Bioinformatic protein family characterisation

Joel Hedlund
The front cover shows a tree diagram of the relations between proteins in the MDR superfamily (papers III–IV), excluding non-eukaryotic sequences as well as four fifths of the remainder for clarity. In total, 518 out of the 16667 known members are shown, and 1.5 cm in the dendrogram represents 10 % sequence differences. The bottom bar diagram shows conservation in these sequences using the CScore algorithm from the MSAView program (papers II and V), with infrequent insertions omitted for brevity. This example illustrates the size and complexity of the MDR superfamily, and it also serves as an illuminating example of the intricacies of the field of bioinformatics as a whole, where, after scaling down and removing layer after layer of complexity, there is still always ample size and complexity left to go around.

The back cover shows a schematic view of the three-dimensional structure of human class III alcohol dehydrogenase, indicating the positions of the zinc ion and NAD cofactors, as well as the Rossmann fold cofactor binding domain (red) and the GroES-like folding core of the catalytic domain (green).

This thesis was typeset using LyX. Inkscape was used for figure layout.

During the course of research underlying this thesis, Joel Hedlund was enrolled in Forum Scientium, a multidisciplinary doctoral programme at Linköping University, Sweden.

Copyright © 2010 Joel Hedlund, unless otherwise noted. All rights reserved.

Joel Hedlund
Bioinformatic protein family characterisation
ISBN: 978-91-7393-297-4
ISSN: 0345-7524
Linköping studies in science and technology, dissertation No. 1343

Till mina kära

A still more glorious dawn awaits
not a sunrise, but a galaxy rise
a morning filled with 400 billion suns

Science is a collaborative enterprise
spanning the generations
we remember those who prepared the way
seeing for them also

— Carl Sagan
Abstract

Biological research is necessary; not only to further our understanding of the processes of life, but also to combat disease, hunger and environmental damage.

Bioinformatics is the science of handling biological information. It entails integrating, structuring and analysing the ever-increasing amounts of available biological data. In practice it means using computers to analyse huge amounts of very complicated data taken from a field that is only partially understood, to see the hidden trends and connections, and to draw useful conclusions.

My thesis work has mainly concerned the study of protein families, which are groups of evolutionarily related proteins. I have analysed known protein families and created predictive models for them, and developed algorithms for defining new protein families. My principal techniques have been sequence alignments and hidden Markov models (HMM). To aid my work, I have written a lot of software, including MSAView, a visualiser for multiple sequence alignments (MSA).

In this thesis, the protein family of inorganic pyrophosphatases (H\(^+\)-PPases) is studied, as well as the two protein superfamilies BRICHO and MDR (medium-chain dehydrogenases/reductases). The H\(^+\)-PPases are tightly membrane bound, proton pumping, dimeric enzymes with ~700-residue subunits and found in bacteria, plants and eukaryotic parasites, and which use pyrophosphate as an alternative to ATP. The BRICHO superfamily is only present in higher eukaryotes, but encompasses at least 8 protein families with a wide range of functions and disease associations, such as respiratory distress syndrome, dementia and cancer. The sequences are typically ~200 residues with even shorter functional forms. Finally, MDR, is a large and complex protein superfamily; it currently has over 16000 members, it is present in all kingdoms of life, the pairwise sequence identity is typically around 25\%, the chain lengths vary as does the oligomericty, and the members are partaking in a multitude of biological processes. The member families include the classical liver alcohol dehydrogenase (ADH), quinone reductase, leukotriene B4 dehydrogenase, and many more forms. There are at least 25 human MDR genes excluding close homologues. There are HMMs available for detecting MDR superfamily membership, but none for the individual families.

For the H\(^+\)-PPase family, we characterised member sequences found using an HMM of a conserved 57-residue region thought to form part of the active site. This region was found to contain two highly conserved nonapeptides, mainly consisting of the four “very early” residues Gly, Ala, Val and Asp, compatible with an ancient origin of the family. The two patterns have charged amino acid
residues at positions 1, 5 and 9, are apparent binding sites for the substrate and parts of the active site, and were shown to be so specific for these enzymes that they can be used for automated annotation of new sequences.

For the BRICHOS superfamily, we were able to find three previously unknown member families; group A, which may be ancestral to the ITM2 families (integral membrane protein 2); group B, which is a close relative to the gastrokine families, and group C, which appears to be a truly novel, disjoint BRICHOS family. The C-terminal region of group C has nearly identical sequences in all species ranging from fish to man and is seemingly unique to this family, indicating critical functional or structural properties.

For the MDR superfamily, we characterised and built stable HMMs for 17 member families using an empiric approach. From our experiences we were able to develop an algorithm for automated HMM refinement that uses relationships in data to produce stable and reliable classifiers, and we used it to produce HMMs for 86 distinct MDR families. We have made the program freely available and it can be readily applied to other protein families. We also developed a web site (http: //mdr–enzymes.org) that makes our findings directly useful also for non-bioinformaticians.

In our analyses of the 86 families, we found that MDR forms with 2 Zn$^{2+}$ ions in general are dehydrogenases, while MDR forms with no Zn$^{2+}$ in general are reductases. Furthermore, in Bacteria, MDRs without Zn$^{2+}$ are more frequent than those with Zn$^{2+}$, while the opposite is true for eukaryotic MDRs, indicating that Zn$^{2+}$ has been recruited into the MDR superfamily after the initial life kingdom separations.

Multiple sequence alignments (MSA) play a central part in most work on protein families, and are integral to many bioinformatic methods. With the ongoing explosive increase of available sequence data, the scales of bioinformatic projects are growing, and efficient and human-friendly data visualisation becomes increasingly challenging, but is still essential for making new interpretations and discovering unexpected properties of the data.

Ideally, visualisation should be comprehensive and detailed, and never distract with irrelevant information. It needs to offer natural and responsive ways of exploring the data, as well as provide consistent views in order to facilitate comparisons between datasets. I therefore developed MSAView, which is a fast, modular, configurable and extensible package for analysing and visualising MSAs and sequence features. It has a graphical user interface and a powerful command line client, and can be imported as a package into any Python program. It has a plugin architecture and a user extendable preset library. It can integrate and display data from online sources and launch external viewers for showing additional details. It also includes two new conservation measures; alignment divergences, which indicate atypical residues or deletions, and sequence conformances, which highlight sequences that differ from their siblings at crucial positions.

In conclusion, this thesis details my work in analysing two protein superfamilies and one protein family using bioinformatic methods; developing an algorithm for automated generation of stable and reliable HMMs, as well as a new conservation measure, and a software platform for working with aligned sequences.
Sammanfattning

Biologisk forskning är nödvändig. Inte bara för att förstå alla de otaliga och invecklade processer i kroppen som gör att vi kan överleva och fortsätta existera från ett andetag till nästa, utan även för att bota sjukdomar och svält, och för att förhindra och läka miljöskador. Frukterna av biologisk forskning är uppenbara i dagens enkla kurer för gamla farsores och lyten, men behoven av ytterligare framsteg är lika uppenbara, till exempel i bristen på botemedel mot stora folkdödare som malaria, och hotet från pandemier av nya virus.

Tyvärr är den biologiska forskningen också väldigt dyr. Mest på grund av att biologi är livet – det är fruktansvärt komplext, och vi förstår det inte! Även enkla experiment kräver lång tid och specialiserad utrustning. Dessutom kan man inte göra vilka experiment som helst, för etik och moral sätter tydliga gränser för vilka genvägar det är acceptabelt att ta. Man testar inte hux flux en ny kemikalie på folk, utan man tar den långa vägen via provrör och modellorganismer, för att gå vidare till människa årtleal av forskning senare. Man måste alltså prioritera, och börja med de experiment som kommer lära oss mest, oavsett utfall, och här kommer bioinformatiken in. Sedan gäller det att krama den maximala mängden kunskap ur varje experiment, och även detta hör bioinformatiken till.

Bioinformatik är läran om informationshantering i biologi. En aspekt är att se till att alla resultat finns lagrade på ett ordnat och lättillgängligt sätt, så att forskare världen över enkelt kan dra nytta av dem. En annan aspekt är att använda de insamlade data; strukturera, sammanställa, jämföra och dra nya slutsatser av gamla resultat, och se nya trender och kopplingar då gamla resultat ses i ljuset av nya. På så sätt kan vi vägleda forskningen, ge uppslag till nya experiment, och se till att utförda experiment kommer till maximal nytta för mänskligheten.

Vår framgång i det första avseendet är en stor utmaning för det andra. De experimentella metoderna utvecklas ständig och går mot större och större datamängder och produktionstakt, samtidigt som resultaten blir mer och mer tillgängliga och kompatibla. Möjliheterna ökar exponentialt, men samtidigt drunknar vi i samma data som vi arbetade så hårt för att samla in. Smartare och snabbarare program och bättre sätt att använda större datorer är i ständig efterfrågan, och automatisering blir en nödvändighet för att hålla oss flytande.

I min avhandling har jag framför allt studerat proteinfamiljer, det vill säga grupper av evolutionärt besläktade protein (och proteiner är de molekyler som faktiskt gör det mesta av det som händer i cellen, medan DNA, vår arvsmassa i genomet,
kan sägas vara ritningar). Till exempel hör alkoldehydrogenas i människa, mus, jästsvampar och bakterier till samma proteinfamilj. Inom en proteinfamilj kan man vara skapligt säker på att proteinerna ser ut på samma sätt, fungerar på samma sätt och reagerar på ungefär samma sätt. Genom att sammanställa data från olika medlemmar i familjen kan man bilda sig en god uppfattning om hur proteinerna fungerar. Behöver man sedan komplettera bilden så kan man i de flesta fall mycket väl utföra experimenten på jästceller i provrör snarare än på människor, vilket ju är fördelaktigt.

Jag har dessutom studerat superfamiljer, det vill säga familjer av proteinfamiljer. Här har medlemmarna av olika familjer inte lika mycket gemensamt, men de stora dragen är oftast fortfarande lika. Genom att titta på likheter och skillnader mellan familjerna kan man hitta molekylära förklaringar till varför familjerna beter sig olika.

Proteiner är långa kedjor av aminosyror (sekvenser av ”biologiska bokstäver”), och genom att titta på sekvensvariationerna inom en familj kan man bygga statistiska modeller för hur en typisk familjemedlem ser ut. Med hjälp av modeller kan man sedan hitta nya familjemedlemmar bland nya sekvenser, till exempel när ett nytt genom blir sekvenserat. Klassificeringen kan dessutom göras helt automatiskt, så ju fler modeller som finns tillgängliga, desto mer kunskap får vi gratis. Tyvärr är det inte helt enkelt att ta fram nya modeller, utan det har krävts manuellt och tidskrävande arbete av experter på området för att resultatet ska bli bra, och detta har hittills varit en rejäl flaskhals.


Vidare har jag tagit fram en ny algoritm för att mäta sekvensvariationer (CScore), och jag utvecklade en programvara för att arbeta med inpassade proteinsekvenser (MSAView), vilket är en central del av många bioinformatiska tillämpningar, och som jag använt flitigt under hela avhandlingsarbetet. MSAView kan mäta och visa många egenskaper hos proteiner, inklusive CScore.

Så, för att sammanfatta det hela kan man säga följande. I mitt avhandlingsarbete har jag sammanställt och dragit nya slutsatser från existerande data för proteiner och proteinfamiljer. Vidare har jag strukturerat data, dels genom att gruppera sekvenser till familjer och bygga klassificeringsmodeller för dem, och dels genom att utveckla en tillförlitlig metod för att ta fram nya klassificeringsmodeller automatiskt. Dessutom har jag tagit fram en effektiv metod för att underlätta arbetet med proteindata. Slutligen har jag även gjort det enkelt att publicera mina artiklar i tidskrifter med öppen tillgång för allmänheten, utan även genom att göra mina program tillgängliga med öppen källkod, och via lättillgängliga webbplatser.
Acknowledgements

Well, here we are. Writing my thesis always seemed such a distant prospect. To think that there would actually come a time where I would sit down and sum it all up in a book actually felt downright implausible at times. Thinking back over the years, with all the peaks of elation and chasms of doubt, it is very easy to slip into nostalgia, and there are so many people that I wish to thank.

First of all, my supervisor Bengt Persson, thank you for giving me this opportunity to contribute to modern science. You have been a true source of inspiration in all my research projects with dauntless optimism and fierce determination. Thank you for all your work and for sharing your insights, and for answering the phone at the oddest hours. Thank you for all your help and support!

Thank you Anders Bresell for all help with that science stuff and all our valuable discussions at work! Looking back through a rough decade of M.Sc. programme and Ph.D. neighbourhood, you’ve always been my test pilot, one half step ahead of me. I trust your taste in all but sports and orange chocolate!

Thank you Jonas Carlsson! We’ve exchanged more ideas on important science over the years than I can possibly remember, but also less pit chagis and nerdy science books than I can possibly allow. Sharing an office with you was a blast (aha!) and easily as much fun as The Moose on the black diamonds. I missed you when you moved out of my office, and I’ll miss you when you’re off to Uppsala.

Thank you Fredrik Lysholm, my newest doctoral buddy, for teaching me to appreciate things like continuous memory and stupid little bits! Your enthusiasm and intrepidity – which against all reason even extend to pieces of alien genetic material nesting in our chromosomes – are an inspiration that deflates the seemingly insurmountable to immediate tractability. Thanks also for watering my plants!

I of course want to thank all my collaborators from my thesis work, so thank you Herrick Baltscheffsky, Margareta Baltscheffsky, Roberto Cantoni and Jan Johansson for all hard work and ideas in the H⁺-PPase and BRICHOS projects. Thank you Hans Jörnvall for all your work and ideas, and for keeping me in the MDR loop. I realise only now as I’m writing this that we now have four papers together. May we fill a library!

Thank you also, my valued collaborators from other projects; Charlotte Immerstrand, Karl-Erik Magnusson, Tommy Sundqvist, Kajsa Holmgren Peterson, Hanna Eriksson, Johan Lengqvist, Kristina Uhlén, Lukas Orre, Bengt Bjellqvist, Janne Lehtiö, Per-Johan Jakobsson, Tomas Bergman, Udo Oppermann, Ella Cederglund, Lars Hjelmqvist, Jawed Shafqat, Annika Norin and Wing-Ming Keung!
Furthermore, I’d like to thank Michael Grønager, Josva Kleist and the rest of NDGF for funding Biogrid, and also the Swegrid admins for installing our REs. Jens Larsson and Leif Nixon at NSC deserve special mention for tolerating being on my speed dial as a personal crash response team, and so does Olli Tourunen, for being my closest collaborator and fellow hard rocker in the project.

A big thank you also goes out to my old compbio colleagues Roland Nilsson, Kristoffer Hallén, Jose Peña and Jesper Tegnér, and of course also Tiimo Koski from MAI, for helping me bring Vapnik to heel and giving my research a solid Bayes. (By the way, Rolle; I’m still using your brewer!) I also want to thank Jan-Ove Järnhed for doing an exemplary job keeping our servers serving, and Kerstin Vestin, Lejla Kronbäck, Åsa Forsell and Ingegärd Andersson for being admirable proactive guides through the bewildering world of academic bureaucracy.

Thanks also go to all the IFM people who have made my workplace an enjoyable place to be; past group members, old M.Sc. classmates, interdisciplinary and undisciplined lunchtime pipe dreamers and wild speculators, gentlemen of Arbiter Elegantiae, Forum Scientium folks (and especially its head Stefan Klintström), my assigned mentor Uno Carlsson, and many others.

And thank you all friends near and far, in Linköping, Uppsala, Stockholm or where ever you may have gone off to nowadays, and thank you especially friends from olden years in Idenor; Jens Larsson, Magnus Ehnebom and Mathias Dahlgren-Hauswolff, and Kajsa and Martin Östemar, my best friends on the Wrong coast. Thank you for being true friends and for sticking with me for so long! Thanks, also, esteemed band members Johan Larsson, Lauri Siipo and Linus Fredriksson for being dedicated to the art and helping me grind cores (and blow speakers) also in my spare time. I Bow my head to you (repeatedly and furiously)!

And thank you in-laws, nephews and nieces and assorted relatives who have supported me in my doctoral plight over the years! Maybe now I won’t hole up in your basements with a laptop anymore when I visit!

I must also direct the most sincere of thanks to my loving parents, Irène and Per Hedlund. I know you like to credit me with everything, but without your upbringing I doubt I would have had the brain nor the curiosity or tenacity to get through higher education, let alone this science business. Our home was always an inspirational one, and you always took the time to create all manner of fantastic stuff with me, from krotophones to solar systems on strings (space monsters inclusive). And Martin, brother and friend! Thank you for being there all these years, and for being someone I know I will always be able to count on. I hope when we are ninety, we’ll still exchange lard sculptures at Christmas!

Just as protein families are the focus of my thesis, my focus out of work has been my own family; my wonderful wife Åsa and my little ray of sunshine Oskar. Oskar, when you are old enough to read this, know that for unwinding your thesis freaked-out father, you were the best. After an intense day of hard writing and equally hard deadlines, there was nothing like your beaming greeting at the door, with Rally-Rakel and “dansa droven”, for grounding me out and helping me find my mellow again.

Åsa, love of my life, thank you for being my soul mate through better and worse. I hope I can be there for you like you always were for me. Let’s have a long and bright future together! I love you forever!
Papers

Paper I
Analysis of ancient sequence motifs in the H⁺-PPase family.
Joel Hedlund, Roberto Cantoni, Margareta Baltscheffsky, Herrick Baltscheffsky and Bengt Persson.

Paper II
BRICHOS – a superfamily of multidomain proteins with diverse functions.
Joel Hedlund, Jan Johansson and Bengt Persson.

Paper III
The MDR superfamily.
Bengt Persson, Joel Hedlund and Hans Jörnvall.

Paper IV
Stable subdivision of the MDR superfamily through iterative HMM refinement.
Joel Hedlund, Hans Jörnvall and Bengt Persson.
BMC Bioinformatics 2010, 11:534

Paper V
MSAView: flexible multiple sequence alignment visualisation.
Joel Hedlund.
In Manuscript.
Publications not included in the thesis

Paper SI
Organelle transport in melanophores analyzed by white light image correlation spectroscopy.
Charlotte Immerstrand, Joel Hedlund, Karl-Erik Magnusson, Tommy Sundqvist, Kajsa Holmgren Peterson.

Paper SII
Quantitative membrane proteomics applying narrow range peptide isoelectric focusing for studies of small cell lung cancer resistance mechanisms.
Hanna Eriksson, Johan Lengqvist, Joel Hedlund, Kristina Uhlén, Lukas M. Orre, Bengt Bjellqvist, Bengt Persson, Janne Lehtiö and Per-Johan Jakobsson.

Paper SIII
Superfamilies SDR and MDR: From early ancestry to present forms, emergence of three lines, a Zn-metalloenzyme, and distinct variabilities.
Hans Jörnvall, Joel Hedlund, Tomas Bergman, Udo Oppermann and Bengt Persson.

Paper SIV
MDR-ADH enzymes: Novel species variants add resolutions in the class I/III and the sub-class I gene duplications.
Ella Cederlund, Joel Hedlund, Lars Hjelmqvist, Jawed Shafqat, Annika Norin, Wing-Ming Keung, Bengt Persson and Hans Jörnvall.
Submitted to *Chem Biol Interact.*
Contents

1 Background 1
  1.1 Sequence analysis ........................................ 2
  1.2 Biological sequences ....................................... 3
  1.3 Evolution ..................................................... 12
  1.4 Origin of the genes ......................................... 16
  1.5 Homology ....................................................... 19
  1.6 Bioinformatic challenges ..................................... 20

2 Materials and Methods 23
  2.1 Bioinformatic databases ................................... 23
    2.1.1 Sequence databases ..................................... 23
    2.1.2 Functional classifier databases ......................... 25
    2.1.3 Other databases ......................................... 25
  2.2 Sequence alignment ........................................ 26
    2.2.1 Pairwise alignment and homology search ................ 26
    2.2.2 Multiple sequence alignment ........................... 29
  2.3 Dendrograms ................................................ 31
  2.4 Conservation ................................................. 33
  2.5 Pattern recognition ......................................... 34
  2.6 Support vector machines .................................... 34
  2.7 Artificial neural networks .................................. 37
  2.8 Hidden Markov models ...................................... 41
    2.8.1 Examples .................................................. 42
      2.8.1.1 The cheating croupier ............................. 42
      2.8.1.2 Profile hidden Markov models .................... 43
      2.8.1.3 A transmembrane topology predictor .............. 44
    2.8.2 Algorithms ............................................... 44
      2.8.2.1 Some definitions ................................. 46
      2.8.2.2 Finding the most probable state path ............ 46
      2.8.2.3 Finding the probability of a sequence .......... 47
      2.8.2.4 Finding the probability to be in a specific state at a
            given position ................................... 47
      2.8.2.5 Building an HMM .................................. 48
      2.8.2.6 Generalisation ..................................... 49
CONTENTS

3 Current investigations 51
  3.1 Aims .................................................. 53
  3.2 Paper I – Inorganic pyrophosphatases ......................... 54
  3.3 Paper II – BRICHO and the CScore .......................... 60
  3.4 Paper III – The MDR superfamily .............................. 65
  3.5 Paper IV – Automated HMM refinement ......................... 72
  3.6 Paper V – MSA visualisation ................................. 79
  3.7 Conclusions ............................................. 84

4 Future challenges 87
  4.1 Automated protein family discovery ............................ 87
  4.2 Human-friendly supercomputing ............................... 88

Bibliography 95
Chapter 1

Background

Biological research is necessary. Not only to get a better understanding of the myriad of intricate and interwoven processes that go on in our very own bodies in order to ensure our continued existence, but also for combating new diseases and understanding our role in the environment. The benefits of biological research are evident in our current simple cures to old plagues and crippling diseases, but the need for further progress is equally apparent, for example in our lack for cures for genocides like malaria, and the threat of emergent pathogens like H1N1.

