad a_i = b \\ \text{and} \quad X_i = X}} 1 \tag{2.11}$$

which just says "to find n_{abX}, count the number of times that the i'th step of the path is from state a to state b and the i'th residue of the sequence is X".

The derivatives of the sequence likelihood $\Pr[\mathbf{X}|\mathbf{t}]$ are then given by:

$$\begin{aligned} \frac{\partial}{\partial t_{abX}} \Pr[\mathbf{X}|\mathbf{t}] &= \frac{\partial}{\partial t_{abX}} \sum_{\mathbf{a}} \Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}] \\ &= \sum_{\mathbf{a}} n_{abX} \frac{\Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}]}{t_{abX}} \end{aligned}$$

$$\begin{aligned}
&= \sum_{\mathbf{a}} \sum_{\substack{i\,:\,a_{i-1}=a \\ \text{and } a_i=b \\ \text{and } X_i=X}} \frac{\Pr[\mathbf{a},\mathbf{X}|\mathbf{t}]}{t_{abX}} \\
&= \sum_{i} \sum_{\substack{\mathbf{a}\,:\,a_{i-1}=a \\ \text{and } a_i=b \\ \text{and } X_i=X}} \frac{\Pr[\mathbf{a},\mathbf{X}|\mathbf{t}]}{t_{abX}} \\
&= \sum_{i} \sum_{\mathbf{a}_{i-1,a}} \sum_{\bar{\mathbf{a}}_{i,b}} \Pr[\mathbf{a}_{i-1,a},\mathbf{X}|\mathbf{t}] \Pr[\bar{\mathbf{a}}_{i,b},\mathbf{X}|\mathbf{t}] \\
&= \sum_{i} \left(\sum_{\mathbf{a}_{i-1,a}} \Pr[\mathbf{a}_{i-1,a},\mathbf{X}|\mathbf{t}] \right) \left(\sum_{\bar{\mathbf{a}}_{i,b}} \Pr[\bar{\mathbf{a}}_{i,b},\mathbf{X}|\mathbf{t}] \right) \\
&= \sum_{i} F_{i-1,a} B_{i,b} \qquad (2.12)
\end{aligned}$$

i.e. the derivatives can be calculated directly from the Forward-Backward matrix.

The expectations $E[n_{abX}|\mathbf{X},\mathbf{t}]$ of the counts n_{abX} over the posterior path distribution may be related to the derivatives of the sequence likelihood. First, note that differentiating (2.10) with respect to t_{abX} gives:

$$\frac{\partial}{\partial t_{abX}} \Pr[\mathbf{a},\mathbf{X}|\mathbf{t}] = \frac{n_{abX}}{t_{abX}} \Pr[\mathbf{a},\mathbf{X}|\mathbf{t}] \qquad (2.13)$$

Therefore the posterior expectations of the n_{abX} are given by:

$$\begin{aligned}
E[n_{abX}|\mathbf{X},\mathbf{t}] &= \sum_{\mathbf{a}} n_{abX}(\mathbf{a}) \Pr[\mathbf{a}|\mathbf{X},\mathbf{t}] \\
&= \sum_{\mathbf{a}} \frac{n_{abX}(\mathbf{a}) \Pr[\mathbf{a},\mathbf{X}|\mathbf{t}]}{\Pr[\mathbf{X}|\mathbf{t}]} \\
&= \sum_{\mathbf{a}} \frac{t_{abX} \frac{\partial}{\partial t_{abX}} \Pr[\mathbf{a},\mathbf{X}|\mathbf{t}]}{\Pr[\mathbf{X}|\mathbf{t}]} \\
&= \frac{t_{abX} \frac{\partial}{\partial t_{abX}} \left(\sum_{\mathbf{a}} \Pr[\mathbf{a},\mathbf{X}|\mathbf{t}] \right)}{\Pr[\mathbf{X}|\mathbf{t}]}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{t_{abX} \frac{\partial}{\partial t_{abX}} \Pr[\mathbf{X}|\mathbf{t}]}{\Pr[\mathbf{X}|\mathbf{t}]} \\
&= t_{abX} \frac{\partial}{\partial t_{abX}} \log \Pr[\mathbf{X}|\mathbf{t}] \qquad (2.14)
\end{aligned}
$$

The final term on the right - the derivative of the sequence log-likelihood - is called the "Fisher score" [JH98].

An interesting use of HMMs is as a pre-processing step to kernel-based methods such as Support Vector Machines [JH98, Bur98, Mac97]. Very crudely, this method feeds the Fisher scores into a perceptron which then attempts to discriminate between sequences from the family that the HMM was trained to model and other sequences. It seems that this is a more discriminative measure than simply looking at the likelihood - which agrees with intuition, in that there should be more information in the derivatives of the likelihood than in the raw Forward score. Another view of this "Fisher kernel" is that the derivatives give the perceptron an idea of the n_{abX} and hence of the most likely alignment. A theoretical justification of Fisher kernels is offered in [JH98].

2.6 Fitting parameters to HMMs

Two standard tasks in Bayesian analysis using generative models are *training* and (less commonly in biological sequence analysis - though the principle is good) *model comparison*. Both involve exploration of the parameter space $\{\mathbf{t}\}$ of the model. Analogous to the Viterbi and Forward algorithms, training involves finding the most likely parameterisation of the model, whereas model comparison involves integrating (or summing) over all possible values of the parameters.

The motivation for training is easy: one wants to find the best set of parameters for modelling sequences and thus recognising homologies. The motivation for integrating across the parameter space is perhaps less obvious; the basic idea is to put a fair penalty on models with more parameters (the principle of "Occam's Razor" [Mac92b]). Optimising over the parameter space would not necessarily be fair, even if a prior distribution over parameters were used, because models with more parameters also have more uncertainty in the maximum-likelihood value of those parameters. A fair way to deal with parameter spaces of differing dimensionality is to integrate over the lot. An example of a comparison between hypotheses with different numbers of parameters may be found in Chapter 6, where time-dependent models are fitted to divergent intron sequences.

A prior distribution over the parameter space $\Pr[\mathbf{t}]$ is required if different parameter values are to be compared. It will be assumed in this section that this prior is adequately modelled by a Dirichlet distribution with pseudocounts $\alpha = \{\alpha_{abX}\}$ corresponding to the transition probabilities $\mathbf{t} = \{t_{abX}\}$ (see e.g. [DEKM98]). This distribution will be written $\mathcal{D}(\mathbf{t}|\alpha)$.

For HMMs, both training and model comparison are trivial if the alignment $\mathbf{a}$ is known. In this case, the likelihood $\Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}]$ is just the multinomial distribution (2.10) whose coefficients n_{abX} may be obtained from the alignment as in equation (2.11). The posterior distribution is a Dirichlet with parameters $n_{abX} + \alpha_{abX}$. The t_{abX} that maximise this probability are given by:

$$t_{abX} = \frac{n_{abX} + \alpha_{abX}}{\sum_{b',X'} n_{ab'X'} + \alpha_{ab'X'}} \tag{2.15}$$

The integral of the likelihood over parameter space is the normalising factor for the Dirichlet posterior:

$$\Pr[\mathbf{X}] = \int \Pr[\mathbf{a}, \mathbf{X}|\mathbf{t}] \Pr[\mathbf{t}] dt = \frac{\prod_{a,b,X} \Gamma(\alpha_{abX} + n_{abX})}{\prod_a \Gamma(\sum_{b,X} \alpha_{abX} + n_{abX})} \tag{2.16}$$

where $\Gamma(x)$ is the gamma function. The extension of equations (2.15) and (2.16) to the case where the alignment $\mathbf{a}$ is unknown will be tackled below; essentially, the Dirichlet posterior becomes a mixture of $\sim N^L$ Dirichlets and the maximisation/integration is most easily handled approximately.

2.6.1 Maximising the likelihood in parameter space: training

When the alignment is unknown, the counts n_{abX} must be treated as missing data. A powerful algorithm for maximising the likelihood of a model with missing data is the expectation-maximization (EM) algorithm; the application of this algorithm to HMMs is called the Baum-Welch algorithm.

The basic idea of Baum-Welch is to calculate expected values $E[n_{abX}|\mathbf{X}, \mathbf{t}]$ for the counts n_{abX} given a particular value of the parameters $\mathbf{t}$ using equation (2.14) (the *expectation* step), then to maximize the sequence likelihood with respect to the parameters by plugging these expected counts back into equation (2.15) to give new values for the parameters (the *maximization* step).

It can be shown that both steps of this procedure increase the sum of the sequence log-likelihood and the Kullback-Leibler divergence between successive posterior distributions for the n_{abX} [NH93]. This sum is analogous to a variational free energy in statistical mechanics.

A problem with EM is that it can get stuck in local maxima of the likelihood. Modifications of Baum-Welch that attempt to address this problem include noise injection during the estimation of the n_{abX} [KBM$^+$94, HK96] (or the related technique of sampling the t_{abX} from the tempered Dirichlet posterior rather than taking the mean), Gibbs sampling [LAB$^+$93] and simulated Viterbi annealing [Edd95].

If the t_{abX} are not independent variables but can be expressed parametrically in terms of independent variables (such as a time parameter), then equations (2.14) and (2.15) will no longer apply. In this case it should still be possible to find the derivatives of the sequence likelihood with respect to the independent variables, by applying the chain rule to (2.12). Standard gradient-ascent algorithms can then be applied; for examples of such algorithms see [Bis95].

2.6.2 Integrating the likelihood over parameter space: model comparison

If the t_{abX} are independent variables, one way to estimate the integral $\Pr[\mathbf{X}] = \int \Pr[\mathbf{X}|\mathbf{t}]\mathcal{D}(\mathbf{t}|\alpha)dt$ is by importance sampling [Nea98]. Roughly speaking, this works as follows: sample points in parameter space $\mathbf{t}$ are generated from the Dirichlet prior distribution $\mathcal{D}(\mathbf{t}|\alpha)$, as described in [DEKM98]; the likelihoods $\Pr[\mathbf{X}|\mathbf{t}]$ are calculated for each point; these likelihoods are then averaged to estimate $\Pr[\mathbf{X}]$:

$$\Pr[\mathbf{X}] = \lim_{N\to\infty} \frac{1}{N}\sum_{i}^{N} \Pr[\mathbf{X}|\mathbf{t}_i] \quad \text{where } \Pr[\mathbf{t}_i = \mathbf{t}] \equiv \mathcal{D}(\mathbf{t}|\alpha) \tag{2.17}$$

The iteration can be stopped when, for example, the change in the estimate for $\Pr[\mathbf{X}]$ becomes sufficiently small.

Various modifications to this procedure might improve its efficiency. For example, it is possible to adapt the prior on-the-fly as more data points are seen; this is known as annealed importance sampling [Nea98]. An alternative approximate procedure is Markov chain Monte Carlo sampling [Nea96].

Another improvement to the importance sampling method would be to replace the $\Pr[\mathbf{X}|\mathbf{t}_i]$ term on the right-hand side of (2.17) with a more sophisticated estimate for $\Pr[\mathbf{X}]$. Speculatively speaking, it might be possible to derive this estimate by calculating expectation values for the counts n_{abX} using equation (2.14) and plugging these expectations into equation (2.16). By analogy with the Baum-Welch algorithm, this might be hoped to speed up convergence as it seems to take more account of the nature of the likelihood distribution.

If the t_{abX} are not independent, but instead are parametric functions of a set of independent variables, then importance sampling can proceed by sampling from a prior over this independent variable set. If there are just a few independent variables (such as a time parameter) it may be more convenient just to take a fixed set of sample points from a regularly spaced grid, though in general this can be expected to perform slightly worse than random sampling [Nea]. Notwithstanding, this is the method used in Chapter 6.

2.6.3 Incremental Baum-Welch and sparse envelopes

The view of the EM algorithm outlined in Section 2.6.1 of this chapter and presented in full in [NH93] suggests that, since the E and M steps may both be viewed as incremental maximizations of the same "variational free energy" function, even more incremental variants of the algorithm (where the variational free energy is improved, but not quite optimised, with respect to the probability distribution over the n_{abX} at each M-step) may speed up computation and hence convergence.

There are at least two ways this could be applied to the Baum-Welch algorithm for HMM training. One way would be if a set of K sequences $\{\mathbf{X}_k\}_{k=1}^{K}$ were being used to train the HMM, and the counts n_{abX} were obtained by summing the individual sequence counts, i.e. $n_{abX} = \sum_{k=1}^{K} n_{abX}(k)$; in this case, the individual sequence counts $n_{abX}(k)$ could be updated one at a time, and the t_{abX} re-estimated after the Forward-Backward algorithm was performed on each sequence, so that the dynamic programming M-step was only performed on each sequence every K'th E-step of the iteration. This corresponds to the incremental algorithm described in [NH93].

The second proposed optimisation to Baum-Welch can work on just one sequence. It can be applied not just to Baum-Welch but also to approximate numerical integration over the parameter space; it also works when the t_{abX} are not independent but are parametrically dependent on a reduced independent variable set. The optimisation corresponds loosely to the sparse algorithm described in [NH93]; here, it is called the "sparse envelopes" method and may be explained as follows.

The basic idea is that after the first run of the Forward-Backward algorithm, it should be obvious which alignments are the most probable, since these alignments will lie in the regions of the dynamic programming matrix where most of the probability distribution $\Pr[(i \diamond a)|\mathbf{X}, \mathbf{t}]$ is concentrated. Accordingly, cells that have extremely low probability can be "frozen" at their low-probability levels and not updated in subsequent runs. In practise it is often easier to freeze the likelihoods rather than the probabilities of these cells; alternatively, they can just be set to zero.

The set of cells that is chosen for inclusion in future updates is called the *envelope*. The choice of envelope can be managed as follows. For each residue i in the sequence, the posterior probabilities $\Pr[(i \diamond a)|\mathbf{X}, \mathbf{t}]$ (corresponding to a column in the dynamic programming matrix) must sum to unity. Choose some threshold $\epsilon \ll 1$ and find the lowest value of p such that the all the posterior probabilities $\Pr[(i \diamond a)|\mathbf{X}, \mathbf{t}]$ that are greater than p add up to more than $1 - \epsilon$, i.e.:

$$p(i) = \min\{p' : \left(\sum_{j:\Pr[(i \diamond a)|\mathbf{X},\mathbf{t}] \geq p'} \Pr[(i \diamond a)|\mathbf{X}, \mathbf{t}] \right) \geq 1 - \epsilon\}$$

The cells to be masked out for this value of i are those cells whose posterior probability is lower than $p(i)$. For pair HMMs, the dynamic programming matrix becomes a cube and the co-ordinates of a cell are $(i_1 \diamond i_2 \diamond a)$ rather than $(i \diamond a)$: each value of i_1 corresponds to a "slice" of the dynamic programming matrix since there are two parameters to sum over (i_2 and a) rather than just one (i). It may be convenient not to mask out quite all of the cells whose probability is lower than p', either to ensure that there is always a valid path through the matrix, or to avoid storing complicated masks (a convenient alternative for pair HMMs is just to keep track of an interval $i_2^{\min} - i_2^{\max}$ for each value of i_1, and mask out cells that fall outside this interval).

The approximation of the sparse envelopes method is rather similar to conditioning the integral over parameter space on the Viterbi alignment as in (2.16) in that, if the Viterbi alignment has probability greater than $1 - \epsilon$, the envelope will contain just the Viterbi path. However, sparse envelopes seem slightly more principled, as they allow a variable tradeoff between summation over suboptimal alignments and high computation time.

The incremental and the sparse variants of Baum-Welch are not guaranteed to converge on the same local minimum that the standard EM algorithm is guaranteed to find; indeed, the incremental algorithm can fail to converge at all, instead oscillating between results. However, there are plausible arguments that these algorithms may find the neighbourhood of the minimum more quickly. A sensible strategy might be to run several iterations of the approximate algorithm, then return to the standard EM algorithm for the last few iterations. This should combine the speed advantages of the approximate algorithms with the accuracy advantages of the exact algorithm.

2.7 Score and length distributions of an HMM

It may be useful to know the probability distribution of the scores of the paths that an HMM emits. For example, if the HMM is being used to search for instances of a sequence family, the score distribution can be used to estimate the probability that a true family member will score below the cutoff. An elementary result from the theory of Markov chains allows calculation of any number of moments of the score distribution.

Note that the following derivation assumes there is at most one transition between any pair of states. The HMMs considered so far have allowed multiple transitions between a pair of states, so long as they each emit a different residue. However, any HMM with multiple transitions can be converted to an HMM with single transitions by simply augmenting the state space, so no generality is lost in this assumption.

Suppose that the score of the transition from state a to state b is σ_{ab} (if the score is a straightforward log-likelihood, then $\sigma_{ab} = \log t_{ab}$ and the expected score will be the entropy of the model). Let $f_a(s)$ be the probability density function of the score s starting from state a. Then:

$$f_a(s) = \sum_b t_{ab} f_b(s - \sigma_{ab}) \tag{2.18}$$

Let $\phi_a(k)$ be the Fourier transform of s, i.e. $\phi_a(k) = E_a[\exp[\imath k s]]$ where $\imath = \sqrt{-1}$. The equivalent of equation (2.18) for $\phi_a(k)$ is:

$$\phi_a(k) = \sum_b t_{ab} \exp[\imath k \sigma_{ab}] \phi_b(k) \tag{2.19}$$

which is a matrix equation (although the entries in the matrix are functions of k).

Since $\phi_a(k)$ is the characteristic function of $f_a(s)$, the n'th moment of s can be evaluated by taking the value of the n'th derivative of $\phi_a(k)$ at $k = 0$. Differentiating equation (2.19) n times and setting $k = 0$ gives:

$$\begin{aligned} E_a[s^n] &= \frac{d^n \phi_a}{d(\imath k)^n}(0) \\ &= \sum_b t_{ab}(\sigma_{ab}^n + \frac{d^n \phi_b}{d(\imath k)^n}(0)) \\ &= (\sum_b t_{ab} E_j[s^n]) + (\sum_b t_{ab} \sigma_{ab}^n) \end{aligned} \tag{2.20}$$

This is a matrix equation which may be solved for any n by inverting the transition matrix t_{ab} and setting $E_{\mathcal{E}}[s^n] = 0$. It is thus possible to calculate any number of moments of the score distribution from any state of the model. Calculating the first two moments is sufficient to approximate the distribution with a Gaussian [KT75].

The result can be generalised to the case where the σ_{ab} are themselves random variables by replacing the n'th power of σ on the right-hand side of (2.20) with the n'th moment $\langle \sigma_{ab}^n \rangle$.

By setting all the σ_{ab} to 1 and identifying the score distribution from a state with the waiting time from that state to the end state, equation (2.20) can be used to derive constraints for modelling (sensibly shaped) sequence length distributions to arbitrary precision.

It was mentioned above that if $\sigma_{ab} = \log t_{ab}$, then the expected score of a path is the entropy of the HMM. More correctly, this score is the entropy of the joint distribution over paths and sequences $\Pr[\mathbf{a}, \mathbf{X}]$. The variance of the score is analogous to fluctuations in the statistical mechanical entropy. To find the entropy of the marginal distribution over sequences $\Pr[\mathbf{X}]$ is more difficult, but more relevant to the question of whether a sequence will score high enough to be observed, since the sequence likelihood $\Pr[\mathbf{X}]$ is the score returned by the Forward algorithm (and will be close to the Viterbi score for many cases of interest; see the comment at the end of Section 2.3.2 of this chapter). A similarly difficult problem is to find the relative entropy $D(\tilde{P}_1||\tilde{P}_2)$ between sequence probability distributions $\tilde{P}_n \equiv \Pr[\mathbf{X}|M_n]$ generated by two alternative models M_1 and M_2 (for example, a null model and a model of a protein domain). Implicitly, when programs such as HMMER [Edd96] fit extreme-value or other distributions to the observed scores from a database search, they are heuristically estimating the fluctuative behaviour of the relative entropy. A very simple approximation towards calculating the relative entropy fluctuations is described in Chapter 3. A more sophisticated approach has been developed by Hwa and Lässig [HL96], who apply renormalisation group techniques from the theory of critical phenomena in statistical physics to find the scaling behaviour of various properties of the Viterbi path.

2.8 Generalised HMMs

A useful generalisation of the HMMs described here is to relax the idea that each transition t_{abX} emits a single residue X and allow each transition to have a probability distribution over the length and content of the sequence it emits. This is called a "generalised HMM" after [KHRE96].

The simplest example of this kind of system keeps the basic structure of the HMMs described above, but instead of the length of the sequence emitted by each state being geometrically distributed like a Markov waiting time [KT75], a flat distribution for the length of sequence emitted by each state is used. The total emitted sequence length is conditioned on as a separate constraint. A prior can be put on the total sequence length, although it is usually an observed quantity and so the prior is only relevant during model comparison.

This kind of generalised HMM can be obtained from the HMMs described in Section 2.2 of this chapter as follows. Consider the simple two-state (*Loop*,*End*) model shown at the top of Figure 2.2. The model starts in the *Loop* state and at each step it either returns to the *Loop* state with probability $1 - \varepsilon$ or it moves on to the *End* state (and stops there) with probability ε. On every transition a residue is emitted. The probability that the model emits L residues is $\Pr[L] = (1-\varepsilon)^{L-1}\varepsilon$. Consider what happens as ε becomes small. The model will tend to stay in the *Loop* state for longer and longer times and the probability distribution of the emitted sequence length will get flatter and flatter. Formally, as $\varepsilon \to 0$, terms of $O(\varepsilon^2)$ become negligible and $\Pr[L] \to \varepsilon$.

So it is possible to design a very simple HMM where the probability of getting a sequence of a particular length L is almost independent of L, but the price one pays is that the likelihood of any individual alignment is very small - virtually zero, in fact. This is because L could be huge, and the probabilities of all possible values of L have to add up to one. However, given a sequence of a particular length L, Bayes' rule specifies how to work out the likelihood of an alignment $\mathbf{a}$ given the length L by *conditioning* on L: one simply has to divide the joint likelihood by the marginal length likelihood, i.e. $\Pr[\mathbf{a}|L] = \Pr[\mathbf{a}, L]/\Pr[L]$. Since $\Pr[L] \simeq \varepsilon$, this corresponds to cancelling out the final ε from the alignment likelihood. This can be woven seamlessly into the dynamic programming algorithm by pretending that the *Loop*$\to$*Loop* and *Loop*$\to$*End* transitions both have a "probability" of 1.

Now consider the three-state (*Loop*$_1$, *Loop*$_2$, *End*) model shown at the bottom of Figure 2.2. The same trick can be done to flatten the length distribution from each state by letting $\varepsilon \to 0$. However, the probability of getting a particular

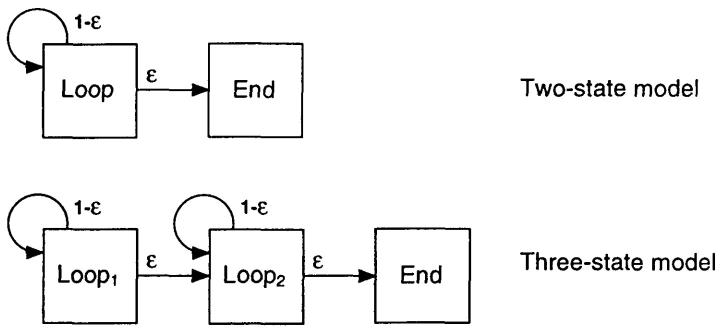


Figure 2.2: Looping models that tend towards flat length distributions as $\varepsilon \to 0$.

sequence length L is now $\Pr[L] = (L-1)(1-\varepsilon)^{L-1}\varepsilon^2$ and as $\varepsilon \to 0$ then $\Pr[L] \to (L-1)\varepsilon^2$. The extra factor of ε arises because there each path now has to make two low-probability transitions to reach the *End* state, rather than one. The extra factor of $L-1$ arises because there are $L-1$ different ways that a path can get to the *End* state in L steps, depending on when it chooses to move from *Loop*$_1$ to *Loop*$_2$.

In general, for a k-state model (*Loop*$_1$ ... *Loop*$_{k-1}$, *End*) there will be $k-2$ such choices and $\begin{pmatrix} L-1 \\ k-2 \end{pmatrix} = \frac{(L-1)!}{(k-2)!(L-k+2)!}$ ways of getting to the *End* state in L steps. The correct length-conditioned alignment likelihood $\Pr[\mathbf{a}|L]$ can be computed by doing dynamic programming with all the ε-transitions artificially set to 1 and dividing the result by $\begin{pmatrix} L-1 \\ k-2 \end{pmatrix}$. Once the conditional distribution $\Pr[\mathbf{a}|L]$ has been obtained, it can be multiplied by a more realistic prior distribution for L if this is desired.

The model underlying the Smith-Waterman algorithm with affine gaps is an example of a generalised HMM, since there is no prior length distribution on the flanking states which distinguish it from the Needleman-Wunsch model [SW81]. However, it is not a linear architecture like the HMMs in Figure 2.2 since the reciprocal transitions between the match and indel states form an internal cycle. Also, the affine gap costs put implicit priors on both the length and the number of gaps. A natural extension is to roll out the model, expanding the state

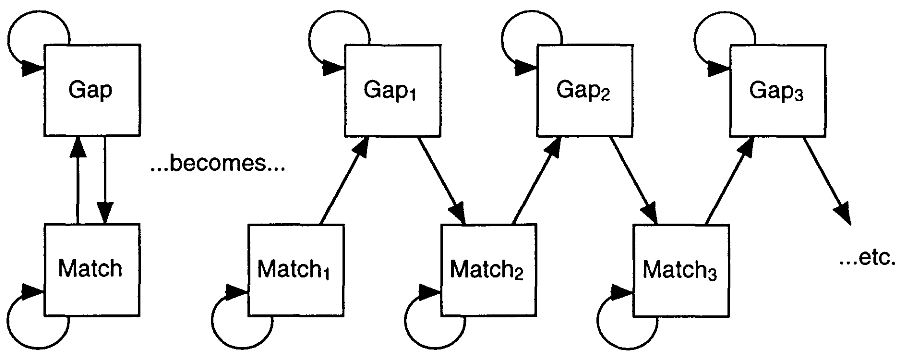


Figure 2.3: Unrolling the states of the Smith-Waterman model leads to the Bayes block aligner.

space up to a size proportional the maximum number of gaps as illustrated in Figure 2.3. Prior distributions can then be placed on the sequence lengths and the number of gaps. This is called the "Bayesian block aligner" [ZLL98].

The probability distribution of the sequences emitted by a state of a generalised HMM does not have to come from a length-conditioned HMM; more complex probabilistic models can be used, such as neural networks. A review of the use of generalised HMMs in gene-finding is provided in [Hau98].

Chapter 3

Dynamic Programming Alignment Accuracy

3.1 Introduction

Alignments of biological sequences generated by computational algorithms are routinely used as a basis for inference about sequences whose structure or function is unknown. The standard approach is to find the best-scoring alignment between a pair of sequences, where the the score rewards aligning similar residues, and penalises substitutions and gaps. The best-scoring alignment can be found by *dynamic programming* [NW70]. Other approaches that are frequently used, such as FASTA [LAK89] and BLAST [AG96], approximate this.

An important question for a biologist faced with the results of such a program is: How accurate is the proposed alignment? It is clearly desirable that an alignment algorithm return the most accurate alignment it can, but the notion of alignment accuracy implies the existence of a "correct" alignment, the definition of which is non-trivial. One approach is to construct a definitive structural alignment (based on crystallographic data and/or human judgement) which can then be compared with alignments returned by the algorithms in question. However, this is a difficult process to automate and it is not always clear what is really wanted biologically.

Another approach is to take a closer look at the inherent properties of the alignment algorithm itself. One can view the algorithm as a system for identifying the relationships between two sequences which have diverged due to random mutations (substitutions and indels) [TKF92]. By repeatedly simulating the experiment of randomly mutating a pair of initially identical sequences, then feeding the two sequences into the alignment algorithm, one can obtain a measure of the accuracy of the algorithm. In this paper the results of such empirical experiments are fist given. A theoretical estimate of the accuracy is then developed, and shown to provide a good approximation to the observed behaviour. A table from which accuracy values can be predicted for commonly used scoring systems is also given. Finally it is described how to calculate the expected accuracy of a given alignment, and how this can be used to construct an optimal accuracy alignment algorithm which performs demonstrably better than standard dynamic programming.

Other attempts to quantify and predict the accuracy of alignments have mainly been empirical and have focused on multiple alignments [MVF94], [Got96]. Mevissen and Vingron [MV96] have addressed pairwise alignment reliability recently, and Hwa, Lässig and Drasdo have developed theoretical approaches complementing those presented here [HL96, DHL97b].

3.2 Definitions and notation

This chapter will consider in detail the global alignment in which the entire length of the two input sequences must be aligned [NW70], although most of the results obtained will be equally applicable to the corresponding algorithms for local alignment [SW81].

3.2.1 Definition of the alignment fidelity

In this chapter, a pairwise alignment $\mathbf{a}$ between two sequences $(\mathbf{X},\mathbf{Y})$ is described by the set of aligned residues or *couplings* $(i \diamond j)$ between residue i of $\mathbf{X}$ and residue j of $\mathbf{Y}$.

Given a correct alignment $\mathbf{a}_{\text{real}}$, define the fidelity $F(\mathbf{a})$ of $\mathbf{a}$ as the fractional overlap between $\mathbf{a}$ and $\mathbf{a}_{\text{real}}$, i.e.:

$$F(\mathbf{a}) = \frac{|\mathbf{a} \cap \mathbf{a}_{\text{real}}|}{|\mathbf{a}_{\text{real}}|} \tag{3.1}$$

This corresponds to the partial overlap fraction metric defined in Chapter 2.

3.2.2 Choice of scoring parameters

Let us first treat the simplest biologically-relevant case: global alignment of two DNA sequences $(\mathbf{X},\mathbf{Y})$ with linear gap costs and a "flat" substitution matrix (one that doesn't differentiate between e.g. purine-purine and purine-pyrimidine substitutions). The score $S_{\mathbf{a}}$ for a particular alignment $\mathbf{a}$ is then:

$$S_{\mathbf{a}} = a\alpha + b\beta + c\gamma$$

where a, b and c are (respectively) the number of match, mismatch and gap columns in the alignment $\mathbf{a}$, and α, β and γ are match, mismatch and gap scores (typically but not necessarily with $\alpha > 0$, $\beta < 0, \gamma < 0$).

Although the score $S_{\mathbf{a}}$ depends on three free parameters (α, β and γ), the maximum scoring alignment $\mathbf{a}_{\mathrm{max}}$ only depends on one effective parameter. To see this, note first that global alignments must account for every residue in $\mathbf{X}$ and $\mathbf{Y}$, and so:

$$2a + 2b + c = L_{\mathbf{X}} + L_{\mathbf{Y}}$$

where $L_{\mathbf{X}}$ and $L_{\mathbf{Y}}$ are the lengths of $\mathbf{X}$ and $\mathbf{Y}$. Now consider the transformed score $S'_{\mathbf{a}}$:

$$S'_{\mathbf{a}} = \frac{S_{\mathbf{a}} - \frac{\alpha}{2}(L_{\mathbf{X}} + L_{\mathbf{Y}})}{\alpha - \beta} = -b - c\lambda$$

where

$$\lambda = \frac{\alpha/2 - \gamma}{\alpha - \beta} \tag{3.2}$$

Since $S'_{\mathbf{a}}$ differs from $S_{\mathbf{a}}$ only by an offset and a scaling factor, both of which are independent of the particular alignment $\mathbf{a}$, it follows that the ordering of the scores $S_{\mathbf{a}}$ of all possible alignments $\mathbf{a}$ (and hence the choice of maximally-scoring alignment $\mathbf{a}_{\mathrm{max}}$) is determined uniquely by λ.

The parameter λ can be considered to be an effective gap penalty. When $\lambda \gg \frac{1}{2}$, then $\beta \gg 2\gamma$ and the highest-scoring alignment will be minimally gapped as mismatches will be favoured over gaps. When $0 < \lambda < \frac{1}{2}$, then $\beta < 2\gamma < \alpha$ and gap regions will score higher than mismatches, with the consequence that all substitutions will be misidentified as pairs of indels. When $\lambda \leq 0$, then $2\gamma \geq \alpha$ (assuming $\alpha > \beta$) and gap regions will score higher than matches, which is clearly disastrous [VW94].

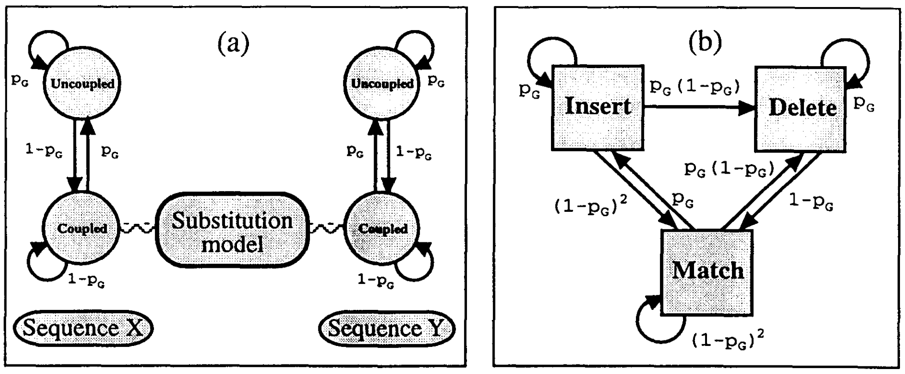


Figure 3.1: (a) Coupled Markov model of sequence evolution. Each sequence is represented by a semi-independent Markov chain, coupled by a point substitution model. (b) The corresponding finite state automaton for sequence alignment.

3.2.3 Probabilistic interpretation

Figure 3.1a shows a probabilistic model of sequence evolution that will be seen to correspond to the alignment algorithm described in Section 3.2.2. Each sequence is modelled by a hidden Markov chain with two states, labelled *coupled* and *uncoupled*. When both sides of the model are in the coupled state, aligned residues are emitted in pairs, one on each side. When either side is in the uncoupled state, unaligned residues are emitted singly on that side. Coupled emissions stem from a common ancestral residue; the joint probability distribution for the residue pair is derived from a point substitution model. Uncoupled emissions are unaligned and independent. Transitions from the coupled into the uncoupled state occur with probability p_G, as do self-looping transitions in the uncoupled state. (N.B. for affine gaps, the *coupled→uncoupled* transition still has probability p_G, but the self-looping *uncoupled→uncoupled* transition is assigned the independent gap-extension probability p_E.) The independence of the two Markov chains is restricted by the requirement that neither chain is allowed to enter the coupled state on its own (both must enter it simultaneously).

3.2.4 A simple point substitution model

For the experiments described below, the following simplified one-parameter model of nucleotide substitution was used. Start with identical residue pairs, one in each sequence, chosen at random from the set {A,C,T,G}. For each of the two residues, replace it with a randomly-chosen nucleotide with probability p_S. The replacement nucleotide has a one in four chance of being identical to the residue it is replacing. The probability q_{XY} of the residue pair (X, Y) being emitted in the coupled state is thus:

$$q_{XY} = \begin{cases} \frac{1}{16}(1 + 3(1 - p_S)^2) & \text{if } X = Y \\ \frac{1}{16}(1 - (1 - p_S)^2) & \text{if } X \neq Y \end{cases} \tag{3.3}$$

The probability q_X of the residue X being emitted in the uncoupled state is:

$$q_X = \frac{1}{4} \tag{3.4}$$

Note that if

$$p_S = 1 - e^{-2kt}$$

where k is a point substitution rate and t is a time-like parameter, this model is identical to that proposed by Jukes and Cantor [JC69].

3.2.5 Relationship between probabilistic model and alignment algorithm

Figure 3.1b depicts a stochastic finite-state machine for traversing the combined state space of the coupled Markov chains of Figure 3.1a. The *match* state of the automaton in Figure 3.1b emits coupled residue pairs in both sequences, whereas the *insert* and *delete* states emit uncoupled residues in **X** and **Y** respectively. Note the asymmetry of the *insert*→*delete* transition, which is required to preserve the independence of the gap length distributions in each sequence.

The automaton in Figure 3.1b is itself a hidden Markov model, albeit one which models two sequences rather than one. Alignment of sequences to hidden Markov models is performed using the Viterbi dynamic programming algorithm. To identify the most likely alignment **a** for a pair of sequences related under the simple indel model, one uses the Viterbi algorithm to align the sequences to the automaton in Figure 3.1b. This turns out to be mathematically equivalent to the standard (Needleman-Wunsch) alignment algorithm; that is, Needleman-Wunsch finds the most likely set of ancestral residue couplings under the probabilistic mutation model given a pair of sequences (**X**,**Y**).

Assuming the substitution model described in Section 3.2.4, and using the scoring notation of Section 3.2.2, it is found that the alignment score $S_{\mathbf{a}}$ is equal to the posterior log-likelihood of the sequence pair if the following match, mismatch and gap scores are chosen:

$$\alpha = \log \frac{(1-p_G)^2(1+3(1-p_S)^2)}{16} \tag{3.5}$$

$$\beta = \log \frac{(1-p_G)^2(1-(1-p_S)^2)}{16} \tag{3.6}$$

$$\gamma = \log \frac{p_G}{4} \tag{3.7}$$

If one is not interested in the exact score of the alignment obtained, but only in ensuring that its score is maximised, and if one restricts oneself to global alignments, then one need only specify a single scoring parameter such as the parameter λ defined in (3.2). Denote by $\hat{\lambda}$ the probabilistic value for λ, which is obtained by substituting equations (3.5)-(3.7) into equation (3.2):

$$\hat{\lambda} = \frac{\log\left[(\frac{1}{p_G}-1)\sqrt{1+3(1-p_S)^2}\right]}{\log\left[\frac{1+3(1-p_S)^2}{1-(1-p_S)^2}\right]} \tag{3.8}$$

Given that $\hat{\lambda}$ returns the alignment with the highest log-likelihood under the generative model, it is natural to predict that it is the optimal value of λ for reconstructing the correct alignment, in the sense that it maximises the fidelity $F(\mathbf{a}_{\mathsf{max}})$.

3.3 Results

3.3.1 Simulation 1: Optimisation of the alignment fidelity with respect to the scoring scheme

In order to test the prediction that $\hat{\lambda}$ is optimal, 50 pairwise alignments were randomly generated, each with 1000 aligned residue pairs plus gap regions, according to the evolutionary model of Section 3.2.3 with p_G and p_S set to a range of different values. The pairs of sequences thus generated were then independently re-aligned by the Needleman-Wunsch algorithm using a range of different values of λ, and the fidelities of the returned alignments were measured. With this procedure the value of λ that is optimal for reconstructing the alignment can be estimated and compared with the value $\hat{\lambda}$ predicted by equation (3.8).

3.3.2 Simulation 2: Measurement of the alignment fidelity

The sequence generation procedure of simulation 1 was performed at various different values of p_G and p_S and the sequences re-aligned using $\lambda = \hat{\lambda}$. The fidelity was measured and the process repeated until the mean re-alignment fidelity was known to within an error margin of ± 0.1 (this was a 95% confidence limit, assuming the fidelity of an alignment to be a Gaussian distributed random variable).

3.3.3 The probabilistic prediction $\hat{\lambda}$ is supported experimentally

Figure 3.2 shows values of $\hat{\lambda}$ for different values of p_G and p_S. Note that when $\hat{\lambda}$ drops below zero, effective reconstruction of the alignment is impossible, as gaps score higher than matches. This regime is indicated by the shaded region in Figure 3.2.

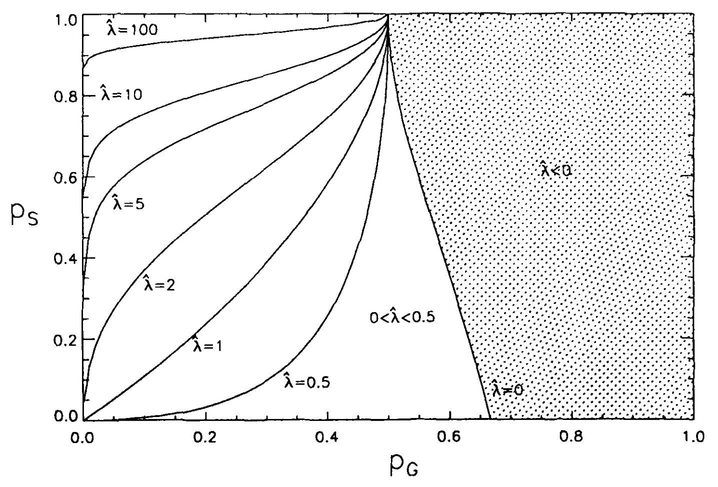


Figure 3.2: Contours of constant $\hat{\lambda}$ in mutation parameter space. $\hat{\lambda}$ is the effective gap penalty. The shaded region on the right-hand side of the plot represents $\hat{\lambda} < 0$, where pairs of indel events are more likely than matches and accurate alignment is effectively impossible.

Figure 3.3 shows how the fidelity F changes as a function of λ when $p_G = 0.1$ and $p_S = 0.2$. For $\lambda \leq 0$ the optimal alignment is all gaps and the fidelity is zero; for high λ the optimal alignment is minimally gapped and the fidelity flattens out, eventually reaching a plateau. In between these extremes there is a value of λ which maximises the fidelity.

By definition, setting $\lambda = \hat{\lambda}$ will find the most likely alignment, but there is no proof that this alignment will be the most faithful one. Figure 3.4 plots the observed optimal values of λ against the predicted values $\hat{\lambda}$. There is a good correspondence, supporting the hypothesis that the likelihood scoring approach is valid.

3.3.4 The fidelity decreases as p_G and p_S are increased

The graphs in Figure 3.5 show the dependence of the maximal fidelity F on the gap probability p_G and the substitution probability p_S. Figure 3.5a plots F as

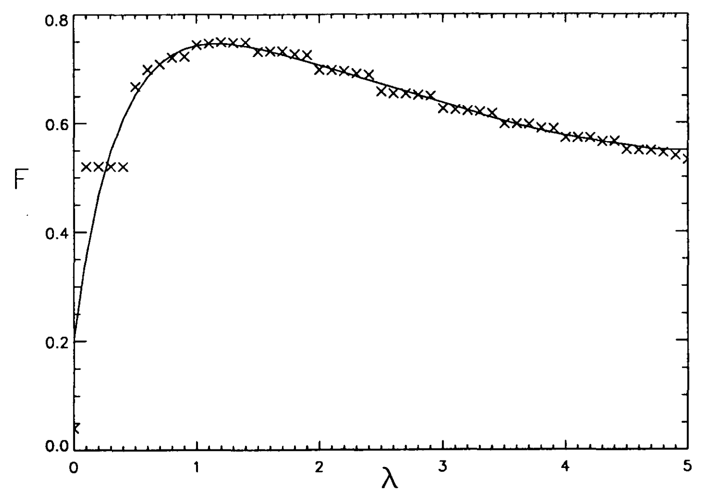


Figure 3.3: The fidelity F of alignments returned by dynamic programming for a range of values of the effective gap penalty λ, with p_G and p_S set to 0.1 and 0.2 respectively. When $\lambda \sim 0$, the optimal alignments are all gaps and $F \to 0$. As $\lambda \to \infty$, the optimal alignment tends to become minimally gapped, causing F to plateau. The data in this Figure are from simulation 1.

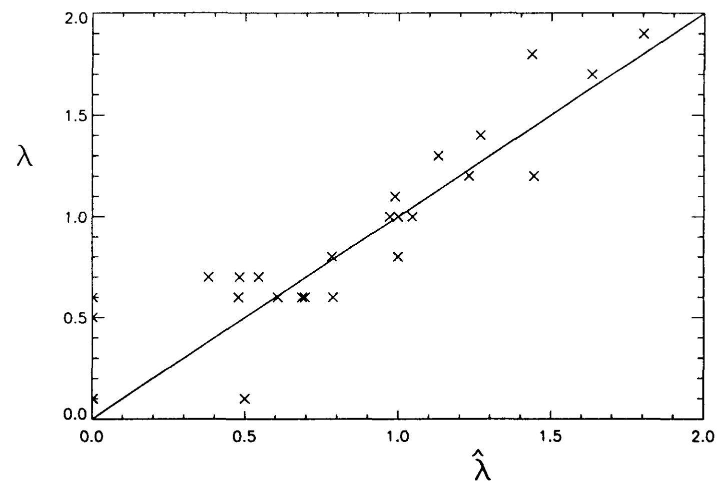


Figure 3.4: Values of λ which are observed from the simulation data to be optimal are compared with the values $\hat{\lambda}$ predicted by the likelihood scoring approach. There appears to be a strong correlation, with slope unity (solid line). The data in this Figure are from simulation 1.

a function of p_G at various different constant values of p_S and Figure 3.5b plots F against p_S at different constant values of p_G.

It can be seen that in general F decreases monotonically as the mutation parameters increase. The dependence of F on p_G and p_S is nearly linear up to around $(p_G, p_S) \sim (0.2, 0.2)$. Notable deviations from this behaviour are observable, for example at $(p_G, p_S) \simeq (0.2, 0.04)$ and again at $(p_G, p_S) \simeq (0.3, 0.1)$. At both these points the fidelity appears to be discontinuous. Referring back to Figure 3.2, it is seen that these points are on the locus $\lambda = 0.5$, which is recalled from Section 3.2.2 as the point at which mismatches become more likely than gaps. So the discontinuity can be identified with the scoring scheme entering a region of parameter space where substitution events are recognised.

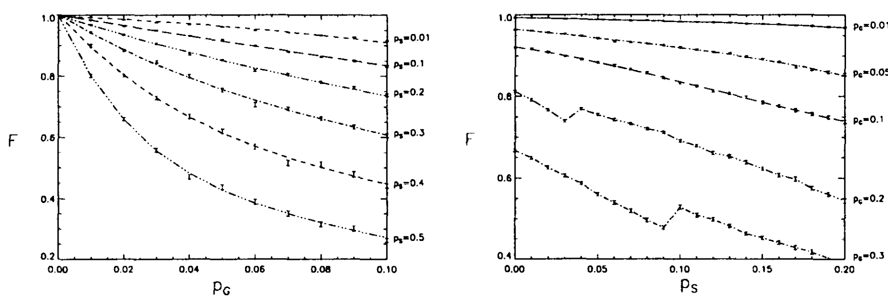


Figure 3.5: These graphs show the variation of the fidelity F (a) as a function of p_G at fixed p_S, and (b) as a function of p_S at fixed p_G. Note the discontinuities at $(p_G, p_S) \sim (0.2, 0.04)$ and $(0.3, 0.1)$, explained in the text. The data in this Figure are from simulation 2.

3.3.5 An analytic approximation to the alignment fidelity

Motivated by the near-linearity of the fidelity at low (p_G, p_S), an analytic approximation to the alignment fidelity can be developed.

To follow the analysis of the following section it is useful to be able to view an alignment geometrically, as a path through a dynamic programming matrix. The horizontal and vertical axes of the matrix represent the two aligned sequences $\mathbf{X}$ and $\mathbf{Y}$. A global alignment $\mathbf{a}$ is represented by a path from the top left to the bottom right of the matrix connecting all the coupled residue pairs $(x, y) \in \mathbf{a}$. Diagonal segments of the path correspond to match and mismatch regions and horizontal and vertical segments correspond to gaps. The fidelity of an alignment path $\mathbf{a}$ is its fractional overlap with the correct alignment path $\mathbf{a}_{\text{real}}$.

When the mutation probabilities are small, the Viterbi alignment path $\mathbf{a}_{\text{max}}$ returned by the dynamic programming algorithm is tightly bound to the correct path $\mathbf{a}_{\text{real}}$. The main source of errors is misplacement of gaps by the algorithm, as illustrated in Figure 3.6. This effect is called *edge wander.* The fidelity in this regime is governed by the average displacement distance of each gap (the mean edge wander) and by the frequency of gaps. The next section describes

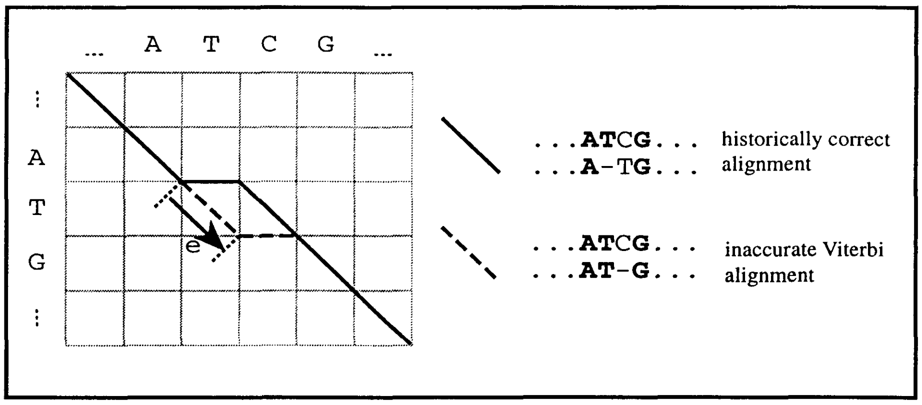


Figure 3.6: Edge wander - minor deviation of the Viterbi alignment path from the correct path - is the principal source of error in alignments between closely related sequences. In this toy example, the historically correct alignment (solid line) contains a mismatch next to an indel, but the Viterbi algorithm inevitably misaligns the two T residues (dotted line). The Viterbi edge wander e is defined to be the number of residues by which the gap is misplaced (here $e = 1$).

how to calculate the mean edge wander.

3.3.6 Calculation of the edge wander

Let the *edge wander* e be the displacement, in residues, of a gap in some near-perfect alignment $\mathbf{a}$ compared with the same gap in the correct alignment. Let $S(e)$ be the score of that segment of $\mathbf{a}$ which extends E residues to the left and right of the correct location of the gap, where E is some integer such that $(1/p_G) \gg E \gg e$. If v_k and w_k are the individual scores of the k'th residue pairings along adjacent diagonals ($\mathbf{v}$ and $\mathbf{w}$) of the dynamic programming matrix, with $k = 0$ at the correct location of the gap (so that, in the notation of Section 3.2.1, v_k corresponds to residue pairing $(i + k \diamond j + k)$ and w_k to residue pairing $(i + k + 1 \diamond j + k)$, where i and j are such that the correct gap location sits between residue pairings $(i \diamond j)$ and $(i + 2 \diamond j + 1)$), then one can write:

$$
\begin{aligned}
S(e) &= \sum_{k=-E+1}^{e} v_k + \gamma + \sum_{k=e+1}^{E} w_k \\
&= \sum_{k=-E+1}^{E} w_k + \gamma + \sum_{k=-E+1}^{e} (v_k - w_k) \\
&= S_W + \gamma + \mathcal{R}(e)
\end{aligned}
$$

where γ is the gap score, $S_W = \sum_{k=-E+1}^{E} w_k$ is the score along diagonal $\mathbf{w}$, and $\mathcal{R}(e) = \sum_{k=-E+1}^{e}(v_k - w_k)$ is the difference in score between alignment $\mathbf{a}$ and diagonal $\mathbf{w}$, minus the gap penalty γ.

Note that since the v_k and w_k are independent random variables, $\mathcal{R}(e)$ is a Markov process. (Strictly, the series $(v_k, v_{k+1}, ...)$ is not independent of the series $(w_k, w_{k+1}, ...)$, since v_k and w_k represent residue pairings in the same row of the dynamic programming matrix. However, $\mathcal{R}(e)$ is still Markov.)

The v_k and w_k are not identically distributed for all k, since the correct path crosses over from $\mathbf{v}$ to $\mathbf{w}$ between $k = 0$ and $k = 1$. For convenience rewrite v_k and w_k in terms of the scores t_k and s_k of residue pairings on and off the correct path, respectively:

$$
v_k = \begin{cases} t_k & \text{if } k \leq 0 \\ s_k & \text{if } k > 0 \end{cases}
$$

$$
w_k = \begin{cases} s_k & \text{if } k \leq 0 \\ t_k & \text{if } k > 0 \end{cases}
$$

An expression for $\mathcal{R}(e)$ can now be written in terms of $r_k \equiv s_k - t_k$:

$$
\mathcal{R}(e) = \begin{cases} \sum_{k=-E+1}^{e}(-r_k) & \text{if } e \leq 0 \\ \sum_{k=-E+1}^{0}(-r_k) + \sum_{k=1}^{e} r_k & \text{if } e > 0 \end{cases} \tag{3.9}
$$

The random behaviour of $\mathcal{R}(e)$ is illustrated in Figure 3.7. On average, $\mathcal{R}(e)$ will be zero at $e = 0$ and negative elsewhere; in any specific case, however, the maximum of $\mathcal{R}(e)$ may be some distance away from $e = 0$ and this is where the alignment algorithm will place the gap.

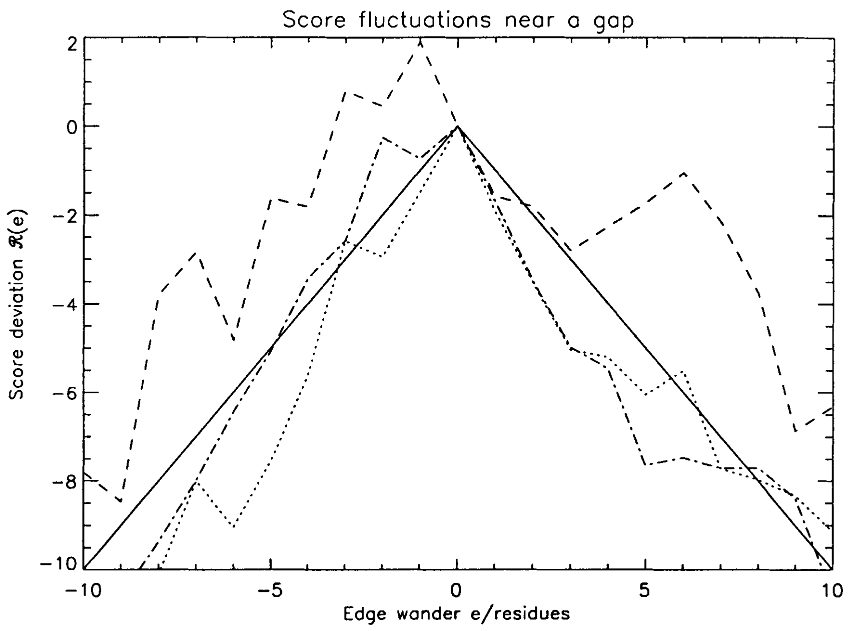


Figure 3.7: Variations in the alignment score when a gap is moved away from its correct position by sliding it along a diagonal. The solid line shows the mean behaviour: on average, the score will decrease as the gap is moved away from its correct position, so the score is maximal at $e = 0$. The dotted lines show examples of the behaviour in specific cases. Due to random fluctuations, the peak of $\mathcal{R}(e)$ may be somewhere away from $e = 0$. This means the optimal-scoring position for the alignment algorithm to place the gap will not be the correct position.

The joint probability distribution function (p.d.f.) $\varsigma(s,t)$ of s and t depends on the joint probability distribution q_{XY} of correlated residue pairs and the prior probability q_X of individual residues, defined in (3.3) and (3.4):

$$\varsigma(s,t) = \sum_{X,Y,Z} q_{XY} q_Z \delta(s - \log \frac{q_{XY}}{q_X q_Y}) \delta(t - \log \frac{q_{XZ}}{q_X q_Z})$$

where $\delta(x)$ is the Kronecker delta function:

$$\delta(x) \equiv \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases}$$

For convenience the scores are here written as log odds-ratios with respect to a "null" model whereby all residues are uncorrelated; this does not affect the final result.

The p.d.f. $\rho(r)$ of $r \equiv s - t$ is derived from $\varsigma(s,t)$:

$$\rho(r) = \sum_t \varsigma(r+t,t) = \sum_{X,Y,Z} q_{XY} q_Z \delta(r - \log \frac{q_{XY} q_Z}{q_{XZ} q_Y}) \qquad (3.10)$$

Now consider the Viterbi alignment $\mathbf{a}_{\mathbf{max}}$. Since this is the highest scoring alignment, the *Viterbi edge wander* $e_{\mathbf{max}}$ is given by:

$$e_{\mathbf{max}} = \operatorname*{argmax}_e \mathcal{S}(e) = \operatorname*{argmax}_e \mathcal{R}(e)$$

i.e. the edge wander is determined by the behaviour of $\mathcal{R}(e)$. If the peak of $\mathcal{R}(e)$ is ambiguous, so that there are two or more possible values for $\operatorname{argmax}_e \mathcal{R}(e)$, then $e_{\mathbf{max}}$ is defined to be the largest of those values.

Let $\mathcal{E}(e)$ be the p.d.f. of $e_{\mathbf{max}}$:

$$\mathcal{E}(e) = \Pr\left[e_{\mathbf{max}} = e\right]$$

Utilising the Markov property of $\mathcal{R}(e)$, factorise $\mathcal{E}(e)$ by splitting the process (3.9) into three parts, cutting at $k = 0$ and $k = e$ and summing over allowable values of the difference $y = \mathcal{R}(e) - \mathcal{R}(0)$:

$$\mathcal{E}(e) = \left\{ \begin{array}{ll} \sum_y \mathcal{C}_L(e+E,0).\mathcal{X}_R(-e,y,0).\mathcal{C}_R(E,-y) & \text{if } e \le 0 \\ \sum_y \mathcal{C}_L(E,-y).\mathcal{X}_L(e,y,0).\mathcal{C}_R(E-e,0) & \text{if } e > 0 \end{array} \right.$$

where $\mathcal{C}_L$, $\mathcal{C}_R$, $\mathcal{X}_L$ and $\mathcal{X}_R$ are bounding probabilities defined on sums of r_k ($\mathcal{C}$ signifies a cumulative distribution and $\mathcal{X}$ an exact distribution, and the L and R suffices mean "left of the peak" and "right of the peak"):

$$\mathcal{C}_L(x,z) = \Pr\left[\forall n \in \{1,2,...,x\} : \sum_{k=1}^{n} r_k \le z\right]$$

$$\mathcal{C}_R(x,z) = \Pr\left[\forall n \in \{1,2,...,x\} : \sum_{k=1}^{n} r_k < z\right]$$

$$\begin{aligned} \mathcal{X}_L(x,y,z) &= \Pr[\sum_{k=1}^{x}(-r_k) = y \quad \text{and} \\ &\qquad \forall n \in \{1,2,...,x\} : \sum_{k=1}^{n}(-r_k) \le z] \end{aligned}$$

$$\begin{aligned} \mathcal{X}_R(x,y,z) &= \Pr[\sum_{k=1}^{x}(-r_k) = y \quad \text{and} \\ &\qquad \forall n \in \{1,2,...,x\} : \sum_{k=1}^{n}(-r_k) < z] \end{aligned}$$

The $\mathcal{C}_L$, $\mathcal{C}_R$, $\mathcal{X}_L$ and $\mathcal{X}_R$ can be found by recursive decomposition, separating the first step from the $(x-1)$ succeeding ones:

$$\mathcal{C}_L(x,z) = \left\{ \begin{array}{ll} \sum_{r \le z} \rho(r)\mathcal{C}_L(x-1, z-r) & \text{for } x > 0 \\ 1 & \text{for } x = 0 \end{array} \right. \tag{3.11}$$

$$\mathcal{C}_R(x,z) = \left\{ \begin{array}{ll} \sum_{r < z} \rho(r)\mathcal{C}_R(x-1, z-r) & \text{for } x > 0 \\ 1 & \text{for } x = 0 \end{array} \right. \tag{3.12}$$

Score matrix	$\langle \lvert e \rvert \rangle$	p_G	$F_{g=12}$
PAM40	0.326	$2^{-g/2}$	0.99
PAM80	0.688	$2^{-g/2}$	0.98
PAM120	1.246	$2^{-g/2}$	0.96
PAM160	1.884	$2^{-g/2}$	0.94
PAM200	2.573	$2^{-g/3}$	0.68
PAM250	3.888	$2^{-g/3}$	0.53
BLOSUM100	0.794	$2^{-g/3}$	0.90
BLOSUM75	1.332	$2^{-g/2}$	0.96
BLOSUM62	1.826	$2^{-g/2}$	0.94
BLOSUM50	3.286	$2^{-g/3}$	0.60
BLOSUM45	3.671	$2^{-g/3}$	0.56
BLOSUM30	~24	$2^{-g/5}$	-

Table 3.1: Edge wander for various common amino acid substitution matrices.

$$\mathcal{X}_L(x,y,z) = \begin{cases} \sum_{r\geq -z} \rho(r)\mathcal{X}_L(x-1,y+r,z+r) & \text{for } x>0 \\ \delta(y) & \text{for } x=0 \end{cases} \tag{3.13}$$

$$\mathcal{X}_R(x,y,z) = \begin{cases} \sum_{r> -z} \rho(r)\mathcal{X}_R(x-1,y+r,z+r) & \text{for } x>0 \\ \delta(y) & \text{for } x=0 \end{cases} \tag{3.14}$$

where $\delta(y)$ is the Kronecker delta again.

A program **edge** [1] has been written to calculate the mean absolute edge wander $\langle |e| \rangle$ for various common substitution matrices; the results are listed in Table 3.1. To find the expected fidelity given the mean edge wander, use the following formula:

$$F = 1 - \langle |e| \rangle \left(1 - (1-p_G)^2\right) \tag{3.15}$$

[1] C++ source code for the **edge** program is available at `http://www.sanger.ac.uk/Users/ihh/edge.html`

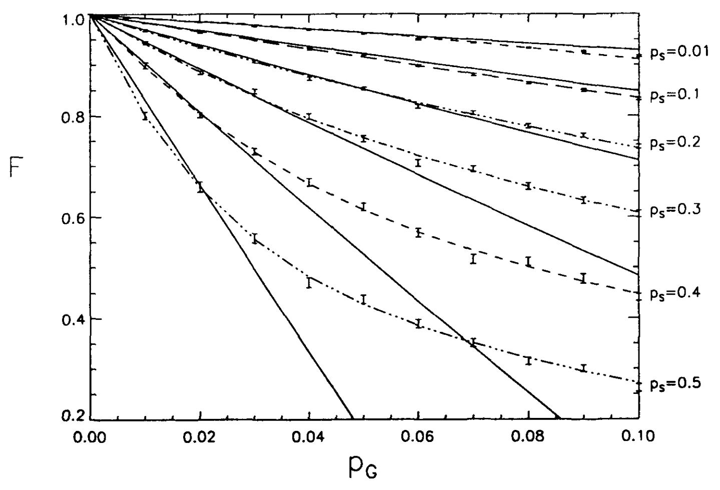


Figure 3.8: The fidelity data of Figure 3.5a (dashed lines), plotted along with the predictions of the edge wander theory (solid lines). Near $p_G \sim 0$, the edge wander theory always slightly overestimates the fidelity. When p_S is small, this trend continues for higher p_G, but for higher p_S (notably $p_S = 0.5$) the edge wander quickly exceeds the mean path fragment length and the theory consequently underestimates the fidelity.

taking p_G to be the observed gap frequency per strand. Alternatively, p_G can be calculated from the gap opening penalty ($-g$, where $g > 0$) using the formulae in the third column of Table 3.1. The values in the final column (labelled $F_{g=12}$) are the expected fidelities when $g = 12$. Note that the prediction for F is independent of the particular gap model being used (e.g. linear or affine). Equations (3.11)-(3.14) describe a random walk with an absorbing barrier and a reflection at the origin. These equations appear amenable to further manipulation to speed up calculations; for example, the distribution (3.10) might be successfully approximated by a more tractable distribution such as a Gaussian.

Figure 3.8 compares the predictions of this section with some of the results from simulation 2. There is a good correspondence between the edge wander predictions and the simulation data.

3.3.7 Estimating the fidelity of a particular alignment

Given a probabilistic model such as the one shown in Figure 3.1, the posterior probability of a particular coupling $(i \diamond j)$ can be calculated:

$$\Pr[i \diamond j] = \sum_{\mathbf{a}:(i \diamond j) \in \mathbf{a}} \Pr[\mathbf{a}]$$

The sum is over all paths that contain this coupling and is straightforward to compute using the Forward-Backward algorithm described in Chapter 2.