Unfortunately, biological research is also very costly. This is mostly because biology is life; it is horribly complex, and we don’t understand it! Even a simple experiment, like for example culturing bacteria in a test tube and measuring their reaction to certain stimuli, is influenced by innumerable variables that need to be precisely controlled in order to ensure consistent results. What’s worse, many of these variables are difficult or impossible to measure, and an unknown number of these variables are simply unknown and therefore impossible to even assess. There are also often numerous confounding factors. For example, the bacterium in question may have several mechanisms in place to react to that specific type of stimulus, only some of which produce the response that is being measured. Furthermore, biological experiments are nearly always very time consuming. The measurements in our simple example would probably only take hours, but would likely be preluded by days of rigorous preparation, growing the bacteria under exact and reproducible conditions, and painstakingly ensuring that no contamination occurs along the way.

There is also of course the ever present ethical imperative. In our modern society it is thankfully unthinkable to take the most direct route to that new biological knowledge that is most relevant to us humans, so instead of trying out new drugs on humans directly, we tend to take the long way around, starting with test tubes and yeast cells and slowly and laboriously moving up to animals and eventually people, progressing only at the slow pace set by the rigours of acceptable safety.

We obviously can’t do all the experiments we want to do. Money, time and ethics are the universal limits for global efforts as well as small lab groups, so
CHAPTER 1. BACKGROUND

therefore we have to prioritise. Preferably, we should start with those experiments that will teach us the most. Also, we should make those experiments count, making sure we pry the maximal amount of knowledge out of every speck of data that we collect.

Bioinformatics is the science of handling information on biology. One of its aspects is to ensure that experimental results are stored in an accessible and orderly fashion, so that other scientists worldwide can best benefit from them. Another aspect is to use the collected data; to process it in various ways in order to synthesise new theories, and to discern new knowledge on biological entities and processes, for example finding new genes, or explaining infection mechanisms for new viruses.

Our success in the first aspect is a great aid and a great challenge for the second. Laboratory methods develop and constantly move toward higher throughput and larger data volumes, so as accessibility and interoperability increases, the possibilities for new discoveries of course increase exponentially, but at the same time we are drowning in that same data we strove so hard to collect. Bioinformatics is a constant battle against its own success. Smarter and faster algorithms and better ways of using bigger and faster computers are perpetually in high demand, and automation becomes a necessity in order to stay afloat.

Again, bioinformatics is the science of handling biological information. It entails integrating, structuring and analysing the ever-increasing amounts of data produced by biological laboratories around the world. Its goal is to discern new knowledge on biological entities or processes, and its purpose is to provide inspiration for designing new experiments that best help fill the holes in our current understanding of biology, and to formulate new theories for these fundamental mechanisms of life.

In practise it means using computers to analyse huge amounts of very complicated data taken from a field that is only partially understood, to see the hidden trends and connections, and to draw useful conclusions.

Since biological information is at least as diverse and complex as life itself there are of course many ways of handling that information. There are therefore many disciplines that could potentially be collected under the term bioinformatics. The word bioinformatics has however come to be associated primarily with the analysis of biological sequences, and this is the primary focus of this thesis.

1.1 Sequence analysis

Sequences are very efficient information carriers. They are used in many forms and for a variety of purposes, not only in human activities but also as a foundation for life itself. But we’ll get to that in due time. Before launching ourselves into a running start, let’s first dwell a bit on some less mysterious sequences.

One very commonplace use of sequences as information carriers is written text. Words, sentences, books and even entire libraries are essentially just that; sequences of symbols (Japanese, Roman or otherwise) strung together by the author to represent ideas, views, emotion or accounts of events, all of which then
1.2 BIOLOGICAL SEQUENCES

lie in wait to be translated into knowledge in the mind of the reader. Another everyday application is in the world of computers, because at the end of the day, all kinds of computer memory are just sequences of bits; 0 and 1 symbols. It is in the interpretation of these sequences that all these zeros and ones can be translated into programs, songs, digital photographs or any of the other useful things that may be kept on a computer’s hard disk. Some parts of the sequence may correspond to a piece of an important spreadsheet document, while others may merely be pieces of junk awaiting dismissal in the recycler, while yet others may be completely random, corresponding to unused disk space. The information on what is what in the sequence is typically also stored within the sequence itself, as sequences of zeros and ones, in a well defined segment of its own.

This information on how to interpret the rest of the information is called metainformation, and if the metainformation is somehow lost or corrupted, for example as a result of mechanical failure or a sudden power surge, we no longer know exactly how to interpret the rest of the information. Suddenly, all that remains of all these important files and documents is a long, jumbled and bewildering sequence of zeros and ones that no longer makes the least bit of sense, neither to the computer nor to a human observer. This can of course be quite infuriating, because we know that the data is still in there, but since we no longer have the metainformation, we can’t readily retrieve it. The data can of course still be salvaged, but reconstructing all these jumbled and nonsensical bits of zero-and-one sequence into coherent documents is a very difficult and time consuming process, even for trained professionals.\(^1\) These people are called computer forensics, and it’s their job to analyse binary sequences, bring order to chaos and find hidden messages in scattered bits of digital data. These people have it easy. At least, they have it easy compared to bioinformaticians, because they already have a pretty good idea of what most of the data should be, while much of the biological information is still hic draconis territory. But before we delve into the subtleties of bioinformatics, we need to get better acquainted with the core of the biological information: the biological sequences.

1.2 Biological sequences

While computers string together long sequences of bits to store information, life strings together sequences of molecules to do the same, and so much more. The chromosomes for example are sequences of nucleotides, and they store all the inherited information in the cell. The proteins on the other hand make up most of the molecular screwdrivers and power tools of the cellular tool box, and they are in essence sequences of amino acids. This happenstance is very fortunate, because it allows bioinformaticians to do very nifty things to biology in computers.

The nucleotide base is the smallest unit of inherited information in the cell; the biological bit, if you like. As seen in Fig. 1.1, it comes in four flavors; adenine, thymine, cytosine and guanine, or A, T, C and G. These bases are strung together along a sugar-phosphate backbone to form deoxyribonucleic acid, or DNA. In their polymerised, strung-together form, these bases have a strong tendency to

\(^1\)by which I of course mean it’s silly expensive. Remember to back up your data, kids!
CHAPTER 1. BACKGROUND

Figure 1.1: Chemical structure of nucleotides. The structure of the scaffold that is common to all nucleotides is shown on the left. The nucleotide bases, the letters of the biological alphabet, are shown on the right. DNA uses the 2-deoxyribose pentose ring and the bases A, G, C and T, while RNA uses ribose and the bases A, G, C and U. The various bases are attached to the pentose ring via the glycosidic bond, indicated with a dotted line in the structure. Illustration adapted from wikipedia.org, used with permission.

parallel antiparallel

Figure 1.2: Parallel versus antiparallel

form specific pairwise hydrogen bonds – A to T and C to G. This process is called base pairing, and it causes complementary segments of DNA to bind strongly to each other in antiparallel pairs (cf. Fig. 1.2). The basic principles for base pairing are shown in Fig. 1.3. Thus paired, the molecules are stabilised in the characteristic double helix form that has now become a staple showcase item in all science TV shows, and whose discovery awarded James D. Watson and Francis Crick the Nobel prize in 1962 [1]. The DNA sequences of our chromosomes range from about fifty to two hundred and fifty million base pairs in length. They are truly gigantic molecules, and all of our cells each have 23 pairs of them.\(^2\)

Just as the sequence of bits on a computer’s hard disk is subdivided into files, the sequence of nucleotides in the chromosome is subdivided into genes. Genes can be thought of as blueprints for cellular components, and to push this analogy, chromosomes can be thought of as storage cabinets for the genes. It is good practise to keep backups of important files, and as we shall see, all files in these storage cabinets are stored in duplicate.

Nucleotide sequences have direction, just like sentences and computer files, meaning there is one proper direction in which they are meant to be read and where they can make sense. This direction is often called the downstream or

\(^2\)Well OK, not all of them. The red blood cells for example have no chromosomes because that would make them too thick to pass through our smallest capillaries unhindered. And the gametes only have half the number of chromosomes, but we’ll get to that (cf. section 1.3).
Figure 1.3: Chemical structure of DNA. In a nucleotide sequence, the first phosphate group of one monomer is connected to the pentose ring of the next monomer, forming a phosphodiester bond. Here we see a dimer of the two base paired DNA sequences ACTG and CAGT, strongly bound together with complementary hydrogen bonds. The large grey arrows in the background indicate the reading direction, often called the downstream or 5′ → 3′ direction, while the opposite direction is called the upstream or 3′ → 5′ direction. A-T pairs (clear) form two hydrogen bonds, while C-G pairs (grey) form three, which gives higher stability to C-G rich regions. This phenomenon is for example used by thermophilic species whose C-G rich genomes enable them to thrive in extreme conditions, like near hydrothermal vents on the ocean floor, where the water temperature can sometimes exceed 100°C. Base pairing of course works identically in RNA. Illustration adapted from wikipedia.org, used with permission.
CHAPTER 1. BACKGROUND

5' → 3' direction. This is also shown in Fig. 1.3. Since chromosomes consist of pairs of antiparallel complementary nucleotide strands, in the context of a gene, the strand that contains the readable blueprint is called the sense strand, or the forward strand. The other strand is called the antisense strand, or the reverse strand, and it can be thought of as a backup copy of the blueprint, or a failsafe for proofreading. Of course, genes can reside on any of the two strands (and genes on opposing strands can in some cases even overlap) so the sense and antisense concepts are of course quite arbitrary conveniences that only make sense in the context of individual genes.3

So seeing genes as molecular blueprints, when something needs to be built in the cell, a working copy of the corresponding blueprint is checked out from the chromosomal storage cabinet, in a process called transcription, or gene expression. Only the sense strand is copied, and instead of DNA, the copy uses ribonucleic acid (RNA), which is identical to DNA except that the thymine T base is substituted for the uracil U, and that the backbone sugar has a hydroxyl group replacing the 2' hydroxyl hydrogen atom.

We eukaryotes (who have cellular nuclei) also have exons and introns in our genes, and these allow us to adapt our RNA working copies to suit the current needs in the cell, in a process called splicing. Here, the RNA molecule is processed by the spliceosome, which removes all the introns and currently unneeded exons from the sequence, yielding a concatenated sequence of only those exons that provide the currently needed functionality. Using the genetic blueprint analogy, this is much akin to ordering a prefab home, and – instead of going with the whole shebang of ornamental fountains, furnished first floor, optional extra car port and tiled living room that would leave you with a seriously strained economy – only choosing those little extras that you actually need or appreciate. The variety of spliced sequences that can be produced from one gene are called splice variants or spliceoforms. The prokaryotes (bacteria and archaea) do not have this splicing capability, and this is only one of many instances where the eukaryotic cellular machinery is more intricate and advanced than that of the prokaryotes.

In some cases the RNA molecule itself is the final product, as for example the tRNA that transfers amino acids to the ribosome, or the ribosomal rRNA molecules themselves for that matter. Sometimes the regions excised by the spliceosome also have biological functionality, as for example in regulation of gene expression by RNAi interference, or antisense aRNA or miRNA signalling. However, nucleotides are mainly used as information carriers in the cell, while most of the things that actually get done are done by proteins. They serve a multitude of purposes, from very simple ones like structural elements (biological bricks if you like) to critical roles like selective molecular gatekeepers in the cell membrane making sure nutrients come in while waste goes out, while at the same time everything else stays where it's supposed to. Proteins are the tools, servitors, mediators and building blocks of the cell, so the most common product of gene expression is the messenger RNA molecule (mRNA). mRNA are used as templates for the

3A note on direction: The 5' and 3' nomenclature (five-prime and three-prime) stems from the enumeration of carbon atoms in the pentose sugar rings that are involved in the phosphodiester bindings that connect the nucleotide backbone (cf. Fig. 1.1 and Fig. 1.3). This nomenclature is confusing. Just accept it.
1.2. BIOLOGICAL SEQUENCES

ribosomes, which translate these sequences of nucleotides into sequences of amino acids. That is, they read the blueprints and build the proteins.

The first step of the translational process is that the ribosome binds to a specific region in the beginning of the mRNA, whereafter it scans the mRNA downstream toward the first occurrence of an AUG base triplet in the mRNA sequence. This triplet, or codon, signals the start of the protein coding region and is therefore often called the start codon. As can be seen in Table 1.1, AUG corresponds to methionine, so at this point translation will halt until a matching tRNA-Met with an attached methionine amino acid is brought to the ribosome.

In solution, tRNA sequences fold into a characteristic curled cloverleaf shape, where three unpaired bases at the tip of the central leaf are exposed, waiting to bind to their complementary codon in the mRNA. During translation, one tRNA at a time will bind to the mRNA at the current position of the ribosome, and the ribosome will then fuse the attached amino acid to the growing protein and send the spent tRNA away for recharging. And so translation progresses, codon by codon, elongating the nascent protein by one amino acid at a time. Fig. 1.4 shows the chemical reaction that creates the peptide bonds, and Fig. 1.5 shows the backbone structure of a short polypeptide chain. The ends of an amino acid sequence are labelled according to the groups that would bind another amino acid had the chain been longer, so the start is called N-terminal because of the nitrogen (N) containing amide group, and the end is called C-terminal because of the carboxyl group.

In Table 1.1, we can see that some amino acids are represented by several codons (which is why the genetic code is said to be degenerated), but more importantly, we can see that the three codons UAA, UAG and UGA have no associated amino acid, and it’s therefore impossible to continue translation past one of these codons. As a result, when any of these codons is encountered during translation, the finished protein is released from the ribosome and the translation process is terminated. These codons are therefore called stop codons.

The sequence from the start codon to the stop codon in the mRNA is called the coding region, while the flanking sequences are called non-coding regions, or untranslated regions (5'-UTR and 3'-UTR). These latter regions are still very important because they may contain sequences that act as signals to other cellular mechanisms, for example affecting the stability and degradation of the mRNA or even influencing translation.

Now, the mRNA template can be reused, and typically this is done in tandem as multiple ribosomes are processing the same mRNA molecule like beads on a string (cf. Fig. 1.6). Eventually though, the mRNA will be degraded by RNAase enzymes to ensure that the cell is not filled up with unnecessarily many copies of the protein.

So, a protein is a sequence of amino acid residues, generally hundreds in length, held together along a peptide bonded backbone. The amino acid residues can have quite drastically different properties, and quite substantial efforts have

---

4which is why amino acid sequences are also often called peptides. The word peptide is generally used in reference to shorter sequences or parts of sequences, while the word protein is generally used for long sequences, or even in reference to protein complexes (cf. below and Fig. 1.7).
CHAPTER 1. BACKGROUND

<table>
<thead>
<tr>
<th>Second base</th>
<th>U</th>
<th>C</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>UUU F Phe</td>
<td>UCU S Ser</td>
<td>UAU T Tyr</td>
<td>UGU C Cys</td>
</tr>
<tr>
<td>UGG L Leu</td>
<td>UCG S Ser</td>
<td>UAG * Stop</td>
<td>UGG W Trp</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>UCU L Leu</td>
<td>CCC P Pro</td>
<td>CAU H His</td>
<td>CGU R Arg</td>
</tr>
<tr>
<td>CGU L Leu</td>
<td>UCC S Ser</td>
<td>UAC T Tyr</td>
<td>CGC R Arg</td>
<td></td>
</tr>
<tr>
<td>CUG L Leu</td>
<td>UCA S Ser</td>
<td>UAA * Stop</td>
<td>CGG R Arg</td>
<td></td>
</tr>
<tr>
<td>CUA L Leu</td>
<td>UCC S Ser</td>
<td>UAG * Stop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCG L Leu</td>
<td>UCC S Ser</td>
<td>UAG * Stop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CUG L Leu</td>
<td>UCC S Ser</td>
<td>UAG * Stop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>AUU I Ile</td>
<td>ACC T Thr</td>
<td>AAA N Asn</td>
<td>AUG S Ser</td>
</tr>
<tr>
<td>AUC L Leu</td>
<td>ACC T Thr</td>
<td>AAC N Asn</td>
<td>AUG S Ser</td>
<td></td>
</tr>
<tr>
<td>AUA I Leu</td>
<td>ACC T Thr</td>
<td>AAA N Asn</td>
<td>AUG S Ser</td>
<td></td>
</tr>
<tr>
<td>AUG M Met</td>
<td>ACC T Thr</td>
<td>AAA N Asn</td>
<td>AUG S Ser</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>GUU V Val</td>
<td>GGU A Ala</td>
<td>GAO D Asp</td>
<td>GGO Q Gly</td>
</tr>
<tr>
<td>GUC V Val</td>
<td>GCC A Ala</td>
<td>GAC D Asp</td>
<td>GGC G Gly</td>
<td></td>
</tr>
<tr>
<td>GUA V Val</td>
<td>GCA A Ala</td>
<td>GAA E Glu</td>
<td>GGA G Gly</td>
<td></td>
</tr>
<tr>
<td>GUU V Val</td>
<td>GCG A Ala</td>
<td>GAG E Glu</td>
<td>GGG G Gly</td>
<td></td>
</tr>
</tbody>
</table>

Table 1.1: The genetic code.

Figure 1.4: Peptide bond formation. The peptide bond between two amino acids (dotted line) is formed through the removal of one water molecule, shown here in boldface before and after the reaction. The side chains of the amino acids are abbreviated as R1 and R2.

Figure 1.5: Backbone of a protein, with peptide bonds indicated with dotted lines.
Figure 1.6: Translation in progress. The picture shows an electron micrograph of multiple ribosomes (black blobs) bound to a single mRNA molecule (long strand, barely visible), producing multiple copies of a protein in tandem (short, thick strands extending from ribosomes). Ribosomes bind to the ribosome binding site at the 5’ end of the mRNA (the arrow to the right), and as translation progresses, the ribosomes move toward the 3’ end of the mRNA (left) as the amino acid sequence is progressively elongated, as is visually apparent in this picture. Note especially the differences in scale between the mRNA and the protein that it encodes. Image © The Nobel Foundation, used with permission.
be made to quantify their differences. Table 1.2 shows some of them, but this admittedly paints a rather crude picture of their complexity, since there are now over 500 different measures for their chemical, structural and physical differences [2].

When put in solution, intramolecular forces between the residues will start tugging at the chain; hydrophobic residues will seek security in numbers to escape the surrounding water, residues with opposing charges will attract each other, and so on, all forces dragging all the rest of the chain around with them. Often, there is one optimal configuration that minimises the internal stress, and this is generally the configuration that puts all the important bits in the right places for the protein to perform its intended function, putting the catalytic residues, cofactors binders, interaction surfaces et cetera in their proper place and in correct relation to each other. This process is called protein folding, and predicting the correct fold for a particular amino acid sequence is currently one of the biggest challenges in bioinformatics. There are many atoms in a protein and all of them interact, and all of the interactions contribute to the optimal configuration. This problem has dizzyingly many variables, with the number of interactions increasing quadratically with the number of atoms in the protein, making it very computationally expensive and only tractable for very small peptides.

The three-dimensional fold of an amino acid sequence is often referred to as the tertiary structure of the protein; the primary structure being the sequence of amino acids and the secondary structure its division into (comparatively) easily recognisable structural elements, like α-helices and β-strands. The quaternary structure of proteins refers to the composition and assembly of protein complexes, where multiple peptide sequences aggregate in specific ways, like cogs and pistons.
Figure 1.7: Protein structure. These two images show two different visualisations of the quaternary structure of human prostaglandin reductase 1 from the MDR superfamily. This protein complex is a homotetramer, meaning that it consists of four identical subunits with identical tertiary structure. The image to the left shows a ribbon representation of the peptide backbone in the four subunits, coloured by secondary structure element; α-helices in red, β-strands in yellow, and the loops connecting them in green. The image to the right shows the surface of the protein complex, as if it were grey and visible with the naked eye, showing the superficial atoms as tiny spheres. The source structure was obtained from the PDB protein structure database [3] (id: 1ZSV), and was visualised using Molsoft ICM Browser.
in a sophisticated piece of machinery, combining their functional surfaces in order to perform highly specialised tasks. Fig. 1.7 illustrates these concepts.

There are also a number of possible post-translational processing steps that further compound the protein structure prediction problem. For example; some residues may be affixed with additional functional groups, and some parts of the sequence may actually be cleaved away entirely. Furthermore, cysteine residues can bind to each other and form strong disulphide bridges between different parts of the sequence, and some parts of the sequence may even correspond to transmembrane segments, meaning that the neighbouring segments will be located on opposing sides of a lipid membrane. Some proteins also need transportation to reach their intended place of activity. When a protein is released by the ribosome it generally is just dropped off into the cytosol, but some proteins are needed only in specific sections of the cell, or even outside the cell, and it would be quite dangerous to release them just anywhere. For example, a peptidase meant to break down harmful proteins in the lysosome or an RNAase meant to attack viruses outside the cell could cause quite substantial damage inside the cell if they were unleashed prematurely. For this reason, such proteins have an internal targeting sequence that acts as a kind of combined biological address tag and transport safety pin. This tag is then cleaved off before the protein can carry out its potentially dangerous function. Sometimes, achieving the optimal fold is so difficult that even nature needs help to do it right, and in these cases chaperone protein complexes are employed; cellular body shops that specialise in straightening out misfolded proteins rather than wrecked cars.

Additionally, some segments of a protein are natively disordered, meaning that they have no defined three-dimensional structure under normal circumstances. These segments may for example lend essential flexibility to the protein, or fill crucial roles in DNA binding or other types of molecular recognition [4].

Furthermore, proteins continuously attain further modifications corresponding to different states of activation or modes of activity (phosphorylation of specific residues being a very common example), and often, even one such modification can release whole cascades of further modifications in other proteins in a rich, complex and intricately interconnected network of signals and feedback loops.

1.3 Evolution

The previous section detailed transcription and translation of nucleotide sequences, and showed just a glimpse of the awesome complexity of these fundamentals of life. One equally important piece of the puzzle that was intentionally left out is replication, the process where all the billions of nucleotide bases in the genome are meticulously duplicated. Generally, this is done as one of the steps in cell division where the cell creates copies of itself (called mitosis for us eukaryotes). For single-cell life forms like bacteria or yeast, this of course creates offspring; newborn, separate, single cell organisms. This consequently makes the generation time for such organisms very short indeed, measurable in hours rather than decades. For multicellular organisms like humans and giraffes however, replication is almost exclusively a means for tissue repair; healing wounds, combating disease and...
replacing old and worn-out cells. One important exception is meiosis, where gametes are created (sperm cells and ova). Rather than producing exact copies of the cell, meiosis produces cells that only have half the number of chromosomes – one chromosome from each pair. The purpose of these cells is of course to fuse into a chromosomally fully equipped cell as a result of sexual reproduction, giving rise to new baby giraffes and other adorable things.