Using this result one can write down an expression for the expected overlap $\hat{A}(\mathbf{a})$ between a given alignment $\mathbf{a}$ and paths sampled from the posterior distribution. This is equivalently the expected number of correct matches in $\mathbf{a}$, which is a natural measure of the overall accuracy of $\mathbf{a}$.

$$\hat{A}(\mathbf{a}) = \sum_{(i \diamond j) \in \mathbf{a}} \Pr[i \diamond j]$$

where the sum is over all aligned pairs in $\mathbf{a}$.

It is also possible to write down $\hat{M}$, the expected number of matches in a path sampled from the posterior distribution (and the expected total number of matches in the real alignment):

$$\hat{M} = \sum_{\text{all } (i \diamond j)} \Pr[i \diamond j]$$

The above two quantities are posterior expectations of the numerator and denominator of (3.1). An estimate for the fidelity $\hat{F}(\mathbf{a})$ of a given alignment $\mathbf{a}$ is:

$$\hat{F}(\mathbf{a}) = \frac{\hat{A}(\mathbf{a})}{\hat{M}} \tag{3.16}$$

3.3.8 An optimal accuracy alignment algorithm

Given this new type of score for an alignment, it is possible to find the alignment that maximises this score, and hence has the highest predicted accuracy (by this defintion of accuracy, of course). The algorithm to do this has been described elsewhere [DEKM98] and is revisited here. The method required is identical to standard dynamic programming, but uses score values given by the posterior probabilities of pair matches; gap costs are not used. The dynamic programming recursion equations are:

$$A(i,j) = \max \begin{cases} A(i-1,j-1) + \Pr[i \diamond j] \\ A(i-1,j) \\ A(i,j-1) \end{cases}$$

and the standard traceback procedure will produce the best alignment [DEKM98]. The structure of this recursion ensures that the returned alignment will be legitimate, and the calculation of the cost function ensures that the alignment is optimised for the sum of the $\Pr[i \diamond j]$ terms along its path. Interestingly the same algorithm works for any sort of gap score; what will change with different scores are the $\Pr[i \diamond j]$ terms themselves, which are obtained from the standard, scoring scheme-specific dynamic programming algorithms referred to above.

An implementation of the optimal accuracy algorithm is available from `http://www.sanger.ac.uk/Users/ihh/optacc.html`

3.3.9 Simulation 3: Evaluation of the optimal accuracy algorithm

In order to test the prediction that the optimal accuracy alignment algorithm outperforms the Viterbi algorithm when the assumed model is correct, the sequence generation and re-alignment procedure of simulation 2 was repeated using the optimal accuracy algorithm.

Figure 3.9a shows the results of these simulations compared with the corresponding data for the Viterbi algorithm from simulation 2. It is clear the optimal accuracy algorithm has a significant advantage. Figure 3.9b is a plot of the expected fidelity (3.16) of these alignments against the measured fidelity. The correspondence is evident, supporting the validity of this particular statistic.

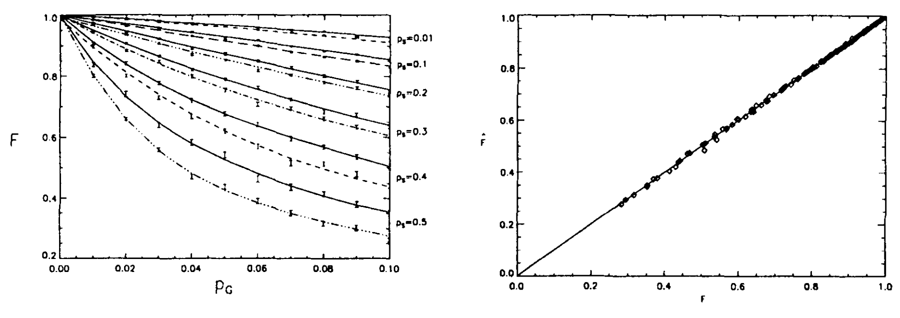


Figure 3.9: Evaluation of the optimal accuracy alignment algorithm. (a) Fidelity data for the Viterbi algorithm (dashed lines) plotted with data for the optimal accuracy algorithm (solid lines). (b) The expected fidelity plotted against the measured fidelity for the data points in (a). The Viterbi data are from simulation 2 (see Figure 3.5a) and the optimal accuracy data from simulation 3.

3.4 Discussion

It has been demonstrated that using a ***maximum likelihood*** scoring with the dynamic programming algorithm also appears to give maximally faithful alignments. With the aid of alignment fidelity measurements collected using a simulated model of evolution, the dependence of the alignment fidelity on the underlying mutation parameters has been discussed, and an analytic approximation (the *edge wander* approximation) describing this dependence has been presented along with a method for calculating the expected fidelity of a given alignment and an algorithm for finding the expected optimal-accuracy alignment.

These results demonstrate that the edge wander theory is a useful first-order approximation up to large values of p_S. Application of the theory to common substitution matrices predicts the extent of the unrecoverable loss of alignment information. The more distant the similarity, the less accurate we can expect the alignment to be. When aligning sequences diverged by 250 PAMs, for example, one must assume an average error of around 3.9 residues in the positioning of every gap, whereas that expected error is only 1.2 residues at 120 PAMs. In particular, we must not expect alignments for matches in the twilight zone of detectability to be accurate.

There is a statistical physics analogy that may help to give insight into the edge wander approximation. Consider the variable r whose probability density function $\rho(r)$ is given by (3.10). The mean value of r, $\bar{r} = \langle r \rangle_\rho$, is a relative entropy or Kullback-Leibler divergence between two probability distributions, representing the adjacent diagonals that the Viterbi path could lie on. The variance of r, $\langle (r - \bar{r})^2 \rangle_\rho$, is related to the fluctuations in this entropy-like quantity. The relative sizes of $\langle (r - \bar{r})^2 \rangle_\rho$ and $\bar{r}^2$ indicate the extent of the score fluctuations and equations (3.11)-(3.14) relate this to the error in the gap positioning, i.e. the edge wander. The edge wander approximation essentially assumes that the entropy (score) fluctuations are small and that the Viterbi path is "bound" to the correct path. This approximation is similar to perturbative approaches in statistical physics [LL80]. When edge wander breaks down, a full treatment of the critical scaling phenomena of the path behaviour is required. Terence Hwa, Michael Lässig and Dirk Drasdo [HL96, Hwa96, DHL97b, DHL97a] have published analyses of this problem that apply the theory of the renormalisation group, successfully used in areas of physics as diverse as quantum electrodynamics and chaos theory. The behaviour of the optimal path turns out to be analogous to the pinning of magnetic flux lines by randomly scattered defects in superconductors and the statistical behaviour of *directed polymers* in a random potential, both of which are well-studied by physicists. The renormalisation group is mathematically difficult compared to the probability theory used in this chapter, but it apparently has a lot to offer to the theory of sequence alignment algorithms. A notable result is that the renormalisation group theory predicts an optimal scoring scheme [HL96] that contradicts (3.8). This result is deserving of further investigation; a good starting-point would be to repeat Simulation 1 to greater precision.

The optimal accuracy algorithm described here and in [DEKM98] provides a marked improvement on the Viterbi algorithm. It will be interesting to see if this improvement carries over to real biological alignments. The simulations presented here also verify that the expected fidelity of an alignment is a useful indicator of alignment accuracy.

The observation that perfect alignment recovery is theoretically unattainable reinforces the idea that for some applications, it may be advantageous to consider a set or *envelope* of suboptimal alignment paths rather than singling out the highest-scoring path. Examples of such envelopes might include only residue couplings whose likelihood exceeded some cutoff value, or be defined by a set of path constraints chosen to maximise the sum of the likelihoods of the paths thus contained. An example of the former type has been proposed by Miyazawa [Miy94]; the issue of alignment reliability has also been addressed by Mevissen and Vingron [MV96].

In conclusion, it is noted once again that many of the results presented here are applicable to any dynamic programming based sequence homology algorithm, not just Needleman-Wunsch with linear gap penalties. Once there is a gap, the score changes involved in moving it as in the edge-wander calculation are the same for affine and linear gap penalties, and also for local and global alignments. It is hoped that the quantitative results for the alignment fidelities will be of use both to researchers in molecular evolution and to users of sequence alignment software.

Chapter 4

`postal`: Software for Checking Multiple Alignment Accuracy

4.1 Introduction

The idea of site-to-site reliability indicators for pairwise and multiple alignments is not a new one. Several such indicators have been proposed, ranging from residue conservation at a site, through sliding-window and exclusion [MV96] techniques to the fully probabilistic [Miy94, ZLL97]. There are many potential uses for a good reliability indicator; in addition to providing information that could help interpret alignments, such an indicator could be used to identify regions of a pairwise or multiple alignment that may be poorly aligned, thus providing further assistance to the sequence analyst.

As projects to classify proteins attempt to keep up with the expansion of databases, such automated sanity checks turn from luxuries to necessities. Release 3.1 of the Pfam database contains 1313 multiple alignments, each representing a protein domain [SEB+98]. Inspecting all these alignments for errors by eye is unfeasible and there is a clear need for automation. Recent efforts to establish a probabilistic basis for sequence alignment suggest posterior probabilities as a natural way of estimating alignment reliability [Miy94, ZLL97, DEKM98, Kro94, BH96, HD98]. Motivated by this, new software has been developed to check multiple sequence alignments for suspicious regions using posterior probabilities as alignment accuracy indicators.

In this chapter the mathematics of posterior probability are first reviewed. A new software tool - `postal` - based on the HMMER2.0 distribution [Edd95], that displays site-to-site posterior probabilities for multiple alignments and flags low-scoring regions for special attention, is then presented. The software is evaluated by running it on the October 1998 release of Pfam and assessing the pathology of the candidate misaligned regions that the program picks out. Further potential applications of Bayesian methods in sequence alignment are discussed.

4.1.1 Mathematical overview

In the probabilistic view, the score of an alignment $\mathbf{a}$ between a set of sequences $\{\mathbf{X}\}$ is proportional to the log of $\Pr[\mathbf{a}, \{\mathbf{X}\}]$, the likelihood of that alignment under some model that represents our assumptions about the way sequences evolve. (For example, the model might be that "pairs of related protein sequences have local regions of homology, with randomly scattered indel events and independently distributed patterns of amino acid substitution"; this contains the assumptions of the Smith-Waterman algorithm.) The likelihood $\Pr[C, \{\mathbf{X}\}]$ of a particular alignment segment C (such as, for example, an individual residue pair in a Smith-Waterman alignment) can be found by summing the likelihoods of all alignments that include that segment (i.e. $\Pr[C, \{\mathbf{X}\}] = \sum_{\mathbf{a}:C\in\mathbf{a}} \Pr[\mathbf{a}, \{\mathbf{X}\}]$), and the ***posterior probability*** $\Pr[C|\{\mathbf{X}\}]$ of the segment C is found by dividing the likelihood of C by the total likelihood of all possible alignments (i.e. $\Pr[C|\{\mathbf{X}\}] = \Pr[C, \{\mathbf{X}\}] / \sum_{\mathbf{a}} \Pr[\mathbf{a}, \{\mathbf{X}\}]$). This quantity $\Pr[C|\{\mathbf{X}\}]$ is the desired reliability indicator for the segment C.

This may be illustrated with a concrete example. Suppose one has a hidden Markov model (HMM) profile of a multiple alignment and a sequence $\mathbf{X}$ that one wants to fit to that profile. Suppose further that one wants to assess the evidence for whether position i of the query sequence is aligned to a particular state j in the profile, representing a site of interest. Begin by laying out the HMM profile and the query sequence on the vertical and horizontal axes (respectively) of a dynamic programming matrix (Figure 4.1). The alignment $(i \diamond j)$ of residue i to site j corresponds to the cell marked C in the matrix. To find the likelihood $\Pr[C, \mathbf{X}]$ that i is aligned to j, one must compute the sum of the likelihoods $\Pr[\mathbf{a}, \mathbf{X}]$ of all alignment paths $\mathbf{a}$ that run from the top left corner of the matrix through cell C and on to the bottom right corner. The algorithm for calculating the sum of alignment likelihoods is very similar in appearance to the Viterbi algorithm for calculating the highest-scoring alignment. Since alignment scores are additive, the top-left and the bottom-right quadrants can be treated as

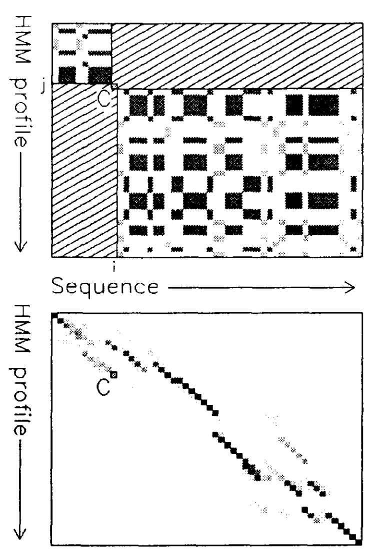


Figure 4.1: The dynamic programming matrix representation of the Forward-Backward algorithm. The Forward likelihood-sum-over-alignments (corresponding to paths spanning the upper left quadrant) is multiplied by the Backward sum (the lower right quadrant) to find the likelihood of all alignment paths passing through the cell C.

independent global alignments and the likelihood sum for each quadrant can be found separately. The likelihoods can then be combined to find $\Pr[C|\mathbf{X}]$. This procedure is known as the Forward-Backward algorithm and is described in more detail in Chapter 2.

For any sufficiently simple (i.e. Markov) model, $\Pr[C|\{\mathbf{X}\}]$ may be calculated using the procedure described above. It has been shown to recover the true probability distribution of C in simulations using a simple Needleman-Wunsch model (see Chapter 2 and [HD98]). From a Bayesian viewpoint, it makes more sense to work with the full posterior distribution $\Pr[C|\{\mathbf{X}\}]$ than to throw out all the cells C that aren't in the optimal alignment; the latter tactic is analogous to making over-precise numerical measurements without taking account of experimental errors.

Ideally, the dynamic programming approach described above would be carried over to simultaneous multiple alignment of many sequences. However, the size of the dynamic programming matrix scales geometrically with the number of sequences being considered and, while various heuristic methods may be employed to home in on the most likely alignment [HBF92, Edd95, NH96, LAB+93] an application of these methods towards estimation of the sums of likelihoods of many alignments remains appealing but untried. Alignment of multiple sequences to ready-made profile HMMs, on the other hand, scales much better computationally and requires no approximations or guesswork (other than during the construction of the HMM itself), which makes finding posterior probabilities for profile-based alignments that much easier. The software presented here uses the profile approach.

4.2 The `postal` software

The `postal` program builds a HMMER profile from a multiple alignment using C functions from the HMMER package [Edd95]. For each sequence in the multiple alignment, it calculates all the posterior probabilities along the alignment path of that sequence to the profile (this alignment is known, since it was used to construct the profile in the first place). The original multiple alignment is output together with single-digit annotation (indicating the first digit of the posterior probability for each site) in the `MUL` format for the `BELVU` program [SD94]. This format can also be read - and attractively displayed using colour - by the `jalview` Java multiple alignment viewer [Cla98].

The `postal` program has a number of options for advanced usage. For example, it can attempt to improve the multiple alignment (see also Section 4.2.3) or it can write the posterior probability tables directly to a file to be read by other programs. An algorithm utilising `postal` probabilities is in development at the time of publication [Gol98].

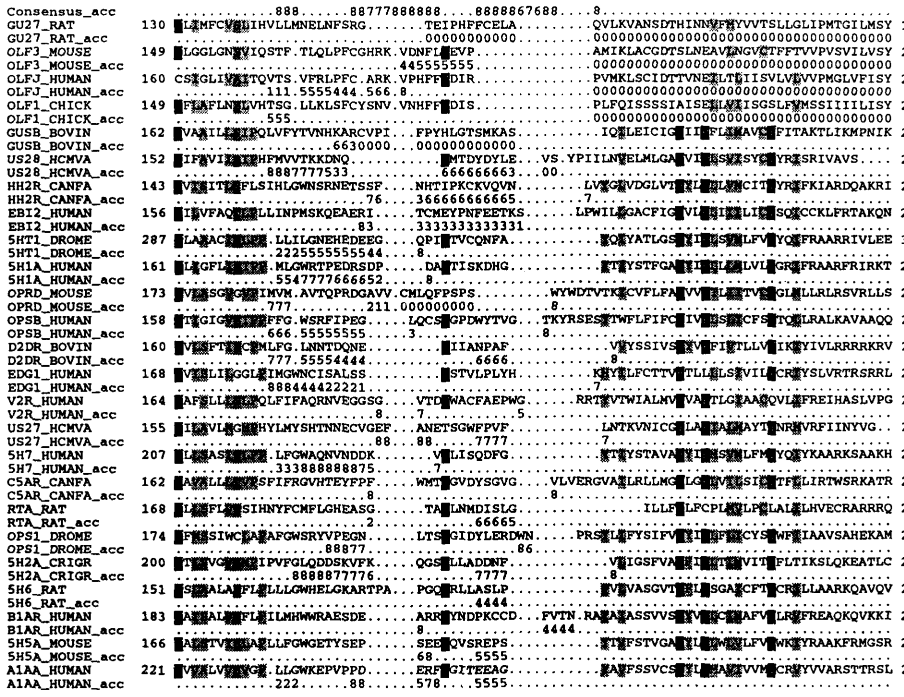


Figure 4.2: Accuracy levels for the 7tm_1 rhodopsin-like domain from Pfam (accession number PF00001). The sequences are sorted with the most suspicious at the top. The block of mammalian olfactory receptors with "0" accuracy on the top right-hand side are misaligned; to see this, note that the rest of the alignment has a column of conserved tyrosine residues aligned to residue 185 of the top sequence, while the misaligned block has a corresponding column of tyrosines at residue 199 of the top sequence. In other parts of the alignment, weak matches near gapped regions are often seen to have low probability.

4.2.1 Usage

Figure 4.2 shows an example of the output of `postal`, plotted by the `Belvu` multiple sequence alignment viewer [SD94]. The displayed alignment is part of the 7tm_1 rhodopsin-like domain from Pfam (accession number PF00001) - the same alignment as in Chapter 1, Figure 1.1. The aligned sequences are sorted with the most suspiciously aligned at the top. Beneath each sequence is an accuracy line, with the digits `0-9` indicating confidence levels for each site (low numbers signify low-accuracy regions, with a '`9`' indicating predicted perfect accuracy, a '`5`' indicating ambiguity and a '`0`' indicating that HMMER would rather put that residue with a different column). The top line is a "consensus accuracy" line obtained by averaging all the accuracy levels in a column, if the column comprises less than $\frac{2}{3}$ gap characters. By default, the (usually prevalent) digit '`9`' is masked out with dots to make suspicious regions easier to pick out. This alignment contains several suspicious regions, including one section that is clearly misaligned and several others where the column conservation is poor (see figure legend).

4.2.2 A note on interpretation

The posterior probabilities described here denote the confidence of the alignment model in a particular alignment. A low probability indicates ambiguity as to how a particular residue should be aligned. This is due to the absence of a strong signal, maybe because the sequence has little information content in this region, or because there are a lot of gaps nearby or even because the HMM training method is flawed. A high probability means that a sequence is well anchored, though not necessarily prettily aligned. For example, a run of mismatches sandwiched in the middle of an ungapped block will often have a high probability if the flanking sequences match the block consensus (though this may also depend on the gap-insertion policy of the algorithm). To take another example, given a handful of unrelated sequences, it is usually possible to train a probabilistic model to recognise these sequences well, despite the lack of homology between them. Assuming the sequences have any information content whatsoever, the (trained) model will then assert that the posterior probabilities of the training sequences are close to 1. In other words, probability theory is only as good as the underlying assumptions; the use of posterior probabilities may reveal ambiguities in an alignment, but without making new assumptions one may not be able to detect all the sequences that are badly aligned.

4.2.3 The optimal accuracy algorithm and `postal`

The `postal` program implements the optimal accuracy algorithm described in Chapter 3 for HMM profile alignment. This feature remains experimental and has not been systematically evaluated for HMM profiles.

4.2.4 More complex models

At the core of the `postal` software is the program `hmmbuildpost`, which calculates and prints a table of posterior probabilities for the alignment of a single sequence to a profile HMM. In addition, it finds the Viterbi and optimal-accuracy alignments of the sequence to the HMM (the optimal-accuracy algorithm finds the alignment $\mathbf{a}$ that maximises the expected overlap score $E(|\mathbf{a} \cap \mathbf{a}_{\mathbf{real}}|) = \sum_{C \in \mathbf{a}} \Pr[C|\{X\}]$; see Chapter 3 or [HD98] for an evaluation of this algorithm). The `hmmbuildpost` program is transparently invoked by `postal` and should rarely need to be used on its own.

The `hmmbuildpost` program has the same output format as the `modelpost` program, a more general tool for working with posterior alignment probabilities and optimal-accuracy alignments that is independent of the HMMER package. Details of this program are available on the `postal` web site.

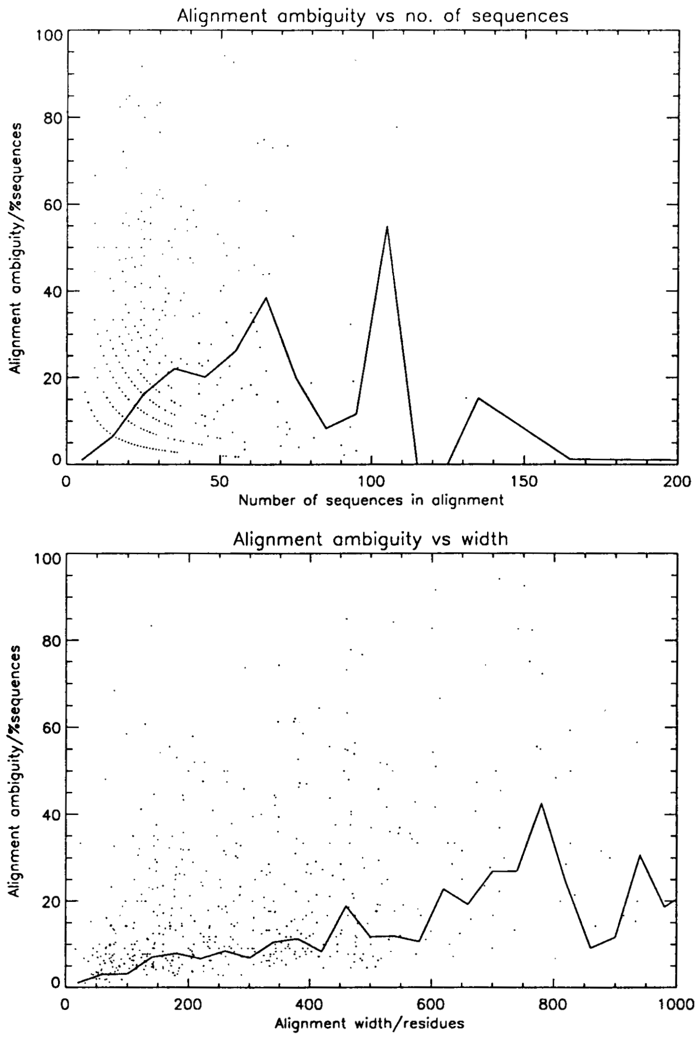


Figure 4.3: The two scatterplots show the proportion of sequences containing ambiguous regions, plotted against the total number of sequences in the alignment (upper plot) and the alignment width (counting both residues and gaps) (lower plot). There is a direct correlation between alignment ambiguity and alignment size. The solid lines represent y-averages for binned x-values.

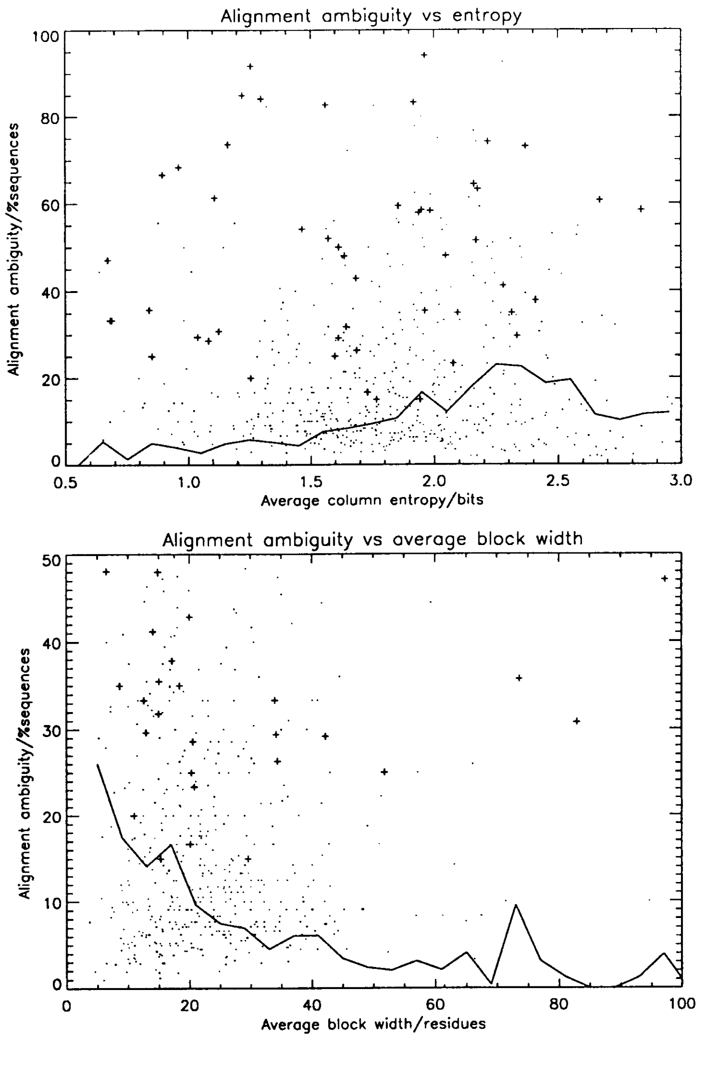


Figure 4.4: These two scatterplots show the proportion of sequences containing ambiguous regions, plotted against the average compositional column entropy (upper plot) and the mean size of ungapped blocks in the alignment (lower plot). Both plots show a distinct correlation. Data points corresponding to alignments with low consensus-accuracy regions are marked with "+" symbols. The solid lines represent y-averages for binned x-values.

4.3 Evaluation: using `postal` as a semi-automated quality check for Pfam

To test-drive the `postal` software, the program was run on each of the 1353 seed alignments in the October 1998 release of Pfam. The seed alignments were then sorted by the number of suspiciously aligned sequences they contained. (A sequence was regarded as suspicious if it had a run of at least 4 residues with probability less than 0.8; these are the default `postal` parameters.)

Of the 1353 seed alignments, 548 have suspiciously aligned regions - a total of 9459 individually suspicious sequences. The fraction of ambiguously aligned sequences is plotted against various properties of the alignment in Figure 4.3 and Figure 4.4. The ambiguity of an alignment appears to be directly correlated to its size (i.e. its length and width - see Figure 4.3). This is intuitively reasonable if poorly-fitting sequence segments are randomly distributed. The average size of ungapped blocks in the alignment and the average column entropy of the alignment, both plausible measures of alignment quality, are also good indicators of ambiguity (Figure 4.4).

As well as calculating site-to-site posterior probabilities for each sequence, `postal` also calculates a reliability indicator for the whole alignment - the "consensus accuracy" - by averaging the probabilities in each alignment column. Like the individual sequence accuracies, the consensus accuracy can be scanned for runs of low values to locate blocks in an alignment where many sequences are ambiguously aligned. In fact, the majority of ambiguously aligned sequences that are detected are due to blocks of this kind, as can be seen from Figure 4.4 where the low-consensus-accuracy regions are marked with "+" symbols. To suppress this effect, `postal` allows masking of low-consensus-accuracy regions; this feature is switched on by default. With low-consensus accuracy masking switched on, the 9459 ambiguously-aligned sequences reduced to 3569, though the number of ambiguous families (510) was comparable to the previous figure (548).

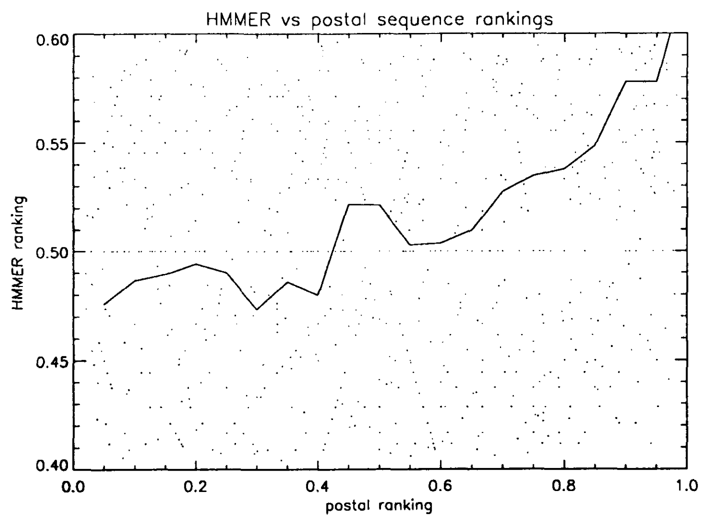


Figure 4.5: The mean rank of a sequence within an alignment, according to HMMER, plotted as a function of the rank according to `postal`. Ranks are fractional, ranging from zero for a poor score to one for a good score. For clarity, only every fifth scatter point is plotted. The solid line represent y-averages for binned x-values.

The worst 20 families in Pfam (without low-consensus accuracy masking) are listed in Table 4.1. Some turn out to be uninteresting from a practical viewpoint, since the low-accuracy stretches correspond to long inserts flanked by weak match states in the HMM and although the alignment of the sequence to the HMM is ambiguous, the effect on the multiple alignment is minor. Other poorly-scoring sequences seem to be outliers, distantly related to the other family members. Since outliers are expected to score poorly in an HMM search anyway, this raises the question: "do posterior probabilities perform any differently at detecting misaligned sequences to straightforward HMMER scores?". It is evident from a plot of the comparative rankings (Figure 4.5) that both methods rank sequences within a particular alignment in a similar way; however, `postal` gives "added value" in that it also reports which parts of an alignment are suspicious.

Accession no.	Family name	Alignment source	%(no.) misfits
PF00065	neur chan	CLUSTALW	94% (48)
PF00128	alpha-amylase	HMM simulated annealing	92% (50)
PF00516	GP120	CLUSTALW	91% (22)
PF00257	dehydrin	CLUSTALW	85% (17)
PF00500	late protein L1	CLUSTALW	84% (16)
PF00125	histone	CLUSTALW	83% (25)
PF00513	late protein L2	CLUSTALW	82% (24)
PF00555	endotoxin	CLUSTALW	82% (19)
PF01298	Lipoprotein 5	CLUSTALW	82% (14)
PF00933	glycosyl hydr14	CLUSTALW	80% (20)
PF00073	rhv	HMM built from alignment	77% (84)
PF00501	AMP-binding	CLUSTALW	76% (23)
PF00067	p450	Structure superposition	75% (48)
PF00009	GTP EFTU	CLUSTALW	74% (46)
PF00089	trypsin	CLUSTALW	74% (46)
PF01010	oxidored q1 C	CLUSTALW	73% (53)
PF00069	pkinase	CLUSTALW	73% (49)
PF00429	ENV polyprotein	CLUSTALW	72% (13)
PF00260	protamine P1	CLUSTALW	68% (13)
PF00360	phytochrome	CLUSTALW	66% (6)

Table 4.1: Top 20 suspicious seed alignments in Pfam. For each family, the fraction of sequences with low-probability runs is indicated, as is the source of the multiple alignment. The high representation of CLUSTALW [THG94a] reflects the fact that 87% of the alignments in Pfam were generated using this program.

4.4 Discussion

The `postal` program provides an indication of the local reliability of multiple alignments by using posterior probabilities as an accuracy measure. Tests on the Pfam database of protein domains lend promise to the program as a practical tool; often, putative low-accuracy regions correspond to areas of the alignment that the alignment algorithm finds ambiguous but that can quickly be resolved by inspection. For a large database like Pfam, manual correction of each alignment is unfeasible and a certain level of automation in the curation is mandatory; a system like `postal` offers a solution to the problem of maintaining the quality of over a thousand multiple alignments.