While the previous section gave a grossly oversimplified view of the gloriously intricate transcription and translation processes, the previous paragraph neatly brushes aside the equally wondrous replication process in an almost criminal manner. However, this is all that will be said on the matter in the scope of this thesis. The rest of this section will be devoted to the instances where these things go wrong, and specifically the benefits of wrong-going.

Life, a broiling soup of opportunity and challenge, is a constant competition between organisms, and between the organisms and the elements, where the bigger and stronger often see themselves outflanked by the small and fastidious. Anything that can give an organism an edge in this competition will of course give it an increased chance of surviving long enough to have offspring, and thus puts more organisms like it into the world, and this is the basis of evolution.

There are many things that can give an organism an edge, like the ability to ingest a new type of food, or the ability to run faster than your prey (or your predator for that matter), or the ability to climb trees or steep slopes. All forms of specialisation help the organism to find and exploit their specific niche, but as overspecialisation can easily be a bane in a changing environment, adaptability is another clear edge-giving trait (cf. Fig. 1.8). Which adaptions are beneficial and which are deleterious are rarely readily apparent, but are rather emergent traits of the system that is the biological world as a whole.

But where do these adaptions come from? As mentioned in the previous section, the chromosomes hold all the inherited information in the cell, which means they contain all the genetic material passed on from parents to offspring. As should have been made somewhat apparent by the previous sections, the intricacies of the cellular mechanisms are vast and of a staggering complexity. In a system this large and complex, accidents are bound to happen. It is important to never forget that while these systems are often depicted in literature as neat boxes connected by precise arrows, they are in reality not driven by cold mathematical logic at all, but rather good old chemistry, which of course makes them susceptible to all the trappings of thermodynamic and quantum mechanical chaos. The DNA polymerases that replicate the chromosomes are not infallible, and once or twice in a trillion of bases, an error or two can slip past the proofreading. The DNA ligases that repair broken DNA may fuse the wrong bits together, or a retrovirus may insinuate altogether foreign genes into the sequence. Changes in the genome are called mutations, and the affected cell or organism is called a mutant. In principle, there are three types of mutations; deletions, insertions and substitutions, respectively corresponding to the removal, addition or replacement of a segment of the sequence. For example, if a polymerase in a rare act of rebellion

---

Interested parties are sincerely recommended to indulge themselves in a bit of self education on these subjects from other sources. It’s really interesting reading.
CHAPTER 1. BACKGROUND

Figure 1.8: Tardigrades are highly adaptable. They are microscopic animals capable of surviving extreme conditions like years of complete dehydration, temperature extremes from close to the absolute zero to well past boiling, radiation levels a thousand fold past the lethal dose for humans and, as was recently shown, in the vacuum of space [5]. Image from [6], used with permission.

puts a T in a chromosome copy where it by all rights should have put a G, and if in a moment of distraction this event slips by the proofreading machinery, this would give rise to a substitution mutation.

Mutations can have many different effects, most of them disastrous. For example if the error occurs in a gene, or even close to it, there are many ways in which this single error can cause total disruption of the functionality of the gene. For example it may change the promotor region or a transcription factor binding site so that its expression levels are nullified; it may change a splice site causing important exons to be erroneously excised; it may disrupt a ribosome binding site rendering the mRNA totally useless; it may disrupt a start codon or introduce a premature stop codon, truncating the protein; it may change the targeting sequence so that the protein ends up in the wrong part of the cell,\(^6\) it may cause a catalytic residue to be substituted for an inert one, effectively neutering the protein; it may substitute a polar residue for a hydrophobic one, causing the protein to misfold, and so and so forth. In such a finely tuned system almost any random change is bound to have a negative effect, and just as a fun exercise, try re-reading the previous section and think of instances where a single error could lead to catastrophic failure – there are myriads of them.

If this were a critical gene, the cell would not survive. In reality, many of the most important systems in the cell are fail-soft with multiple fallback solutions put in place in case accidents should happen, but a gene disruption will generally produce a weakened cell that is just that much less viable. For unicellular organisms like bacteria, this means that this particular offspring will do less well than others of its kind, and evolution will eventually see the mutant eradicated from the population. For multicellular organisms like ourselves, the situation is somewhat alleviated because only mutations that occur in gametes can actually be

\(^6\)which is the molecular biology equivalent of having the moving company dump off all your furniture at a random house in Lidköping instead of Linköping, as a result of sloppy handwriting.
1.3. EVOLUTION

passed on to the offspring, and additionally, there is hopefully an intact copy of the gene present in that other half of the chromosome pair that comes from the other parent. But the basic principle still applies; mutations generally lead to less viable offspring, which will have a harder time than their likes in the competition of life.

But then there is the flip side of the coin. Once in a billion trillions, the mutation may hit just the right spot on the chromosome to do the exact right thing in the exact right place. This may substitute a large and bulky residue for a smaller one, widening an access cleft in a catalysing enzyme and permitting the organism to digest another type of sugar, or perhaps to break down a dangerous toxin, which will in turn broaden the organism’s menu of admissible foodstuffs. Or it may cause the organism’s tendons to become more elastic, eventually permitting it to bounce at surprising speeds and at virtually no energy expenditure over large expanses of arid desert, allowing it to reach more of the widely scattered water holes for grazing. These are obvious benefits which will give the organism a substantial lead in the competition. Its offspring will prosper, and evolution will eventually see this mutation dominant in the population.

And then there’s the middle ground. Mutations that have little or no noticeable effect on the organism. The mutation may affect an unused gene, or pseudogene, in which case it has no effect at all. It may change a codon into another that encodes the exact same residue, called a synonymous mutation or a silent mutation, in which case it has no effect on the protein and it’s debatable whether or not it has any effect on the organism.\(^7\) Or it may change one peripheral amino acid residue into another quite similar residue, in which case it may change some aspects of some of the interactions that the protein participates in, but where the overall net effect on the cell, beneficial or harmful, will be negligible.

Some mutations can have unforeseen benefits, like for example the glutamate-to-valine point mutation in the sixth codon of \(\beta\)-globin that causes sickle-cell anaemia in humans, but also gives protection against malaria [8, 9]; or the 32 bp deletion in the human chemokine receptor gene CCR5 which confers resistance to HIV, and whose prevalence in Europeans suggests that it also may have constituted the edge their ancestors had for surviving the black plague during the medieval ages [10].\(^8\)

So, over generations the organisms will accumulate mutations. Most will affect non-critical sites,\(^9\) but some will be where it really counts. These will be groundbreaking and earth-shattering events, providing extraordinary new capabilities to the organism. But most will be minute, providing a slow and

\(^7\) Some widely used algorithms for quantifying evolutionary relations even operate under the assumption that synonymous mutations occur infinitely more often than non-synonymous mutations, and that seems to work. One example is protpars from the popular phylogeny package PHYLIP [7].

\(^8\) although some evidence points to the now eradicated smallpox pathogen as being the evolutionary driving force [11].

\(^9\) because poking at vital parts generally breaks stuff. Or in this case, will kill the organism, and the offending mutation will not be retained by its non-existent offspring.
gradual adaption to the conditions in the organisms’ respective environments. This is the process called evolution.

1.4 Origin of the genes

Now, there remains only one glaring omission that needs be addressed before we can proceed with the bioinformatics proper. The previous two sections have dealt with genes; their expression, function and gradual adaptation. But where do the genes actually come from?

The origins of the very first gene are of course lost in distant prehistory, and the only speculation on that topic that will be included in this thesis is that it stands to reason that a self replicating pattern could withstand the ravages of time and thermodynamics.\(^{10}\) And nucleotide sequences with their capability to dimerise and achieve stability through complementary hydrogen bonds are just that; self replicating patterns. Nucleotide monomers are naturally occurring, and the components of the cellular replicatory machinery are just sugar, catalysts rather than necessities. Once chance creates such a pattern it will be perpetuated, and evolution (on a molecular chemistry scale) will take it from there. But enough of this. Let’s concentrate on validatable things, like the emergence of new genes!

A new gene arises when an old gene is copied, either through speciation or gene duplication, and as the two copies gradually accumulate different mutations their functions will slowly begin to drift apart, and at long last a new gene is born.

Speciation is the emergence of a new species from an existing one, and can for example occur when two populations from the same species get separated by a barrier. As generations pass and the two populations separately adapt to their respective conditions, they will gradually accumulate mutations as described above, genetically drifting apart to the point where mating between individuals from different populations would no longer produce fertile offspring, at which point the populations are defined to be separate species.

Most genes in the new species will at this time only exhibit subtle differences compared to their counterparts in the original species, as they are in both instances under evolutionary pressure to keep working and serving the organism as well as possible. Thus, important bits like catalytic sites, cofactor binders and regulatory sites are likely to remain unaffected, and are therefore generally well conserved in evolution. In contrast, non-important segments such as for example linker regions will accumulate mutations at a faster rate, and are consequently often poorly conserved. The pattern of conserved and non-conserved positions generally becomes apparent when comparing a number of genes that are believed to be related, and this of course gives a pretty good idea of what parts are essential and which are less important. It is also possible to quantify the evolutionary distance between different species by comparing their genes, and from this it is possible to reconstruct their relations and their relative positions in the tree of life (so-called phylogenetic trees, cf. Fig. 1.9 and 1.10). It is of course also possible to construct phylogenetic trees for related proteins or genes. This subject will be further explored in sections 2.2 and 2.3.

\(^{10}\)but I can highly recommend reading some of the existing theories [12–14]. It’s interesting stuff!
Figure 1.9: Tree of life, as understood by Ernest Haeckel and published in his book "evolution of man" in 1874. Humans are found at the top of the tree, quite close to the chimpanzees.
Figure 1.10: Tree of life, as deduced from comparisons of completely sequenced genomes. In this illustration, the root is in the middle and the branches radiate outwards in all directions, much like a tumbleweed. Bacteria are shown in blue, archaea in green and eukarya in red. Humans are found at the top of the red field, quite close to the chimpanzees [15,16].
1.5 Homology

Genes related through one or more speciation events are called orthologues, and this relation can be thought of as the molecular biology equivalent of being direct descendants from the same ancestor. Orthologous genes are quite likely to retain the same function and be regulated in the same manner, like for example class I alcohol dehydrogenases in mouse and human.

Gene duplication is exactly what it sounds like. Somehow, an organism ends up with two copies of the same gene. This can for example occur when a replication enzyme makes a (quite literal) slip-up and accidentally writes the same gene twice into the new chromosome copy. Contrary to the orthologue situation, the new gene will in this case be under no evolutionary pressure at all. Since the fully functional original still remains and can carry out all its duties, all the one-error-kills-organism restrictions from before no longer apply, and evolution is left with free reins to play around with this new and redundant sandbox of opportunity. Consequentially, these genes generally evolve very quickly, either toward attaining new and exciting functions and specificities, or mutating beyond all utility into oblivion.

Genes related through one or more gene duplication events (and additionally however many speciation events) are called paralogues, and while paralogous genes are likely to share many properties, they often differ in crucial aspects like specificities, regulation or interactions. Prostaglandin reductase 1 from human and cinnamoyl alcohol reductase 1 from tobacco are for example paralogues. Both are dehydrogenases / reductases and share the same fold, but that’s about it.

Genes that are somehow related, sharing a common evolutionary predecessor, are called homologues. The word homologue is frequently used analogously to the word cousin, or relative, in the sense that genes are often referred to as being close or distant homologues. Orthologues in closely related species are of course the closest homologues, while paralogues from distantly related species are more distant homologues.

The astute reader is probably wondering at this point if it makes sense to talk about homology at all. If all new genes are derived from existing predecessors, does it really make sense to speak of non-homologous genes? Pushing the envelope, does a non-negatable term really have any utility, or is homology just a buzz word? It is of course conceivable that the first genes actually could have arisen separately, and that would then make their respective descendants internally homologous but non-homologous between groups. However, establishing which genes stemmed from which hypothetical primordial predecessor would be ludicrous – both impossible and useless – and this again leaves us with an ineffectual homology definition. No, an amendment is in place.

The word homologous is most often implied to mean that two sequences are similar enough that it is unlikely to think that the observed similarities could have come about from just a series of random mutations, or in other words: they are more similar than you could expect to observe in random noise.

With these definitions, we can start grouping homologues together. Just as a gardener can prune a tree by cutting off a branch at a suitable length, so can we
cut a branch off a phylogenetic tree. The gardener then ends up with a collection of connected leaves, while the bioinformatician ends up with a group of related homologues. For a small branch, with close homologues, one could expect many features of the proteins to be identical (specificities, regulation or interactions...), and one could call this group a protein family. A larger branch, possibly with many small branches attached to it, would then connect more distant homologues from different protein families into a protein superfamily, and one could expect to find less shared properties in this larger group (function, structure...). A very large branch, or an entire tree, could then be said to connect even more distant homologues from different superfamilies into something called a fold, where indeed only the general three-dimensional fold of the protein is conserved.

Anyone who has ever gardened knows how hard it is to determine where exactly to cut a branch in order to achieve the most pleasing result. There are no magic signs and no indisputable definitions to determine the exact placement of the optimal cut. In the end, there is only wood and judgement. The same is true for the definition of a protein family. Sometimes, the data may make the choice appear obvious, but almost always some marginal cases will remain uncertain, or new data will challenge old decisions.

Nevertheless, grouping protein together at different levels is highly useful, not only because it brings structure to the growing mountains of sequence data produced worldwide, but also because it allows scientists to draw new conclusions from emergent features of the groupings. When studying a protein, it is very useful to investigate what features are present among its closest homologues, in its family, in its superfamily and in its fold. Comparisons at different evolutionary levels can also help trace the evolutionary history of the protein, and put observed functional differences in context.

But now, with these definitions finally sorted out, we're properly equipped to start talking bioinformatics!

1.6 Bioinformatic challenges

As we've seen so far in this chapter, pretty much everything that happens in the cell involves interaction between biological sequences, and the nature of these interactions is determined by the characteristics of the sequences. The previous sections are brimming with examples. The ribosome binding site on the mRNA molecule for instance; in itself it is just a stretch of phosphodiester bonded nucleotides like any other, but its specific sequence of nucleotide bases gives it a high affinity for binding ribosomes. A polyadenylation signal at the end of an mRNA, typically a simple AAUAAA sequence, will cause the mRNA to be extended with a poly-A tail, where hundreds of adenyl nucleotides are added to the end of the sequence. This of course greatly increases the time it takes for the RNAases to degrade the mRNA, proportionally increasing the number of translated proteins that it will give rise to in its lifetime. Another example are the typically 60 base pair long SECIS elements that can cause the stop codon UGA to be translated to selenocysteine by the ribosome, rather than causing translation to terminate. Intronic splicing segments and promotor regions are
further examples; seemingly innocuous and nondescript regions of nucleotide sequence, but whose specific sequences give them function and meaning. And proteins are no different. Targeting sequences, post-translational modification sites, enzymatic cleavage sites, interaction surfaces and even tertiary structure; all emergent systemic characteristics encoded in their amino acid sequences.

If it is true that the driving force behind all these emergent traits is hidden in plain sight in these sequences, then if we could learn the language of nucleotides and amino acids, it would allow us to read the genomes as if they were books, or even user manuals. Just as computer forensics use sequence analysis to pry knowledge from broken hard disk drives, bioinformaticians do the same to nucleotides and proteins, uncovering more and more of their strange language along the way.

The main challenge is of course diversity and variation, the very thing that drives evolution forward. There is no such thing as absolutes or standards in the realm of biology, but rather tendencies and correlation. Consequentially, bioinformaticians generally rely on comparative studies, using homology and conservation to find these tendencies and correlations. The reliability of the results for such methods is of course directly influenced by the amount and quality of the data used in the comparisons, so it is of course desirable to incorporate as much data as possible in each study. However, larger amounts of data are of course more time consuming to process, requiring faster computers and more clever programming. The algorithms for analysing the data are also continuously being improved or replaced by more advanced successors, always increasing reliability but at the same time also increasing the need for computational power.

At the same time, the methods for collecting the data are continuously being improved and allow progressively more ambitious projects to be undertaken. We have now not only managed to sequence the human genome [17], but also moved beyond sequencing single genomes. Variability within the human genome is now being investigated by sequencing the genomes from 1000 humans [18]. Furthermore, in the new field of metagenomics, the focus is not even on sequencing individual species, but rather sequencing as much as possible of the nucleotide content in a specific environment, such as a scoop of Texan soil [19] or a filtrate of Sargasso sea water [20], for the purpose of finding genomic data on species that are impossible to culture.

Consequentially we see an almost exponential growth in the amount of available data in all publicly available data repositories. Managing all this data is in itself becoming a challenge, which in turn is compounded by our own progress, because as we uncover more of the secrets of biology we learn more about how we should have stored the data in the first place. Biology is still full of exceptions and surprises, and dogma are continually being upended and reevaluated.

Bioinformatics is therefore a field of changing data on entities in change that uses changing tools. It is a field that, just as its subject, constantly evolves. It’s a field where groundbreaking discoveries can be made. It’s all very interesting.
Chapter 2

Materials and Methods

The materials in bioinformatics are of course the biological data. Currently we have many publicly available repositories that hold the amassed results and derivative data from biological experiments around the world; sequences, structures, sequence motifs, and many other fascinating things. The following sections will first describe some of the popular databases in general terms and then move on to describe the methods employed to use or create the data, focusing on the methods used in the research underlying this thesis. Since the main focus of this thesis is on proteins, any sequences discussed in this chapter are implied to be amino acid sequences, unless otherwise is explicitly stated.

2.1 Bioinformatic databases

2.1.1 Sequence databases

There are many sequence databases currently available online, ranging from public repositories for newly sequenced genomes to highly curated specialist databases for specific fields, and there is often a trade-off between data quantity and quality. EMBL-Bank [21], NCBI GenBank [22] and DDBJ [23] are the three major repositories for nucleotide sequences, which accept direct submissions and serve to make the raw data available to the public as soon as possible. They also synchronise all submissions to ensure data harmonisation and availability. The August 2010 release contained over 195 million sequences, comprising over 292 billion bases. These are accessible in bulk or by taxonomic subdivision, such as Bacteria, Invertebrates, Viruses or Primates.

UniProtKB [24] is the universal resource for protein sequence information. It started in 2002 as a collaboration between the European Bioinformatics Institute (EBI), the Swiss Institute of Bioinformatics (SIB, originators of Swissprot [25]) and the Protein Information Resource (PIR) [26] in an attempt to coordinate annotation efforts and increase data harmonisation. It has has two subsections, Swissprot and TrEMBL. Swissprot was originally a separate database, which was launched already in 1986, and as such is one of the oldest databases in bioinformatics. It
contains manually reviewed and annotated sequences and is considered highly reliable. However, manual curation speed is orders of magnitude slower than the production rate of new sequence data, so TrEMBL was created to fill that gap. TrEMBL contains unreviewed, automatically annotated sequences, mostly in silico translations of coding sequences from EMBL-Bank. In the October 2010 release Swissprot contained just over 521 thousand sequences comprising close to 184 million residues, while TrEMBL contained over 12 million sequences comprising over 3.9 billion residues.

Ensembl [27] is an eukaryote centric genome database that provides genomic mappings for known and predicted sequence features, such as genes, transcripts, splice variants and proteins. As the human genome was sequenced it became apparent that it would be unfeasible for any team of scientists to perform this mapping manually, so the Ensembl project was undertaken with the purpose of constructing an annotation pipeline that could do this automatically [28, 29]. More genomes have since been added and the August 2010 release contained genomes for 50 species. The Ensembl software has also been put to use for genomes from other taxonomic divisions in the EnsemblGenomes project, providing the same functionality for 23 metazoan, 8 protist, 12 fungi, 8 plant and 183 bacterial genomes (July 2010 release) [30]. Having complete genomes available is of course very valuable, because it allows bioinformaticians to draw confident conclusions based on the absence of expected sequences in different species.

The Comprehensive Microbial Resource (CMR) [31] at the J. Craig Venter Institute is a database that provides a unified access point to all publicly available fully sequenced bacterial genomes.
2.1. Functional classifier databases

As previously noted, as the constantly accelerating influx of new sequence data continues to outdistance our meager human capacity for manual curation, automated annotation becomes ever more important. Many applicable techniques exist, and some of them will be covered in more detail below, however the general idea is the same for all of them; generate a model that describes a particular trait of a sequence and use the model with a new or unknown sequence to make a prediction of whether or not that particular trait is present in the sequence. The trait in question may for example be a sequence feature like an active site or a signal sequence, or something more abstract like membership of a certain protein family, or general fold of the protein.

When assembled into a larger library, these models have synergistic value, as a single search against the library is likely to produce a wealth of information even for a previously unknown sequence. Pfam [32] is one such database, which uses HMMs (cf. section 2.8.1.2) to detect protein domains and family membership. Its high-quality PfamA section contained close to 12000 HMMs in the October 2009 release. PROSITE [33] is a similar database that uses HMM like profiles and pattern recognition (cf. section 2.5) to detect protein domains and smaller motifs like active sites of post-translational modification sites, and the April 2010 release contained just over 2000 entries.

InterPro [34] is an example of an integrative resource that presents a unified view of models collected from other sources, such as Pfam and PROSITE. The October 2010 release contained over 21000 entries, corresponding to for various sequence traits like active sites, binding sites, domains and repeats.

2.1.3 Other databases

The Genomes OnLine Database (GOLD), shown in Fig. 2.1, contains information on complete and ongoing genome projects as well as information on metagenomic projects [35].