In Bayesian statistics, the full posterior distribution is generally regarded as a more stable basis for inference than just taking the most likely parameter values. Using posterior probabilities to estimate alignment accuracy is just one example of how this principle could be fruitfully applied to problems in sequence analysis. In common with Lawrence *et al* [ZLL97], it is anticipated that wherever numerical quantities are estimated from alignments, these quantities should be more accurately estimated by averaging over the entire posterior distribution of all alignments, particularly when the sequences are highly divergent and the alignment probability distribution is, consequently, broadly peaked.

4.4.1 Availability

Installation of the `postal` program requires the source code distribution, which includes the HMMER2.0 distribution [Edd95] and is available (under the terms of the GNU public license [GPL]) at the following URL:

`http://www.sanger.ac.uk/Users/ihh/postal/`

Installation and usage instructions are provided on the web site.

Part II

Studies in Evolution

Chapter 5

Wormdup: a Database of DNA Duplications in *Caenorhabditis elegans*

5.1 Chapter introduction

The evolution of new genes by duplication is a key component of molecular evolution. Of fundamental interest are the mechanisms by which genes are duplicated and the scale on which these duplications take place. There are many examples of searches for large-scale block duplications involving diverse genes (see e.g. [SKR89, WS97, Hug98]), perhaps the most striking of which was Wolfe and Shields' publication of evidence for a tetraploid duplication of the entire yeast genome [WS97]. Numerous examples of local tandem duplications of single genes, giving rise to two or sometimes more daughter genes, are also present in the literature (see e.g. [Sid96, BTR98, FBT+91, Eis98]) and indeed the abundance of this type of duplication is evident from a cursory inspection of the annotation of published genomic sequence.

From a neutralist argument one might expect that gene duplications represent special cases against a background of continuous turnover - involving duplication and reciprocal deletion - of non-coding as well as coding DNA. There are three main processes by which it is recognised that DNA duplication can occur in eukaryotes: (i) polyploidy, whereby an organism acquires a duplicate copy either of a single chromosome or its entire genome [Ohn70, WS97, BB98]; (ii) copying of host DNA during the process of transposon integration [Jur98] or excision repair [MKW91]; and (iii) unequal crossing-over between chromosomes during meiotic recombination [LG91]. The last of these - unequal crossing-over - may be triggered by numerous causes; experimental evidence suggests it can happen quickly where there is an existing tandem gene duplication [BTR98] and it can also be triggered by multiple, adjacent copies of a repetitive element [FBT+91].

Our understanding of the dynamics of gene creation and the relative importances of the different ways DNA can be copied is far from perfect, although population genetic models for these processes have been explored [TK98, Oht91]. With the increasing amount of genomic sequencing the paucity of data is likely to be replaced by the technical difficulties of gross analyses as the main obstacle to better understanding.

With these issues in mind, a database - "Wormdup" - oriented specifically towards researchers interested in studying aspects of genomic duplications has been constructed. Wormdup contains co-ordinates and age estimates of unique, single duplications of non-coding DNA in the recently sequenced [CSC98] genome of the nematode *Caenhorabditis elegans.* The focus is on non-coding DNA so that general features of duplications may be studied in the absence of gene-specific selective pressures. The score cutoffs used in the creation of Wormdup were chosen so that no duplications large enough to contain a gene should be missed. Various pre-calculated filters on the data are offered, including (amongst others) raw BLAST matches, gapped matches constructed using dynamic programming, duplications involving genes and large repeat families. In addition a suite of tools has been developed to facilitate the construction of more complex custom filters.

Section 5.2 of this chapter describes the structure of Wormdup, including the tools and algorithms that were used in its construction and may be used to query it. In Section 5.3, the Wormdup data are used to calculate various parameters of molecular evolution for *C.elegans.* These include the duplication size and separation distributions, the fixation rate of duplications and the subsequent rates of divergence by stochastic accumulation of substitutions and deletions. Section 5.4 investigates the number of Wormdup entries found in conformations suggestive of repetitive-element-mediated duplication. In Section 5.5, the molecular evolutionary parameters for non-coding duplications are compared to the parameters for coding duplications. It is found that the apparent fixation rate of gene duplications is higher than the rate for non-coding duplications of the same size. The implications of this discrepancy are discussed. In Section 5.6, the results of the evolutionary parameter-fitting are summarised and discussed.

5.2 Methods

This section describes the techniques used in the construction of Wormdup. The section begins with an overview of the nature of the algorithms required for a project of this kind, and proceeds to detail the process of construction of the core units of Wormdup.

The starting point for the entire analysis was the 72Mb of finished *C.elegans* DNA sequence available in May 1998.

A schematic view of the main stages in the construction of Wormdup is shown in Figure 5.1. The Wormdup data files and many of the tools and protocols are available from the following URL:

`http://www.sanger.ac.uk/Users/ihh/Wormdup/`

5.2.1 Overview of methods

Many of the common tasks involved in gross analyses of sequence features can be reduced to a series of manipulations on sets (or ordered sets) of sequence co-ordinates, where a set of co-ordinates for these purposes is defined as a ***(name,startpoint,endpoint)*** tuple (henceforth NSE). An example of an NSE tuple is the location of the gene AH6.2, which spans residues 5054 to 6308 of *C.elegans* cosmid AH6: the appropriate NSE is *(AH6,5054,6308)* in cosmid co-ordinates and *(CHROMOSOME_II,9624958,9626212)* in chromosome co-ordinates (since cosmid AH6 starts at base number 9619904 on chromosome II, according to the map used for this work).

A useful if basic format for representing lists of NSEs is GFF, the Gene Finding Format, developed in collaboration between the Sanger Centre, the University of California at Santa Cruz and other participants [GFF]. Each line of a GFF flat-file describes a single NSE with some additional information (such as, to continue the above example, the orientation of the gene AH6.2). This extra information is irrelevant to many of the algorithms (though this is by no means a hard-and-fast rule, with scoring information being the most frequent

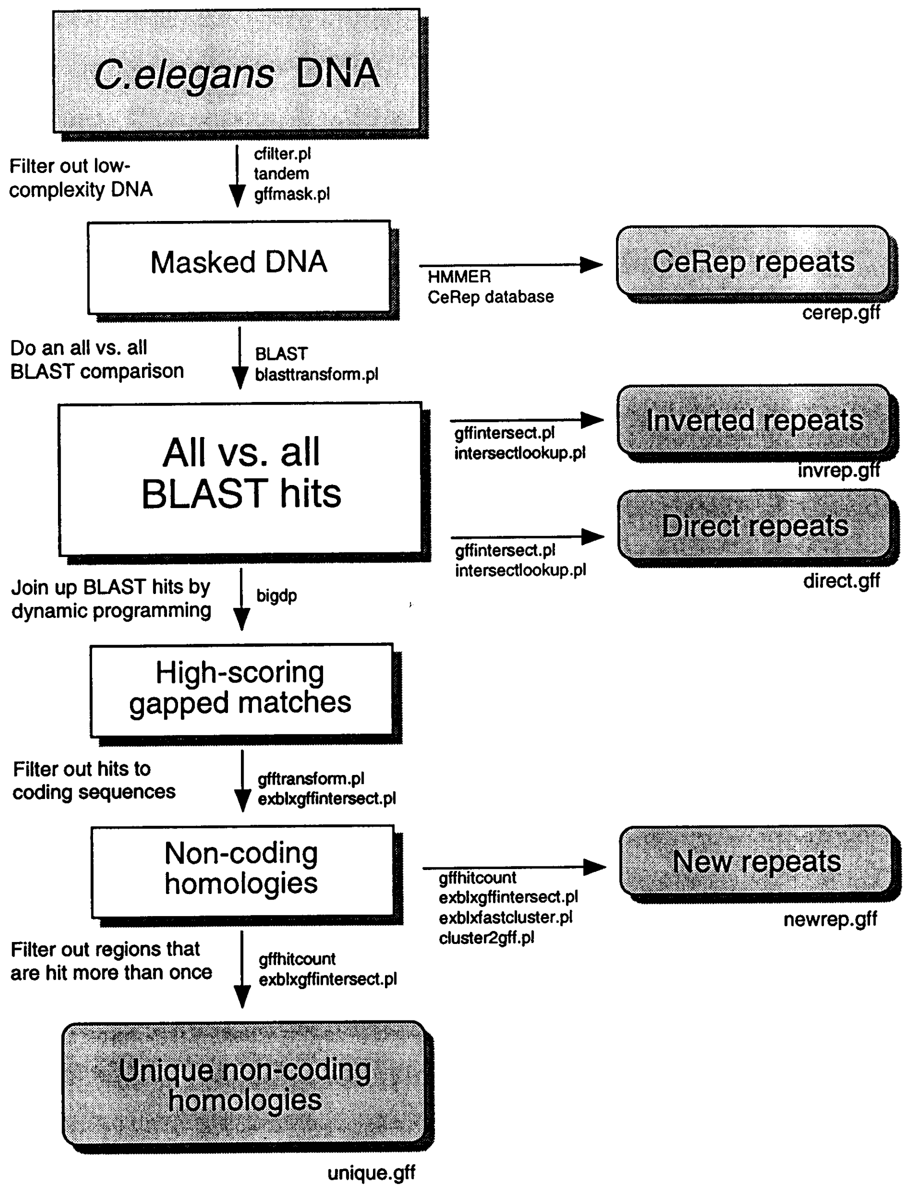


Figure 5.1: Schematic view of the construction of Wormdup. Names of key scripts and programs are shown to the right of/beneath arrows linking intermediate stages. Not all script names are shown, but all relevant scripts are described in Appendix A and available from the Wormdup website. The filenames with ".gff" suffices beneath shaded boxes refer to data files on the website.

exception). Single NSEs are inadequate to represent certain kinds of information (for example, homologies); a GFF-pair protocol exists for this purpose, though EXBLX (an output format of the BLAST post-processor `MSPcrunch` [SD94]) is in some ways preferable as a format for representing NSE-pairs as it is both more compact and more symmetrical. In any case conversion tools between these various formats were developed early in the analysis.

Apart from format conversion, the most common elementary operations that can be performed on (ordered) sets of NSEs include: (i) set intersection, (ii) set exclusion, (iii) set union, (iv) filtration, (v) sorting, (vi) merging (of sorted lists), (vii) transformation (of co-ordinate systems) and (viii) dereferencing (access to the described sequence). With a suitably flexible definition of NSE similarity, these operations form a basis for more sophisticated algorithms like clustering and tiling. Pointers to a comprehensive set of tools and links for manipulating NSEs in GFF and EXBLX format, including the GFFTools programs that were developed for this project, are maintained on the GFF website [GFF].

GFF is a relatively new format at the time of writing, and in many cases the tools described here were the first available for this format. In more cases, they were the fastest, being designed with respect to the consideration that reading millions of NSEs and NSE-pairs into memory at once is not practical.

Unless explicitly referenced, the tools described in the following sections were all developed principally for the Wormdup project. A full description of the GFFTools package may be found in Section A.4 of Appendix A.

Both GFF and EXBLX were found to be adequate formats for the present project; though it is the author's opinion that the single most pertinent feature of both formats, at least for working at the shell level, is the correspondence of a single line to a single feature.

5.2.2 Filtering low-complexity regions

The most commonly used low-complexity filter for use with DNA sequence is the `dust` program; however, the filtering heuristic used by `dust` is somewhat *ad hoc* [Tat]. A slightly more analytically supported method is the sliding-window entropy filter used by the `seg` program [Woo94], but this is designed for protein sequences. For the low-complexity masking of the worm DNA, a parameterisable sliding-window low-entropy filter `cfilter.pl` was specially written in Perl (see Appendix A).

Using the `cfilter.pl` program and the `tandem` program from the GCG package [But98], low-entropy regions (12-mer windows whose single-base compositional entropy did not exceed 0.5 bits) and microsatellites were identified, recorded in GFF format and masked from all subsequent analyses.

5.2.3 Preliminary scan for repetitive elements

A preliminary screen for hits to the CeRep database of repetitive sequence profiles was performed using the HMMER1.7 program. Local inverted and direct repeats (those missed by the `tandem` program) were also searched for by BLASTing each cosmid against itself. The lists of repeats were reduced by looking at the self-intersection of the list and taking the closest sequence-pair of every intersecting set, using the `gffintersect.pl` and `intersectlookup.pl` programs described in Appendix A.

The `tandem` search yielded $\sim$13000 tandem repeat regions; the average length of the tandemly repeated regions was 38 bases and on average there were 9 copies of this region. The number of inverted repeats was greater ($\sim$71000); this is probably because the `tandem` program attempts to join up multi-copy repeats whereas the inverted repeats are single copies.

The results of the screen for the CeRep elements are summarised in Table 5.1. A more thorough search for *mariner*-like transposable elements was also performed and is described in detail in Chapter 7.

Repeat type	Copy number in 72Mb	Expected copy number in 100Mb
CeRep10	2944	4088
CeRep11	551	765
CeRep12	2271	3154
CeRep13	1128	1566
CeRep14	1089	1512
CeRep15	713	990
CeRep17	635	881
CeRep18	597	829
CeRep19	2001	2779
CeRep20	504	700
CeRep21	469	651
CeRep22	316	438
CeRep23	1870	2597
CeRep24	2535	3520
CeRep28	540	750
CeRep29	545	756
CeRep30	102	141
CeRep31	145	201
CeRep32	719	998
CeRep33	109	151
CeRep34	2008	2788
CeRep35	1134	1575
CeRep36	782	1086
CeRep37	475	659
CeRep38	1017	1412
CeRep39	93	129
CeRep40	128	177
CeRep41	311	431
CeRep42	760	1055
CeRep43	3737	5190

Table 5.1: Copy numbers of CeRep elements in *C.elegans*.

5.2.4 Finding duplicated blocks

An all versus all ungapped BLAST comparison of the finished *C.elegans* DNA formed the basis for nearly all the rest of the Wormdup database. The search was performed with the version of the program designed for nucleotide-nucleotide comparisons, `blastn`, using the default scoring parameters (+5 for a match, -4 for a mismatch; this ratio corresponds to a Jukes-Cantor substitution distance $kt \simeq 0.16$ with a score-to-likelihood ratio of $S/L \simeq 5.2$). The score threshold for reporting hits was 120; low-complexity and tandem regions (but not CeRep, inverted or direct repeats) were masked out. The BLAST results were converted to EXBLX format by `MSPcrunch` [SD94] then transformed into chromosomal co-ordinates by the `blasttransform.pl` program described in Section A.4 of Appendix A. The data were sorted by chromosome-pair and redundancies and self-hits due to overlaps between cosmids were trimmed using the `exblxsort.pl` and `exblxtidy.pl` programs (also in Appendix A).

The ungapped BLAST hits were joined together by dynamic programming, using a program called `bigdp` that implements a modified version of the Waterman-Eggert algorithm [WE87] requiring $O(n)$ space, with a simple optimisation heuristic that reduces the expected compute time from $O(n^2m^2)$ to $O(n^2m)$ (where n and m are the query sequence lengths). The `bigdp` algorithm is described - with a worst-case scenario - and compared to other large-scale sequence comparison methods in Section A.5 of Appendix A.

The dynamic programming used a gap open penalty of 6 and a gap extend penalty of 0.8; using the score-to-likelihood ratio stated above, this corresponds very roughly to a gap frequency of 0.3 gaps per residue per strand and a geometrically distributed gap length with mean 6 residues. The cutoff score for reporting hits was 600, corresponding to a run of 120 perfect matches using BLAST. This high cutoff will exclude many small duplications (which are expected to be more numerous than large duplications) but it should pick up duplications large enough to potentially encode genes or exons, which are of primary interest. Hits scoring higher than the cutoff are referred to below as "high-scoring duplications".

5.2.5 Excluding genes and repetitive elements

The complete set of high-scoring duplications includes a large number of sequences with multiple hits. Some of these are matches between homologous genes in a multi-gene family and many more are matches between highly repetitive elements. These matches were excluded from the data set. Gene duplications are treated separately in Section 5.2.6 below and repetitive elements are addressed in Chapter 7.

Duplications involving genes were first identified from the *C.elegans* annotation and excluded. Next, repetitive sequence loci were identified using the `gffhitcount` program (described in Appendix A) which counts the number of times each base on a chromosome is covered by a high-scoring duplication. The distribution of base hit counts is shown in Figure 5.2. In total 48.7Mb (94.7%) bases of non-coding DNA were not hit by any high-scoring duplications at all; 625kb (1.2%) were hit once and 2.10Mb (4.1%) were hit more than once, where the percentages in brackets are the proportions of non-coding DNA that these totals represent.

A thorough classification of all the repeat families corresponding to multi-hit regions was not attempted. A preliminary rapid clustering identified 51 putative repeat families with over 10 copies in the genome (mean copy number 20 and mean sequence length 260). Among the clusters were the Tc3, Tc7 and Tc11 elements described in Chapter 7. These families account for 20% of the multi-hit bases suggesting (by extrapolation) that there may be as many as 200 additional families. Separating these by clustering is non-trivial: statistical analyses of repeat sequences show that certain families are often found together in the genome (Section 7.4.5 of Chapter 7) which can lead to conflation of clusters.

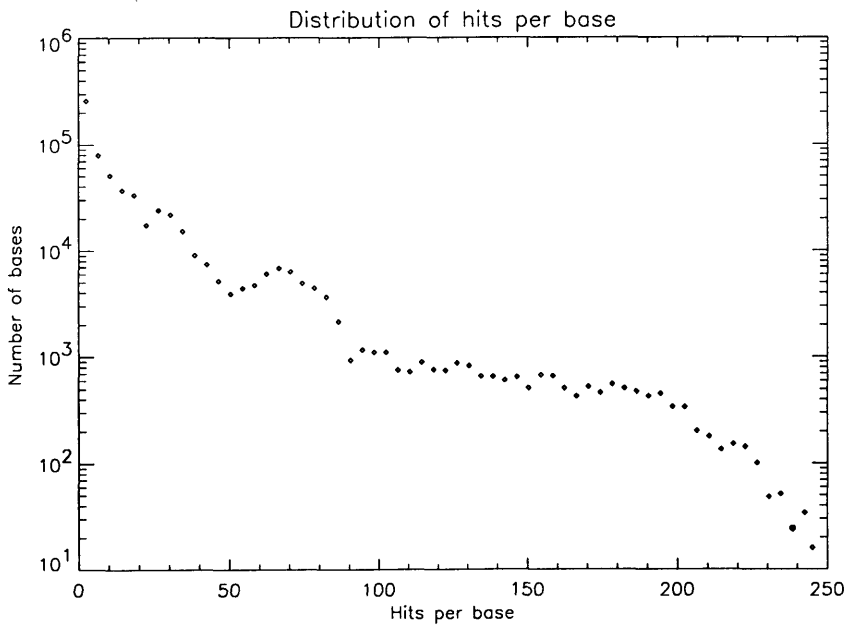


Figure 5.2: Frequency distribution of the number of times a base is involved in a high-scoring duplication.

A list of the putative new repeat families and the co-ordinates of their members is available from the Wormdup website. In total, 4520 regions that were multiply covered by the duplications data set were identified, leaving 1211 unique non-overlapping duplications in the data set. These were used for all subsequent analysis.

5.2.6 Gene duplications

In order to compare the fixation rates of non-coding and coding DNA duplications, a data set of gene duplications was independently constructed as follows. An initial search for homologies was performed for homologies between the 8065 proteins in the November 1996 release of Wormpep (the *C.elegans* database of predicted genes [SD97]) using the `blastp` program [AGM$^+$90] with the default BLOSUM62 substitution matrix [HH92]. Gene clusters were identified on the basis of homologies scoring over 1000 (i.e. 500 bits) or sharing over 80% sequence identity (with a score cutoff determined by the BLAST expected-hit-count parameter E=10). Minimal spanning trees for these clusters were constructed by neighbour-joining.

The above search yielded 369 multi-gene families, comprising 1035 genes. Construction of the minimal spanning tree resulted in a total of 666 duplicate pairs. This clustering is rather tight and splits up some large families; the main objective was to find a set of representative gene pairs for comparison with the non-coding pairs in Wormdup.

5.3 Statistics of duplications in Wormdup

Of the 1211 unique high-scoring duplications in Wormdup, 532 are on the same chromosome, with an approximately even split between same-orientation and inverted-orientation duplications. This suggests that many duplications are local and indeed, 52% of all same-chromosome duplicated blocks are separated by no more than 50kb. Wormdup duplications are more frequent near the ends of chromosomes, but this seems to be due to the bias induced by throwing out duplications involving genes (since genes are more densely clustered near the centre of chromosomes in *C.elegans* [ZR95]). The mean size of high-scoring duplicated blocks is 360 bases.

The number of pseudogene-like duplications in *C.elegans* (homologies between a predicted gene and a piece of non-coding DNA) was also estimated and found to be comparable to the number of non-coding duplications. The mean size of pseudogene-like duplications was also comparable to that of non-coding duplications.

5.3.1 Age distribution of duplications: the duplication fixation rate

A sequence alignment may be "dated" by finding a maximum-likelihood parameterisation of a time-dependent sequence divergence model. This is a standard technique in phylogenetic analysis and many such models have been developed; several are described in Chapter 2. The model used here was the 6-parameter model first described by Hasegawa *et al* [HKY85]; it assumes evolutionary neutrality and substitution rate constancy - the "molecular clock hypothesis".

Only the ungapped regions of the aligned duplicated blocks were used for fitting the time-dependent model. Although gap-aware models of DNA evolution exist [TKF92], their aptness is questionable. Some of these models are investigated in Chapter 6.

The Hasegawa model takes as parameters the background nucleotide composition (36% GC-content, in the case of the worm), the transition rate, the transversion rate, and the divergence time. This leaves a choice of scale for the divergence time; to fix this scale, the transition rate was set arbitrarily to 1, yielding the transversion/transition ratio k as a new parameter. This choice of scale fixes a unit of time at approximately 200 million years, though this is a general rate obtained by averaging synonymous substitution rates for a variety of phyla [LG91]. It is hard to obtain a nematode-specific figure because of the paucity of the nematode fossil record. However it is believed that the effective mutation rate has been abnormally high along the *C.elegans* lineage [Bla98].

The maximum-likelihood value of the transversion/transition ratio k and the divergence times $\{t_i\}$ of the Wormdup duplications were estimated by first choosing an empirical seed value of $k^{(0)} = 0.46$, then performing the following iteration, starting with $n = 0$: (i) fixing $k = k^{(n)}$, find the maximum likelihood times $\{t_i^{(n+1)}\}$; (ii) fixing $\{t_i\} = \{t_i^{(n+1)}\}$, find the maximum likelihood $k^{(n+1)}$. The optimisations at steps (i) and (ii) of this algorithm could be performed quickly by binary chop, since the posterior distributions of k and t_i are unimodal

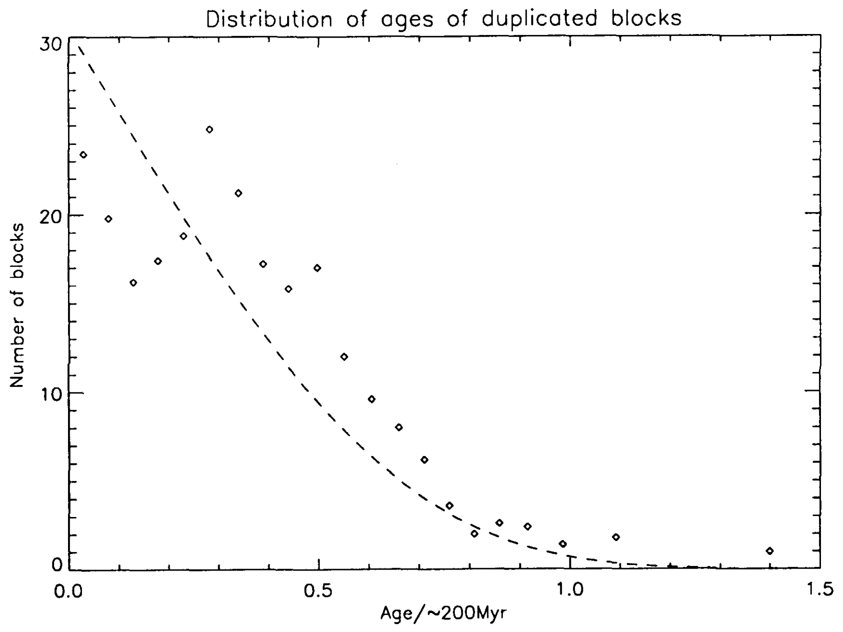


Figure 5.3: Age distribution of high-scoring duplications. The dotted line shows the hypothetical distribution that might be observed if duplication lengths were exponentially distributed, no duplications were deleted, and the only factor modulating the observed age distribution was the probability that, due to the random accumulation of substitutions and indels at the measured rates, the duplication would not score high enough to be picked up by the dynamic programming.

gamma distributions if the alignment is fixed. The algorithm was found to converge on $k = .49$ after 4 iterations.

The distribution of ages of Wormdup duplications is shown in Figure 5.3. The distribution is modulated by the probability that a duplication that old will score high enough to be picked up by the dynamic programming search (see Chapter 2). The dotted curve on Figure 5.3 illustrates this modulation assuming an initial geometric distribution of duplication lengths, which appears to be a reasonable approximation to the size distribution of recent duplications (see Figure 5.6). The observed data do not deviate plausibly from this distribution,

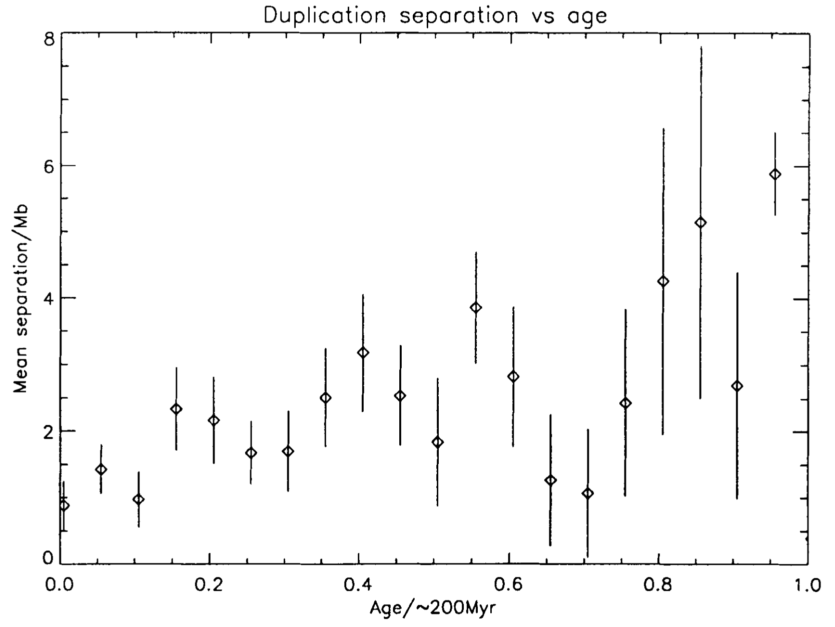


Figure 5.4: Mean separation of same-chromosome duplications plotted by age. The vertical bars indicate the standard deviation of the mean separation for each age bin.

suggesting that the rate of new duplications has remained roughly constant for the last $\sim$ 300Myr. An approximate fixation rate for high-scoring duplications can be estimated: 224 high-scoring duplications are detectable from the past 20 million years ($t \leq 0.1$), a rate of approximately 11 duplications per million years.

Figure 5.4 shows the separation of same-chromosome duplicated blocks plotted against the age of the duplications. The plot shows a slight upward trend. There is some statistical support for this; the log-odds ratio $\log \frac{\Pr[D|\mathcal{M}_1]}{\Pr[D|\mathcal{M}_0]}$ was calculated, where D is the observed data, $\mathcal{M}_0$ is a uniform Gaussian noise model for the separation data with an exponentially distributed mean (decay width 2Mb) that was integrated out and uniformly distributed standard deviation (up to 5Mb) that was optimised, and $\mathcal{M}_1$ is a linear regression model with the same

Gaussian noise and an additional exponentially-distributed gradient parameter (decay width 2Mb/t, where t is time in $\sim$ 200Myr units) that was integrated out. The log-odds score was 5.3 bits (though the hyperparameters were chosen after inspection of Figure 5.4).

This means that there is weak evidence that older duplicate blocks tend to be further apart than younger ones. Two possible explanations for this trend are offered here. The first possibility is that local duplications are being removed. One mechanism for this might be unequal crossing-over during recombination. Another might be if insertions tended to be smaller and more frequent than deletions. Although on average, the product of size and rate has to be equal for insertions and deletions if genome size is to be maintained, it is possible that relatively small, frequent insertions are balanced by relatively large, infrequent deletions (or vice versa). Large deletions in the region between a pair of duplicated blocks will be likely to delete one of the two blocks unless they are distantly separated, so the observed effect will be an excess of insertions. The second proposed explanation for the trend of older blocks to be further apart is the effect of large-scale conservative re-arrangements of the genome, such as reciprocal chromosomal translocations. Both explanations are consistent with the upward trend of Figure 5.4.

5.3.2 Length distribution of duplications: indel rates

Figure 5.5 shows the variation of average duplication size with age. The shape of this curve is mainly determined by the score cutoff. The initial downward slope is due to the accumulation of indel events with time, which modulate the length distribution (older duplications are likely to be split into smaller fragments). If the underlying distribution of gap lengths was exponential with a mean of 6 residues (corresponding to the affine gap scoring scheme used by the dynamic programming), then this effect would not be seen. This observation therefore implies that the probability of getting very large gaps is bigger than allowed for

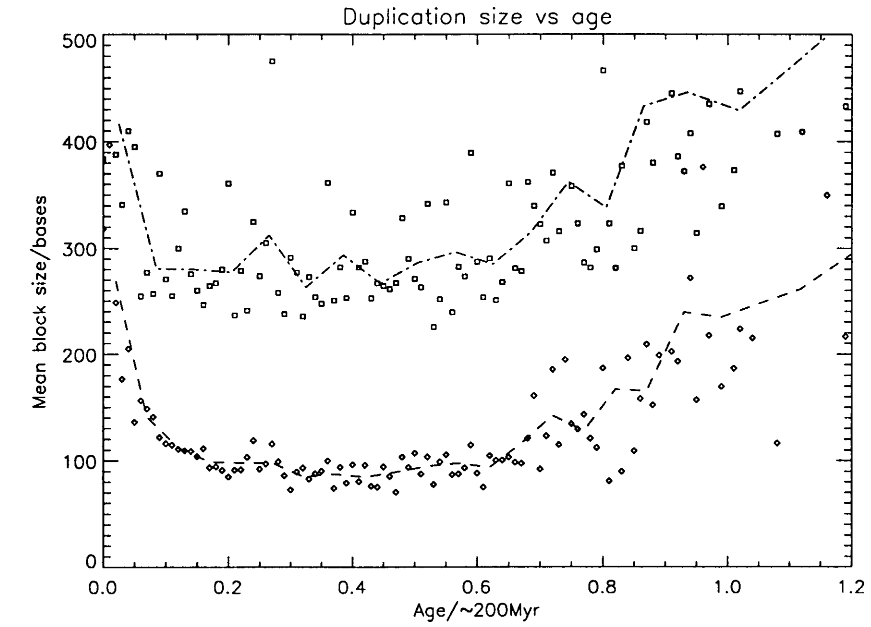


Figure 5.5: Variation of observed duplication size with age. The datapoints on the upper curve are the mean lengths of the entries in Wormdup for each age bracket. The points on the lower curve are the lengths of the constituent ungapped BLAST hits.

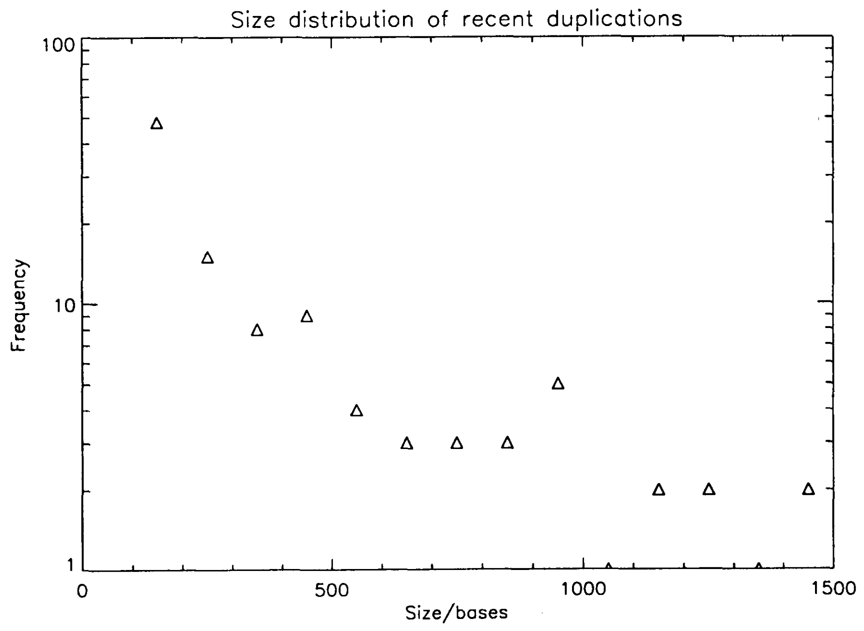


Figure 5.6: Size distribution of recent (< 10Myr) duplications. The frequency is plotted on logarithmic axes. A geometric approximation seems like a reasonable fit, although there is a hint of a broader tail suggesting that a power law distribution might be more appropriate.

by the exponential distribution. The real gap length distribution has a longer tail. Measurements of indel sizes in pseudogenes [GL95b] suggest that the gap lengths may be better modelled by e.g. a power-law distribution.

The upward turn of the graph at $t > .5$ happens because when the sequences are highly diverged, only the larger duplicated blocks stand any chance of scoring higher than the threshold for detecting hits.

A correction for the cutoff-induced bias to the observed length distribution may be derived from Bayes' theorem, and is included here for completeness although the method is not actually used. Write $\Pr[s|O,t] = \frac{\Pr[s|t]\Pr[O|s,t]}{\Pr[O|t]}$ where O is the event that a duplication is observed, s is the size of the duplication and t is the age of the duplication, which is conditioned upon throughout. An exponential approximation for the size distribution is $\Pr[s|t] \alpha \exp[-s(gt + 1/\mu)]$ where g is an indel rate and μ is the mean initial duplication size. (The actual distribution of sizes of recent duplications (younger than $\sim$10Myr) is shown in Figure 5.6. There is a hint of a broad tail, suggesting that a power-law distribution might be slightly more appropriate than an exponential distribution, but an exponential seems like a reasonable approximation to the distribution in Figure 5.6. The distribution will tend towards an exponential with time anyway, due to fragmentation by randomly scattered indels.) The probability $\Pr[O|s,t]$ that a match of length s scores high enough to be seen may be found by approximating the match score distribution with a Gaussian (see Chapter 2), whereupon $\Pr[O|t]$ may be found numerically.

A simpler approach is to first pick a maximum age (in this case $t = 0.25$) and throw away anything older than this, then find a mean size $\hat{s}_t$ for each age t, then fit a straight line to a plot of $1/\hat{s}_t$ vs t. The gradient of this line is the rate g of fragmentary indels (i.e. indels so big they wreck the chances of putting the pieces back together again) and the y-intersect is $1/\mu$. Applying this to the Wormdup duplications gives values of $g \sim .005$ (one big indel per 40kb per million years) and $\mu \sim 400$ (the average original size of the high-scoring

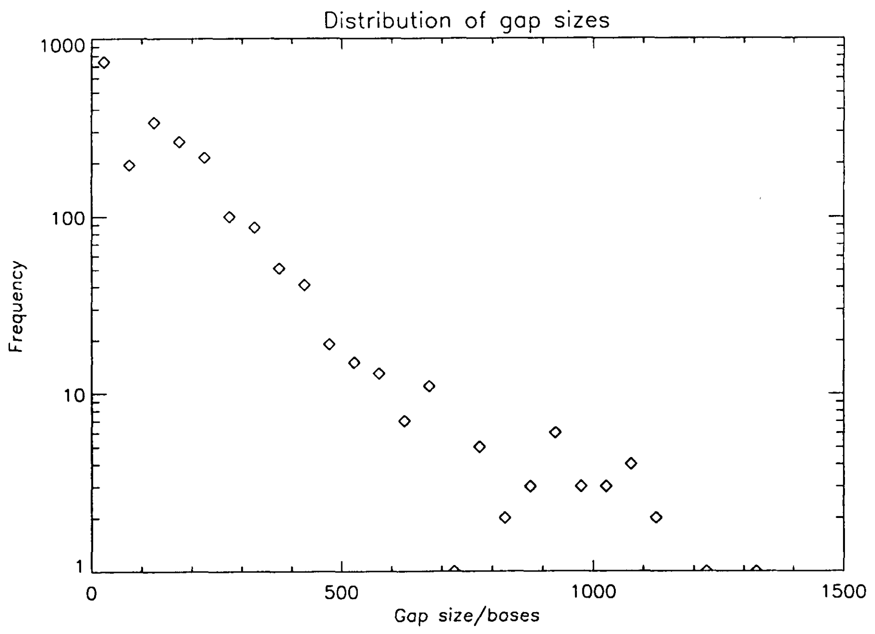


Figure 5.7: Distribution of indel sizes.

duplications was 400 bases).

A rate for small indels can also be calculated by looking at the age-length distribution of the BLAST hits from which the Wormdup duplications were derived. This gives a rate of $g \sim .03$ (one small indel per 7kb per million years) and $\mu \sim 224$.

Exactly what is meant by "big" and "small" indels? The size of indels in Wormdup appears to be exponentially distributed, with mean $\sim$ 150 bases (see Figure 5.7). This suggests that "small" indels are around 150 bases. The relative rates for small and big indels (.03 and .005) suggest that approximately 14% of indels are "big". The total rate for both big and small indels is $g_{\text{tot}} \sim .035$, or one indel per 6kb per million years.

5.4 Repetitive element-mediated duplications

Repetitive elements are known to cause duplications in a number of ways, including imperfect double-strand break repair following transposon excision [MKW91], accidental transpositon of non-transposon sequence [GL95b] and unequal crossing-over during meiotic recombination as a consequence of misalignment of adjacent copies of an element (as in Figure 1.3 of Chapter 1) [FBT+91]. In order to assess the relative importance of the latter two of these processes, a search of the *C.elegans* DNA was performed for patterns of repeat-flanked duplications whose proximity and relative orientation was indicative of repeat-mediated duplication.

The particular tool used was the `gffdp.pl` program, which implements dynamic programming using a generalised hidden Markov model of programmable structure with a pushdown stack. The `gffdp.pl` program is described (along with one of the models used for this search) in Appendix A.

In total 36 potential repeat-mediated duplications were found in the search, 22 spanning coding regions (though none of the duplicated blocks themselves intersected with coding regions). The maximum permitted separation between match segments for this search was 10kb; there are 226 matches in Wormdup that are this close together, so the hit rate to these matches was 16% (1 in 6). Nine of the 36 hits were of the form *repeat → match → repeat → match*, the pattern expected for unequal crossing-over events of the type shown in Figure 1.3. All 36 matches are available from the Wormdup website.

5.5 Comparison of non-coding duplications and coding duplications

Table 5.2 lists the 30 largest of the 369 clusters in the gene duplications data set whose construction is described in Section 5.2.6. Of the 666 gene pairs in the minimal spanning tree, 346 were on the same chromosome; of these, 198 were separated by under 20kb. Of the 346 same-chromosome gene duplications, 201 (58%) were oriented the same way; this proportion is even higher (64%) for those separated by under 20kb of sequence. No correlation between age and separation was found for these duplications.

The `bigdp` program (see Section A.5 of Appendix A) was used to search for blocks of genes duplicated *en masse*. Only one long-range (over 100kb) duplication involving over two pairs of genes was found, on chromosome II (pairing the three genes T05C12.3, F35C11.2 and F35C11.3 with W01C9.4, M05D6.3 and M05D6.1) and this was not very convincing, since the first gene-pair in the group is separated from the others by over 70kb. Seven long-range two-pair blocks were found.

The ages of gene duplications were estimated by fitting a time-dependent model to the observed frequencies of synonymous substitutions between the coding sequence pairs; this method is described in greater detail in Chapter 6. Most gene families are seen to have members with a wide range of ages, with notable exceptions being certain families of transposase and RNA-directed DNA polymerase proteins which probably dispersed rapidly.

If the molecular clock hypothesis holds, then the data would indicate a fixation rate of approximately 5 gene duplications per million years. 60% of these duplications involve multi-gene clusters; the rate of fixation of duplications not involving clusters is 2 per million years. Gene duplications tend to be bigger than non-coding duplications; the average *C.elegans* coding sequence is 2500 bases long, whereas the mean size of non-coding duplications in Wormdup is 400 bases. The fixation rate of gene-sized duplications in Wormdup, including pseudogenes, is estimated at 0.3 per million years. If the speed of the synonymous-substitution clock for coding DNA (the "codon clock") is the same as the speed of the substitution clock for non-coding DNA (the "background clock"), then this would indicate that gene duplications are fixed 7 times more frequently than non-coding duplications. The fraction of coding DNA in *C.elegans* is roughly

Family	Example	Youngest	Oldest	Brief identification
size	member	duplication/∼ 200Myr		
32	C25A8.1	0.04	6+	Major sperm protein (msp-142)
24	F38E1.1	0	6+	transposon reverse transcriptase
20	B0280.6	0	6+	transposable element
14	ZK666.2	0	6+	DNAJ protein like
12	C50F7.5	0	6+	cuticular collagen
11	F45E1.6	0.11	6+	Histone H3
10	C03G5.5	0	1.65	unknown
9	T06C10.5	0	0.46	RNA-directed DNA polymerase
9	K03A1.6	0.05	2.17	his-10, histone-H4
8	F45F2.2	0	1.65	histone H2B
8	ZC412.2	1.96	6+	guanylate cyclase
8	F55C10.3	0	2.79	cuticular collagen
8	F55G1.10	0	1.6	histone H2A
8	T01C1.1	0	6+	reverse transcriptase
8	K07F5.9	0.11	2.36	unknown
7	F08G12.6	0	6+	transposable element Tcl transposase
7	M03A1.5	0.08	2.17	small histidine-alanine-rich protein precursor (SHARP)
6	C08H9.7	0.39	6+	chitinase domains
6	F10F2.6	0.89	6+	C-lectin binding domain
5	T10B9.1	0.45	2.16	cytochrome P450
5	F38A5.9	0	0.06	unknown
5	F15B9.4	0.23	6+	unknown
5	ZK1248.9	0	6+	unknown
5	R04D3.1	1.27	6+	cytochrome P450
4	F52D2.3	0	0.36	transposition protein
4	T10E10.2	1.58	6+	collagen
4	C33G8.10	2.8	6+	C4-type zinc finger domain
4	F44F4.11	0	5.11	tubulin alpha-2 chain
4	ZK402.2	0.97	1.92	unknown
4	C24A3.1	0	6+	repetitive proline-rich cell wall protein

Table 5.2: The 30 largest duplicated gene families in *C.elegans*, with most recent and most ancient duplication ages also shown (dates older than 1200 million years are truncated). The clustering was tight, so that several large families were split up (e.g. cytochrome P450).

25%, so that the duplication fixation rate per base of coding DNA appears to be 20 times higher than per base of non-coding DNA.

Could the rate discrepancy between coding and non-coding duplications be due to the clocks being out of sync? That is, could the synonymous-substitution clock (or "codon clock") for coding sequences be running slower than the substitution clock ("background clock") for non-coding sequence? It is certainly easy to imagine how a wider range of mutations could affect non-coding sequence compared to coding sequence; any kind of mutation involving more than a single base will be strongly selected against in coding DNA. This would tend to make the background clock appear to run faster. On the other hand, the codon clock actually appears to run faster than the intron clock (the intron clock is based on counting the number of substitutions and indels that have accumulated inside introns; this clock is evaluated in Chapter 6). Although it is possible that the codon and intron clocks both run slower than the background clock due to selection pressures on both codons and introns, the rate for small insertions for the intron clock is similar to the background clock, suggesting an approximate correspondence. Furthermore, the observed divergence of introns is consistent with selection pressures on some introns, but not all. Variation in molecular clock rates have been reported elsewhere [GWD98] although the variations here are slightly larger.

If the fixation rate discrepancy is real, then it could be explained by positive selection pressure acting on gene duplications, greatly elevating their chances of fixation. Duplications of non-coding DNA are expected to be essentially neutral or even mildly deleterious, due to the increased DNA load. This effect would tend to elevate the relative rate of fixation of gene duplications, particularly if (as a hypothetical example) there were a selective sweep for increased dosage levels of a particular gene.

5.6 Discussion

A database of genome duplications called Wormdup has been developed from 72Mb out of the 97Mb of *C.elegans*, including a variety of tools for accessing the data set. Statistics for the database have been described, including the copy numbers of CeRep repeats and size, length and age distributions of unique duplications.

Unique non-coding duplications of the size range considered in Wormdup (mean 400 bases) become fixed at a rate of approximately 20 duplications per million years, including pseudogenes. This is a conservative estimate as multi-copy duplications were excluded. Although it is not yet clear what are the most important causes of duplication, some general trends are apparent: around half of all duplications are local in nature and no preference is shown for parallel *versus* inverted orientation. Around 1 in 6 of highly local duplications (separation <10kb) are near repetitive elements in conformations suggestive of repeat-mediated duplication, with around a quarter of these consistent with the kind of unequal crossing-over event illustrated in Figure 1.3 of Chapter 1.

Duplications do not appear to be systematically deleted on the million-year time scale, either by counterselection or by processes such as unequal crossing-over. The main process by which duplications deteriorate is stochastic accumulation of substitution and indel events. The data in Wormdup can be used to estimate the fixation rates of these kinds of small, local mutation. Fixing the rate of transitions at one substitution per 200 bases per million years, the transversion/transition ratio is estimated to be 0.49 - i.e. transitions are twice as common as transversions. Indels occur 1 every 6kb per million years; an exponential distribution with mean 150 bases models 86% of these indels. There is weak evidence that the mean separation between duplicated blocks increases with time. Two explanations have been proposed for this trend: (i) local removal of duplications, due perhaps to insertions being smaller and more frequent than deletions; and (ii) large-scale conservative re-arrangements such as reciprocal chromosomal translocations.

The ratio between the apparent duplication fixation rates of coding and non-coding DNA is rather large at 20:1. This may mean that that the estimated non-coding rate is too conservative or that the molecular clocks are mis-calibrated. If the difference is real, it would suggest that most non-coding duplications are lost from the population. This would imply that most gene duplications that become fixed have a selective advantage.

Selection favours gene duplications that preserve orientation, even though the underlying mechanisms of duplication appear not to discriminate between preserved- or inverted-orientation duplications. A possible explanation for the preference for same-orienation duplication is that operons are used to maintain similar regulatory control over both copies of a duplicate gene pair [BS97].

The analysis shows that there are many unclassified repeat families in *C. elegans*. 48 new families were identified by a very basic clustering and it is estimated that there are around 200 more. Around 60% of these repeats are located in the outer 50% of chromosomes. A full classification and derivation of consensus sequences for repeat families would be a non-trivial project, but worthwhile if only because of the potential role of repetitive sequences in triggering duplications and the consequences for their role in evolution.

5.6.1 Availability

The Wormdup data sets are available in full online, at the following URL:

`http://www.sanger.ac.uk/Users/ihh/Wormdup/`

Chapter 6

Intron Clocks: Time-Dependent Models of Intron Evolution

6.1 Introduction

The most widely accepted method of finding divergence times between coding sequences is to exclude all but the silent sites - bases which, due to the redundancy of the genetic code, may be varied without changing the translated protein sequence [LG91]. These are presumed to be free of selective pressures. Under the further assumptions that the average rate of substitutions at a given site is constant (the "molecular clock hypothesis") and that neighbouring-base effects can be safely ignored (see [Bul86] for an evaluation of the error due to this approximation), substitution at silent sites can be modelled with independent continuous-time finite Markov chains and Bayes' theorem applied to yield maximum *a posterior* (MAP) estimates of the divergence time.

There are several ways the "codon clock" method (as it is referred to from now on) can go wrong. Firstly, neighbouring base effects are non-negligible. Unfortunately, modelling these effects in full takes resources that scale exponentially with the sequence length. Secondly, the molecular clock hypothesis has been shown experimentally to be flawed [GWD98]. Thirdly, the assumption that silent sites are unselected ignores the effects of codon bias [SAL$^+$95] as well as the possibility that there are other signals in coding sequences. Finally, the MAP divergence time estimate only represents the maximum of the posterior distribution, which might be very broad.

In an attempt to address the problems of limited data and silent site selection, an alternative method of obtaining molecular clock information from introns has been developed and evaluated in comparison to the silent site approach. The "intron clock" method is straightforward in conception: ancestrally conserved introns are identified from an alignment of coding sequences by looking for aligned pairs of residues that are both on exon boundaries (it is assumed that the probability of deletion and re-insertion of an intron at the same locus is negligible). Time-dependent models of substitution and small indel events, of the sort described in Section 2.4 of Chapter 2, are then used to generate likelihood distributions over the divergence time. These likelihoods can be compared or combined with the likelihoods from codon clocks in a principled Bayesian way.

Section 6.2 of this chapter investigates general patterns of intron evolution, in order to assess the extent to which the patterns of mutations in introns are compatible with the models of Section 2.4; i.e. whether it is legitimate to fit small-indel and single-base substitution models at all. It is found that while many pairwise intron alignments are indicative of infrequent small indels, there is a significant fraction of intron pairs whose lengths are very different. A number of these contain high copy-number repetitive sequences and it is proposed that repeat element insertion is the most plausible cause of large mutations. This conclusion is discussed in the light of recent suggestions that new ***Drosophila melanogaster*** introns originate by duplication [TRTA98]. Following on from this, Section 6.3 suggests a Bayesian methodology for dealing with the problems of large insertions and uses the GFFTools and BayesPerl packages described in sections A.4 and A.3 of Appendix A to estimate molecular evolutionary parameters for intron evolution and to assess the performance of intron clocks relative to codon clocks. Finally, in Section 6.4 the results are summarised and discussed.

6.2 General patterns of intron evolution

The data set of introns was derived from the set of gene duplications described in Section 5.2.6. Conserved intron loci were identified from Smith-Waterman alignments of these coding sequences using the ACeDB annotation. Of the 1035 genes in closely-related families found by the search procedure described above, 46% were found to have at least one ancestrally conserved intron (and on average, two to three), yielding a total of 1142 conserved intron pairs. Visual inspection of the protein alignments suggested that 52 of these pairs contained an intron with a mispredicted splice site, since changing the splice site would radically improve the protein alignment. These 52 introns, and a further 9 that looked as if they might be mispredicted, were removed from the data set, leaving

1081 pairs.

Before investigating this data set further, some ideas are reviewed about the signals that are known to exist in introns and the selection pressures that are expected to apply.

6.2.1 Conserved signals in introns

Splicing - excision of introns from messenger RNA - takes place during the passage of the mRNA to the ribosome. The first stage of splicing is the binding of the U1 and U2 small nuclear ribonucleoproteins (snRNPs) to the splice site consensus sequences which span the 5' and 3' exon-intron boundaries respectively, to form a committment complex. This is followed by the ATP-driven binding of the U4, U5 and U6 snRNPs and subsequently by catalysed intron excision. During the first stage of intron excision, the 5' splice site is cleaved and rejoined in a "lariat" structure to an adenine residue located at the branch point, which is separated from the 3' splice site by a short pyrimidine tract and is also bound by U2 during committment. In the final stage of intron excision the 5' and 3' splice sites are joined and the lariat intron excised [HK94, Bir]. Splicing signals known to be present in introns thus include the 5' consensus (bound by U1) and the 3' consensus, the branch point and the intervening polypyrimidine tract (all bound by U2). The canonical *C.elegans* 5' splice site consensus is thought to extend at least 3 bases upstream and 7 bases downstream of the 5' splice site; the 3' splice site is shorter, but often merges into the polypyrimidine tract [Bir, CLB93, ZB96]. There may well be additional signals subtle enough to have escaped detection. The picture is further clouded by the presence of an alternative splicing system involving U12 snRNPs with a stronger branch-point consensus and a weaker 3' signal, although no U12-type introns have been found in *C.elegans* [BPS98].

Most (62%) *C.elegans* introns are between 40 and 60bp in length, so that selection pressures due to the need for the above splicing signals may be expected to act on around 15% to 25% of bases, most of which will be close to the splice sites. Although this will retard the effective substitution rate near the splice sites, one can also expect to see a higher substitution rate in the less selected regions relative to coding sequences, since DNA damage is often not confined to a single base and there are correlations in the local probability of substitution [LKW97, SO95, KB95].

6.2.2 Sizes of indels in introns

Apart from substitutions, the effects of insertions and deletions must be considered. Standard dynamic programming algorithms typically assume an exponential prior distribution over gap lengths [DEKM98], though a study of processed pseudogenes suggests a power-law distribution to be more accurate [GL95b] and algorithms implementing alternative gap penalties have been described [MM88, ZLL97]. One can get an overview of the sizes of indel events by plotting the log-frequency distribution of percentage differences in length between paired introns Figure 6.1. (The number of indels is expected to be proportional to the length of the sequence, so percentage differences may be more informative than absolute differences.)

Figure 6.1 shows that while the frequency distribution is reasonably well-approximated by an exponential fit up to around a 15% difference in length, there is a long tail that is not well described by an exponential or, indeed, by a power-law distribution. 23% of the intron pairs in the data set lie in this long tail region.

In an attempt to explain the observed elevated frequency of large indels in introns, the `gffintersect.pl` program described in Appendix A was used together with the HMMER [Edd95] and GCG suites, the published *C.elegans* annotation and the CeRep database of *C.elegans* repeat families to look for repetitive elements present in one but not both members of an intron pair. Of the 1081 pairs of introns in the data set, 7% contain repetitive elements in at

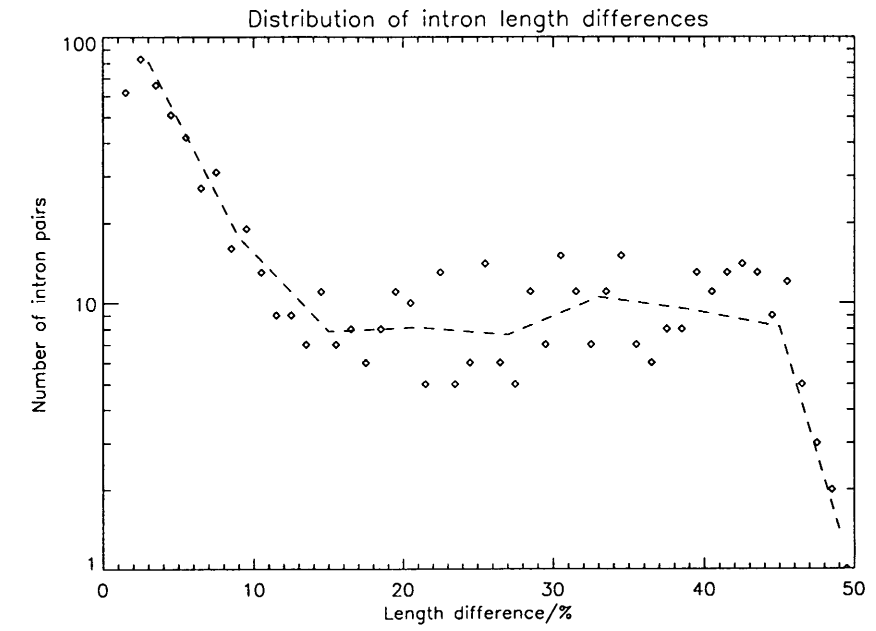


Figure 6.1: Distribution of fractional differences in conserved intron lengths. The dotted line is obtained by averaging nearby points. A large number of introns have length differences not easily explained by small indel models.

least one of the introns. The frequency of length mismatches between repeat-containing intron pairs is over 60% - considerably higher than the frequency for the whole data set. In all cases of length mismatch involving repeats, a repeat is found in one of the introns but not the other. Repeat insertion often appears to be associated with the formation of inverted and tandem repeats. These results are strongly suggestive that repetitive element insertion is a major cause of large indels in introns.

6.2.3 Intron mobility

It has been claimed that some *Drosophila melanogaster* introns at unaligned positions show significant homology, and that this is evidence for autonomous intron replication [TRTA98]. As part of the present study of *C.elegans* introns, a search for homologous introns was conducted using a BLAST search as a prefilter to the the Probabilistic Smith-Waterman (PSW) algorithm [BH96]. Gaussian distributions were fitted to PSW score frequency-distributions for random length-separated samples of the intron database to estimate "significant" (to 4 standard deviations) score thresholds. This is not a Bayesian approach, although a plausibility argument on Bayesian grounds is given in Section 2.7 of Chapter 2. The reason this approach was used was the woeful inadequacy of the "null" Bayesian generative model for introns: Figure 6.2 shows score-frequency curves for a pseudo-data set of introns generated from 4-mer frequencies (as might be sampled from a naive null model) and a real data set taken from the *C.elegans* intron database; there is a difference of 11 nats (= 16 bits) in the mean scores, though the variances are similar. To assess the significance of comparisons between introns of different lengths A and B (with $A < B$), a Gaussian score distribution with mean μ and variance σ^2 is used, where $\mu = \mu_A$ is the mean of a Gaussian fit to the score distribution of a random sample of sequences of length A, and $\sigma^2 = \sigma_B^2$ is the variance of a Gaussian fit to the score distribution of a random sample of sequences of length B. Roughly speaking, the

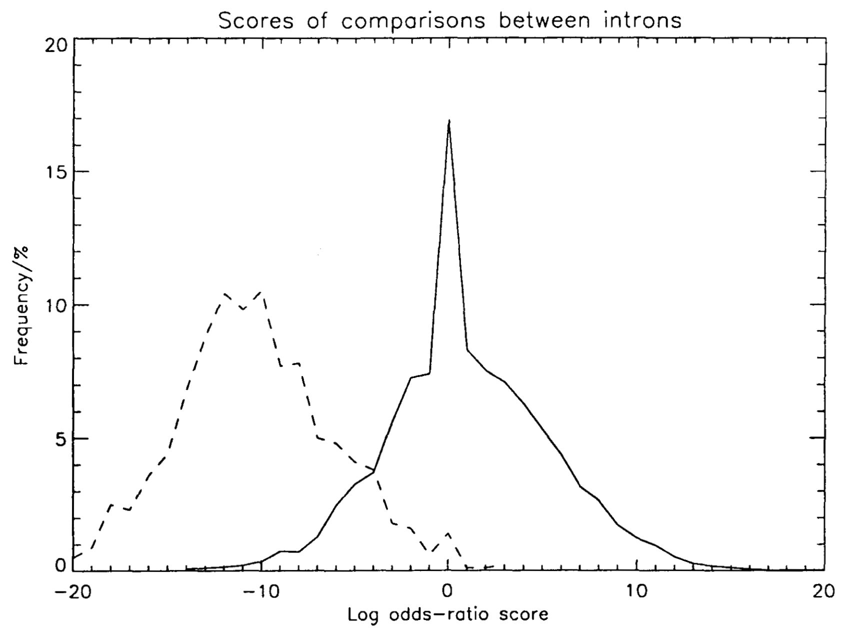


Figure 6.2: Log odds-ratio scores (relative to a null model of single-base composition) obtained by summing over alignments between randomly selected pairs of introns from *C.elegans* (solid line) and between pseudo-introns generated using 4-mer frequencies from real introns (dotted line). The sharp peak at zero is a fixed-precision rounding error.

average score can be expected to decrease with sequence length, but the variance to increase with length; so using μ_A and σ_B^2 gives the most conservative significance estimate.

The above search procedure yielded 110 intron pairs with "significant" homology. 61 of these were at aligned coding sites and 49 were unaligned. However, further analysis revealed that 34 of the 49 potential "mobile introns" in fact contained repeat sequences, accounting for their surprising homology by enclosed mobile elements, rather than intrinsic mobility. Three more of the homologies are between introns on the same protein, suggesting a tandem duplication. The remaining 12 significant intron homologies are listed in Table 6.1. Given the observed diversity of transposon species in the worm genome, and the low copy

Intron #1	Intron #2
AC3.6i2	C33D9.4i1
AC3.6i2	K03H1.5i7
B0024.6i16	C35A5.2i1
C05C10.2i6	E02H4.4i5
C05C10.2i6	F55E10.5i4
C09B9.1i2	K04C2.4i7
C03B8.4i2	W04D2.4i3
C05G5.6i4	T27B1.1i11
C05C10.2i6	PAR2.2i3
C05D9.6i3	F40E10.5i2
C11H1.4i11	F17A2.3i4
C18H2.2i5	K06B9.4i2

Table 6.1: Unexplained intron homologies in *C.elegans*.

number of some of these species, it seems entirely possible that many or all of the homologies in Table 6.1 correspond to uncharacterised mobile elements that happen to have landed in these introns.

A more sensitive search was carried out for examples of the proposed phenomenon of "intron drift" (slight dislocations in the positions of conserved introns relative to the coding sequences). No trace of this phenomenon was found.

6.3 Fitting time-dependent models to pairs of introns

Using the data set of conserved intron pairs, the hypothesis that introns are informative molecular clocks can be tested. Throughout this section the time-dependent coupled HMM with time-independent exponentially distributed gap lengths and Hasegawa substitution matrix described in Chapter 2 will be used.

To work with the time-dependent model, two pieces of software designed for this project were used. The first was a set of C++ classes designed to evaluate log-likelihoods of the form $\log \Pr[D|M, \Theta]$ where M is a (pairwise or single-sequence) hidden Markov model and Θ is a point in the parameter space of M.

To work with the likelihood data generated by the first program, a second piece of software that was designed to perform common manipulations on tables of log-likelihood values in multi-dimensional subspaces (including addition, multiplication, integration, marginalisation *et cetera*) was used; this software was written in Perl 5.0. Both these pieces of software are described in Appendix A.

6.3.1 Down-weighting uninformative pairs

Given the high number of introns disrupted by mobile elements or other kinds of mutation blitz, it would be useful to have a way of weighting intron pairs according to whether they look useful or not. A general, Bayesian way of doing this is as follows: Suppose that d is an element of data (in this case, a pair of introns) and that $D = \{d_i\}$ is an entire data set, and that it is desired to estimate a parameter Θ (or even a set of parameters, such as the divergence time of each pair). Suppose further that each data point d_i has an associated missing boolean variable $s_i \in \{0, 1\}$ indicating whether it is of relevance or not. More specifically, say that the data point d_i was generated by one of two models, M_0 or M_1, where M_0 is a null model that is independent of Θ (i.e. $\Pr[d|\Theta, M_0] = \Pr[d|M_0]$), and that the choice of model is determined by s_i; so that if $s_i = 0$, then the data point was generated by the null model and is uninformative for the estimation of Θ. To make this work, the posterior probability of Θ is marginalised over S as follows:

$$
\begin{aligned}
\Pr[\Theta|D] &= \sum_{\text{all } S} \frac{\Pr[D, S, \Theta]}{\Pr[D]} \\
&= \frac{\prod_i \sum_{s_i \in \{0,1\}} \Pr[d_i, s_i|\Theta] \Pr[\Theta]}{\Pr[D]} \\
&= \prod_i \sum_{s_i \in \{0,1\}} \Pr[d_i|s_i, \Theta] \Pr[s_i|\Theta] \frac{\Pr[\Theta]}{\Pr[D]}
\end{aligned}
$$

This approach weights contributions to the MAP estimate of Θ according to the posterior probability that the paired sequences are alignable (i.e. that they have not been disrupted by a transposon insertion). The prior probabilities $\Pr[s_i|\Theta]$ determine the weighting bias towards "alignable" or "unalignable". In theory, a time-dependent prior could be used, but there are too many different types of transposon-induced disruption to estimate a meaningful transposon insertion rate from the present data. For the present work, the dependence of the s-prior on Θ was dropped and a score cutoff of 10 bits (6.9 nats) was used, corresponding approximately to a $1000:1$ weighting against informative pairs ($\Pr[s_1] \simeq 0.001$).