The Worldwide Protein Data Bank (wwPDB) is a unified database of three-dimensional structure data on macromolecules, mainly protein structures [36]. It was initiated as a collaboration between molecular structure centres including the well known RSCB PDB [3], and its purpose was to harmonise annotation and formats and to ensure open and free access to structural information on an equal basis across the globe. As of June 2010 the database contained just over 65000 structures.

The American National Centre for Biotechnology Information (NCBI) hosts a large number of databases; some counterpart the ones described above, but there are also a number of additional databases of special interest for this thesis [37]. While not claiming to be a authoritative resource on taxonomic information, NCBI Taxonomy [22] provides a very useful and structured ordering of organisms which is referenced by many other databases, such as UniProtKB and EMBL-Bank. As of June 2010, NCBI Taxonomy lists over 160000 species, where each catalogued species and species group is assigned a distinct identifier. This helps reduce confusion in cases where species (or some subspecies) would otherwise be
referred to using their (often abbreviated or misspelled) Latin names, and which themselves have a high degree of synonymy. Online Mendelian Inheritance in Man (OMIM) is an NCBI database describing disease related genes and diseases with genetic components, and which contained information on just over 20000 entries as of June 2010 [38].

2.2 Sequence alignment

Sequence alignment is just what it sounds like: matching up sequences to find their best fit. Conceptually, this is done through sliding one sequence along the other and noting which offset gives the most similarities. It is often necessary to “stretch” one or both sequences at different places in order to find the optimal fit, introducing imaginary “gaps” corresponding to insertions or deletions in either sequence. This results in what is called a gapped alignment. You can do this for two sequences at a time and get a pairwise alignment, or for many sequences at once and get a multiple alignment. The optimal alignment will ideally show which residues have corresponding roles in the different sequences; having the same position in tertiary structure, fulfilling the same part in the biological function, and so on. This can of course be used for many purposes, not only in determining homology but also for inferring functional features from well-studied sequences onto new and unknown sequences.

The alignment itself is generally done programmatically by assigning score awards to matches and penalising mismatches and gaps [39]. Many algorithms employ an expensive gap opening cost and a lower gap extension cost, reasoning that larger insertions or deletions are not that much more unlikely than short ones. Finding the best fit then becomes the optimisation problem of maximising the score, which is most often solved using dynamic programming, along with heuristics in order increase execution speed.

For nucleotides (transitions and transversions) the scoring scheme is quite straightforward, awarding 1 point to matches and –1 point to mismatches. For amino acid residues however it is also necessary to take into account which the matching or mismatching residues are in order to assess the impact on the protein. Substitution scoring matrices such as BLOSUM [40] or PAM [41] are used to quantify the similarities (cf. Fig. 2.2), awarding the highest scores to identical residues and the lowest scores to very dissimilar residues (usually as determined from observed substitution frequencies, mutational distance or physicochemical properties).

2.2.1 Pairwise alignment and homology search

Pairwise alignments are most often used for searching sequence databases for homologues to unknown sequences, for example using the BLAST [39] or FASTA [42] programs. While the alignment score is as good an optimisation target as any for this purpose, it is less useful as a quality measure for the results. For example, longer sequences obviously have a higher chance of accumulating a higher score, and as databases grow and more unrelated sequences are added, more sequences
2.2. SEQUENCE ALIGNMENT

Figure 2.2: The BLOSUM62 substitution matrix [40]. Each row and column lists scores for substituting one amino acid residue for another, and the diagonal (bold) shows the score awarded to conserved residues. For example, having a conserved cysteine residue (C) yields 9 points, while substituting it for aspartic acid (D) gives a −3 point penalty. Substitution scoring matrices are symmetrical, meaning that the part under the diagonal (grey) is a mirror image of the part over the diagonal. B, J and Z denote pairs of residues that are indistinguishable from one another (B: asparagine or aspartic acid, J: leucine or isoleucine, Z: glutamine or glutamic acid). X denotes residues that have not been possible to determine during sequencing.

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

Figure 2.2: The BLOSUM62 substitution matrix [40]. Each row and column lists scores for substituting one amino acid residue for another, and the diagonal (bold) shows the score awarded to conserved residues. For example, having a conserved cysteine residue (C) yields 9 points, while substituting it for aspartic acid (D) gives a −3 point penalty. Substitution scoring matrices are symmetrical, meaning that the part under the diagonal (grey) is a mirror image of the part over the diagonal. B, J and Z denote pairs of residues that are indistinguishable from one another (B: asparagine or aspartic acid, J: leucine or isoleucine, Z: glutamine or glutamic acid). X denotes residues that have not been possible to determine during sequencing.
Figure 2.3: A BLAST [39] pairwise alignment between human prostaglandin reductase 1 (QUERY) and alcohol dehydrogenase from the bacterium Staphylococcus aureus (SBJCT). The line between the QUERY line and the SBJCT line shows the detected similarities, where letters indicate identical residues and + characters indicate highly similar residues. Gaps, denoted with – characters in either sequence, indicate either a deletion in the sequence or an insertion in the other sequence. The sequences come from different protein families and different species, but both protein families are members of the MDR protein superfamily (medium-chain dehydrogenases / reductases). In other words; they are homologous, but not very close, and the the E-value of 0.001 reflects this. The alignment is an excerpt from a homology search against the UniProtKB/Swissprot sequence database [24].

have the potential to achieve decent scores just by chance. Defining a minimum score for probable homology is nontrivial because it depends on many variables that are variant between databases and query sequences. The E-value was developed to help mitigate this problem [39], and it does this by integrating factors like database size and composition to present a stable confidence value for the comparison. To be exact, the E-value can be defined as such:

\[
E = \frac{2L}{n} \left( \frac{1}{e^2} \right)
\]

The E-value, or expectation value, is a statistical estimate on the number of non-homologous sequences that can be expected to achieve an equal or higher score in the same search, purely by chance.

Fortunately, while this concept is fairly complicated, it is also really simple to use.

Example 1: We search a database for homologues to a sequence, and one of the matches is assigned 200 points and the E-value 10. This means that the score 200 is in fact so bad that we should expect to see 10 non-homologous sequences turn up in this search with scores in excess of 200. This indicates that the match is very bad and we should probably disregard it.

Example 2: Now, what if the E-value would have been 10^{-6} instead? This would mean that the score 200 is in fact so good that we should only expect to see one millionth of a non-homologous sequence turn up with equal or better scores in this search. That is of course a very low number and we can be confident that this is a good match.
2.2. SEQUENCE ALIGNMENT

In other words, the E-value makes a statement on how much credence you should put on a given match under the current conditions, and the lower the E-value, the better. It is of course not practical to statically define a universal cutoff for acceptable probability of homology, because this is very much determined by the needs in the current experiment. As a rule of thumb however, $10^{-6}$ can be passable for close homology while $10^{-3}$ could indicate more distant relations, and matches with E-values above 1 are almost certainly unrelated. As a reference for what E-values imply in terms of visible sequence similarities, Fig. 2.3 shows a BLAST alignment with an E-value of $10^{-3}$. Because it is so straightforward to use, and because it is reasonably invariant between databases and searches, the E-value is currently the most widely used reliability measure for homology searches.

PSI-BLAST is an extension to the BLAST algorithm, which strives to increase sensitivity to remote homologues [39]. Here, a “pairwise multiple sequence alignment” is built for the query sequence using the matching database sequences, and the observed frequencies for each residue type in each position are recorded to a position specific scoring matrix, or PSSM. PSSMs are frequently called profiles, since they can be seen as general descriptions of the amino acid compositions in the working set of homologous sequences. The database search is then reiterated using the PSSM instead of the normal substitution scoring matrix, in each reiteration collecting additional, more distantly related sequences to the result and the PSSM. The inclusion of new sequences can either be done manually or according to a user set E-value threshold. The procedure is terminated at convergence or after a predetermined number of iterations.

2.2.2 Multiple sequence alignment

Multiple sequences alignments, MSAs, are usually built from a number of homologous sequences, for example a protein family as shown in Fig. 2.4. Using multiple sequences improves the quality over pairwise alignments simply by reducing the risk that random mutations will cause misalignments, effectively increasing the signal-to-noise ratio. However, aligning many sequences simultaneously is a multidimensional optimisation problem which is computationally expensive to solve. Fortunately, there are many different approaches for quickly finding an acceptable solution, however most often there is a trade-off between speed and quality of the results. At one end of the spectrum lie the so called progressive methods, which start by aligning the most similar sequences and then progressively align the resulting alignments. These methods tend to be the quickest and least computationally expensive. One representative of this group is the classic ClustalW program [44]. At the other end of the spectrum we find the iterative methods, which continually reiterate the alignment process, finding and improving poorly aligned parts in each iteration until an optimum has been reached. MAFFT L-INS-i is an example of such a method [45]. Other examples of MSA algorithms include dialign [46], POA [47], Kalign [48], MUSCLE [49] and finally T-Coffee which is an integrative method that combines the results from multiple other alignment algorithms to create the final consensus alignment [50]. Some

\[ \text{and of course the size of the comfort zone of the scientist in question.} \]
Figure 2.4: An MSA of a small MDR protein family, with some additional visualisations. The large multicoloured area occupying most of the view above shows the full MSA, encompassing 21 sequences with an aligned length of 290 positions, as can be seen from the rulers at the bottom and to the left. The residues have been coloured according to biochemical properties (using faded ClustalX [43] like colours), and gaps are clearly indicated in blue. The identifiers for the sequences are shown to the left. Below the MSA is a large blue bar that shows the location of the cofactor binding domain (cf. back cover illustration). Between this and the bottom ruler is a bar chart that shows the rate of conservation at the different positions in the alignment, where totally conserved positions are indicated in green. The horizontal bar chart to the left shows sequence conformance, e.g. how well individual sequences fit in with the rest (cf. paper V). The image below shows a smaller portion of the same MSA. Images produced using MSAView, presented in paper V in this thesis.
2.3. DENDROGRAMS

Figure 2.5: Time complexity of four MSA algorithms. N denotes the number of sequences and O indicates the time complexity. \( O(N^2) \) means that the algorithm scales with the square of the number of sequences used, or that twice the amount of sequences will take four times as long to align. T-Coffee failed to complete with more than 40 sequences on the test machine due to its higher memory complexity.

algorithms like Praline [51] or Expresso [52] strive to increase alignment quality by including additional information on the sequences, like secondary or tertiary structure, or transmembrane topology.

While high quality is often desirable, care has to be taken when choosing an MSA algorithm. As can be seen in Fig. 2.5, not only can computation times differ several orders of magnitude between different alignment algorithms when aligning hundreds (or thousands) of sequences, but memory consumption can be a very real obstacle when using more demanding algorithms.

MSAs are frequently used as starting points for many bioinformatic methods, but can also be used for investigating relations between sequences as well as determining properties of different positions and regions in the sequences, as will be shown in the following sections.

2.3 Dendrograms

Neighbour joining dendrograms (as shown in Fig. 2.6 and on the cover of this book) are a quick way of showing relations between sequences. They are based on MSAs and can be built using for example ClustalW. The MSA is used to compute estimates for the evolutionary distance between the sequences. These distances are then used to join neighbouring sequences into clusters, and eventually joining neighbouring clusters into larger clusters. The join order and distances are then used to build a tree representation of the possible speciation and gene duplication events that could have produced the sequences.

In order to assess the reliability of the dendrogram, a technique called boot-
Figure 2.6: A neighbour joining dendrogram built by ClustalW based on the MSA in Fig. 2.4 and visualised using Dendroscope [53]. Bootstrap values are indicated at each branch point, and the sequence distance scale (top, left) indicates the length corresponding to ten percent difference in sequence identities.
2.4. CONSERVATION

Conservation is one of the properties that is quantifiable from an MSA (cf. Fig 2.4). As previously stated, conservation is closely related to biological importance, so a highly conserved residue or region can be expected to be important for the protein’s structure, function or interactions, while linkers or regions whose mere purpose is to hold the important parts together can be expected to be notably less conserved.

Fig. 2.4 shows an MSA of a small MDR protein family. The rate of conservation for the positions in the alignment is shown using a bar chart just below the alignment, using a colour gradient from red (poorly conserved) to yellow (well conserved) for emphasis. Some positions are indicated with green in the bar chart, namely those that are strictly conserved. These are the positions that are likely to be functionally important for the protein family. On the other hand, a position may be poorly conserved in the alignment as a whole while it is in fact highly conserved in distinct subgroups in the alignment, albeit to different residue types. Such positions can provide valuable insight into the mechanisms behind differences between the subgroups, and are said to be differentially conserved.
CHAPTER 2. MATERIALS AND METHODS

2.5 Pattern recognition

Most patterns in biology are staggeringly complex, but in some cases they can be really simple. For instance, an asparagine residue is likely to become glycosylated if it is followed by a tripeptide where the first and last position hold any residue but proline and the second position holds serine or threonine [55]. This pattern can be expressed as N–{P}–[ST]–{P} using PROSITE notation [33], and scanning sequences for matching motifs is very easily done for example using any regular expression aware software. Pattern recognition is a comparatively very fast technique, however it is nontrivial to evaluate the reliability of the detected matches. Additionally, many patterns are short and therefore likely to match randomly in sequences purely due to chance. It is possible to alleviate this problem using domain knowledge, as exemplified by the Eukaryotic Linear Motif resource (ELM) [56] which uses pattern recognition similar to PROSITE augmented with a series of filter rules which decrease the amount of spurious matches. These can for example block matches for interacting motifs to segments that are deeply buried in the three-dimensional structure of the protein, or block post-translational modification motifs from matching proteins located in cellular compartments that are known to lack the modifying enzyme in question.

2.6 Support vector machines

Many real life problems are very complex. A more elaborate wording of this truism would be to state that many real life outcomes and observable traits are governed by a multitude of underlying factors, and uncovering the relevant factors and determining the degree of their importance is very often nontrivial.

One way of doing it is to measure as many variables as is practically possible on as many samples as is practically possible, and then afterwards try to see trends in the measurements, that is, try to assess which of the properties that have the highest influence on the trait of interest. This is an exercise in multivariate data analysis, and multidimensional data is a tricky subject for the human mind. Support vector machines (SVM), often coupled with kernel methods, are a popular way of dealing with this kind of problem. Both subjects will be superficially covered in this section, but are described in rigorous detail for example in books on statistical learning theory by Vapnik [57], or Schölkopf and Smola [58].

Let’s for example say we need a quick way of telling if an apple will be delicious or disgusting, and obviously, we do not want to taste it to find out. We could then measure a number of easily determined variables (size, colour, weight...) for a large enough number of apples, and let an expert panel of judges determine whether or not the apples are tasty. An SVM could then take the measurement vectors and verdicts for the samples and find the decision hyperplane (the diving line\(^3\)) that separates the two classes by the widest possible margin, as shown in Fig. 2.7.

\(^2\)or as a Perl-like regular expression: N[^P][ST][^P]

\(^3\)A hyperplane in two dimensions is a line, just as shown in Fig. 2.7. A hyperplane in one dimension is a point, like the 0°C on a thermometer that divides freezing and melting temperatures; and a hyperplane in three dimensions is a plane surface, like a large tarpaulin stretched taut overhead
Figure 2.7: A trivial SVM classification example. For a number of samples (markers in the plot), a large number of variables have been measured (in this case only two, $v_1$ and $v_2$). The two classes of samples (circles and squares) are separated by the decision hyperplane (solid) and with the maximum possible margins (dotted lines). The support vectors (the samples on the margins) are filled, and misclassified samples (on the so-called soft margin) are coloured grey. The samples could for example be apples, and circles could represent delicious apples while squares could represent disgusting apples. The measured variables $v_1$ and $v_2$ could for instance be size and redness; implying that small apples can be tasty if they are red enough, and green apples can be tasty if they are big enough, and big and red apples are sure to be delicious. Note that some of the apples will be misclassified by the decision hyperplane in this example, possibly on account of factors (or combinations of factors) not covered in the present study, such as wrinkledness or wormholedness.

2.7. Now we can repeat the same set of measurements for hundreds or thousands of batches of apples and be able to make a confident guess on the tastiness of each apple simply by determining which side of the decision hyperplane its sample vector falls on.

Determining the optimally separating hyperplane is of course an optimisation problem, and it can be shown that the optimum can be found by solving [58]:

$$\maximize_{\alpha} \quad W(\alpha) = \sum_i a_i - \frac{1}{2} \sum_{i,j} a_i a_j y_i y_j (x_i \cdot x_j)$$

subject to $\quad a_i \geq 0, \quad \sum_i a_i y_i = 0$

where $(x_i \cdot x_j)$ is the dot product of sample vectors $i$ and $j$, $y_i \in \pm1$ is the class of sample $i$ (for example +1 for delicious) and $a_i$ are the Lagrange multipliers introduced to solve the problem.

In the solution to the above problem, $a_i$ will be zero for all sample vectors that are not located exactly on the maximal margin. In other words, the final location of the hyperplane will be determined only by the sample vectors that are closest to the opposing class, which for this reason are called support vectors, lending their name to the technique [58]. Most often, the number of resulting support vectors is much smaller than the total number of samples.

Sometimes it will however not be possible to find a hyperplane that cleanly in a deluge, separating dry places from places that will get wet. Imagining higher dimensionality hyperplanes is left as an exercise for the reader.
separates the classes. This may for example be due to misclassified samples or overlapping classes, which of course are everyday occurrences in all real world applications, and especially in the complex field of biology. Using so-called soft margins, it is possible for SVM to produce reliable results even under reasonably difficult conditions. The only difference in finding the decision hyperplane is in this case an additional constraint that \( \alpha_i \leq C \), where \( C \) is a user set parameter that determines the trade-off between margin maximisation and error tolerance [58].

Determining which side of the decision hyperplane a new sample falls on can be done using the decision function

\[
f(x) = \text{sign}\left( \sum_i y_i \alpha_i (x \cdot x_i) + b \right)
\]

where \( b \) is an offset given by averaging

\[
b = y_{SV} - \sum_j \alpha_j y_j (x_j \cdot x_{SV})
\]

for all support vectors \( x_{SV} \) (such that \( a_{SV} < C \), if soft margins are used) [58].

A distinguishing feature of the equations presented above is that the optimisation target function \( W(\alpha) \) as well as the decision function \( f(x) \) are only dependent on the sample vectors through pairwise dot products \((x_i \cdot x_j)\). This allows us to apply kernel transformations to the data, replacing the dot products with another similarity measure; a kernel function

\[
k(x_i, x_j) = (\psi(x_i) \cdot \psi(x_j))
\]

where \( \psi(x) \) is a mapping function from sample space to a (generally higher dimensionality) feature space where separation is more likely to be achievable [58].

This is called the kernel trick. It is a means to achieve separation even for seemingly completely inseparable classes by increasing the dimensionality of the data and exploiting relations between variables. For instance, combining weight and size gives density, which could conceivably be a key factor in apple tastiness, since density deviations could indicate over or under ripeness. However, for density there is likely a (quite literal) sweet spot where apples are truly tasty, and a decision hyperplane in sample space would by definition only be able to exclude either overripe or underripe disgusting apples from the delicious ones, and never be able to exclude both kinds at once. Fortunately, this kind of nonlinear classification is also feasible using an appropriate kernel.

A useful property of kernel functions is that only the pairwise dot products between mapping functions need be computed. This means that the hyperplane placement and decision making in feature space will be implicit, and we are free to choose very exotic (and useful) mapping functions that may still have dot products that are very simple to compute.\(^4\) For example, the popular polynomial kernel

\[
k(x_i, x_j) = (x_i \cdot x_j)^d
\]

\(^4\)In fact, the mapping functions themselves need not even be computable.
maps the sample vectors into a feature space where the dimensions are comprised of all possible monomials of degree $d$ of the sample variables, increasing the dimensionality to $(n+d-1)$ for $n$ measured variables [59], while the commonly used Gaussian radial basis function (RBF) kernel

$$k(x_i, x_j) = \exp \left( -\frac{||x_i - x_j||^2}{2\sigma^2} \right)$$

has an infinite dimensional feature space [58]. It is also possible to use still more elaborate kernels, for example for variables that are not even numerical. One could for instance let $k(s_i, s_j)$ represent the number of shared subsequences between two biological sequences $s_i$ and $s_j$ [60].

Beyond classification, it is also possible to use SVM for regression [61]. This means that we could let our panel of judges score the apples, for example on a scale from 1 to 10, and have the SVM machinery produce similar assessments for new, untasted apples.

SVM with kernel and regression support have been implemented in the software packages SVMlight and LIBSVM [62–64]. Both are comparatively easy to use even for non-mathematicians, and are also available as libraries that can be included from a variety of programming languages.

DISOPRED2 is an SVM driven bioinformatic algorithm [4] which predicts segments of proteins that are likely to be natively disordered. The sample vectors in this case are 15-position windows of PSI-BLAST PSSMs, and the training samples used to find the decision hyperplane were collected from PDB structures whose states of disorder have been determined using high-resolution crystallography.

### 2.7 Artificial neural networks

Nerve cells, or neurons, are what make up our entire nervous system; brain and peripheral nerve centres as well as sensory and motor nerves. Fig. 2.8 shows a neuron, and prominently, its most notable features – its dendrites and its single axon.

A neuron works by integrating electrical impulses received by its many dendrites, and depending on the combined strength of the incoming impulses, the neuron will become activated and propagate the signal along its axon. The axon can then branch out and connect to dendrites of other neurons, which may then in turn propagate the signal further through the nervous system.

The complete nervous system consists of a vast structure of layer upon layer of interconnecting neurons, so while the basic principles for nerve impulse propagation are very simple, the system is given considerable discriminatory and cognitive power by its very structure. Artificial neural networks (ANN) in machine learning are an attempt to construct an analogous system from mathematics.

ANNs can be used for both classification and regression, and there are numerous ANN implementations, for example the FANN package or in the R language for statistical computing [65,66].
Figure 2.8: Schematic view of a myelinated vertebrate motor neuron. Its many dendrites branch out from multiple sites on its large soma (left) while its single axon extends to the right, branching out only far distally. Image from wikipedia.org, used with permission.
2.7. ARTIFICIAL NEURAL NETWORKS

Figure 2.9: A small ANN with three input nodes and two output nodes, and four nodes in the single hidden layer. This ANN is not fully connected; so only nonzero definite connection weights $w$ are shown. If this ANN were trained to classify apples, the nodes in the input layer ($a_{0,1}$, $a_{0,2}$ and $a_{0,3}$) could for example represent size, redness and weight of an apple, while the nodes in the output layer ($a_{2,1}$ and $a_{2,2}$) could represent two different predicted categories of apples, for instance delicious apples and disgusting apples.