6.3.2 Testing intron clocks

To test the intron clock hypothesis, the likelihoods of four different models were evaluated:

- Model $\mathcal{M}_0$: All introns mutate at different rates.
- Model $\mathcal{M}_1$: All introns mutate at the same rate, but this rate is not correlated to the rate of synonymous substitutions in coding sequences.
- Model $\mathcal{M}_2$: All introns mutate at the same rate, which is exactly identical to the rate of synonymous substitutions in coding sequences.
- Model $\mathcal{M}_3$: All introns mutate at the same rate, which is advanced or retarded by a constant factor, relative to the rate of synonymous codon substitutions.

Implicit in each model are the assumptions that (i) the molecular clock hypothesis is valid for synonymous substitutions in coding sequences and (ii) the previously described time-dependent gap HMM is a valid model for neutral intron evolution.

Since each of these models has a different number of parameters, it is necessary to integrate the likelihood across the entire parameter space of each (see e.g. [Mac92a] for a readable explanation of why integrating across the whole parameter space penalises models with more parameters). An approximation to this

Model	Brief description of model	$\log_2 [\Pr [\text{data}\|\text{model}]] - \log_2 [\Pr [\text{data}\|\mathcal{M}_0]]$
$\mathcal{M}_1$	Unsynchronised w/codons	655
$\mathcal{M}_2$	Perfectly synchronised w/codons	91
$\mathcal{M}_3$	Imperfectly synchronised w/codons	485

Table 6.2: Log-odds-ratios of synchronisation hypotheses for intron and codon clocks, relative to the null hypothesis that introns do not show clock-like behaviour at all ($\mathcal{M}_0$).

integral was found using the trapezium rule with a finite range for divergence times of $0 \leq t \leq 10$ with a time-step $\Delta t = 0.05$. Model $\mathcal{M}_3$ has an extra parameter $r = t_i/t_c$ determining the relative rates of the intron and codon clocks; this was integrated over $0 \leq r \leq 2$ with $\Delta r = 0.1$. Uninformative (flat) priors were used for r and t. Strictly speaking, parameters such as the gap-open rate g, the mean gap length l and the transversion/transition ratio k should be integrated over as well, but these were also approximated by $g = 0.039$, $l = 1.2$ and $k = 0.53$, values which were obtained by a crude approximate Viterbi-likelihood method. Uninformative intron pairs were down-weighted, as described in the previous section. The likelihood calculations and the numerical integration were performed with the aid of the LogSpace and BayesPerl packages described in sections A.2 and A.3 of Appendix A.

The log odds ratios (in bits) of models $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ to the null model $\mathcal{M}_0$ are shown in Table 6.2. The clear winner is model $\mathcal{M}_1$: intron clocks are synchronised between introns, but do not bear any relation to the synonymous codon substitution clock. A clue as to why this might be is offered by the superior performance of model $\mathcal{M}_3$ over model $\mathcal{M}_2$; recall that model $\mathcal{M}_3$ allowed intron and codon clocks to be out of sync by a constant ratio, whereas $\mathcal{M}_2$ required that they stay in exact step. In fact the maximum-likelihood value of the relative clock-rate parameter r was $\hat{r} = 0.1$, suggesting that to make intron clocks work under the present model, they would have to run significantly slower than the codon clocks. When the analysis was repeated without alignment weighting (which will tend to introduce a negative bias to the intron clock rate) the ML value for r rose to $\hat{r} = 0.5$ but the ranking of the four models remained unchanged.

It is possible that a time-dependent prior for whether sequences were alignable would improve the performance of the clock-like models. The limited range over which the time parameter was integrated may also be a source of error.

These results suggest that while there is hope of fitting time-dependent models to non-coding DNA (and, in particular, to introns), the current models are far from perfect and are not yet suitable sources of clock information. The use of codon clocks is itself known to be a flawed technique (see, for example, [GWD98]). With the increasing availability of non-coding DNA sequence, it might be a good idea for studies of non-coding DNA evolution to consider unpredictable, traumatic mutations as well as the tractable single-base substitutions and small indel events that are more typical of coding DNA.

6.4 Discussion

Introns within the same gene evolve at the same rate in *C.elegans*, but this rate does not correspond well to the rate of syonymous substitution in the coding sequence. If the correspondence is made, however, it is better to allow the introns to evolve at a slower rate than the synonymous sites. This suggests that the selection pressure on introns is greater than on synonymous codons. One reason for this could be that *C.elegans* introns are rather short and around 20% of the average intron sequence length is taken up by splicing signals. Another reason could be small genes (for e.g. snRNAs) in *C.elegans* introns; these will be subject to selection.

The distribution of length differences between introns in homologous positions suggest that while 77% of intron pairs have diverged according to the kind of time-dependent stochastic model of small-indel accumulation proposed by Thorne *et al* [TKF92], the remaining fraction of pairs have been subject to large insertions or deletions. Introns containing repetitive elements are strongly associated with this effect, suggesting that repetitive element insertion is a primary cause of large mutations in intron sequences. It is proposed that repetitive elements also account for some of the surprising homologies that are found to exist between intron sequences.

The inadequacy of the default null model for introns has implications for the design of genefinding algorithms. A more sophisticated model should not only take account of the known splicing signals within introns, but also reflect the empirically observed propensity of unselected sequence for low-complexity regions such as poly-AT tracts. Modelling these kinds of features with HMMs demands large state spaces since the lengths of the features are not geometrically distributed. To avoid the training problem, the effective number of parameters can be reduced. An outline of the derivation of constraints on HMM parameters for modelling complex length distributions is given in Section 2.7 of Chapter 2.

6.4.1 Availability

The gene duplication data described here are available with the rest of Wormdup at the following URL:

`http://www.sanger.ac.uk/Users/ihh/Wormdup/`

Chapter 7

Classification of DNA Transposons in *Caenorhabditis elegans*

7.1 Abstract

7.2 Introduction

Transposable elements - parasitic elements, integrated into the host genome but exhibiting mobility in their genetic locus - constitute a significant fraction of eukaryotic genomes [Jur98, Smi96]. From a practical viewpoint, repetitive DNA has a detrimental effect on database searching as it spoils the assumption of sequence "randomness" on which statistical methods rely. Transposons are also emerging as players in genomic evolution, with occurences of repetitive elements reported in introns [Wes89], promoter regions [OGB95] and coding regions [BHP89]; they have also been hypothesised to trigger meiotic recombination [NCC$^+$92, YWB97, FBT$^+$91]. Transposons provide useful vectors for germline transformation [PvL97]. As model selfish genetic elements, their population dynamics is an interesting topic [YWB97, LC97, HLL97, HLNL97].

Transposons are distinguished from other types of repetitive DNA - such as microsatellite repeats and unique tandem duplications - by their spontaneous re-insertion at new positions in the genome. In doing this they pass through an intermediate phase either as RNA which is then reverse-transcribed back into the genome, or as double-stranded DNA which is excised and re-integrated elsewhere. The two kinds of transposon are described as "class I" or "class II". Both classes can be further categorised according to whether they are autonomous or non-autonomous: transposons in the former category contain genes coding for all the transposase proteins necessary for mobilising transposition, whereas those in the latter, non-autonomous category depend on enzymes provided by the former category and are often nicknamed "hitch-hikers" [Smi96]. Hitch-hikers may be closely related to autonomous elements by mis-sense mutations or may display similarity restricted to the tranposase binding sites [RvLDP97, OGB96].

The presence of parasitic hitch-hikers is posited to be detrimental to the reproductive success of transposons, especially so for DNA tranposons, whose tranposase proteins may have a more difficult job finding the particular sequences from which they were transcribed, leaving them vulnerable to parasitic mimics [HLL97, LC97, HLNL97]. Two mechanisms by which a transposon may avoid becoming overburdened with hitch-hikers include: (1) evolution of new specificity in its transposase-nucleic acid interactions; (2) invasion of fresh host genomes that are free of hitch-hikers. It may be envisaged that these work in tandem, i.e. new genomes provide the spatial heterogeneity necessary for new specificity to evolve. The presence of hitch-hikers is just one factor proposed to restrict the mobility of transposons; others include DNA methylation [YWB97] (though this is absent in *C.elegans*), self-inhibition [LC97] and titration by defective transposase proteins [HLL97].

One of the most widely studied families of DNA-mediated transposable elements is the Tcl/*mariner* family [PvL97, HLL97]. Members of this ubiquitous family typically contain a two-exon gene of around 300-400 codons flanked by short (11-80bp) terminal inverted repeats (invreps). The Tcl transposase, which has been demonstrated to be sufficient to mediate transposition in *C.elegans* [VBP96], catalyses the staggered double-strand endonuclease cleavage of the DNA substrate and re-integration of the transposon into the sequence TA [HLL97, Cra95, LCR96, vLCP94, VBP96]. Tcl excision is followed by double-stranded DNA breakage repair, which can entail a variety of mutations including deletions, insertions and duplications [MKW91]. The putative domain structure of the Tcl transposase is shown in Figure 7.1. Three domains have been proposed [VvLP93]: (i) a specific DNA-binding domain that binds between bases 5 and 26 of the Tcl invrep and shows weak transitive homology to the DNA-binding domain of the Drosophila *paired* gene, a transcription factor involved in embryonic development [FLD$^+$94, GW92]; (ii) a non-specific DNA-binding domain that might be responsible for DNA-protein interactions determining the structure of the transpososome [VvLP93]; and (iii) a catalytic domain that belongs to the D35E superfamily of transposases and retroviral integrases, the struc-

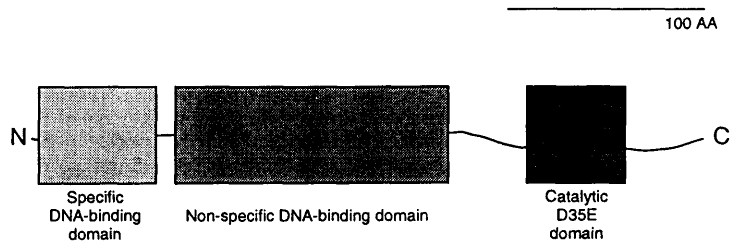


Figure 7.1: The putative domain structure of the 343-amino acid Tcl transposase protein. The 63 N-terminal residues bind specifically between bases 5 and 26 of the Tcl terminal invreps. The corresponding domain in the *Minos* elements from *Drosophila hydei* show weak homology to the *paired* gene in *Drosophila* [FLD$^+$94]. Amino acids 71 to 203 contain a non-specific DNA-binding domain [VvLP93] and amino acids 247 to 296 are the D35E catalytic motif that is highly conserved in a number of transposase and viral integrase proteins [DDJH94].

tures of several members of which have been solved [DDJH94, GL95a]. There may be additional, cryptic DNA-protein interactions affecting transposase activity [VvLP93]. It is thought that the terminal 6 bases of the Tcl invrep are important for catalysis [VP94]. The Tc3 transposase has a similar catalytic mechanism but binds to two regions in the (longer) Tc3 invrep, rather than one [CvLP94].

Computational analyses based on the clustering of inverted repeats have uncovered several putative families of transposable elements in various organisms [OGB95, OGB96, Smi96] including the partially sequenced genome of the Bristol N2 strain of *C.elegans* [SDT$^+$92]. At least one of these families has since been demonstrated to be mobile [RvLDP97]. In this chapter, a comprehensive list of all previously characterised transposons (including those described above) in the *C.elegans* genome is first presented. A computational approach to identifying new members of a DNA transposon family is next described; this approach is used to identify several previously uncharacterised subfamilies of the Tcl/*mariner* group. Phylogenetic evidence suggests that at least one of these families may have been active in the recent past. Profile hidden Markov models [DEKM98] of the inverted repeat sequences characterising the new families are published as part of the *C.elegans* annotation [CSC98] and in the Wormdup release along with annotation files describing the locations of identified elements.

7.3 Methods

The basis for the present analysis was the 90Mb of *C.elegans* DNA sequence available as of October 1998, together with the published annotation in ACeDB [ED95] and the CeRep database of common worm repeats [CSC98].

7.3.1 Construction of the transposon family data set

Three principal techniques were used in constructing the data set of novel transposable elements:

- the identification of inverted repeats (following [OGB95]),
- sequence homology and clustering at the protein coding level (following [Smi96]) and
- sequence homology and clustering at the DNA level.

Inverted repeats

A list of all invreps was constructed by screening each cosmid against itself using `blastn` version 1.4.7 [AGM+90]. The cosmid-by-cosmid approach introduces a coarse-graining over the ideal approach of comparing chromosomes whole; however, the separation of most *C.elegans* transposon invreps is considerably less (around 3kb) than the typical length of a cosmid (around 40kb) and the artificial length cutoff introduced should be negligible; this differs from the approach in [OGB95]. Before performing the BLAST search, low-complexity regions (using the `cfilter.pl` program described in Appendix A) and tandem repeats (using the `tandem` program from the GCG package) were identifed and masked out.

The invrep data were reduced by a factor of approximately two by taking the closest invrep-pair wherever a conflict arose, using the `gffintersect.pl` and `intersectlookup.pl` programs described in Appendix A. This left approximately 72000 invreps.

Protein sequence homology

To reduce the size of the invrep list, homology information was used to restrict the search to elements encoding a transposase protein of the D35E superfamily. This family is widely diverged [DDJH94] and it is anticipated that there will also be pseudogene-containing variants with internal deletions or insertions [OGB96]. For these reasons the `blastx` program, which searches the six-frame conceptual translation of the genomic sequence, may not be sensitive enough. A more sensitive program is GeneWise [BD97], which finds the optimal alignment of conceptually translated genomic sequence to a hidden Markov model (HMM) and is robust to gaps and frameshifts. HMMs were trained individually on three separate seed alignments: the first produced using CLUSTALW [THG94a] from all the *mariner* transposases in SP-TREMBL and the latter two derived from previous analyses of D35E subfamilies [DDJH94, SR96] and homologous sequences in SP-TREMBL. The shortest seed alignment is shown in Figure 7.2.

The available worm DNA was searched with these HMMs using GeneWise, and the matches combined with the list of invreps by dynamic programming using the `gffdp.pl` program described in Appendix A, to yield a set of predicted transposons. This set was partitioned into single-linkage clusters by flanking invrep sequence similarity using the `seqcluster.pl` program described in Appendix A.

DNA sequence homology

From the transposon family data set, a set of canonical invrep sequences for each family was extracted. Each set was used to train an HMM, which was then searched against the available worm DNA in order to obtain comprehensive data

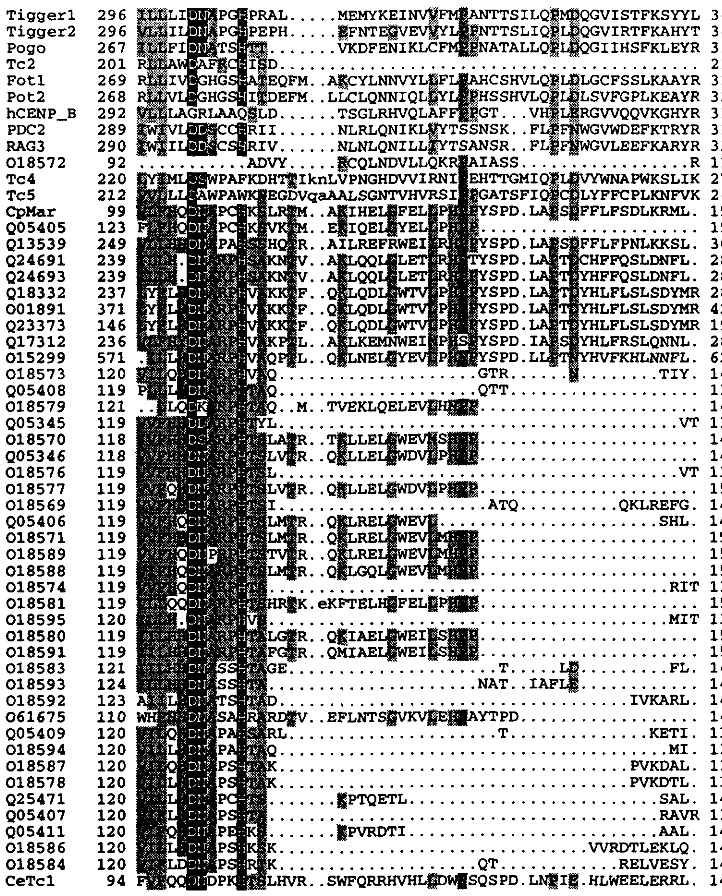


Figure 7.2: Alignment of D35E motifs from [DDJH94] and similar proteins from SP-TREMBL.

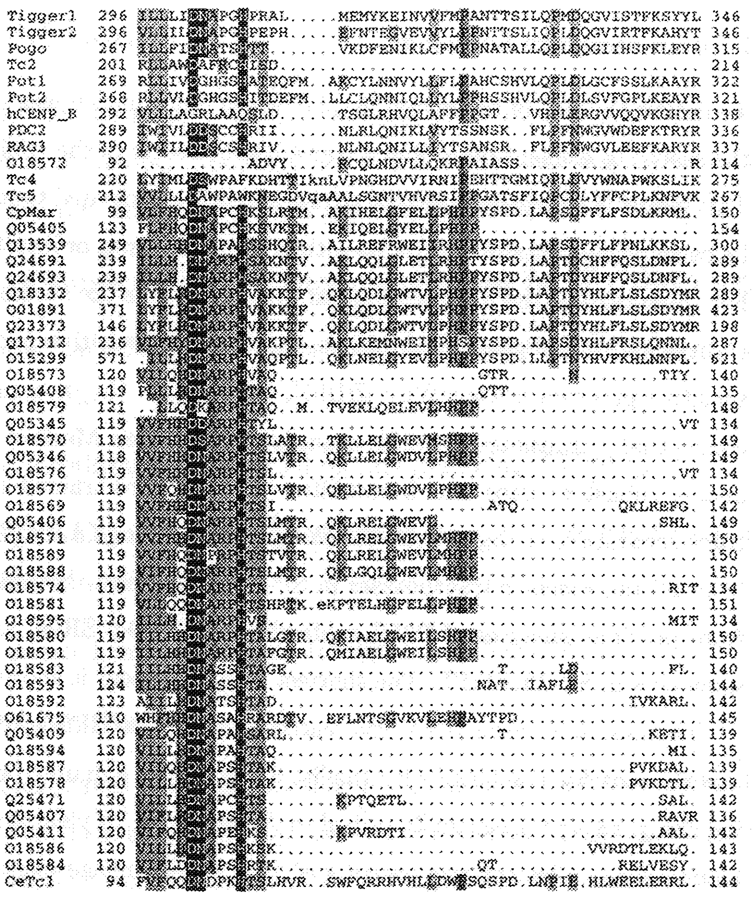


Figure 7.2: Alignment of D35E motifs from [DDJH94] and similar proteins from SP-TREMBL.

on the representation of each family in the worm genome, including instances where one half of the invrep had been deleted. HMMs trained on sequences from previously described transposon families (including Tc1-Tc7 [PvL97, RvLDP97] and Cele1-Cele7 [OGB95]) were also searched against the worm genome. Invreps were paired together and associated with GeneWise-predicted transposase genes using the GFF dynamic programming software `gffdp.pl` described in Appendix A.

7.3.2 Analysis of the transposon family data set

For each autonomous transposon family, a multiple alignment of the predicted transposase genes was made using CLUSTALW. These alignments were phylogenetically analysed by the UPGMA method using BELVU [SD94].

7.4 Results

7.4.1 Previously characterised transposon families

Table 7.1 lists the results of searching the *C.elegans* DNA with HMMs constructed from inverted repeat sequences typical to known transposon families Tc1-Tc7 [PvL97, RvLDP97], Cele1-Cele7 [OGB95] and Cele11-Cele14 [OGB96].

It is interesting to compare the results of this computational analysis with the element counts predicted from experimental data [PvL97]. The only element whose count is lower than experimentally predicted is Tc1. If a direct `blastn` search is performed using the Tc1 sequence as a query, the higher, predicted count is obtained. A possible explanation for this is that the "missing" Tc1 sequences do not fit the pattern of transposase homology flanked by invreps. Closer inspection reveals this to be so: in most cases, one of the invrep sequences is missing and in one case (*C.elegans* cosmid T10B5, invreps start at 37632) both invreps are present but do not flank the (usually) internal sequence (adjacent at position 39465).

Name	Example invrep sequence	Copies		Typical length	
		Pairs	Single invreps	Invrep only	Whole element
Tc1 (‡)	TACAGTGCTGGCCAAAAAGA...	25	10	81	1620
Tc2 (†♠)	CCGTATATTCTCTATTAGTG...	49	108	24	120
Tc3 (†♣)	TACAGTGTGGGAAAGTTCTA...	28	21	469	2350
Tc4 (†§)	CTAGGGAATGACCAGAATAA...	20	6	139	1610
Tc5 (¶)	CAAGGGAAGTCAAAAAACTG...	50	29	137	640
Tc6	CAGTGCTCCACATAATGATA...	22	886	656	1610
Tc7	TACAGTGCTGGCCAAAAAGA...	54	67	346	930
Cele1	CAAAATATCTCGTAGCGAAA...	73	280	36	230
Cele2	TACCHGGTCTCGACACGACA...	141	464	85	260
Cele4	TGGGTCTCGTTAGGTATTHG...	43	163	37	150
Cele5	GGTCTCGAAACGAYYGAAAY...	5	37	37	200
Cele6	TATTAMGRRAHCAHNARWTC...	19	42	32	150
Cele7	TAGTGHNAAANTATAGAAAA...	33	83	66	150
Cele14 (†)	CACGTGGAGTCAAAAAGTCC...	669	1095	36	180

Table 7.1: Previously characterised transposon families in the worm genome. Notes: (†) more copies than predicted [PvL97, OGB96]; (‡) 22 of the pairs enclose a transposase with 2 exons, lengths 155/875bp; (♠) includes 3 Cele11 and 32 Cele12 elements described in [OGB96], `blastn` searches reveal an additional 9 Cele11 and 7 Cele12 elements (approx.); (♣) 14 pairs enclose a transposase with 2 exons, lengths 416/572bp, 3 pairs form a putative nonautonomous subfamily, the rest appear internally heterogenous; (§) includes 4 copies of the putatively autonomous element Tc4v (3kb long); (¶) includes 20 copies of 1400bp and 25 copies of 600bp variants described in [OGB96], only 4 copies are "genuine" Tc5.

In all other cases, the database searches find about as many copies as experimentally predicted, with the exceptions of Tc2, Tc3, Tc5 and Cele14, whose copy numbers are elevated. In the former three cases this is due to the presence of putatively nonautonomous families sharing homology with the named families in the terminal regions of the flanking inverted repeats. The families associated with Tc2 and Tc5 have been previously described [OGB96], but the Tc3-associated family is new. Tc3 has been predicted to occur approximately 15 times in the Bristol N2 strain of *C.elegans* [CFA89] and 14 transposase-carrying copies are indeed found in this search; however, this only accounts for half the paired hits to the invrep HMM. Three of the remaining 14 pairs were found to share strong (over 90%) internal sequence identity, forming a new family of 1400bp proposed Tc3-hitchhikers with 574bp invreps, the terminal 247bp of which are similar to the Tc3 invrep. No strong internal similarity between the other Tc3-like elements was found.

The number of copies of the Cele14 invrep is an order of magnitude greater than predicted in [OGB96], probably because of the increased sensitivity of an HMM-based search over a BLAST search.

7.4.2 Previously uncharacterised transposon families

The search procedure described in 7.3.1 revealed six new Tc1/*mariner*-like families of transposon, named Tc11-Tc16 (this continues the Tc naming convention but leaves Tc8-Tc10 unused, allowing for independent transposon discoveries). Tc11-Tc16 contain coding sequences homologous to the *mariner* transposase flanked by characteristic inverted repeats. The definition of a family that was used - a group of transposons with near-identical invrep sequences - was supported by the phylogeny of the genes bracketed by these invreps, which clustered in the same way as the invrep sequences. Representation data for these transposons are listed in Table 7.2.

The exon structure of the predicted Tc11-Tc16 transposase genes in *C.elegans*

Name	Example invrep sequence	Copies		Typical length	
		Pairs (coding)	Single invreps	Invrep only	Whole element
Tc11 (†)	`TATTAGGTTGAACCGGAAGT...`	24 (11)	14	34	1230
Tc12 (‡)	`TATTAGGTTGGTCGAAAAGT...`	36 (19)	17	34	1250
Tc13 (♠)	`TATCAGGTCGTCCCATAAGT...`	59 (33)	16	34	1240
Tc14	`TACAGGGTGAGTCAAAATTA...`	12 (6)	47	30	1290
Tc15	`CTCGGCAATTCGTATCGTAC...`	4 (1)	7	40	1110
Tc16	`TATTAGGTTGTGAAAAAAGT...`	4 (2)	2	33	1260

Table 7.2: Previously uncharacterised Tc1/*mariner*-like transposon families in the worm genome. The numbers of coding-sequence containing pairs shown in brackets in the third column are based on the conservative *C.elegans* annotation rather than the Genewise predictions. Notes: (†) invrep similar to Tc13 and very similar to Tc12; (‡) invrep similar to Tc13 and very similar to Tc11; (♠) invrep similar to Tc11 and Tc12, tree suggests recent dispersion.

varies. The tranposase is most often predicted as a single exon over 1000bp long. None of these families has been characterised in the literature, although Oosumi *et al* found several copies of Tc13 after a `blastn` search with Cele14 as a probe [OGB96].

The chromosomal loci of the members of the transposon families listed in Tables 7.1 and 7.2 are published on the Wormdup website at the following URL:

`http://www.sanger.ac.uk/Users/ihh/Wormdup/`

7.4.3 Variation between transposon families

Transposon families Tc11-Tc13 display considerable similarity in their invrep sequences. This contrasts with previously described transposon families in *C.elegans*, which form distinct groups whether clustered by invrep or by internal sequence similarity. In particular the terminal 6 bases of the Tc11, Tc12, Tc13 and Tc16 invreps are almost completely conserved. These bases are thought to be important for catalysis in Tc1 [VP94].

Figure 7.3 shows an alignment of representative transposase proteins from the Tc1, Tc3 and Tc11-Tc15 families (the Tc16 coding sequence did not align

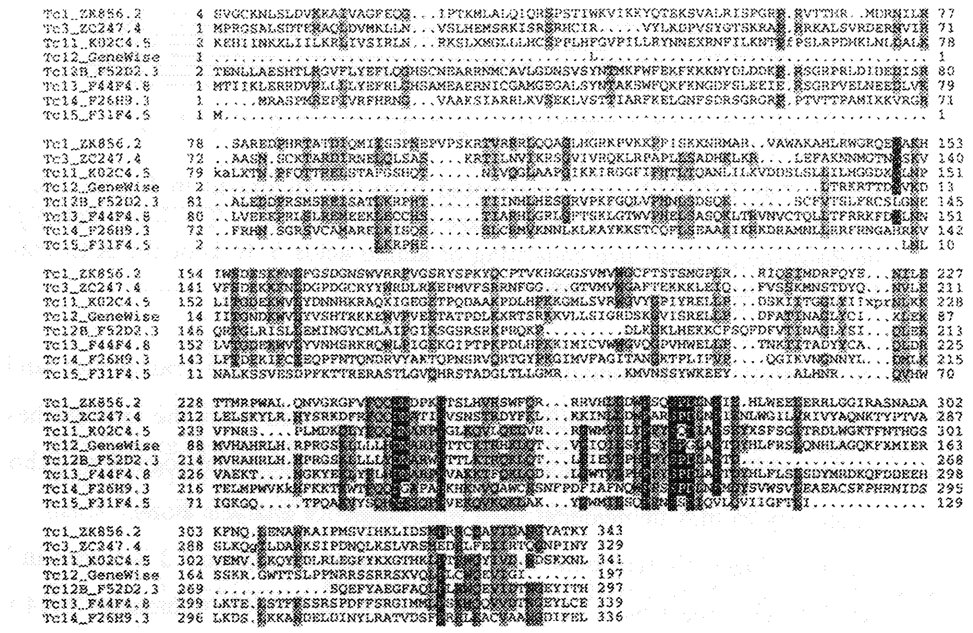


Figure 7.3: Alignment of transposase proteins from the Tc1, Tc3 and Tc11-Tc15 families. This alignment was constructed by removing redundant sequences from a multiple alignment of all predicted tranposase genes in *C.elegans* produced by CLUSTALW [THG94a]. Two distinct variants of the Tc12 transposase (Tc12 and Tc12B) are included. Residues 1 to 63 of Tc1 contain the *paired* DNA-binding domain, residues 71 to 207 contain the non-specific DNA-binding domain and residues 247-282 contain the D35E domain. Note that the Tc12 transposase gene was predicted using a version of GeneWise that does not extend the gene prediction beyond the optimal match to the HMM [BD97].

well to this set and was excluded). Two variants of the Tc12 protein appear (labelled Tc12 and Tc12B), as it was observed that this family forms two distinct subgroups when clustered by coding sequence. Where possible, the gene predictions from the ACeDB annotation were used in preference to the GeneWise predictions, as they tended to be more complete (GeneWise currently does not predict exons outside the region of homology).

Although the proteins are divergent, with the closest neighbours (the two Tc12 transposases) sharing under 50% sequence identity, they are also clearly homologous. Furthermore, the sequence homology is considerably greater over

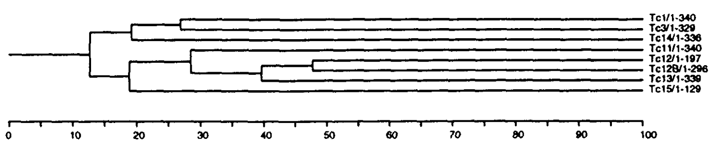


Figure 7.4: UPGMA tree constructed from the alignment in Figure 7.3. The horizontal scale marks out percentage sequence identity. The label TcN.X/Y-Z denotes the subsequence consisting of amino acids Y to Z of the X'th copy of the TcN transposase protein.

the catalytic D35E domain. This is consistent with the transposon families sharing a catalytic mechanism. It is also consistent with the families having evolved different invrep sequences for specific transposase recognition, although specificity cannot be demonstrated from sequence analysis alone.

A phylogenetic tree built from the alignment in Figure 7.3 is shown in Figure 7.4. The tree groups Tc15 with Tc11-Tc13 and (more tenuously) Tc14 with Tc1/Tc3.

7.4.4 Variation within transposon families

The Tc1, Tc3, Tc11, Tc12 and Tc13 families are sufficiently numerous that the intra-family variation - that is, the variation between coding sequences for members of the same family - can also be analysed (Figures 7.5 and 7.6). Several interesting points emerge from a study of these trees. All of the trees tend to be skewed, favouring the view that most new duplicates of an element are inactive, doomed to accumulate mutations while transposition is dominated by a few active copies [YWB97, HLL97, LC97, HLNL97]. The short branch lengths of the Tc1, Tc3 and Tc13 trees are evidence that these elements have been active in recent history (indeed, Tc1 and Tc3 are known to be currently active in the Bristol N2 strain [PvL97]), whereas the longer branch lengths of Tc11 and Tc12 suggest that they ceased activity earlier. It can also be seen that Tc12B subgroup of Tc12 transposases form a distinct group, as mentioned above.

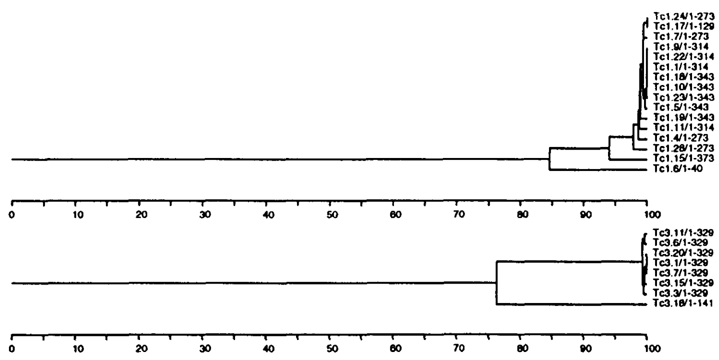


Figure 7.5: UPGMA trees constructed from alignments of the Tcl (top) and Tc3 (bottom) transposases. The horizontal scale marks out percentage sequence identity. The label TcN.X/Y-Z denotes the subsequence consisting of amino acids Y to Z of the X'th copy of the TcN transposase protein.