In ANNs the neurons are called nodes, and these are typically arranged into layers as shown in Fig. 2.9. ANNs are most often purpose-built for solving specific tasks and are therefore generally significantly smaller than their biologic counterparts, but similarly to their biologic counterparts, they must be trained with real world examples before they can be used to reliably evaluate new samples. Also similarly to their biologic counterparts, training an ANN entails finding the connection strengths, or weights, that give the network the best possible performance.

A typical ANN has one input layer (which can be thought of as sensor neurons collecting stimuli from the surrounding world) and one output layer (which can be thought of as motor neurons controlling the muscles). These are most often separated by a single hidden layer (more are rarely needed) and this can be thought of as the brain of the ANN.

The number of nodes in the input layer is given by the number of different variables we have measured on the type of object we want the ANN to evaluate. For example, size, redness and weight of an apple would give three input nodes.

The number of nodes on the output layer is usually given by the number of predicted classes. For example, the two classes of delicious versus disgusting apples would give two output nodes. For regression, the number of output nodes will equal the dimensionality of the output; for example two nodes for a longitude-latitude coordinate, however for most everyday applications only one node is needed (monetary value, weight, velocity, concentration etc...).

The number of nodes in the hidden layer is most often empirically chosen, as is the degree of connectivity in the ANN. In a fully connected ANN, all of the nodes in each layer are connected to all of the nodes in the subsequent layer, however
this may not always be desirable. In a large ANN, this would cause the number of connections (and training parameters) to explode, yielding an ANN that is both time consuming to train and highly susceptible to overtraining\(^5\).

In the mathematics of it, \(a_{l,i}\) denotes the activation state of node \(i\) in layer \(l\), for \(i > 0\) and \(l > 0\). The input layer has \(l = 1\), and in an ANN with \(L\) layers, the output layer has \(l = L\). As already noted, \(a_{l,i}\) is usually given by the measurements or readings that we want the ANN to evaluate. In subsequent layers, \(a_{l,i}\) is given by an activation function \(f(S_{l,i})\) that depends on a weighted sum of the activation states of the nodes in the preceding layer [67]:

\[
S_{l,i} = \sum_{j=0}^{N_l-1} w_{l-1,j,i} a_{l-1,j} , \quad 1 < l \leq L
\]

where \(N_l\) is the number of nodes in layer \(l\), and \(w_{l,j,k}\) is the weight of the connection between node \(i\) in layer \(l\) and node \(k\) in the subsequent layer. The connectivity of the ANN can then be controlled by explicitly requiring a portion of the weights to be 0. This sum also introduces a computational bias \(a_{l,0} = 1\) with accompanying weights \(w_{l,0,k}\), which can thought of as the general “activatability” of the nodes.

As for the actual activation function, a transform bounded to the interval \(0 \leq f(x) \leq 1\) is most commonly used, such as the sigmoidal

\[
f(x) = \frac{1}{1 + e^{-x}}
\]

or the hyperbolic tangent

\[
f(x) = \tanh(x)
\]

Training the ANN is generally done by assigning a cost to the prediction errors, for example the mean square error [67]

\[
\text{MSE} = \frac{1}{X} \sum_x \sum_i (y_{i,x} - a_{L,i}(x))^2
\]

and then minimise it using gradient descent. Here, \(X\) indicates the total number of training examples, while \(y_{i,x}\) and \(a_{L,i}(x)\) indicate the desired and actual value of node \(i\) in the output layer given a training example \(x\). The minimisation can be performed by initialising the connection weights to small random numbers, and updating them progressively by feeding one training sample at a time into the ANN, calculating the error in the output, and propagating the weight updates back through the layers of the ANN. The weight update is then

\[
\Delta w_{l,j,i} = \eta a_{l,i} \delta_{l+1,j}
\]

\(^5\)Overtraining refers to when a model has been so meticulously tailored to excel at a very specific task that it fails abysmally even for minuscule deviations from the expected, which less specialised models would handle with no problem. This is also often called overfitting, and can be likened to a regular pair of shoes which will fit almost anyone, compared a tailor made orthopaedic ergonomic pair which will only truly fit one single person and be uncomfortable for everyone else.
2.8 Hidden Markov models

Strictly speaking, hidden Markov models (HMM) are statistical models that describe hidden states underlying sequences of observable symbols.

Simply put, they let us make qualified guesses about things that are difficult to measure, based on recognizing patterns in things that are easier to measure. For example, it would be extremely useful for a stock broker to know if the prices for some stocks currently are in a general state of rising or if they are in a general state of falling. This is of course very difficult (or impossible) to determine beforehand, however it should be possible to make some kind of guess from looking at the price history for the stocks, which is of course a sequence of observable symbols that is readily available, and therefore much easier to determine.

HMMs were originally developed for computer speech recognition, but have since found many applications in bioinformatics. They can for example be used with genomes to determine what parts of the sequence that are actually genes, or to find the transmembrane topology for proteins by determining which parts of the sequence that correspond to transmembrane segments and which parts that appear to belong on the inside or the outside of the membrane. Readers whose curiosities about HMMs are piqued by their rather brief treatment in this thesis are highly recommended a read of the excellent book on biological sequence analysis from 1999 by Durbin et al [69] for a much more rigorous examination and many further examples.

HMMs are products of machine learning, meaning that an individual HMM has to be built and trained for every specific use case, and having high quality training...
data is crucial for the quality of the resulting model. This makes construction of reliable HMMs a time consuming labour even for experts in the field. That said, good HMMs are among the best tools currently available to bioinformaticians.

HMMs consist of three parts: states, transition probabilities and emission probabilities. The states are the hidden properties of the sequences, which most often are what we are interested in determining. The transition probabilities connect the states and dictate how likely it is to move from one state to another. The emission probabilities specify how likely it is for each type of symbol to occur in each state.

HMMs can be thought of as flow diagrams for sequence generation; moving from state to state according to the transition probabilities, and adding (emitting) one symbol at a time to the sequence, according to the emission probabilities at the visited states. For this reason, HMMs are said to be generative models.

Moving through the states in the model creates a state path, and many of the problems that are solved using HMMs revolve around reconstructing the most probable state path for a specific observed sequence. But before we get to that I would like to give you an idea of what HMMs really are, using three concrete examples in escalating order of complexity.

2.8.1 Examples

2.8.1.1 The cheating croupier

The perhaps easiest way of understanding HMMs is through an example where a croupier sometimes rolls suspiciously many sixes in a row [69]. You may suspect that this pattern may be a consequence of an occasional switch between regular and loaded dice. If this were true, you could probably win big if you could just determine when the loaded die is used, and bet accordingly. Now if you write down the die rolls you have a sequence of easily observable data, and the hidden states of the sequence are the type of die being used at the time. We can now build an HMM to help us beat the system with confidence.

This HMM will have two states (loaded and regular), transition probabilities for moving between the states (switching dice or keeping them for another roll), and a list of emission probabilities for each state (the probability for each number to come up when using the specific die). Fig. 2.10 shows an illustration of such an HMM. In this example, we can see that the croupier has a 50 % chance of rolling...
2.8. HIDDEN MARKOV MODELS

a six using the loaded die. Also, the chances of switching states are much lower than the chances of rolling the same type of die again, which means that switching will occur only infrequently (possibly in order to minimise the risk of getting caught switching). Also, the chance to switch to the loaded die is only a fifth of the chance of switching away from it, which means that the regular die will be used on average five times as often as the loaded die (possibly in order not to raise suspicion by rolling too many sixes). So what we want to do now is to fit a recent series of rolls to the model, and from this we can determine the probabilities for being in a certain state for the next roll (that is, whether or not the croupier is currently cheating) and place our bets accordingly (see section 2.8.2.4 for how to do this).

The astute reader may wonder where all these numbers for transition and emission probabilities came from, and alas, I must admit that they at this point are mere guesswork. However, these estimations will do fine as a starting point for section 2.8.2.5, where proper parameter estimation of HMMs is covered.

2.8.1.2 Profile hidden Markov models

As may be evident from the name, the purpose of profile HMMs is similar to the PSI-BLAST PSSMs, which are often called profiles (cf. section 2.2.1). Profile HMMs are implemented for example in the widely used HMMER package, and are an example of when a biological problem is solved by finding the probability of a sequence for a given model. The math on how to do this is described in section 2.8.2.3.

A profile HMM describes a group of related proteins, such as a protein family, and particularly the conservation patterns and mutations that commonly occur within the family. Based on this information, the profile HMM can be used to reliably estimate whether or not an unknown sequence actually is a member of its corresponding protein family, and a library of HMMs such as Pfam [32] can therefore be highly useful for example to automatically classify genes in newly sequenced genomes.

Each profile HMM can be thought to describe the typical member of its corresponding protein family, and each position in this hypothetical typical sequence is modelled by a state triplet representing the different mutation types; a delete state (models deletions), an insert state (models amino acid insertions between this position and the next one), and a match state whose emission probabilities describe substitutions. The insert states are most often ascribed a uniform emission distribution (naturally, since it would be unfeasible to correctly determine the correct distributions for all potential insertions that may occur in protein). The delete states do not have emission distributions since they do not emit any symbols. There are three additional states; a begin state and an end state which represent the start and end of the protein sequence, and the initial insert state that represent poorly conserved residues that may occur in member proteins before the first residue in the hypothetical typical sequence. Like the delete states, the begin and end states do not emit symbols and are therefore called silent states.

Fig. 2.11 shows an illustration of a very small profile HMM. We can see that insertions are rare in this family, as are deletions past the first position. Some
Figure 2.11: A profile HMM of a small region of an MDR family. Circles represent delete states, diamonds represent insert states, and rectangles represent match states. The emission probabilities for each amino acid type are shown using horizontal bars in each match state. Transition probability is indicated by the thickness of the lines connecting the states, except for the insert state reentry probabilities which instead are shown in percent for each state. The numbers in the delete states simply enumerate the positions in the model. Normally, profile HMMs are hundreds of positions in length, just as the proteins they model, however this one has been truncated for brevity. This HMM was built using the HMMER package [70] and the figure was produced using the SAM package [71].

positions also seem to be very well conserved in this family; for example position 2 which is almost strictly conserved to lysine (K), with only a small allowance for arginine (R, also positively charged). It would also seem that a few members of this family have long trailing insertions, as evidenced by the low transition probability to the final insert state, and its 91% reentry transition probability.

2.8.1.3 A transmembrane topology predictor

Finally, an example where finding the most probable state path answers the biological question; a transmembrane topology predictor. Membrane spanning proteins are often threaded back and forth multiple times though the membrane, with specific parts exposed to each side. The apparent properties of the protein will depend strongly on which parts are exposed to which side, determining which stimuli it can react to, what possible antigens it presents, which signals it can detect and propagate, and which segments may come together in tertiary structure to form active sites and functional motifs.

Such a predictor can be created using an HMM where the states correspond to different locations in the topology, so that when a sequence of interest is threaded through the model, its most likely transmembrane topology will be given by the resulting most probable state path. TMHMM is one implementation [72] and its state network is shown in Fig. 2.12 as an example of a more complex model.

2.8.2 Algorithms

The HMM algorithms are described in elaborate and succinct detail in [69] and will be outlined only briefly here.
2.8. HIDDEN MARKOV MODELS

Figure 2.12: States and permissible transitions in the HMM used in the transmembrane topology predictor TMHMM. a) Overall structure of the HMM. Boxes indicate states or groups of states which are explained in detail in b and c. Boxes with identical labels have identical transition and emission probabilities. b) States and transitions for the boxes Globular, Loop and Cap. c) States and transitions for the Helix core boxes. Illustration adapted from [72], where the model is also described in detail.
CHAPTER 2. MATERIALS AND METHODS

Conceptually, most solutions to HMM problems are straightforward; evaluate all possible paths for a sequence through a model, take the product of all transition and emission probabilities along the way for each path, and choose the outcome that has the highest resulting probability. However, for longer sequences and more complex models, the straightforward solution quickly becomes intractable as the number of possible paths explodes. Luckily, HMMs\(^6\) have a very useful property that again makes this problem tractable, namely that the probability to be in any given state at position \(p\) only depends on which state the sequence was in at position \(p - 1\), meaning that we only have to remember where we were previously – not how we got there. This is very convenient because it allows us to solve many HMM problems through simple recursion.

2.8.2.1 Some definitions

Let’s define a few concepts to simplify the coming sections. Let’s say we have an HMM with transition probabilities \(t_{kl}\) for transitions from state \(k\) to state \(l\), and emission probabilities \(e_k(c)\) to emit symbol \(c\) in state \(k\). Furthermore, let’s say we have a sequence \(s\) of length \(L\), where \(s_i\) denotes the symbol at position \(i\) in the sequence. We can thread the sequence through the states in the model in many different ways, each giving rise to a different state path \(\pi\), where \(\pi_i\) is the hidden state of symbol \(s_i\), and we call the most probable state path \(\pi^\star\). \(t_{0k}\) denotes the transition probability from the begin state to state \(k\), and transitions from state \(k\) to the end state are denoted \(t_{k0}\).

For now, we will also assume that the HMM has a begin and an end state, and no silent states. These restrictions will be lifted in section 2.8.2.6.

2.8.2.2 Finding the most probable state path

The Viterbi algorithm finds the optimal state path \(\pi^\star\) through an HMM for a given sequence by recursing through the positions in the sequence. At each step, the most probable transitions \(x_i(l)\) are determined through maximising the Viterbi variables \(v_k(i)\), each of which holds the probability for the best path that ends in state \(k\) at position \(i\). The algorithm is as follows [69]:

Initialisation:
\[
v_k(0) = \begin{cases} 
1 & k = 0 \\
0 & \text{otherwise}
\end{cases}
\]

Recursion, \(i = 1...L\):
\[
v_i(l) = e_l(s_i) \max_k (v_k(i - 1) t_{kl})
\]
\[
x_i(l) = \arg \max_k (v_k(i - 1) t_{kl})
\]

Termination:
\[
P(s, \pi^\star) = \max_k (v_k(L) t_{k0})
\]
\[
\pi^\star_L = \arg \max_k (v_k(L) t_{k0})
\]

State path reconstruction, \(i = L...2\):
\[
\pi^\star_{i-1} = x_i(\pi^\star_i)
\]

\(^6\)and Markovian networks in general
This algorithm can for example be used with the TMHMM model to find the transmembrane topology of a protein, or with a profile HMM to determine for example which parts correspond to insertions or commonly conserved regions.

**2.8.2.3 Finding the probability of a sequence**

The forward algorithm finds the probability that a sequence matches a given model. It closely resembles the Viterbi algorithm, but the maximisations are replaced by sums. The forward variable \( f_k(i) \) holds the probability for the sequence up to position \( i \) along any path that ends in state \( k \). The algorithm is as follows [69]:

**Initialisation:**
\[
 f_k(0) = \begin{cases} 
 1 & k = 0 \\ 
 0 & \text{otherwise} 
\end{cases}
\]

**Recursion, \( i = 1 \ldots L \):**
\[
 f_l(i) = e_l(s_i) \sum_k f_k(i-1) t_{kl}
\]

**Termination:**
\[
P(s) = \sum_k f_k(L) t_{k0}
\]

This algorithm can for example be used with a profile HMM to determine whether or not a protein sequence belongs to the family modelled by the HMM, or in the croupier case, we can use it to assess whether the croupier is sticking to her usual switching scheme, or if we need to reevaluate our model.

**2.8.2.4 Finding the probability to be in a specific state at a given position**

To determine the probability that a sequence is in a given state at a specific position, we need to evaluate not only all possible paths leading up to the position, but also all the paths from the actual position to the end of the sequence. This is done in reverse using the backward algorithm, which is very similar to the forward algorithm, except in initialisation and directionality [69]:

**Initialisation:**
\[
b_k(L) = t_{k0}
\]

**Recursion, \( i = L \ldots 1 \):**
\[
b_k(i) = \sum_l f_{kl} e_l(s_{i+1}) b_l(i+1)
\]

**Termination:**
\[
P(s) = \sum_l t_{l0} e_l(s_1) b_k(1)
\]

Now, the probability for the sequence being in a state \( k \) at position \( i \) is simply
\[
P(\pi_i = k | s) = \frac{f_k(i) b_k(i)}{P(s)}
\]
and we can thus determine the most likely state for a specific position in the sequence by maximisation:
ENGINEERING DESIGN

CHAPTER 2. MATERIALS AND METHODS

\[ \hat{\pi}_i = \text{argmax}_k P(\pi_i = k | s) \]

Determining \( \hat{\pi}_i \) is called posterior decoding, and this method is called the forward-backward algorithm. Since this approach evaluates all possible paths instead of only the most probable one, it has the potential to deliver more reliable results for individual positions, even though \( \hat{\pi} \) may not be as probable as \( \pi^* \) (or even valid) throughout the model.

2.8.2.5 Building an HMM

The first step of HMM creation is to define the topology of the model (the states and the permissible transitions) and this is usually done manually. One should strive for a design that is complex enough to capture the complexity of the modelled system, but at the same time having as few free parameters as possible, as training a model with many free parameters will require more training data in order to become reliable. For example, in the croupier case, if we assume that the regular die is in fact regular, we can fix all its emission probabilities at 1/6, yielding a model with only half as many emission probabilities to train, and which will become more reliable with the given amount of available training data (and given that our assumption is true).

The remaining problem is then to determine the transition and emission probabilities. This is done using reliable training data on the modelled system; proteins with well known transmembrane topology for TMHMM, proteins from a specific family for a profile HMM, and our painstakingly recorded die rolls in case of the croupier.

In the TMHMM case, training is straightforward because the true state paths for the training sequences are known (e.g. the transmembrane topology has been determined experimentally). Similarly, for the profile HMM, we can determine the most probable state paths from an MSA of the sequences. In this situation the transition and emission probabilities are given by their maximum likelihood estimators [69]:

\[ t_{kl} = \frac{T_{kl}}{\sum_{l'} T_{k'l'}}, \quad e_k(b) = \frac{E_k(c)}{\sum_{c'} E_k(c')} \]

where \( T_{kl} \) is the observed number of transitions from state \( k \) to state \( l \), and \( E_k(c) \) is the number of observations of the symbol \( c \) in state \( k \). It is common to add a small number (pseudocounts) to \( T_{kl} \) and \( E_k(c) \) in order avoid overtraining, which would otherwise render the model unsuitable for sequences having yet unobserved transitions and emissions.

In the case of the cheating croupier however this approach is inapplicable, because there is no practical way of acquiring the true state paths for the observed sequences (apart from actually asking the croupier, which is out of the question). Lacking the correct state paths, finding the probabilities now becomes an optimisation problem that is typically solved using the Baum-Welch algorithm. Here,  

\footnote{which is not that unreasonable; the croupier will have to have something bona fide to show if the authorities suddenly decide to turn up.}
arbitrary initial probability estimates are iteratively improved until some stopping criterion defined on the quality of the model has been satisfied; usually that the probability of the training data given the current model parameters is sufficiently stable from one iteration to the next, which should indicate convergence to a (hopefully global) maximum.

In each iteration, the new probabilities are calculated using the maximum likelihood estimators for $t_{kl}$ and $e_k(c)$ as before. However $T_{kl}$ and $E_k(c)$ are now expectation values (with added pseudocounts, $p$) which are derived from the probabilities from the previous iteration by summing over every position $i$ in every sequence $j$ in the training data [69]:

$$T_{kl} = p + \sum_j \frac{1}{P(s^j)} \sum_i f^j_k(i) t_{kl} e_{j}(s^j_{i+1}) b^j_l(i + 1)$$

$$E_k(c) = p + \sum_j \frac{1}{P(s^j)} \sum_{i | s^j_i = c} f^j_k(i) b^j_k(i)$$

where $j$ superscripts are used in reference to sequence $j$ in the training data.

Calculating the probability of the training data given the model parameters is straightforward under the assumption that the sequences in the training data are independent, since it then becomes the product of the probabilities of the individual sequences. In logarithmic space\(^8\) this becomes the sum

$$l(S|t,e) = \sum_j \log P(s^j|t,e)$$

which is equal to the log-likelihood of the model [69]. The Baum-Welch algorithm then becomes [69]:

**Initialisation:**
Assign arbitrary probabilities to $t_{kl}$ and $e_k(c)$.

**Iteration:**
Calculate expectation values for $T_{kl}$ and $E_k(c)$ (with added pseudocounts).
Calculate new maximum likelihood values for $t_{kl}$ and $e_k(c)$.
Calculate new log likelihood for the model $l(S|t,e)$.

**Termination:**
Stop when the change in $l(S|t,e)$ is adequately small.

### 2.8.2.6 Generalisation

Adapting these algorithms to models without begin or end states is straightforward, because without an end state the $t_{k0}$ transition disappears and the recursion is instead terminated at the end of the sequence, and a begin state can for the sake

---

\(^8\) Actual numerical evaluation of HMMs is normally done in log space, because products of multiple probabilities will quickly vanish below the floating point precision for most computers. As an added benefit, sums are also much faster to compute.
of simplicity always be assumed, in which case the $t_{0k}$ probabilities can instead be seen as a probability distribution for sequence initiation in any of the states.

A little more care has to be taken when introducing silent states into the model. The best method is to design the model so that all possible transitions between silent states become acyclic, meaning that all silent states should be enumerable so that transitions always occur from lower to higher numbered states, just as in Fig. 2.11. Then, the adaptation for the forward algorithm becomes [69]:

1. For all non-silent states $l$ calculate $f_l(i + 1)$ from $f_k(i)$ as before.

2. Starting from the lowest numbered silent state $d$, set

   
   $f_d(i + 1) = \sum_k f_k(i + 1)t_{kd} + \sum_l f_l(i + 1)t_{ld}$

   for all non-silent states $k$ and silent states $l < d$.

The adaptations are very similar for the other algorithms.
Chapter 3

Current investigations

The focus of my thesis work has been the study of protein families; I have analysed known protein families, I have created predictive models for protein families, and developed algorithms for automated discovery and definition of new protein families. To this end I have used many bioinformatic techniques, and chapter 2 details my most frequently employed ones. My principal techniques however have been sequence similarity search, MSA and HMM.

This chapter describes my work in bioinformatic research, starting with a general overview and summary of aims in the included projects, and with summaries of the work done in the individual papers in the following sections. The major conclusions are summarised at the end of this chapter.

One could say that protein family characterisation is a four step process. The first step is to find as many member sequences as possible, from multiple databases, using both annotations and sequence similarity as search criteria. Since annotation error and partial sequences abound in the databases, it is nearly always necessary to include a sanity check of the amassed data as a second step, in order to apprehend erroneously included nonmember sequences as early as possible, lest they insinuate themselves into the calculations and contaminate the conclusions of an eventual paper. With spurious sequences removed, the third step is to take the sequences and aggregate, slice, cluster and partition them in as many ways and levels as possible, to align them in parts and as a whole, to run predictions, to integrate data from different sources and to compare the results; within groups, between different groups and between the groups and the whole, and look for similarities and discrepancies. The fourth step is to gather the all findings and publish a paper.

This four step scheme is deceptively simple. In reality, it consists of a myriad of small and interwoven steps, with interchanging data and parameters, and using a variety of computer programs, few of which were ever designed to be on speaking terms. Furthermore, as new sequence data continue to pour in, and the scales of the bioinformatic projects continue to outgrow the computers, it may become necessary to run the calculations on supercomputers, or distribute them over
computer clusters\textsuperscript{1}, adding all the particularities of supercomputing to the already simmering pot of bioinformatic complexities.

Throughout the process of bioinformatic research, it is therefore also necessary to be adept at computer programming, so you can complement the existing tools with new software; for answering specific questions, or showing information in an accessible way, or gluing stubbornly non-communicating components together.

In the papers included in this thesis I have analysed the protein family of inorganic pyrophosphatases (H\textsuperscript{+}-PPases, paper I) as well as the two protein superfamilies BRICHOS (paper II) and MDR (medium-chain dehydrogenases/reductases, paper III–IV). I have also developed an algorithm for automated generation of stable and reliable HMMs for protein families, and I have applied this algorithm to construct stable and reliable HMMs for both known and novel protein families within the MDR superfamily (paper IV). Furthermore, I have developed a new algorithm for quantifying conservation (paper II and V), and I have developed a software platform for working with MSAs (paper V).

Papers I–IV are open access (OA) articles, meaning that they are available free of charge for everyone to read and download online, and paper V will be submitted to an OA journal.

\textsuperscript{1}or clusters of computer clusters, cf. chapter 4.
3.1 Aims

This section lists some of the aims in the projects included in this thesis;

To develop an algorithm for automated protein family discovery, definition and classification (paper IV).

To develop algorithms and applications to aid protein family characterisation (papers II and V).

To investigate protein families and superfamilies using bioinformatic methods;

- The family of inorganic pyrophosphatases (H⁺-PPases, paper I);
  - To find putative functional sites and regions.
  - To determine the uniqueness of their characteristic pyrophosphate binding primitive nonapeptide motifs.

- The BRICHOS superfamily (paper II);
  - To find putative functional sites and regions.
  - To discover and characterise previously unknown member families.

- The superfamily of medium-chain dehydrogenases/reductases (MDR, paper III–IV);
  - To determine the size and complexity of the superfamily.
  - To characterise the member families.
  - To develop predictive models for member families, for use in functional classification of uncharacterised sequences.
  - To structure knowledge on MDRs by discovering and characterising previously unknown member families.

To make our findings accessible outside the field of bioinformatics.
CHAPTER 3. CURRENT INVESTIGATIONS

Figure 3.1: Conservation profile and predicted transmembrane topology for the H\(^+\)-PPase family. A sliding mean plot of Clustal\(X\) \[43\] \(q\)scores shows that the regions of high conservation consistently coincide with cytosolic or transmembrane regions, showing that there is little to no conservation on the non-cytosolic side of the membrane. Five regions have distinct peaks above 55 (dotted line), indicating strong conservation. Sequence indexes from \textit{Streptomyces coelicolor} \(H^+\)-PPase.

3.2 Paper I – Inorganic pyrophosphatases

This paper focuses on characterisation of the protein family of inorganic pyrophosphatases (H\(^+\)-PPases).

The H\(^+\)-PPases \[73,74\] are tightly membrane bound proton pumps \[75\] that are dimeric with \(~700\)-residue subunits, and which produce energy by photosynthesis of inorganic pyrophosphate \[74\]. They are the only known alternative to ATP formation in biological electron transport phosphorylation, and may well predate ATP-driven cellular energy production. There are two distinct subtypes, where type 1 is \(K^+\)-dependent and type 2 is \(K^+\)-independent. H\(^+\)-PPases are present in bacteria, plants and primitive eukaryotes (parasites and protists), and UniProtKB \[76\] currently lists close to 800 known members. Recently, a H\(^+\)-PPase was also reported in starlet sea anemone \[77\], however as this genome is still ongoing and as this (partial) sequence shares the highest similarity with bacterial enzymes, it cannot be excluded that this finding may be from a contaminating sequence.

In our study, we built a HMMER \[70\] profile HMM for the well-conserved cytosolic loop between transmembrane regions 5 and 6 (loop 5–6, peak 1 in Fig. 3.1) and used it to search for homologues in UniProtKB and GenomeLKPG (locally compiled database of available completed genomes, unpublished collaboration...
Figure 3.2: Dendrogram of the H\(^{+}\)-PPases, clearly separating type 1 and type 2 enzymes. Plant sequences are shown in green, primitive eukaryotes in red (alveolates and euglenozoans), bacterial in blue and archaeal in yellow. The two provisional *Xenopus* sequences (purple, clustering tightly with the bacterial sequences) were later found to be from contaminants in the sequencing project, and were removed in a subsequent release of the genome. Red dots indicate branch points with bootstrap values equal or higher than the division between type 1 and type 2 enzymes (888/1000).
with Anders Bresell), and at the time we found a total of 166 sequences. Fig. 3.2 shows a redundancy reduced dendrogram of these sequences, showing also the species distribution and the clear division between the subtypes.

We also searched a metagenomic database containing approximately one million environmental sequences from the Sargasso sea [20] yielding an additional 164 sequences, which somewhat surprisingly showed notably higher sequence similarity (cf. Fig. 3.3B).

Our HMM was shown to be highly specific, and therefore highly reliable for detection of new H\(^+\)-PPases. The region also seemed unique for the H\(^+\)-PPase family, which might be seen as evidence of early separation from other families, making H\(^+\)-PPases a low-positioned branch in the genealogical tree of protein families. We created additional HMMs for the cytosolic loops containing peak 2–3 and 4–5, which also were found to only match H\(^+\)-PPases, further emphasising the uniqueness of the family.

We found considerable multiplicity of this enzyme in some species. Several species present multiple forms which in many cases show little sequence similarity (~38 % identities), and belong to separate types. For cress (Arabidopsis thaliana), there are five type 1 and six type 2 enzymes, whereas for rice (Oryza sativa) there exist 18 enzymes of type 1 and two enzymes of type 2, and we saw similar multiplicity for the blood parasites Tetrahymena, Plasmodium falciparum and Plasmodium yoelii. Some archaea have multiple H\(^+\)-PPases, however most bacteria were found to have only one form.

Loop 5–6 contains three very well conserved sequence motifs that are believed to bind pyrophosphate and form part of the active site [80] of the enzyme: GGG, DVGADLVGD and DNVGDNVGD (cf. Fig. 3.3), where the nonapeptide motifs consist of three “primitive” tetrapeptide motifs (DVGA, DLVG and DNVG) [81–85]. The location of these motifs is indicated with red boxes in loop 5–6 in Fig. 3.4. We made diagnostic patterns for these motifs (disregarding residue types with only one observation among all known H\(^+\)-PPases), and remarkably, out of the 192 and 32 possible combinations of different nonapeptides, only 11 and 7 were found to have matches in UniProtKB, and out of all matching sequences, all but 1 and 2 are annotated as H\(^+\)-PPases. Thus, the nonapeptide patterns are, with these few exceptions, specific for the H\(^+\)-PPases. Also, expanding the patterns to allow any residue type in the variable positions dramatically increases the number of non-H\(^+\)-PPase matches.

The HMM for loop 5–6 found distant similarity to loop 15–16 in plant H\(^+\)-PPases. Here, the second nonapeptide motif is well conserved, with all three aspartic acid residues unchanged. For the first nonapeptide motif, charges are found at positions 1, 5 and 9 in the order Asp, Asp, Lys in loop 5–6, whereas the order is Asp, Lys, Asp in loop 15–16. Notably, the positional spacing between the two nonapeptides in the two loops is identical (26 residues). Furthermore, the GGG motif preceding the nonapeptides in loop 5–6 could correspond to a GGA or GGS motif preceding the nonapeptide in loop 15–16. These observations, taken together, might reflect an ancient gene duplication event.

In order to characterise the differences between sequences of types 1 and 2, we compared the positions strictly conserved within each type, but differing between
Figure 3.3: Consensus sequences in loop 5–6. The conservation in sequences found in (A) databases and genomes, and (B) Sargasso sea metagenomic sequences, is shown using sequence logos [78, 79] coloured according to chemical properties; green represents polar residues, blue basic, red acidic and black hydrophobic residues. For legibility, the residue letters are also shown under the axis, ordered by decreasing prevalence.
CHAPTER 3. CURRENT INVESTIGATIONS

Figure 3.4: Differentially conserved positions in type 1 and type 2 H$^+$-PPases. The green regions represent the five peaks in Fig. 3.1. The red boxes indicate the placement of the characteristic nonapeptides. The bolded sequence backbone indicates positions where the two most common residue types in the position are present in more than 95% of the family. Positions that are differentially conserved are labelled AB/CD, to indicate that more than 90% of the type 1 enzymes have residue A or B at this position, while 90% of the type 2 enzymes have C or D, and that A and C are the most common ones. Boldface residue letters indicate that more than 85% of the sequences in the corresponding subtype have this residue. Conserved substitutions within a ClustalW “strongly conserved group of residues” are shown using dash symbols on the backbone, and ring symbols are similarly used for ClustalW “weakly conserved groups of residues”. Bullet symbols on the backbone denote conserved substitutions between two residues that do not occur together in any of the ClustalW groups, which implies functional impact. Sequence indexes from *Streptococcus coelicolor* H$^+$-PPase.
the types, as indicated in Fig. 3.4. We found seven such differences, four in the transmembrane segments and three on the cytosolic side, while none was found on the noncytosolic side. Two of the conserved differences concerned residues with different physicochemical properties, implying possible functional impact (Fig. 3.4, residues indicated by bullet symbols), which has already been established for one of these (position 507 in Fig. 3.4), where an alanine to lysine mutation has been shown to confer K⁺-independence to a normally K⁺-dependent type 1 enzyme [86]. At position 253 in the loop between transmembrane segments 5 and 6, close to one of the conserved nonapeptides, the type 1 enzymes have predominantly hydrophobic isoleucine or valine, while type 2 enzymes have polar cysteine or serine.
Hydrophobic Linker BRICHOS C-terminal

Figure 3.5: Domain organisation of the BRICHOS containing human ITM2A protein.

3.3 Paper II – BRICHOS and the CScore

This paper focuses on characterisation of the BRICHOS superfamily, and also presents an algorithm for quantification of conservation.

The BRICHOS superfamily is present mainly in vertebrates but also in worm, sea anemone and various insects. There are 8 known BRICHOS families, showing a wide range of functions and disease associations [87]; the cancer associated GKN1, GKN2 (gastrokine-1 and -2) and LECT1 (chondromodulin-1) [88–90], the three dementia associated ITM2 (integral membrane protein 2 A, B and C) families [91, 92], the respiratory disease associated proSP-C (pulmonary surfactant protein C precursor) [93], and TNMD (tenomodulin). The sequences are typically ~200 residues in length, but are generally posttranslationally processed and cleaved down to even shorter functional forms [94–96]. BRICHOS proteins have four regions; hydrophobic, linker, BRICHOS and C-terminal (shown in Fig. 3.5). The hydrophobic region is most often a transmembrane segment but may also be a signal peptide (or both) in some member families. The BRICHOS domain itself is thought to have a chaperone function [97], as mutations in this domain can cause amyloid formation from other parts of the peptide [98].

There is little sequence identity between the families, and there are no structures even for remote homologues in the PDB database [36]. There are currently only 208 known members in the UniProtKB database and, possibly due to its specialised species distribution, its growth has not paralleled the dramatic increase recently seen in most other protein families.

In order to find previously unknown BRICHOS sequences, we built a HMMER [70] profile HMM for the family and used it to search UniProtKB and GenomelKPG (locally compiled database of available completed genomes, unpublished collaboration with Anders Bresell), yielding 309 proteins, after removal of sequence fragments.

Fig. 3.6 shows a redundancy reduced dendrogram of the sequences, which clearly separates the superfamily into 12 groups. Eight of these represent known families; one is a divergent group of primitive sea organisms, and three represent novel BRICHOS families (group A, B and C).

Group A clusters closely with the ITM2 families, and its position in the dendrogram (and its insect and Caenorhabditis members) indicates that it may represent an evolutionary predecessor to the ITM2 families.

The divergent group has little sequence identity internally or to other groups, and its position in the tree and its Echinoderm and Amphioxus sequences are compatible with an ancestral nature of these sequences.
Figure 3.6: Dendrogram of the BRICHOS superfamily. 12 groups are clearly distinguished; proSP-C (pulmonary surfactant protein C precursor), group C, GKN2 and GKN1 (gastrokine-2 and -1), group B, LECT1 (chondromodulin-1), TNMD (tenomodulin), the divergent group, group A, and ITM2A, ITM2C and ITM2B (integral membrane protein 2 A, C and B).
Group B has so far only been found in mouse, rat, cow and dolphin. It is closely colocalised with the GKN families both in the genome and in the dendrogram, which suggests that group B may represent a third type of gastrokine, possibly resulting from a recent gene duplication.

Group C is a distinct group of tetrapod and fish sequences. It has little sequence similarity to the other BRICHOS families and may thus represent a truly disjoint and novel BRICHOS family.

We find that conservation differs considerably among the regions and families. The spread in average pairwise percent identities for the hydrophobic ranges from 26% in group A to 96% in proSP-C, indicating drastically different functional constraints. Conversely for the BRICHOS region, all families have 51–83% sequence identities, indicating similar functions among the families.

Although the degree of conservation across the superfamily is low, we find remarkable coherence in secondary structure predictions [99–101], and not only in the BRICHOS domain (cf. Fig. 3.7). Also, the few natively disordered regions [4] are with few exceptions predicted to be N-terminal of the hydrophobic region, indicating that the proteins may have otherwise well defined tertiary structures.

Remarkably, although the degree of conservation is high in individual families, we find only three residues completely conserved in the superfamily; one aspartic acid and two cysteines, all in the BRICHOS region (cf. Fig. 3.7). The cysteines form an internal disulphide bridge in proSP-C [102], which could be true for all families. Furthermore, we find two cysteines in the C-terminal region that are strictly conserved in all families except group A, where they are absent from all sequences. These cysteines might also form a disulphide bridge.

We also found the C-terminal region of group C to be extremely well conserved, with nearly identical sequences in all species ranging from fish to man (as seen in Fig. 3.8). Also, this region has no detectable homologues in UniProtKB, making it unique to this family and especially interesting for further studies.

Furthermore, we find that the BRICHOS domain of ITM2 has a conserved net negative charge, correlated with a conserved net positive charge in the C-terminal region. The most extreme net charges being $\pm 5$ and $\pm 6$ in the different regions of ITM2A. This trait is shared but less pronounced in group A. Furthermore, group A lacks the remarkably high number of conserved hydrophobic residues in the ITM2 BRICHOS domains, but is more similar to the other families in this respect. These characteristics are in accordance with group A being ancestral to ITM2.

In this paper we also present the CScore, a conservation measure which, similarly to the ClustalX [43] qscore, computes conservation at individual MSA positions as a diminishing function of the average distance between residue vectors from a substitution scoring matrix (such as BLOSUM62). The differences are that in the CScore algorithm unknown characters (such as gaps) do not contribute to the placement of the residue vector centroid, while in the qscore algorithm the centroid is proportionally displaced toward the origin, and that the distance-to-score transform is linear rather than exponential, and with well-defined bounds. The details of the qscore algorithm are only available in source-code form, however a noticeable difference between the two is that the CScore penalises partially gapped positions less severely than does the qscore.
Figure 3.7: Conservation and predictions for secondary structure and native disorder in each region, shown using only one human representative from each family (Caenorhabditis for group A). The upper half of each figure shows positions that are highly conserved within each family in blue (strict conservation in dark blue). The lower half shows predicted secondary structure using red H for helix, green E for strand and black C for coil, with prediction strengths indicated on a scale from white (unreliable) to blue (reliable). Red rectangles indicate regions with probable native disorder.
Figure 3.8: Multiple sequence alignment of the C-terminal region of group C. Positions with at most one divergent residue are indicated with asterisks.
3.4 Paper III – The MDR superfamily

This paper focuses on characterisation of the protein superfamily of medium-chain dehydrogenases/reductases (MDR). In the preparatory work for this originally envisioned literature review, we discovered that the superfamily had grown considerably since a recent review [103], both in size and complexity. The project was therefore soon expanded to also include new bioinformatic experiments, in order to be able to accurately depict these new characteristics. This summary will mainly focus on the findings and conclusions from those experiments.

The MDR superfamily currently has over 16000 members, it is present in all kingdoms of life, the pairwise sequence identity is typically around 25 %, and the members are partaking in a multitude of biological processes. MDRs can be mono-, di- or tetrameric with subunits that vary in length and can bind 0, 1 or 2 zinc ions (Zn\(^{2+}\)) as catalytic and structural cofactors (cf. Fig. 3.9 and 3.10). The subunits are typically around 350 residues in length and divided into two domains, where the coenzyme binding domain is folded from much of the C-terminal half of the chain into a typical Rossmann fold [104] of a six stranded parallel \(\beta\)-sheet sandwiched between \(\alpha\)-helices on each side. The substrate binding domain is co-folded from most of the N-terminal half and a C-terminal segment, into a core of antiparallel \(\beta\)-strands with surface positioned \(\alpha\)-helices, showing distant homology with the GroES structure [105]. The domains are separated by a cleft containing a deep pocket which accommodates the active site and a NAD or NADP cofactor (nicotinamide adenine dinucleotide or the phosphorylated form). The structural Zn\(^{2+}\) (Zn2) stabilises a loop in the substrate binding domain [106] and enables the enzyme to function as a dimer instead of a tetramer.

Classical liver alcohol dehydrogenase (ADH) is often chosen as a representative for the MDRs, and so also here. The ADHs are dimeric zinc metalloproteins that occur as 5 different classes in the human, resulting from gene duplications during vertebrate evolution, the first one traced to ~500 million years ago (MYA) from an ancestral formaldehyde dehydrogenase line [108]. Like many duplications at that time, it correlates with enzymogenesis of new activities, contributing to conditions for emergence of vertebrate land life from osseous fish. Speed of changes correlate with functions, as do differential evolutionary patterns in separate segments. Overall, variability is large, but like for many dehydrogenases, subdivided into constant and variable forms, corresponding to housekeeping and emerging enzyme activities, respectively (cf. Table 3.1, and Fig. 3.10 for the

<table>
<thead>
<tr>
<th>Property</th>
<th>ADH III</th>
<th>ADH non-III</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary structure</td>
<td>Constant</td>
<td>Variable (~3–5 x III)</td>
</tr>
<tr>
<td>Segment variability</td>
<td>Non-functional (at 2 sites)</td>
<td>Functional (at 3 sites)</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>Often single form(s)</td>
<td>Often multiple forms</td>
</tr>
<tr>
<td>Main substrate specificity</td>
<td>Strict</td>
<td>Wide</td>
</tr>
</tbody>
</table>

Table 3.1: Characteristic differences in properties between “constant” and “variable” oligomeric dehydrogenases.
CHAPTER 3. CURRENT INVESTIGATIONS

Figure 3.9: A representative MDR structure; a human ADH class III monomer. The substrate binding domain (left) and the catalytic domain (right) are divided by a cleft containing a deep pocket which accommodates the active site and a NAD or NADP cofactor (nicotinamide adenine dinucleotide or the phosphorylated form; NAD, in this case, is shown using ball-and-stick representation). The coenzyme binding domain has a typical Rossmann fold (red) while the substrate binding domain shows distant homology with the GroES structure (green). This structure has two Zn$^{2+}$ cofactors (blue); both the catalytic (Zn1, upper) and the structural (Zn2, lower). Image rendered using Molsoft ICM Browser.

Figure 3.10: Notable sequence features in MDR-ADH. The continuous line shows the location of the cofactor binding domain (red) and the catalytic domain (black). The cofactor ligands are well-spread over most of the catalytic domain in case of the “catalytic” zinc (Zn1), but are closely spaced in case of the “structural” zinc (Zn2). Blue regions indicate the most variable segments in the “variable” classes of the protein (marked I) and in the “constant” classes (marked III) [107].
locations of the variable segments in the sequence) [107]. Combined, they have specific substrates in metabolic pathways, some with wide substrate specificity, and several with little known functions.

Beside ADH, the MDR member families include quinone reductases, leukotriene B4 dehydrogenases, and many more forms. There are HMMs available for detecting MDR superfamily membership, but there are none for the individual families.