7.4.5 Location of transposons within the *C.elegans* genome

The distribution of transposons within the genome is of interest, not only because the insertion of a transposon into a gene or regulatory sequence can disrupt its function [Wes89, OGB95, BHP89], but also because the presence of transposons has been suggested to precipitate meiotic recombination [NCC$^+$92, YWB97]. The present study finds no evidence that the chromosomal location of a transposon is correlated with that of its nearest intra-familial relative. However, significant numbers of transposons were found within coding sequences and 5' upstream regions (Table 7.3). The high number of Tcl and Tcl3 elements overlapping with exons may be due to mispredicted genes in the *C.elegans* database.

The total fraction of transposons in or near coding sequences (68%) is higher than the proportion expected by chance (55%). DNA transposons thus display a clear preference for coding sequence in their choice of integration site.

Different types of repeats are often found to be associated together [Jur, PvL97]. As part of the preliminary screen for repetitive elements, the `gfffilter.pl` and `gffintersect.pl` programs (Appendix A) were used to find

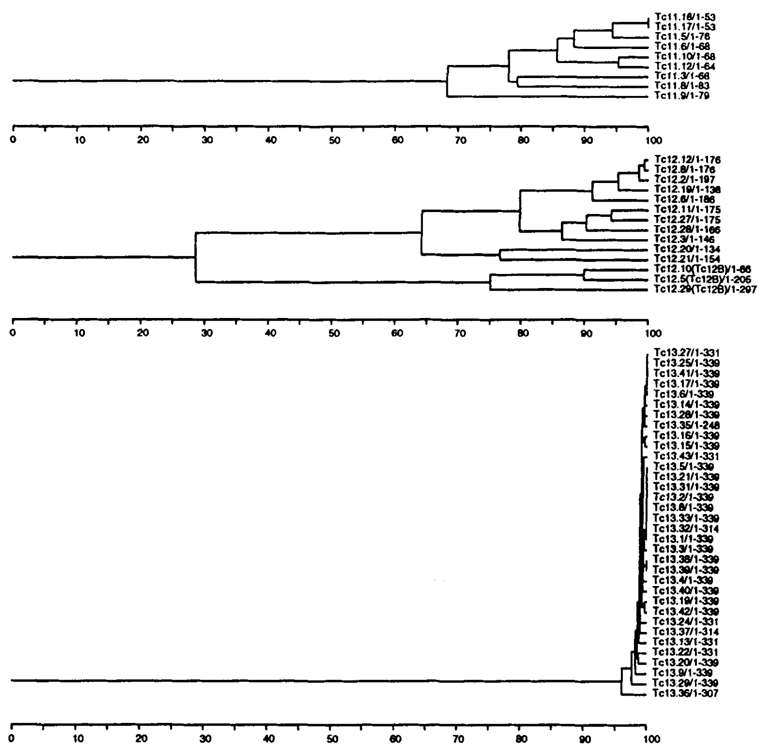


Figure 7.6: UPGMA trees constructed from alignments of the Tc11 (top), Tc12 (middle) and Tc13 (bottom) transposases. The horizontal scale marks out percentage sequence identity. The label TcN.X/Y-Z denotes the subsequence consisting of amino acids Y to Z of the X'th copy of the TcN transposase protein.

Transposon family	Copies in exons	Copies in introns	Copies in 1kb 5' regions
Tc1	5 (15%)	2 (6%)	19 (59%)
Tc2	0 (0%)	15 (37%)	8 (20%)
Tc3	1 (5%)	2 (10%)	12 (60%)
Tc4	0 (0%)	4 (28%)	3 (21%)
Tc5	0 (0%)	12 (30%)	14 (35%)
Tc6	1 (6%)	2 (12%)	1 (6%)
Tc7	1 (2%)	11 (26%)	7 (16%)
Cele1	1 (2%)	16 (38%)	10 (23%)
Cele2	1 (0%)	48 (42%)	37 (32%)
Cele4	0 (0%)	16 (53%)	6 (20%)
Cele5	0 (0%)	3 (60%)	2 (40%)
Cele6	1 (6%)	5 (31%)	3 (18%)
Cele7	2 (8%)	13 (54%)	7 (29%)
Cele11	0 (0%)	4 (8%)	16 (32%)
Cele12	1 (2%)	3 (7%)	10 (25%)
Cele14	16 (3%)	190 (37%)	133 (26%)
Tc11	3 (14%)	2 (9%)	14 (66%)
Tc12	4 (13%)	1 (3%)	19 (63%)
Tc13	13 (30%)	0 (0%)	24 (55%)
Tc14	2 (22%)	0 (0%)	6 (66%)
Tc15	1 (25%)	1 (25%)	2 (50%)
Tc16	1 (33%)	0 (0%)	2 (66%)

Table 7.3: Proximity of transposon families to coding sequence. The percentages in brackets indicate the fraction of the total copy number in each category.

the propensities of different repeats to associate with one another. An *association score* $\log\left[\frac{f_{xy}f}{f_x f_y}\right]$ (where f_{xy} is the frequency with which repeat x is associated with repeat y, f_x is the frequency with which x is associated with any other repeat and f is the total number of associations) was calculated for every pair of repeats x and y; some pairs of repeats with scores over 10 bits are listed in Table 7.4. There are clear clusters of repeats that are often found together, for example CeRep43, CeRep34 and CeRep23. These association propensities may indicate co-dependencies or similiarities in the mechanisms or preferred sites of integration.

7.5 Discussion

An exhaustive list of the chromosomal loci of all known DNA transposons in the Bristol N2 strain of *Caenhorabditis elegans* has been published on the Internet. In general DNA transposons display a clear preference for gene-proximal sequence in their choice of integration site. Statistical patterns of association between different classes of repetitive element have also been demonstrated. For example, 30% of Cele11 repeats are found to be near a copy of Tc5; and CeRep34, CeRep23 and CeRep43 are often found together. These association patterns may be indicative of similarities in the mechanisms of transposition.

A search using hidden Markov models has revealed putative new families of autonomous DNA transposon and one new subgroup of Tc3 elements in the *C.elegans* genome. Phylogenetic evidence suggests recent activity on behalf of one of the new families. The existence of several distinct species of transposon in the same genome with such striking homology between their flanking sequences has implications for the study of transposon ecology and evolution. There are several known mechanisms by which transposons could competitively interact. Transposase proteins of other members of the Tc1/*mariner* family bind specifically to the invrep sequences of transposons of that family *in vitro* [PvL97]. Furthermore, excessive expression of Tc1 transposase protein induces the phe-

Repeat type	Associated repeats (association score/bits)
CeRep10	Cele2 (11.6), CeRep14 (10.6), CeRep11 (10.4), CeRep37 (10.2)
CeRep11	Cele4 (11.1), CeRep10 (10.4)
CeRep12	CeRep13 (11.4)
CeRep13	CeRep18 (12.2), CeRep12 (11.4), CeRep30 (10.6), CeRep33 (10.4)
CeRep14	CeRep10 (10.6), Cele1 (10.4)
CeRep15	Cele7 (11.1)
CeRep17	CeRep19 (12.1), CeRep32 (11.9)
CeRep18	CeRep13 (12.2), CeRep33 (11.1), CeRep30 (11)
CeRep19	CeRep32 (12.2), CeRep17 (12.1)
CeRep22	CeRep37 (11)
CeRep23	CeRep34 (11.8), CeRep43 (11.8)
CeRep24	CeRep38 (12.5), Cele14 (12)
CeRep29	CeRep36 (12.7), CeRep35 (11)
CeRep30	CeRep18 (11), CeRep13 (10.6)
CeRep32	CeRep19 (12.2), CeRep17 (11.9)
CeRep33	CeRep18 (11.1), CeRep13 (10.4)
CeRep34	CeRep43 (12.4), CeRep23 (11.8)
CeRep35	CeRep36 (11.1), CeRep29 (11), CeRep40 (11)
CeRep36	CeRep29 (12.7), CeRep35 (11.1)
CeRep37	CeRep22 (11), CeRep10 (10.2)
CeRep38	CeRep24 (12.5), Cele14 (11.4)
CeRep40	CeRep35 (11)
CeRep41	Tc3 (11.8)
CeRep43	CeRep34 (12.4), CeRep23 (11.8)
Cele1	CeRep14 (10.4)
Cele2	CeRep10 (11.6)
Cele4	CeRep11 (11.1)
Cele7	CeRep15 (11.1)
Cele11	Tc5 (10.5)
Cele14	CeRep24 (12), CeRep38 (11.4)
Tc3	CeRep41 (11.8)
Tc5	Cele11 (10.5)

Table 7.4: Propensities for *C. elegans* repeat types to be found within 1kb of each other. The association scores in brackets are logs of the odds-ratios $\frac{f_{xy} f}{f_x f_y}$ where f_{xy} is the frequency of association of x and y, f_x is the number of associations for x and f is the total number of associations for everything. Only association scores over 10 bits are reported.

nomenon of "overproduction inhibition", reducing transpositional activity in what arguably functions as a regulatory negative-feedback mechanism [HLL97]. It has also been observed that missense mutations in the *mariner*-like MOS1 transposase gene have a dominant-negative effect; the "poisoning" of transposase oligomeric complexes by inactive subunits has been proposed as a mechanism to explain this [HLNL97]. All these mechanisms may work together with host-specific mechanisms to regulate transpositional activity [LC97, HLNL97]. The discovery of dormant *mariner* subfamilies with slight variations in their putative DNA-binding domains and transposase-binding nucleotide sequences may offer new opportunities to study the evolution of DNA-protein specificity in transposon ecology.

Chapter 8

Conclusion

The evolution of genetic material is fundamental to the whole of biology; the sequencing and analysis of whole genomes has begun a revolution in scientific understanding. As we gather more and more data about how cellular processes work, we will need more powerful software agents to cluster, filter, organise and digest the information so that we can get on with the creative process of interpreting, describing and acting upon it. Designing these tools - and being the first to use them - is what bioinformatics is all about.

This thesis has presented work on the theory of biological sequence alignment and the evolution of the first sequenced animal genome, *Caenorhabditis elegans*. The sequence alignment theory has looked at a number of issues, addressing the question of accuracy - how accurate can an alignment be? - and discussing how to side-step this issue by summing over all alignments (and over a range of scoring schemes). The evolutionary investigations have looked at questions associated with genomic duplication in the nematode worm *C.elegans*, including causes and characteristics of duplications and the random divergence of sequences following duplication. The two approaches have informed each other closely: e.g. the measurement of substitution and indel rates motivated the development of parameter-estimation algorithms and the development of a Bayesian framework for model comparison enabled the evaluation of a new molecular clock based on introns.

The fusion of ideas from computer science and molecular biology is one of the things that make bioinformatics an exciting field. The development of the Bayesian view of sequence alignment is a shining example of this fusion. Sequence alignment is the king pin of homology analysis which, at a structural, functional and evolutionary level, shapes our understanding of protein families and the patterns of process that nature has used. With the data that fund this analysis multiplying rapidly and the study of protein families blossoming, now is the right time to build solid foundations for sequence analysis so that issues like uncertainty and accuracy are not awkward embarassments but robust parameters of the theory. Bayesian statistics, with its concept of a probability as a level of belief in a hypothesis, is an ideal framework in which this process can work. It is hoped that this dissertation has pointed out some of the fronts on which progress can be made. It has been said that Biologists stole Statistics from Physicists [Jay86]; they now have a chance to steal it again.

Flourishing technologies such as expression analysis by microarrays and ESTs provide yet more opportunities. Computational biology has an important role as interpreter for data - such as these - that are so voluminous they can only be visualised at a statistical remove. The basic kinds of operation that one tends to want to do on these data include clustering, pattern identification, construction of models for these patterns, classification of the data according to these models, construction of new models and collection of new data - the iterative loop of refining models and, behind the models, biological ideas. The particular algorithms for analysing each class of data will be different; finding the right approach demands mathematical skill and intuition together with a broad biological awareness so that the practical issues may be separated from the mathematical curiosities. However, bioinformatics has matured and is ready for the challenge. A network of computational biologists exists, with the background and the abilities to respond to these kinds of technological advances.

One of the things that can drive new algorithm development is a convincing format or specification. As an example, the GFF processing tools developed for this project would probably have remained a collection of throwaway scripts were it not for the stabilisation of a good, simple format that not only represented annotative data well, but was seen to gain enough support in the bioinformatics community that it seemed inevitable that it would catch on. This has a lot in common with the history of HMMs and Bayesian methods in sequence analysis. With the increasing heterogeneity of data and the diversity of sequenced animal genomes, standardisation will tend to become the path of least resistance as organisations start to seek common ways of handling and sharing their data without wasting time on different protocols for each organism or experiment.

This brings us to another exciting aspect of bioinformatics which is, of course, its proximity to the genome projects. As more complete animal genomes become available we may hope to move beyond individual, anecdotal accounts of positive selection or selective sweep and towards an understanding of evolutionary dynamics that may be increasingly quantitative. The ecology of transposons and viral elements and their role in stimulating animal evolution is a fascinating topic that has led to much speculation. This speculation is now confronting hard data produced by the sequencing effort. Soon we can hope to start to develop a rounded account of evolutionary history that speaks directly of molecular mechanisms. This will be a major humanistic triumph and a direct consequence of the genome projects.

Bibliography

[AB94] T. K. Attwood and M. E. Beck. PRINTS – a protein motif fingerprint database. *Protein Engineering*, 7:841–848, 1994.

[AF94] T. K. Attwood and J. B. Findlay. Fingerprinting G-protein-coupled receptors. *Protein Engineering*, 7:195–203, 1994.

[AG96] S. F. Altschul and W. Gish. Local alignment statistics. *Methods in Enzymology*, 266:460–480, 1996.

[AGM+90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[AKW+98] N. R. Ashcroft, M. E. Kosinski, D. Wickramasinghe, P. J. Donovan, and A. Golden. The four cdc25 genes from the nematode Caenorhabditis elegans. *Gene*, 214:59–66, 1998.

[AMS+97] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

[BB98] M. L. Beçak and W. Beçak. Evolution by polyploidy in Amphibia: new insights. *Cytogenetic Cell Genetics*, 80:28–33, 1998.

[BC94] P. Baldi and Y. Chauvin. Smooth on-line learning algorithms for hidden Markov models. *Neural Computation*, 6(2):307–318, 1994.

[BD97] E. Birney and R. Durbin. Dynamite: a flexible code generating language for dynamic programming methods used in sequence comparison. In Gaasterland et al. [GKK$^+$97], pages 56–64.

[BH96] P. Bucher and K. Hofmann. A sequence similarity search algorithm based on a probabilistic interpretation of an alignment scoring system. In States et al. [SAG$^+$96], pages 44–51.

[BH97] L. Bloom and H. R. Horvitz. The Caenorhabditis elegans gene unc-76 and its human homologs define a new gene family involved in axonal outgrowth and fasciculation. *Proceedings of the National Academy of Sciences of the USA*, 94:3414–3419, 1997.

[BHK$^+$93] M. Brown, R. Hughey, A. Krogh, I. S. Mian, K. Sjölander, and D. Haussler. Using Dirichlet mixture priors to derive hidden Markov models for protein families. In L. Hunter, D. B. Searls, and J. Shavlik, editors, *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology*, pages 47–55, Menlo Park, CA, 1993. AAAI Press.

[BHP89] K. Banki, D. Halladay, and A. Perl. Cloning and expression of the human gene for transaldolase: A novel highly repetitive element constitutes an integral part of the coding sequence. *Journal of Biological Chemistry*, 269:2847–2851, 1989.

[Bir] Ewan Birney. Personal communication.

[Bis95] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, UK, 1995.

[BJVU98] A. Brazma, I. Jonassen, J. Vilo, and E. Ukkonen. Predicting gene regulatory elements in silico on a genomic scale. Preprint, to appear in Genome Research, 1998.

[Bla98] M. Blaxter. Caenorhabditis elegans is a nematode. *Science*, 282:2041–2046, 1998.

[Blu98] T. Blumenthal. Gene clusters and polycistronic transcription in eukaryotes. *Bioessays*, 20:480–487, 1998.

[BPS98] C. B. Burge, R. A. Padgett, and P. A. Sharp. Evolutionary fates and origins of U12-type introns. *Molecular Cell*, 2:1–20, 1998.

[BR93] T. R. Burglin and G. Ruvkun. The Caenorhabditis elegans homeobox gene cluster. *Current Opinion in Genetics and Development*, 3:615–620, 1993.

[BS87] G. J. Barton and M. J. E. Sternberg. A strategy for the rapid multiple alignment of protein sequences. *Journal of Molecular Biology*, 198:327–337, 1987.

[BS97] T. Blumenthal and K. Steward. RNA processing and gene structure. In D. Riddle, T. Blumenthal, B. Meyer, and J. Priess, editors, *C.elegans II*, chapter 6, pages 117–145. Cold Spring Harbor Laboratory Press, Plainview, NY, 1997.

[BTR98] C. Brown, K. Todd, and R. F. Rosenzweig. Multiple duplications of yeast hexose transport genes in response to selection in a glucose-limited environment. *Molecular Biology and Evolution*, 15:931–942, 1998.

[Bul86] M. Bulmer. Neighbouring base effects on substitution rates in pseudogenes. *Molecular Biology and Evolution*, 3:322–329, 1986.

[Bur98] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. Available from http://svm.research.bell-labs.com/SVMdoc.html, 1998.

[But98] B. A. Butler. Sequence analysis using GCG. *Methods of Biochemical Analysis*, 39:74–97, 1998.

[CFA89] J. Collins, E. Forbes, and P. Anderson. The Tc3 family of transposable elements in Caenorhabditis elegans. *Nature*, 328:726–728, 1989.

[Cho92] C. Chothia. One thousand families for the molecular biologist. *Nature*, 357:543–544, 1992.

[CK98] M. Cline and K. Karplus. On alignment shift and its measures. Technical report, University of California Santa Cruz, 1998. Available from http://www.cse.ucsc.edu/~cline/shiftscore.html.

[CL88] H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal of Applied Mathematics*, 48:1073–1082, 1988.

[Cla98] M. Clamp. Jalview. In press, 1998.

[CLB93] R. Conrad, R. Liou, and T. Blumenthal. Functional analysis of a C.elegans trans-splice acceptor. *Nucleic Acids Research*, 21(4):913–919, 1993.

[Cra95] N. Craig. Unity in transposition reactions. *Science*, 270:253–254, 1995.

[CSC98] The C.elegans Sequencing Consortium. Genome sequence of the nematode Caenorhabditis elegans. A platform for investigating biology. *Science*, 282:2012–2018, 1998.

[CvLP94] S. D. Colloms, H. G. van Luenen, and R. H. Plasterk. DNA binding activities of the Caenorhabditis elegans Tc3 transposase. *Nucleic Acids Research*, 22:5548–5554, 1994.

[DDJH94] T. Doak, F. Doerder, C. Jahn, and G. Herrick. A proposed superfamily of transposase genes: transposon-like elements in ciliated protozoa and a common "D35E" motif. *Proceedings of the National Academy of Sciences of the USA*, 91:942–946, 1994.

[DEKM98] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, Cambridge, UK, 1998.

[DGPB98] L. Duret, N. Guex, M. C. Peitsch, and A. Bairoch. New insulin-like proteins with atypical disulfide bond pattern characterized in Caenorhabditis elegans by comparative sequence analysis and homology modeling. *Genome Research*, 8:348–353, 1998.

[DH97] R. E. Davis and S. Hodgson. Gene linkage and steady state RNAs suggest trans-splicing may be associated with a polycistronic transcript in Schistosoma mansoni. *Molecular and Biochemical Parasitology*, 89:25–39, 1997.

[DHL97a] D. Drasdo, T. Hwa, and M. Lässig. DNA sequence alignment and critical phenomena. Available from http://matisse.ucsd.edu/~hwa/pub.html, 1997.

[DHL97b] D. Drasdo, T. Hwa, and M. Lässig. Scaling laws and similarity detection in sequence alignment with gaps. Preprint, available from http://matisse.ucsd.edu/~hwa/pub.html, 1997.

[DIB97] J. L. DeRisi, V. R. Iyer, and P. O. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278:680–686, 1997.

[DSO78] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. In M. O. Dayhoff, editor, *Atlas of Protein Sequence and Structure*, volume 5, supplement 3, pages 345–352. National Biomedical Research Foundation, Washington, DC, 1978.

[ED95] F. H. Eeckman and R. Durbin. ACeDB and Macace. *Methods in Cell Biology*, 48:583–605, 1995.

[Edd95] S. R. Eddy. Multiple alignment using hidden Markov models. In Rawlings et al. [RCA$^+$95], pages 114–120.

[Edd96] S. R. Eddy. Hidden Markov models. *Current Opinion in Structural Biology*, 6:361–365, 1996.

[Eis98] J. A. Eisen. A phylogenomic study of the MutS family of proteins. *Nucleic Acids Research*, 26(18):4291–4300, 1998.

[EMD95] S. R. Eddy, G. J. Mitchison, and R. Durbin. Maximum discrimination hidden Markov models of sequence consensus. *Journal of Computational Biology*, 2:9–23, 1995.

[EZM$^+$97] D. Evans, D. Zorio, M. MacMorris, C. E. Winter, K. Lea, and T. Blumenthal. Operons and SL2 trans-splicing exist in nematodes outside the genus Caenorhabditis. *Proceedings of the National Academy of Sciences of the USA*, 94:9751–9756, 1997.

[FAW$^+$95] R. D. Fleischmann, M. D. Adams, O. White, R. A. Clayton, E. F. Kirkness, A. R. Kerlavage, C. J. Bult, J. F. Tomb, B. A. Dougherty, J. M. Merrick, et al. Whole-genome random sequencing and assembly of Haemophilus influenzae Rd. *Science*, 269:496–512, 1995.

[FBT$^+$91] D. Fitch, W. Bailey, D. Tagle, M. Goodman, L. Sieu, and J. Slightom. Duplication of the γ-globin gene mediated by L1 long interspersed repetitive elements in an early ancestor of simian primates. *Proceedings of the National Academy of Sciences of the USA*, 88:7396–7400, 1991.

[FD87] D.-F. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25:351–360, 1987.

[FLD$^+$94] G. Franz, T. G. Loukeris, G. Dialektaki, C. R. Thompson, and C. Savakis. Mobile Minos elements from Drosophila hydei encode a two-exon transposase with similarity to the paired DNA-binding domain. *Proceedings of the National Academy of Sciences of the USA*, 91:4746–4750, 1994.

[GFF] GFF: an exchange format for gene-finding features. Webpage at http://www.sanger.ac.uk/Software/GFF/.

[GKK$^+$97] T. Gaasterland, P. Karp, K. Karplus, C. Ouzounis, C. Sander, and A. Valencia, editors. *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, Menlo Park, CA, 1997. AAAI Press.

[GL95a] N. Grindley and A. Leschziner. DNA transposition: from a black box to a color monitor. *Cell*, 83:1063–1066, 1995.

[GL95b] X. Gu and W-H. Li. The size distribution of insertions and deletions in human and rodent pseudogenes suggests the logarithmic gap penalty for sequence alignment. *Journal of Molecular Evolution*, 40:464–473, 1995.

[Gol98] N. Goldman. Phylogeny with alignment uncertainty. Preprint, Isaac Newton Institute for the Mathematical Sciences, Univ. of Cambridge, UK, 1998.

[Got82] O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, 162:705–708, 1982.

[Got96] O. Gotoh. Significant improvement in accuracy of multiple protein alignments by iterative refinement as assessed by reference to structural alignments. *Journal of Molecular Biology*, 264:823–838, 1996.

[GPL] The GNU Public License. Available in full from http://www.fsf.org/copyleft/gpl.html.

[GV96] M. Gribskov and S. Veretnik. Identification of sequence patterns with profile analysis. *Methods in Enzymology*, 266:198–212, 1996.

[GW92] P. Gruss and C. Walther. Pax in development. *Cell*, 69:719–722, 1992.

[GWD98] P. Gibbs, W. Witke, and A. Dugaiczyk. The molecular clock runs at different rates among closely related members of a gene family. *Journal of Molecular Evolution*, 46:552–561, 1998.

[Hau98] D. Haussler. Computational genefinding. Preprint; available from http://www.cse.ucsc.edu/research/compbio/research.html, 1998.

[HBF92] D. G. Higgins, A. J. Bleasby, and R. Fuchs. CLUSTAL V: improved software for multiple sequence alignment. *Computer Applications in the Biosciences*, 8(2):189–191, 1992.

[HD98] I. Holmes and R. Durbin. Dynamic programming alignment accuracy. *Journal of Computational Biology*, 5(3):493–504, 1998.

[HH91] S. Henikoff and J. G. Henikoff. Automated assembly of protein blocks for database searching. *Nucleic Acids Research*, 19:6565–6572, 1991.

[HH92] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the USA*, 89:10915–10919, 1992.

[HK94] D. S. Horowitz and A. R. Krainer. Mechanisms for selecting 5' splice sites in mammalian pre-mRNA splicing. *Trends in Genetics*, 10:100–106, 1994.

[HK96] R. Hughey and A. Krogh. Hidden Markov models for sequence analysis: extension and analysis of the basic method. *Computer Applications in the Biosciences*, 12:95–107, 1996.

[HKMS93] D. Haussler, A. Krogh, I. S. Mian, and K. Sjölander. Protein modeling using hidden Markov models: analysis of globins. In T. N. Mudge, V. Milutinovic, and L. Hunter, editors, *Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences*, volume 1, pages 792–802, Los Alamitos, CA, 1993. IEEE Computer Society Press.

[HKY85] M. Hasegawa, H. Kishino, and T. Yano. Dating the human-ape splitting by a molecular clock of mitochondrial DNA. *Journal of Molecular Evolution*, 22:160–174, 1985.

[HL96] T. Hwa and M. Lässig. Similarity-detection and localization. *Physical Review Letters*, 76:2591, 1996.

[HLL97] D. Hartl, A. Lohe, and E. Lozovskaya. Modern thoughts on an ancyent *Marinere*: function, evolution, regulation. *Annual Review of Genetics*, 31:337–358, 1997.

[HLNL97] D. Hartl, E. Lozovskaya, D. Nurminsky, and A. Lohe. What restricts the activity of *mariner*-like transposable elements? *Trends in Genetics*, 13(5):197–201, 1997.

[HS89] D. G. Higgins and P. M. Sharp. Fast and sensitive multiple sequence alignments on a microcomputer. *Computer Applications in the Biosciences*, 5:151–153, 1989.

[Hug98] A. L. Hughes. Phylogenetic tests of the hypothesis of block duplication of homologous genes on human chromosomes 6, 9 and 1. *Molecular Biology and Evolution*, 15:854–870, 1998.

[Hwa96] T. Hwa. From flux pinning to DNA sequence alignment. In K. Fujikawa and Y. A. Ono, editors, *Quantum Coherence and Decoherence*, pages 109–114. Elsevier, 1996.

[Jay86] E. T. Jaynes. Bayesian methods: general background. In J. H. Justice, editor, *Maximum Entropy and Bayesian Methods in Applied Statistics*, pages 1–25. Cambridge University Press, Cambridge, UK, 1986.

[JB97] R. C. Johnsen and D. L. Baillie. Mutation. In D. Riddle, T. Blumenthal, B. Meyer, and J. Priess, editors, *C.elegans II*, chapter 4, pages 79–95. Cold Spring Harbor Laboratory Press, Plainview, NY, 1997.

[JC69] T. H. Jukes and C. Cantor. Evolution of protein molecules. In *Mammalian Protein Metabolism*, pages 21–132. Academic Press, New York, 1969.

[JH98] T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. Preprint, Dept. of Computer Science, Univ. of California, available from http://www.cse.ucsc.edu/~haussler/pubs.html, 1998.

[Jur] J. Jurka. Personal communication.

[Jur98] J. Jurka. Repeats in genomic DNA: mining and meaning. *Current Opinion in Structural Biology*, 8:333–337, 1998.

[KA90] S. Karlin and S. F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences of the USA*, 87:2264–2268, 1990.

[KA93] S. Karlin and S. F. Altschul. Applications and statistics for multiple high-scoring segments in molecular sequences. *Proceedings of the National Academy of Sciences of the USA*, 90:5873–5877, 1993.

[Kab95] D. B. Kaback. Yeast genome structure. In A. E. Wheals, A. H. Rose, and J. S. Harrison, editors, *The Yeasts*, volume 6, pages 179–222. Academic, London, 1995.

[KB95] S. Karlin and C. Burge. Dinucleotide relative abundance extremes: a genomic signature. *Trends in Genetics*, 11:283–290, 1995.

[KBM$^+$94] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, Feb. 1994.

[KHRE96] D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman. A generalized hidden Markov model for the recognition of human genes in DNA. In States et al. [SAG$^+$96], pages 134–142.

[Kim80] M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16:111–120, 1980.

[Kim83] M. Kimura. *The Neutral Theory of Molecular Evolution.* Cambridge University Press, Cambridge, UK, 1983.

[KM95] A. Krogh and G. J. Mitchison. Maximum entropy weighting of aligned sequences of proteins or DNA. In Rawlings et al. [RCA$^+$95], pages 215–221.

[KMH94] A. Krogh, I. S. Mian, and D. Haussler. A hidden Markov model that finds genes in E. coli DNA. *Nucleic Acids Research*, 22:4768–4778, 1994.

[Kro94] A. Krogh. Hidden Markov models for labeled sequences. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, pages 140–144, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[KT75] S. Karlin and H. Taylor. *A First Course in Stochastic Processes.* Academic Press, San Diego, CA, 1975.

[LAB$^+$93] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262(5131):208–214, 1993.

[LAK89] D. J. Lipman, S. F. Altschul, and J. D. Kececioglu. A tool for multiple sequence alignment. *Proceedings of the National Academy of Sciences of the USA*, 86:4412–4415, 1989.

[LC95] M. R. Leroux and E. P. Candido. Characterization of four new tcp-1-related cct genes from the nematode Caenorhabditis elegans. *DNA and Cell Biology*, 14:951–60, 1995.

[LC97] M. Labrador and V. Corces. Transposable element-host interactions: regulation of insertion and excision. *Annual Review of Genetics*, 31:381–404, 1997.

[LCR96] D. Lampe, M. Churchill, and H. Robertson. Purified *mariner* transposase is sufficient to mediate transposition in *vitro*. *EMBO Journal*, 15:5470–5479, 1996.

[LG91] W-H. Li and D. Graur. *Fundamentals of Molecular Evolution*. Sinauer Associates, Sunderland, MA, 1991.

[LKW97] T. Lindahl, P. Karran, and R. D. Wood. DNA excision repair pathways. *Current Opinion in Genetics and Development*, 7:158–169, 1997.

[LL80] L. D. Landau and E. M. Lifshitz. *Statistical Physics Part I*, volume 5 of *Course of Theoretical Physics*. Pergamon Press, Oxford, UK, 1980.

[Mac92a] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4:415–447, 1992.

[Mac92b] D. J. C. MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, 1992.

[Mac96a] D. J. C. MacKay. Density networks and their application to protein modelling. In *Maximum Entropy and Bayesian Methods*, pages 259–268. Kluwer Academic Publishers, Netherlands, 1996.

[Mac96b] D. J. C. MacKay. Equivalence of Boltzmann chains and hidden Markov models. *Neural Computation*, 8(1):178–181, 1996.

[Mac97] D. J. C. MacKay. Introduction to Gaussian processes. Available from http://wol.ra.phy.cam.ac.uk/mackay/, 1997.

[MD95] G. J. Mitchison and R. Durbin. Tree-based maximal likelihood substitution matrices and hidden Markov models. *Journal of Molecular Evolution*, 41:1139–1151, 1995.

[Miy94] S. Miyazawa. A reliable sequence alignment method based on probabilities of residue correspondence. *Protein Engineering*, 8:999–1009, 1994.

[MKW91] D. G. Moerman, J. E. Kiff, and R. H. Waterston. Germline excision of the transposable element Tcl in C.elegans. *Nucleic Acids Research*, 19:5669–5672, 1991.