We initiated this study by searching UniProtKB [76] using the two MDR specific Pfam [32] HMMs, yielding 10888 entries, which marked an order of magnitude increase in only five years [103]. In order to get an overview of the relationships within the superfamily we clustered the sequences at different identity levels, and noticed that approximately 20 % of the sequences are eliminated already at the 99 % identity level, representing closely related forms or duplicates. At the 30 % identity level, we found 476 members, corresponding to about 5 % of the total number, and in most cases only one remaining representative from each known MDR family. We therefore gave an estimate of the total number of MDR families at around 500. Only 8 of these clusters had 200 or more members, representing the most widespread MDR families, and these comprise 3165 sequences or roughly a third of all known MDR forms. However, as the majority of the sequence clusters (334 families) had 10 or fewer members, we found that the spread of the MDR superfamily is similar to the complexity of its sister superfamily of short-chain dehydrogenases/reductases (SDR) [109], which was a novel characteristic at the time, and which had been undetectable before the arrival of data from many large scale genome sequencing projects.

In a dendrogram of the superfamily we were able to discern not only the 8 large MDR families, but also 9 smaller families of special interest, being for instance homogeneous clusters with human representatives. We made HMMER [70] profile HMMs for these families in order to find previously undetected family members, and in order to improve their reliability, we rebuilt the models iteratively through sequential database searches and progressive manual refinement of the seed, until the models were stable.

We then found the largest MDR families to be ADH and CAD (as shown in Table 3.2). However, it should be noted that the functional importance of each MDR family is not correlated with the size of the family. If anything, it might even be the opposite, since the most strict substrate specificity is often associated with house-keeping enzymes of low multiplicity (cf. Table 3.1).

In our investigations, we saw that knowledge of multiplicities within the MDR superfamily had been in steady increase (Fig. 3.11). In 1983, ADH and tetrameric dehydrogenases (TADH) were the only MDR families in the databases, while our other families of interest emerged in various later years. We also noted that all these families, and the MDR superfamily alike, had since grown approximately exponentially.

Disregarding species variations we still found considerable multiplicity, with at least 25 representatives in the human. Looking at the dendrogram of human...
Figure 3.11: MDR family sizes versus time of appearance in UniProtKB. The curve ALL shows the total number of MDR members.
Figure 3.12: Dendrogram of human and mouse MDR forms. Blue lines represent human MDRs and red lines mouse MDRs. MDR families are shown on the right. Labels show the UniProtKB name excluding species designation. Tree drawn using NJplot [110].
Table 3.2: MDR families by size as detected in UniProtKB and its subsections Swissprot (high quality) and TrEMBL (large size), using the corresponding HMMs with E-values 1e–100 or better.

<table>
<thead>
<tr>
<th>Family</th>
<th>Members</th>
<th>Swissprot</th>
<th>TrEMBL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADH</td>
<td>931</td>
<td>99</td>
<td>832</td>
<td>Alcohol DH</td>
</tr>
<tr>
<td>CAD</td>
<td>520</td>
<td>35</td>
<td>485</td>
<td>Cinnamyl ADH</td>
</tr>
<tr>
<td>LTD</td>
<td>427</td>
<td>15</td>
<td>413</td>
<td>Leukotriene DHs</td>
</tr>
<tr>
<td>TADH</td>
<td>330</td>
<td>40</td>
<td>290</td>
<td>Tetrameric ADHs</td>
</tr>
<tr>
<td>YHDH</td>
<td>295</td>
<td>1</td>
<td>294</td>
<td>E. coli yhdH homologues</td>
</tr>
<tr>
<td>BPDH</td>
<td>229</td>
<td>3</td>
<td>226</td>
<td>Bacterial and plant DHs</td>
</tr>
<tr>
<td>PDH</td>
<td>218</td>
<td>18</td>
<td>200</td>
<td>Polyol DH</td>
</tr>
<tr>
<td>TDH</td>
<td>215</td>
<td>49</td>
<td>166</td>
<td>Threonine DHs</td>
</tr>
<tr>
<td>BurkDH</td>
<td>67</td>
<td>0</td>
<td>67</td>
<td>DHs from <em>Burkholderia</em> bacteria</td>
</tr>
<tr>
<td>MCAS</td>
<td>58</td>
<td>1</td>
<td>57</td>
<td>Mycocerosic acid synthesis protein</td>
</tr>
<tr>
<td>MECR</td>
<td>49</td>
<td>13</td>
<td>36</td>
<td>Mitochondrial trans-2-enoyl-CoA-reductase</td>
</tr>
<tr>
<td>VAT1</td>
<td>39</td>
<td>6</td>
<td>33</td>
<td>Vesicle amine transport protein 1</td>
</tr>
<tr>
<td>QOR</td>
<td>28</td>
<td>8</td>
<td>20</td>
<td>Quinone oxidoreductase</td>
</tr>
<tr>
<td>ACR</td>
<td>25</td>
<td>4</td>
<td>21</td>
<td>Acyl-CoA reductase</td>
</tr>
<tr>
<td>DOIAD</td>
<td>16</td>
<td>7</td>
<td>6</td>
<td>Deoxy-scyllo-inosamine dehydrogenases</td>
</tr>
<tr>
<td>QORL</td>
<td>14</td>
<td>3</td>
<td>11</td>
<td>Quinone oxidoreductase-like proteins</td>
</tr>
<tr>
<td>RT4I</td>
<td>10</td>
<td>3</td>
<td>7</td>
<td>Nogo-interacting mitochondrial proteins</td>
</tr>
<tr>
<td>Total</td>
<td>3471</td>
<td>307</td>
<td>3164</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.13: Human ADH gene arrangement on chromosome 4. The arrows indicate the chromosomal location of the ADH genes, with Swissprot names inside the arrows, class designations above and chromosomal coordinates below.

Investigating the localisations of MDR genes in the human genome [27], we found that the only MDR genes that appear to be clustered are the ADH genes, which all are located within a region of 0.5 million bases (Mb), on the reverse strand 100 Mb into chromosome 4 (cf. Fig. 3.13). All the other MDR forms were found to be spread out on different chromosomes or separated by several tens of million base pairs.

We also investigated the presence of MDRs in complete genomes (locally compiled database, unpublished collaboration with Anders Bresell), and found that on average 0.2 % of all protein-coding genes in an organism are MDR enzymes.
However, there is a large span, ranging from close to 0 to 0.85 %, being highest for different *Mycobacterium* species, *Rhodococcus*, and *Rubrobacter*. Among those with no or only few MDRs, we find genomes with low numbers of open reading frames, e.g. *Mycoplasma* and *Clamydia*, and several of these do not have complete metabolism but act as parasites. For comparison, the human genome has 0.78 % MDR forms. The genomes with the largest number of MDR forms include plants and bacteria. For the plants this may be explained by the tetraploidy of the genomes which contributes with several closely related forms. Disregarding identical and closely related sequences, the numbers drop considerably already at the 99 % or 90 % level for most of the bacterial genomes, and at the 40 % level, only about 1/4 of the plant sequences remain.

Furthermore, we found that archaeal organisms only have a few MDR representatives. The most frequent MDR form in bacteria is ADH, but still only 36 % of the genomes have such a representative. In eukaryotes, most of the MDR families are frequently represented, ADH being the most common and present in 79 % of the eukaryotic genomes. Somewhat surprisingly, some MDR families thought to be ubiquitous in eukaryotes were missing in several species. Notably, eukaryotes missing ADH in the genome databases are some invertebrates, single-cell plants, parasites living in higher cells, or little studied forms. The rule of widespread ADH members is therefore still applicable, but may perhaps not apply to all lower eukaryotes, or MDR-ADH may be replaced with SDR-ADH, which is compatible with the mixed presence of these two families in metabolic pathways [111].
3.5 Paper IV – Automated HMM refinement

This paper describes RefineHMM, an algorithm for automated discovery and definition of new protein families. It also details its application to subdivide the large and complex protein superfamily of MDR (investigated in paper III), and the characterisation of the defined families. For brevity, this summary will mainly describe the RefineHMM algorithm and its application for MDR subdivision, while our findings from characterising the MDR families will be treated mostly in general terms.

In paper III, we built stable profile HMMs [70] for 17 MDR protein families, using a manual and empiric approach. From our experiences we were able to devise an improved and automated HMM refinement algorithm that uses relationships found in data to produce stable and reliable HMMs. We then used this new algorithm to produce HMMs for 86 MDR families. We also characterised these 86 families, including species distribution, conservation, zinc content and NAD(P) cofactor preference, as well as their interrelations and other properties.

The RefineHMM algorithm is similar to the manual HMM refinement method developed for paper III, in that it uses iterative database searches to refine the models; in each step expanding the seed set with those new sequences that match the HMM at least as well as the seed sequences, and terminating when no new candidates are found. However, it is supplemented by a leave-one-out inclusion control step where the resulting model is iteratively rebuilt, each time leaving out a single seed sequence which is tagged as spurious if it no longer matches the HMM better than random sequences from the database. If all sequences pass the inclusion control step then the HMMs is considered stable; otherwise the spurious sequences are removed and algorithm restarted. A great strength of our novel algorithm is that it does not rely on user input beyond initial HMM creation, but instead purely uses relations found in data to produce the resulting models.

In RefineHMM, the individual refinement processes are independent, so the algorithm is trivially parallelisable. Also, the output files and the variant input files are small, while the larger input files are all constant and can therefore be cached on the compute resource. These traits make RefineHMM well suited for burst computing on shared resources as well as distributed resources. Indeed, all refinements and leave-one-out comparisons to UniProtKB were computed on Swegrid [112] using Biogrid runtime environments [113].

The RefineHMM algorithm uses version 2.3 of HMMER, but is written so that a transition to HMMER 3.0 will be straightforward once a stable release with glocal alignment capabilities is available [114]. Preliminary tests have shown that transitioning will likely increase the speed of the RefineHMM algorithm by at least an order of magnitude, and quite possibly even more.

Applying RefineHMM to the MDR superfamily we started from a set of 15136 sequences that we found in UniProtKB [76], similarly as in paper III. Clustering sequences with identity higher than 40 % left 1032 clusters, where known families such as ADH, PDH and LTD were uniquely represented. 146 of these clusters consisted of more than 20 sequences, and were used as seed sets for the initial...
3.5. PAPER IV – AUTOMATED HMM REFINEMENT

Table 3.3: MDR families by name used in paper III. The CAD, TADH and MECR families are now represented by two or three HMMs, and the MCAS and ACR families are now incorporated into a single, much larger HMM. The DOIAD, QORL and RT4I families are not included, because the amount of data available on these families was insufficient to satisfy the minimum inclusion size employed here.

<table>
<thead>
<tr>
<th>Family</th>
<th>Number</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADH</td>
<td>MDR001</td>
<td>931 → 2217</td>
</tr>
<tr>
<td>CAD</td>
<td>MDR010, MDR021</td>
<td>520 → 661, 122</td>
</tr>
<tr>
<td>LTD</td>
<td>MDR002</td>
<td>427 → 774</td>
</tr>
<tr>
<td>TADH</td>
<td>MDR016, MDR020, MDR029</td>
<td>330 → 313, 128, 29</td>
</tr>
<tr>
<td>YHDH</td>
<td>MDR012</td>
<td>295 → 481</td>
</tr>
<tr>
<td>BPDH</td>
<td>MDR011</td>
<td>229 → 575</td>
</tr>
<tr>
<td>PDH</td>
<td>MDR005</td>
<td>218 → 328</td>
</tr>
<tr>
<td>TDH</td>
<td>MDR014</td>
<td>215 → 351</td>
</tr>
<tr>
<td>BurkDH</td>
<td>MDR041</td>
<td>67 → 114</td>
</tr>
<tr>
<td>MCAS, ACR</td>
<td>MDR003</td>
<td>58, 25 → 706</td>
</tr>
<tr>
<td>MECR</td>
<td>MDR007, MDR037</td>
<td>49 → 51, 20</td>
</tr>
<tr>
<td>VAT1</td>
<td>MDR008</td>
<td>39 → 50</td>
</tr>
<tr>
<td>QOR</td>
<td>MDR009</td>
<td>28 → 22</td>
</tr>
<tr>
<td>DOIAD</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>QORL</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>RT4I</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

protein family HMMs, making a grand total of 10193 sequences. These initial HMMs were subsequently refined using the RefineHMM algorithm.

In some cases the refinement of two separate initial HMMs converged on the same protein family, producing partially or fully overlapping HMMs. These overlaps were in almost all cases trivially resolvable, either through simple mergers of seed sets, or through minute increases of the inclusion thresholds for either or both models to exclude the overlapping sequences.

It is also noteworthy that some proteins, mainly long chained polyketide synthases and type I fatty acid synthases, contained several MDR domain pairs that were sufficiently divergent to fall into separate clusters under our chosen clustering rule. No special regard was given to these clusters during neither HMM construction, refinement or inclusion control, but still all of their refinement processes eventually converged on the same evolutionary group, showing proof of the ability of the algorithm to produce results that correlate well with evolutionary entities.

After resolving overlaps we obtained stable HMMs for 86 MDR families, providing coverage of just over 76% of the MDR superfamily. We also found that these families correlated very well with our more empirically chosen family definitions from paper III, lending further credence to the soundness of the algorithm. Table 3.3 shows the growth of the MDR families between papers III and IV.

Two out of the 86 stable HMMs retained too few sequences for continued
refinement after the first inclusion control step, comprising a total of 42 seed sequences. These HMMs are stable, and were thus still kept (but explicitly annotated as potentially less reliable). Past the first inclusion control step, it was possible to classify all remaining HMMs as both stable and reliable.

34 of the refinement processes stabilised on distinct subgroups to 14 of these 86 families, and these were retained separately because of their potential utility for subclassification.

As can be seen in Fig. 3.14, just over half of the families found were purely prokaryotic, while just under half of the families had at least one eukaryotic representative. None of the families were found to be purely archaeal.

We assigned a numbering scheme to the MDR families from MDR001 to MDR086, where the families are enumerated by decreasing family size as found in present investigations, starting with families having human representatives, followed by families with eukaryotic representatives, and ending with the purely prokaryotic families.

Fig. 3.15 shows a dendrogram based on a MAFFT L-INS-i [45] alignment of one representative sequence from each MDR family, clearly showing that the families have little clustering among themselves, but are almost equidistantly related. A bar plot of the size distribution among the families (Fig. 3.16) showed that the human families are on average the largest and the purely prokaryotic families are on average the smallest. Looking at the species distribution among the families (Fig. 3.17) we find nine families with purely plant members and one with purely members from the slime mold Dictyostelium discoideum.

Most MDRs bind nicotinamide adenine dinucleotide (NAD) or NAD phosphate (NADP) as a cofactor, and can also bind a structural and a catalytic Zn\(^2+\) cofactor. We mapped the MDR families to their most similar structure from PDB in order to investigate their cofactor specificities, as determined from conserved putative ligands in tertiary structure.

We found that 44 of the families seem to bind NAD and are thus likely to
Figure 3.15: Dendrogram of the MDR families. Almost all families that bind NAD and 2 Zn$^{2+}$ are found within the yellow frame, while families that bind NADP and 0 Zn$^{2+}$ generally are found within the purple frame. Exceptions are indicated within each frame with opposing colour for NAD(P) preference and a number of filled or empty bullet symbols for Zn$^{2+}$ preference, where half-filled bullets indicate that the ligands for one of the Zn$^{2+}$ cofactors are conserved only among part of the member sequences. The lines are coloured according to species representation, where families with human representatives are blue, eukaryotic green, and purely prokaryotic black.
CHAPTER 3. CURRENT INVESTIGATIONS

Figure 3.16: Size distribution among MDR families, ordered by MDR family number. The number of sequences varies from 20 to 2217 with a median of 42, and the average is around 137 sequences.

Figure 3.17: Species distribution in MDR families, shown using a gradient from white to dark blue (0 % to 100 % of the family) and ordered by MDR family number. The groups are the prokaryotes A=archaea and B=bacteria, followed by the eukaryotic NCBI divisions Pla=plants, Inv=invertebrates, Ver=vertebrates, Mam=mammals, Rod=rodents, Pri=primates.
be dehydrogenases, while 40 seemingly bind NADP and thus are likely to be reductases. In two of the families, different members seemingly have different NAD(P) specificity. Furthermore, we found 35 MDR families that appear to have two zinc and 38 that seem to have no zinc ligands. Of the 7 families that have only one zinc ligand, two seem to have conserved ligands for the structural rather than catalytic zinc, which is highly unusual and an unprecedented find. Six of the families have conserved catalytic zinc ligands but seem to partially lack the structural zinc cofactor.

Also, in Bacteria MDRs without Zn\(^{2+}\) are more frequent than those with Zn\(^{2+}\), while the opposite is true for eukaryotic MDRs, indicating that Zn\(^{2+}\) has been recruited into the MDR superfamily after the initial life kingdom separations.

Furthermore, we found a clear and novel correlation between Zn\(^{2+}\) content and NAD(P) preference, which is clearly visible in Fig. 3.15. Of the 35 MDR families with 2 Zn\(^{2+}\), 30 were found to prefer NAD as cofactor, thereby generally acting as dehydrogenases. Similarly, of the 38 families with no Zn\(^{2+}\), 34 exhibited preference for NADP and would consequently largely be reductases. As exceptions from these general patterns we found four families that appeared to bind NADP despite having both the structural and catalytic Zn\(^{2+}\) cofactor, and four families that appeared to prefer NAD despite lacking the structural and catalytic Zn\(^{2+}\) cofactors.

We also investigated conserved net charges in the 86 families, as this may provide an indication for special binding properties or interaction with strongly charged substrates. The families found to have the largest conserved net charges are listed in Table 3.4. We also noted that in nearly all of these families the conserved charges were even further enhanced by the nonconserved charges, and even in the two exceptional cases, the nonconserved charges did not fully balance the conserved ones. We therefore suggested that MDR037 was likely to have negatively charged substrates or interaction partners, while the others likely would have have positive ones.

We also compared the number of residues in the two domains that attained CScores over 95 % (cf. paper II) and found 14 MDR families with more than twice as many conserved residues in one domain compared to the other. 10 of these had twice as many conserved residues in the catalytic domain as in the coenzyme

<table>
<thead>
<tr>
<th>Family</th>
<th>Conserved</th>
<th>Nonconserved</th>
<th>(\Sigma)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDR037</td>
<td>+11</td>
<td>-2.3</td>
<td>+8.7</td>
</tr>
<tr>
<td>MDR038</td>
<td>-5</td>
<td>+1.4</td>
<td>-3.6</td>
</tr>
<tr>
<td>MDR086</td>
<td>-5</td>
<td>-6</td>
<td>-11.0</td>
</tr>
<tr>
<td>MDR033</td>
<td>-5</td>
<td>-4.3</td>
<td>-9.3</td>
</tr>
<tr>
<td>MDR070</td>
<td>-6</td>
<td>-5.1</td>
<td>-11.1</td>
</tr>
<tr>
<td>MDR054</td>
<td>-6</td>
<td>-1.2</td>
<td>-7.2</td>
</tr>
<tr>
<td>MDR041</td>
<td>-7</td>
<td>-1.6</td>
<td>-8.6</td>
</tr>
</tbody>
</table>

Table 3.4: MDR families with large conserved net charges, listed with their nonconserved charges and resulting average charge (not including Zn ligand glutamate).
CHAPTER 3. CURRENT INVESTIGATIONS

binding domain, while the opposite was true for 4 of them, and we suggested that these differences might reflect differences in functional and/or structural properties.

Finally, we validated the consistency of our HMMs by comparing annotations with 118 novel Swissprot MDR sequences, whose release dates postdated our training databases. Our HMMs detected all of these new sequences except 14, and none of the matching sequences had annotations that conflicted with our previous observations on their respective families. Out of the 14 sequences that were not detected, 12 were annotated as probable polyketide synthases of *Dictyostelium discoideum*, and would perhaps have been expected to match MDR030 (dFAS). The remaining two had annotations relating the proteins to ADH-like enzymes, but nothing more functionally specific. Since our models professedly do not provide 100% coverage of the MDR superfamily, it should be expected that some new MDR sequences will not be classifiable through our current library of HMMs. However as more sequence data on MDRs is amassed, further HMMs may be developed using our algorithm, improving the library and its coverage of the MDR superfamily even further.

In order to make our results directly useful also for scientists outside the bioinformatics area, we developed a web site to present our findings in an easily accessible manner, enabling users to search and display detailed information about the MDR families, using textual and numeric queries against many of their characterised properties. It also permits online sequence scanning against our library of HMMs, making it possible to submit a novel MDR sequence and have it instantly analysed and classified.

The RefineHMM algorithm as well as the database files underlying the web-site have been made freely downloadable, and the website can be found at http://mdr-enzymes.org.
3.6 Paper V – MSA visualisation

In this paper I present MSAView, my main tool for working with and visualising multiple sequence alignments (MSA). I also present the two new conservation measures sequence conformance and residue divergence, which are extensions to the CScore algorithm presented in paper II.

In papers I–IV, analysis of multiple sequence alignments played a central part. Most often using computer-assisted numerical methods, but also frequently by simply looking at the aligned sequences with the naked eye. Fast as computers are at quantifying correlations you know to look for, they are simply no match for a human brain (yet!) when it comes to identifying correlations you did not know to look for.

In many circumstances, it quickly became apparent to us that the existing tools for the job were unsuitable for our purposes; either by not being quite capable of answering our specific questions, or not showing the information in an accessible way, or by simply being intractable with the amounts of data at hand. The solution was often to complement the existing tools by writing new, small, one-off programs for solving those specific problems. Thus, over the years, I amassed a growing and disparate library of small programs for slicing, aggregating and visualising data from MSAs in various ways, all to drag those elusive new correlations and undiscovered sequence features out into the limelight. In time, most of these programs found reuse in new combinations, and each new adaptation generally required some manual tweaking of the code to suit the new purpose or the new types of data.

Eventually, I developed MSAView as a framework to integrate all these small units of functionality, to make recombinations and reuse trivial.

On top of this, I added a layer of visualisation, designed to be fast, flexible and highly configurable, to allow for convenient ocular exploration of the data. Ideally, visualisation should simultaneously be comprehensive and detailed and never distract with irrelevant information. It needs to offer natural and responsive ways of exploring the data, as well as provide customisable and consistent views in order to highlight properties currently of interest and to facilitate comparisons between different data sets.

MSAView offers this, and more. It is a platform independent, extensible and modular toolkit for analysing and visualising MSAs. It has a plugin architecture and a flexible preset library which offers highly customisable data flows and modes of presentation. It offers data integration from online resources and also provides a graphical user interface (GUI) that remains responsive even for large data sets.

The core ideals in MSAView are modularity and flexibility, which makes the program highly adaptable to different needs in different projects. To achieve this, the program constructs its data flows using self-organising, reactive building blocks which can self-assemble into functioning component trees, as shown in Fig. 3.18. This arrangement also helps to increase the speed of the program, since when only the necessary components are placed in a data flow, unnecessary computation is avoided. Additional components (for analysis, visualisation, etc...
CHAPTER 3. CURRENT INVESTIGATIONS

Figure 3.18: A component tree, created by integrating two components to the root; 1) a residue divergence renderer, and 2) a residue letter renderer. Most of the tree is created in step 1 (components labelled 1.1 – 1.5). Step 2 only adds a single component to the tree, since all prerequisites are already in place.

can be integrated at any time, and they will automatically find their proper place in the hierarchy, automatically integrating any necessary adaptor components. The integrated components can then react to new data and produce results, which may then in turn be used as new input for other components. The trees and components are of course highly configurable, and any configuration can be saved as a preset and reused. Presets can also be exported and shared among users.

MSAView can visualise properties for the sequences and the columns in the MSA as well as for the MSA itself. It can also display multiple types of information simultaneously in one view. This is done using layered renderers, each adding specific new information to the resulting image, optionally using alpha blending for translucence effects or fading less pertinent data to make it less obtrusive. MSAs can also be rendered to any size, and will be rendered differently for different zoom scales, suppressing details that would anyway be illegible at a given scale.

The MSAView GUI is designed to be minimalistic, unintrusive and configurable. The only static parts are the top menu bar, the right and lower scroll bar, and the status bar at the bottom, which shows basic information on what is currently under the mouse cursor. The exact appearance of the GUI of course depends on the user’s current configuration, but a typical layout is shown in Fig. 3.19. The GUI is fast and responsive, which is an absolute necessity for GUIs, or people will not feel compelled to use them. Slow and clunky GUIs will kill the user’s curiosity, which of course should be anathema to scientific software. MSAView takes a number of steps to ensure that the GUI remains responsive even in the face of heavy computation:

1. Unnecessary computation is avoided by placing only the necessary analysis...
Figure 3.19: The MSAView GUI is designed to be minimalistic, unintrusive and configurable. The standard menus contain only basic commands for file handling, selection and zooming, while most of the functionality is made available through context sensitive menus. The status bar at the very bottom shows basic information on what is currently under the mouse cursor; alignment coordinates, sequence ID, sequence position and residue letter with surrounding tripeptides. This particular view shows conservation in the full alignment for one of the Pfam MDR superfamily models (ADH_N, Pfam:PF08240) at the survey zoom level (all 16667 sequences and 582 positions in view). Since the sequence identifiers are obviously too numerous to render legibly at this resolution, MSAView instead presents them using a grey shading of equal length.
CHAPTER 3. CURRENT INVESTIGATIONS

components in the data flow.

2. Key parts have optimised C implementations (with Python fallbacks to ensure full platform independence).

3. Time consuming calculations are performed in the background, and the views are continually updated as results arrive.

4. Static parts of the MSA image are rendered on demand as tiled fragments, meaning only fragments that are actually displayed are ever rendered.

5. Rendered fragments are saved in a cache for reuse when the same portion of the MSA again comes into view, keeping time consuming rendering to a minimum.

6. Cache use is maximised by maximising the reuse of zoom scales, limiting free zoom in a way implemented to be virtually unnoticeable to the user.

7. Slow renderers are deferred until there are free processor cycles available, showing partially rendered fragments in the meantime. This allows users to keep their orientation and ensures that the GUI remains responsive even when browsing previously unexplored portions of an MSA, which is especially useful during free zoom.

Apart from the GUI, MSAView also has a powerful command line interface that exposes all functionality from the GUI. It is also possible to import MSAView as a standard package into any Python program and access all the features programmatically.

Most of the functionality in MSAView is presented to the user in the form of actions. These are small, atomary pieces of code that operate on existing component trees, and they are never further away from the user than a right-click in the GUI. The MSAView action framework also handles documentation as well as parameter values and defaults for the actions, and can show all relevant information to the user on demand, either on the command line or using tooltips in the GUI.

The program is designed to be easy to extend, and has a plugin architecture for this purpose. The standard installation has a number of plugins, which extend the functionality by providing additional components, actions and presets. Actions are especially easy to develop, and are therefore the most straightforward way of adding features to the program.

MSAView can import data from external or online sources, such as sequence feature annotations from UniProtKB [76], Pfam [32] or HMMER hmmsearch [70] (as shown in Fig. 2.4 on page 30), genome localisations from Ensembl [27] and predictions for natively disordered regions from DISOPRED2 [4], and it can download and display Pfam alignments. It can also launch external viewers to show further details, like PyMOL [115] for structures or web interface views for database entries (UniProtKB or PDB [3]). It supports several alignment formats (fasta, Clustal [44], NEXUS [116], etc...) and produce raster or vector graphics (PNG, PDF, SVG). It can
3.6. PAPER V – MSA VISUALISATION

Figure 3.20: The MSAView CScore layout, showing conservation in one part of a large MSA. The bottom bar plot shows the positional CScores using a colour gradient from red to yellow, and indicating strictly conserved positions in green, while the bar plot to the left shows sequence conformances in percent, also using a gradient from red to yellow. The MSA view highlights large alignment divergences using a colour gradient from white to red, which indicates notable discrepancies from the conserved norm. This view makes it trivial to quickly identify atypical sequence features, such as the poorly-conserved insertions in sequences 33 and 39, or the positively charged arginine at the otherwise hydrophobic position 182. In this case, MSAView has also coloured the sequence identifiers depending on their database origin, showing all UniProtKB identifiers in red, except those from the high-quality Swissprot subdivision, which are shown in blue.

Also calculate and display residue metrics like Kyte-Doolittle hydropathy [117] or Atchley indexes [2]. The program uses PyGTK [118] for event handling and GUI elements, NumPy [119] for calculations and pycairo [120] for rendering graphics.

The CScore plugin in the standard installation of MSAView computes and visualises CScores, which were presented in paper II. CScores are computed for positions in the MSA, and can help scientists to quickly find putative functional motifs and residues. This paper also presents two new conservation measures that are extensions of the CScore algorithm, namely alignment divergences and sequence conformances. These two facilitate finding atypical residues or deletions, and sequences that differ from their siblings at crucial positions.

Alignment divergences are calculated for each symbol in the MSA and are a means to detect unusual irregularities. The alignment divergences take values close to zero for typical residues at well-conserved MSA positions, but take values close to 100% for residues in nonconserved insertions and for atypical substitutions or deletions at otherwise well-conserved positions. This is exemplified in Fig. 3.20 where divergences close to 100% are coloured red, and are immediately identifiable.

Sequence conformance on the other hand is a measure of how similar a sequence is to the others in the set, where “typical” sequences get values close to 100%, while lower values indicate sequences with many nonconserved insertions and/or deletions or unusual substitutions at otherwise well-conserved positions.
3.7 Conclusions

This section briefly presents some of the major conclusions from the projects included in this thesis.

We developed an algorithm for automated protein family definition (RefineHMM, paper IV);

- It generates stable and reliable HMMs from relations found in data.
- It does not rely on user input, apart from an initial HMM.
- It is well suited for burst computing on shared or distributed resources.
- The HMMs correspond to evolutionary entities, and correlate well with HMMs from manual and empirical efforts (e.g. from paper III).

We developed an algorithm for measuring conservation (CScore, papers II and V);

- Positional CScores detect conserved residues.
- Alignment divergences detect unusual insertions and atypical residues or deletions at crucial positions.
- Sequence conformances detect sequences that differ in crucial positions.

We developed a framework for working with MSAs (MSAView, paper V);

- It is fast, flexible, platform independent, extensible and modular.
- It has a plugin architecture and a user extendable preset library.
- It offers data integration from online resources, and highly customisable data flows and modes of presentation.
- It can be used either through a fast and responsive GUI or a powerful CLI, or it can be imported as a standard package in any Python program.

We developed a website that makes our findings on the MDR superfamily directly useful for the greater community (paper IV);

- It has a point-and-click interface that is easy to use.
- Users can search and view families and instantly classify novel sequences.
- All results, data and source code from the project can be freely downloaded.
3.7. CONCLUSIONS

We investigated the inorganic pyrophosphatase family (H\(^{+}\)-PPases, paper I);

- Most or all functionality is located in transmembrane or cytosolic segments.
- Their characteristic pyrophosphate binding primitive nonapeptide motifs are unique to this family.
- Patterns based on these motifs can be used in functional classification of uncharacterised sequences.

We investigated the BRICHOS superfamily (paper II);

- The BRICHOS superfamily contains three previously unknown families;
- Group A, found in insect and worm, may be ancestral to ITM2.
- Group B, found in mouse, rat, cow and dolphin, may be third Gastrokine.
- Group C, present in tetrapods and fish, is little studied, and may be a novel and truly disjoint BRICHOS family.
- Group C has a functionally yet uncharacterised C-terminal segment that is nearly strictly conserved throughout evolution from fish to man, and which is unique to this family.

We investigated the MDR superfamily (papers III–IV);

- There are around 500 MDR families, with at least 25 genes in the human and at least 11 activity types in higher vertebrates.
- We characterised and developed HMMs for 86 MDR families, covering 76% of the superfamily.
- These HMMs can be used as basis for a stringent MDR nomenclature scheme.
- Zn and NAD(P) cofactor preference is correlated in MDRs;
  - Presence of both Zn\(^{2+}\) cofactors is correlated with NAD preference.
  - Zn\(^{2+}\) absence is correlated with NADP preference.
- MDR Zn\(^{2+}\) recruitment likely occurred after the life kingdom separations.
CHAPTER 3. CURRENT INVESTIGATIONS
Chapter 4

Future challenges

A scientist’s work is never done, as each answer begs two more questions. The projects in my thesis are by no means closed chapters, and I have also been involved in several projects outside the scope of this thesis, and all beckon with opportunities for new discoveries. In this chapter I will briefly outline two of the topics that I hope to explore in a continued career in science; an automated protein family discovery and annotation machinery, and an effort to bring bioinformatic supercomputing right into granny’s iPhone.

4.1 Automated protein family discovery

For me, the MDR superfamily is still an open field with many opportunities still left to explore. In paper IV we made HMMs for the 86 largest families, and characterised them based on several properties such as species distribution, conservation, zinc content and cofactor presence. We also made a web site (http://mdr–enzymes.org) to make these findings easily accessible for the greater MDR community.

One challenge will be to see to it that our stable and reliable HMMs stay reliable in the face of new discoveries, that is, to handle the arrival of new data in a consistent way. Most new sequences will likely fall into one of the larger, already defined families, which already contain sufficiently many members to accurately describe the variabilities within the family, and thus will not require much further attention. However, some new sequences will match small families and may impart a large shift on their composition. A basic tenet in science is to always keep an open mind, and adapt your world view when faced with new observations. So in this case, in the light of new data, it may become necessary to modify some of the models to adjust for novel marginal cases. Or it may become necessary to divide or merge some of the families, if that is what the new data dictates. Yet other sequences will match none of the known families, and as more of these sequences arrive, it may become possible to create new HMMs for groups that are presently too small, and thus improve the coverage of our HMM library.
As we estimated the total number of MDR families to be around 500, I foresee ample work opportunities for RefineHMM in the future.

Another challenge is to keep the family annotations updated, so that they are consistent with all new and updated findings.

Yet another challenge is to incorporate further sources of information, for example gene expression patterns, proteotypic peptides, genomic localisation and variability, and disease correlation, just to name a few.

The grand challenge however is to automate all of the above. What I would like to do is to set up a characterisation pipeline that scans new releases of the source databases for new members of the known families, clusters the rest and builds HMMs for the clusters that are of sufficient size, and finally annotates the remainder with what known family they are most similar to. Then, the characterisation machinery should scour the databases for new annotations for the family members, and present all common denominators as well as peculiarities at the family level, preferably along with some kind of statistical reliability measure.

A more personal challenge is that I would like for http://mdr-enzymes.org to become a central meeting place and a common reference for scientists in the MDR field, but to win the appeal of the community, it needs to be at once comprehensive, detailed, accurate and up to date. That’s a challenge, but if the above points can be addressed, I believe it can be done.

### 4.2 Human-friendly supercomputing

More than once in this thesis I have touched upon the subject of supercomputing and distributed calculations, and at this point I would like to broach the subject of why we need it, how it is currently done, and how we could make it better (these are all challenges).

As laboratory methods develop and move toward higher throughput and more detailed reports, and as data accessibility and interoperability increase, the scales of bioinformatic projects are growing at a rapid pace. As a result, the possibilities for new discoveries increase exponentially, but at the same time we need to find new ways of riding the data wave, lest we hit the break and drown in it.

One way of getting on top of the game is to buy bigger computers on a subscription basis, but staying on the bleeding edge of performance this way will quickly become painful financially. Ironically, the need for computation in science is also often periodic, with peaks during experimental analysis and valleys during paper writing and reviewing, so these fancy new machines will not actually be used very much, when averaged over time. It therefore makes sense for groups to collaborate and build systems where they share the available time. Not only will this distribute the use more evenly, but pooling resources will also afford larger and faster machines. And not only will this make everyone’s calculations go faster, it also puts a lot of computational power directly available if need should suddenly arise, for example when a new database is released.\(^1\) Additionally, it’s easier to

\(^1\)the popular term is “burst computing”, or “outburst computing”, which is an euphemism for “oh s#!&, is that the deadline?!?".
4.2. HUMAN-FRIENDLY SUPERCOMPUTING

Figure 4.1: What we’ve got; the tools of the trade. Large computers still fill a room (upper pictures). They can use about as much power as a residential area (upper left), contain network cable bundles as thick as a horse’s thigh (bottom pictures) and enough node interconnecting switches (centre picture, left and right) to warrant several additional interconnecting switches just to keep track of interconnecting switches (centre of centre picture). All of these are things that the average scientist will not want to know about.

To put into perspective what I mean by a large computer, see Fig. 4.1, which features Neolith, a large computer cluster at the National Supercomputer Centre in Linköping Sweden (NSC), which was born as the 23rd fastest computer in the world with its 60 trillion floating point operations per second, distributed over 6440 processor cores in 805 full bisection InfiniBand interconnected computers.

The drawback is that larger computers are often harder to use. Also with a shared system, there are other people to consider, and all will have to play nice together. It can all be a bit intimidating for a beginner in the field, and there are often many new things take in and learn, such as job scripting, scheduling, time and storage quotas, software availability and compilation, account security and so on. And every new system will exhibit just the right amount of subtle differences in critical places to make a jolly mess of it all.

So, all scientists want more compute power, and if they’re lucky, they’ll get a piece of what’s in Fig. 4.1, but what all scientists secretly yearn for (exactly!) is shown in Fig. 4.2. The challenge, of course, is to provide this.

During my time as a Ph.D. student, I became involved in the Nordic DataGrid Facility (NDGF) Biogrid project [113, 121, 122], which had the specific purpose of
NDGF was set up as a collaboratory effort in the Nordic countries to store and process data from the Large Hadron Collider (LHC) at the European Organisation for Nuclear Research (CERN) [123, 124], and was created (by design!) as a distributed computational grid, with several large computers set up in several different countries,² all connected together and working together.

The idea was to have a system that abstracts away as much as possible of the messy intricacies that the user quite frankly does not want to be concerned with; a system where the user can just supply a list of requirements for the job and trust the machines to work out the details. To solve this, everyone on the grid, users and systems alike, would get personal certificates for identification, and then special global information systems would keep track of who was allowed on which system. Then, when someone would want to do computation, they would send a detailed job description to one of the resource broker systems, including all the requirements like number of processors, amount of memory and pre-installed user applications. The resource broker would then match the job description against all available compute resources and then direct the job to one of the systems with enough brain power and capabilities to do the job.³

Needless to say, this design adds another magnitude of complexity to just connecting together a bunch of computers in a room (which by itself is by no means trivial, cf. Fig. 4.1), and it could perhaps initially seem like a quantum

²including Russia and Australia
³much like advertising an open position at an employment agency, really.

Figure 4.2: What we want; insert data in one tamper proof end and fetch results as they come out the other, and never mind the messy particularities in between.
leap in the wrong direction. One could perhaps also expect that such a design could easily implode under the massive weight of its own complexity and form a black hole, from which no data could ever escape, but as it turns out and possibly against all reason, it works out really well.

A large part of the explanation is that in such a project, rigorous harmonisation and standardisation become paramount in order to get anywhere. Incidentally, this is also exactly what you’d need in the end using any other approach, only in that case you wouldn’t discover where it’s needed until things catch on fire well into production.

Also, once you are standardised and harmonised, distribution suddenly becomes a great strength, because it becomes synonymous with “multiple redundancy”, which means that if one system breaks, for any reason, there are plenty others available to step up and pick up the slack. By comparison, if you had bought only one giant system, and it breaks for any of the same reasons, you’re effectively stranded for the duration. These harmonisation efforts also extend to how pre-installed user applications should be configured, effectively packaging entire runtime environments (RE) with every setting from paths and environment variables to program versions and convenience helper scripts explicitly defined, so that the user can just specify the necessary runtime environments in the job descriptions, and then expect everything to “work as usual”.

Now on to Biogrid’s part in this. For a start, Biogrid has already packaged a number of commonly used bioinformatic applications into grid REs, including HMMER [70], BLAST [39], MAFFT [45], ClustalW [44] and MUSCLE [49]. Most often, bioinformatic projects also depend heavily on databases and database searches, and since consistency is paramount, one generally defines what database versions to use at the start of the project (generally, the freshest available at the time), letting those be canon throughout the project. Since projects well can outlive the availability of the selected database versions, and to enable validation of the results long past their respective projects’ lifetimes, Biogrid has started to provide stable access to bioinformatic databases on the grid’s distributed and cached storage network [125]. Currently, the protein sequence databases UniProtKB and UniRef [76] are available, and these are automatically indexed and uploaded for every new subrelease. Additionally, the database files are usually the only large files that are needed for bioinformatic jobs, and as noted in paper IV, these are constant and can be cached on the compute resource, and Fig. 4.3 explains the details behind why this is a good thing. However, the punchline is that Biogrid provides nigh immediate access for grid jobs to all database versions ever stored on the grid.

One challenge is to expand the current focus in Biogrid from protein sequence analysis to other areas, such as molecular dynamics [126] and phylogenetics [7,54], and to provide additional types of databases, such as nucleotides (e.g. EMBL-Bank [21]), complete genomes (and translations, e.g. Ensembl [27,30]), and functional classifier databases such as Pfam, PROSITE and InterPro [32–34].

Another, perhaps greater challenge, lies at the user end of the spectrum. The purpose of grid technology is to make computation trivial, so that in the future,
Figure 4.3: Benefits of cached database storage. Three bioinformatic jobs (A, B and C) use the same large database files as input from the distributed and cached grid storage (dCache). When job A is submitted to the grid, it is assigned to a compute resource, where it is typically placed in queue to wait nicely for its turn to start, however the resource will immediately ask dCache where to get the files (arrows 1–2), and download them (compressed) directly from the corresponding storage element (arrow 3) and cache them locally. This is a slow step, as it involves transferring large files over the Internet. So, when job A (and B, C, D, E, F...) starts, all necessary files are already in place, and can be transferred to the compute nodes over typically very fast interconnect network (arrows 4–6), uncompressing the files on the fly (which is faster than the transfer speed and therefore free). This will only add a few seconds to the typically hours long runtime, and will generally ensure that the database files will exist in system file cache on the compute nodes, enabling lightning fast access to the data. When the resource no longer receives any more jobs that require those same files, they will eventually be out-competed from the cache by other required files, freeing cache space for other uses.
4.2. HUMAN-FRIENDLY SUPERCOMPUTING

Computation should be just another facility that you can subscribe to, like electricity or tap water (cf. Fig. 4.2). To leverage a popular British form of comedy, one could say that we are not quite there yet. There are still many elements that you must master in order to be able to run jobs on the grid; writing a job script that runs and does not throw away your results, estimating total runtime and memory requirements, keeping tabs on all data files and their corresponding result files, figuring out what REs you need, and so on and so forth. On top of this you also have to learn how to write grid job descriptions, which for some reason must be written in a cryptic prefix notation called XRSL (which Oxana Smirnova has done a commendable job of summarising in [127]). In its defence, XRSL is something you would write only once for every type of job, as a template which you would then just generate more instances of for similar jobs, but still, you will have to write it that once (and be computer savvy enough to know how to template it and generate further instances).

What one could do is write helper applications for common tasks, which would figure out decent defaults and save user preferences. For example in sequence homology search, you could have a large file of sequences for which you would need to analyse homologues. One helper application could partition the queries into sufficiently small batches for the jobs to finish within the user’s estimated attention span, ask which database to use (with probable default), generate blast grid job descriptions, submit and track the jobs, and download the results to an appropriate directory when all jobs have finished. Another helper application could perform common tasks on the resulting blast result files, for example; determine the 10 most frequently found homologues, find query sequences that have radically different top 100 homologue patterns, calculate length distribution among homologues, list all non-partial homologous sequences with E-values under a given threshold, and so on.

A more generalised example would be a helper application which you could tell to run a certain command line on the grid, as a certain job type from a configurable portfolio of job types.

There are many conceivable applications, and one challenge is to implement them. Another is to find out which of them would be useful, and how they should behave in order for users to feel compelled to use them. Many of these could probably be also wrapped up in a point-and-click interface, a sort-of grid dashboard, which would probably be appreciated by many users, but a challenge to implement in a consistent, error-proof and attractive way.

So, there are clearly many challenges still in our way before we get to Fig. 4.2. But we’re getting there!

...and analysing them, all the billions and trillions of result files...
Bibliography


[38] *Online Mendelian Inheritance in Man, OMIM (TM)*. McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins University (Baltimore, MD) and National Center for Biotechnology Information, National Library of Medicine (Bethesda, MD), 2010-11-02. World Wide Web URL: http://www.ncbi.nlm.nih.gov/omim/.