[MM88] E. W. Myers and W. Miller. Optimal alignments in linear space. *Computer Applications in the Biosciences*, 4(1):11–17, 1988.

[MV96] H. T. Mevissen and M. Vingron. Quantifying the local reliability of a sequence alignment. *Protein Engineering*, 9(2):127–132, 1996.

[MVF94] M. A. McClure, T. K. Vasi, and W. M. Fitch. Comparative analysis of multiple protein-sequence alignment methods. *Journal of Molecular Evolution*, 11:571–592, 1994.

[NCC$^+$92] G. Naclerio, G. Cangiano, A. Coulson, A. Levitt, V. Ruvolo, and A. La Volpe. Molecular and genomic organization of clusters of repetitive DNA sequences in Caenorhabditis elegans. *Journal of Molecular Biology*, 226:159–168, 1992.

[Nea] R. M. Neal. Personal communication.

[Nea96] R. M. Neal. *Bayesian Learning in Neural Networks.* Springer (Lecture Notes in Statistics), New York, Berlin, 1996.

[Nea98] R. M. Neal. Annealed importance sampling. Technical report no. 9805, Dept. of Statistics, Univ. of Toronto, available from http://www.cs.utoronto.ca/~radford/, 1998.

[NH93] R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. Preprint, Dept. of Computer

Science, Univ. of Toronto, available from ftp://archive.cis.ohio-state.edu/pub/neuroprose/neal.em.ps.Z, 1993.

[NH96] C. Notredame and D. G. Higgins. SAGA: sequence alignment by genetic algorithm. *Nucleic Acids Research*, 24(8):1515–1524, 1996.

[NKML98] L. S. Nelson, K. Kim, J. E. Memmott, and C. Li. FMRFamide-related gene family in the nematode, Caenorhabditis elegans. *Brain Research. Molecular Brain Research*, 58:103–111, 1998.

[NW70] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[OGB95] T. Oosumi, B. Garlick, and W. Belknap. Identification and characterization of putative transposable DNA elements in solanaceous plants and Caenorhabditis elegans. *Proceedings of the National Academy of Sciences of the USA*, 92:8886–8890, 1995.

[OGB96] T. Oosumi, B. Garlick, and W. Belknap. Identification of putative nonautonomous transposable elements associated with several transposon families in Caenorhabditis elegans. *Journal of Molecular Evolution*, 43:11–18, 1996.

[Ohn70] S. Ohno. *Evolution by gene duplication.* Springer Verlag, Berlin/Heidelberg/New York, 1970.

[Oht89] T. Ohta. Role of gene duplication in evolution. *Genome*, 31:304–310, 1989.

[Oht91] T. Ohta. Multigene families and the evolution of complexity. *Journal of Molecular Evolution*, 33:34–41, 1991.

[Ols91] M. V. Olson. Genome structure and organization in Saccharomyces cerevisiae. In J. R. Broach, J. R. Pringle, and E. W. Jones, editors, *The Molecular and Cellular Biology of the Yeast Saccharomyces*, volume 1, pages 1–40. Cold Spring Harbor Laboratory Press, New York, 1991.

[PBBC96] A. G. Pedersen, P. Baldi, S. Brunak, and Y. Chauvin. Characterization of prokaryotic and eukaryotic promoters using hidden Markov models. In States et al. [SAG$^+$96], pages 182–191.

[PL88] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the USA*, 4:2444–2448, 1988.

[Pov98] M. Povinelli. Density networks with application to protein modelling. M.Phil thesis, Univ. of Cambridge, UK. Available from http://www.mit.edu/~mpovinel/pub.html, 1998.

[PS98] V. E. Papaioannou and L. M. Silver. The T-box gene family. *Bioessays*, 20:9–19, 1998.

[PvL97] R. Plasterk and H. van Luenen. Transposons. In D. Riddle, T. Blumenthal, B. Meyer, and J. Priess, editors, *C.elegans II*, chapter 5, pages 97–116. Cold Spring Harbor Laboratory Press, Plainview, NY, 1997.

[Rab89] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.

[RCA$^+$95] C. Rawlings, D. Clark, R. Altman, L. Hunter, T. Lengauer, and S. Wodak, editors. *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, Menlo Park, CA, 1995. AAAI Press.

[REKH97] M. G. Reese, F. H. Eeckman, D. Kulp, and D. Haussler. Improved splice site detection in Genie. *Journal of Computational Biology*, 4(3):311–323, 1997.

[Rob98] H. M. Robertson. Two large families of chemoreceptor genes in the nematodes Caenorhabditis elegans and Caenorhabditis briggsae reveal extensive gene duplication, diversification, movement, and intron loss. *Genome Research*, 8:449–463, 1998.

[RvLDP97] R. Rezsohazy, H. van Luenen, R. Durbin, and R. Plasterk. Tc7, a Tc1-hitch hiking transposon in Caenorhabditis elegans. *Nucleic Acids Research*, 25(20):4048–4054, 1997.

[SAG+96] D. J. States, P. Agarwal, T. Gaasterland, L. Hunter, and R. F. Smith, editors. *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, Menlo Park, CA, 1996. AAAI Press.

[SAL+95] P. M. Sharp, M. Averof, A. T. Lloyd, G. Matassi, and J. F. Peden. DNA sequence evolution: the sounds of silence. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 349:241–247, 1995.

[SD94] E. Sonnhammer and R. Durbin. A workbench for large-scale sequence homology analysis. *Computer Applications in the Biosciences*, 10(3):301–307, 1994.

[SD97] E. L. Sonnhammer and R. Durbin. Analysis of protein domain families in Caenorhabditis elegans. *Genomics*, 46:200–216, 1997.

[SDF+96] C. Savage, P. Das, A. L. Finelli, S. R. Townsend, C. Y. Sun, S. E. Baird, and R. W. Padgett. Caenorhabditis elegans genes sma-2, sma-3, and sma-4 define a conserved family of transforming growth factor beta pathway components. *Proceedings of the National Academy of Sciences of the USA*, 93:790–794, 1996.

[SDT+92] J. Sulston, Z. Du, K. Thomas, R. Wilson, L. Hillier, R. Staden, N. Halloran, P. Green, J. Thierry-Mieg, L. Qiu, et al. The C.elegans genome sequencing project: a beginning. *Nature*, 356:37–41, 1992.

[SEB+98] E. Sonnhammer, S. Eddy, E. Birney, A. Bateman, and R. Durbin. Pfam: multiple sequence alignments and HMM-profiles of protein domains. *Nucleic Acids Research*, 26(1):320–322, 1998.

[Sha97] E. I. Shakhnovich. Theoretical studies of protein folding thermodynamics and kinetics. *Current Opinion in Structural Biology*, 7:29–40, 1997.

[Sid96] A. Sidow. Gen(om)e duplications in the evolution of early vertebrates. *Current Opinion in Genetics and Development*, 6:715–722, 1996.

[SK83] D. Sankoff and J. B. Kruskal. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.* Addison-Wesley, London, 1983.

[SKB+96] K. Sjölander, K. Karplus, M. Brown, R. Hughey, A. Krogh, I. S. Mian, and D. Haussler. Dirichlet mixtures: a method for improved detection of weak but significant protein sequence homology. *Computer Applications in the Biosciences*, 12(4):327–345, 1996.

[SKR89] K. Schughart, C. Kappen, and F. Ruddle. Duplication of large genomic regions during the evolution of vertebrate homeobox genes. *Proceedings of the National Academy of Sciences of the USA*, 86:7067–7071, 1989.

[SLP$^+$96] T. A. Starich, R. Y. Lee, C. Panzarella, L. Avery, and J. E. Shaw. eat-5 and unc-7 represent a multigene family in Caenorhabditis elegans involved in cell-cell coupling. *Journal of Computational Biology*, 134:537–548, 1996.

[Smi87] M. M. Smith. Molecular evolution of the Saccharomyces cerevisiae histone gene loci. *Journal of Molecular Evolution*, 24:252–259, 1987.

[Smi96] A. Smit. The origin of interspersed repeats in the human genome. *Current Opinion in Genetics*, 6:743–748, 1996.

[SO95] A. Shinohara and T. Ogawa. Homologous recombination and the roles of double-strand breaks. *Trends in the Biochemical Sciences*, 20:387–391, 1995.

[SR96] A. Smit and A. Riggs. *Tiggers* and other DNA transposon fossils in the human genome. *Proceedings of the National Academy of Sciences of the USA*, 93:1443–1448, 1996.

[SW81] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[Tat] R. Tatusov. Personal communication.

[Tay87] W. R. Taylor. Multiple sequence alignment by a pairwise algorithm. *Computer Applications in the Biosciences*, 3:81–87, 1987.

[TCD$^+$95] E. R. Troemel, J. H. Chou, N. D. Dwyer, H. A. Colbert, and C. I. Bargmann. Divergent seven transmembrane receptors are candidate chemosensory receptors in C.elegans. *Cell*, 83:207–218, 1995.

[THG94a] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.

[THG94b] J. D. Thompson, D. G. Higgins, and T. J. Gibson. Improved sensitivity of profile searches through the use of sequence weights and gap excision. *Computer Applications in the Biosciences*, 10:19–29, 1994.

[TK98] H. Tachida and T. Kuboyama. Evolution of multigene families by gene duplication: a haploid model. *Genetics*, 149:2147–2158, 1998.

[TKF91] J. L. Thorne, H. Kishino, and J. Felsenstein. An evolutionary model for maximum likelihood alignment of DNA sequences. *Journal of Molecular Evolution*, 33:114–124, 1991.

[TKF92] J. L. Thorne, H. Kishino, and J. Felsenstein. Inching toward reality: an improved likelihood model of sequence evolution. *Methods in Enzymology*, 34:3–16, 1992.

[TRTA98] R. Tarrio, F. Rodriguez-Trelles, and F. J. Ayala. New Drosophila introns originate by duplication. *Proceedings of the National Academy of Sciences of the USA*, 95:1658–1662, 1998.

[TW96] M. G. Tomlinson and M. D. Wright. A new transmembrane 4 superfamily molecule in the nematode, Caenorhabditis elegans. *Journal of Molecular Evolution*, 43:312–314, 1996.

[VBA$^+$98] P. Vincens, L. Buffat, C. Andre, J. P. Chevrolat, J. F. Boisvieux, and S. Hazout. A strategy for finding regions of similarity in complete genome sequences. *Bioinformatics*, 14:715–725, 1998.

[VBP96] J. Vos, I. De Baere, and R. Plasterk. Transposase is the only nematode protein required for in vitro transposition of Tc1. *Genes and Development*, 10:755–761, 1996.

[Vit67] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, pages 260–269, 1967.

[vLCP94] H. van Luenen, S. Colloms, and R. Plasterk. The mechanism of transposition of Tc3 in C.elegans. *Cell*, 79:293–301, 1994.

[VP94] J. C. Vos and R. H. Plasterk. Tcl transposase of Caenorhabditis elegans is an endonuclease with a bipartite DNA binding domain. *EMBO Journal*, 13:6125–6132, 1994.

[VvLP93] J. C. Vos, H. G. van Luenen, and R. H. Plasterk. Characterization of the Caenorhabditis elegans Tcl transposase in vivo and in vitro. *Genes and Development*, 7:1244–1253, 1993.

[VW94] M. Vingron and M. S. Waterman. Sequence alignment and penalty choice: review of concepts, case studies and implications. *Journal of Molecular Biology*, 235:1–12, 1994.

[Wat95] M. S. Waterman. *Introduction to Computational Biology*. Chapman & Hall, London, 1995.

[WE87] M. S. Waterman and M. Eggert. A new algorithm for best subsequence alignments with application to tRNA–rRNA comparisons. *Journal of Molecular Biology*, 197:723–725, 1987.

[Wes89] S. Wessler. The splicing of maize transposable elements from premRNA - a minireview. *Gene*, 82:127–133, 1989.

[Woo94] J. C. Wootton. Non-globular domains in protein sequences: automated segmentation using complexity measures. *Computational Chemistry*, 18:269–85, 1994.

[WP84] M. S. Waterman and M. D. Perlwitz. Line geometries for sequence comparisons. *Bulletin of Mathematical Biology*, 46:567–577, 1984.

[WS97] K. Wolfe and D. C. Shields. Molecular evidence for an ancient duplication of the entire yeast genome. *Nature*, 387:708–713, 1997.

[WS98] R. Waterston and J. E. Sulston. The Human Genome Project: reaching the finish line. *Science*, 282:53–54, 1998.

[YWB97] J. Yoder, C. Walsh, and T. Bestor. Cytosine methylation and the ecology of intragenomic parasites. *Trends in Genetics*, 13(8):335–340, 1997.

[ZB96] H. Zhang and T. Blumenthal. Functional analysis of an intron 3' splice site in Caenorhabditis elegans. *RNA*, 2(4):380–388, 1996.

[ZLL97] J. Zhu, J. Liu, and C. Lawrence. Bayesian adaptive alignment and inference. In Gaasterland et al. [GKK$^+$97], pages 358–368.

[ZLL98] J. Zhu, J. S. Liu, and C. E. Lawrence. Bayesian adaptive sequence alignment algorithms. *Bioinformatics*, 14:25–39, 1998.

[ZR95] M. Zetka and A. Rose. The genetics of meiosis in Caenorhabditis elegans. *Trends in Genetics*, 11:27–31, 1995.

Appendices

Appendix A

Software

A.1 Introduction

This appendix describes the software that was developed for the work described in this dissertation and is available for public use under the conditions of the Gnu Public License [GPL].

The tools fall into four groups, namely:

LogSpace A library of C++ classes for finding log-likelihoods, posterior probabilities and alignments using HMMs.

BayesPerl Perl modules and associated scripts for performing common manipulations (such as integration, transformation and marginalisation) on tables of log-likelihood data.

GFFTools A collection of perl scripts and C code for working with GFF format.

bigdp A program for joining together BLAST hits by dynamic programming.

Each of these groups will be described in turn.

A.2 LogSpace: C++ classes for working with HMMs

LogSpace is a set of classes for finding logs of likelihoods associated with single and pairwise hidden Markov models of any architecture, and for performing algorithms associated with these models.

In the following brief introduction to LogSpace, class names are shown in `typewriter` font.

The atoms of the LogSpace object model are:

- the `Parameter` class (P), representing a parameter of a probabilistic model;
- the `ParamVals` class ($\Theta : P \rightarrow \Re$), a mapping of `Parameter` objects to `doubles`), representing a point in parameter space); and

- the abstract Function class ($\log F : \Theta \to \Re$, $\frac{d \log F}{dP} : \Theta \to \Re$), representing the log-value of a mathematical function and the first derivatives of the log-value.

The most powerful and general of these classes is Function. This class can be overridden to describe any function on a parameter space that is everywhere non-negative and differentiable. Calling the methods of this object, supplying a ParamVals object, causes the value of the function (or its derivatives) to be evaluated at that point in parameter space. The emphasis in the API on logarithms of the function value (rather than the values themselves) encourages all calculations to be performed in log-space, where they are robust to scaling. A number of function calls facilitating common calculations in log-space (such as the log-sum-exp function $\Sigma(x, y) = \log[\exp x + \exp y]$) are also provided; these functions are zero-safe and often accelerated by means of lookup tables.

A range of useful Function subclasses are provided, some implementing basic mathematical functions such as the exponential and polynomial families, others allowing a Function to be defined in terms of other Functions (e.g. FunctionSum $F(\Theta) = G_1(\Theta) + G_2(\Theta)$, FunctionProduct $F(\Theta) = G_1(\Theta)G_2(\Theta)$, ChainFunction $F(\Theta) = G(\Phi)$ where $\Phi = \{H_i(\Theta)\}$), implementing basic results from calculus such as the chain rule and the product rule.

A pairwise hidden Markov Model M is represented in LogSpace as a set of links between the states of the model. Links can be added dynamically. Each link has associated with it a probabilistic substitution matrix, each entry of which is a Function on the parameter space. So this allows for quite general functional forms for the transition probabilities, corresponding to (for example) a time-dependent evolutionary model, or a model where several transitions are constrained to have the same (or related) probabilities. Links that correspond to insertions or deletions and hence emit only on one side of the model are also possible, and single-sequence models (like profile HMMs) are obtained as special cases of pairwise models.

The abstract `Envelope` class describes an iteration over a dynamic programming matrix, says which transitions are allowed and also describes how the matrix is to be laid out in memory. An `Envelope` requires a `Model` and a `SequencePair` (the latter is fairly self-descriptive); the default `Envelope`, `FullEnvelope`, just iterates over all the cells of the matrix. A `SparseEnvelope` class implementing the sparse envelopes described in Chapter 2 is also provided.

The abstract `DPMBase` is the base class for all dynamic programming algorithms including Viterbi, sum-over-paths and optimal-accuracy, using integer precision or double precision. `DPMBase` takes an `Envelope` in its constructor. The `FBM` (forward-backward matrix) provides posterior probabilities for every point in the matrix.

The details of the dynamic programming are invisible if the `Likelihood` class $L(\Theta) = \Pr[S|M, \Theta]$ is used. `Likelihood` is a subclass of `Function` that takes an `Envelope` in its constructor. Recall that an `Envelope` describes a pair of sequences S and a model M; so, a `Likelihood` object calculates the value and derivatives of the log-likelihood score of a pair of sequences given a model for any particular parameterisation of that model.

The derivatives of the likelihood can be fed straight into a discriminative classifier [JH98] or used to train the model [DEKM98]. A number of classes such as `JointLikelihood`

There are many other classes and methods in the LogSpace libraries, including alignments, time-dependent models and optimisation algorithms, that are not described here. It is hoped that the above short introduction is sufficient to give an idea of the range of these libraries.

A.2.1 Posterior probabilities for profile HMMs

The LogSpace classes were used to develop a posterior probability framework for HMMER2.0 profiles, in parallel with the code described in Chapter 4. This parallel implementation (and a perl program `hmm2mf.pl` for converting between

HMMER2.0 and LogSpace model file formats) is included in the LogSpace distribution.

A.2.2 Availability

The LogSpace source code can be found at the following URL:

`http://www.sanger.ac.uk/Users/ihh/LogSpace/`

A.3 BayesPerl: Perl modules and scripts for working with tables of log-likelihood data

The LogSpace code described above can be used to sample the log-likelihood of sequences over the parameter space of a hidden Markov model. These operations are typically compute-intensive and it is convenient to save the log-likelihood tables in intermediate data files before performing further numerical manipulations. BayesPerl is a set of perl modules and scripts for manipulating these data files and the tables they contain.

The format of the data files operated on by these programs is as follows. Each line of the file represents a single entry in the table, and thus a single data point. Each line has $N + 1$ numeric fields separated by whitespace, where N is the dimensionality of the parameter space. The first N fields are the parameter values (the co-ordinates in the parameter space) and the $(N + 1)$'th field is the value of the log-likelihood at that point.

The main component of the BayesPerl modules is the `LogLikeTable.pm` package, which contains the following main methods:

`new` Creates a new table of log-likelihood values.

`clone` Clones an existing table.

`newFromHandle` Reads a table from a file handle.

`newFromFile` Reads a table from a file.

`newFromArray` Creates a table from an array of values.

`newFromFunction` Creates a table from a perl function reference, which is evaluated at every point.

`combine` Combines two tables to give a new table with higher dimensionality.

`combineEntriesRule` Sets the rule for combining two log-likelihoods at the same point (by default, the log-likelihoods are summed).

`paramRange` Returns the range of values taken by a particular parameter in a table.

`findMode` Finds the mode of a table (the maximum-likelihood parameters).

`absorb` Multiplies (or adds, depending on the combineEntriesRule) a table by another table; useful for e.g. priors.

`marginalise` Marginalises parameters of a table by integrating them out; the result is a table of lower dimensionality.

`integrate` Finds the log-integral of the likelihood across the entire space; equivalent to marginalising all parameters.

`normalise` Subtracts the log-integral from all the log-likelihoods; turns a likelihood distribution into a posterior distribution.

`interpolate` Uses straight-line nearest-neighbour interpolation to find the log-likelihood anywhere in the parameter space.

`transform` Projects the table onto a new co-ordinate system.

`print` Displays a table (or dumps it to a file).

Much of the functionality of the `LogLikeTable.pm` package is duplicated by the (slightly more efficient) `LogLikeGrid.pm` package, which assumes that its data points lie on an irregular grid.

There is also a package `LogSpace.pm` that supplies some basic constants and functions compatible with the LogSpace C++ classes described above.

Some of the methods are quite slow on large tables. Much of this slowness is due to the Perl object-orientation layer, so a set of procedural scripts that mirror some of the `LogLikeTable.pm` methods (but faster) has also been developed.

A.3.1 Availability

The BayesPerl release can be found at the following URL:

`http://www.sanger.ac.uk/Users/ihh/Bayes/`

A.4 GFFTools: Perl scripts for processing GFF files

GFF (Gene-Finding Format) is a one-line-per-record format for marking up genomic sequence. It was originally designed as a common format for sharing information between gene-finding sensors such as splice site and coding sequence predictors, but its uses go beyond gene-finding: a GFF file is a convenient way of representing a set of many kinds of feature. The chief drawback of GFF - its simplicity - could also be said to be its chief strength, since a wide range of perl scripts and modules for operating on GFF sets has been developed in a short space of time ($\sim$ 1 year). The latest version of `ACeDB` has the facility to export all *C.elegans* genome annotation in GFF format.

A single record in a GFF data set is a line with 9 tab-separated fields. These fields are:

- Sequence name
- Source (the program that generated the data)
- Feature name
- Sequence startpoint

- Sequence endpoint
- Score of feature
- Strand (can be "+", "-" or ".")
- Frame (for frame-sensitive features such as introns)
- Group (a catch-all miscellaneous field)

Fields can contain spaces (but not tabs or newlines). A proposed extension to GFF is the "GFF pair", wherein first three whitespace-delimited words in the "group" field represent the sequence name, startpoint and endpoint of a homologous sequence. Other than this, there is no consensus on syntax for fields, though a "tag=value" approach to including extra information in the "group" field may be favoured.

Soon after the GFF format was agreed, Tim Hubbard at the Sanger Centre developed a set of useful GFF-related Perl modules [GFF]. Many of the scripts described here duplicate functionality included in these modules, yet they were developed later. Why is this? The reasons, simply, were speed and space. Perl's object-orientation slows computation time considerably and greater interactivity was required than the GFF Perl modules allowed; as for space, reading entire chromosome-sized GFF files into memory is often impractical.

A GFF record is a special case of a class of objects that may be described as annotated NSE's. A basic NSE consists of a *(name,start,end)* tuple. Many of the algorithms described here would work without modification on other NSE-like formats.

The following list contains brief descriptions all the GFF programs developed, together with several programs for working with EXBLX, an alternative format for representing NSE pairs that is used by the `MSPcrunch` program [SD94].

- GFF sorting/filtering programs:

`gffsort.pl` Sorts a stream of GFF records by sequence name and startpoint.

`gfffilter.pl` Filters a GFF stream according to a user-specifiable test condition.

`gffmerge.pl` Merges two or more pre-sorted GFF streams.

- Programs to convert between GFF co-ordinate systems and manipulate GFF-described sequences:

`gfftransform.pl` Converts from one GFF co-ordinate system to another (e.g. from clones to chromosomes). Works with GFF pairs.

`gff2seq.pl` Given chromosome co-ordinates, a clone database and a physical map co-ordinate file, returns the specified section of chromosomal sequence.

`gffmask.pl` Masks GFF-specified sections of a FASTA sequence database with N's.

`GFFTransform.pm` Perl module to convert between GFF co-ordinate systems. Used by the `gfftransform.pl` and `exblxtransform.pl` scripts.

`SequenceMap.pm` Perl module to access a clone database given a map file. Used by the `gff2seq.pl` script.

`FileIndex.pm` Perl module to build a quick lookup index for flatfiles. Used by the `SequenceMap.pm` module, and others.

- Programs to find intersections and connections between GFF data sets:

`gffintersect.pl` Efficiently finds the intersection (or exclusion) between two (sorted) GFF streams. Definition of "intersection" allows for near-neighbours and minimum-overlap.

`intersectlookup.pl` Used with `gffintersect.pl` to do inverse lookups and other manipulations on the result of an intersection test. Can be useful for self-comparisons, e.g. to find the highest-scoring non-overlapping subset of a GFF file.

`gffhitcount` C++ program that counts the number of times each base in a GFF file is hit, and outputs the results as a GFF tiling. Uses a lot of memory. Works with GFF pairs.

`exblxgffintersect.pl` Similar to `gffintersect.pl`, but finds the intersection between a GFF set and an EXBLX file (similar to a list of GFF pairs). Useful for e.g. filtering out all hits between genes from an all-vs-all comparison of genomic DNA.

`gffdp.pl` Parses GFF data using a finite-state automaton with a push-down stack. The FSA is entirely user-specifiable and may include Perl expressions that are evaluated dynamically for each GFF record. This program is described in greater detail below.

- Miscelleneous GFF-related programs:

`blasttransform.pl` BLASTs a clone database against itself, then transforms, sorts and merges the results into chromosome co-ordinates according to a physical (sequence) map.

`cfilter.pl` Uses a sliding-window oligomer-counting method to find GFF co-ordinates for low-entropy regions in a sequence database.

- Miscellaneous EXBLX-related programs:

`exblxsym.pl` Symmetrises an EXBLX file (ensures that for every pair $A \leftrightarrow B$ there is a single corresponding pair $B \leftrightarrow A$).

`exblxasym.pl` Asymmetrises an EXBLX stream (filters through only those pairs $A \leftrightarrow B$ for which $B > A$).

`exblxcluster.pl` Builds single-linkage clusters from an EXBLX stream, optimising for cluster size.

`exblxfastcluster.pl` Builds clusters from an EXBLX stream using a fast incremental heuristic.

`seqcluster.pl` Builds single-linkage clusters from an EXBLX stream, optimising for cluster size and ignoring sequence start and endpoint.

`exblxindex.pl` Builds a quick lookup index for an EXBLX file.

`exblxsingles.pl` Filters through only non-overlapping entries from an EXBLX stream.

`exblxsort.pl` Sorts an EXBLX stream.

`exblxtidy.pl` Tidies up an EXBLX stream (joins overlapping matches, prunes out BLAST errors, etc.).

`exblxtransform.pl` Converts from one EXBLX co-ordinate system to another (e.g. from clones to chromosomes).

- Format conversion programs:

  `exblx2gff.pl` From EXBLX to GFF pairs.

  `gff2exblx.pl` From GFF pairs to EXBLX.

  `scan2gff.pl` From scan (GCG) to GFF.

  `tandem2gff.pl` From tandem (GCG) to GFF.

  `spcwise2gff.pl` From spcwise (GeneWise) to GFF.

  `cluster2gff.pl` From single-line lists of NSE clusters to GFF.

  `hmm2gff.pl` From HMMER1.7 to GFF.

  `hmmsearch2gff.pl` From HMMER2.0 to GFF.

The most flexible of the GFF programs is `gffdp.pl`. This assembles GFF segments using dynamic programming. The model architecture for the dynamic programming is specified in a text file using a syntax that allows perl expressions to be evaluated on-the-fly. The finite state automaton has a LIFO stack that allows nested structures (such as inverted repeats) to be handled in a sensible way. All alternative states of the stack are maintained in the dynamic programming matrix. All the arcs in the finite state machine can emit variable-length sequences, so the program can emulate a generalised HMM [Hau98]. The dynamic evaluation of perl expressions during the dynamic programming allows for calculation of complex scoring schemes, such as logarithmic gap penalties. Transitions along an arc may be required to overlap or align with a GFF segment exactly, loosely or not at all. GFF pairs are also provided for; the co-ordinates of the paired segment can accessed and it can even be inserted into the upcoming GFF buffer.

Figure A.1 shows a simplified version of one of the `gffdp.pl` model files used for the repeat-mediated duplications search described in Section 5.4 of Chapter 5. The `gffdp.pl` program was also used in Chapter 7 to find invrep sequences flanking predicted transposase genes. There are many other uses for the `gffdp.pl` program; one obvious use is genefinding - assembling exon predictions from a variety of sensors. This is the task that GFF was originally designed for. A `gffdp.pl` model file for genefinding is available from the GFF website.

A.4.1 Availability

Further information and resources relating to the `gffdp.pl` program and the GFF format may be obtained from the GFF website, whose URL is:

`http://www.sanger.ac.uk/Software/GFF/`

A.5 bigdp: A program for assembling BLAST hits by dynamic programming

`bigdp` is a program that joins together BLAST hits with an affine gap penalty by doing linear space divide-and-conquer dynamic programming [DEKM98]. The program does not itself examine the sequence to which the BLAST HSPs refer

```
name { rep1rep2 } flushlen { 30000 }

link { from { start } to { repeat1 } maxlen { 3000 }
       endfilter { $gffsource eq "repeat"
                   && $linkend == $gffend + 1 }
       startfilter { $linkstart == $gffstart }
       push { $gfffeature } push { $gffstrand }
}

link { from { repeat1 } to { match1 } maxlen { 5000 }
       endfilter { $gffsource eq "match"
                   && $gffstrand eq '+'
                   && $linkend == $gffend + 1
                   && $gffend-$gffstart > 20 }
       startfilter { $linkstart <= $gffstart }
       insertgff { }
       popfilter { $temp_repstrand = $_; 1 }
       popfilter { $temp_repname = $_; 1 }
       push { $insertgffid } push { $temp_repname } push { $temp_repstrand }
}

link { from { match1 } to { repeat2 } maxlen { 4000 }
       endfilter { $gffsource eq "repeat"
                   && $linkend == $gffend + 1 }
       startfilter { $linkstart <= $gffstart }
       popfilter { $_ eq $gffstrand }
       popfilter { $_ eq $gfffeature }
}

link { from { repeat2 } to { end } maxlen { 5000 }
       endfilter { $gffsource eq "match"
                   && $gffstrand eq '+'
                   && $linkend == $gffend + 1 }
       startfilter { $linkstart <= $gffstart }
       popfilter { }
       display { print "Found a hit ending at $gffend\n" }
}

link { from { end } to { start } }
```

Figure A.1: Model file for the *repeat* → *match* → *repeat* → *match* pattern. Whether a GFF line represents a *match* or a *repeat* is indicated in the endfilter field. Matches are parsed as GFF pairs.

but merely finds optimal-scoring connections between the HSPs given their coordinates. bigdp was designed to cope with large amounts of data, such as might be generated by BLASTing whole chromosomes against one another. It is essentially similar to the Smith-Waterman algorithm [SW81], except that all *match* $\rightarrow$ *match* transitions must correspond to BLAST hits and consequently many cells in the dynamic programming matrix can effectively be dispensed with.

The bigdp program returns all (non-overlapping) alignments above a certain score threshold by a method of excluding previously-used segments; the closest relative of this method is the procedure described by Waterman and Eggert [WE87]. Rather than covering the whole dynamic programming matrix, the algorithm stops if an alignment above the score threshold has been found and there have been no better alignments after a distance δ has been covered. Additionally, rather than start from the beginning of the dynamic programming matrix after an optimal alignment has been found, the algorithm starts a distance ϵ left of the startpoint of the highest-scoring alignment found on the previous run. The startpoint information is propagated using a "shadow matrix" technique [BD97]. Choosing δ to be much greater than the maximum gap length and ϵ to be much greater than the maximum low-scoring subalignment length reduces the expected running time to $O(MN^2)$, where M and N are the sequence lengths (since there are $\sim MN$ expected alignments and each takes time $\sim N$ to find), rather than $O(M^2N^2)$ (the expected running time if the entire matrix were to be scanned for each alignment) without missing any hits. Alignments may be missed if they overlap high-scoring alignments and contain subsegments longer than ϵ that score lower than the alignment detection threshold.

Other programs to extend BLAST to give gapped alignments include gapped BLAST [AG96, AMS+97] and MSPcrunch [SD94]. There are other ways of searching large sequences quickly for multiple matches, e.g. [VBA+98]. All these programs use heuristic methods and not full dynamic programming; in some cases the heuristics may be more sensitive. The statistical bias induced by the heuristic methods on the observed data is likely to be different than the dynamic programming.

A.5.1 Availability

The `bigdp` program is available from the following URL:

```
http://www.sanger.ac.uk/Users/ihh/bigdp/
```